UNRaf UNIVERSIDAD NACIONAL DE RAFAELA

Licenciatura en Producción de Videojuegos y Entretenimiento Digital

MÉTODO DE DESARROLLO DE UN PROTOTIPO DE REALIDAD AUMENTADA EN UNITY3D EN DISPOSITIVOS ANDROID 11

Trabajo Final de Carrera - Diseño de prototipo

Acuña, Tomás

Jorge, Álvaro

Rafaela, Santa Fe - 23 de Mayo de 2025

Resumen

En este trabajo se propone un método para el desarrollo de aplicaciones que hagan uso de la realidad aumentada para los dispositivos Android 11 para el cual se realizó una investigación, búsqueda y recopilación de recursos teóricos para su creación. Dicho método fue ejecutado para el desarrollo de una aplicación la cual es usada como caso de prueba y evidencia para el método propuesto, el cual se utiliza para demostrar la funcionalidad y posibilidades que brinda la tecnología de realidad aumentada a la producción de aplicaciones para plataformas móviles.

Para ello, hemos realizado una comparación entre las definiciones de nuestras palabras claves elegidas desde el punto de vista de diversos autores, logrando diferenciar las primeras concepciones de estos conceptos relevando las más actuales y tomándolas de referencia.

También se realizó una observación de distintas aplicaciones con Realidad Aumentada centrada en las funciones de las mismas, las cuales fueron listadas en un cuadro comparativo, el cual fue usado para determinar cuáles de estas funciones fueron incluidas en nuestro plan de desarrollo.

Podemos decir que la aplicación desarrollada es un producto final que funciona como una demostración de las capacidades de la tecnología, como resultado y/o consecuencia de seguir el método que hemos diseñado a través del cumplimiento de nuestros objetivos específicos.

Abstract

This paper proposes a method for the development of applications that make use of augmented reality for Android 11 devices for which research, search and collection of theoretical resources for its creation was conducted. This method will be executed for the development of an application which will be used as a test case and evidence for the proposed method, which will be used to demonstrate the functionality and possibilities offered by augmented reality technology to the production of applications for mobile platforms.

To that end, we have done a comparison between the definitions of our keywords from the point of view of different authors, managing to differentiate the first conceptions of these concepts in order to highlight the most current ones and take them as a reference.

We also conducted an observation of different Augmented Reality applications, focusing on their functions and listing them in a comparative table, in order to determine which of these functions would be included in our development plan.

We can say that the developed app is a final product that serves as a demonstration of this technology's capabilities as a result of following our proposed method through the completion of our specific objectives.

PALABRAS CLAVE: Unity3D, Realidad Aumentada, Aplicación, Android 11.

Índice

Índice de Ilustraciones

1. Introducción

La Realidad Aumentada (de ahora en más referido por el acrónimo “AR”) es uno de los campos emergentes con mayor potencial para transformar y revolucionar el mundo del entretenimiento y el laboral, ofreciendo una nueva forma al usuario de relacionarse con el entorno y los procesos de producción. Como expresan en la revista Harvard Business Review los señores Michael E. Porter y James E. Heppelmann (2017), la AR mejora la eficiencia en la cadena de valor en el desarrollo de productos, fabricación, marketing, servicios, y otras numerosas áreas.

Entre los ejemplos de uso de AR podemos encontrar el de la empresa Nintendo, la cual publicó en 2016 el videojuego Pokemon GO (desarrollado por Niantic Labs), en el que se utilizaba AR para ver a los personajes a través de la cámara del dispositivo, siendo este el mayor ejemplo de videojuegos que utilizan esta tecnología, superando las 678 millones de descargas y posicionándose entre los 20 videojuegos más descargados en Android y iOS (Pérez, 2023). Otro ejemplo de uso de esta tecnología es el de la empresa IKEA, la cual se dedica a la venta de muebles para el hogar, presentó una aplicación llamada IKEA Place, que permite a los clientes previsualizar en tiempo real los muebles en el espacio de su casa donde deseen tenerlos, y de esta manera corroborar qué artículos comprar.

Si bien, el motor Unity3D nació como un entorno de desarrollo que proporciona herramientas para la creación de videojuegos, en la actualidad se lo utiliza para diversos sectores, como es el caso de la arquitectura con el aeropuerto de vancouver, en el que se generó un gemelo digital dentro de Unity3D para mejorar las operaciones y la sostenibilidad. También, Unity3D ofrece varias herramientas para animación y diseño.

Esta tecnología es un elemento creciente en los distintos campos laborales, incorporándose en la medicina, videojuegos, turismo, diseño, etc. A pesar de esto, el desarrollo de aplicaciones con AR no es un campo tan explorado y por ende puede presentar cierta cantidad de problemas o inconvenientes respecto a optimización o estabilidad de la aplicación.

2. Problema

Luego de indagar en diferentes fuentes y foros de internet, hemos notado una falta de manuales/guías concretas que sigan un desarrollo de una aplicación AR en Unity3D desde el principio hasta el final.

A pesar de la abundancia de guías o tutoriales sobre funciones específicas como mostrar un objeto o detectar una imagen, hay una percibida carencia de un tutorial que abarque la totalidad de un desarrollo.

Esto puede resultar en una pérdida de tiempo significante, porque cada una de estas asume una base o proceso de inicialización diferente, como el uso de una versión específica del motor de desarrollo, la instalación de programas o complementos externos (ej. Vuforia), o requiere el seguimiento de un tutorial anterior dependiendo de la complejidad de la solución que propone la guía.

Como ejemplo de esto, la guía encontrada en la página oficial de Unity3D, a pesar de ser una de las fuentes más completas, también padece de una parte de los problemas establecidos anteriormente, particularmente el uso de un complemento externo conocido como Script Graph el cual propone una forma de desarrollar aplicaciones usando Visual Scripting (Programación visual, la cual reemplaza la escritura de comandos en líneas de código por un sistema en el que el usuario ensambla la aplicación de forma gráfica usando bloques y nodos), recurso que no es utilizado en otras guías.

Es por lo escrito anteriormente que consideramos necesario recopilar, analizar y unificar dicho conocimiento para que este se encuentre disponible sin necesidad de estar buscando múltiples guías y tutoriales y que su seguimiento resulte en el desarrollo de una aplicación que posea las características necesarias para el uso efectivo de la AR y sus funciones aplicables a cualquier rubro, con el objetivo de que futuros desarrolladores de nivel intermedio tengan a su disposición este antecedente como plantilla y/o tutorial paso a paso para producir aplicaciones en Unity3D utilizando AR.

La plataforma objetivo del desarrollo será dispositivos Android 11, ya que este es el sistema operativo móvil más utilizado en el mundo, en 2022, Android tenía aproximadamente el 86.2% del mercado global de smartphones (Beckman, 2023). Esto significa que desarrollar para dispositivos Android le permitirá a los desarrolladores alcanzar una audiencia a nivel global. Además, este sistema operativo es conocido por su flexibilidad, habilitando a los desarrolladores a personalizar aplicaciones y acceder a funcionalidades avanzadas del hardware del dispositivos. Esto es menos restringido en comparación con otros sistemas operativos, como iOS (Beckman, 2023). Y posee una integración estrecha con los servicios

de Google, como Google Maps, Google Drive y Google Assistant, mejorando las posibilidades y compatibilidad de la aplicación (StatCounter, 2024).

En cuanto al motor de desarrollo, se decidió utilizar Unity3D ya que su editor visual permite arrastrar y soltar componentes, lo que facilita la creación de entornos complejos sin necesidad de un profundo conocimiento de programación (“Editor visual” no refiere a lo mismo que “Visual Scripting”). Otro de los puntos tomados en cuenta para esta decisión fue el soporte para AR y Virtual, Unity3D es una de las plataformas líderes para el desarrollo de aplicaciones AR y VR, proporcionando herramientas integradas y soporte para dispositivos que utilizan esta tecnología, lo que facilita la creación de experiencias inmersivas. También, posee capacidades de optimización que permiten ajustar el rendimiento según las necesidades de cada proyecto. (Unity Technologies, s.f)

3. Objetivos

A partir de lo escrito anteriormente, nuestro objetivo general es diseñar un método para el desarrollo de un prototipo que incluya las funciones básicas de AR del motor Unity3D para dispositivos Android 11.

Con esto establecido, nuestros objetivos específicos serán:

- Relevar e identificar las funciones básicas de AR en dispositivos Android.
- Proponer las funciones de AR que se incorporan al prototipo basados en la información obtenida.
- Desarrollar y documentar el desarrollo del prototipo con un mínimo de 5 funciones de AR.

4. Estado del arte

En lo que respecta a autores, podemos destacar 2 obras, siendo estas “Handbook of Augmented Reality” (2011), editado por Borko Furht, y "Understanding Augmented Reality: Concepts and Applications" (2013), escrito por Alan B. Craig. En ambos escritos, se explora la AR desde su definición hasta sus diversas posibilidades de aplicación en varios campos laborales.

Ya que este tema no fue trabajado en muchos más libros de carácter académico o la información en estos se encuentra desactualizada por su fecha de publicación, mencionamos

también un artículo un poco más actual de la revista Harvard Business Review en su edición Noviembre - Diciembre de 2017, redactado por Michael E. Porter y James E. Heppelmann, titulado "Por qué Todas las Organizaciones Necesitan una Estrategia de Realidad Aumentada", en donde buscan explicar y predecir la aplicabilidad de tecnologías de AR en distintas industrias y los posibles beneficios tanto en costos como efectividad.

En cuanto a guías ya existentes, una de las que consideramos más completas es el tutorial oficial brindado por la página de cursos de Unity Technologies, "Mobile AR Development", publicada en el año 2022, que abarcan varias funcionalidades y características, pero hace uso de Visual Scripting, herramienta que tiende a ser usada para enseñar a programar en vez de producir aplicaciones completas. Además, se encuentra limitado a una versión específica del motor: 2021.3

Roca (2020) utiliza la herramienta Vuforia, la cual proporciona una forma efectiva de escanear superficies, pero suele estar altamente limitada en su versión base, requiriendo realizar una inversión para obtener la versión premium y desbloquear todas sus características. A su vez, esta guía solo se limita al desarrollo de aplicaciones capaces de utilizar AR con marcadores, ignorando todos los elementos utilizables para otros tipos de AR.

En cuanto a tutoriales completos, tenemos un video titulado *"Augmented Reality for Everyone - Full Course"* (Canal freeCodeCamp.org, 2022), el cual tiene una duración de 11 horas y media, conteniendo descripciones conceptuales de tanto programación como AR, utilizando la herramienta Vuforia. A pesar de ser el material más completo, su público objetivo son los principiantes absolutos, ya que previos a las 7 horas de video, el contenido refiere solo a definiciones y explicaciones de conceptos, incluso enseñando bases de programación. Una vez pasadas estas 7 horas de video, se enseña de forma completa el desarrollo de 3 prototipos con funciones comunes de las aplicaciones AR con marcadores y una sin marcadores, siendo esta última un seguimiento de cara, utilizado comúnmente por filtros de cámara.

5. Marco teórico

Anteriormente hemos resaltado el potencial de la AR para transformar y revolucionar el mundo del entretenimiento y el laboral, ofreciendo una nueva forma al usuario de

relacionarse con el entorno y los procesos de producción. A continuación presentamos la definición de este término según distintos autores.

Valderrama, Valderrama y Rivera (2014) definen la AR como el fruto de técnicas que procesan imágenes cuadro a cuadro unido a los dispositivos que sobrescriben la información que reciben de una porción específica del mundo real, mezclando contenidos virtuales con el mundo real.

La página Microsoft Dynamics 365 define la AR como una versión mejorada e interactiva de un entorno del mundo real que se logra a través de elementos visuales digitales, sonidos y otros estímulos sensoriales mediante tecnología holográfica.

Ajenjo Jurado (2022) la define como tecnología capaz de proyectar elementos virtuales en el entorno físico. Esta nueva información se proyecta sobre el mundo captado por la cámara del dispositivo en tiempo real, sin ningún tipo de procesado posterior.
Teniendo en cuenta la finalidad de este desarrollo, se tomará como referencia esta última definición dada por Álvaro Ajenjo Jurado.

Según Rafael Valencia García, un motor de desarrollo de videojuegos es un kit de software diseñado para la creación y desarrollo de videojuegos para diferentes dispositivos como ordenadores, dispositivos móviles y consolas (2016). Para el desarrollo de este proyecto y subsecuente informe, se utilizará el motor conocido como Unity3D, ya que este tiene incorporada la funcionalidad para producir aplicaciones con AR dentro del editor.

No se debe confundir la AR con la realidad virtual, que si bien ambas proyectan elementos virtuales, la primera los introduce a nuestro mundo, mientras que la otra busca reemplazarlo por uno virtual (Ajenjo Jurado, 2022).

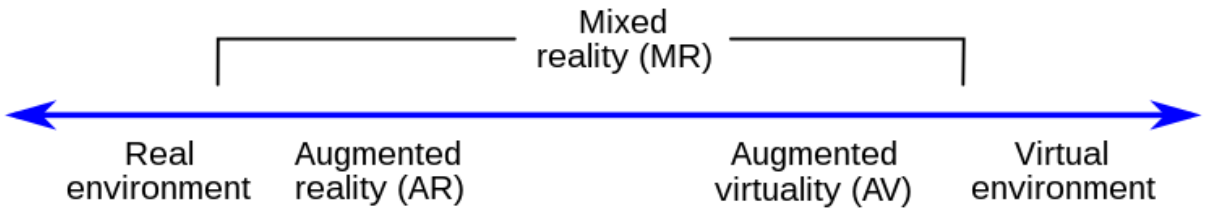


Ilustración 1. Representación gráfica de los diferentes campos de realidad. (Milgram. P, Kishino, F. 1994)

Aunque parezca lo contrario, la AR no es un concepto tan novedoso, ya que la primera tecnología a la que pudimos llamar de esta manera nació en 1968 en Harvard, cuando

el informático Ivan Sutherland, conocido como el “padre de la computación gráfica”, desarrolló La Espada de Damocles, el primer sistema de realidad mixta, afirma que la AR

> Servin (1968) afirmaba que la Espada de Damocles: “consistía en dos diminutos tubos de rayos catódicos, conectados a una computadora que representaba imágenes gráficas sobreimpresionadas en la escena real, gracias a un sistema de espejos. El dispositivo estaba suspendido del techo mediante un brazo mecánico, lo que permitía conocer la posición y orientación de la cabeza del usuario.” (p 4).

Más adelante, en 1993, Steven Feiner, Blair MacIntyre and Doree Seligmann presentaron KARMA (*Knowledge based Augmented Reality for Maintenance Assistance*), un banco de pruebas que explora y permite el diseño automatizado de realidades aumentadas para explicar labores de mantenimiento y reparación (Feiner, Macintyre, Seligman, 1993).

Yendo a un caso más actual, podemos encontrar el videojuego Pokemon GO lanzado para dispositivos móviles en el año 2016, que ofrece la posibilidad de traer los personajes del universo de Pokémon al mundo real, viendo los modelos 3D a través de la cámara del dispositivo. La jugabilidad consiste en activar la ubicación y en base a la posición del usuario, se encuentra a ciertas criaturas, incentivando a los jugadores a explorar sus ciudades en busca de Pokemones (Pineda, 2016).

Ilustración 2. Captura del juego mencionado, se puede ver el modelo 3D superpuesto en la captura de la cámara (Pokemon GO).

Existen dos tipos de AR bien definidos, y se los diferencia por su forma de implementación y funcionalidad. El primero es la AR con marcadores, que como dice su nombre, utiliza un marcador o indicador que puede estar presente de forma física o en una pantalla, al detectarse con la cámara, el objeto virtual se fijará en ese punto. El segundo tipo es la AR sin marcadores, la cual utiliza los siguientes recursos para detectar elementos; Visión Artificial, Localización simultánea y Mapeo (SLAM), Depth Tracking (Seguimiento de profundidad) (Ajenjo Jurado, 2022).

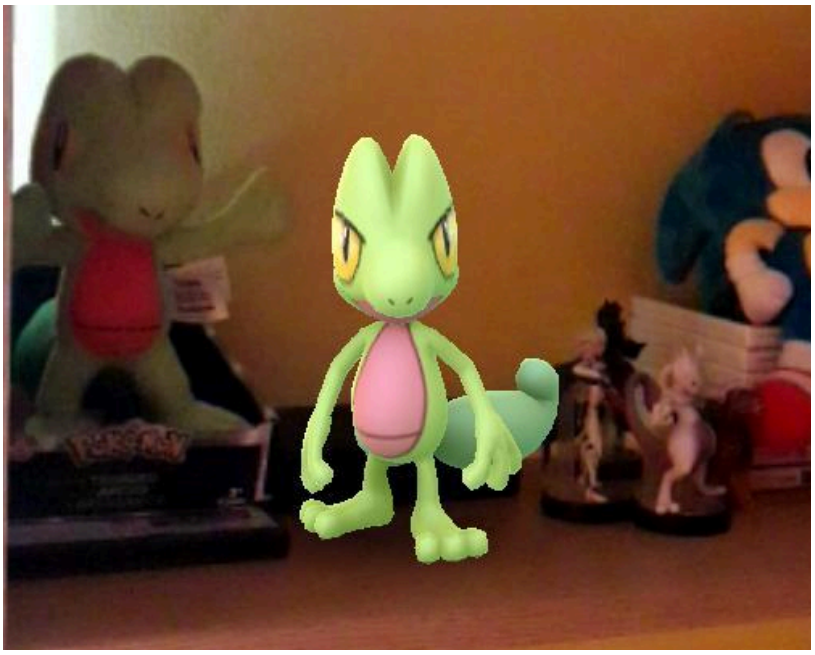


Ilustración 3. Proyección de modelo 3D en una superficie plana, nótese como este se encuentra alineado con el resto de los objetos en la mesa (Pokemon GO).

Ilustración 4. Proyección de modelo 3D sobre un fragmento de un libro, nótese cómo este se encuentra proyectado por encima de una imagen con un patrón único (Marketing Capacitación).

Para el desarrollo de nuestro prototipo se analizará una población de aplicaciones de AR, con el objetivo de señalar el uso de las siguientes funciones:

- AR con Marcadores utiliza un marcador o indicador que puede estar presente de forma física o en una pantalla, la posición de los objetos AR está limitada a la posición del marcador; la AR sin Marcadores, no presenta estas limitaciones ya que utiliza diversas tecnologías (mencionadas anteriormente) para ubicar los objetos AR.

- Visualización de Objetos 3D: Permite mostrar modelos tridimensionales (3D) superpuestos al entorno real mediante la cámara del dispositivo. El usuario puede ver objetos digitales como si estuvieran físicamente presentes en el mundo real.

- Rotación de Objetos 3D: Permite al usuario girar los modelos 3D sobre sus propios ejes (X, Y, Z) para observarlos desde distintos ángulos. Esta funcionalidad mejora la comprensión espacial del objeto, siendo especialmente útil en aplicaciones educativas, museográficas o de visualización de productos.

- Visualización 2D: Consiste en mostrar elementos planos como imágenes, textos o interfaces gráficas sobre los planos detectados, superponiendolas sobre objetos físicos en el entorno real.

- Detección de Planos: Es la capacidad de la aplicación de AR para identificar superficies planas en el entorno real, como mesas, pisos o paredes. Estos planos sirven como base para colocar objetos virtuales de forma realista y estable, facilitando la integración entre el contenido digital y el espacio físico.

- Escalado de Objetos 3D: Permite aumentar o reducir el tamaño de los objetos 3D dentro del entorno de AR, para ajustarlos a la escala real o al gusto del usuario.

Basándonos en nuestra experiencia, definimos estas funciones de AR como las más comunes en aplicaciones móviles que utilizan esta tecnología por lo que serán tomadas como objetivo para nuestro relevamiento y así identificar las funciones que formarán parte de nuestro prototipo. Creemos que, con las funciones definidas anteriormente, podremos generar un diseño y desarrollo que implementen las características comunes asociadas al uso de AR en aplicaciones móviles.

6. Metodología

En este apartado se explicará nuestra metodología por la cual obtuvimos los datos necesarios para diseñar y planificar el desarrollo de nuestra aplicación, asegurándonos de que esta sea representativa de las diferentes funciones que puede llegar a tener una App que utilice AR en Android. Nuestra técnica de recolección para obtener estos datos es la observación. A su vez, tuvimos una segunda instancia metodológica en la cual se utilizó la información obtenida en la instancia anterior para desarrollar un prototipo de aplicación de Realidad Aumentada, cumpliendo con un mínimo de 5 funciones.

6.1. Enfoque metodológico

En el marco de este desarrollo tomaremos una metodología mixta con foco en lo cualitativo ya que se analizaron diversas aplicaciones para relevar e identificar si cumplen con determinadas funciones de AR, obteniendo datos estadísticos sobre el uso de estas

funciones. El dato cualitativo proviene del análisis de las funciones, mientras que el cuantitativo será definido por la cantidad de apariciones de las mismas.

El objetivo de este análisis es determinar la cantidad de ocurrencias que poseen las funciones de AR y así, utilizando la moda de estos datos, definir cuáles son pertinentes para desarrollar en nuestro prototipo, logrando así filtrar las funciones que la mayoría de los desarrolladores y empresas consideran importantes. Además, a través de este análisis se cumplirán 2 de nuestros objetivos específicos, “Relevar e identificar las funciones básicas de AR en dispositivos Android” y “Proponer las funciones que se incorporan al prototipo basados en la información obtenida”.

6.2. Población y muestra

Las aplicaciones de AR analizadas fueron seleccionadas en base a los siguientes artículos: “Las 15 mejores aplicaciones con realidad aumentada” redactado por Iván Vázquez (2023) para el portal App Marketing News; “The 10 Best AR Apps, That Aren’t Pokemon Go, Bringing Us to the Future” escrito por Brian McDonald (s.f.) y publicado en el sitio web de la Universidad de Bentley; ”The 23 best ARKit apps for iPhone and iPad.” escrito por Holland, P., Broida, R., Parker, J., Savvides, L. (2018) y publicado en el sitio Cnet.com. En estas publicaciones se describen las características de algunas de las aplicaciones más utilizadas que implementan esta tecnología, estos artículos fueron seleccionados debido a que contemplan una amplia variedad de aplicaciones de AR que son utilizadas en distintos rubros ya que implementan las mismas funcionalidades para alcanzar distintos objetivos.

A su vez, se toma de referencia las recomendaciones de aplicaciones de AR ofrecidas por la tienda digital App Store, de la empresa Apple, Inc.

Una vez analizados los artículos, se obtuvieron un total de 27 aplicaciones lanzadas entre 2010 y 2021, las cuales conforman la población de esta etapa de la investigación y fueron observadas con el fin de cuantificar el uso de las distintas funciones de AR y obtener así insumos de información. El análisis de estas aplicaciones se realizó durante el mes de Agosto de 2024, a través de la instalación y prueba de las mismas por nuestra parte, corroborando el uso de las funciones de AR.

Aplicaciones	Aplicación 1	Aplicación 2	Aplicación 3
Función 1	X	✓	✓
Función 2	✓	X	✓
Función 3	✓	X	✓

X: Carece de la función, ✓: Posee la función

6.3. Prototipo

En lo que respecta al desarrollo del prototipo de aplicación y la documentación del mismo, estos serán trabajados a la par rigiéndose con una serie de hitos o milestones, los cuales son momentos claves en el desarrollo que nos permiten documentar un proceso aislado antes de continuar con el siguiente.

Estos fueron nuestros Milestones:

- Diseño general de la aplicación
- Implementación en Unity3D los recursos de XRcore (complemento para AR).
- Desarrollo de las capacidades básicas de la aplicación.
- Conclusiones finales y cierre.

El flujo propuesto por esta metodología sería el siguiente:

Se diseña la aplicación (1er milestone), y luego se documenta este mismo diseño, antes de continuar con el siguiente milestone, este proceso de desarrollo y documentación intercalados se repite una vez por milestone, generando así el cuerpo del documento.

7. Resultados

7.1. Análisis de las Aplicaciones AR

Previo al desarrollo de la aplicación, se ha realizado un análisis de cierta cantidad de aplicaciones que utilizan AR para determinar las funciones básicas con las que contará

nuestro diseño, para esto, tomaremos como referencia los artículos mencionados anteriormente.

A continuación, se exponen los resultados de las pruebas realizadas a las distintas aplicaciones para comprobar sus usos de AR y características relacionadas con esta tecnología:

Apps	Ikea Place	Wanna Kicks	Inkhunter	Wallame	MYTY AR	Quiver	Pokemon GO	HOUSE CRAFT	EDMUNDS	GIPHY WORLD
Visualización de Obj 3D	✓	✓	X	X	✓	✓	✓	✓	✓	✓
Rotación de Obj 3D	✓	X	X	X	✓	X	✓	✓	✓	X
Visualización 2D	X	X	✓	✓	X	X	✓	X	X	X
Detección de planos	✓	X	X	X	✓	X	✓	✓	✓	X
AR con marcador	X	X	✓	✓	X	✓	X	X	X	✓
AR sin marcador	✓	✓	X	X	✓	X	✓	✓	✓	X
Escalado de Obj 3D	✓	X	X	X	X	X	✓	✓	X	X

Tabla número 1 de nuestra autoría, realizada en base al artículo de Iván Vázquez (2023).

En esta tabla número 1 podemos ver el análisis de 10 aplicaciones AR, las cuales muestran una marcada inclinación por el 3D, siendo que 8 de las 10 ofrecen visualización de objetos en 3 dimensiones, dejando a Inkhunter (tatuajes AR) y Wallame (Dibujos 2D con AR) como las únicas 2 que se dedican exclusivamente al 2D. Las funciones menos utilizadas fueron el escalado de objetos 3D y la visualización 2D, contando ambas con un total de 3 apariciones.

Apps	Vuforia Chalk	Wayfair	GeoGebra AR	Ghostbusters World	JigSpace	Magic Plan	World Brush
Visualización de Obj 3D	X	✓	✓	✓	✓	X	✓
Rotación de Obj 3D	X	✓	✓	✓	✓	X	X
Visualización 2D	✓	X	X	✓	✓	✓	✓
Detección de planos	✓	✓	✓	✓	✓	✓	✓
AR con marcador	X	X	X	X	X	X	X
AR sin marcador	✓	✓	✓	✓	✓	✓	✓
Escalado de Obj 3D	X	X	✓	✓	✓	X	X

Tabla número 2 de nuestra autoría, realizada en base al artículo de Brian McDonald (s.f.).

En esta segunda tabla encontramos 7 aplicaciones AR, podemos notar que todas utilizan AR sin marcadores y la detección de planos, a su vez, ninguna emplea la AR con marcadores, con estos resultados se vuelve bastante probable que nuestro prototipo no cuente con esta tecnología y optemos por la detección de planos como alternativa. A diferencia de la tabla anterior, en este caso la visualización 2D está a la par con la de objetos 3D, lo que nos da la posibilidad de optar incluir esto en el prototipo.

Apps	Sky Guide	Instragram	Kings of Pool AR	Porsche AR	LEGO AR-Studio	Smash Tanks! AR	EXperiencia Patrón	Zombie Gunship Revenant AR	Stack AR	ARise
Visualización de Obj 3D	X	✓	✓	✓	✓	✓	✓	✓	✓	✓
Rotación de Obj 3D	X	X	✓	✓	✓	X	X	✓	X	✓
Visualización	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

2D										
Detección de planos	✓	X	✓	✓	✓	✓	✓	✓	✓	✓
AR con marcador	X	✓	X	X	X	X	X	X	X	X
AR sin marcador	✓	X	✓	✓	✓	✓	✓	✓	✓	✓
Escalado de Obj 3D	X	X	✓	✓	X	✓	X	✓	X	X

Tabla número 3 de nuestra autoría, realizada en base al artículo de Holland, P., Broida, R., Parker, J., Savvides, L. (2018).

En nuestra tercera tabla analizamos 10 aplicaciones AR, las funciones más utilizadas son la visualización 2D y 3D, la detección de planos y la AR sin marcador, tras ver esta tabla es bastante seguro que al menos esas 4 funciones serán parte de nuestro prototipo.

El análisis de estas aplicaciones fue realizado a través de la instalación y prueba de las mismas por nuestra parte, realizando diferentes pruebas para corroborar el uso de las funciones de AR. En el caso de las aplicaciones Ikea Place, Edmunds y Porsche AR se realizó al análisis a través de videos debido a que las aplicaciones no estaban disponibles por bloqueo regional, lo que refiere a que estas empresas no permiten la descarga de estas aplicaciones en nuestro país.

Resultados:

Esta siguiente tabla muestra la cantidad de apariciones de cada función de AR.

Cantidad de Apps	27
Visualización de Obj 3D	22
Rotación de Obj 3D	14
Visualización 2D	18
Detección de planos	21
AR con marcador	5
AR sin marcador	22
Escalado de Obj 3D	10

Tabla número 4. Cantidad de apariciones de funciones AR. Elaboración propia.

Teniendo estos datos, podemos calcular en qué porcentaje de las aplicaciones analizadas se encuentran presentes las funciones observadas, los cuales quedarían de la siguiente manera, ordenados de mayor a menor:

- Visualización de Objetos 3D: 81%
- AR sin marcador: 81%
- Detección de planos: 78%
- Visualización 2D: 67%
- Rotación de Objetos 3D: 52%
- Escalado de Objetos 3D: 37%
- AR con marcador: 19%

Los datos muestran que las funciones más frecuentes son la visualización de objetos 3D y el uso de AR sin marcador, ambas presentes en el 81% de las aplicaciones. A estas les siguen la detección de planos (78%) y la visualización 2D (67%). En menor proporción se encuentran la rotación (52%) y el escalado (37%) de objetos 3D, mientras que la AR con marcador es la menos común (19%).

A partir del análisis cuantitativo realizado, se determina que las funcionalidades más representativas, es decir, aquellas presentes en más del 50% de las aplicaciones estudiadas, deben ser consideradas como esenciales para el diseño y desarrollo de nuestra aplicación, la cual debe contar con las siguientes funciones: **Visualización de Objetos 3D; Rotación de Objetos 3D**; **AR sin marcador**; **Detección de planos**; **Visualización 2D**.

7.2. Desarrollo de la aplicación

7.2.1. Milestone 1: Diseño General de la Aplicación.

En este apartado se expone el desarrollo de nuestra aplicación, diseñada en base a los resultados obtenidos en la primera etapa de la investigación, a partir de los cuales desarrollamos las varias características de la misma. A continuación se explica la navegación de escenas y los comportamientos de las mismas para mostrar las funciones de AR.

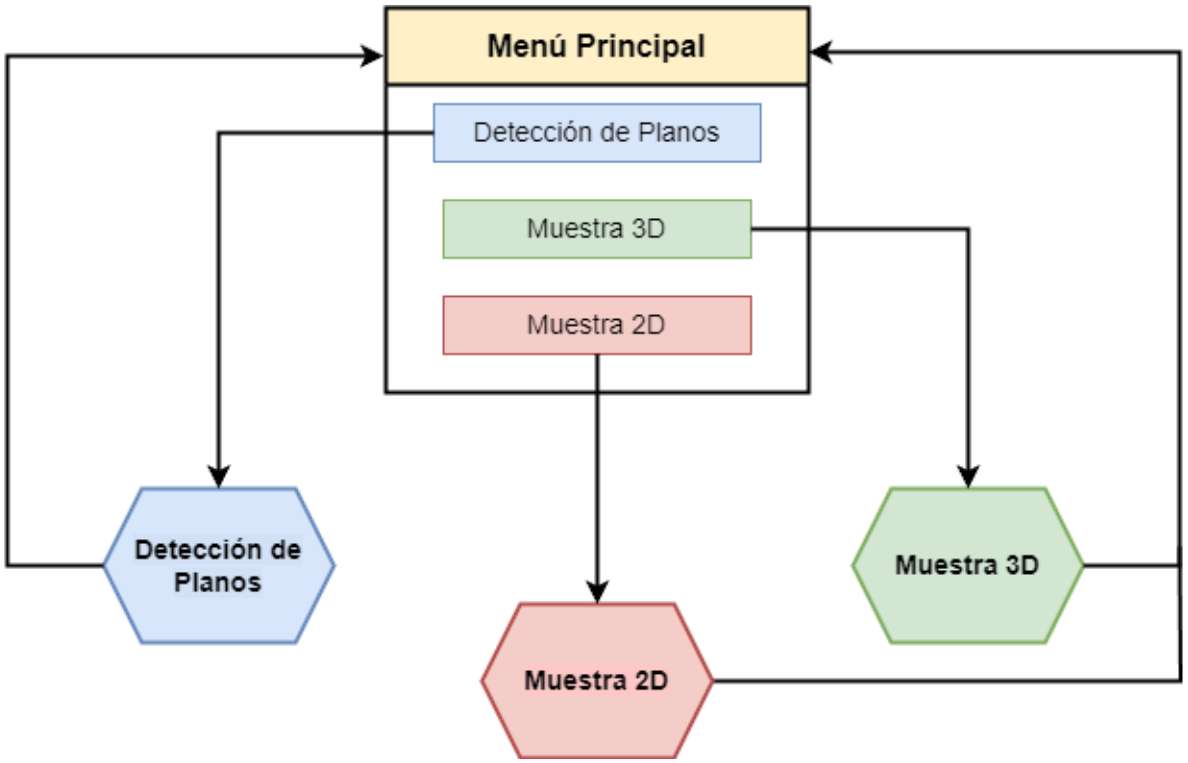


Ilustración 5. Diagrama que muestra el flujo propuesto para la aplicación. Imagen de nuestra autoría.

En este diagrama de flujo se grafica la navegación por las interfaces de nuestra aplicación, al iniciarla nos encontramos con un menú principal donde podemos seleccionar las funciones que queremos demostrar/probar, una vez seleccionado el botón de la función se nos lleva a una nueva escena que demuestra la misma, a su vez, cada escena posee un botón para volver al menú principal.

Dentro de la sección de **Detección de Planos** podemos ver la cámara del dispositivo y un texto que dice “Mueva su dispositivo apuntando a superficies planas”, una vez que la aplicación detecta el plano, se genera un indicador visual sobre el mismo para mostrarle al

usuario cual es el plano detectado, tras esto se puede seguir buscando múltiples planos en el espacio o volver al menú principal.

En la **Muestra 3D** se pide al usuario que "Mueva su dispositivo apuntando a superficies planas" para utilizar la tecnología de detección de planos, una vez generado el indicador visual del plano, el usuario puede colocar objetos 3D tocando sobre la superficie donde quiera generarlos, una vez colocados, puede mantenerlos pulsados para moverlos en el plano y/o rotarlos, también, tiene la opción de borrar los objetos y comenzar nuevamente. Cuando lo desee, puede volver al menú principal.

En cuanto a la **Muestra 2D**, se le solicita al usuario que "Mueva su dispositivo apuntando a una superficie plana", una vez detectada se puede ver la interfaz y aparecen las opciones de visualización 2D, estas ofrecen la capacidad mostrar una imagen sobre el plano, rotarla y eliminarla de la escena. Cuando lo desee, puede volver al menú principal.

7.2.2. Milestone 2: Implementar en Unity3D los recursos de XRcore.

Para realizar esto, primero es necesario iniciar sesión en la aplicación de Unity Hub y generar un nuevo proyecto utilizando la plantilla de “AR Mobile”:

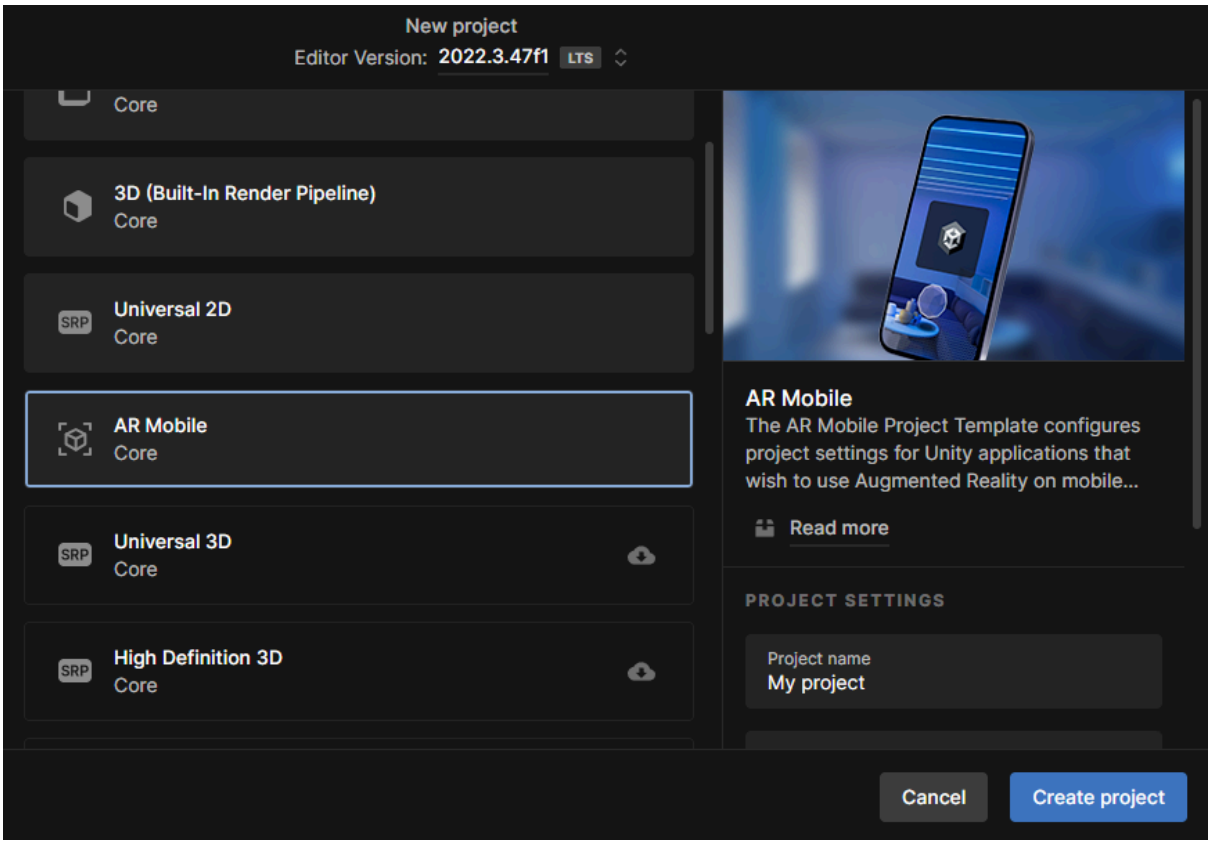


Ilustración 6. El Menú de creación de proyectos de Unity Hub, el cual permite crear proyectos usando plantillas que contienen características comunes ya implementadas. Imagen de nuestra autoría.

Si realizamos este método, se abrirá a continuación el editor de Unity con una escena que debería verse de la siguiente manera:

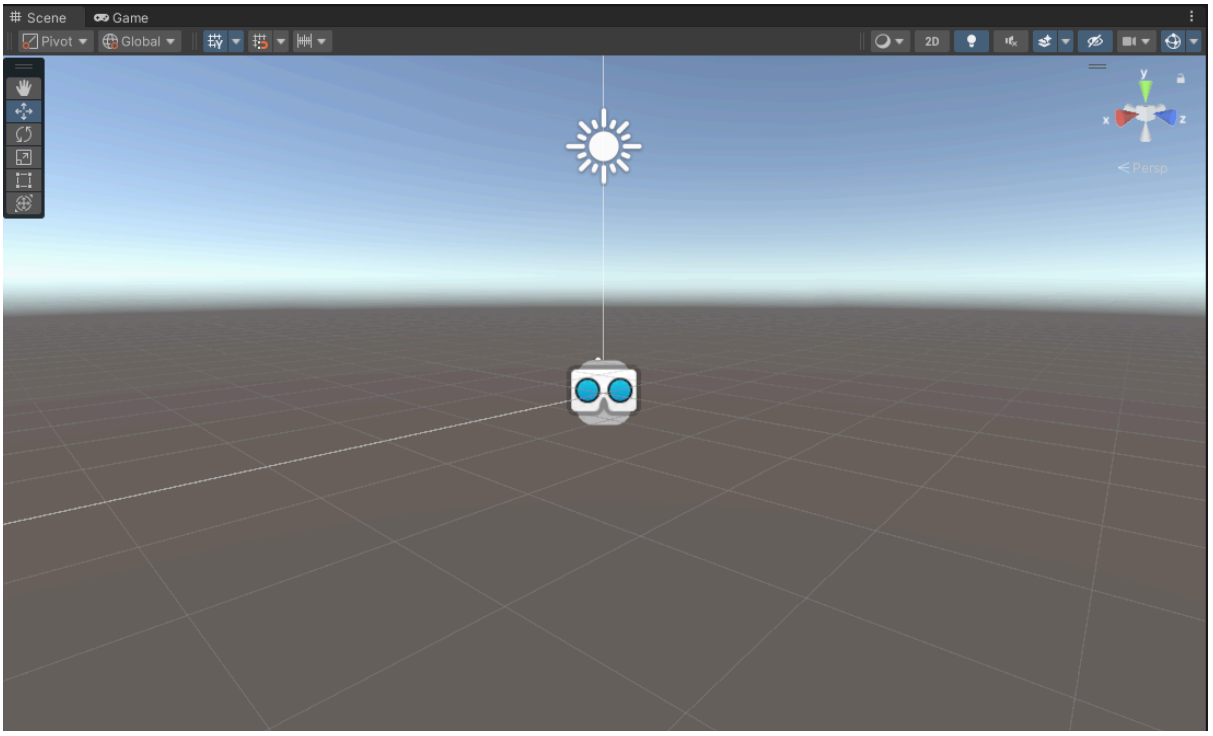


Ilustración 7. La vista de escena dentro del editor, los lentes en el centro de la imagen representan los componentes de AR presentes en la misma. Imagen de nuestra autoría.

Esta plantilla de "AR Mobile" ofrecida por Unity ya tiene instalado el complemento de XRcore, por si fuera posible encontrarse algún inconveniente con la instalación, a continuación se muestra como instalar XRcore en cualquier proyecto de Unity:

- Paso 1: Buscamos en Unity la ventana de Package Manager.

- Paso 2: Dentro del Package Manager seleccionaremos la opción de visualizar los paquetes de Unity Registry.

- Paso 3: Buscamos dentro de la lista "Packages" el paquete llamado "XR Interaction Toolkit", una vez encontrado seleccionamos "Install". Este paquete agrega al proyecto de Unity el complemento XR Core entre otras herramientas de Unity que se utilizan para AR.

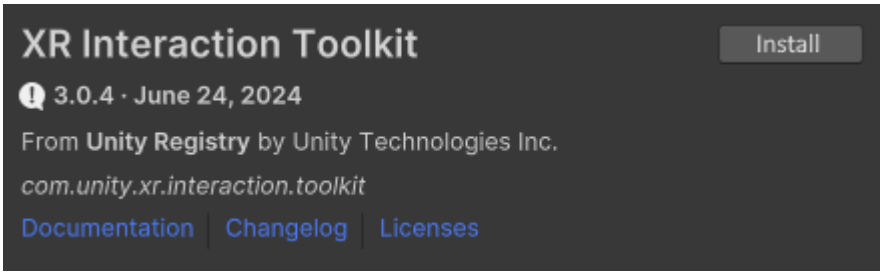


Ilustración 8. El paquete "XR Interaction Toolkit", visto en el package manager. Imagen de nuestra autoría.

7.2.3. Milestone 3: Desarrollo de las capacidades básicas de la aplicación.

El lenguaje de programación a utilizar es C#, ya que Unity utiliza este lenguaje de forma nativa para la programación de scripts, permitiendo a los desarrolladores controlar el comportamiento de los objetos, la lógica de la aplicación y la interacción con el usuario. A diferencia de otros motores que pueden utilizar lenguajes como C++ o JavaScript, Unity ha optimizado su entorno para trabajar con C#, lo que nos lleva a tomar esta decisión.

Lo primero que hacemos al comenzar con el desarrollo es sentar las bases de la aplicación, generando unos botones para crear un prototipo de menú principal, estos botones son "Detección de Planos" y "Salir", con el primero podemos ingresar a la primera escena o muestra, ya que la detección de planos debe ser la primera en desarrollarse, al ser una tecnología necesaria para el resto de muestras e implementaciones de AR, con el botón de "Salir" podemos cerrar la aplicación.

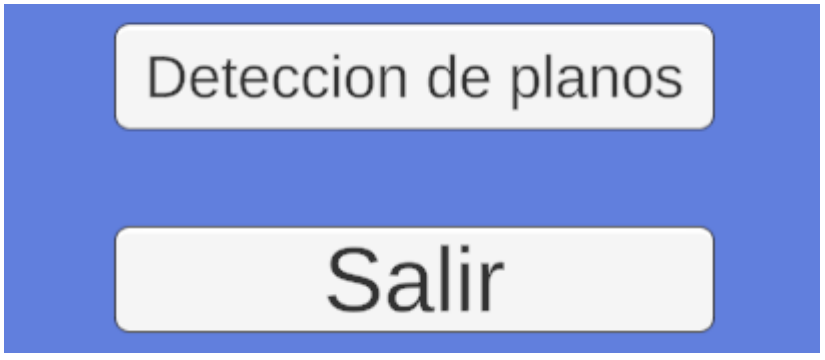


Ilustración 9. Los botones mencionados, puestos en la escena. Imagen de nuestra autoría.

Para comenzar con la escena "Detección de Planos" creamos un nuevo objeto tipo XR Origin (Mobile AR), este objeto contiene los componentes necesarios para la detección de

planos, la interacción con la pantalla táctil del dispositivo y la relación con la cámara del mismo.

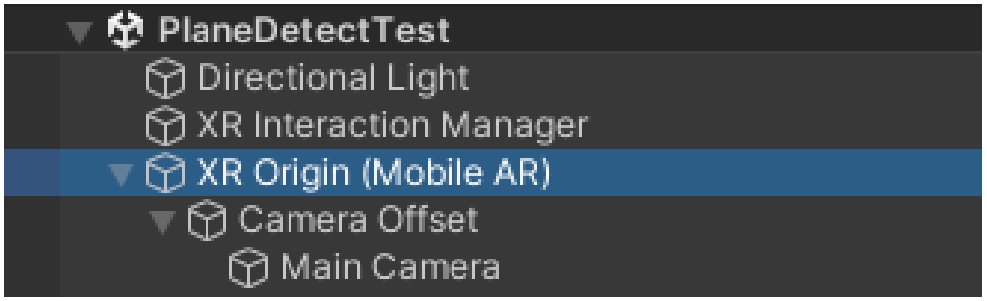


Ilustración 10. Objeto XR Origin en la jerarquía de la escena. Imagen de nuestra autoría

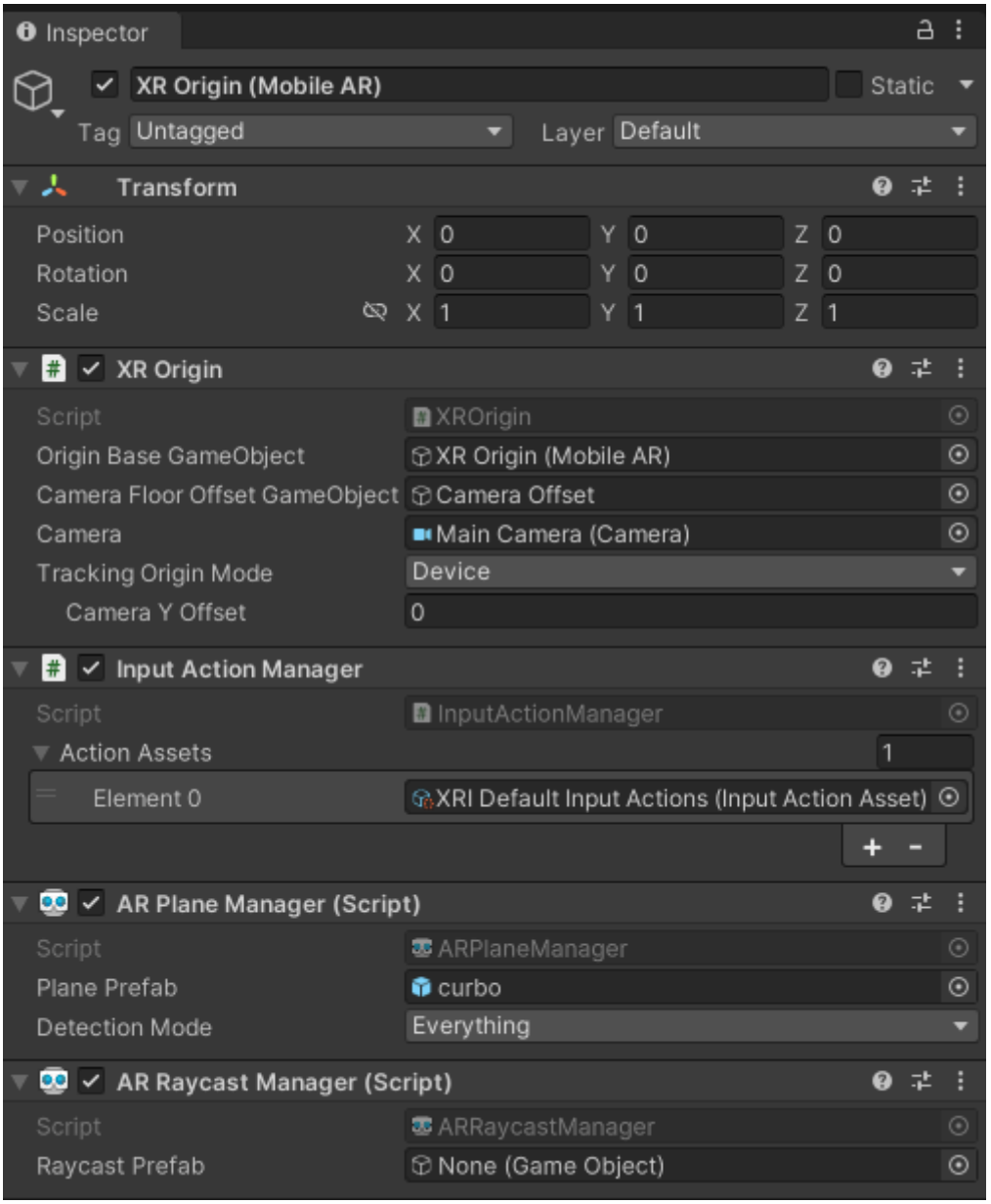


Ilustración 11. El objeto "XR Origin" y sus componentes. Imagen de nuestra autoría.

El primer componente, "XR Origin" controla la relación con la cámara del dispositivo, y la determina como el centro de la imagen donde se generan las acciones de la aplicación. El "Input Action Manager" nos permite interactuar con los elementos AR a través de la pantalla táctil, ya que por defecto solo podríamos tocar botones, con este componente accedemos a la interacción con el plano. Tanto "AR Plane Manager" y "AR Raycast Manager" se encargan de generar la detección de planos y determinar la distancia entre el dispositivo y el plano en el espacio real.

Además de estos componentes, vamos a agregar un extra que viene con XR Core, este es el "AR Input Manager", que se encarga de relacionar la posición del dispositivo con lo visto en la escena:

Ilustración 12. El componente "AR Input Manager". Imagen de nuestra autoría.

Gracias al complemento XR Core, que nos brinda estos componentes, ya poseemos una detección de planos funcional, lo que resta por hacer es darle una interfaz visual para que el usuario reconozca el plano. Por el momento, para visualizar el plano detectado se generará un Cubo en el espacio del plano.

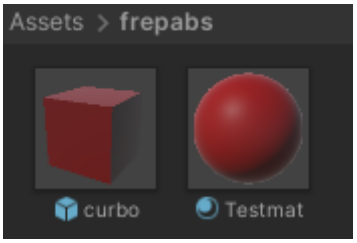


Ilustración 13. El objeto de prueba y el material. Imagen de nuestra autoría.

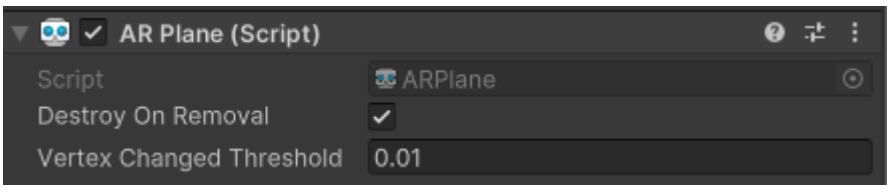


Ilustración 14. Se le agrega al cubo el componente "AR Plane" que le permite ser reconocido por el sistema de detección de planos como su indicador visual. Imagen de nuestra autoría.

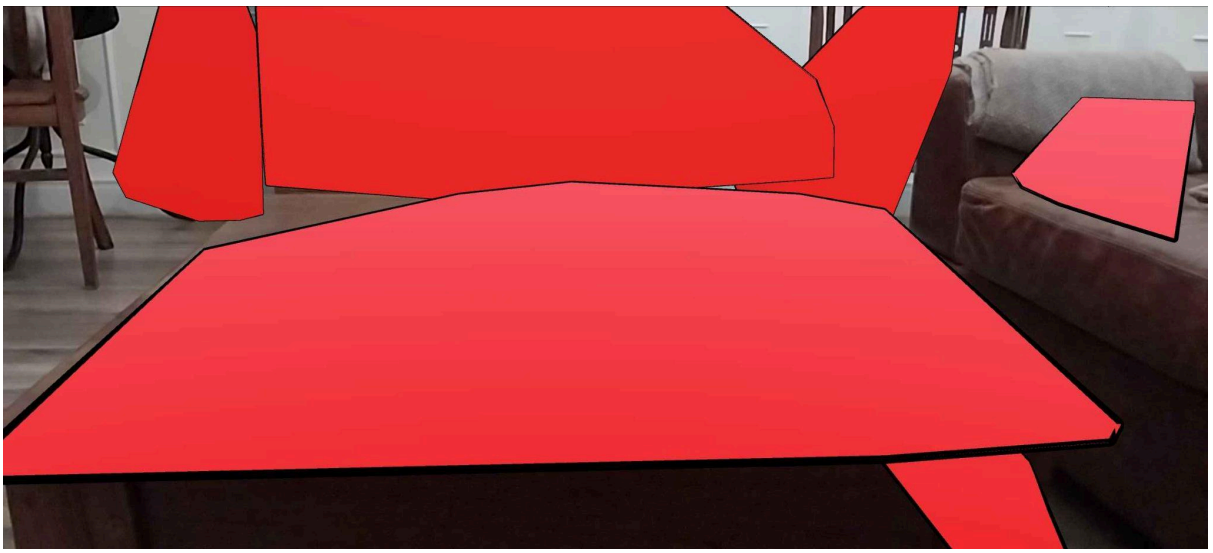
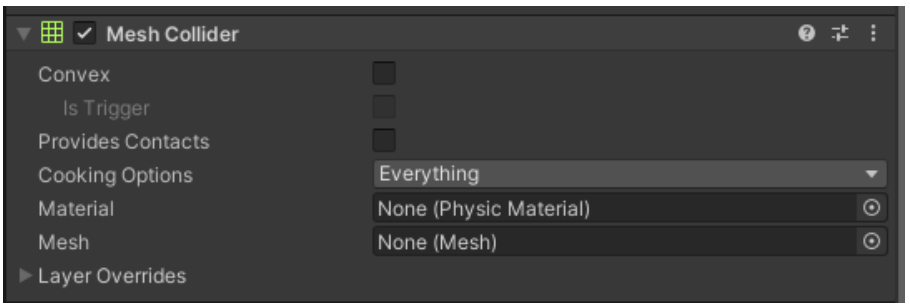
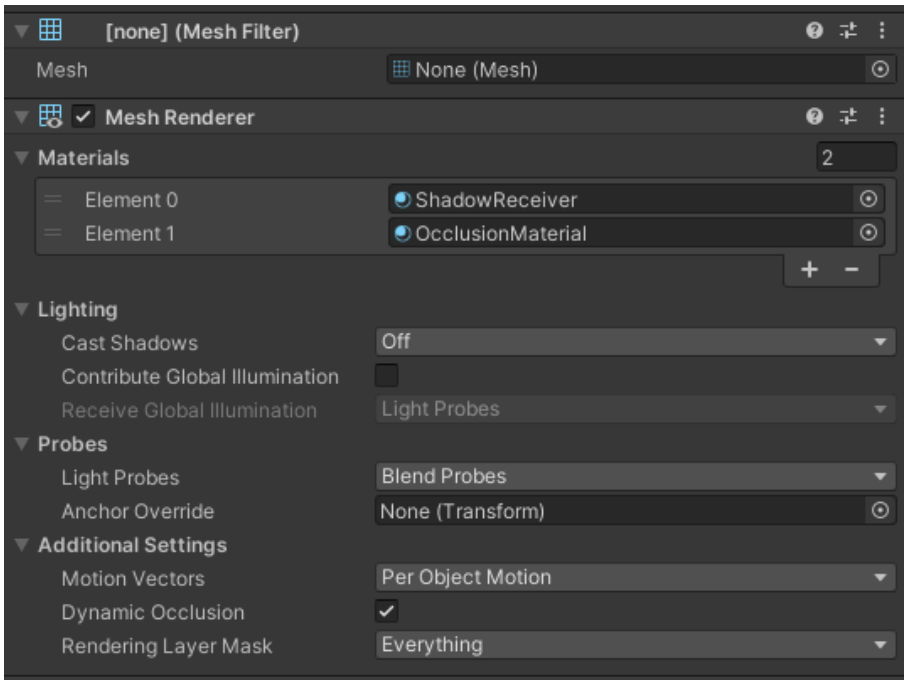


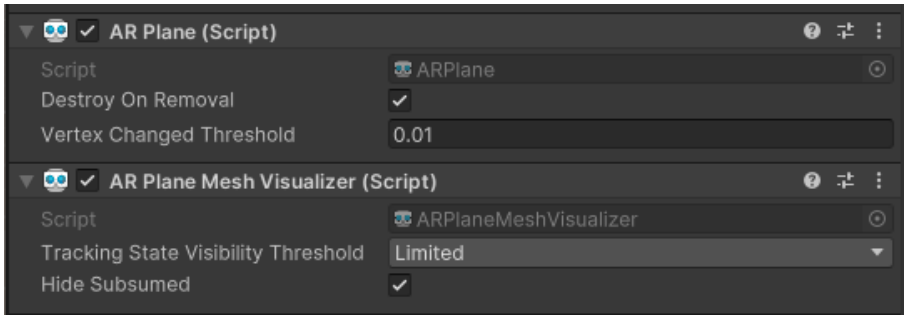
Ilustración 15. Ejemplo visual de los cubos estirados para representar planos reconocidos por la cámara. Imagen de nuestra autoría.

En este caso, la aplicación está detectando diversos planos en el espacio, y se encarga de estirar el cubo para amoldarse al tamaño de ellos. Ya que diferenciar los planos resulta complicado al tener un color sólido, a continuación se reemplazará el cubo por un objeto con los componentes correspondientes y las características necesarias para mejorar la compatibilidad con la detección de planos. Entre estos componentes se encuentran los siguientes: Mesh Collider; Mesh Filter; Mesh Renderer; AR Plane Mesh Visualizer; AR Feathered Plane Visualizer y AR Feathered Plane Mesh Visualizer Companion.

Asi quedarían los componentes después de realizarles algunos ajustes:


Mesh Collider
Convex
Is Trigger
Provides Contacts
Cooking Options
Everything
Material
None (Physic Material)
Mesh
None (Mesh)
Layer Overrides


[none] (Mesh Filter)
Mesh
None (Mesh)
Mesh Renderer
Materials
2
Element 0
ShadowReceiver
Element 1
OcclusionMaterial
Lighting
Cast Shadows
Off
Contribute Global Illumination
Receive Global Illumination
Light Probes
Probes
Light Probes
Blend Probes
Anchor Override
None (Transform)
Additional Settings
Motion Vectors
Per Object Motion
Dynamic Occlusion
Rendering Layer Mask
Everything


AR Plane (Script)
Script
ARPlane
Destroy On Removal
Vertex Changed Threshold
0.01
AR Plane Mesh Visualizer (Script)
Script
ARPlaneMeshVisualizer
Tracking State Visibility Threshold
Limited
Hide Subsumed

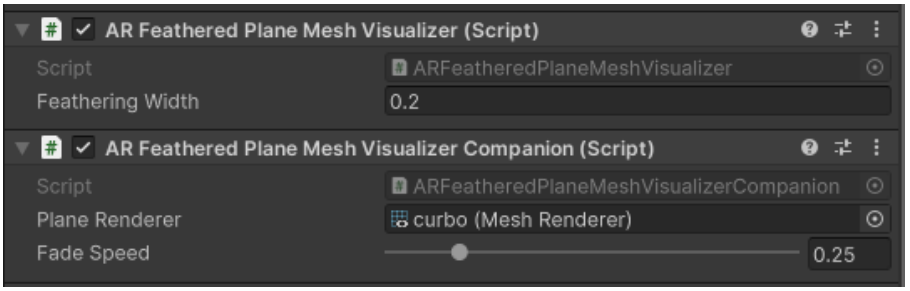

AR Feathered Plane Mesh Visualizer (Script)
Script
ARFeatheredPlaneMeshVisualizer
Feathering Width
0.2
AR Feathered Plane Mesh Visualizer Companion (Script)
Script
ARFeatheredPlaneMeshVisualizerCompanion
Plane Renderer
curbo (Mesh Renderer)
Fade Speed
0.25

Ilustración 16. Los componentes de nuestro objeto, ajustados para su uso como plano de prueba. Imagen de nuestra autoría.

Esto resulta en una representación visual menos intrusiva, ya que la transparencia agregada permite al usuario ver el objeto físico del cual se creó el plano, además de un trazado de puntos para mejorar la distinción del área con la que se puede interactuar:

Ilustración 17. Nuestro plano de prueba, superpuesto en la superficie de la mesa detectada por el AR Plane manager. Imagen de nuestra autoría.

Una vez hecho esto, seguiremos con la siguiente sección del proyecto: *Instanciar y Manipular Objetos 3D*. Lo primero que debemos hacer es crear un objeto y agregarle los componentes “Cube (Mesh Filter)”, “Mesh Renderer” y “Mesh Collider”:

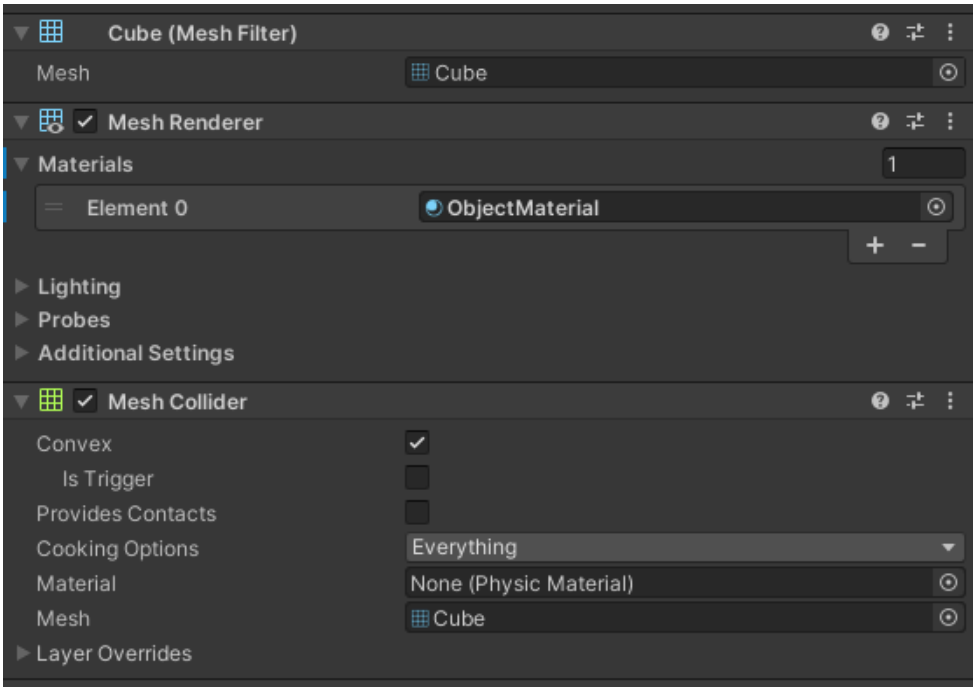


Ilustración 18. Los componentes mencionados anteriormente asignados al objeto. Imagen de nuestra autoría.

Para asegurarnos que los objetos instanciados actualicen sus posiciones y poses de forma correcta, es necesario agregar el complemento ARSessionManager.

La forma más simple de lograr esto es creando un cubo en la escena, arrastrarlo hacia el explorador de archivos de Unity y borrando el objeto en la escena. Esto creará un Prefab del cubo listo para usar (un prefab es un objeto prefabricado, que agregamos a la escena desde Unity). Este prefab de cubo tendrá los siguientes componentes: “Rigidbody”, “XR Grab Interactable”, “AR Transformer”.

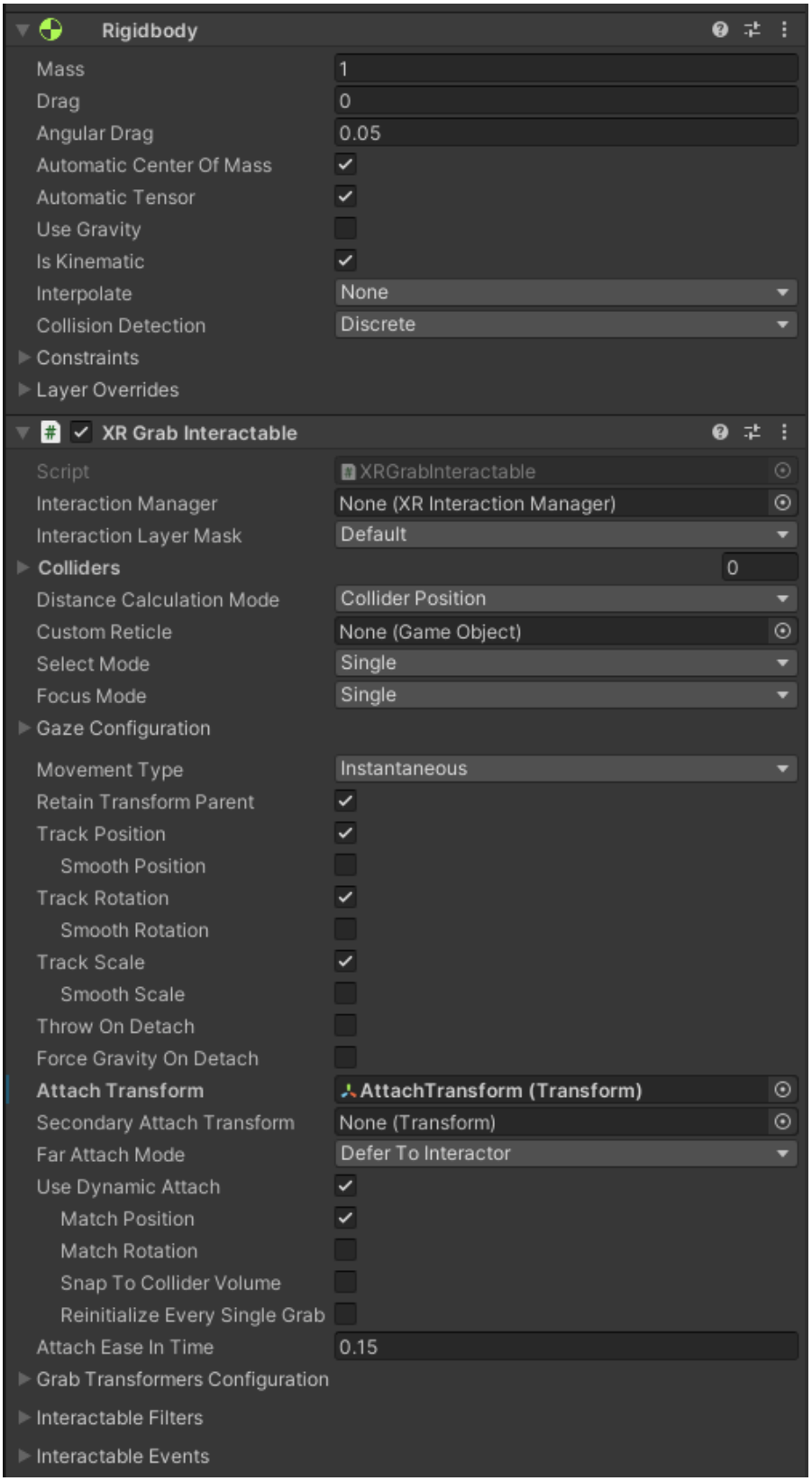

Rigidbody
Mass
1
Drag
0
Angular Drag
0.05
Automatic Center Of Mass
Automatic Tensor
Use Gravity
Is Kinematic
Interpolate
None
Collision Detection
Discrete
Constraints
Layer Overrides
XR Grab Interactable
Script
XRGrabInteractable
Interaction Manager
None (XR Interaction Manager)
Interaction Layer Mask
Default
Colliders
0
Distance Calculation Mode
Collider Position
Custom Reticle
None (Game Object)
Select Mode
Single
Focus Mode
Single
Gaze Configuration
Movement Type
Instantaneous
Retain Transform Parent
Track Position
Smooth Position
Track Rotation
Smooth Rotation
Track Scale
Smooth Scale
Throw On Detach
Force Gravity On Detach
Attach Transform
AttachTransform (Transform)
Secondary Attach Transform
None (Transform)
Far Attach Mode
Defer To Interactor
Use Dynamic Attach
Match Position
Match Rotation
Snap To Collider Volume
Reinitialize Every Single Grab
Attach Ease In Time
0.15
Grab Transformers Configuration
Interactable Filters
Interactable Events

Ilustración 19. El componente "XR Grab Interactable" puesto en nuestro objeto. Imagen de nuestra autoría.

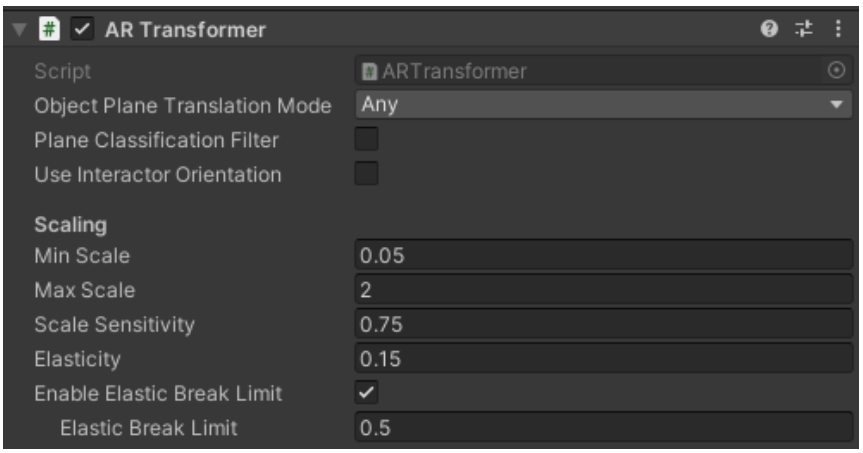


Ilustración 20. El componente "AR Transformer". Imagen de nuestra autoría.

Este mismo cubo tiene 2 objetos hijos, a los que llamamos "AttachTransform" y "Visuals", el primero de estos solo recibirá un cambio de escala, de 0 a 5, el segundo tendrá los componentes "Cube (Mesh Filter)", "Mesh Renderer" y "Mesh Collider".

El "AttachTransform" se encargará de definir la ubicación del cubo cuando el usuario lo mueva, mientras que el "Visual", como dice su nombre, se encarga de mostrar el apartado visual del objeto. Estos componentes sumados a el código que se explica a continuación, nos dan como resultado la habilidad de instanciar nuestro objeto sobre el plano detectado:

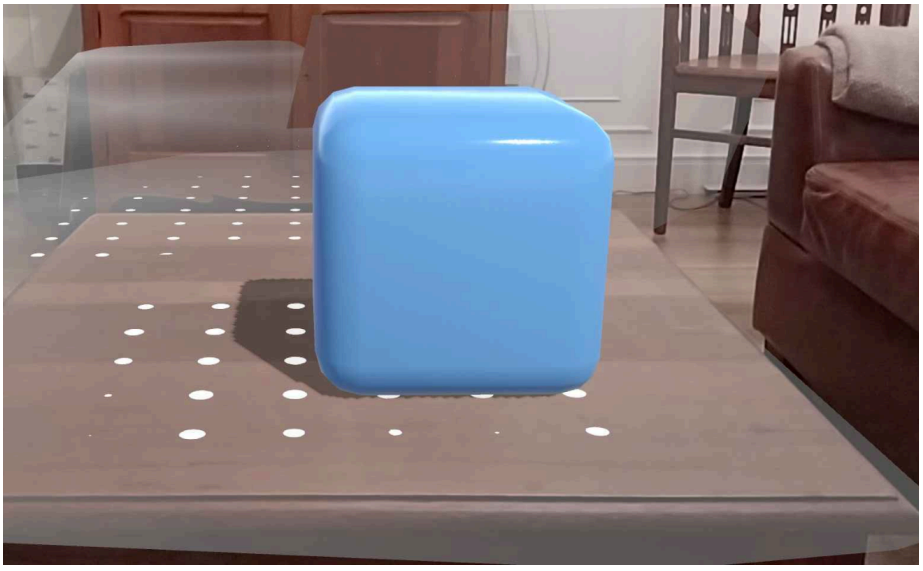


Ilustración 21. Nuestro objeto está proyectado sobre la mesa usando un plano detectado. Imagen de nuestra autoría.

Para la siguiente parte necesitaremos construir nuestra propia lógica de interacción con los sistemas de Unity para poder implementar los comportamientos que buscamos.

Para empezar, crearemos un nuevo script con las siguientes propiedades:
El formato de estos elementos es Tipo/Nombre, estas son la variables que utilizamos dentro del código:

```
- ARRaycastManager m_RaycastManager;
- List<ARRaycastHit> m_hits = new List<ARRaycastHit>();
- GameObject ObjectPrefab;
- GameObject ArPlaneMgr;
- Camera ARcam;
- GameObject instanceofObjectPrefab;
- float staleDist = 0;
- int LayerIgnoreRaycast;
- public XRIDefaultInputActions asset;
- bool hitObjectPrefab = false;
- InputAction touchAR;
- InputAction touchCountAR;
- InputAction touchPosAR;
- InputAction dragCurrentPos;
- InputAction dragAR;
- InputAction twistStartAR;
- InputAction twistAR;
```

Ilustración 22. Las variables a definir dentro de nuestro script principal. Imagen de nuestra autoría.

Esto nos permitirá comunicarnos con varios sistemas:

- El ARRaycastManager nos permitirá encontrar los planos en el espacio.
- El XRIDefaultInputActions es la referencia a nuestro listado de interacciones que puede realizar el usuario con el dispositivo, esto es proporcionado por XR Core
- Los InputActions nos permitirán definir acciones de entrada para facilitar la interacción con nuestro objeto y reconocer el dispositivo.
- La Cámara nos permitirá calcular la dirección de los toques del usuario.

Luego empezamos a inicializar nuestros objetos con la información necesaria para ser usados, a continuación se muestran las funciones dentro del script:

```
void Awake()
    {
        instanceofObjectPrefab = null;
        ARcam = GameObject.Find("Main Camera").GetComponent<Camera>();
        LayerIgnoreRaycast = LayerMask.NameToLayer("Ignore Raycast");
    }

void Start()
    {
        asset = new XRIDefaultInputActions();
        touchAR = asset.FindAction("Spawn Object");
        touchCountAR = asset.FindAction("Screen Touch Count");
        touchPosAR =  asset.FindAction("Tap Start Position");
        dragAR = asset.FindAction("Drag Delta");
        dragCurrentPos = asset.FindAction("Drag Current Position");
        twistAR = asset.FindAction("Twist Delta Rotation");
        twistStartAR = asset.FindAction("Twist Start Position");

        touchAR.Enable();
        touchCountAR.Enable();
        touchPosAR.Enable();
        dragCurrentPos.Enable();
        dragAR.Enable();
        twistStartAR.Enable();
        twistAR.Enable();
    }
```

Ilustración 23. Las funciones de inicialización de varias funciones clave del programa. Imagen de nuestra autoría.

Estas 2 funciones son ejecutadas al inicio del programa, una después de otra, con el objetivo de realizar la configuración inicial de los objetos necesarios para realizar nuestros comportamientos.

- El awake() es responsable de establecer una referencia a la cámara principal de la escena, inicializar nuestra instancia del cubo (Le decimos que no existe todavía) y creamos otra referencia, esta vez a la capa “Ignore Raycast”.

- El start() se encarga de inicializar nuestras InputActions y habilitarlas para que el programa sepa que el script procesará la entrada del usuario.

Una vez hecho esto, podremos empezar a implementar las opciones básicas de interacción con el objeto instanciado:

7.2.3.1. Movimiento

```
void Update()
{
    RaycastHit hit;
    Ray ray = ARcam.ScreenPointToRay(touchPosAR.ReadValue<Vector2>());

    if (m_RaycastManager.Raycast(touchPosAR.ReadValue<Vector2>(),
m_hits) && touchAR.IsPressed())
    {
        if (instanceofObjectPrefab == null)
        {
            if (Physics.Raycast(ray, out hit))
            {
                if (hit.collider.gameObject.tag == "Spawnable")
                {
                instanceofObjectPrefab = hit.collider.gameObject;
                hitObjectPrefab = true;
                }
                else
                {
                      SpawnPrefab(m_hits[0].pose.position);
                }
            }
         }
     }
}
private void SpawnPrefab(Vector3 spawnPosition)
     {instanceofObjectPrefab = Instantiate(ObjectPrefab, spawnPosition,
Quaternion.identity);}
```

Ilustración 24. La lógica que determina con cuál objeto está interactuando el usuario. Imagen de nuestra autoría.

La función Update se ejecuta constantemente en repetición, por esto mismo, la utilizamos para comprobar cuando se interactúa con el dispositivo y reaccionar de manera acorde. Primero utilizamos nuestro RaycastHit para encontrar los planos, luego comprobamos si el usuario toca alguno de los planos, una vez recibida esta entrada, nuestro objeto 3D aparece sobre el mismo plano, en la ubicación de la intersección entre este y la posición que tocó el usuario, esto se da llamando a la función SpawnPrefab, la cual se encarga de instanciar el objeto.

Se guardará una referencia al objeto instanciado para impedir que aparezcan más objetos en entradas consecuentes. Luego de esto, comprobamos la interacción "Drag" que se aplica para mover el objeto, si el usuario presiona sobre el objeto, este seguirá el movimiento de la entrada. Para esto utilizamos como receptores las entradas de acción establecidas previamente: "dragCurrentPos" y "dragAR", implementando el booleano "hitObjectPrefab" para comprobar que el usuario haya presionado sobre el objeto.

7.2.3.2. Rotación

```
if (instanceofObjectPrefab != null)
            {
                float twistDelta = twistAR.ReadValue<float>();

                if (Mathf.Abs(twistDelta) > 0.01f)
                {
                 instanceofObjectPrefab.transform.Rotate(Vector3.up,
twistDelta * Time.deltaTime * 100f, Space.World);
                }
            }
```

Ilustración 25. Condición lógica que determina a que dirección se rota el objeto. Imagen de nuestra autoría.

Este código se agrega dentro del Update y utiliza las entradas de acción "twistStartAR" y "twistAR", las cuales comprueban que el usuario está utilizando 2 entradas o 2 dedos y girandolos sobre la pantalla para rotar el objeto según el movimiento de estos. El mismo control de entradas se asegura por defecto de que la función de movimiento no se

ejecute mientras utilizamos la rotación y viceversa, evitando que estos comportamientos se superpongan de manera indeseada.

Habiendo preparado los pasos anteriores, ahora resta generar la escena en Unity, para esto vamos a rellenar los campos de nuestros códigos en el inspector:

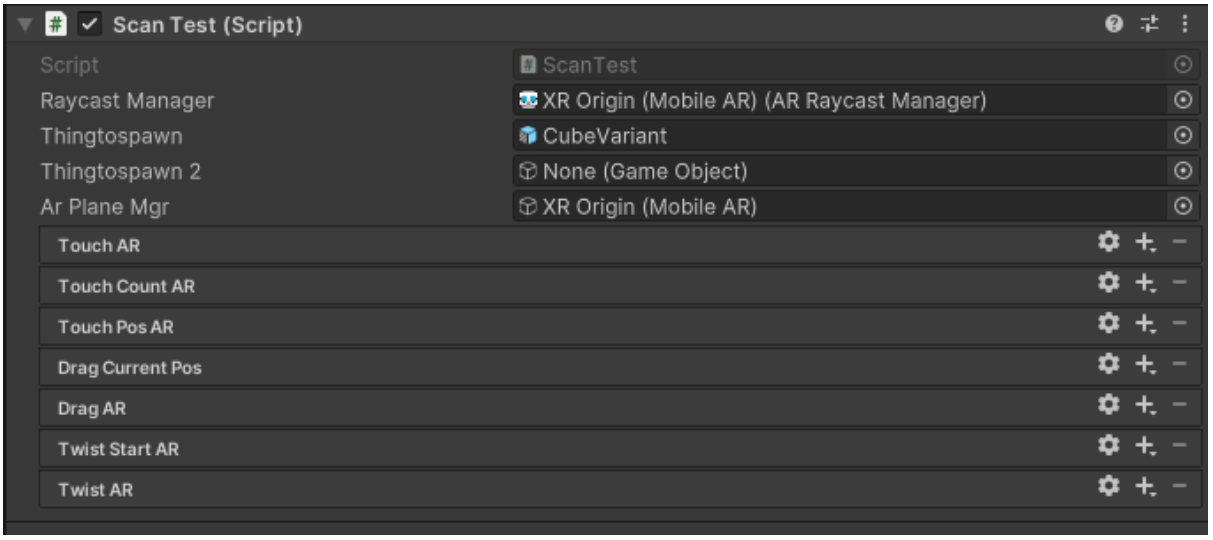


Ilustración 26. Nuestro script configurado con las instancias necesarias. Imagen de nuestra autoría.

Sólo basta con seleccionar los objetos en la escena y arrastrarlos en los campos del inspector, en este caso, los únicos campos a completar son el de “Raycast Manager”, “ObjectPrefab” (objeto que queremos instanciar) y “Ar Plane Mgr” (controlador de planos).

Los campos restantes se autocompletan al ejecutar el programa, ya que estos son cargados por nuestro código, sobrescribiendo los parámetros establecidos en el inspector y por ende quedarán vacíos.

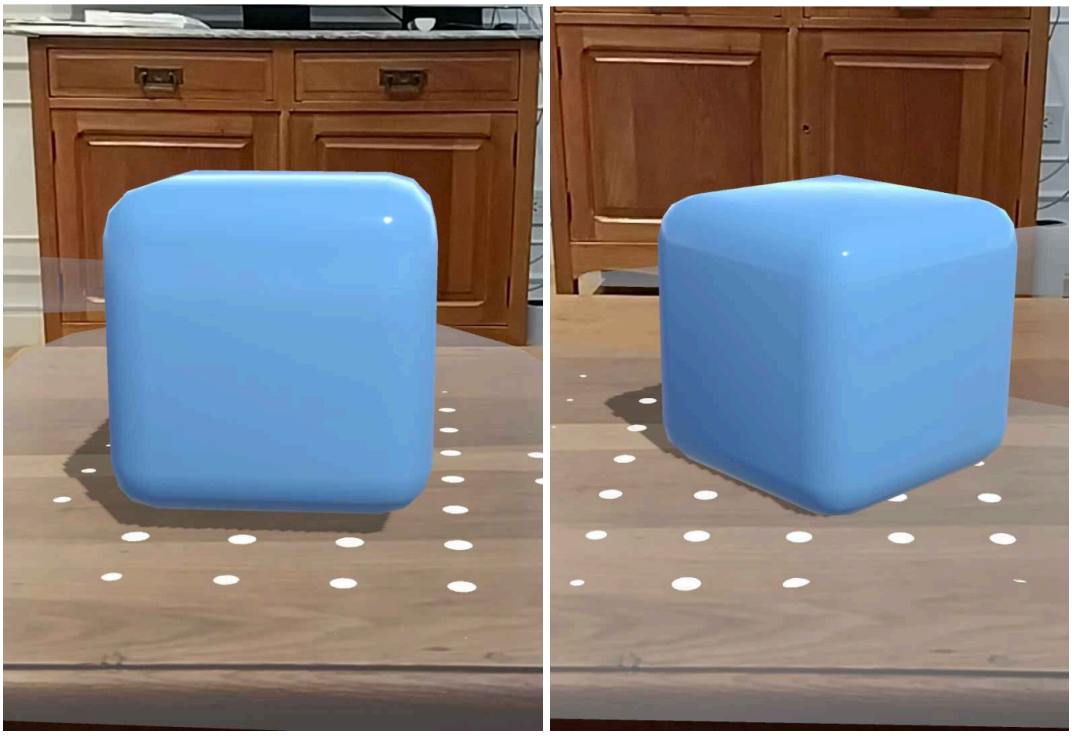
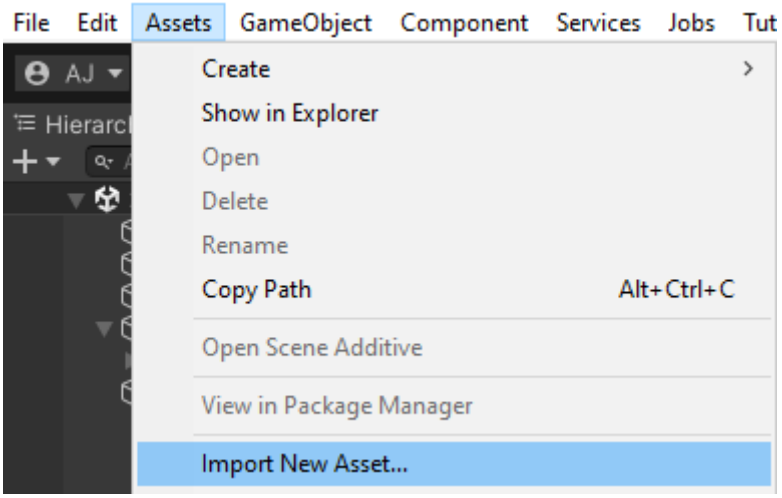


Ilustración 27. El objeto rotando sobre su propio eje. Imagen de nuestra autoría.

La combinación de estos comportamientos resulta en una escena que demuestra de forma simple como instanciar y manipular objetos 3D, haciendo uso de los varios complementos y funcionalidades otorgadas por el paquete proveído por XR Foundation, el sistema de entrada de Unity y scripts propios que sacan ventaja de los 2 complementos anteriores.

Continuamos con el desarrollo de una tercera escena, en este caso la escena 2D, donde buscaremos generar un plano que muestre una imagen plana determinada en el espacio 3D, con la posibilidad de rotarla. Para esto, necesitamos agregar nuestra imagen a Unity, dirigiéndonos a la ventana “Assets” y seleccionar “Importar nuevo asset”, donde se abrirá nuestro explorador de archivos para buscar la imagen.



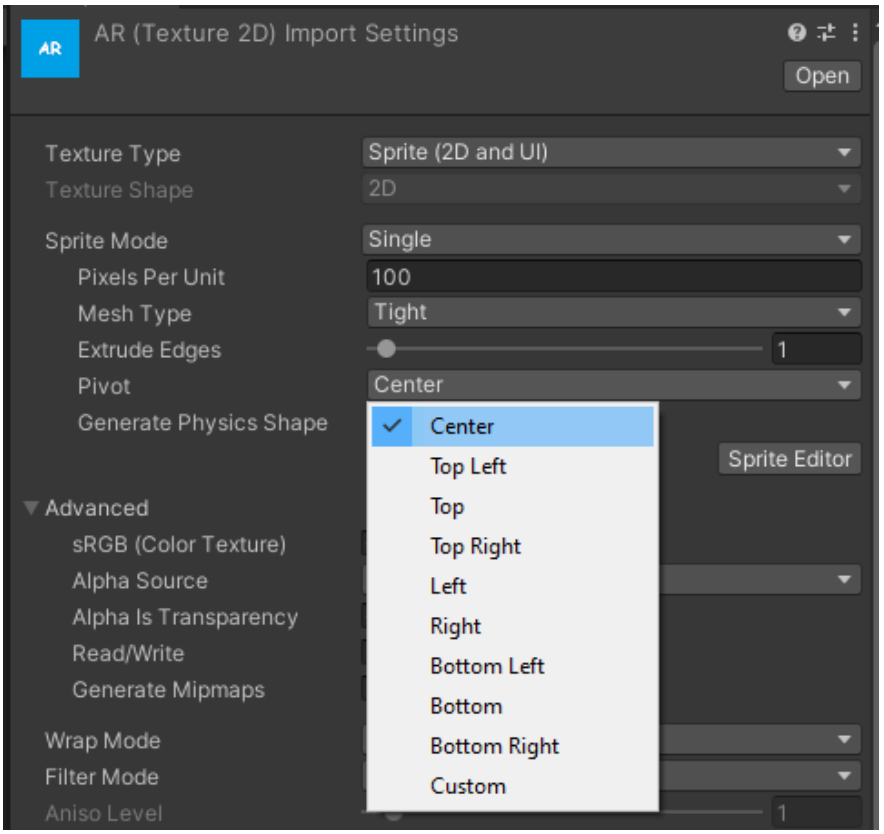


Ilustración 28. La opción "Sprite (2D and UI)" seleccionada. Imagen de nuestra autoría.

Una vez agregada la imagen a Unity es importante definir el tipo de textura como "Sprite (2D and UI)" y darle el pivot en "Center" o centro de la imagen. Nuestra imagen de ejemplo es la siguiente:

Ilustración 29. Nuestra textura de ejemplo. Imagen de nuestra autoría.

Para mostrarla en nuestra escena debemos hacer un prefab de la imagen, y al igual que con el cubo, le daremos los componentes "Rigidbody", "XR Grab Interactable", "AR Transformer", también agregaremos el objeto hijo "AttachTransform" para definir la posición del objeto. Es importante recordar que nuestra imagen debe tener su Rigidbody con la propiedad "is Kinematic" activada, esto evita que podamos moverla utilizando el mismo código Scan Test que utilizamos en la escena 3D.

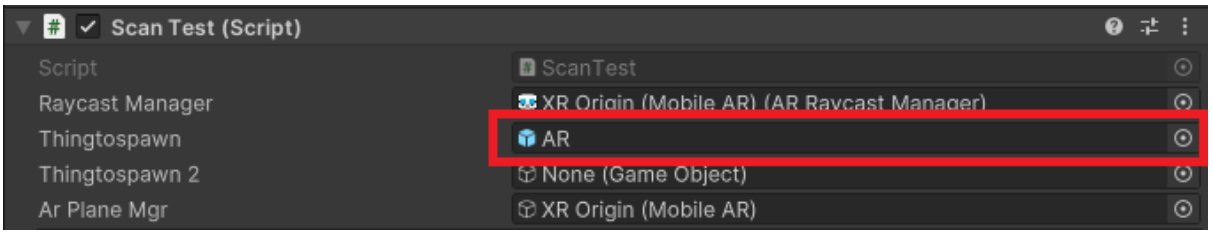


Ilustración 30. La textura puesta como parámetro en el script. Imagen de nuestra autoría.

Una vez creado este prefab debemos agregarlo a nuestro objeto "XR Origin" para reemplazar el prefab del cubo por el de la imagen en nuestro código. Se añadió un booleano en ScanTest para definir si la escena es 2D o 3D, ya que en nuestro caso queremos rotar la imagen 2D para mostrarla acostada sobre el plano:

```
if (_2DScene == true)
      {
       instanceofObjectPrefab.GetComponent<Transform>().Rotate(90,0,0);
       instanceofObjectPrefab.GetComponent<Transform>().localPosition =
new
Vector3(instanceofObjectPrefab.GetComponent<Transform>().localPosition.
x,      instanceofObjectPrefab.GetComponent<Transform>().localPosition.y
+0.001f,
instanceofObjectPrefab.GetComponent<Transform>().localPosition.z);


      }
```

Ilustración 31. La lógica en cuestión. Imagen de nuestra autoría.

Hecho esto obtendremos el siguiente resultado:

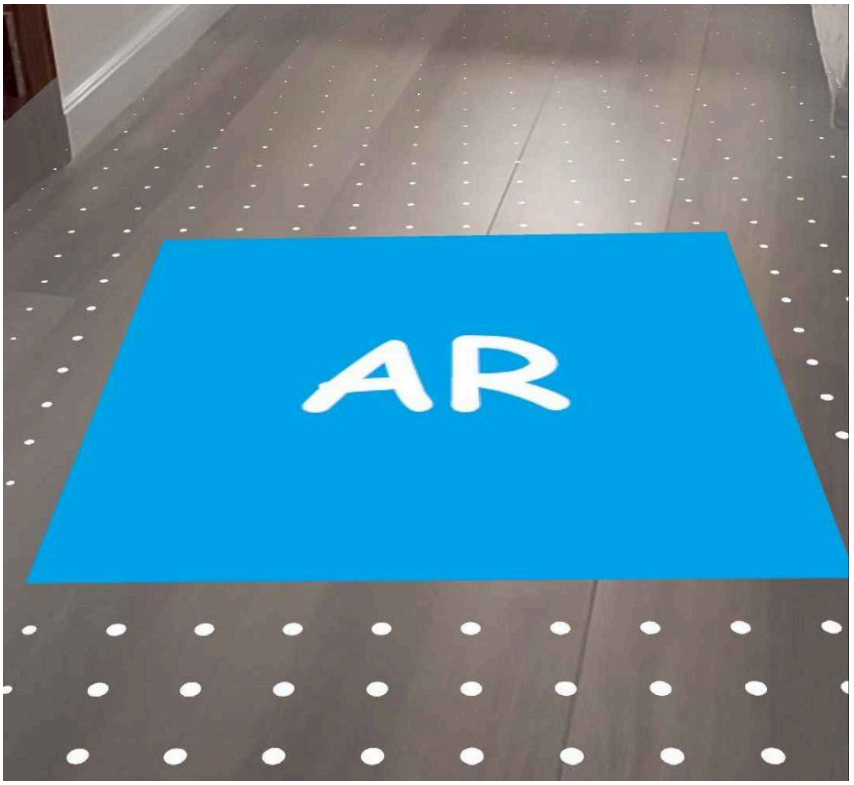


Ilustración 32. Nuestra textura, puesta por encima del piso detectado como plano. Imagen de nuestra autoría.

Con esta configuración ya tenemos una escena de muestra de las funciones 2D, en la cual utilizamos el código Scan Test para instanciar el objeto y rotarlo al igual que lo hacíamos en la escena 3D.

Teniendo estas 3 escenas, el siguiente paso es conectarlas a través de un menú principal, para esto volvemos a nuestro prototipo de menú y agregamos los botones faltantes, uno para cada escena y uno para salir de la aplicación. Para esto vamos a programar un código con 5 funciones, una por escena, una para volver al menú y una para salir de la aplicación, estas mismas funciones serán activadas al presionar uno de los botones.

Dentro del código podemos distinguir una función y una corrutina por cada botón, para controlar el cambio de escenas y un efecto visual al salir de la escena actual.

7.2.3.3. Menú Principal

```
public class MenuManager : MonoBehaviour
{
    [SerializeField]
    Animator fade;
    public void PlaneScene()
    {
        fade.SetTrigger("end");
        StartCoroutine(Plane2());
    }

    public void Scene3D()
    {
        fade.SetTrigger("end");
        StartCoroutine(Scene3D2());
    }
    public void Scene2D()
    {
        fade.SetTrigger("end");
        StartCoroutine(Scene2D2());
    }
    public void Menu()
    {
        fade.SetTrigger("end");
        StartCoroutine(Menu2());
    }
    public void Exit()
    {
        fade.SetTrigger("end");
        StartCoroutine(Exit2());
    }
    IEnumerator Plane2()
    {
        yield return new WaitForSeconds(0.25f);
        SceneManager.LoadScene("PlaneDetectTest");
    }
    IEnumerator Scene3D2()
    {
        yield return new WaitForSeconds(0.25f);
        SceneManager.LoadScene("3DSceneTest");
    }
    IEnumerator Scene2D2()
    {
```

```
        yield return new WaitForSeconds(0.25f);
        SceneManager.LoadScene("2DSceneTest");
    }
    IEnumerator Menu2()
    {
        yield return new WaitForSeconds(0.25f);
        SceneManager.LoadScene("MainMenu");
    }
    IEnumerator Exit2()
    {
        yield return new WaitForSeconds(0.25f);
        Application.Quit();
    }
}
```

Ilustración 33. El script responsable por las transiciones entre escenas. Imagen de nuestra autoría.

Una vez definidas, sólo basta con conectarlas con sus respectivos botones.

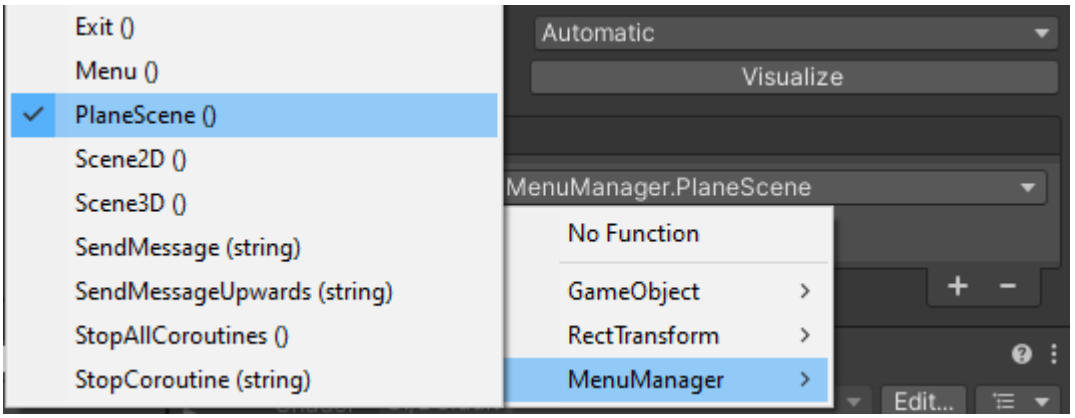


Ilustración 34. Demostración de conexión de las funciones con los botones. Imagen de nuestra autoría.

Luego de agregar un título para la aplicación y copiar los componentes del resto de escenas para utilizar la cámara del dispositivo como fondo (Menos los sistemas de detección, el objetivo siendo que la cámara esté inicializada desde el inicio de la aplicación), el resultado obtenido es el siguiente menú principal:

Ilustración 35. El menú completo, como fue propuesto en la ilustración 4. Imagen de nuestra autoría.

7.2.3.4. Interfaz de las escenas

Dentro de las escenas se agregaron textos y explicaciones gráficas a modo de tutorial para indicar al usuario cómo usar las funciones de la aplicación. Lo primero que vemos en todas las escenas es un texto que dice “Mueva su dispositivo apuntando a superficies planas”, de este modo se ejecuta la detección de planos, una vez encontrados el texto cambia dependiendo la escena: Detección de planos: “Planos detectados”; Muestra 3D: “Presione sobre el plano para generar un objeto”; Muestra 2D: “Presione sobre el plano para generar una imagen”.

En cuanto a tutoriales gráficos, se incorporaron 2 tutoriales que forman parte del complemento de XR Core y muestran la capacidad de mover el objeto y rotarlo, ambos tutoriales se pueden ver en la escena de Muestra 3D y el tutorial de rotación en la Muestra 2D.

Ilustración 36. Los tutoriales mencionados. Unity Technologies.

Además de los tutoriales, también se agregaron 2 botones nuevos a todas las escenas que permiten la navegación entre estas y la limpieza de las mismas, siendo estos una imagen de una casa para volver al menú principal y una papelera para borrar los objetos de la escena y poder colocarlos nuevamente.

Ilustración 37. Los iconos de menú principal y borrar objeto, respectivamente. Luca Burgio

Para el botón de volver al menú utilizamos lo que preparamos en el apartado anterior dentro del MenuManager:

```
public void Menu()
    {
        fade.SetTrigger("end");
        StartCoroutine(Menu2());
    }
IEnumerator Menu2()
    {
        yield return new WaitForSeconds(0.25f);
        SceneManager.LoadScene("MainMenu");
    }
```

Ilustración 38. La nueva función. Imagen de nuestra autoría.

En el caso del botón de papelera, se incorporó una función de borrado en el código ScanTest:

```
public void Delete()
    {
        Destroy(instanceofObjectPrefab);
        hitObjectPrefab = false;
        instanceofObjectPrefab = null;
    }
```

Ilustración 39. La lógica de la papelera. Imagen de nuestra autoría.

7.2.4. Milestone 4: Finalización del desarrollo y pulido

7.2.4.1. Optimización y Pruebas

Una vez desarrolladas todas las funciones de la aplicación, continuamos con un proceso de pruebas en varios dispositivos Android para detectar errores, fallas en el funcionamiento o problemas de optimización.

Tras realizar varias pruebas hemos notado 2 problemas principales, la acumulación de datos innecesarios o "basura" causada porque los planos generados no se borraban al cambiar de escena, debido a esto, el rendimiento de la aplicación se veía afectado, sin embargo, esto fue solucionado agregando una llamada a la función reset del código "AR Session" a la hora de utilizar el botón de volver al menú principal, borrando los datos basura en el proceso.

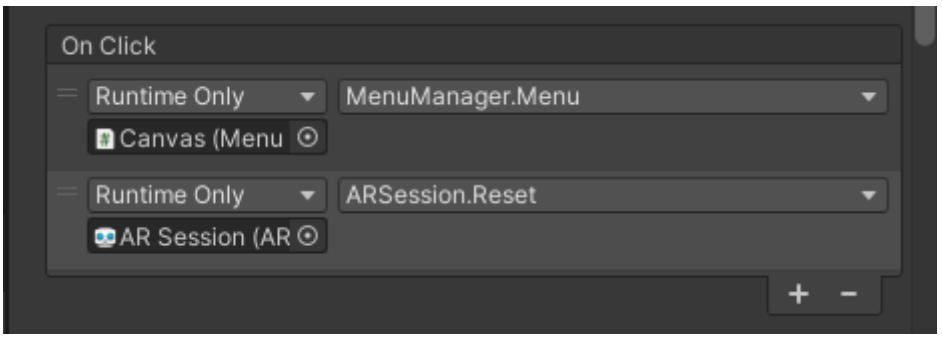
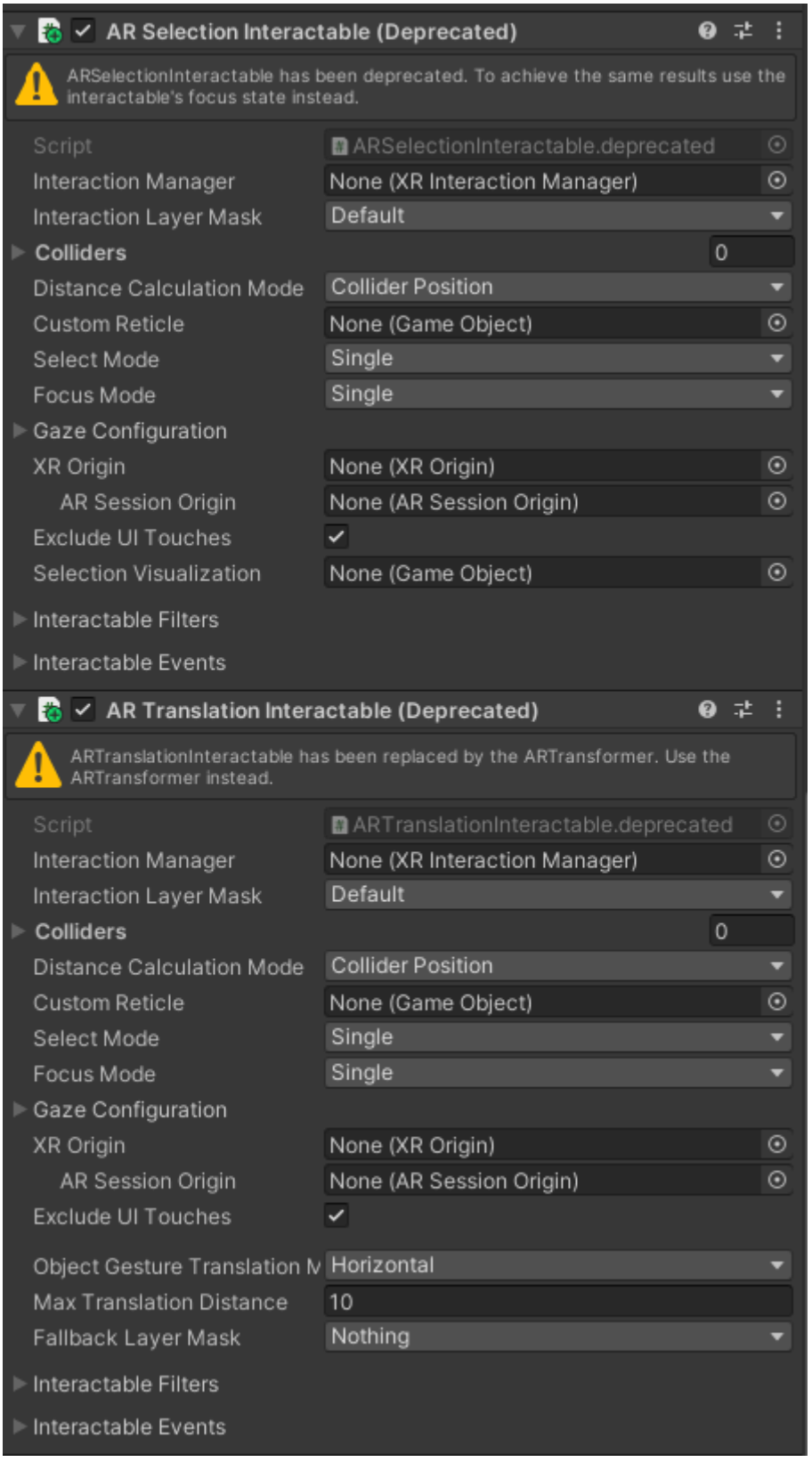


Ilustración 40. La llamada a la función Reset(), en la lógica del botón.

El segundo problema está relacionado con el comportamiento de los objetos generados en el plano, ya que con nuestro código inicial podíamos arrastrar los objetos 3D con algunos errores: velocidad inconsistente, generando una diferencia entre la posición del objeto y el dedo del usuario; al mantener pulsado luego de un movimiento, el objeto seguía moviéndose por su cuenta. Para solucionar estos comportamientos se realizaron varias pruebas reemplazando los cálculos realizados por el código a la hora de definir la posición del objeto.

Luego de muchas pruebas, se descubrió que este comportamiento funcionaba independientemente del código agregado, ya que esto es parte de una función por defecto de XR Core, que en versiones anteriores correspondía a un componente llamado "AR Selection Interactable", siendo mas especifico, a su variante "AR Translation Interactable", que se encargaba de mover el objeto una vez seleccionado, sin embargo, este sistema fue reemplazado durante el mes de agosto de 2023 con la salida de la versión 2.5.0 del XR Interaction Toolkit de Unity, que unificó los comportamientos de AR a un solo componente llamado "AR Transformer":


AR Selection Interactable (Deprecated)
ARSelectionInteractable has been deprecated. To achieve the same results use the interactable's focus state instead.
Script
ARSelectionInteractable.deprecated
Interaction Manager
None (XR Interaction Manager)
Interaction Layer Mask
Default
Colliders
0
Distance Calculation Mode
Collider Position
Custom Reticle
None (Game Object)
Select Mode
Single
Focus Mode
Single
Gaze Configuration
XR Origin
None (XR Origin)
AR Session Origin
None (AR Session Origin)
Exclude UI Touches
Selection Visualization
None (Game Object)
Interactable Filters
Interactable Events
AR Translation Interactable (Deprecated)
ARTranslationInteractable has been replaced by the ARTransformer. Use the ARTransformer instead.
Script
ARTranslationInteractable.deprecated
Interaction Manager
None (XR Interaction Manager)
Interaction Layer Mask
Default
Colliders
0
Distance Calculation Mode
Collider Position
Custom Reticle
None (Game Object)
Select Mode
Single
Focus Mode
Single
Gaze Configuration
XR Origin
None (XR Origin)
AR Session Origin
None (AR Session Origin)
Exclude UI Touches
Object Gesture Translation M
Horizontal
Max Translation Distance
10
Fallback Layer Mask
Nothing
Interactable Filters
Interactable Events

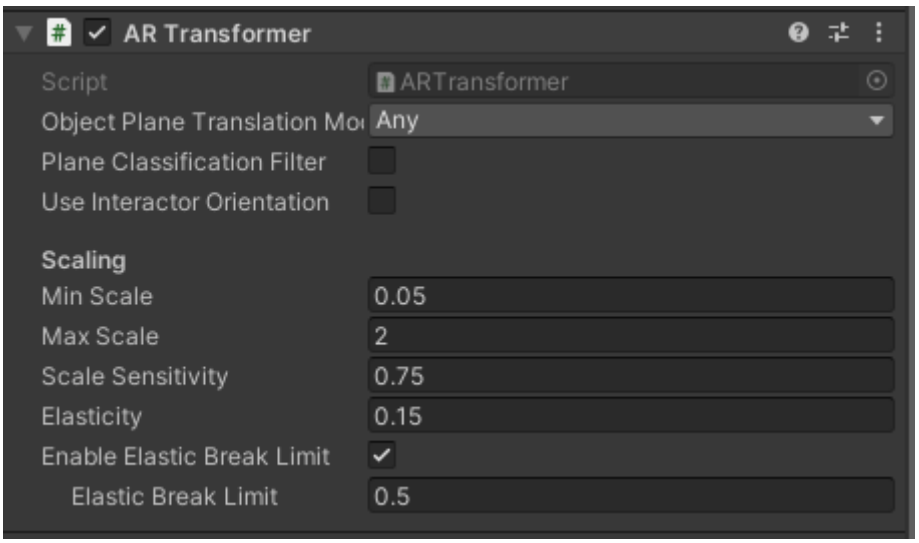


Ilustración 41. Las funciones obsoletas y su reemplazo, nótese la notificación usando el símbolo de advertencia informando de su obsolescencia. Imagen de nuestra autoría.

Al finalizar el desarrollo, deberíamos tener las siguientes escenas:

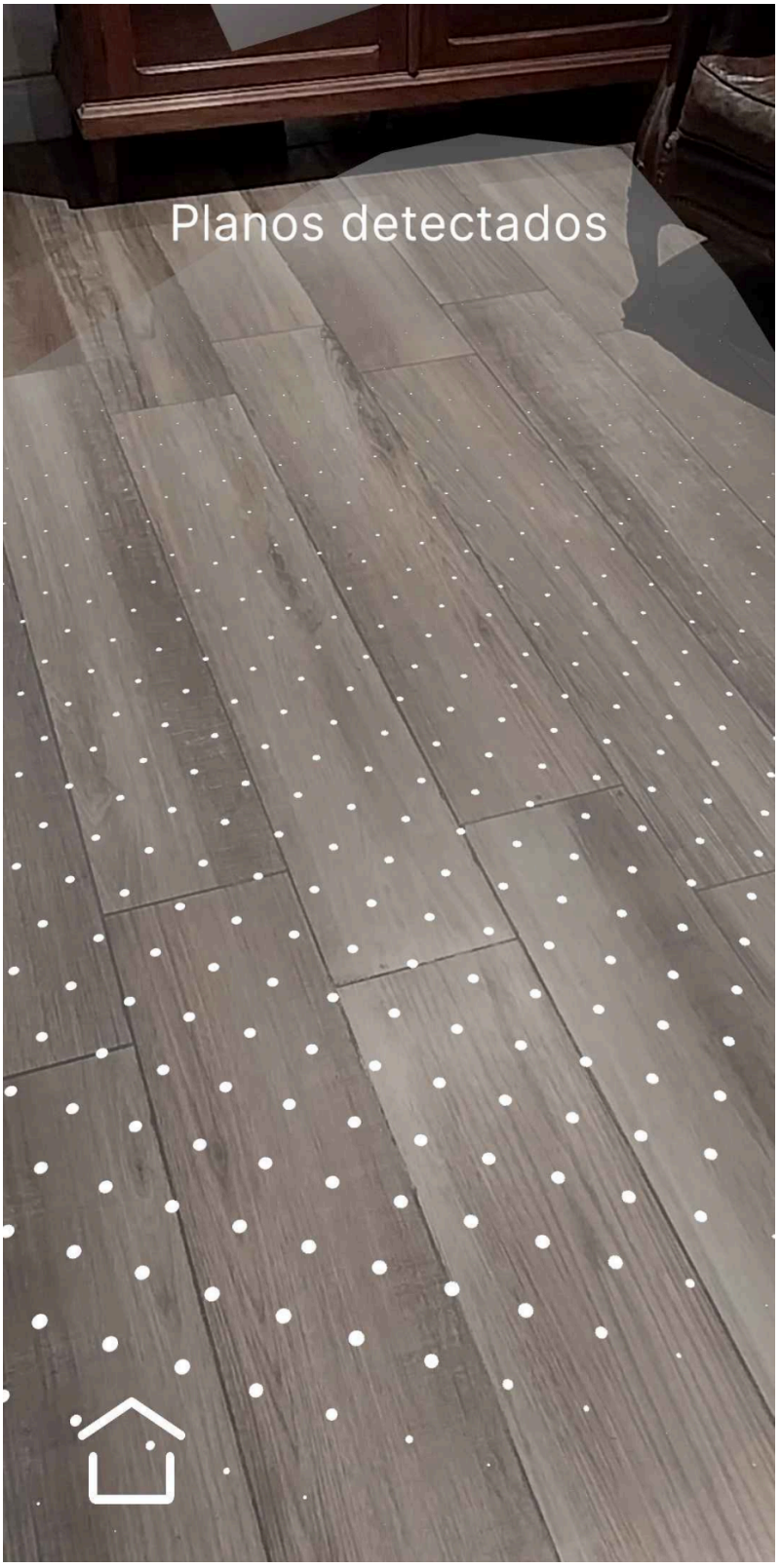


Ilustración 42. La escena finalizada de detección de planos. Imagen de nuestra autoría.

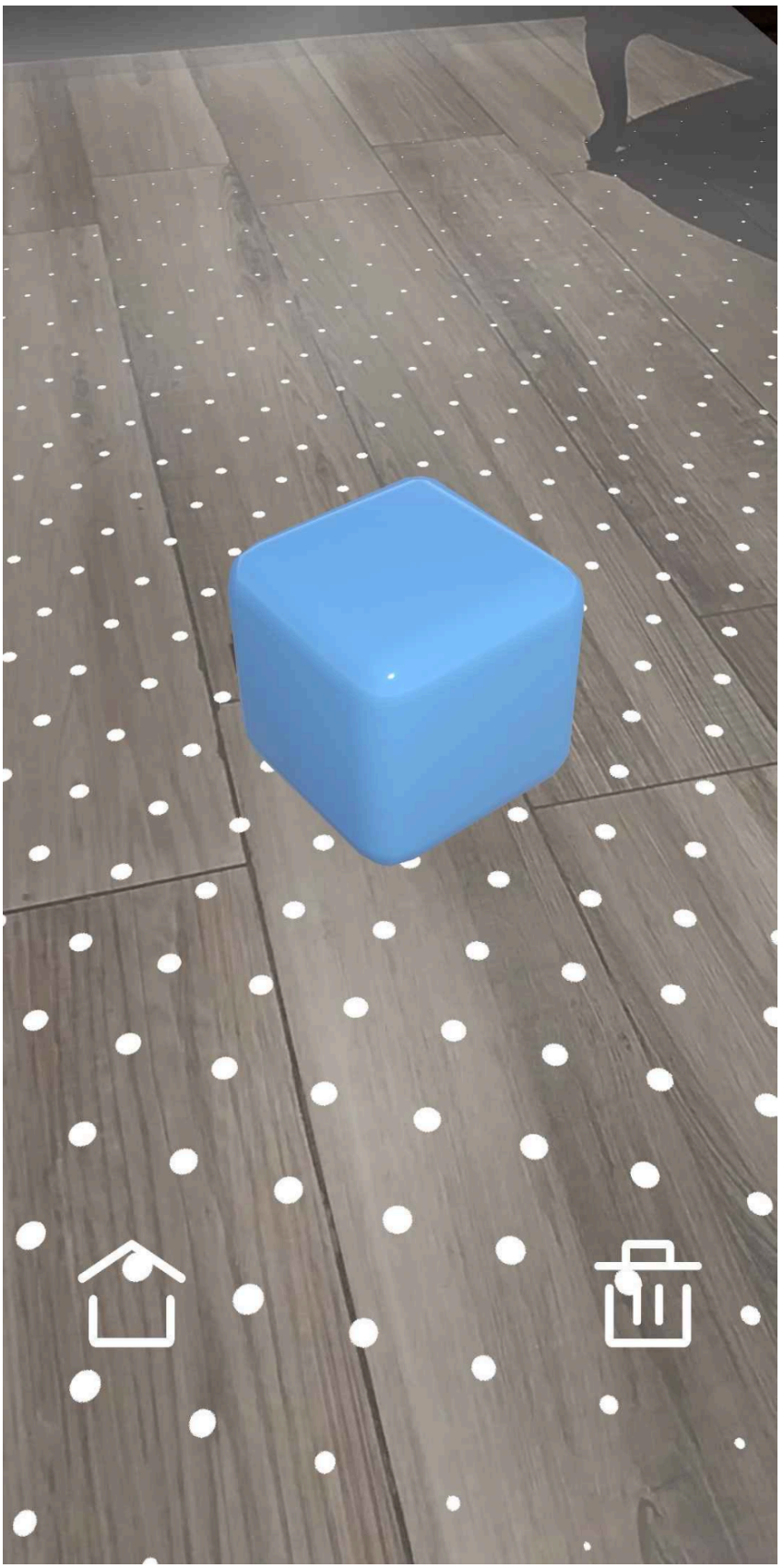


Ilustración 43. La escena finalizada de manipulación de objetos 3D. Imagen de nuestra autoría.

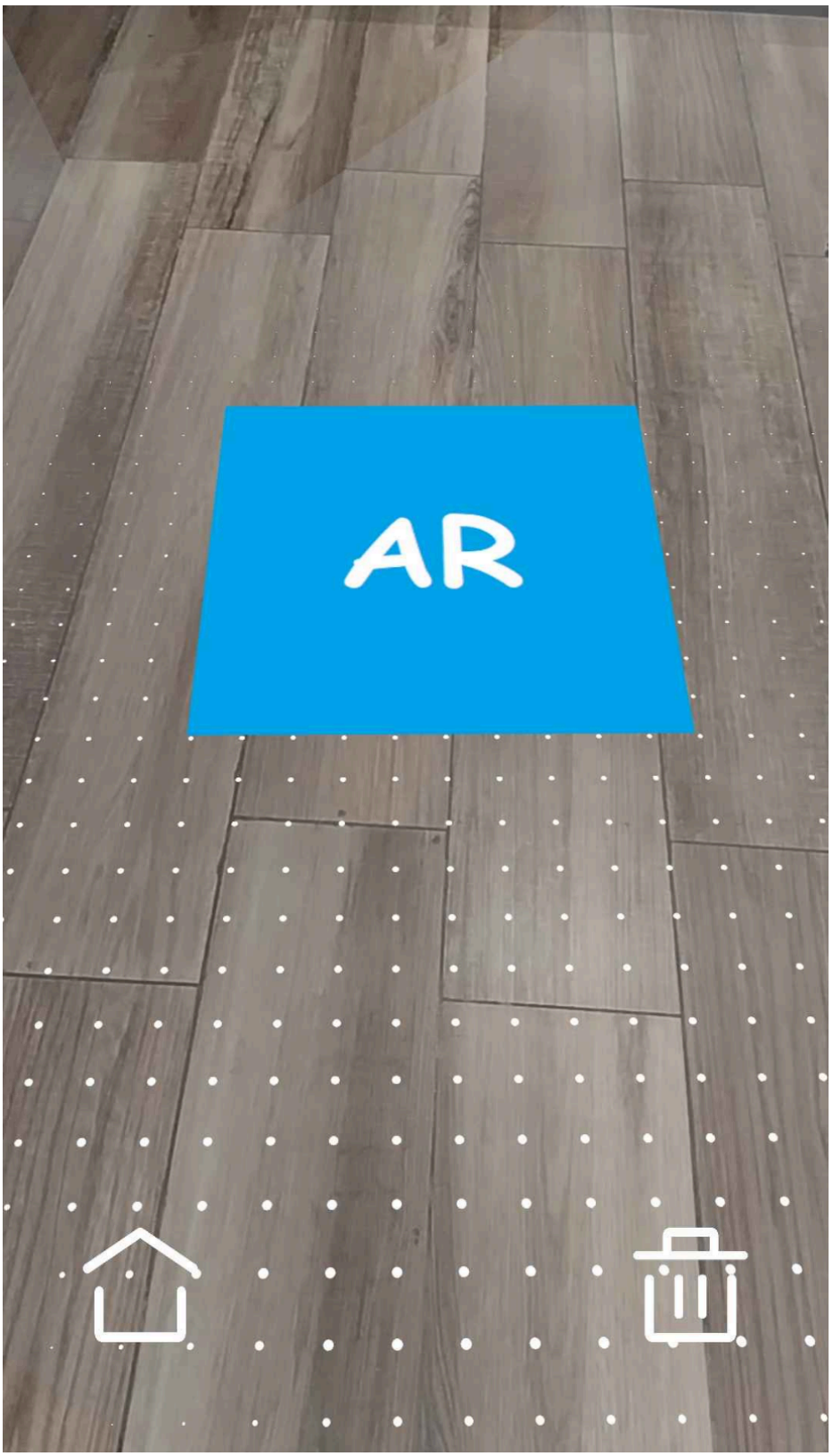


Ilustración 44. La escena finalizada de proyección de texturas 2D. Imagen de nuestra autoría.

8. Conclusión

Durante el desarrollo de esta investigación y del prototipo de aplicación de AR en Unity3D hemos navegado por varias fuentes, documentos y herramientas, explorando diversas definiciones de AR, los orígenes de esta tecnología y sus usos más comunes. Hemos recopilado e interpretado estos recursos para su plasmado en este método de desarrollo.

Uno de los principales aportes de este trabajo es la propuesta de un método estructurado que permite guiar a desarrolladores en la creación de aplicaciones AR de manera integral, abarcando desde la identificación de funciones esenciales hasta la implementación de un prototipo funcional. Este enfoque busca cubrir la carencia de documentación unificada sobre el desarrollo completo de aplicaciones AR en Unity, ofreciendo una alternativa accesible y detallada para desarrolladores de nivel intermedio.

El análisis comparativo de distintas aplicaciones de AR permitió identificar las funciones más relevantes y utilizadas en el mercado, lo que facilitó la definición de las características fundamentales del prototipo desarrollado. Como resultado, la aplicación implementada incluye funciones esenciales como la detección de planos, la visualización y manipulación de objetos 3D y 2D, y la integración de interfaces de usuario intuitivas.

El uso de Unity3D junto con el paquete XR Core demostró ser una solución eficiente para el desarrollo de aplicaciones AR, proporcionando herramientas que facilitan la implementación de funcionalidades avanzadas sin requerir desarrollos desde cero. No obstante, se identificaron desafíos en términos de optimización de rendimiento y la necesidad de un manejo eficiente de los recursos gráficos para garantizar una experiencia fluida.

Podemos decir con seguridad que la aplicación desarrollada es la culminación y resultado de las tareas mencionadas, logrando así un producto final que funciona como una demostración de las capacidades de la tecnología, como el resultado y/o consecuencia de seguir el método que hemos diseñado a través del cumplimiento de nuestros objetivos específicos. A futuro, existiría la posibilidad de agregar otras funciones, como la AR por marcadores, o nuevas interfaces como un menú de objetos 3D para utilizar en la escena.

En términos metodológicos, la combinación de observación de aplicaciones existentes con un diseño de prototipo basado en las funciones encontradas permitió validar la viabilidad del método propuesto. Además, el proceso de documentación paralela al desarrollo contribuyó a estructurar una guía clara y funcional que puede servir como referencia para futuros desarrollos.

Finalmente, esta investigación reafirma el potencial de la AR como una tecnología con aplicaciones en diversos sectores, desde el entretenimiento hasta la educación y la industria. El método presentado sienta las bases para futuras investigaciones y desarrollos en este campo, promoviendo la accesibilidad y la mejora continua en la creación de experiencias inmersivas con AR.

9. Bibliografía

- Ajenjo Jurado, Á. (2022). *Desarrollo de una aplicación móvil en Unity 3D con la API ARCore de Realidad Aumentada.* Archivo Digital UPM. https://oa.upm.es/69834/.
- Apple, Inc. (s.f.). *AR Apps and Games - App Store*. App Store. Recuperado el 1 de Octubre de 2024 de https://apps.apple.com/us/story/id1288297581.
- Beckman, J., Popa, A. (2023). *The Key Android Market Share Statistics for 2023*. The Tech Report. https://techreport.com/statistics/android-market-share-statistics/.
- Craig, A. B. (2013). *Understanding Augmented Reality: Concepts and Applications*. Elsevier Science.
- Edmunds Cars. (2017). *Augmented Reality Feature | Edmunds App for iPhone.* Youtube. https://youtu.be/VSJIFzO1J0w?si=ACdl5YHX-MTK5TbP.
- Feiner, S., Macintyre, B., & Seligmann, D. (1993). *Knowledge Based Augmented Reality*. ACM Digital Library. https://dl.acm.org/doi/pdf/10.1145/159544.159587.
- freeCodeCamp.org. (2022). *Augmented Reality for Everyone - Full Course*. https://www.youtube.com/watch?v=WzfDo2Wpxks.
- Furht, B. (Ed.). (2011). *Handbook of Augmented Reality*. Springer New York.
- Holland, P., Broida, R., Parker, J., Savvides, L. (2018). *The 23 best ARKit apps for iPhone and iPad*. CNET. https://www.cnet.com/pictures/best-ar-apps-for-ios-that-you-need-to-try.
- IKEA. (2017). *Launch of new IKEA Place app.* IKEA Global. https://www.ikea.com/global/en/newsroom/innovation/ikea-launches-ikea-place-a-new-app-that-allows-people-to-virtually-place-furniture-in-their-home-170912/.
- McDonald, B. (s.f.). *The 10 Best AR Apps, That Aren't Pokemon Go, Bringing Us to the Future*. Bentley University. https://www.bentley.edu/centers/user-experience-center/10-best-ar-apps-arent-pokemon-go-bringing-us-future.
- Microsoft. (s.f.). *What is augmented reality or AR?*. Recuperado el 14 de junio de 2024 de https://www.microsoft.com/en-us/dynamics-365/topics/augmented-reality/what-is-augmented-reality.
- Milgram, P., Kishino, F. (1994). *A Taxonomy of Mixed Reality Visual Displays*. George Mason University. https://cs.gmu.edu/~zduric/cs499/Readings/r76JBo-Milgram_IEICE_1994.pdf

- Pérez, F. (2023). *Pokémon GO se ha descargado casi 700 millones de veces desde su lanzamiento*. La Vanguardia. https://www.lavanguardia.com/andro4all/juegos/pokemon-go-se-ha-descargado-casi-700-millones-de-veces-desde-su-lanzamiento.
- Pineda, F. (2016). *Pokémon GO aterriza en móviles: un éxito de Nintendo que estaba cantado*. El Economista. https://www.eleconomista.es/tecnologia-videojuegos/noticias/7689454/07/16/Pokemon-GO-aterriza-en-moviles-un-exito-de-Nintendo-que-estaba-cantado.html.
- Porter, M. E., Heppelmann, J. E. (2017). *Why Every Organization Needs an Augmented Reality Strategy*. Harvard Business Review, Noviembre-Diciembre, 46-57. https://hbr.org/2017/11/why-every-organization-needs-an-augmented-reality-strategy?ab=seriesnav-spotlight.
- Roca, D. (2020). *Realidad Aumentada con Unity y Vuforia*. Escola Espai. https://www.espai.es/blog/2020/12/realidad-aumentada-unity-vuforia/.
- Servin, C., Naimark, M. (1968). *Espada de Damocles*. IDIS. https://proyectoidis.org/espada-de-damocles/.
- Statcounter Global Stats. (s.f.). *Android Version Market Share Worldwide.* StatCounter Global Stats. Recuperado el 23 de mayo de 2024 de https://gs.statcounter.com/os-version-market-share/android.
- TECHcetera. (2018). *Ikea Place - App de Realidad Aumentada para el iPad | Noticias tecnológicas*. Youtube. https://youtu.be/ALpVlVsH6M8?si=HpiCtPyoFf3HDKv4.
- UnderRev. (2017). *Porsche AR demo.* Youtube. https://youtu.be/BpqwJK6kjug?si=fwOoJRgaOKO_UGLd.
- Unity Technologies. (s.f.). *Mobile AR Development*. Unity Learn. Recuperado el 31 de mayo de 2024 de https://learn.unity.com/pathway/mobile-ar-development?uv=2021.3.
- Unity Technologies. (s.f.). *Software 3D para arquitectura, ingeniería y construcción*. Unity. Recuperado el 14 de Junio de 2024 de https://unity.com/es/solutions/architecture-engineering-construction.
- Unity Technologies. (s.f.). *Plataforma de desarrollo en tiempo real de Unity | Motor de 3D, 2D, VR y AR*. Unity. Recuperado el 23 de Mayo de 2024 de https://unity.com/es.

- Unity Technologies. (2023). *[2.5.0] - 2023-08-17 Changelog*. Unity Docs. https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit%402.5/changelog/CHANGELOG.html#250---2023-08-17.
- Valderrama, I., Valderrama, R., Rivera, K. (2014). *Interacción de modelos 3D con realidad aumentada*. Sistemas, Cibernética e Informática. https://www.iiisci.org/journal/pdv/risci/pdfs/xa349tl14.pdf.
- Valencia-García, R., Lagos-Ortiz, K., Alcaraz-Mármol, G., del Cioppo, J., Vera-Lucio, N. (Eds.). (2016). *Technologies and Innovation: Second International Conference, CITI 2016, Guayaquil, Ecuador, November 23-25, 2016, Proceedings*. Springer International Publishing.
- Vázquez, I. (2023). *Las 15 mejores aplicaciones con realidad aumentada*. App Marketing News. https://appmarketingnews.io/las-15-mejores-aplicaciones-con-realidad-aumentada/.

10. Licencias / Terceros

Repositorio Github: https://github.com/alva531/XRexperience

Fuente Logo: https://www.dafont.com/es/spaceport-2006.font?l[]=10&back=theme

Assets UI: https://prinbles.itch.io/analogue-buttons-pack-i

Icon Trash: https://iconduck.com/icons/125292/trash

Icon Home: https://iconduck.com/icons/124753/home

HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.