UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Diseño y desarrollo de una aplicación para creación de contenido docente de realidad aumentada basada en marcas

Trabajo Fin de Grado

Grado en Tecnología Digital y Multimedia

AUTOR/A: Planelles Lorente, Ramiro

Tutor/a: Rey Solaz, Beatriz

Cotutor/a: Monzó Ferrer, José María

CURSO ACADÉMICO: 2022/2023

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE UPV INGENIEROS DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Diseño y desarrollo de una aplicación para creación de contenido docente de realidad aumentada basada en marcas

Trabajo Fin de Grado

Grado en Tecnología Digital y Multimedia

AUTOR/A: Planelles Lorente, Ramiro

Tutor/a: Rey Solaz, Beatriz

Cotutor/a: Monzó Ferrer, José María

CURSO ACADÉMICO:2023-2024

RESUMEN

La Realidad Aumentada (AR) es una innovadora tecnología que tiene un enorme potencial para enriquecer y mejorar la docencia, así como para ofrecer nuevas experiencias educativas más atractivas y eficaces, ya que consigue que los estudiantes interactúen y participen de forma más activa durante su aprendizaje.

Es por ello, que este proyecto se marca como objetivo general el desarrollo de un proyecto base para generar una aplicación de Realidad Aumentada basada en marcas para dispositivos móviles con principal utilidad en la docencia, contribuyendo así con el cuarto Objetivo de Desarrollo Sostenible (ODS), "Educación de Calidad".

Para lograr este objetivo, se ha empleado el motor de desarrollo de aplicaciones Unity. Este programa contiene la actual librería *ARFoundation* que permite implementar y hacer uso de numerosas funcionalidades de Realidad Aumentada.

Por ende, en el presente trabajo de final de grado se explica detalladamente el proceso de desarrollo del proyecto, donde se configuran y asocian los marcadores a objetos 3D, imágenes y vídeos, así como la generación de una aplicación que es capaz de instanciar dichos recursos digitales tras la detección de los marcadores. Todo esto, ofreciendo un entorno sencillo y accesible para el uso del proyecto por parte de los y las docentes en base a sus preferencias.

RESUM

La Realitat Augmentada (AR) és una innovadora tecnologia que té un enorme potencial per a enriquir i millorar la docència, així como per a oferir noves experiències educatives més atractives y eficaces, ja que aconsegueix que els estudiants interactuen i participen de forma més activa durant el seu aprenentatge.

És per això, que aquest projecte es marca com a objectiu general el desenvolupament d´un projecte base per a generar una aplicació de Realitat Augmentada basada en marques per a dispositius mòbils amb principal utilitat en la docencia, contribuint així amb el quart Objectiu de Desenvolupament Sostenible (ODS), “Educació de Qualitat”.

Per a aconseguir aquest objectiu s´ha emprat el motor de desenvolupament d´aplicacions Unity. Aquest programa conté l´actual llibreria *ARFoundation* que permet implementar i fer ús de nombroses funcionalitats de Realitat Augmentada.

Per tant, en el present treball de final de grau s’explica detalladament el procés de desenvolupamente del projecte, on es configuren i associen els marcadors a objectes 3D, imatges i vídeos. Així com la generació d’una aplicació que és capaç de instanciar aquests recursos digital després de la detecció dels marcadores. Tot això, oferint un entorn senzill y accessible per a l’ús del projecte per part dels i les docents sobre la base de les seues preferències.

ABSTRACT

Augmented Reality (AR) is an innovative technology that has enormous potential to enrich and enhance education, as well as to provide new, more attractive and effective educational experiences. It enables students to interact and participate more actively in their learning.

For this reason, this project sets it's overall goal as the development of a foundational project to generate an Augmented Reality application based on markers for mobile devices, with the main use in teaching. This contribution aligns with the fourth Sustainable Developmet Goal (SDG), "Quality Education".

To achieve this objective, the Unity application development engine has been used. This program incorporates the current *ARFoundation* library, which allows for the implementation and use of numerous Augmented Reality functionalities.

Therefore, in this final degree project, the development process is thoroughly explained. It includes the configuration and association of markers with 3D objects, images, and videos. Additionally, it encompasses the creation of an application that is able to instantiate these digital resources after the detection of the markers. All of this offering a simple and accesible environment designed for the use of educators based on their preferences.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

1.1 Contexto

Actualmente, se puede apreciar como la tecnología ya es uno de los recursos más llamativos e influyentes en el proceso de aprendizaje en cualquier ámbito de la docencia. Es más, los avances tecnológicos no solo están ayudando a la forma de impartir enseñanza, sino que está consiguiendo cambiar la manera en la que los alumnos y alumnas adquieren la información y aprenden.

La famosa cita de Albert Einstein: *"Temo por el día en el cual la tecnología sobrepase nuestra interacción humana. El mundo tendrá una generación de idiotas"*, nos recuerda el temor que ha existido y existe hacia el uso de las tecnologías. Sin embargo, es evidente que en la actualidad la tecnología nos brinda numerosas oportunidades para mejorar el aprendizaje, y es que Aristóteles no se equivocaba al describir *téchne* como una forma de conocer el mundo (1).

Cabe destacar que la tecnología como modelo de enseñanza permite transformar el rol pasivo que siempre ha jugado el alumnado, a un papel activo, pues ahora los estudiantes se han convertido en "prosumidores", es decir, son tanto productores como consumidores de su conocimiento al mismo tiempo, cumpliendo así con el paradigma educativo (2).

Una de las herramientas con la que se ha conseguido este cambio en el papel del alumnado ha sido la Realidad Aumentada, una tecnología innovadora con un gran potencial en el ámbito educativo. Y es que con ella, ha sido posible generar en los estudiantes una oportunidad de explorar el entorno que les rodea de una manera diferente, disfrutando de un proceso de aprendizaje interactivo y enriquecedor, ya que es posible combinar imágenes del mundo real con otras virtuales para así crear un entorno lleno de estímulos como objetos 3D, audios, videos, imágenes…

1.2 Justificación y relevancia

Este proyecto se basa en el reconocimiento del impacto de las nuevas tecnologías, en particular la Realidad Aumentada (AR) en el campo educativo. La incorporación de la AR en la docencia crea una enorme cantidad de posibilidades para mejorar el proceso de enseñanza y aprendizaje, brindando experiencias más interactivas, inmersivas y personalizadas para el estudiante.

Al desarrollar una aplicación basada en marcadores con Unity, se pretende sacar partido al potencial de la AR para proporcionar a los profesores una herramienta adaptable y flexible, con la que puedan crear experiencias de aprendizaje enriquecedoras y adaptadas a las necesidades de sus alumnos y alumnas.

Además, con relación a la nueva normalidad educativa y el auge que está generando la docencia no presencial, este proyecto cobra más relevancia. La composición entre la tecnología AR, los marcadores, objetos 3D, imágenes y vídeos dan un sentido innovador y efectivo para fomentar la indagación, el aprendizaje y despertar el interés del estudiante, al mismo tiempo que se impulsa su autonomía y participación en el proceso de aprendizaje (3).

Es importante recalcar que este Trabajo Final de Grado contribuye significativamente a los Objetivos de Desarrollo Sostenible (ODS), concretamente al ODS número 4: “Educación de Calidad”. A través de este proyecto de aplicación, se promueven mejoras dentro del ámbito de la educación garantizando un proceso de aprendizaje novedoso que consiga despertar en los alumnos un interés proactivo en su educación. Este hecho, resalta el compromiso que presenta el presente trabajo para participar en un mundo más justo a través de la docencia (4).

2. OBJETIVOS DEL PROYECTO

Teniendo en cuenta las posibilidades que ofrece la tecnología si se aplica correctamente a la docencia, así como las oportunidades que puede ofrecer la Realidad Aumentada, este proyecto tiene como objetivo general:

- Desarrollar la interfaz de un proyecto base para la generación de una aplicación de Realidad Aumentada basada en marcas para dispositivos móviles con principal uso en la docencia.

Del mismo modo, para alcanzar este objetivo general, se plantean los siguientes objetivos específicos:

- Estudiar, comprender e implementar el marco de trabajo "*ARFoundation*" dentro del entorno de desarrollo Unity.
- Permitir al usuario personalizar la aplicación para el uso de sus propios prototipados.
- Utilizar diferentes recursos de Realidad Aumentada, tanto marcadores como objetos 3D, imágenes y vídeos.
- Generar contenido de apoyo para el docente con el fin de facilitar el aprendizaje y uso del proyecto, así como videotutoriales y guias explicativas.

3. MARCO TEÓRICO

3.1 Realidad Aumentada (AR)

La **Realidad Aumentada** (AR) es una tecnología que permite combinar los elementos físicos del mundo real con elementos virtuales (*Figura 1*). Es decir, es una herramienta con un gran potencial con la que además de interactuar con la información física y tangible, es posible que un usuario obtenga información adicional mediante la virtualidad, estando así, en un contexto real aumentado en el que el usuario alcanza una experiencia inmersiva e interactiva (5). Esta diferenciación entre lo real y lo virtual, queda perfectamente ilustrada en el esquema que propone Milgram mostrado en la *Figura* 1, donde defiende que desde esta perspectiva el mundo de la Realidad Aumentada está más próximo del mundo real que de uno virtual (6).

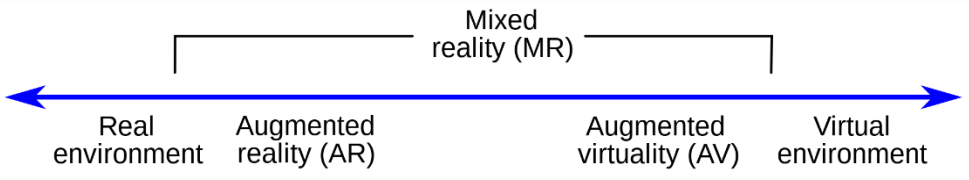


Figura 1: Reality-virtuality continuum de Milgram

Si nos retomamos a los orígenes de esta tecnología, fue en 1992 cuando se comenzó a usar la expresión "Realidad aumentada", concretamente Tom Caudell fue quien dejó grabado el término, y decía que la AR se puede definir claramente en tres ideas: es una combinación de elementos reales y virtuales, es interactiva en tiempo real y está registrada en tres dimensiones (7).

Este proceso de combinación entre el mundo real y el virtual, se registra mediante el uso de dispositivos tecnológicos, como teléfonos inteligentes, *tablets* o incluso propias gafas de AR. En estos, la AR incorpora información virtual como pueden ser objetos 3D, imágenes, vídeos, textos… sobre la información que está ocurriendo y se está viendo a través de la pantalla en tiempo real (5).

Figura 2: Uso Cotidiano de Realidad Aumentada

Principalmente, la Realidad Aumentada se basa en la capacidad de detección y seguimiento de imágenes y objetos identificables, consiguiendo percibir y situar el contenido virtual en relación con el contenido real detectado. De esta forma, los usuarios son capaces de percibir lo virtual como parte del entorno físico, pudiendo actuar, manipular y explorar los objetos virtuales, para así obtener información adicional sobre los elementos físicos e incluso tener nuevas experiencias interactivas (8).

Actualmente, se puede decir que la AR toma protagonismo en nuestras vidas, debido a que es muy común ver muchos ejemplos de su uso que ayudan en las principales labores del ser humano.

Por ejemplo, el uso de la AR en cirugías asistidas permite a los facultativos visualizar información relevante en tiempo real superpuesta sobre el paciente (*Figura 3*). Por otro lado, en cuanto al comercio, algunas marcas ofrecen a los clientes la experiencia de probarse virtualmente sus prendas de vestir para agilizar el proceso de compra (*Figura 4*). Otra de sus muchas utilidades se encuentra en el ámbito de la educación, donde se están implementando cada vez más aplicaciones de AR para que los y las estudiantes tengan experiencias educativas e interactivas visualmente enriquecedoras (9).

Figura 3: Uso Realidad Aumentada en Medicina

Figura 4: Uso Realidad Aumentada en Comercios

3.2 Aplicaciones de Realidad Aumentada (AR) en la educación

Teniendo en cuenta la información anterior, queda demostrado que la Realidad Aumentada (AR) es una tecnología prometedora y que está cada vez más presente en nuestras vidas, y en el ámbito educativo, no se queda atrás.

En la **docencia**, la AR se utiliza como una herramienta innovadora con mucho potencial para mejorar la enseñanza y el aprendizaje en las nuevas modalidades. Con la AR es posible conseguir regenerar la realidad de un aula y ver los objetos desde otro punto de vista para alcanzar un mejor estudio y conocimiento sobre ellos (10).

El principal objetivo de la AR en la educación es que el rol pasivo del estudiante desaparezca y, que por el contrario, se ensalce su participación e interactividad siendo los mismos estudiantes tanto consumidores como productores del conocimiento, lo que genera una mayor inmersión y despierta el interés del alumno en las disciplinas (2).

Del mismo modo, en este ámbito la AR representa la mejor forma de mejorar y facilitar el proceso de enseñanza del alumnado (*Figura 5*). Gracias a la conexión entre la realidad física y el contenido digital se brinda un apoyo significativo a los y las estudiantes a la hora del aprendizaje de los conocimientos (11).

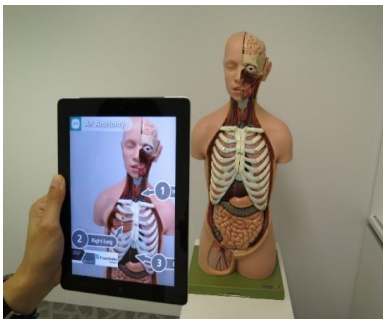


Figura 5: Realidad Aumentada en docencia

Asimismo, se debe tener en cuenta que la AR no solo consta de la visualización de objetos en tiempo real, sino que también admite crear experiencias prácticas y colaborativas, es decir, se puede aplicar tanto a sesiones teóricas como prácticas. Gracias a esto, los y las estudiantes también pueden trabajar con simulaciones interactivas y resolver problemas virtuales, incluso participar en proyectos cooperativos utilizando elementos virtuales compartidos.

Además, la AR tiene la ventaja de mejorar la accesibilidad y la inclusión, ya que al poder utilizar texto con traducciones visuales, imágenes y vídeos, ofrece al alumnado con necesidades especiales un acceso más efectivo a la educación (12).

3.3 Unity

Unity es la plataforma líder del mundo para crear y operar contenido interactivo en tiempo real creada por *Unity Technologies* (*Figura 6)*. Esta plataforma tan popular, disponible tanto para *Microsoft Windows*, *Mac OS* y *Linux*, es un motor de desarrollo de juegos y aplicaciones, donde su software está basado en una serie de rutinas de programación mediante las cuales el usuario puede confeccionar el diseño y el funcionamiento de los juegos y aplicaciones interactivas creadas (13).

Figura 6: Logo Unity

Cabe destacar el proceso de instalación de aplicativos de Unity, ya que en primer lugar, se debe tener instalado *Unity Hub*, que no es más que una aplicación independiente la cual permite gestionar ordenada y centralizadamente todas las versiones y proyectos de Unity. Tras esto es necesario descargar una versión en concreto de Unity e incorporarla a *Unity Hub*. De esta forma, es posible dar comienzo a un nuevo proyecto (13).

Asimismo, es importante mencionar que esta plataforma cuenta con diferentes planes a la hora de adquirirla, lo cual hace posible adaptar las necesidades de cada usuario (*Figura 7)*. Estos planes son: un plan *Personal*, completamente gratuito con el que se puede crear juegos y aplicaciones con facilidad, y por otro lado, tres planes de pago, *Pro*, P*lus* y *Entreprise*, que conceden servicios premium, ofrecen soporte prioritario y descuentos en las tiendas, y permiten generar ingresos con las licencias correspondientes, ya sea a nivel de empresa o particular (14).

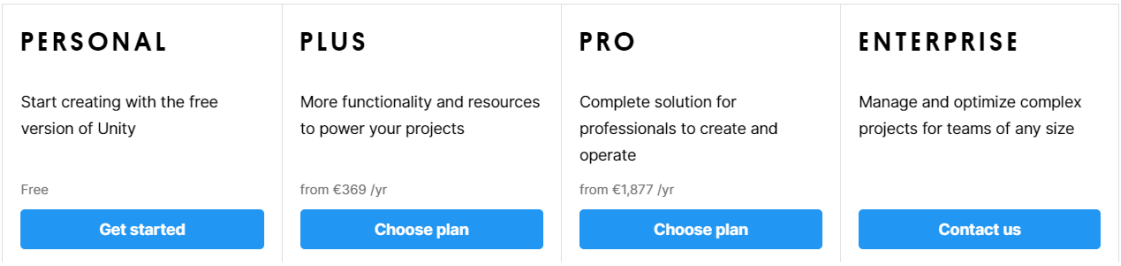


Figura 7: Planes Unity

Además, uno de los puntos más fuertes con los que cuenta Unity, es la gran comunidad de usuarios que presenta, ya que comparten diferentes grupos y foros en los que reciben ayuda mutua a la hora de resolver problemas, compartir experiencias, aportar valoraciones a proyectos, recomendar tutoriales o cursos…(13).

Unity también destaca por su interfaz clara y organizada, lo que la hace intuitiva para los usuarios y es de gran utilidad a la hora de trabajar (*Figura 8)*. Cuenta con la ventana de escena que es donde se construye el juego, ya sea 2D o 3D, colocando y manipulando objetos. La ventana de jerarquía está vinculada a la escena y muestra una lista de los objetos presentes en ella, lo cual facilita su ubicación y su uso. Otra de sus ventanas es la ventana proyecto, esta muestra los recursos disponibles, al igual un explorador de archivos. Por último, el inspector permite visualizar y editar las propiedades de los elementos, donde se añaden las funciones AR (15).

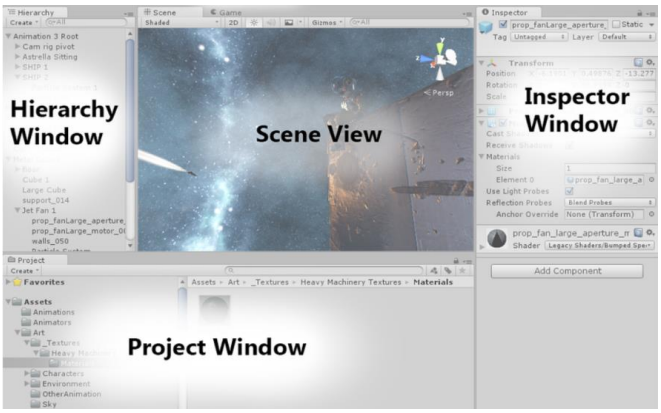


Figura 8: Interfaz Unity

3.3.1 *AR Foundation*

Por otro lado, Unity ha ganado reconocimiento por su versatilidad y capacidad en la creación de experiencias de Realidad Aumentada. Aquí es donde entra en juego la librería *ARFoundation*.

ARFoundation (*Figura 9*) es un marco de trabajo desarrollado por *Unity Technologies* que permite a Unity la creación de aplicaciones de Realidad Aumentada multiplataforma. Con este paquete de trabajo, los usuarios pueden agregar diversas funciones de AR a sus proyectos, lo cual les permite activar componentes virtuales cuando se ejecuta el juego o aplicación y dar paso al mundo virtual. Aparte, cuando se ejecuta la aplicación *ARFoundation* utiliza *ARSDK*, es decir, el *software* de programación nativo de la plataforma, lo que permite crear el juego o aplicación solo una vez y que sea válido para el resto de las plataformas AR del mundo (16).

Figura 9: Logo AR Foundation

Esta librería de Unity tiene una destacable ventaja, la capacidad para trabajar con diferentes tecnologías de AR como *ARKit* de *Apple* o *ARCore* de *Google* (*Figura 10)*. Estas, permiten crear juegos y aplicaciones compatibles con cualquier dispositivo sin la necesidad de crear códigos específicos para cada plataforma (17).

Figura 10: ARKit y ARCore

ARFoundation ofrece una amplia gama de funcionalidades, incluyendo el seguimiento de movimientos, la detección de planos e imágenes y la interacción con objetos físicos entre otros. Los desarrolladores pueden crear juegos y aplicaciones de AR que integren objetos virtuales de manera precisa en el entorno físico, y también pueden implementar reconocimiento de gestos y seguimiento de objetos físicos (17).

En conclusión, *ARFoundation*, combinado con los marcos de trabajo *ARCore* o *ARKit* dependiendo de la plataforma, amplía notablemente las posibilidades de creación de experiencias con Realidad Aumentada. Explotando todas sus funciones como la capacidad de seguimiento y detección de objetos, planos o demás elementos, *ARFoundation* consigue combinar de manera efectiva el mundo real con el virtual y generar experiencias enriquecedoras para los usuarios (15).

4. DESARROLO DE UN PROYECTO CON UNITY

4.1 Requisitos y herramientas necesarias

Para poder comenzar con la creación del proyecto en Unity, se debe tener en cuenta una serie de requisitos que los dispositivos deben cumplir, así como, una serie de herramientas a tener disponibles o instaladas en los mismos.

En cuanto a los requisitos del sistema de Unity que debe cumplir la plataforma de desarrollo, Unity admite sistemas operativos *Windows* 7 SP1 o versiones superiores, *Windows* 8 y Windows 10 únicamente en las versiones de 64 bits; *Mac OS* 10.13 o versiones superiores; *Ubuntu* 16.04, *Ubuntu* 18.04 y *CentOS* 7. También, es importante tener en cuenta que dicha plataforma debe contener una GPU y tarjeta gráfica con capacidades DX10 (shader modelo 4.0), y que es recomendable tener una memoria RAM como mínimo de 4 GB (18).

Asimismo, para la implementación del código del proyecto en *scripts*, es necesario un editor de código como Visual Studio, Visual Studio Code, MonoDevelop… Con la descarga de Unity Hub, es posible incluir el editor Visual Studio si no disponías de ninguno de ellos anteriormente.

Por otro lado, es necesario asegurarse de que el dispositivo de prueba donde se desea instalar la aplicación tenga compatibilidad con el sistema AR Core (*Android*) o AR Kit (iOS), ya que estos son los encargados de permitir todas las funcionalidades de Realidad Aumentada compiladas con el SDK de Unity.
Principalmente, un teléfono móvil *Android* admite el sistema AR Core si originalmente se envía con Google Play Store y si tiene una versión *Android* 7.0 o una posterior; y el iOS admite el sistema AR Kit, si estos ejecutan la versión iOS 11.0 o una posterior (19).

Además, a la hora de compartir la aplicación (archivo .apk) desde la plataforma de desarrollo al dispositivo de prueba, se puede realizar de dos formas diferentes, por lo que es importante considerar una serie de cambios en la configuración de ajustes del teléfono móvil, así como otros factores.

El primer caso es el más sencillo para el usuario final, ya que en un principio solo tiene que realizar el proceso de descarga de la aplicación una única vez. Se trata de generar la aplicación (archivo .apk) en la propia plataforma de desarrollo y después compartirla al dispositivo de prueba, para ello es necesaria una aplicación o web intermediaria como Google Drive, Mega, Gmail… que facilite descargarla desde el dispositivo de prueba.

Esto será posible siempre y cuando que en los ajustes de configuración del teléfono móvil, se haya habilitado la opción de instalar aplicaciones desconocidas, que se explicará posteriormente.

En cambio, para el desarrollador del proyecto es más cómodo transferir la aplicación de la plataforma de desarrollo al dispositivo de prueba mediante el cable de carga correspondiente de tipo USB, ya que este realiza constantes pruebas del resultado de la aplicación hasta obtener el final. Para conseguirlo, es necesario activar las opciones de desarrollador en el teléfono y realizar una serie de configuraciones que se detallarán específicamente más adelante.

Finalmente, para la implementación de la aplicación es esencial contar con los modelos y recursos en 3D, además de los recursos visuales que se van a dar uso, tanto para los marcadores que se escanearán como para los objetos 3D que se instanciarán.

4.2 Preparación del entorno

A continuación, se detallará el proceso completo de creación y desarrollo del proyecto. En primer lugar, se describirán las plataformas, herramientas y dispositivos que se han empleado concretamente, junto con la instalación y preparación del hardware utilizado. Posteriormente, se explicará la creación del proyecto y todo el desarrollo de configuración, codificación y manipulación de objetos en Unity que se ha seguido paso a paso.

Antes de iniciar el proceso de desarrollar la aplicación, se debe verificar que los dispositivos que se van a emplear cumplen los criterios necesarios, los cuales han sido explicados en el anterior apartado de este capítulo.

Así pues, para la creación de la aplicación, se ha utilizado como plataforma de desarrollo un ordenador portátil HP Pavilion Gaming Intel® Core™ i5-9300H CPU @ 2.40 GHz 9th Gen con memoria RAM instalada de 8,00 GB y sistema operativo Windows 11 de 64 bits. Además, cuenta con una GPU Intel® UHD Graphics 630 y una tarjeta gráfica integrada NVIDIA GeForce GTX 1050. Todas estas características, cumplen con los requisitos de Unity Hub mencionados anteriormente.

En cuanto al dispositivo de prueba, se hecho uso de un teléfono móvil Xiaomi MI 11 Lite 5G NE de 128GB de almacenamiento y 6+2 GB de RAM con versión de *Android* 13 y versión de MIUI 14.0.6.0.

Debido a que el dispositivo empleado es *Android*, se ha dado uso del Kit de Desarrollo de Software(SDK) de Realidad Aumentada ARCore para Unity, el cual es compatible con dicho móvil, ya que venía de serie instalado Google Play Store y cuenta con una versión superior a *Android* 7.0, que son los requisitos de compatibilidad del SDK con Unity, como se ha detallado en el anterior apartado.

Como bien se ha dicho, el hecho de que el teléfono móvil venga con *Google Play Store* instalado es importante para cumplir con los requisitos de AR Core, dado que al darse este factor, también viene instalado *Google Play Services for AR*. Este viene a ser el aplicativo del sistema *ARCore* que permite que Unity sea capaz de reconocer el dispositivo cuando se conecte al ordenador mediante el cable tipo USB, esto se puede comprobar dentro de *Google Play Store* (*Figura* 11).

Figura 11: Aplicativo AR Core

Además, en lo que respecta al teléfono móvil, para poder compartir y descargar la aplicación se necesita una previa configuración de sus ajustes. La forma más sencilla de realizar este proceso va a ser explicada en el apartado 5.3, ya que como se ha comentado, este caso es para el usuario final. Sin embargo, en cuanto al desarrollador, la forma más cómoda y ágil es mediante el cable USB, ya se realizarán constantes pruebas para la comprobación del resultado de las distintas fases de la aplicación. Por tanto para la configuración, primero se debe buscar en los ajustes del teléfono móvil, el apartado Ajustes Adicionales y después Opciones de Desarrollador y activar la opción (*Figura* 12). (Ajustes adicionales >Opciones de Desarrollador)

Figura 12: Opciones de desarrollador

Una vez activada esta función, se buscará en el apartado de depuración la opción “Depuración USB” y se habilitará (*Figura* 13). Al hacerlo, se activarán automáticamente las demás opciones complementarias, pero en este caso, al tratarse de un dispositivo Xiaomi, para que se instale la aplicación correctamente se tendrá que deshabilitar la opción ‘Verificar aplicaciones a través de USB’ (*Figura* 14).

Figura 14: Depuración USB

Figura 13: Verificar aplicaciones a través de USB.

Por otra parte, se ha descargado en el ordenador la plataforma Unity Hub versión 3.5.0.desde la página oficial de Unity (https://unity.com/es) con el plan gratuito personal.

Como se ha mencionado, esta aplicación tan solo gestiona las versiones, proyectos y módulos de Unity, por lo que también es necesario instalar una versión de Unity especifica. Dicha versión es muy importante a tener en cuenta, ya que es con la versión que se ha realizado el desarrollo del proyecto y con la que se asegura su correcto funcionamiento. En esta ocasión, se ha utilizado la versión 2021.3.1f1 LTS.

Estas últimas siglas (LTS), significan *Long-Term Support*, es decir, que la versión recibe un periodo prolongado de mantenimiento y soporte por parte de los desarrolladores y proveedores (20). Se ha utilizado una versión LTS, debido a que ante cualquier problema, es más sencillo hallar su solución a través de páginas webs y foros oficiales por parte de Unity, lo cual es una gran ventaja.

Al instalarse la versión de Unity en Unity Hub, se deben aceptar los módulos necesarios del sistema *Android*, JDK y SDK. El editor de código Visual Studio no ha sido necesario instalarlo debido a que ya se tenía instalado el editor Visual Studio Code y existe un mayor conocimiento sobre su uso. Pero en caso de no tener instalado ningún editor de código, sí que será necesaria su instalación (*Figura* 14).

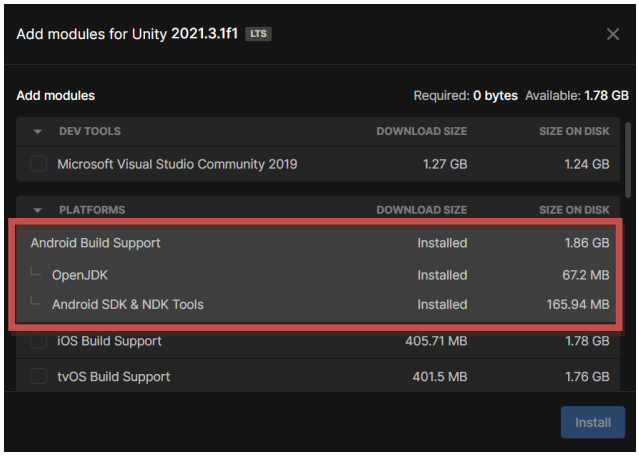


Figura 15: Módulos versión Unity

El proceso de instalación de Unity Hub, la instalación de la versión de Unity y la gestión de módulos se puede seguir mediante el siguiente videotutorial creado con dicho fin como parte del proyecto.
https://drive.google.com/file/d/1WcO97Z4tzqvY2zhLGr84TRecAHw8Lx4f/view?usp=sharing

Por último, hay que mencionar que los recursos gráficos y elementos 3D empleados, serán presentados con una mayor descripción en la sección específica destinada a ello.

4.3 Desarrollo del proyecto

Como primer punto a tratar, es importante reincidir en que el objetivo principal de este proyecto es desarrollar un proyecto en Unity que genere una aplicación personalizable para el usuario, donde un usuario final sea capaz de escanear marcadores en forma de imágenes con su dispositivo para instanciar con Realidad Aumentada diversos objetos 3D, imágenes y vídeos sobre ellos y proporcionarle una experiencia inmersiva.

Por lo que, una vez instalado todo lo necesario para hacer uso de los aplicativos de Unity, comprobado que tanto la plataforma de desarrollo como el dispositivo de prueba cumplen con los requisitos necesarios, y después de realizar las modificaciones necesarias en la configuración del dispositivo de prueba para su correcta utilización, ya es posible comenzar con el desarrollo de la aplicación.

Así pues, como primer paso al abrir Unity Hub, en el apartado *Projects* se ha creado un nuevo proyecto *New Project* (*Figura* 16), donde se ha seleccionado la plantilla AR dándole como título *AR_Marker_Project* y un directorio adecuado donde ir almacenando los avances del proyecto. Con esto, al pulsar *Create Project*, se añade a la lista de proyectos en Unity Hub. (*Figura* 17).

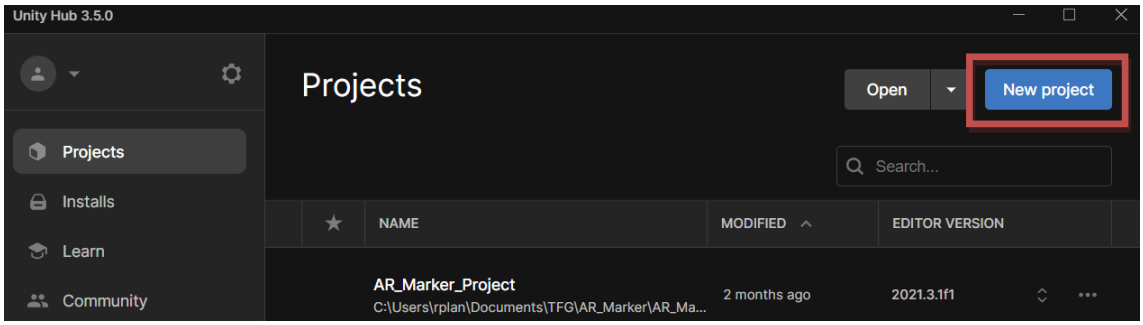


Figura 16: Creación proyecto en Unity

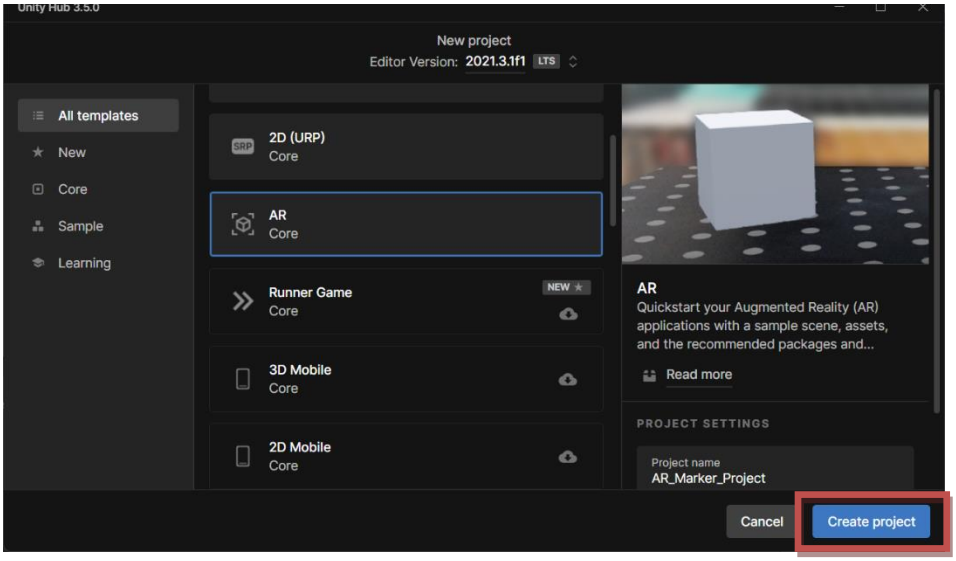


Figura 17: Creación proyecto AR Unity

Tras crear el proyecto, este se abre automáticamente y ya se puede ver la estructura que la plantilla AR contiene (*Figura* 18).

En la ventana *Project* se distinguen dos carpetas: la carpeta *Packages*, donde se encuentran manifiestos, archivos de cache, metadatos, paquetes con librerías y demás archivos, los cuales no se van a indagar en mayor medida durante el desarrollo de la aplicación; y la carpeta *Assets*, donde se almacenarán posteriormente todos los recursos a emplear.

Esta última carpeta contiene a su vez cuatro directorios más: la carpeta *ExampleAssets*, que no se ha utilizado debido a que todos los recursos empleados han sido creados desde cero o extraídos legalmente de páginas webs; la carpeta *Plugins*, la cual tampoco se ha utilizado; la carpeta *Scenes*, donde se encuentra una escena en la cual se ha desarrollado el trabajo y donde poder almacenar otras escenas, y por último, la carpeta X*R*. Esta carpeta es muy importante dado que contiene todos los archivos necesarios para poder trabajar con la Realidad Aumentada, así como AR Core y permite integrarlo al proyecto. En conclusión, se ha dado uso de las dos últimas carpetas mencionadas, las dos primeras se han eliminado (*Figura* 18).

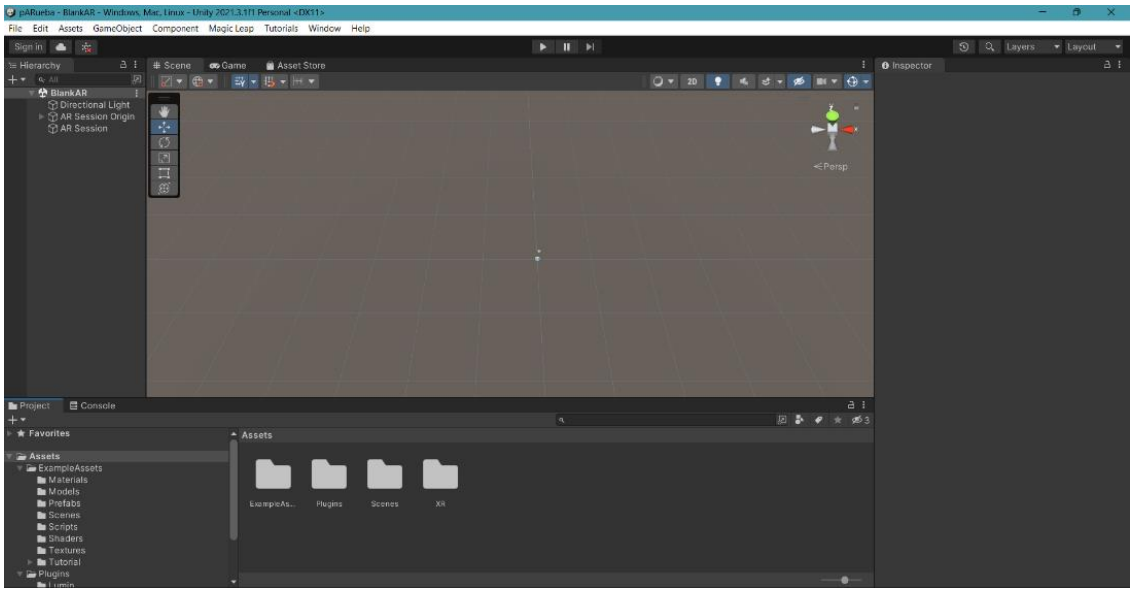


Figura 18: Interfaz AR Unity

En el interior del directorio *Scenes*, se encuentra ya una escena creada a la cual se le ha dado el nombre de *MultipleMarkers*, donde se ha trabajado la mayor parte del proyecto. En su interior, en la ventana *Hierarchy* vienen ya creados tres *GameObjects* (término que hace referencia a un objecto en Unity) (*Figura* 19).

El primero de ellos, *Directional Light*, un foco que emite luz en una única dirección y está colocada a una distancia infinita (21). En segundo lugar, *ARSessionOrigin*, que es el objecto que representa al dispositivo de prueba en el espacio, en este proyecto representa al teléfono móvil que escaneará los marcadores. *ARSessionOrigin* cuenta en su interior con el *GameObject ARCamera* conectado él, este objeto es la cámara principal de la aplicación para permitir las funciones AR, y por ello, tiene como etiqueta el término *Main Camera*.

Por último, se encuentra el *GameObject ARSession* que supervisa cada acción especifica de la experiencia en AR, la velocidad en fotogramas, la comunicación con el dispositivo, así como otras funciones (22).

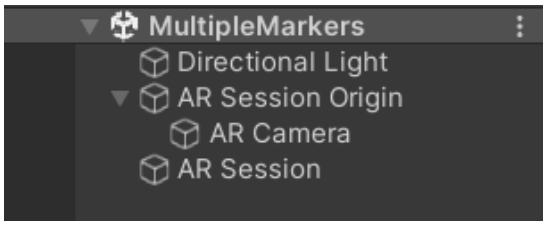


Figura 19: GameObjects escena AR

Tras conocer la estructura de la plantilla AR, se ha importado todo lo necesario del paquete *ARFoundation* al proyecto para hacer uso de las funciones de Realidad Aumentada. Para ello, en el menú principal del proyecto, al seleccionar *Window* en la barra superior, y después PackageManager, es posible observar una lista de los paquetes que contiene dicho proyecto si se tiene seleccionada la opción *Package:InProject*(*Figura* 20).

En estos ajustes se debe instalar los paquetes *ARFoundation*, que ofrece los *GameObjects* y clases esenciales para crear la experiencia AR; *ARCoreXRPlugin*, que permite trasladar las funciones AR a los dispositivos móviles, y el paquete *XRPluginManagement*, que es simplemente un instrumento que gestiona todos los accesorios de XR destinados al marco de *ARCore* (22).

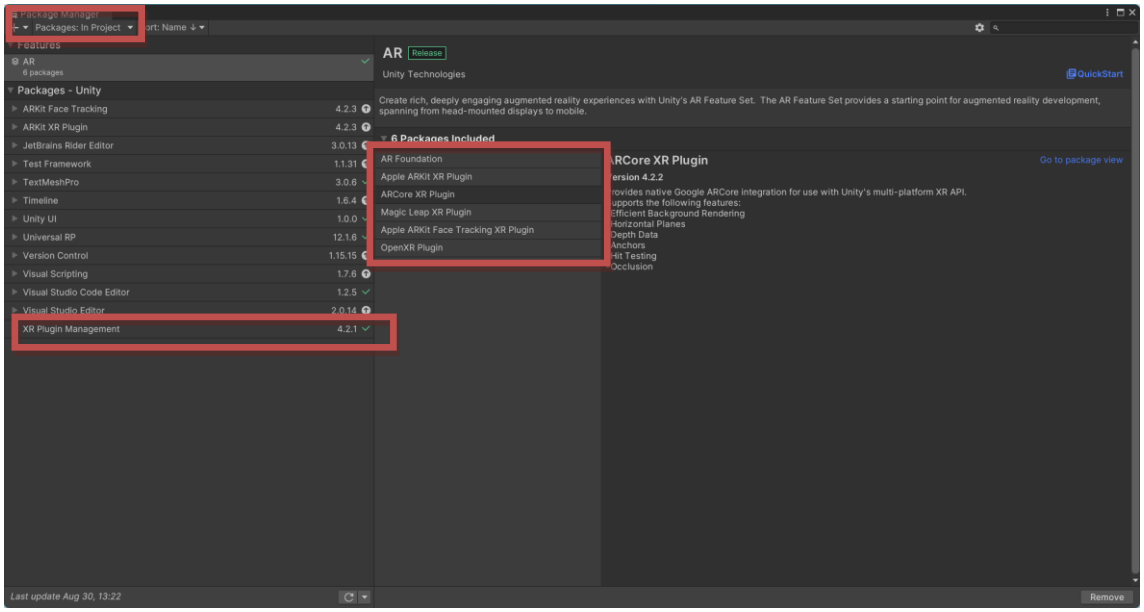


Figura 20: Package Manager

Después de haber instalado los paquetes que permiten las funciones de AR, es necesario comprobar si estos se han activado correctamente en el proyecto. Para llevar a cabo esto, se requiere ir a la opción *Edit* en la ventana superior del menú principal, y después en los ajustes *ProjectSettings*. En la barra lateral, se selecciona *XRPlugin-inManagement*, y se comprueba que en el apartado *Android* este activa la opción A*RCore* (*Figura 21*).

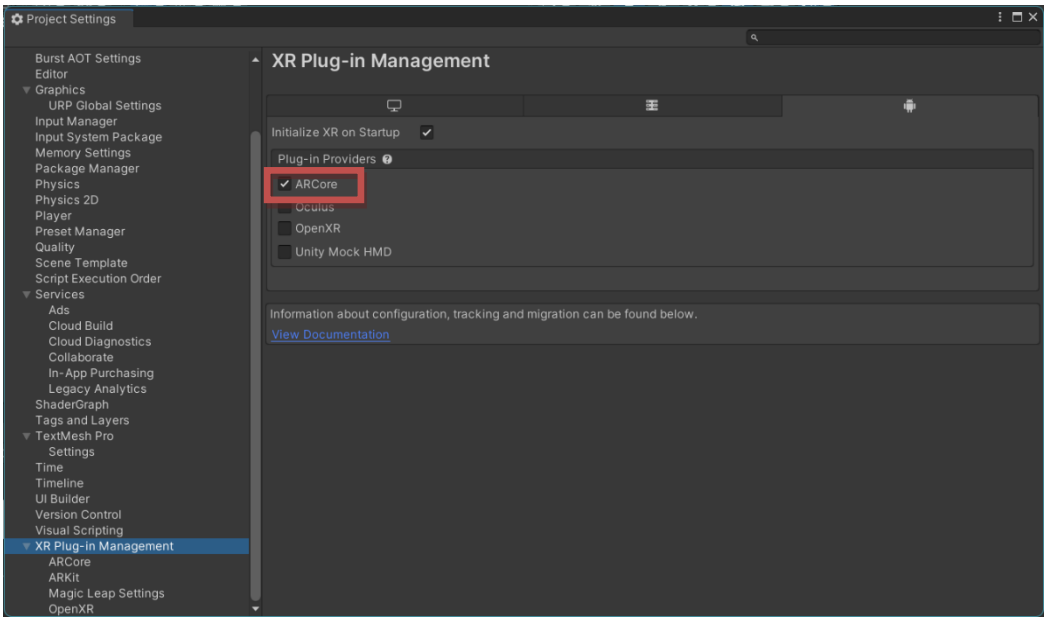


Figura 21: XR Plug-in Management

Para acabar con las primeras configuraciones importantes, antes de añadir las funcionalidades AR al proyecto, se debe cambiar la plataforma de lanzamiento de aplicación a *Android*. Esto se realiza desde la opción *File*, en la barra superior del menú principal, y se entra en los ajustes, *Build Settings*. Aquí, se selecciona la plataforma *Android* y se pulsa en *Switch Platform* (*Figura 22*).

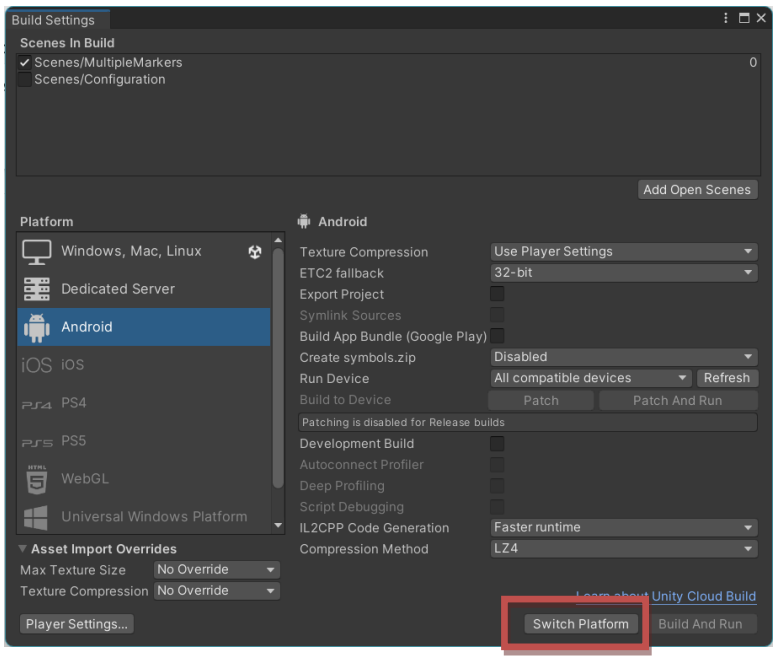


Figura 22: Cambio de plataforma a Android

4.3.1 Implementación de la detección de marcadores

Con toda la configuración verificada en el proyecto para poder generar una aplicación de AR con *ARFoundation* en un dispositivo *Android*, ya es posible añadir las funcionalidades que se requieren para hacer uso de la Realidad Aumentada y dar paso al mundo virtual.

Como primer resultado, se busca que el teléfono móvil sea capaz de detectar marcadores en forma de imágenes, en este caso, serán imágenes personalizadas. Para conseguirlo, se debe implementar una serie de componentes a los *GameObjects ARSession* y *ARSessionOrigin* de la escena.

En primer lugar, el *GameObject ARSession* ya viene de serie con dos *scripts* incorporados, que se pueden observar en la ventana inspector (explicada anteriormente en la sección 3.3) a la parte derecha del proyecto. El primer *script* se denomina de la misma manera, *ARSession*, el cual se encarga principalmente de gestionar y rastrear la información del mundo real, para así, que los elementos del mundo virtual se coloquen y se visualicen en el lugar correspondiente(23). El otro *script*, llamado *ARInputManager*, que controla la vida útil de la clase *XRInputSubsystem* y obtiene información constante del dispositivo (24).
Es importante destacar que solo es posible tener una clase *ARSession* activa por escena, ya que si existiese más de una, todas se comunicarían con la misma acción y colapsarían (*Figura* 24).

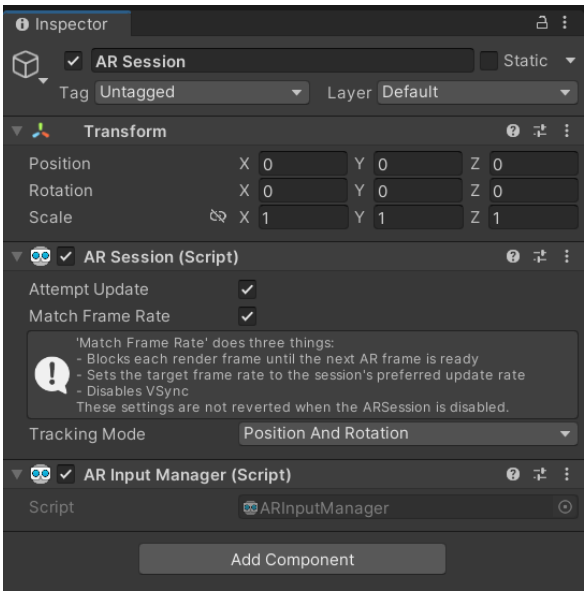


Figura 23: GameObject ARSession

Por otro lado, el *GameObject ARSessionOrigin* simplemente viene de serie con un *script* llamado igual, *ARSessionOrigin*. Este *script* tiene un campo llamado cámara, al que se le asigna el *GameObject ARCamera* que tiene conectado el *GameObject ARSessionOrigin*, como se ha explicado anteriormente.

Su función es la transformación y representación de los objetos virtuales, manejando sus coordenadas, corrigiendo su escala y rotación, ofreciendo una mayor estabilidad al seguimiento, y manteniendo la cámara y cualquier elemento detectado en el espacio relativo, que viene a ser la ubicación inicial del dispositivo, en este caso, el teléfono móvil (25).

Pero para la detección de marcadores, es necesario añadir otras componentes en *ARSessionOrigin*. La componente *ARTrackedImageManager* es la encargada de crear un *GameObject* por cada marcador detectado, así que para incorporarla, se selecciona el *GameObject ARSessionOrigin* y se pulsa en *Add Component*, después se busca el nombre de la componente y se confirma (*Figura* 24).

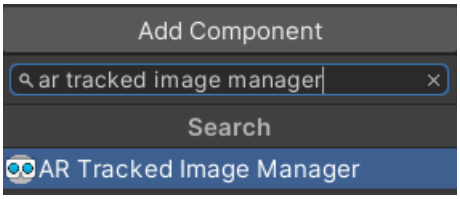


Figura 24: Añadir componente ARTrakedImageManager

Como se puede observar en la *Figura* 25 en esta componente existen 3 campos, que serán explicados a continuación. .

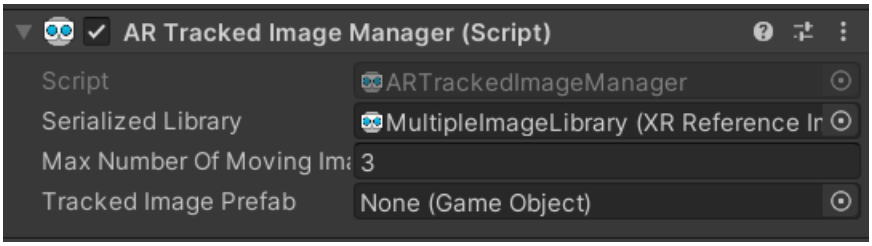


Figura 25: ARTrakedImageManager

- ***Serialized Library*:** se trata de una librería de imágenes la cual es creada por el usuario y sirve para saber cuáles son las imágenes que trabajarán como marcadores, es decir, la componente *ARTrackedImageManager* solo creará *GameObjects* en caso de que la imagen detectada se encuentre dentro de esta librería de imágenes. (26)

Para generar esta librería de imágenes, en primer lugar se ha creado un nuevo directorio en la ventana *Project* llamado *_ARMarker* para tener más organizada la disposición de los recursos. En el interior de esta, se ha incluido una nueva carpeta llamada *Markers*, donde se albergarán tanto los marcadores como esta librería de imágenes. Así pues, una vez en su interior se debe pulsar el botón derecho y seguir el siguiente proceso: *Create>XR>Reference Image Library* (*Figura* 26).

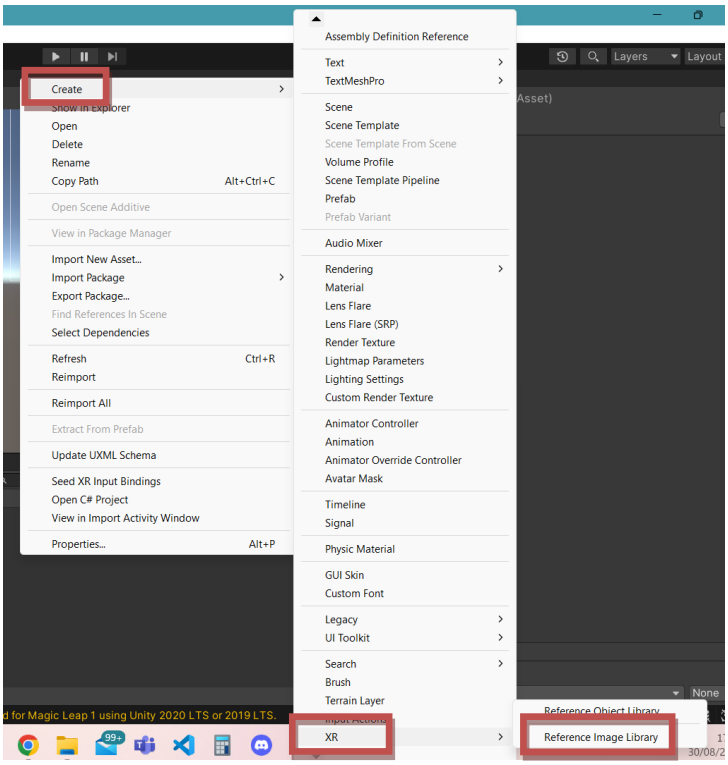


Figura 26: Creación librería de imágenes ReferenceImageLibrary

En este caso, se le denomina *MultipleImageLibrary* porque en la aplicación es posible escanear diferentes marcadores. Tras esto, se arrastra la librería desde la ventada *Project* hasta el campo *SerializedLibrary* de la componente *ARTrackedImageManager* como se ve en la *Figura* 25.

- **Max. Number of Moving Images To Track:** en este campo se escribe el máximo número de marcadores que se pueden escanear simultáneamente. En este caso son tres, pero puede ser otro valor, aunque un número alto puede consumir gran cantidad de recursos en cuanto a CPU (26).

- **Tracked Image *Prefab*:** se refiere al *prefab* que se instanciará en caso de que el dispositivo detecte un marcador incluido en la librería de imágenes. De esta información se conoce que, para instanciar un *GameObject* debe ser un *prefab*, y que, solamente es posible incluir un único objeto a instanciar, lo cual no es de interés, ya que se pretende instanciar diferentes objetos asociados a sus marcadores correspondientes. Por este motivo, el campo está relleno con el valor *None*. Posteriormente se explicará de qué manera se consigue instanciar diferentes objetos asociados a unos marcadores en concreto.

Conocido ya el funcionamiento de la componente *ARTrackedImageManager*, se ha realizado un primer test de la aplicación simplemente incorporando una sola imagen como marcador en la librería de imágenes, y un solo *prefab* a instanciar en el campo *TrackedImagePrefabToTrack*.
Para ello, se escogió una imagen cualquiera y se creó un objeto 3D, en concreto un cubo, como *prefab.* El resultado fue el esperado, al abrir la aplicación se iniciaba la cámara del teléfono móvil y al detectar la imagen correspondiente aparecía el cubo instanciado.
Al ser la primera vez de ejecución de la aplicación, se debe establecer la resolución de pantalla para que los elementos de la escena se adapten a ella. Esto, se realiza con la ventana *Game* seleccionada, donde se pulsa en el tercer desplegable y se selecciona la opción *16:9 Portrait*, que es la escala correcta para lanzar una aplicación en un teléfono móvil (*Figura* 27).

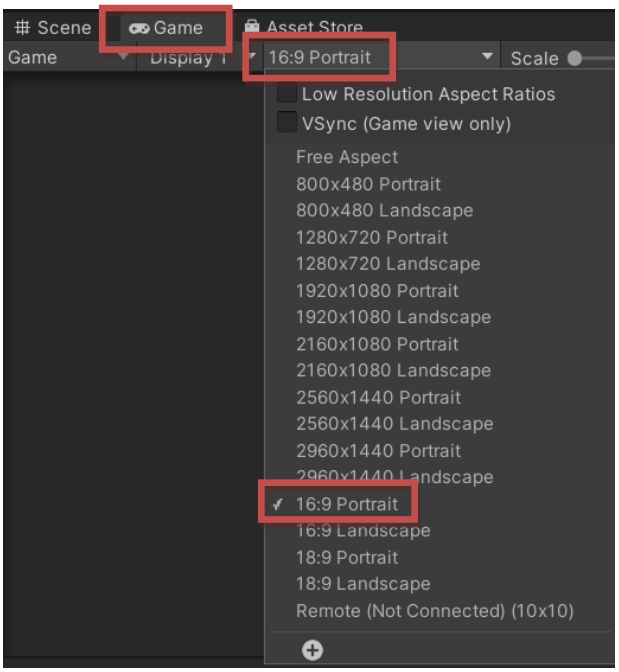


Figura 27: Resolución 16:9 Portrait

Además, es muy importante destacar una serie de conclusiones que se extrajeron tras esta primera prueba.

Primero, que para poder realizar una instancia, el elemento a instanciar debe ser o estar conectado a un elemento de tipo *prefab*.

Por otra parte, los valores del *prefab* en el campo de *Transformation* en el inspector deben ser unos en concreto para que la instancia del objeto se vea correctamente. No obstante, antes de tener en cuenta las coordenadas, se debe cumplir otro requisito, el objeto a instanciar debe ser un *EmptyObject* (padre) conectado al objeto en concreto (hijo), que actúa como malla o *mesh*.

Una vez creada esta relación entre objetos, ya se pueden establecer las coordenadas pertinentes para el objeto padre e hijo. En el caso de la posición, el *EmptyObject* (padre) debe tener las coordenadas (0,0,1) en base a (X,Y,Z), para que así, se instancie a una distancia prudente respecto al teléfono móvil. Después, en cuanto a la rotación, la malla del objeto (hijo) en cuestión debe tener las coordenadas (90,0,0), para que al instanciarse, el elemento quede de cara al usuario. Y por último, la escala también debe aplicarse a la malla (hijo) y debe reducirse, ya que si no es así, cualquier elemento instanciado a primera vista queda muy grande en cuanto a tamaño y su visualización no es correcta. A continuación, se muestra un ejemplo con un cubo 3D como objeto en la *Figura* 28.

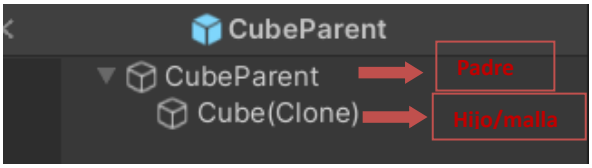


Figura 28: Composición necesaria de prefabs a instanciar

Todos estos factores y valores se deben tener en cuenta a la hora de configurar cualquier recurso que quiera ser instanciado. Su implementación se verá más detalladamente en el apartado 4.3.3.

4.3.2 Asociación de marcadores a objetos 3D, imágenes y vídeos

Tras comprobar que el funcionamiento de la detección de marcadores y la instancia de *GameObjects* es correcta, se puede continuar con el objetivo del proyecto.

El siguiente paso es tratar de, teniendo diferentes imágenes como marcadores en la librería de imágenes, asociar diferentes recursos, objetos 3D, imágenes y vídeos, e instanciar dichos elementos cuando se detecte el marcador al que está asociado.

Antes de desarrollar cómo se ha realizado esta función, es importante mencionar que en este proyecto se han creado marcadores originales para dar un enfoque de mayor calidad.
Para ello los marcadores que se incluyen en la librería de imágenes deben cumplir unas características, con el objetivo de que su detección sea óptima:

- Su resolución debe ser mínimo de 300x300 píxeles y no necesariamente tienen que ser cuadrados.
- La imagen tiene que ser formato PNG o JPG.
- La imagen debe tener un contraste significativo (27).

Los **marcadores** de este proyecto se han generado en el programa *Photoshop* en base a estas características. Como el proyecto está enfocado a la docencia, se han creado tres marcadores para tres grados universitarios distintos: Bellas Artes, Ingeniería de Telecomunicaciones y Tecnología Digital y Multimedia (GTDM). Todas ellas de la Universitat Politécnica de Valencia, que es la universidad que ha sustentado dicho Trabajo de Final de Grado.

A la hora de la creación de estos marcadores en *Photoshop*, para todos ellos se ha escogido primeramente un fondo blanco, se ha colocado en la parte inferior derecha el logo del grado y en la parte superior izquierda el logo de la universidad, en la parte central un elemento característico de cada grado sobre otro cuadrado con fondo blanco, y para facilitar la detección del marcador, se ha incluido mediante la función 'Rellenar con un motivo' la repetición de diferentes elementos como cuadrados, círculos y pentágonos sobre el fondo blanco (*Figura* 28). Para este último proceso se ha seguido el siguiente vídeo. https://www.youtube.com/watch?v=NxFBQlS7Ujs&ab_channel=SaberProgramasImagen

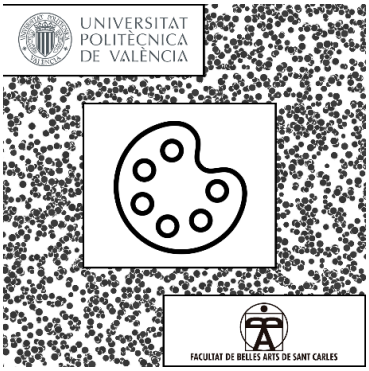


Figura 29: Marcador Bellas Artes

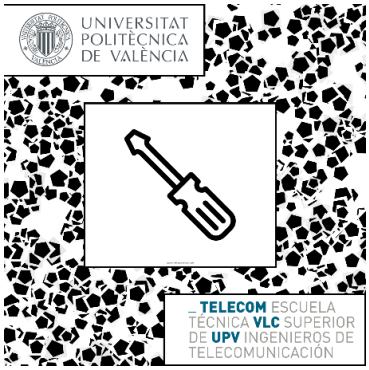


Figura 31: Marcador Telecomunicaciones

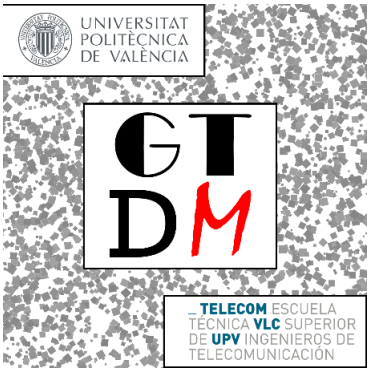


Figura 30: Marcador GTDM

Además, para probar la función que se desea conseguir, se va a utilizar un objeto 3D (*Figura* 32), una imagen (*Figura* 33) y un vídeo (*Figura* 34). Cada elemento estará asociado a un marcador para así utilizar diversos recursos que permite la AR.

Figura 32: Objeto 3D Bellas Artes

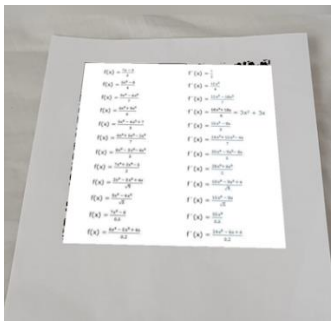


Figura 33: Imagen Telecomunicaciones

Figura 34: Vídeo GTDM

Cabe destacar que el objecto 3D puede ser tanto generado por un usuario en cualquier aplicación de modelado (Blender, Wings 3D, Meshmixer…) o extraído bajo licencia de alguna página web (Mixamo, Fiverr, Anatomy360…). En este proyecto, se ha empleado un balón de fútbol 3D descargado gratuitamente de Mixamo. Se ha escogido de una página web para de esta forma asemejarse a la situación real de lo que realizaría cualquier usuario final, en este caso un o una docente, y así, se ha podido comprobar mejor su funcionamiento.

Con todos los recursos, marcadores y elementos a instanciar preparados, ya es posible realizar pruebas para llegar a conseguir asociar los elementos a los marcadores y solo ser instanciados si se detecta su marcador correspondiente. La forma más factible de conseguirlo es mediante un ***script***, así que, se clica en el *GameObject ARSessionOrigin* y en el inspector se añade una componente de tipo *script* (*Figura 35*).

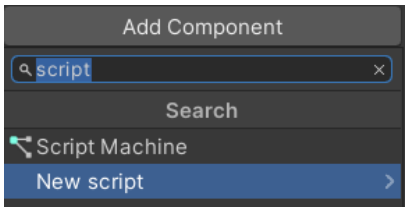


Figura 35: Add component script

Al crearlo, se abre el editor de código predeterminado, en este proyecto se da uso de Visual Studio Code, como ya se ha mencionado. Se nombra el *script*, en este caso ***Partner***, porque es el encargado de asociar marcador y elemento, y se guarda en el proyecto. Para tenerlo ubicado, ya que se necesitará realizar constantes cambios, se crea una nueva carpeta en el interior del directorio *_ARMarker* llamada *Scripts* y se almacena en esta.

La estructura principal del código empleado en el *script Partner*, así como, el entendimiento para cumplimentar las funciones que realiza este, se han obtenido del manual oficial de Unity (28).

En primer lugar, se importan las bibliotecas necesarias para permitir a Unity trabajar y usar la biblioteca *ARFoundation* (*Figura*36).

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
// Librerias AR
using UnityEngine.XR.ARFoundation;
using UnityEngine.XR.ARSubsystems;
```

Figura 36: Importación bibliotecas

Tras esto, en el interior de la clase *Partner* se declaran las variables necesarias que se utilizarán para almacenar valores, las cuales se explican seguidamente (*Figura* 37):

- **m_trackedImageManager:** es la referencia a la componente *ARTrackedImageManager*
- ***prefab*ToInstanciate:** es una lista que alberga los nombres de los marcadores y los *prefab*s pertinentes.
- **instanciate*Prefab*:** es un diccionario que mapea los nombres de los marcadores a los *prefabs* instanciados y almacena la información como par clave (marcador) – valor (*prefab*). Esto sirve para que al detectar un marcador (clave) se busque en el diccionario su *prefab* (valor) y así lo instancie.

```
private ARTrackedImageManager m_trackedImageManager;

[SerializeField]
private TrackedPrefab[] prefabToInstantiate;

private Dictionary<string, GameObject> instanciatePrefab;
```

Figura 37: Declaración de variables

Tras la declaración de las variables, se procede a explicar las diferentes **clases y métodos** que permiten asociar los *prefab*s con los marcadores.

En primer instante se encuentra la función *Awake()*, que tiene como labor inicializar las variables, en este caso, la componente *ARTrackedImageManager* y el diccionario del par marcador-*prefab* (*Figura* 38).

```
private void Awake()
{
    m_trackedImageManager = GetComponent<ARTrackedImageManager>();
    instanciatePrefab = new Dictionary<string, GameObject>();
```

Figura 38: Método Awake

Después, para responder a los marcadores detectados se encuentran las funciones *OnEnable()* y *OnDisable()*, las cuales se suscriben al evento *trackedImagesChanged* de *ARTrackedImageManager* para ser notificadas cada vez que se añada, se elimine o se actualice un marcador (*Figura* 39).

```
private void OnEnable()
{
    m_trackedImageManager.trackedImagesChanged += OnTrackedImageChanged;
}

private void OnDisable()
{
    m_trackedImageManager.trackedImagesChanged -= OnTrackedImageChanged;
}
```

Figura 39: Funciones OnEnable y OnDisable

Ligado a estos dos métodos, es preciso definir la función *OnTrackedImageChanged()*, dado que es la función que se realiza cuando el evento *trackedImagesChanged* reciba una notificación. En su interior consta de tres opciones que pueden ocurrir durante la ejecución de la aplicación (*Figura* 40):

1. ***addedImage*:** recorre la lista de marcadores agregados y cuando se detecta un marcador, se llama a la función *InstantiateGameObject*, que a continuación se explicará su funcionamiento.
2. ***updatedImage*:** recorre la lista de marcadores actualizados y, en base a la calidad de detección del marcador se llama a una función adicional u otra. El estado de calidad de detección se llama *trackingState*. Si el valor de este estado es *Tracking* significa que la calidad es óptima para instanciar el *prefab*, si el estado es *Limited* implica que la calidad es estándar y puede que el *prefab* se instancie o no, y para el caso en el que no se dé ninguno de estos dos estados, quiere decir que la detección es mala, por lo que el *prefab* no se instanciará correctamente.
3. ***removedImage*:** recorre la lista de marcadores eliminados y destruye el *prefab* asociado a él.

```
private void OnTrackedImageChanged(ARTrackedImagesChangedEventArgs eventArgs)
{
    foreach (ARTrackedImage addedImage in eventArgs.added)
    {
        InstantiateGameObject(addedImage);
    }

    foreach (ARTrackedImage updatedImage in eventArgs.updated)
    {
        if(updatedImage.trackingState == UnityEngine.XR.ARSubsystems.TrackingState.Tracking)
        {
            UpdateTrackingGameObject(updatedImage);
        }
        else if(updatedImage.trackingState == UnityEngine.XR.ARSubsystems.TrackingState.Limited)
        {
            UpdateLimitedGameObject(updatedImage);
        }
        else
        {
            UpdateNoneGameObject(updatedImage);
        }
    }

    foreach (ARTrackedImage removedImage in eventArgs.removed)
    {
        DestroyGameObject(removedImage);
    }
}
```

Figura 40: Función OnTrackedImageChanged

Así pues, es importante detallar las funciones que realizan las acciones para cumplir los diferentes casos del método *OnTrackedImageChanged*, estas son:

- ***InstantiateGameObject*:** esta función ocurre cuando se detecta un marcador y es la encargada de instanciar su *prefab* correspondiente. Para ello, recorre la lista *PrefabToInstantiate* con los nombres de *prefab*s y marcadores, compara si son los que están asociados, y si se corresponden, se instancia el *prefab* vinculado, actualizando su posición y rotación. Aparte, se añade el marcador y *prefab* asociado al diccionario como par clave-valor para de esta manera tener ya almacenada dicha información (*Figura* 41).

```
private void InstantiateGameObject(ARTrackedImage addedImage)
{
    for (int i = 0; i < prefabToInstantiate.Length; i++)
    {
        if (addedImage.referenceImage.name == prefabToInstantiate[i].name)
        {

            GameObject prefab = Instantiate<GameObject>(prefabToInstantiate[i].prefab, transform.parent);
            prefab.transform.position = addedImage.transform.position;
            prefab.transform.rotation = addedImage.transform.rotation;

            instanciatePrefab.Add(addedImage.referenceImage.name, prefab);

            currentPrefab = prefab;

        }
    }
}
```

Figura 41: Función InstantiateGameObject

- ***UpdateTrackingGameObject*:** esta es la función a la que se llama cuando el *prefab* está siendo rastreado con precisión. En ella también se recorre la lista *Prefab*ToInstantiate para comprobar que el *prefab* que está instanciado es el asociado al marcador. Su principal función es mantener la vista del *prefab* activa actualizando constantemente su posición y rotación (*Figura 42*).

```
private void UpdateTrackingGameObject(ARTrackedImage updatedImage)
{

    for (int i = 0; i < instanciatePrefab.Count; i++)
    {
        if (instanciatePrefab.TryGetValue(updatedImage.referenceImage.name, out GameObject prefab))
        {
            prefab.transform.position = updatedImage.transform.position;
            prefab.transform.rotation = updatedImage.transform.rotation;
            prefab.SetActive(true);
        }
    }
}
```

Figura 42: Función UpdateTrackingGameObject

- ***UpdateLimitedGameObject*:** esta función interviene cuando el marcador está siendo rastreado con una precisión un tanto limitada (*Figura 43*). Funciona de la misma manera que la anterior pero en este caso se comprueba si el *prefab* instanciado tiene el campo *DestroyOnRemoval* activado o no de la componente *ARTrackedImage* (*Figura 44*). En caso de tenerlo activado, cuando la precisión del marcador no sea lo suficientemente óptima para poder instanciarlo, está función desactivará la vista del *prefab*. Por el contrario, si no se encuentra activada, el *prefab* seguirá instanciado actualizando posición y rotación aunque no se detecte el marcador. La componente *ARTrackedImage* se explicará más adelante en la sección 4.3.3.

```
private void UpdateLimitedGameObject(ARTrackedImage updatedImage)
{
    for (int i = 0; i < instanciatePrefab.Count; i++)
    {
        if (instanciatePrefab.TryGetValue(updatedImage.referenceImage.name, out GameObject prefab))
        {
            if(!prefab.GetComponent<ARTrackedImage>().destroyOnRemoval)
            {
                prefab.transform.position = updatedImage.transform.position;
                prefab.transform.rotation = updatedImage.transform.rotation;
                prefab.SetActive(true);
            }
            else
            {
                prefab.SetActive(false);
            }

        }
    }
}
```

Figura 43: Función UpdatedLimitedGameObject

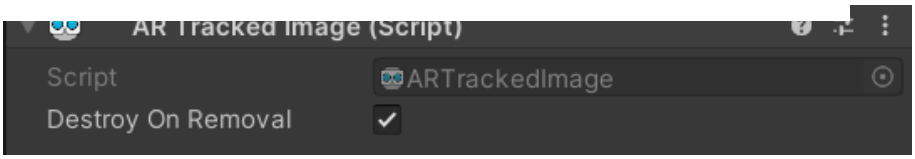


Figura 44: : Campo DestroyOnRemoval de la componente ARTrackedImage

- ***UpdateNoneGameObject*:** se requiere de esta función cuando el marcador ya no se rastrea, por lo que desactiva la vista del *prefab* asociado a ese marcador (*Figura* 45).

```
private void UpdateNoneGameObject(ARTrackedImage updateImage)
{
    for(int i = 0; i < instanciatePrefab.Count; i++)
    {
        if(instanciatePrefab.TryGetValue(updateImage.referenceImage.name, out GameObject prefab))
        {
            prefab.SetActive(false);
        }
    }
}
```

Figura 45: Función UpdateNoneGameObject

- ***DestroyGameObject*:** es el último caso y se le llama cuando la imagen ha sido eliminada o no es visible. Se recorre el diccionario en busca de la clave (marcador) que ha sido eliminada y se elimina también su valor (*prefab*) asociado, además de destruir su instancia (*Figura* 46).

```
private void DestroyGameObject(ARTrackedImage removedImage)
{
    for (int i = 0; i < instanciatePrefab.Count; i++)
    {
        if (instanciatePrefab.TryGetValue(removedImage.referenceImage.name, out GameObject prefab))
        {
            instanciatePrefab.Remove(removedImage.referenceImage.name);
            Destroy(prefab);
        }
    }
}
```

Figura 46: Función DestroyGameObject

Por último, es importante definir la estructura que empaqueta la información aportada por el usuario con las variables, llamada *TrackedPrefab* en este caso: *name*, que hace referencia al nombre del marcador, y *prefab*, que será el *GameObject* a instanciar (*Figura 47)*.

```
[System.Serializable]
public struct TrackedPrefab
{
    public string name;
    public GameObject prefab;
}
```

Figura 47: Estructura TrakedPrefab

Estos valores, son los que aparecen en el inspector al clicar en la componente *ARSessionOrigin*, y es donde el usuario tiene que añadir pares de valores, pulsando en el icono '+' y añadiendo el nombre del marcador y el *prefab* que desea que estén asociados (*Figura 48*).

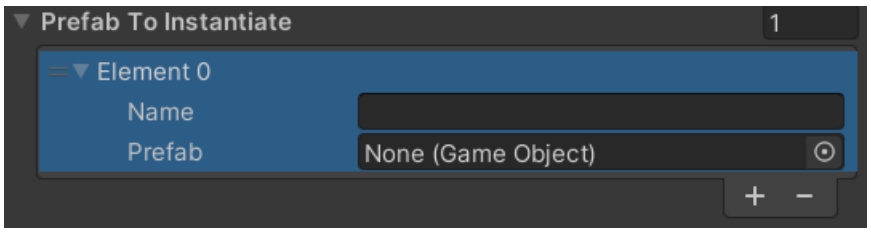


Figura 48: Asociación par marcador-prefab

Tras la generación de este código, en un primer momento la comprobación del funcionamiento del *script* se realiza mediante el uso de objetos 3D únicamente (ni imágenes ni vídeos), por que tan solo se trata de conseguir que el elemento instanciado fuese el elemento asociado al marcador hecho en el inspector, como en la *Figura* 48 .

El resultado fue el esperado, ya que al detectar un marcador añadido en la lista *PrefabToInstanciate* en el inspector de *ARSessionOrigin*, se instanciaba el *prefab* asociado también en esta. Por supuesto, se tuvo en cuenta que todos los elementos eran *prefab* y sus coordenadas de transformación como ya se ha comentado anteriormente.

4.3.3 Funcionalidades adicionales: creación del entorno personalizable para usuario

Cabe recordar, que uno de los objetivos de este trabajo, es poder hacer uso de diferentes recursos en la AR como objetos 3D, imágenes y vídeos, además de generar un entorno personalizable para un posible usuario que decida utilizar el proyecto y la aplicación.

Es importante incidir en que para que el proyecto pueda llegar a ser personalizable para un usuario, es necesario que haya poco que configurar o programar en Unity por su parte, dado que lo más seguro es que el usuario no tenga grandes conocimientos acerca de esta herramienta.

Al investigar acerca de la incorporación de imágenes y vídeos como instancias en el proyecto se ha observado la necesidad de realizar diversas configuraciones en el entorno de Unity como la creación de materiales, incluir componentes, ajustar parámetros, y otras más.

Con todo esto y sumado a que para poder instanciar un elemento 3D, el usuario debe convertir el objeto en *prefab* y ajustar sus coordenadas, se llega a la conclusión de que el entorno personalizable es demasiado complejo para cualquier usuario y es necesario adaptarlo.
Así pues, teniendo en cuenta la complejidad de la preparación y configuración previa del proyecto, se han añadido funcionalidades con el fin de facilitar el trabajo por parte de los y las docentes, cumpliendo así el segundo objetivo citado en anteriormente.

Por consiguiente se ha creado una **nueva escena** en la que el usuario pueda incluir todos los objetos 3D, imágenes y vídeos, que desea instanciar posteriormente de tal forma que, al añadir todos los recursos necesarios en esta escena y ejecutarla, estos quedan configurados correctamente para ser instanciados. Este proceso será explicado a continuación detalladamente.

En primer lugar, dentro de la carpeta *Scenes* se clica el botón derecho y se selecciona: *Create>Scene*, para así crear una nueva escena, que en este caso se denomina *Configuration* (*Figura* 49).

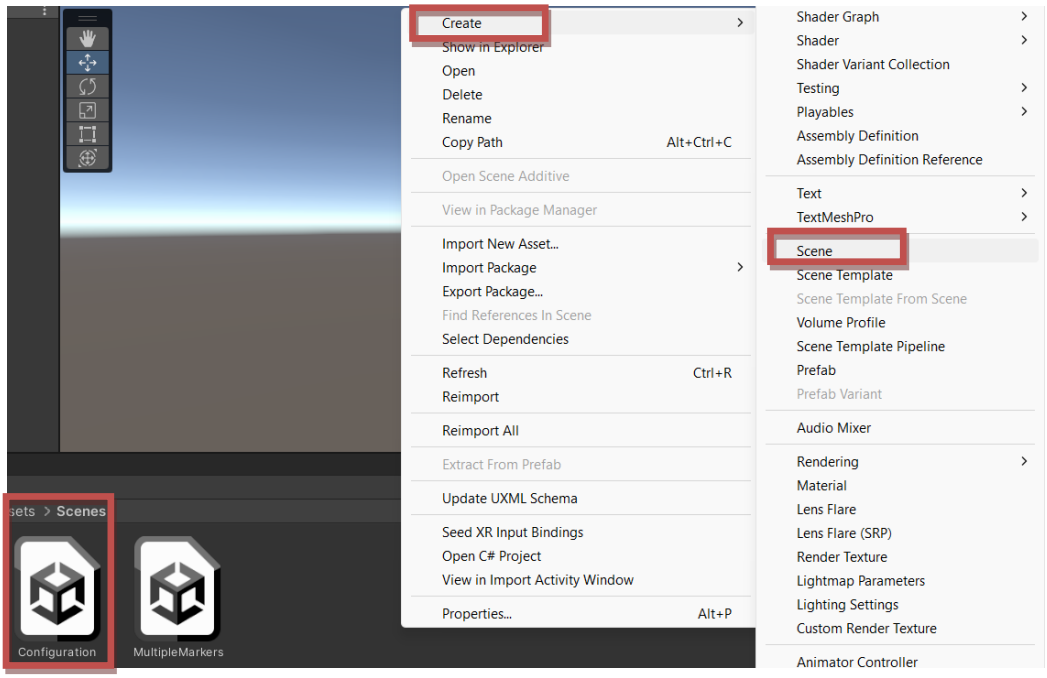


Figura 49: Creación nueva escena

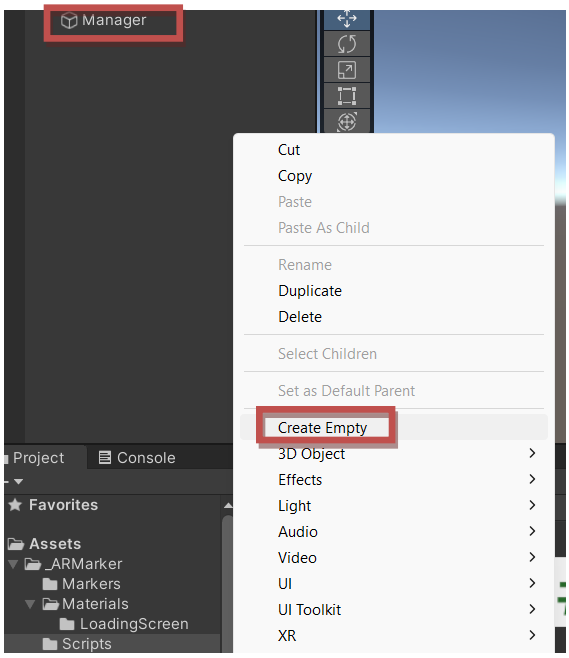


Figura 50: Creación de objeto vacío

Tras esto, se crea un nuevo objeto vacío dentro de la escena clicando el botón derecho en la ventana *Hierarchy* y *CreateEmpty*. Este será el responsable de soportar los *scripts* encargados de gestionar el proceso de configuración y se denomina *Manager* (*Figura* 50).

Así pues, se crean tres *scripts*, uno por cada recurso a convertir (objetos 3D, imágenes y vídeo) y se almacenan en el interior de la carpeta *scripts*. Estos, son los encargados de convertir los objetos 3D, imágenes y vídeos.

- ***SetObjectParent*:** este es el *script* encargado de realizar los cambios necesarios en los objetos 3D que se desean instanciar.

En cuanto a las bibliotecas, se requieren las mismas que en el *script Partner*, añadiendo la bilbioteca *UnityEditor*, la cual permite realizar acciones únicamente en el editor, y se encuentra entre la condición de *UNITY_EDITOR* para que no se ejecute al generar la aplicación (*Figura* 51).

```
#if UNITY_EDITOR
using UnityEditor;
#endif
```

Figura 51: Librería UNITY_EDITOR

Tras esto, dentro de la clase *SetObjectParent* se define como variable pública una lista de tipo *GameObject*, llamada *PrefabsToSetParen*t, que es donde se almacenan todos los objetos 3D que se desean convertir, y un objeto vacío E*mptyObject*, que actúa como padre de objeto 3D.

Con estas variables creadas, primero se recorre la lista rellena de objetos para saber a cuántos aplicar el proceso de conversión. Este consiste primero en crear un objeto vacío, colocar los elementos con la posición, rotación y escala correcta (explicada al final del apartado 4.3.1) y convertir el objeto 3D en hijo del objeto vacío, lo que hace que sean el mismo elemento.

También, se añade la componente *ARTrackedImage*, la cual es necesaria que la tengan incorporada todos los recursos que se desee instanciar para que el proceso funcione correctamente. En esta componente, viene por defecto activado el campo *DestroyOnRemoval* que tiene como función desactivar la vista del objeto instanciado si no se está detectando su marcador asociado.

Finalmente, el elemento se convierte a *prefab* y se almacena en una nueva carpeta llamada *ToInstanciate*, que es el directorio donde se encuentran los elementos ya convertidos y listos para instanciar (*Figura* 52). Todo este proceso se encuentra también entre la condición *UNITY_EDITOR* ya que no debe ejecutarse al generar la aplicación, es solo un proceso intermediario.

```
public class SetObjectParent : MonoBehaviour
{
    public GameObject[] PrefabsToSetParent;
    private GameObject EmptyObj;

    void Start()
    {

#if UNITY_EDITOR

      for(int i = 0; i < PrefabsToSetParent.Length; i++)
      {

        EmptyObj = new GameObject(PrefabsToSetParent[i].name + "Parent");
        GameObject prefabMesh = Instantiate(PrefabsToSetParent[i], new Vector3(0,0,0), Quaternion.Euler(90,0,0));
        prefabMesh.transform.localScale = new Vector3(0.3f,0.3f,0.3f);
        prefabMesh.transform.parent = EmptyObj.gameObject.transform;
        EmptyObj.transform.position = new Vector3(0,0,1);

        ARTrackedImage trackedImageComponent = EmptyObj.AddComponent<ARTrackedImage>();

        string prefabPath = "Assets/_ARMarker/ToInstanciate/" + PrefabsToSetParent[i].name + "Parent.prefab";
        GameObject newObjectParent = PrefabUtility.SaveAsPrefabAsset(EmptyObj, prefabPath);
        if (newObjectParent != null)
        {
           Debug.Log("Prefab creado");
        }
        else
        {
           Debug.Log("Error al crear prefab");
        }
      }
#endif
    }
}
```

Figura 52: SetObjectParent

- ***SetVideoParent*:** dicho *script* se encarga de transformar los vídeos para poder instanciarlos correctamente. En lo referido a las librerías, se mantienen las mismas que en el *script SetObjectParent* y se añade *using UnityEngine.Video*, la cual nos permite trabajar con vídeo.

En cuanto a la estructura, se define una lista de tipo *VideoClip* como variable pública, denominada *VideosToSetParent*, donde serán almacenados todos los vídeos que se desean convertir, y una textura en 2D que será necesaria aplicarle al material del vídeo (*Figura* 53).

```
public VideoClip[] VideosToSetParent;
private Texture2D pantallaCarga;
```

Figura 53: Declaración variables SetVideoParent

Al igual que con los objetos 3D, se recorre la lista *VideosToSetParent* con todos los videos y con cada uno se realiza las siguientes acciones.

Primero, se crea un nuevo material de tipo *Unlit>Texture* y se le adjunta la imagen *LoadingScreen* como textura, que se encuentra en un nuevo directorio que se crea llamado *Materials>LoadingScreen*, y que sirve para ser mostrada hasta que cargue el vídeo. Este nuevo material con textura se guarda en la carpeta *Materials*.

Figura 54: Pantalla de carga de vídeos

En segundo lugar, se crea un objeto de tipo *Quad* y se le aplica las coordenadas correspondientes ya mencionadas. Tras esto, se extrae su componente *Renderer* para añadirle el material.

Además, también se le añade la componente *VideoPlayer* que permite la reproducción de vídeo y audio. En su interior, se establece como clip a reproducir el elemento correspondiente de la lista *VideosToSetParent*, y se activa el campo *IsLooping* para volver a reproducir el vídeo en caso de que este finalice.

Una vez hecho esto, se guarda el vídeo con el material y las componentes añadidas en un nuevo directorio llamado *ToConvert>Mesh* para poder convertirlo en *prefab*. Cabe recordar que esta es una premisa para poder instanciar cualquier recurso correctamente, por lo que teniendo el *prefab* del vídeo generado, se puede destruir el *Quad* de la escena.

Los siguientes pasos son similares al proceso de la conversión de los objetos 3D. Se genera un *EmptyObject* con las coordenadas correspondientes y se conecta al *Prefab* del vídeo. A esto último, también se le conoce como: hacer el *prefab* hijo del *EmptyObject*.

Por último, siguiendo con otro de los requisitos, se añade la componente *ARTrackedImage* y se almacena este nuevo *GameObject* en la carpeta *ToInstanciate*, es decir, el vídeo ya estaría preparado para ser instanciado (*Figura* 55).

```
#if UNITY_EDITOR

        string imagePath = "Assets/_ARMarker/Materials/LoadingScreen/LoadingScreen.png";
        pantallaCarga = AssetDatabase.LoadAssetAtPath<Texture2D>(imagePath);


        for(int i = 0; i < VideosToSetParent.Length; i++)
        {
            Material unlitTextureMaterial = new Material (Shader.Find("Unlit/Texture"));
            unlitTextureMaterial.mainTexture = pantallaCarga;
            AssetDatabase.CreateAsset(unlitTextureMaterial, "Assets/_ARMarker/Materials/" + VideosToSetParent[i].name + "-UTMaterial.mat");

            GameObject quad = GameObject.CreatePrimitive(PrimitiveType.Quad);
            quad.transform.position = new Vector3(0,0,0);
            quad.transform.localScale = new Vector3(0.12f,0.12f,0.12f);
            quad.transform.rotation = Quaternion.Euler(90,0,0);

            Renderer quadRenderer = quad.GetComponent<Renderer>();
            quadRenderer.material = unlitTextureMaterial;

            VideoPlayer videoPlayer = quad.AddComponent<VideoPlayer>();
            videoPlayer.clip = VideosToSetParent[i];
            videoPlayer.isLooping = true;

            string prefabVideoPath = "Assets/_ARMarker/ToConvert/Mesh/" + VideosToSetParent[i].name + "Mesh.prefab";
            GameObject videoQuadMesh = PrefabUtility.SaveAsPrefabAsset(quad, prefabVideoPath);
            Destroy(quad);

            GameObject EmptyObj = new GameObject(VideosToSetParent[i] + "Parent");
            GameObject videoQuadMesh_2 = Instantiate(videoQuadMesh);
            videoQuadMesh_2.transform.parent = EmptyObj.gameObject.transform;
            EmptyObj.transform.position = new Vector3(0,0,1);

            ARTrackedImage trackedImageComponent = EmptyObj.AddComponent<ARTrackedImage>();
            string prefabVideoParentPath = "Assets/_ARMarker/ToInstanciate/" + VideosToSetParent[i].name + ".prefab";
            GameObject videoQuadParent = PrefabUtility.SaveAsPrefabAsset(EmptyObj, prefabVideoParentPath);

            videoQuadMesh_2.SetActive(false);
        }
#endif
    }
}

```

Figura 55: SetVideoParent

- **SetImageParent:** por último, se genera el *script SetImageParent*, el cual es el encargado de convertir las imágenes para poder ser instanciadas correctamente. La definición de su código se muestra únicamente por imagen, ya que se utilizan exactamente las mismas funciones y acciones que en el *script SetVideoParent*, pero sin incluir las características para vídeo (*Figura* 56).

```
#if UNITY_EDITOR
        for(int i = 0; i < ImagesToSetParent.Length; i++)
        {

            Material unlitTextureMaterial = new Material (Shader.Find("Unlit/Texture"));
            unlitTextureMaterial.mainTexture = ImagesToSetParent[i];
            AssetDatabase.CreateAsset(unlitTextureMaterial, "Assets/_ARMarker/Materials/" + ImagesToSetParent[i].name + "-UTMaterial.mat");

            GameObject quad = GameObject.CreatePrimitive(PrimitiveType.Quad);
            quad.transform.position = new Vector3(0,0,0);
            quad.transform.localScale = new Vector3(0.12f,0.12f,0.12f);
            quad.transform.rotation = Quaternion.Euler(90,0,0);

            Renderer quadRenderer = quad.GetComponent<Renderer>();
            quadRenderer.material = unlitTextureMaterial;

            string prefabImagePath = "Assets/_ARMarker/ToConvert/Mesh/" + ImagesToSetParent[i].name + "MESH.prefab";
            GameObject imageQuadMesh = PrefabUtility.SaveAsPrefabAsset(quad, prefabImagePath);
            Destroy(quad);

            GameObject EmptyObj = new GameObject(ImagesToSetParent[i] + "Parent");
            GameObject imageQuadMesh_2 = Instantiate(imageQuadMesh);
            imageQuadMesh_2.transform.parent = EmptyObj.gameObject.transform;
            EmptyObj.transform.position = new Vector3(0,0,1);

            ARTrackedImage trackedImageComponent = EmptyObj.AddComponent<ARTrackedImage>();

            string prefabImageParentPath = "Assets/_ARMarker/ToInstanciate/" + ImagesToSetParent[i].name + ".prefab";
            GameObject imageQuadParent = PrefabUtility.SaveAsPrefabAsset(EmptyObj, prefabImageParentPath);
        }
#endif
    }
}
```

Figura 56: Script SetImageParent

Para concluir con la explicación de la escena *Configuration*, estos son los tres *scripts* asociados al *GameObject Manager* encargados de convertir todos los recursos para que el usuario pueda instanciarlos correctamente.

Es importante denotar que las tres listas con los recursos correspondientes se han creado como variables públicas, de tal forma que para el usuario es sencillo seleccionar que recursos quiere convertir. Si se clica en el *GameObject Manager* en el inspector, se puede observar los campos pertinentes para ello, y tan solo se tiene que añadir los recursos deseados al que corresponda (*Figura* 57).

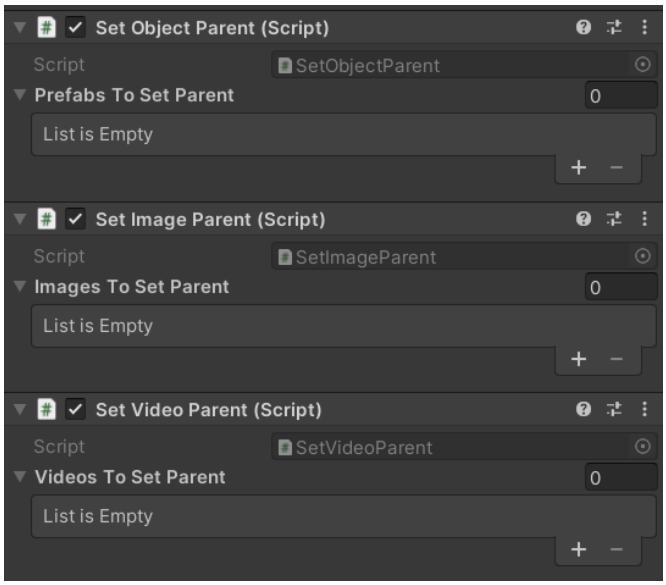


Figura 57: Listas pública de elementos a convertir

Por último, cuando el usuario ya tiene configurado que recursos convertir en las listas correspondientes, tan solo tiene que pulsar en el *Play* dentro de la escena y volver a pulsar en *Pause*, en caso de que la escena no se haya detenido (*Figura* 58). Simplemente con realizar esto, todos los recursos incorporados serán convertidos y almacenados en sus respectivas carpetas listos para configurar posteriormente para ser asociados e instanciados.

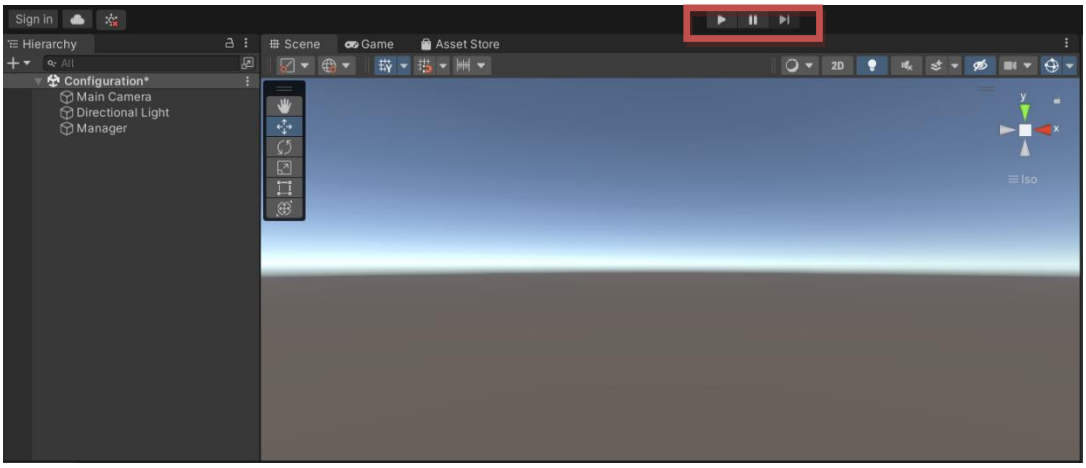


Figura 58: Botón Play

En este punto del proyecto, al haber incorporado otra escena, es necesaria una verificación más minuciosa por parte del desarrollador, lo cual implica la ejecución de varias pruebas aplicaciones de prueba en el dispositivo Xiaomi, con el fin de corregir defectos e implementar mejoras. De esta manera se consigue un camino eficiente y sencillo en el que realizar la conversión de elementos y posteriormente su asociación con marcadores y correcta instancia.

Sin embargo, surge un problema al probar con modelados 3D extraídos de páginas web externas, ya que no es posible acceder a los valores de las coordenadas de estos. Esto implica que en la mayor parte de las ocasiones, el objeto 3D tiene una escala demasiado grande o pequeña, y en la pantalla del teléfono es imposible de distinguir.

Tras varias pruebas e intentos de conseguir una solución, se ha llegado a la conclusión de que la forma más coherente para entender lo que sucede es crear una interfaz. En el siguiente apartado, quedan detalladas todas sus funciones sobre cómo se solventa este problema.

4.4 Interfaz del proyecto

Además de la creación de una nueva escena para adaptar el proyecto a las necesidades de un posible usuario final, y que así, a este le sea más sencillo personalizar su aplicación; dentro del proyecto, se ha creado una **interfaz** que ajusta un problema en cuanto a la instancia de modelados 3D obtenidos de sitios webs externos. Asimismo, esta interfaz ayuda al entendimiento por parte del usuario una vez dentro de la aplicación ejecutada en el teléfono móvil.

Por lo tanto, para la creación de la interfaz, hay que tener en cuenta cuál de las dos escenas se mostrará primero una vez se ejecute la aplicación. En este caso, como la escena *Configuration* forma parte del proceso de preparación y solo se utiliza dentro de Unity, se tiene que seleccionar como la escena activa *MultipleMarkers*. Para realizar esto, se abre la opción *File* desde la barra superior en Unity y se entra en los ajustes de *BuildSettings*. Una vez dentro, en el apartado *Scenes In Build*, si no aparece ninguna de las dos escenas se pulsa en *AddOpenScenes*, una vez se visualicen, se marca únicamente la escena *MultipleMarkers* (*Figura* 59).

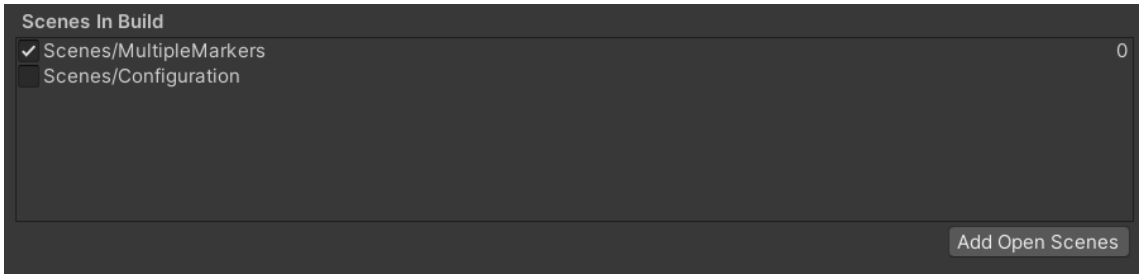


Figura 59:Scenes In Build

Una vez clara la escena que entra en juego al abrir la aplicación, ya se puede comenzar con la creación de la interfaz, que esta estará sobre la escena seleccionada *MultipleMarkers*.

Para lograr su creación, en el interior de la escena *MultipleMarkers* se clica el botón derecho y se selecciona *UI>Canvas*. UI hace referencia a *UserInterface* que significa interfaz de usuario, y *Canvas* es una capa transparente que se crea por encima de la escena a la que se pueden añadir diferentes elementos (*Figura* 60).

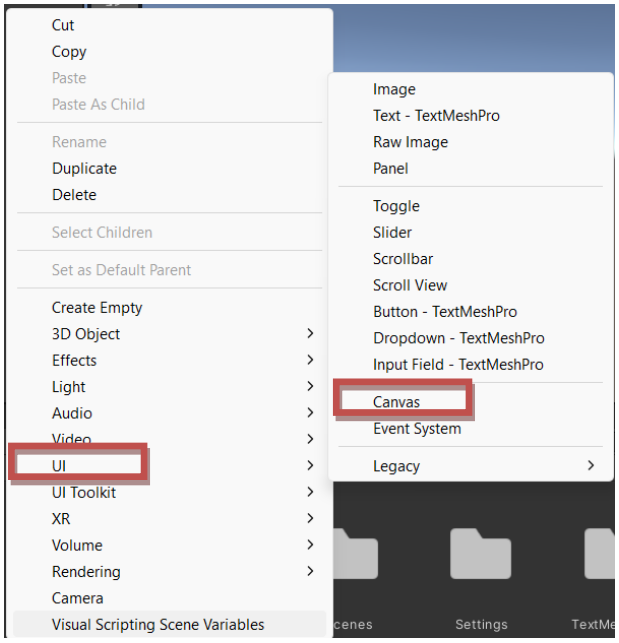


Figura 60: Creación Canvas UI

En este proyecto, para solucionar el problema mencionado anteriormente se añaden tres botones. Primero, un botón con el símbolo de interrogante (?) , el cual al pulsarlo mostrará información sobre cómo funciona dicha interfaz. Y después, dos botones con los símbolos más (+) y menos (-) que servirán para aumentar o disminuir la escala del objeto instanciado en caso de que este no se visualice correctamente.

Con este propósito, se clica en el *GameObject Canvas* y se crea en su interior tres objetos de tipo *Button-TextMeshPro* y otro de tipo *Panel*. Esto, se realiza clicando en el *Canvas*, seleccionando la opción *UI* y el tipo de objeto deseado (*Figura* 61).

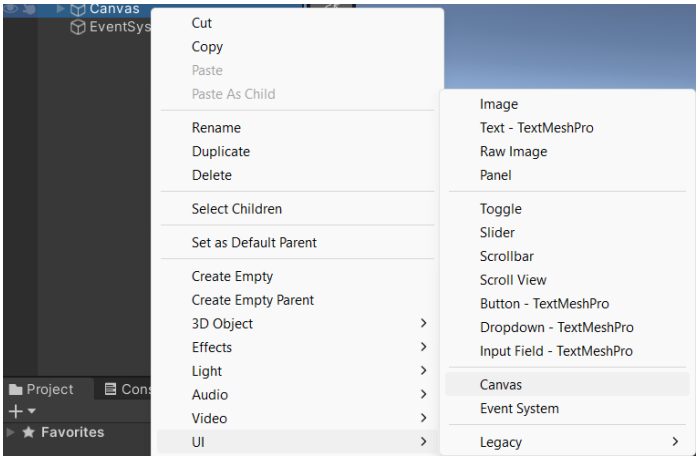


Figura 61: Creación Button-TextMeshPro y Panel

Una vez hecho esto, se denomina cada objeto, que en este caso se les ha llamado *ButtonDecrease* al botón para disminuir la escala, *ButtonIncrease* al de aumentarla, *ButtonInfo* al de mostrar el panel y el *Panel* se ha quedado con ese mismo nombre.

Sumado a esto, como los elementos son de tipo *TextMeshPro*, se genera automáticamente un elemento de texto conectado a estos objetos. Por otra parte, siempre que se genera una interfaz se crea automáticamente un *GameObject* llamado *EventSystem* el cual se encarga de la interacción del usuario con los elementos UI (*Figura* 62)

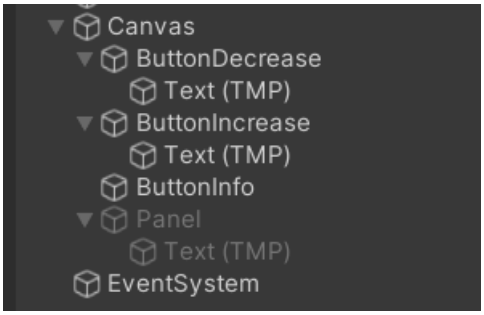


Figura 62: Elementos UI

Después de haber creado todos los elementos de la interfaz, es momento de introducir el código necesario para que dichos elementos realicen las funciones ya mencionadas. Como esta interfaz debe actuar durante todo momento a la vez que se está realizando el funcionamiento principal de la aplicación, que es detectar e instanciar, dicho código se incluye en el *script Partner*.

Para conseguir esto, en primer lugar, se realiza la declaración de las variables. *CurrentPrefab* hace referencia al *prefab* que está instanciado, *CurrentScale* a un valor óptimo de escala del *prefab* y *scaleStep* al valor del salto de escala que se producirá al pulsar el botón de más o menos. Seguidamente, se declaran los tres botones y el panel (*Figura* 63).

```
private GameObject currentPrefab;
private float currentScale = 1f;
private float scaleStep = 0.1f;

private Button buttonDecrease;
private Button buttonIncrease;
private Button buttonInfo;

public GameObject panel;
```

Figura 63: Declaración variables UI

Con las variables ya declaradas, es importante inicializar las variables de los botones obteniendo su componente *Button* y agregando un elemento de escucha de evento para cuando los botones sean clicados (*Figura* 64).

```
buttonDecrease = GameObject.Find("ButtonDecrease").GetComponent<Button>();
buttonIncrease = GameObject.Find("ButtonIncrease").GetComponent<Button>();
buttonInfo = GameObject.Find("ButtonInfo").GetComponent<Button>();

buttonDecrease.onClick.AddListener(DecreaseScale);
buttonIncrease.onClick.AddListener(IncreaseScale);
buttonInfo.onClick.AddListener(PanelInfo);
```

Figura 64: Inicialización UI Buttons

En cuanto a las funciones y métodos que ya constan en este *script* y tienen como función asociar e instanciar, tan solo hay que incorporar en el método de instanciar *InstantiateGameObject* la instrucción *currentPrefab= prefab;* Esta lo que hace es actualizar el valor del *prefab* instanciado de tal forma que cuando se desee aumentar o disminuir la escala siempre será la del instanciado en ese momento, evitando problemas.

No obstante, para cumplimentar el código con las acciones de los botones, sí que es necesario añadir un método por cada botón, que es la acción que realiza cada uno. Estos son:

- ***UpdateScale:*** es un método que establece como nueva escala del objeto instanciado, el resultado de multiplicar un vector unitario por el valor de escala *currentScale*, de tal forma que el resultado obtenido puede ser el aumento o la disminución de la escala del objeto (*Figura* 65).

```
private void UpdateScale()
{
    currentPrefab.transform.localScale = Vector3.one * currentScale;
}
```

Figura 65: Método UpdateScale

- ***IncreaseScale:*** es la función que se encarga de aumentar la escala del objeto instanciado aumentado el valor de *currentScale* por el valor de *scaleStep* (*Figura* 52).

```
private void IncreaseScale()
{
    currentScale += scaleStep;
    UpdateScale();
}
```

Figura 66: Método IncreaseScale

- ***DecreaseScale:*** tiene el funcionamiento contrario que la función *anterior,* pero se añade una instrucción para que, en caso de disminuir mucho la escala del objeto, este no se haga tan pequeño que no se pueda ver. (*Figura* 67).

```
private void DecreaseScale()
{
    currentScale -= scaleStep;
    currentScale = Mathf.Max(currentScale, scaleStep);
    UpdateScale();
}
```

Figura 67: Método DecreaseScale

- ***PanelInfo: e***s el último método del código, y se ocupa de mostrar el *Panel* cuando se pulsa desde el teléfono móvil al *ButtonInfo* (?) y de ocultarlo cuando se vuelve a pulsar (*Figura* 68).

```
private void PanelInfo()
{
    if (panel.activeInHierarchy == true)
    {
        panel.SetActive(false);
    }
    else
    {
        panel.SetActive(true);
    }
}
```

Figura 68: Método PanelInfo

Dentro de este *Panel*, se ha incluido un texto donde se explica que puede llegar a no verse el objeto instanciado debido a problemas de escala, y que la solución es disminuirla mediante los botones. También, están explicadas las funciones de estos botones encima de este texto explicativo sobre el problema con la escala (*Figura* 69).

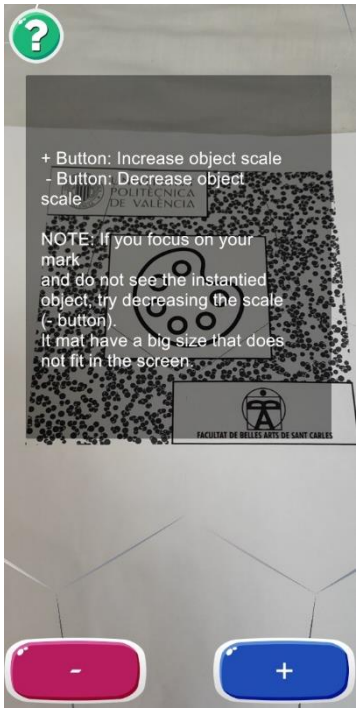


Figura 69: Visualización Panel

Habiendo implementado y configurado todas las funciones de la interfaz de usuario, es de nuevo necesario generar varios test con versiones de aplicaciones prueba para poder corregir errores, perfeccionar la visualización de los botones y sus funciones, para así completar la configuración de la estructura del proyecto.

Finalmente, se consigue un proyecto y aplicación final a la que poder dar un uso eficaz en el mundo de la docencia.

5. IMPLEMENTACIÓN DEL PROYECTO EN LA DOCENCIA

Teniendo conocimiento de la existencia de multitud de factores que influyen en la configuración exitosa del proyecto, con el objetivo de generar una aplicación con las preferencias del usuario, en este apartado se proporciona una descripción acerca de lo que hay que tener en cuenta y que pasos seguir para lograrlo.

5.1 Configuración de la aplicación por parte del usuario

Dado que para conseguir el desarrollo de la aplicación con estos últimos pasos se han creado numerosos directorios y nuevas carpetas, se va a mostrar mediante un esquema, como queda la estructura del proyecto para saber ubicar los diferentes recursos que se utilizan en él (*Figura* 70).

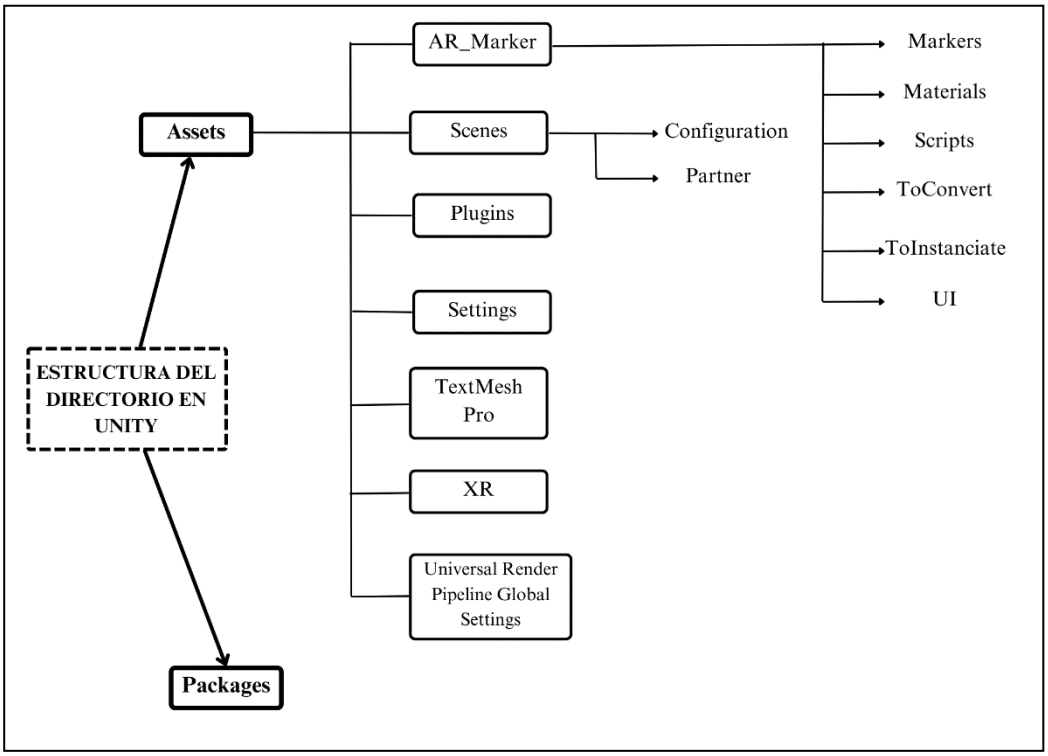


Figura 70: Esquema estructura directorio en Unity. Elaboración propia

En este esquema, se observan los directorios *Assets* y *Packages*, que son las carpetas raíces ya explicadas en la sección 4.3. Dentro de la carpeta *Assets*, existen diversas carpetas y elementos como *Universal Render Pipeline Global Settings*, *XR*, *TextMesh Pro*, *Settings* y *Plugins*, que se crean automáticamente al crear un proyecto AR en Unity y ayudan a realizar dichas funciones.

Por otro lado, *Scenes* es la carpeta donde se almacenan las dos escenas del proyecto y *_ARMarker* es la carpeta que alberga todos los recursos creados por y para el proyecto.

En esta última, se encuentran otras carpetas como *Markers*, donde están las imágenes de los marcadores y la librería de imágenes; *Materials*, donde se almacenan los materiales de los vídeos e imágenes creadas; s*cripts*, que contiene los *scripts* con los códigos; la carpeta *ToConvert*, donde el usuario deposita los elementos que se tienen que convertir para poder ser instanciados correctamente; la carpeta *ToInstanciate*, en la cual se guardan automáticamente los recursos listos para ser instanciados después de ejecutar la escena *Configuration*, y por último, la carpeta *UI*, que dispone de todos los elementos utilizados para la interfaz de usuario.

Conocida la estructura del proyecto, una explicación sencilla sobre cómo desarrollar el ejecutable de la aplicación sería la siguiente.
Primero preparar tanto los marcadores en el interior de la librería de imágenes como los recursos que se desea instanciar dentro de la carpeta *ToConvert*. Tras esto, abrir la escena *Configuration*, incluir los elementos de la carpeta *ToConvert* en el inspector del *GameObject Manager* y ejecutar la escena y pausarla. Una vez estén los recursos listos para instanciar, abrir la escena *MultipleMarkers* y asociar en el inspector *GameObject ARSessionOrigin* los recursos a sus marcadores en las listas correspondientes.

Dado que comprender este procedimiento a través de una descripción escrita puede dar lugar a confusión, se ha elaborado un vídeo adicional que muestra el mismo proceso. Este recurso se ha creado con el fin de facilitar la comprensión por parte del usuario acerca de cómo preparar el proyecto en Unity de manera efectiva, para que así pueda generar la aplicación sin dificultad.
https://drive.google.com/file/d/1K1JwgRXFeZXwuA3qxB0Q1CN2EqxA8c8x/view?usp=sharing

Como aspecto a comentar, antes de generar la aplicación también se puede establecer un nombre al proyecto, que viene a ser, el nombre de la aplicación. En este caso se le ha llamado *ScanXperience*. No se ha optado por buscar un nombre específico ya que se entiende que esto se ofrece a gusto del usuario que desee generar su propia aplicación. Para ello, debe ir a la opción *Edit* de la barra superior en Unity y seleccionar *Project Settings*. Aquí, puede introducir el nombre que desee proporcionarle a la aplicación en el campo *ProductName*, así como, el nombre del desarrollador o de la compañía (*Figura* 71).

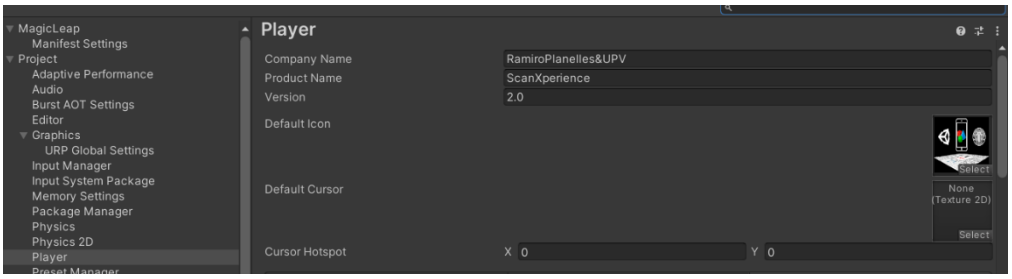


Figura 71: Nombrar desarrollador y aplicación

No obstante, lo que sí que se ha decidido establecer de manera general es el logo de la aplicación, para que tenga siempre la misma visibilidad. Este logo, se ha generado con el programa *Photoshop*, en base a los conocimientos del desarrollador, de la misma forma que los marcadores que se han usado en las versiones de prueba. (*Figura* 72)

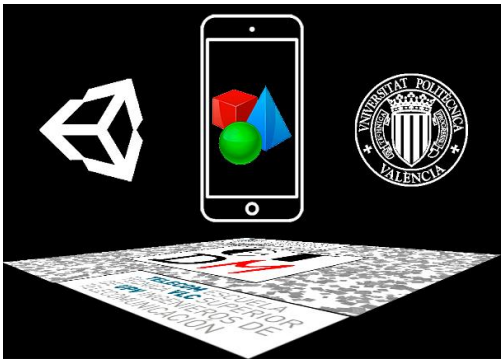


Figura 72: Logo aplicación ScanXperience

5.2 Compilación y generación del archivo ejecutable

Conociendo cómo configurar el proyecto en función de las preferencias del usuario, en este apartado se va a realizar la explicación sobre cómo generar el archivo ejecutable en el propio ordenador.

Para ello, se pulsa en los ajustes *File* de la barra superior en Unity y se selecciona la opción *BuildSettings*. Aquí, se comprueba las escenas seleccionadas y que la plataforma sea *Android*, como ya se ha explicado en anteriores apartados, y se escoge la opción *AllCompatibleDevices* en el campo *Run Device*. Tras esto, se pulsa en *Build* para generar el archivo ejecutable .apk y se define el nombre y la ubicación del archivo (*Figura* 73).

En este proyecto, para mantener la organización de directorios se ha creado una carpeta llamada *Builds* al nivel de las carpetas *Assets* y *Packages* donde almacenar todas las versiones de prueba y la definitiva, y tenerlas bien ubicadas.

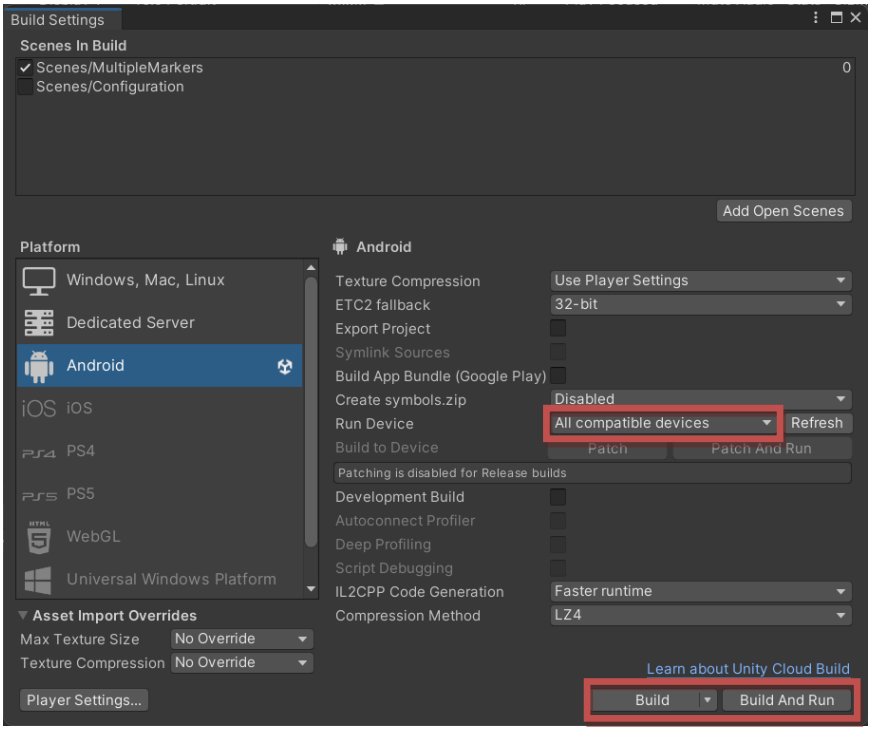


Figura 73: Generación archivo ejecutable .apk

Este procedimiento es el que el usuario debe seguir para generar su archivo de la aplicación y así tenerlo disponible para compartir mediante cualquier medio.

Sin embargo, en el caso del desarrollador del proyecto, para llevar a cabo las numerosas pruebas que se han tenido que hacer, este debe escoger la opción *BuildandRun* como se ve en la *Figura* 73.

Esta opción permite enviar y ejecutar directamente la aplicación en el teléfono móvil mediante el cable de tipo USB. Esto manera resulta más cómoda y rápida para verificar los diferentes pasos de creación del proyecto, evitando así la necesidad de compartir repetidamente la aplicación a través de otras aplicaciones.

5.3 Importación y ejecución de la aplicación en dispositivos móviles

En lo que respecta a la importación al teléfono móvil y ejecución de la aplicación, para el caso del desarrollador, ya se ha comentado que este proceso se realiza automáticamente. Pero para el caso en el que un usuario genere el archivo en el ordenador, este debe compartir el archivo mediante otro medio. El medio que se ha utilizado en este trabajo ha sido Google Drive. Tras generar la aplicación (archivo .apk) en el ordenador, se carga en Google Drive y después en el dispositivo se abre dicho aplicativo, se selecciona el archivo y se pulsa en descargar.

No obstante, una vez se obtenga el aplicativo (archivo .apk) en el teléfono móvil y se disponga a instalar, en ambos casos salta por pantalla un aviso referido a la instalación de aplicaciones desconocidas. Entonces, para poder instalar la aplicación, se debe aceptar en los ajustes de configuración del teléfono móvil en la sección *Privacidad>Gestionar>Permisos Especiales>Instalar aplicaciones desconocidas* (*Figura* 74), el permiso de instalar aplicaciones desconocidas de Google Drive en este caso (*Figura* 75). Cabe destacar que se debe seleccionar la aplicación de donde se ha compartido o descargado la aplicación, en caso de ser una página web, se deben aceptar los permisos del navegador utilizado.

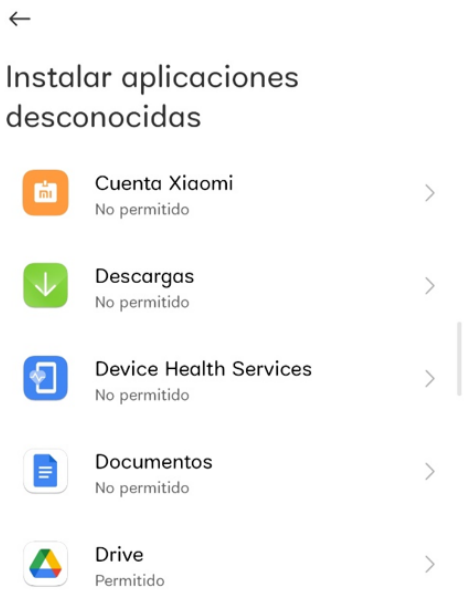


Figura 75: Ajustes instalar aplicaciones desconocidas

Figura 74: Autorización fuente de instalación de la aplicación

Con este ajuste habilitado, la aplicación se instalará correctamente y se podrá hacer uso de ella.

5.4 Pruebas de validación

La **validación** de una aplicación es el último proceso, pero no el menos importante, para garantizar que funciona de una manera óptima. En este apartado, se expone el proceso de validación del proyecto y la aplicación creadas, centrándose en los teléfonos móviles utilizados y el número de personas que ha configurado el proyecto con sus preferencias.

En cuanto a dispositivos móviles en los que se ha instalado la aplicación son: Xiaomi MI 11 5G NE, es el dispositivo principal en el que se han realizado todas las pruebas e instalado todas las versiones de la aplicación desde principio a fin; además, se ha probado su instalación en un Redmi Note 11 Lite 5G y un Samsung Galaxy A10 y se han obtenido resultados correctos. En ambos móviles se ha instalado la versión final creada por parte del desarrollador del proyecto.

Por otro lado, el proyecto de Unity se ha compartido con dos usuarios, un compañero del grado y los docentes encargados del proyecto, para que estos configurasen el proyecto con sus marcadores y recursos a instanciar deseados.

Por último, para facilitar la comprensión de la configuración e implementación del proyecto por parte de cualquier docente. Se encuentra en el *Anexo I*, un documento en formato .pdf con todos los recursos necesarios para este fin.

Gracias a todo el material de apoyo disponible que se ha proporcionado, como el presente documento, los dos vídeos que describen el proceso de configuración y compilación, y el archivo con el proyecto de Unity, estos dos usuarios han sido capaces de generar con sus preferencias una aplicación final óptima.

6. CONCLUSIONES Y PROPUESTA DE TRABAJO FUTURO

6.1 Limitaciones y posibles mejoras

Es un hecho que el desarrollo de un proyecto y una aplicación generados con Unity haciendo uso de la Realidad Aumentada es un proyecto entusiasmado a la par que emocionante. No obstante, como cualquier otro desafío de desarrollo de software o trabajo académico, esta iniciativa conlleva una serie de limitaciones que deben sobrepasarse de manera racional. En esta sección se habla sobre las limitaciones más características que han surgido a la hora de la creación del proyecto.

Como principal limitación a comentar y que define en mayor medida el desarrollo del proyecto, es la investigación y el uso de la librería *ARFoundation*. Dado que esta librería se introdujo en Unity por primera vez en 2018, esta no tiene un estudio totalmente desarrollado y el flujo de información tanto por parte del manual de Unity como de la investigación de excelentes desarrolladores de Unity expresada en foros populares, queda un tanto escasa. Por tanto, extraer información para investigar y probar diferentes funciones de AR en Unity se vuelve una tarea difícil.

Hilado con lo novedosa que puede llegar a ser la librería *ARFoundation*, en los primeros pasos del proyecto, se ha podido observar que al investigar sobre qué componente permite la instancia de recursos, en este caso *ARTrackedImageManager*, llama la atención que esta solo permite la instancia de un solo elemento asociado a un marcador, lo cual se puede considerar una limitación. Como solución se han implementado las funciones requeridas para aplicar la función de esa componente a diversos marcadores y recursos mediante el uso de un script.

Una adversidad más a comentar es la incorporación de objetos 3D extraídos de páginas webs externas, ya que en estos elementos no es posible acceder a sus coordenadas y la configuración de estas a unos valores en concreto es un requisito para poder instanciar correctamente dichos elementos. Realmente no se ha llegado a una solución total, sino que este hecho se ha corregido en tiempo de ejecución gracias a la implementación de la interfaz de usuario, pudiendo ajustar los valores de la escala de esta manera.

En resumen, se han identificado las principales limitaciones que han surgido durante el desarrollo de este proyecto y cómo se han solucionado. En cuanto al resto de contratiempos, se han ido mencionando durante la explicación del proyecto y todos ellos han sido resueltos gracias a los propios conocimientos, la ayuda de los docentes implicados en este trabajo o mediante investigación e indagación de manuales oficiales, foros y páginas webs.

6.2 Propuesta de trabajo futuro

Con respecto a una posible propuesta de trabajo ante este proyecto, sería la implementación de nuevas formas de interacción en la aplicación. Por ejemplo, se podría desarrollar un sistema de gestos que permita al usuario manipular los objetos instanciados con movimientos específicos como rotar el objeto, aumentar o disminuir la escala, arrastrarlo... O incluso, incorporar al sistema reconocimiento de voz y realizar estas acciones mediante mensajes de voz del usuario.

Otra propuesta de mejora interesante seria conseguir simplificar y hacer más entendible la personalización de la aplicación por parte del usuario. Esto se podría incluir adaptando la apariencia de ambas escenas o también, una opción bastante llamativa podría ser la configuración y selección de todos los elementos que se hacen uso durante el proyecto en tiempo real, de tal forma que se evitaría la necesidad de presentar conocimientos técnicos profundos en la herramienta Unity.

Es cierto que tanto la Realidad Aumentada como la librería *ARFoundation* pertenecen a un mundo del que queda mucho por descubrir e investigar, por lo que las mejoras comentadas podrían darse en caso de disponer el tiempo y los recursos necesarios para ello.

6.3 Conclusiones del trabajo realizado

En este capítulo del proyecto, se definen las conclusiones que se han obtenido tras completar el desarrollo de la aplicación, para de esta manera, comprobar si los objetivos mencionados a principio de este documento se han completado y que más conclusiones se pueden obtener.

Así pues, como conclusión general del presente trabajo de fin de grado, se puede afirmar que se ha conseguido la creación de un proyecto base que genere una aplicación de Realidad Aumentada basada en marcadores y que permite la instancia de objetos 3D, imágenes y vídeos al detectar estos.

Otra de las conclusiones que se puede obtener es acerca de la librería *ARFoundation*, que a pesar de ser notablemente actual y proporcionar ciertas limitaciones, esta ha conseguido despertar al desarrollador interés para indagar e investigar acerca de cómo utilizar sus funciones e implementaciones para conseguir el objetivo principal del proyecto.

Finalmente, mediante la incorporación de una nueva escena adicional al proyecto, el diseño de una interfaz de usuario y la aportación de contenido de apoyo, como videotutoriales y guías explicativas a los y las docentes, se ha conseguido facilitar su aprendizaje y uso del proyecto para poder generar aplicaciones personalizadas en base a sus preferencias de la manera más eficaz y accesible posible.

7. BIBLIOGRAFÍA

1. DE LA TECNICA A LA TECHNE [Internet]. [citado 11 de julio de 2023]. Disponible en: http://serbal.pntic.mec.es/~cmunoz11/techne.html

2. Reflexiones sobre la tecnología educativa del futuro [Internet]. [citado 11 de julio de 2023]. Disponible en: https://docenciaydidactica.ecobachillerato.com/2020/06/reflexiones-sobre-la-tecnologia.html

3. Reflexión sobre ” El impacto de la Tecnología en la Educación”. | Sitio Personal Ricardo Fernandez [Internet]. 2013 [citado 11 de julio de 2023]. Disponible en: https://blog.uclm.es/ricardofdez/2013/04/04/reflexion-sobre-el-impacto-de-la-tecnologia-en-la-educacion/

4. Moran M. Educación [Internet]. Desarrollo Sostenible. [citado 4 de septiembre de 2023]. Disponible en: https://www.un.org/sustainabledevelopment/es/education/

5. Realidad aumentada. En: Wikipedia, la enciclopedia libre [Internet]. 2023 [citado 11 de julio de 2023]. Disponible en: https://es.wikipedia.org/w/index.php?title=Realidad_aumentada&oldid=152277469

6. Goff E, Mulvey KL, Irvin M, Hartstone-Rose A. Applications of Augmented Reality in Informal Science Learning Sites: a Review. J Sci Educ Technol. 1 de octubre de 2018;27.

7. Redaccion N. REALIDAD AUMENTADA. PARTE 1. - EOB - Derecho de los e-sport y los videojuegos [Internet]. 2022 [citado 11 de julio de 2023]. Disponible en: https://videojuegos.enriqueortegaburgos.com/realidad-aumentada-parte-1/

8. Corporation M. ¿Qué es la realidad aumentada (AR)? | Microsoft Dynamics 365 [Internet]. [citado 11 de julio de 2023]. Disponible en: https://dynamics.microsoft.com/es-es/mixed-reality/guides/what-is-augmented-reality-ar/

9. Aplicaciones de realidad aumentada: usos y ventajas | Tokio [Internet]. [citado 11 de julio de 2023]. Disponible en: https://www.tokioschool.com/noticias/aplicaciones-realidad-aumentada/

10. REALIDAD AUMENTADA EN LA EDUCACION | Web Oficial EUROINNOVA [Internet]. [citado 11 de julio de 2023]. Disponible en: https://www.euroinnova.edu.es/blog/realidad-aumentada-en-la-educacion

11. Soriano-Sánchez JG, Jiménez-Vázquez D. Las ventajas del uso de la realidad aumentada como recurso docente pedagógico. Rev Innova Educ. 2 de febrero de 2023;5(2):7-28.

12. Blázquez Sevilla A. Realidad Aumentada en Educación.

13. Unity [Internet]. 2022 [citado 12 de julio de 2023]. Unity: ¿Qué es y cómo funciona? Disponible en: https://support.unity.com/hc/es/articles/7642130833812-Unity-Qu%C3%A9-es-y-c%C3%B3mo-funciona-

14. Technologies U. Software potente 2D, 3D, VR y AR para desarrollo de juegos y aplicaciones móviles compatibles con diferentes plataformas. [Internet]. [citado 12 de julio de 2023]. Disponible en: https://unity.com/es/pricing

15. TFG-I-1939.pdf [Internet]. [citado 12 de julio de 2023]. Disponible en: https://uvadoc.uva.es/bitstream/handle/10324/47964/TFG-I-1939.pdf?sequence=1&isAllowed=y

16. AR Foundation | AR Foundation | 5.0.6 [Internet]. [citado 12 de julio de 2023]. Disponible en: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@5.0/manual/index.html

17. fhernandezp1. ¿Que es AR Foundation? [Internet]. Portal de noticias de tecnología, Realidad Virtual, Aumentada y Mixta, Videojuegos. 2023 [citado 12 de julio de 2023]. Disponible en: https://niixer.com/index.php/2023/05/22/que-es-ar-foundation/

18. Technologies U. Descarga el Unity Hub para comenzar tus proyectos creativos | Unity [Internet]. [citado 22 de agosto de 2023]. Disponible en: https://unity.com/es/download

19. Google for Developers [Internet]. [citado 22 de agosto de 2023]. Dispositivos compatibles con ARCore. Disponible en: https://developers.google.com/ar/devices?hl=es-419

20. FAQ: ¿Qué significa LTS? | Ubuntu Fácil [Internet]. [citado 28 de agosto de 2023]. Disponible en: http://www.ubuntufacil.com/2014/03/faq-que-significa-lts/

21. Technologies U. Unity - Manual: Unity User Manual 2022.3 (LTS) [Internet]. [citado 30 de agosto de 2023]. Disponible en: https://docs.unity3d.com/Manual/

22. Unity Learn [Internet]. [citado 30 de agosto de 2023]. ¿Cómo configurar el entorno de desarrollo de AR? Disponible en: https://learn.unity.com/tutorial/como-configurar-el-entorno-de-desarrollo-de-ar

23. Class ARSession | Package Manager UI website [Internet]. [citado 30 de agosto de 2023]. Disponible en: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@1.0/api/UnityEngine.XR.ARFoundation.ARSession.html

24. Clase ARInputManager | Fundación AR | 2.2.0-vista previa.6 [Internet]. [citado 30 de agosto de 2023]. Disponible en: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@2.2/api/UnityEngine.XR.ARFoundation.ARInputManager.html

25. Class ARSessionOrigin | Package Manager UI website [Internet]. [citado 30 de agosto de 2023]. Disponible en: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@1.0/api/UnityEngine.XR.ARFoundation.ARSessionOrigin.html

26. AR Tracked Image Manager | AR Foundation | 2.2.0-preview.6 [Internet]. [citado 30 de agosto de 2023]. Disponible en: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@2.2/manual/tracked-image-manager.html

27. Unity Learn [Internet]. [citado 31 de agosto de 2023]. ¿Cómo generar un objeto sobre un marcador en AR? Disponible en: https://learn.unity.com/tutorial/como-generar-un-objeto-sobre-un-marcador-en-ar

28. Administrador de imágenes con seguimiento AR | Fundación AR | 4.0.12 [Internet]. [citado 31 de agosto de 2023]. Disponible en: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.0/manual/tracked-image-manager.html

ANEXOS

ANEXO I:

MATERIAL DE APOYO PARA LA CONFIGURACIÓN DE LA INTERFAZ EN UNITY

Autor: Ramiro Planelles Lorente

Proyecto: Diseño y desarrollo de una aplicación para creación de contenido docente de realidad aumentada basada en marcas

Tutor/a: Rey Solaz, Beatriz **Cotutor/a:** Monzó Ferrer, Jose María

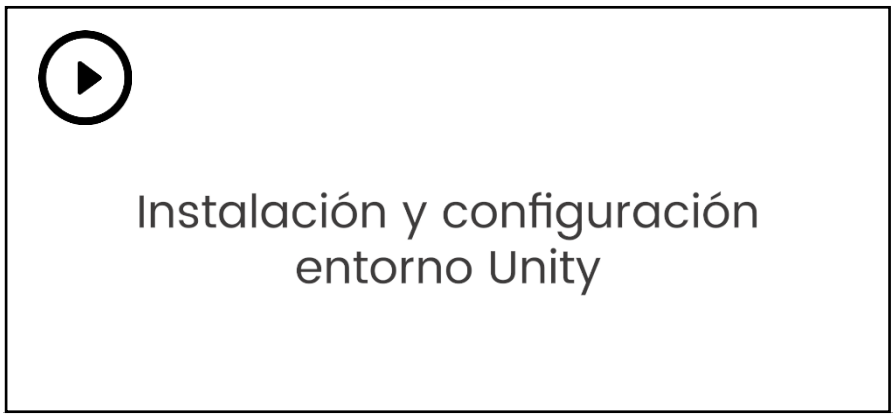
Este documento adicional se ha creado con el fin de aportar una ayuda extra, para que los y las docentes o cualquier tipo de usuario que tenga interés en hacer uso de este proyecto de Realidad Aumentada basado en marcas en Unity, pueda realizar las configuraciones pertinentes en base a sus gustos y preferencias.

En dicho documento, se recopila una amplia gama de materiales de ayuda y orientación, diseñados para guiar a los posibles desarrolladores desde los primeros pasos de la descarga de aplicativos y configuraciones necesarias, hasta la creación de la experiencia interactiva de Realidad Aumentada haciendo uso de marcadores y diferentes recursos digitales (objetos 3D, imágenes y vídeos).
De forma que, ya seas un principiante en el mundo de Unity o un veterano que desea descubrir nuevos conceptos, puedas desarrollar tu propia aplicación aplicando tus propias preferencias y elementos virtuales.

En primer lugar, se adjunta el proyecto base sin ningún tipo de recurso añadido, para que puedas descargártelo y comenzar a trabajar sobre él desde cero con mayor facilidad, siguiendo las guías y consejos que se muestran en este documento.

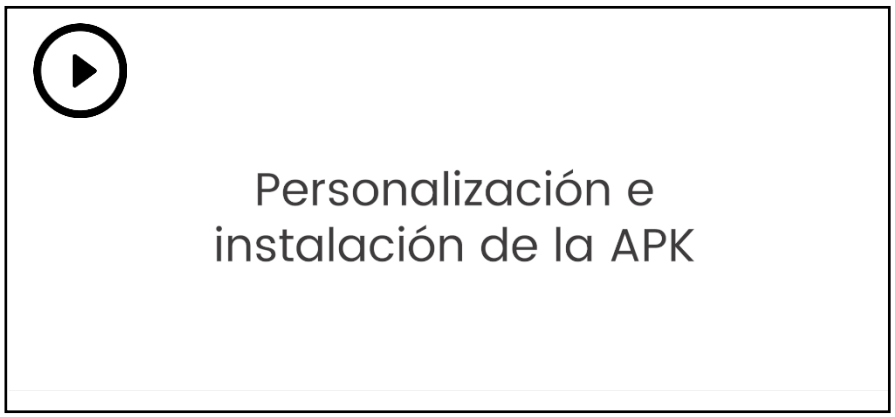
https://drive.google.com/file/d/1ZskS0_O9lNhFqF08vDfiTPH_7Ut7KWli/view?usp=sharing

Así pues, para que puedas hacer uso de este proyecto y disfrutar de todas las funciones, debes tener instalado el programa *Unity Hub* y la librería *AR Foundation*. Si no lo tienes instalado todavía o tiene dudas sobre cómo hacerlo, aquí tienes un vídeo donde se explica paso a paso.



https://drive.google.com/file/d/1WcO97Z4tzqvY2zhLGr84TRecAHw8Lx4f/view?usp=drive_link

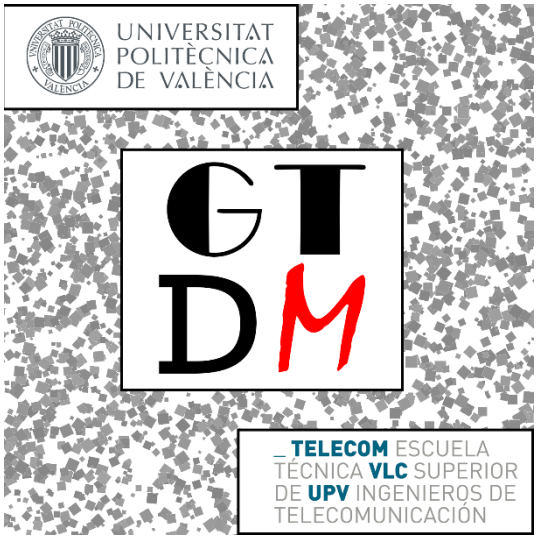
Si ya tienes instalado *Unity Hub*, has abierto el proyecto base ofrecido en Unity y has comprobado que la librería *AR Foundation* está aplicada, ya puedes configurar los elementos que deseas instanciar junto con sus marcadores asociados. Para que consigas una configuración correcta y te asegures de que tu aplicación funciona correctamente, te recomendamos observar el siguiente vídeo y seguir los pasos tal y como se muestra.



https://drive.google.com/file/d/1K1JwgRXFeZXwuA3qxB0Q1CN2EqxA8c8x/view?usp=drive_link

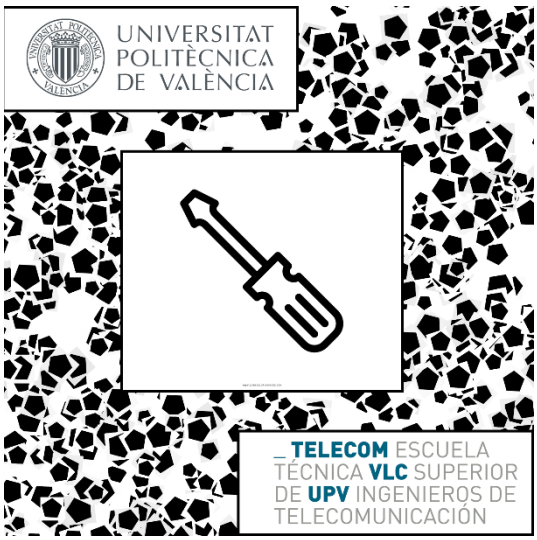
A continuación, se dejan ejemplificados los marcadores que se han dado uso para realizar las diferentes pruebas del desarrollo del proyecto. Estos marcadores han sido generados por el propio desarrollador del proyecto gracias a la aplicación Photoshop. Es importante tener en cuenta, que para que tus marcadores sean escaneables con la aplicación generada, deben cumplir los siguientes requisitos:

- La resolución de los marcadores debe ser mínimo de 300x300 píxeles y no necesariamente tienen que ser cuadrados.
- La imagen del marcador tiene que ser en formato PNG o JGP.
- La imagen que representa al marcador debe tener un contraste significativo.



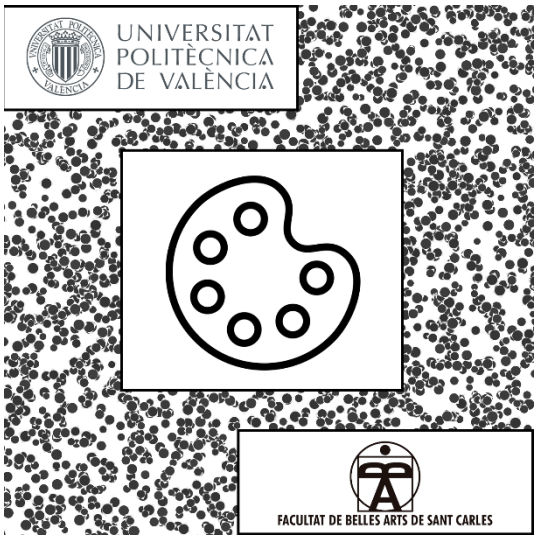
Marcador del Grado en Tecnología Digital y Multimedia (GTDM) asociado a un vídeo.

https://drive.google.com/file/d/16XH3m0cUY9qtWvnvyHHB5AHKHdruHNJ5/view?usp=sharing



Marcador del Grado en Ingeniería de Telecomunicaciones asociado a una imagen.

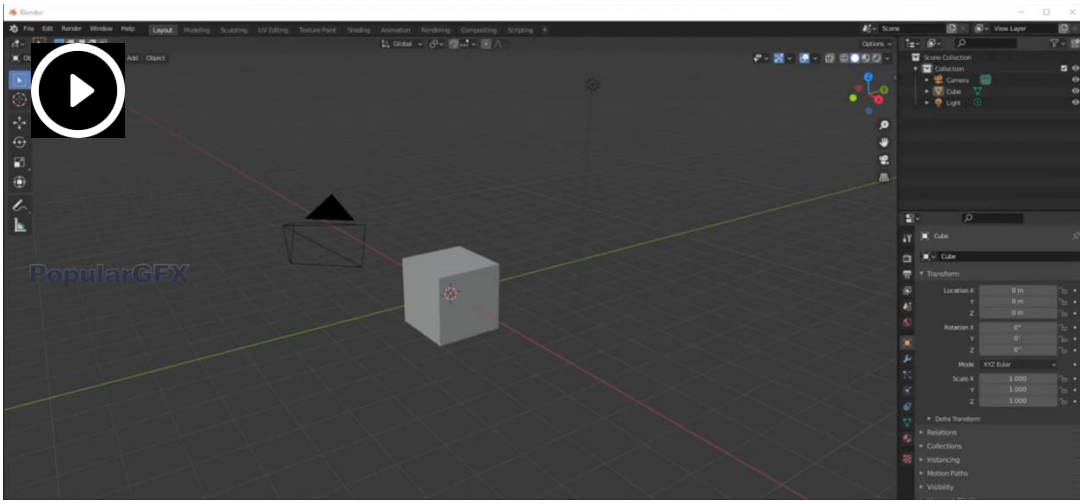
https://drive.google.com/file/d/1fQBAXUvZGEDre85r-uSyAEnovxO-Qk9P/view?usp=sharing



Marcadores del Grado en Bellas Artes asociado a un objeto 3D.

https://drive.google.com/file/d/11oHOCmr76sbacRyooX4WBToRmTLFLkmW/view?usp=sharing

Por otro lado, se adjuntan seguidamente los recursos digitales (objeto 3D, imagen y vídeo) que se han usado como instancias a la hora de detectar los marcadores mostrados anteriormente, para que puedas ver de donde se han extraído que características presentan.



Vídeo extraído de YouTube empleado para el marcador GTDM.

https://drive.google.com/file/d/15gIuzRGS7edbn3zDhQ8Iil7XV0RSGjJV/view?usp=sharing

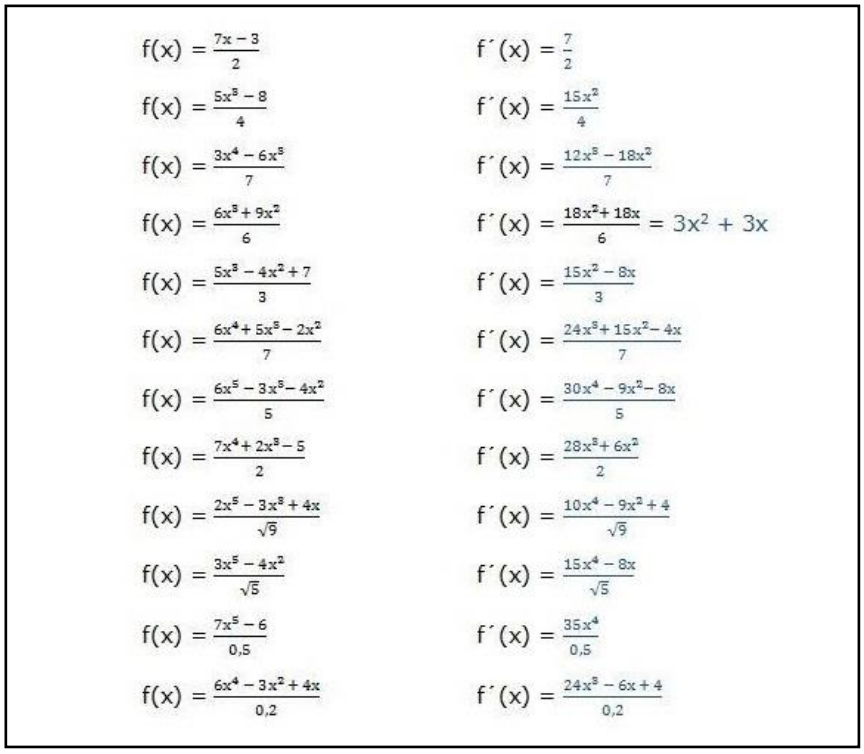


Imagen extraída de *Google Imágenes* utilizada para el marcador Telecomunicaciones.

https://drive.google.com/file/d/1N-1MStxv6byIFZDKIH34eltYRXiiPenI/view?usp=sharing

Objeto 3D extraído de la página web *Mixamo* empleado para el marcador en Bellas Artes

https://drive.google.com/file/d/1hgWUs0It_REXTq9PsoueRQzEg9ZV6_iK/view?usp=drive_link

Para concluir, se adjunta el link de la carpeta que contiene todos los recursos mencionados en el presente documento, por si deseas acceder a todos ellos a la vez.

https://drive.google.com/drive/folders/17wm8vNxpnuCIzowAZJV5esmUkcYIwXl-?usp=sharing