



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Desarrollo de aplicación para la identificación de
embarcaciones y elementos del entorno marítimo
mediante realidad aumentada*

Grado en Ingeniería Mecánica

ALUMNO: Ángel Gamboa Ramos Izquierdo

DIRECTORES: Miguel Rodelgo Lacruz

Xavier Núñez Nieto

CURSO ACADÉMICO: 2022-2023

Universida_{de}Vigo



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Desarrollo de aplicación para la identificación de
embarcaciones y elementos del entorno marítimo
mediante realidad aumentada*

Grado en Ingeniería Mecánica
Intensificación en Tecnología Naval
Cuerpo General

UniversidadeVigo

RESUMEN

En la actualidad, la realidad aumentada está ganando una gran importancia en diferentes ámbitos, como en la industria marítima, donde es importante tener una visión clara de la situación.

El objetivo principal de este trabajo de fin de grado es desarrollar una aplicación de realidad aumentada. Se busca que permita visualizar la información de las embarcaciones, tales como su nombre, bandera, tipo de buque y la distancia al mismo, y también explorar la posibilidad de ofrecer información geográfica y otros datos relevantes. Para ello, se llevará a cabo una comparativa de los diferentes métodos de posicionamiento y herramientas de desarrollo, como las librerías de realidad aumentada y las plataformas de desarrollo, para seleccionar las más adecuadas para el proyecto.

El resultado esperado es un prototipo de herramienta que brinde una representación visual de la información marítima en tiempo real, lo que permitirá a los usuarios colocar marcadores en zonas preestablecidas. Finalmente, se evaluará su funcionamiento en el entorno real para validar su precisión y funcionamiento. Este trabajo de fin de grado busca aportar a la industria marítima una herramienta que permita una mejor gestión y visualización de la información marítima, mejorando la seguridad y eficiencia en el sector.

Además, se busca indagar sobre las posibles aplicaciones que tiene la realidad aumentada en la Armada. Como ideas obtenidas se ha realizado un desarrollo de posibles aplicaciones en: operaciones quirúrgicas en la mar, canales dragadas, operaciones de buceadores, seguridad interior y en un modelo digital para la futura F-110.

PALABRAS CLAVE

Realidad aumentada, AIS, plataforma de desarrollo, pecios, Unity.

AGRADECIMIENTOS

A mi familia. En especial a mi padre, por ser el mayor ejemplo que tengo en mi vida. Muchas gracias por todo.

A mis compañeros de promoción. Sois un apoyo incondicional en mi vida y os considero como a una segunda familia. De todo corazón, muchas gracias.

En especial a mis amigos Garau, Fefe y María. Por confiar en mí mucho antes de que yo lo hiciese. Sin vosotros en especial siento que yo hoy no estaría aquí. En especial a vosotros tres, muchas gracias.

A mis tutores de este proyecto, el profesor Rodelgo y el profesor Nuñez, y a todos mis profesores del CUD, en especial a aquellos que en algún momento se preocuparon no solo por mi formación, sino también por mi persona. Quiero que sepan que siempre fui consciente de su dedicación y quiero agradecerles de todo corazón.

Por último, quiero le agradecer a mi abuelo y a mi amigo Luisete. Por cuidar de mí desde ahí arriba. Se que hoy están de celebración al ver que, después de nunca rendirme y siempre seguir adelante, estoy cumpliendo mi sueño. Esta victoria va por los dos. Muchas gracias por todo.

CONTENIDO

Contenido	1
Índice de Ilustraciones	3
Índice de Tablas.....	5
1 Introducción y objetivos	7
1.1 Realidad aumentada	7
1.2 AIS y proyecto CEMAI/SIRENA.....	7
1.3 Proyecto y objetivos.....	8
1.4 Estructura de la memoria.....	8
2 Estado del arte	11
2.1 La Realidad aumentada	11
Fundamentos de la realidad aumentada	11
2.2 Posibles aplicaciones de la AR en la Armada	13
2.1.1 Canales dragadas	13
2.2.1 Hombre al agua.....	14
2.2.2 Carros de combate	15
2.2.3 Emergencias en puerto.....	15
2.2.4 Operaciones quirúrgicas	16
2.2.5 Unidades de buceo	17
2.2.6 El gemelo digital.....	19
2.2.7 Situación marítima a partir de datos AIS.....	20
2.2.8 Equipos de realidad aumentada.....	21
2.3.1 Gafas de realidad aumentada	21
2.3.2 Teléfonos móviles.....	23
2.3.3 Conclusión sobre los equipos	24
2.3.4 2.4 Plataformas de desarrollo.....	24
2.4.1 Android Studio.....	24
2.5.1 Unity	26
2.5.2 Unreal Engine	27
2.5.3 Comparativa de Plataformas	29
2.5 Librerías de realidad aumentada	31
ARcore	31
ARKit.....	32
Vuforia Engine.....	33

Tabla resumen.....	34
.....	34
Conclusión final.....	35
3 Desarrollo del TFG.....	37
3.1 Descripción del apartado.....	37
2.5.4 Configuración e inicio con Unity	37
2.5.5 Script VPSManager	40
Creación del marcador	44
3.1.1 Poner el marcador en AR.....	49
3.1.2 Solución al error	50
3.1.3 Mejoras del script Demo.....	54
3.1.4 Mejora del marcador	57
3.1.6 Marcadores de faros y peligros	62
3.1.7 Problema con el Street view	64
3.1.9 Mejora de aspecto e implementación de tipo de buque	69
3.1.10	
4 Resultados y Validación.....	73
4.1 Pruebas de la aplicación	73
4.1.1 Embarcaciones	73
4.1.2 Peligros y faros	74
4.2 Validación	77
5 Conclusiones y líneas futuras	79
5.1.1 5.1 Conclusiones	79
5.1.2	
5.1.3 Estado del arte.....	79
La aplicación.....	80
Evaluación de la aplicación	80
5.2 Líneas futuras	80
6 Bibliografía.....	82
Anexo I: Implicaciones Sociales, y/o Económicas, y/o Ambientales	83
Anexo II: Reflexiones Éticas y Sociales	85
Anexo III: Contenido del tercer anexo	¡Error! Marcador no definido.

ÍNDICE DE ILUSTRACIONES

Ilustración 1 ejemplo de canal dragada	13
Ilustración 2 logo de Android Studio	25
Ilustración 3 Logotipo de la compañía Unity	26
Ilustración 4 logo de Unreal	28
Ilustración 5 Captura de pantalla de Marine traffic.....	36
Ilustración 6 cambio a Android	37
Ilustración 8 interfaz Unity.....	39
Ilustración 9 API key.....	40
Ilustración 10 código VSPManager completo.....	41
Ilustración 11 código VPSManager (1).....	42
Ilustración 12 código VPSManager (2).....	42
Ilustración 13 código VPSManager (3).....	42
Ilustración 14 código VPSManager (4).....	43
Ilustración 15 código VPSManager (4).....	44
Ilustración 16 codigo texto	45
Ilustración 17 textos Unity	46
Ilustración 18 primer diseño de marcador	46
Ilustración 19 Cambio a Sprite	47
Ilustración 20 interfaz Unity.....	48
Ilustración 21 marcador con textos.....	49
Ilustración 22 captura de proyecto	49
Ilustración 23 prueba AR con marcador.....	50
Ilustración 24 declaracion del Target	50
Ilustración 25 código posición del taget.....	50
Ilustración 26 captura Cube en VPSManager	51
Ilustración 27 campos del marcador.....	52
Ilustración 28 segunda prueba con marcador en AR.....	52
Ilustración 29 código para posicionar	53
Ilustración 30 tercera prueba del marcador en AR.....	53
Ilustración 31 mejora script Demo	54
Ilustración 32 captura 1 Script Demo.....	55
Ilustración 33 captura 2 Script Demo.....	55
Ilustración 34 captura 3 Script Demo.....	55

Ilustración 35 prueba de mejora de marcador en AR	56
Ilustración 36 Nuevo marcador	57
Ilustración 37 marcador nuevo en AR.....	58
Ilustración 38 código sobre distancia	58
Ilustración 39 diccionario de banderas	59
Ilustración 40 código identificación bandera	59
Ilustración 41 interfaz de unity con banderas	61
Ilustración 42 nuevos marcadores	62
Ilustración 43 código marcadores nuevos	63
Ilustración 44 bajo de Morrazán.....	64
Ilustración 45 proyección mercator	66
Ilustración 46 foto comparativa morrazán.....	67
Ilustración 47 VPSManger nuevo Parte 1	68
Ilustración 48 VPSManger nuevo Parte 2	69
Ilustración 49 marcador definitivo	70
Ilustración 50 código de iconos de buque parte 1	70
Ilustración 51 código de iconos de buque parte 2	70
Ilustración 52 código de iconos de buque parte 3	71
Ilustración 53 Tabarca	73
Ilustración 54 Tambo.....	74
Ilustración 55 tambo comparativa	74
Ilustración 56 carguero atracado	75
Ilustración 57 comparativa carguero	75
Ilustración 58 Giralda atracado	76
Ilustración 59 comparativa Giralda	76

ÍNDICE DE TABLAS

Tabla 2-1 comparativa de los 3 softwares	30
Tabla 4 comparativa de librerías	35

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Realidad aumentada

La realidad aumentada es una tecnología emergente que está comenzando a tener un impacto significativo en muchos campos y sectores. Esta tecnología se basa en la superposición de información digital en el mundo real, lo que permite crear experiencias inmersivas y enriquecedoras para los usuarios. La realidad aumentada funciona mediante el uso de dispositivos como cámaras, sensores y software especializado para superponer contenido digital en el mundo real. Estos dispositivos pueden ser teléfonos inteligentes, tabletas, gafas inteligentes y otros dispositivos que permitan la captura de imágenes y la visualización de información en tiempo real.

La tecnología de realidad aumentada funciona en tres etapas principales: la detección, el seguimiento y la superposición. En la detección, el dispositivo identifica y reconoce el objeto del mundo real sobre el que se superpondrá el contenido digital. En el seguimiento, el dispositivo realiza un seguimiento del objeto en tiempo real para mantener la superposición precisa incluso si el objeto se mueve. En la superposición, el contenido digital se superpone al objeto en el mundo real y se muestra en la pantalla del dispositivo.

La realidad aumentada también puede usar la geolocalización para superponer información digital en un lugar específico del mundo real. Por ejemplo, una aplicación de realidad aumentada puede superponer información sobre un monumento histórico o un punto de interés turístico en la pantalla del dispositivo cuando se está en ese lugar.

1.2 AIS y proyecto CEMAI/SIRENA

En el proyecto CEMAI/SIRENA se ha desarrollado un demostrador que procesa el flujo de datos AIS transmitidos por los buques, junto con otras fuentes de información, para obtener una imagen del entorno marítimo en tiempo real. Cada barco equipado con un sistema AIS puede transmitir su identificación, posición, velocidad y rumbo a través de un enlace de radio VHF. Estos mensajes son

recibidos por otros barcos y puertos que también están equipados con el sistema, lo que permite compartir información en tiempo real sobre su situación.

1.3 Motivación y objetivos

Imaginemos una aplicación de realidad aumentada que integre la información del sistema AIS junto con otras fuentes de información, como el radar y el GPS, para proporcionar una vista en tiempo real del entorno marítimo a los usuarios. Esta aplicación podría mostrar información detallada sobre los barcos cercanos, su posición, velocidad, rumbo y otra información relevante, lo que permitiría a los usuarios navegar de manera más segura y eficiente.

Por consecuencia, en este proyecto se plantean varios objetivos que se centran en la investigación, el desarrollo y la evaluación de una aplicación de realidad aumentada para su uso en el entorno marítimo. A continuación, se detallan los pasos necesarios para cumplir con cada uno de los objetivos establecidos:

1. El primer objetivo específico es investigar el estado del arte de la realidad aumentada. Esto implica revisar los últimos avances en dispositivos, software y plugins de realidad aumentada. Esta investigación permitirá conocer las últimas tendencias en realidad aumentada y explorar nuevas posibilidades de uso en el ámbito naval.
2. El segundo objetivo específico es desarrollar una aplicación de realidad aumentada que represente sobre los buques un marcador con información de interés, como el nombre, la nacionalidad y el tipo de buque. Además, la aplicación deberá marcar sitios como faros, pecios o cables submarinos. Para lograr este objetivo, se necesitará desarrollar una aplicación que pueda geolocalizar los marcadores y mostrar la información relevante. También se necesitará identificar los sitios que deben marcarse y obtener la información necesaria para mostrarla en la aplicación. Se busca el desarrollo de una aplicación capaz de identificar buques y pecios en tiempo real mediante la conexión con un servidor de Elastic Search
3. El tercer objetivo específico es probar la aplicación y evaluar su eficacia. Para esto, se debe realizar una serie de pruebas en un entorno controlado y en condiciones reales. Las pruebas permitirán evaluar la precisión y la fiabilidad de la aplicación, así como identificar posibles problemas y áreas de mejora.

1.4 Estructura de la memoria

1. **Estado del arte:** en este apartado se indaga acerca del funcionamiento de la realidad aumentada. Además, se estudian las plataformas de desarrollo de Unity, Android Studio y Unreal para ver cual es el mejor para la realización de este proyecto. También se estudia las posibles aplicaciones de la realidad aumentada en el ámbito naval.

2. **Desarrollo:** en este apartado se ha descrito como ha sido la realización de la aplicación. He descrito el proceso y se han recalcado los errores que han ido apareciendo y como se han ido resolviendo.
3. **Validación, resultados y pruebas:** en este apartado se han expuesto los resultados de la aplicación terminada. Se han mostrado varios ejemplos de los resultados y se han comparado con imágenes de google maps para estimar su precisión.
4. **Conclusiones y líneas futuras:** en este último apartado se ha hecho una evaluación de los objetivos propuestos en la introducción. Se han sacado conclusiones al respecto y se ha evaluado su eficacia. Finalmente se han propuesto unas líneas futuras sobre este proyecto y sus posibles mejoras.

2 ESTADO DEL ARTE

2.1 La Realidad aumentada

Fundamentos de la realidad aumentada

2.1.1

La realidad aumentada es la combinación del mundo virtual y el mundo físico a través de dispositivos digitales. De esta forma conseguimos representar objetos virtuales de manera holográfica dando la sensación de que el objeto realmente existe en el mundo físico. Además de la representación de objetos se pueden realizar interacciones en tiempo real.

Las aplicaciones de AR sin localización geoespacial funcionan independientemente del lugar donde se encuentre el usuario y pueden superponer objetos virtuales en cualquier entorno que la cámara del dispositivo pueda capturar. Estas aplicaciones se utilizan en una variedad de campos, desde juegos hasta educación y publicidad.

Por otro lado, las aplicaciones de AR con localización geoespacial utilizan la posición del usuario para colocar objetos virtuales en ubicaciones específicas del mundo real. Estas aplicaciones pueden mostrar información adicional sobre puntos de interés o lugares específicos, y se utilizan comúnmente en aplicaciones de turismo o exploración.

La localización geoespacial se logra mediante el uso de sistemas de posicionamiento global (GPS) o la triangulación de señales de redes inalámbricas para determinar la ubicación del usuario. Cuando la aplicación de AR con localización geoespacial se utiliza, se identifica la ubicación del usuario y se superponen objetos virtuales en el mundo real en una ubicación específica.

El ejemplo más conocido de aplicación de AR geoespacial es Pokemon Go. Este juego nos permite traer las criaturas del mundo Pokemon a la realidad, dando la impresión de que se encuentran entre nosotros y que la única forma de verlos es a través de nuestro dispositivo móvil.



Ilustración [5] Pokemon go

La realidad aumentada es una experiencia interactiva que permite a los usuarios ver y manipular objetos virtuales en su entorno real, y es utilizada en una variedad de aplicaciones, desde juegos hasta educación y publicidad.

Para conseguir esto hace falta una serie de componentes, los más comunes son:

1. **Software de realidad aumentada:** son las herramientas que facilitan el acceso a las aplicaciones de AR. Permiten desarrollar este tipo de aplicaciones. Más adelante veremos los ejemplos de las tres plataformas de desarrollo que he comparado para la realización de este proyecto, estos son Android Studio, Unity y Unreal. Veremos también los pluggins que usaremos. Un plugin de AR se integra en una aplicación existente para habilitar funciones de realidad aumentada.
2. **Un dispositivo con cámara:** este dispositivo deberá tener cámaras con las que captar imágenes del mundo real. Los más comunes son los teléfonos móviles y las gafas de realidad virtual. Veremos algunos ejemplos más adelante de ese último.
3. **Un objeto de referencia:** uno real de referencia que nos sirva para la creación de uno virtual. En la realidad aumentada geoespacial, el objeto de referencia puede ser una posición geoespacial y una orientación específicas en el mundo real, en lugar de un objeto físico. También existen otras formas de objetos de referencia para la AR geoespacial, como códigos QR, marcadores visuales o puntos de interés específicos que se pueden identificar mediante la cámara del dispositivo.

Otro elemento que está ganando importancia en este ámbito es el de las inteligencias artificiales. Estas ayudan al procesamiento de la información del software y pueden dar la opción al usuario a de instruir por voz.

La realidad aumentada es por tanto una herramienta para diseñar y dar instrucciones en el mundo real al ofrecer la posibilidad de superponer objetos virtuales sobre objetos físicos.

A continuación, voy a elucubrar en varios campos para la realización de este proyecto. En primer lugar, voy a decidir que plataforma de desarrollo voy a utilizar. Luego pensaré que dispositivo es el más adecuado y por último escogeré un plugin específico de realidad aumentada.

2.2 Posibles aplicaciones de la AR en la Armada

La Realidad Aumentada (AR) es una tecnología emergente que tiene el potencial de transformar la forma en que la Armada Española lleva a cabo sus operaciones. Como ya hemos explicado antes, la AR permite superponer información digital en el mundo real, lo que permite a los usuarios interactuar con la información de una manera más intuitiva y natural.

En la Armada Española, la AR puede ser utilizada en una variedad de aplicaciones que van desde la formación de dotaciones, la planificación de operaciones, la navegación, la identificación de objetivos y la gestión de mantenimiento hasta para mejorar la seguridad y la eficiencia en el transporte marítimo y la defensa costera. Así pues, la AR ofrece una gran oportunidad para mejorar la capacidad operativa de la Armada Española y para aumentar su eficiencia en el futuro.

A continuación, vamos a desarrollar una serie de ideas que enfocan el uso de la realidad aumentada en el ámbito naval:

2.2.1

Canales dragadas

Durante situaciones de guerra, una de las tácticas navales más utilizadas es la negación de un área al enemigo mediante la colocación de minas. Para que el enemigo no pueda entrar, por ejemplo, en la ría de Pontevedra, se colocan minas en toda la zona excepto en unos puntos determinados que solo nosotros conocemos. Esto se hace obviamente para que solo nosotros seamos capaces de entrar y salir del puerto con seguridad. A esta derrota segura que debemos seguir para no pasar por ninguna mina se la conoce como canal dragado. Para realizar esta derrota se realiza lo que llamamos navegaciones de

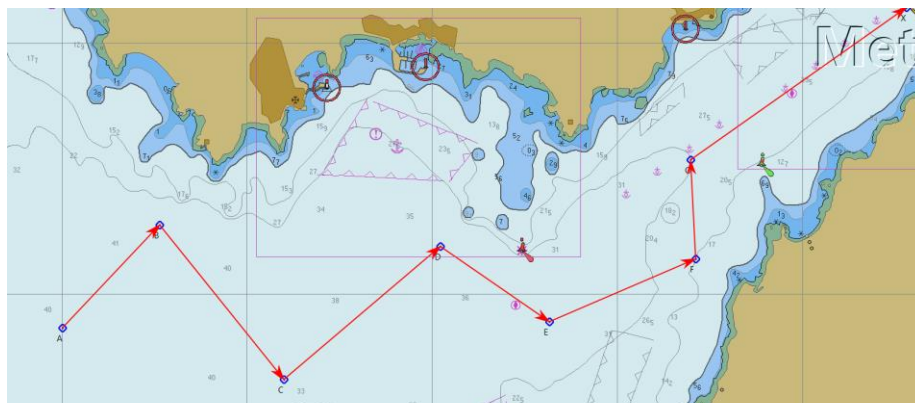


Ilustración 1 ejemplo de canal dragado

precisión, en las que, como su propio nombre indica, debemos ser extremadamente meticulosos a la hora de pasar exactamente por las zonas libres de minas.

Para crear estas canales dragas se toman puntos GPS y se unen. A continuación se da de margen un número de yardas a ambas bandas con respecto al eje del canal. De esta forma el oficial de guardia en puente (encargado de maniobrar el buque) debe conocer cuanta distancia le queda para llegar al próximo punto y cuanto está desviado del eje. Esta información se la da el personal adiestrado del puente del buque. Estas navegaciones de precisión son parte del adiestramiento de los alumnos de la Escuela Naval durante toda nuestra estancia aquí, más especialmente en cuarto año de carrera puesto que es cuando tomamos la responsabilidad de ser oficiales de guardia en puente.

Imaginemos ahora que el oficial de guardia en puente tuviese un dispositivo de realidad aumentada el cual superpusiese la canal dragada, con su eje, los límites a cada banda y los puntos donde debe caer de rumbo de cada tramo de la canal. Además de esto se le podría añadir la distancia exacta al siguiente punto de caída, la velocidad a la que va el buque y otras informaciones de interés. Utilizando sensores y cámaras, los oficiales de la Armada pueden superponer información en tiempo real sobre el entorno marítimo, como la posición de las minas y la distancia segura a las mismas. Esto les permite tomar decisiones más rápidas y precisas, y aumenta significativamente la seguridad de las navegaciones en zonas minadas. Además, la realidad aumentada también puede ser utilizada en la formación de los oficiales, permitiéndoles practicar maniobras de evasión de minas en un entorno simulado antes de enfrentarse a situaciones reales.

Sin duda es una aplicación muy interesante de AR en el ámbito naval, ya que las navegaciones de precisión se realizan con frecuencia. No solo para el adiestramiento en caso de minado, sino también porque estas navegaciones son muy aplicadas en entradas de puerto donde hay aguas someras o bajos que implican un peligro para la navegación. Por hacer la comparación, esos bajos podrían equivaler a las minas, por lo que también podrían ser indicados al oficial de guardia en puente a través del dispositivo de realidad aumentada.

2.2.2

Hombre al agua

Una situación de hombre al agua en la mar es una de las emergencias menos deseadas por cualquier oficial de guardia en puente. La rapidez y precisión en el rescate del náufrago es crucial para garantizar su seguridad y supervivencia. Es una situación de estrés que requiere la máxima concentración de la dotación puesto que la vida del náufrago está en juego y cualquier error puede ser catastrófico. Por lo tanto, es esencial estar preparado y tener protocolos eficientes para manejar estas situaciones de emergencia de manera sobresaliente.

Existen una serie de procedimientos doctrinales que enseñan a las dotaciones cómo deben ser las actuaciones y reacciones inmediatas en el caso de un hombre al agua. Existen también una serie de maniobras que ayudan a invertir el rumbo del buque y así facilitar la búsqueda del náufrago. Pero preguntemos, ¿qué sucede si esta emergencia ocurre después del ocaso cuando ya sea noche cerrada? La búsqueda se complica de manera exponencial cuanto menos visibilidad tengamos. Y pensemos que también las condiciones meteorológicas pueden ser adversas, lo que va a dificultar gradualmente un rescate.

Imaginemos ahora que el oficial de guardia en puente en el momento de hombre al agua dispone de unas gafas que con solo pulsar un botón guardan la posición GPS pudiendo ser utilizada la realidad aumentada en estas situaciones para brindar información en tiempo real sobre la posición del naufrago, así como para guiar al equipo de rescate hasta su ubicación de manera eficiente. Además, también se podría utilizar para visualizar información sobre las condiciones de la mar y el clima en la zona, lo que permitiría tomar decisiones mucho más eficazmente y en menor tiempo aumentando así las posibilidades de éxito en el rescate.

Carros de combate

La realidad aumentada (AR) puede ayudar en el entrenamiento y uso de carros de combate. Esta aplicación estaría enfocada al cuerpo de Infantería de Marina y sería una integración de la AR en los carros de combate para aplicar una serie de mejoras que incluirían:

1. Entrenamiento: AR puede ser utilizada para brindar un entrenamiento más inmersivo y realista a los operadores de carros de combate. Por ejemplo, la AR puede simular situaciones de combate o entornos realistas para permitir que los operadores practiquen en un entorno seguro antes de enfrentarse a situaciones reales.
2. Mantenimiento: AR puede ser utilizada para ayudar en el mantenimiento y reparación de carros de combate. En concreto, la AR puede mostrar información sobre las piezas y componentes del carro de combate, así como instrucciones paso a paso para realizar tareas de mantenimiento y reparación.
3. Misión: AR puede ser utilizada para ayudar a los operadores de carros de combate en su misión. Por poner un ejemplo, la AR puede proporcionar información en tiempo real sobre el terreno, la ubicación de objetivos y otras informaciones relevantes para la misión.

En general, la realidad aumentada puede mejorar la eficiencia, seguridad y efectividad de los carros de combate al brindar información y entrenamiento en tiempo real en un entorno seguro. En el momento de entablar combate, la realidad aumentada (AR) puede ayudar a los operadores de carros de combate proporcionándoles una vista mejorada y más detallada del entorno y del objetivo, lo que puede ser útil en situaciones de combate.

También la AR puede mejorar la comunicación entre los operadores de carros de combate y otros miembros del equipo, lo que llega a ser un factor clave en situaciones de combate. El AR puede proporcionar información en tiempo real sobre la ubicación y el entorno de manera precisa, y esta información detallada puede ser crucial para tomar decisiones en base a determinados hechos y situaciones.

Emergencias en puerto

Una emergencia en Puerto engloba cualquier suceso inesperado que pueda poner en peligro la seguridad de nuestros buques y de su personal siendo los incendios una de las emergencias más comunes en puertos y buques marítimos. Los incendios pueden ocurrir debido a una gran variedad de factores, que incluyen fallas eléctricas, fugas de combustible, cortocircuitos así como otros accidentes. Un incendio a bordo de un buque o en una instalación portuaria puede ser especialmente peligroso debido a la presencia de productos químicos y otros materiales inflamables con lo que es de vital importancia tener medidas preventivas y planes de respuesta inmediata para poder minimizar dicho riesgo garantizando de esta manera una respuesta rápida y efectiva en caso de que lo hubiere.



Ilustración [9] ejercicio de seguridad interior

Para efectuar estas respuestas rápidas la Armada adiestra a su personal en lo que se conoce como seguridad interior. La seguridad interior de un barco es un grupo jerarquizado de miembros de la dotación que se encargan de minimizar los daños de una emergencia. En este grupo podemos encontrar figuras como el jefe de trozo, el cual organiza y coordina, junto al Oficial de Seguridad Interior (OSI), las acciones del resto de manera eficaz y siguiendo una serie de pasos estipulados por doctrina.

La aplicación de la tecnología AR puede ser de gran utilidad en puerto y buques de varias maneras: El grupo de seguridad interior puede recibir información en tiempo real sobre la ubicación del incendio y la gravedad de la situación, lo que les permite responder rápidamente y tomar decisiones de acuerdo con una información precisa. Además, la tecnología AR puede proporcionar una guía visual para la respuesta de emergencia en caso de incendios que incluye la ubicación de extintores así como otros equipos de seguridad. La tecnología AR puede utilizarse también para monitorear la propagación del incendio y la eficacia de las medidas de respuesta en tiempo real, lo que permite una respuesta que siempre será más efectiva y segura. También hay que tener en cuenta que la tecnología AR permite una comunicación más eficiente entre los equipos de respuesta de emergencia, lo que puede ayudar a coordinar una respuesta más rápida y acertada.

Operaciones quirúrgicas

Los militares, por la naturaleza de su trabajo, se exponen a menudo a trabajos que son peligrosos. Yo mismo he vivido circunstancias donde se han tenido que realizar intervenciones quirúrgicas de carácter urgente por el bien de la persona que ha tenido el accidente. Es un peligro que existe y que es real. Teniendo en cuenta además que una simple intervención por sencilla que sea siempre puede complicarse por el simple hecho de estar embarcado.

La realidad aumentada puede mejorar las operaciones quirúrgicas a bordo de un buque ya que puede ofrecer las siguientes ventajas:

1. Proporcionar información en tiempo real: la realidad aumentada puede superponer información médica relevante en el campo quirúrgico, mejorando la visibilidad y precisión del cirujano.
2. La realidad aumentada puede utilizarse para simular situaciones médicas realistas y permitir que los médicos practiquen antes de realizar una operación en un ambiente controlado.
3. Guiar procedimientos: la realidad aumentada puede proporcionar guías paso a paso en tiempo real para procedimientos complejos, mejorando la eficacia y la seguridad de la operación.
4. Conectarse con expertos remotos: la realidad aumentada puede permitir la consulta con expertos médicos remotos en tiempo real, lo que puede ser especialmente útil en entornos aislados como un buque.

De esta forma, la realidad aumentada puede mejorar la eficiencia, precisión y seguridad de las operaciones quirúrgicas a bordo de un buque al proporcionar información en tiempo real, entrenamiento, guías y acceso a expertos médicos remotos.

2.2.6

Unidades de buceo

Las unidades de buceo en la Armada española son grupos especializados que realizan misiones submarinas en el mar y en la costa, como la inspección de barcos y estructuras submarinas, la búsqueda y recuperación de objetos y personas, y la realización de investigaciones marinas. Estas unidades están compuestas por buzos altamente capacitados y entrenados, que cuentan con una amplia variedad de equipos y tecnologías para realizar sus misiones con seguridad y eficacia.

Además de las misiones militares, las unidades de buceo de la Armada española también colaboran en operaciones de emergencia y ayuda humanitaria, como la búsqueda y rescate de personas en peligro en el agua. La presencia de estas unidades en la Armada es esencial para garantizar la seguridad marítima y la protección de los intereses nacionales y internacionales.

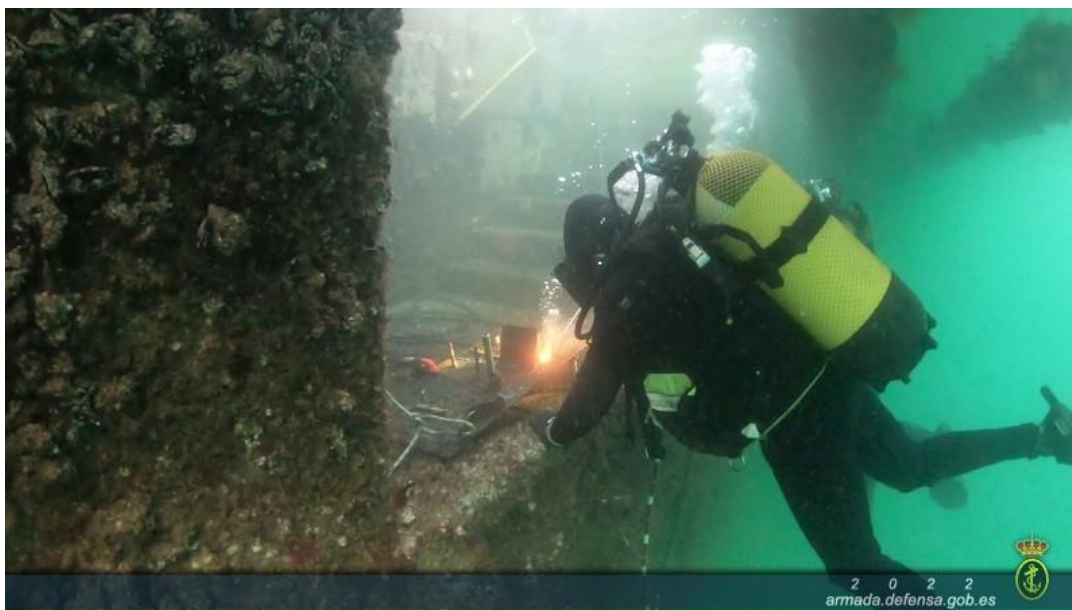


Ilustración [9] buzo realizando mantenimiento

Las unidades de buceo en la Armada española también realizan la desactivación de bombas sumergidas. Esta es una tarea altamente delicada y peligrosa, que requiere de la combinación de habilidades técnicas y experiencia militar. Los buzos de estas unidades están capacitados para identificar y desactivar explosivos sumergidos de manera segura, lo que contribuye a garantizar la seguridad marítima y la protección de la vida humana y el medio ambiente.

Además, las unidades de buceo de la Armada española también pueden trabajar en colaboración con otras agencias y organizaciones, como la policía, para realizar operaciones de desactivación de bombas en el agua. Con su experiencia y habilidades, estas unidades son un valioso recurso para la comunidad nacional e internacional en tareas de seguridad marítima y desactivación de bombas.

El AR puede ayudar a las unidades de buceo de la Armada española de diversas maneras:

1. El AR puede proporcionar un entorno virtual para el entrenamiento de buceo y desactivación de bombas, permitiendo a los buzos practicar y mejorar sus habilidades en un ambiente seguro.
2. El AR puede proporcionar información en tiempo real sobre la ubicación de los objetivos y la navegación durante las misiones submarinas, lo que puede mejorar la eficiencia y la precisión de las tareas.
3. El AR puede ser utilizado para monitorear la salud y el estado de los buzos durante las misiones, proporcionando información sobre la presión arterial, la temperatura y la cantidad de aire restante, lo que incrementa de manera exponencial la seguridad de los buzos.
4. El AR puede ser utilizado para proporcionar información visual y táctil en tiempo real sobre la ubicación y el estado de los explosivos sumergidos, lo que facilita la precisión y la seguridad de las tareas de desactivación de bombas.

5. El AR puede ser utilizado para inspeccionar barcos y estructuras submarinas de manera más eficiente, proporcionando información visual y táctil sobre las condiciones y estructuras.

Hay que entender que esta tecnología puede tardar en llegar, puesto que estamos hablando de aparatos de realidad aumentada que deben sumergirse en el agua a cierta profundidad. No obstante, esta mejora de medios puede hacer que las unidades de buceo en la Armada sean todavía mas eficientes y trabajen de manera más segura.

El gemelo digital

Un Gemelo Digital es un modelo virtual de un sistema o producto físico, que incluye información detallada sobre su funcionamiento, diseño, rendimiento y estado actual. Se utiliza para simular, monitorear y mejorar el rendimiento del sistema o producto físico, y para tomar decisiones informadas sobre su mantenimiento, mejora y operación. Los Gemelos Digitales se utilizan en una amplia variedad de industrias, incluyendo la fabricación, la construcción, la aeroespacial y la defensa.

El Gemelo Digital es una herramienta tecnológica que forma parte del programa de la fragata F110 y se espera que tenga un impacto transformador en la organización. Con el Gemelo Digital, se podrán utilizar nuevas tecnologías como la realidad aumentada, virtual y la impresión 3D. También permitirá un mejor control de la configuración, acceso a datos, modelado de comportamiento de equipos, entre otras funciones, lo que lo convierte en una herramienta valiosa para el apoyo en la toma de decisiones del mando. Además, será un catalizador para el desarrollo digital de la industria auxiliar y un factor clave en la ciberdefensa más robusta. [10]



Ilustración [10] gemelo digital

El uso del Gemelo Digital en el programa de la fragata F110 tiene varias ventajas, tales como:

1. Uso de la realidad aumentada y virtual: permite el uso de tecnologías de realidad aumentada y virtual para mejorar la capacitación, el adiestramiento y la toma de decisiones.

2. Acceso a datos y modelado de comportamiento: facilita el acceso a datos y el modelado de comportamiento de los equipos, lo que permite una mejor toma de decisiones.
3. Simulación y adiestramiento: ofrece nuevas oportunidades para la simulación y el adiestramiento, lo que mejora la capacidad de la organización.
4. Interconexión y gobernanza: proporciona nuevos protocolos de interconexión y gobernanza, lo que garantiza la calidad del dato.
5. Desarrollo digital de la industria auxiliar: sirve como catalizador para el desarrollo digital de la industria auxiliar y del astillero.
6. Ciberdefensa: contribuye a fortalecer la ciberdefensa.
7. Gestión eficiente de instalaciones: la metodología BIM (Building Information Modeling) permitirá una gestión más eficiente de las instalaciones durante su ciclo de vida.
8. Integración del "ecosistema": El Gemelo Digital permitirá integrar el buque, el arsenal de apoyo y sus infraestructuras, el astillero, la industria y los proveedores en un "ecosistema" para mejorar la gestión logística.

2.2.8

Situación marítima a partir de datos AIS

Este proyecto persigue ser una posible aplicación de la AR para la Armada en un futuro. Es por ello que se incluye con el resto de posibles aplicaciones. En la aplicación, se podrían representar sobre cada una de las embarcaciones sus datos identificativos y cinemáticos, como su nombre, bandera, MMSI, COG, SOG, entre otros. Además, se explorará la posibilidad de ofrecer información geográfica y otros datos relevantes, como pecios, cables submarinos, zonas de fondeo, entre otros. Y todo esto a partir de datos del AIS

Un ejemplo de lo que se busca conseguir es la aplicación de AR desarrollada por Marine Traffic (véase Ilustración [11]). Como vemos, en la imagen se nos muestra el nombre del buque, su nacionalidad, la velocidad que lleva y la distancia a la que se encuentra. En la parte superior izquierda podemos observar un pequeño mapa que nos muestra el sector donde estamos apuntando con nuestro teléfono y donde aparecen los barcos que están alrededor.



Ilustración [11] App de AR de Marine Traffic

Para desarrollar esta aplicación, se realizará una comparativa de los métodos de posicionamiento, los dispositivos y las librerías de realidad aumentada disponibles en el mercado. Se evaluarán dispositivos como el Meta Quest 2, teléfonos Android/iOS, y se utilizarán librerías de realidad aumentada como ARCore, Arkit, Vuforia, entre otras. Estas librerías permiten abstraer el funcionamiento de los sensores, como GPS y brújula, y la lógica de posicionamiento y orientación, para representar objetos virtuales anclados a una ubicación geográfica concreta de manera sencilla y sin necesidad de tener muchos conocimientos de programación.

Una vez seleccionadas las herramientas adecuadas, se procederá al desarrollo de la aplicación. Esta será evaluada en un entorno real para validar su funcionamiento y estimar su precisión. De esta manera, se asegurará que la aplicación cumpla con los objetivos propuestos en el proyecto.

Concluyendo, la aplicación desarrollada permitirá representar información del entorno marítimo utilizando realidad aumentada. Esto permitirá una visualización más intuitiva y completa de la información, facilitando la toma de decisiones en tiempo real. Además, el uso de herramientas de realidad aumentada permitirá abstraer la complejidad del proceso de representación de objetos virtuales en un entorno real, lo que facilitará la tarea de los desarrolladores y usuarios.

2.3 Equipos de realidad aumentada

Gafas de realidad aumentada

Las gafas de realidad aumentada son dispositivos que permiten superponer información digital sobre el mundo real. Estas gafas utilizan sensores, cámaras y pantallas para capturar y mostrar imágenes y datos en tiempo real. Algunas gafas de realidad aumentada son autónomas, mientras que otras requieren una conexión a un dispositivo externo, como un teléfono inteligente o una computadora, para funcionar. Algunos ejemplos de gafas de realidad aumentada autónomas son: Microsoft HoloLens, Magic Leap Sony's, SmartEyeglassEpson, Meta Quest, Lenovo Mirage Solo... Estas gafas no necesitan de un dispositivo externo para funcionar y tienen su propio procesador, batería y almacenamiento. No obstante, hay que diferenciar entre gafas de AR y de VR.

Las gafas de Realidad Aumentada (AR) muestran información digital sobre el mundo real, superponiendo gráficos, textos u objetos virtuales a lo que el usuario está viendo. De esta manera, el usuario puede ver el mundo real, pero también tiene acceso a información adicional que enriquece su experiencia. En las gafas de Realidad Aumentada (AR), el sistema de posicionamiento suele ser basado en la localización y el seguimiento de objetos en el mundo real, utilizando sensores como cámaras y GPS. Estos sensores permiten a las gafas de AR conocer la posición del usuario en el mundo real y superponer objetos virtuales en el campo de visión del usuario.

Por otro lado, las gafas de Realidad Virtual (VR) sumergen completamente al usuario en un entorno virtual creado por ordenador, sin mostrar el mundo real. Las gafas de VR a menudo se utilizan en videojuegos y en experiencias de entretenimiento, y ofrecen una inmersión total en el entorno virtual, lo que permite al usuario interactuar con el mundo virtual de manera más natural. En las gafas de VR, el sistema de posicionamiento está diseñado para detectar los movimientos de la cabeza del usuario y para reflejarlos en el mundo virtual. Por lo tanto, las gafas de VR suelen estar equipadas con sensores como acelerómetros y giroscopios y/o cámaras detectan los movimientos de la cabeza del usuario, lo que permite a las gafas de VR actualizar la vista del usuario en tiempo real y ofrecer una experiencia de inmersión total en el mundo virtual. Además de los sensores incorporados en las gafas de Realidad Virtual (VR), existe una tecnología complementaria que se utiliza para mejorar aún más la precisión y la calidad de la experiencia de la Realidad Virtual, y se basa en la colocación de sensores en la habitación en la que se utilizan las gafas de VR. Estos sensores de posicionamiento externo se colocan en la habitación para rastrear los movimientos del usuario en el espacio físico y reflejarlos en el mundo virtual.

Algunas gafas de Realidad Virtual también incluyen cámaras que permiten a los usuarios ver el mundo real mientras utilizan las gafas. Sin embargo, estas cámaras no están diseñadas para proporcionar una experiencia de visualización de alta calidad y pueden presentar algunas limitaciones, como imágenes en blanco y negro con mucha distorsión en algunos modelos, como las Meta Quest 2. Por lo tanto, aunque es posible ver el mundo real mientras se usan las gafas de VR con cámaras incorporadas, esta función no es la principal y la calidad de la imagen puede ser limitada. En general, las gafas de VR están diseñadas principalmente para sumergir al usuario en un entorno virtual y ofrecer una experiencia de inmersión total en el mundo virtual, más que para mostrar el mundo real con alta calidad.

La característica más importante de las gafas de realidad aumentada es su capacidad de superponer información en el mundo real. Esta información puede ser visual, auditiva o táctil, y puede ser utilizada para mejorar la experiencia del usuario, ya sea para fines educativos, de entretenimiento o profesionales.

En cuanto a su funcionamiento, las gafas de realidad aumentada utilizan sensores para detectar y rastrear la posición de la cabeza del usuario, y luego ajustan la imagen en la pantalla para adaptarse a la perspectiva del usuario. Algunas gafas también incluyen sensores de movimiento para detectar el movimiento de los brazos y las manos del usuario, lo que permite interactuar con el contenido de la realidad aumentada de manera más natural.



Imagen 2 Ejemplo de las Hololens

Una de las principales diferencias entre las gafas de Realidad Aumentada y las de Realidad Virtual es que las de AR no bloquean la visión del mundo real. En lugar de eso, utilizan un sensor translúcido sobre el que se proyectan los objetos virtuales, lo que permite al usuario observar el mundo real directamente mientras se superpone la información virtual. Por lo tanto, el usuario puede ver tanto el mundo real como el contenido virtual al mismo tiempo, lo que se conoce como "mezcla" o "superposición" de realidad. Esto ofrece una experiencia más inmersiva y natural, ya que el usuario puede interactuar con el contenido virtual mientras mantiene la conciencia del entorno real.

En el caso de las gafas Microsoft HoloLens utilizan tecnología de posicionamiento interno, que combina datos de sensores como el giroscopio, el acelerómetro y el magnetómetro con información de visión para determinar la posición y orientación de las gafas en el espacio. Además, también utilizan el GPS (no directamente, pero a través de una aplicación y mediante Wi-Fi y Bluetooth) para geolocalizar y determinar su posición en el mundo real. Con esta combinación de tecnologías, las HoloLens pueden superponer gráficos 3D y elementos interactivos en el mundo real de una manera precisa y en tiempo real. Algo a tener en cuenta como desventaja es su precio. El precio de las gafas de realidad aumentada Microsoft HoloLens varía dependiendo del modelo y de la configuración. Y el rango de precio va desde los 3,500 dólares la versión de desarrollador, y alrededor de los 5,000 dólares que cuesta la versión comercial. El precio también puede variar dependiendo de las necesidades y requerimientos específicos del cliente. Es importante mencionar que el precio de las HoloLens puede variar dependiendo del lugar o país donde vivas por lo que es conveniente hacer un estudio de mercado exhaustivo antes de comprar unas para poder comparar precios.

Teléfonos móviles

La realidad aumentada en un teléfono móvil funciona mediante la combinación de varios elementos tecnológicos. En primer lugar, se necesita una cámara para capturar imágenes del entorno real. En segundo lugar, se requiere un procesador potente para poder procesar la información capturada y

superponer elementos virtuales en la imagen y por último se requiere una pantalla para mostrar la imagen final.

Para que la realidad aumentada funcione en un teléfono móvil, también es necesario contar con un sistema de seguimiento de movimiento, que permite al teléfono rastrear su posición y orientación en el espacio. Esto se logra mediante el uso de sensores como el acelerómetro, el giroscopio y el magnetómetro. Efectivamente, los teléfonos móviles suelen incluir sensores de giroscopio, acelerómetro y magnetómetro. El giroscopio se utiliza para medir la rotación del dispositivo en tres dimensiones, mientras que el acelerómetro mide la aceleración del dispositivo en las tres direcciones. El magnetómetro, por otro lado, mide la orientación del campo magnético terrestre y se utiliza para determinar la orientación del dispositivo en el espacio. Estos sensores son esenciales para el seguimiento de movimiento y la detección de la posición y orientación del dispositivo en el espacio, lo que los hace imprescindibles para la realidad aumentada.

El GPS también es un elemento importante en la realidad aumentada móvil, ya que permite al teléfono determinar su posición en el mundo real. Con esta información, la aplicación de realidad aumentada puede superponer elementos virtuales en el entorno en función de la posición del usuario.

Hay que entender que no vale cualquier teléfono móvil. Por lo general las librerías online que utilicemos para nuestros proyectos, como por ejemplo la de ARcore de Google, ya especifican largas listas de los dispositivos con los cuales será compatible. Normalmente las especificaciones no son excesivamente elevadas, por lo que casi cualquier Smartphone nos servirá.

2.3.3 *Conclusión sobre los equipos*

Después de considerar cuidadosamente mis opciones, he llegado a la conclusión de que utilizar un dispositivo móvil es la opción más adecuada para mí en este momento. Entre las razones que me han llevado a tomar esta decisión, destaca principalmente la limitación presupuestaria que me impide adquirir unas gafas de realidad aumentada como las HoloLens. Además, mi familiaridad con el uso del dispositivo móvil y su disponibilidad de GPS, elimina cualquier preocupación en ese sentido. Sin embargo, reconozco el valor de experimentar con gafas de realidad aumentada para evaluar de manera objetiva la experiencia que ofrecen y determinar si son una inversión que merece la pena.

2.4 Plataformas de desarrollo

2.4.1 Una plataforma de desarrollo se define como un conjunto de herramientas, software y tecnologías que permiten a los desarrolladores crear aplicaciones o software para una plataforma específica, como sistemas operativos, dispositivos móviles, consolas de juegos, entre otros.. A continuación, explicaré tres plataformas diferentes que he estudiado para la realización de éste proyecto, exponiendo pros y contras de cada uno.

Android Studio

Android Studio es el principal entorno de desarrollo integrado (IDE) para el desarrollo de aplicaciones para el sistema operativo Android. Fue desarrollado y está siendo mantenido por la empresa Google y es el IDE oficial para el desarrollo de aplicaciones para Android. Es una plataforma completa y gratuita para desarrolladores, que incluye un editor de código avanzado, un depurador, unas herramientas de prueba, un emulador de un teléfono Android y una interfaz de usuario intuitiva. Además, Android Studio tiene un asistente de configuración que ayuda a los desarrolladores a configurar y utilizar diferentes características y herramientas. Android Studio también es compatible con diferentes lenguajes de programación como Java, Kotlin y C++.



Ilustración 2 logo de Android Studio

Tiene varias ventajas para desarrollar aplicaciones, incluyendo:

1. Compatibilidad con Android: Android Studio está específicamente diseñado para desarrollar aplicaciones para Android, lo que permite a los desarrolladores aprovechar al máximo las características y funcionalidades de la plataforma.
2. Depuración y pruebas: Android Studio ofrece herramientas potentes para depurar y probar aplicaciones, como el emulador de dispositivos Android, que permite a los desarrolladores probar su aplicación en diferentes dispositivos y configuraciones sin tener que utilizar físicamente un dispositivo.
3. Integración con Google Play Services: Android Studio está integrado con los servicios de Google Play, lo que permite a los desarrolladores utilizar características avanzadas de Google, como Google Maps, Google Cloud Messaging, entre otros, en su aplicación de realidad aumentada.
4. Amplia gama de plugins: Android Studio tiene una amplia gama de plugins disponibles, como ARCore o Vuforia, entre otros, lo que permite a los desarrolladores utilizar las últimas tecnologías de realidad aumentada.
5. Amplia comunidad: Android Studio cuenta con una gran comunidad de desarrolladores que comparten recursos y conocimientos, lo que permite a los nuevos desarrolladores aprender y obtener ayuda fácilmente.

Por defecto Android Studio utiliza Kotlin. Este es un lenguaje de programación de código abierto desarrollado por JetBrains, creadores de IntelliJ IDEA. Fue lanzado en febrero de 2016 y se ha convertido en uno de los lenguajes de programación más populares en la plataforma Android.

Kotlin es un lenguaje de programación moderno y seguro que se basa en Java, con una sintaxis más limpia y concisa. Tiene características como el manejo seguro de nulos, funciones de primera clase, inferencia de tipos y es interoperable con Java. Es recomendado por Google para el desarrollo de aplicaciones móviles y muchas compañías importantes lo han adoptado o están en proceso de hacerlo.

Unity

Es posible que reconozca el logotipo de Unity (Ilustración 3) pues se trata de una de las compañías de motores de videojuegos más utilizadas en la actualidad. Unity es un motor de juego y entorno de desarrollo integrado (IDE) multiplataforma para la creación de videojuegos en 2D y 3D, aplicaciones interactivas y experiencias virtuales. Fue lanzado por primera vez en 2005 y es desarrollado y mantenido por Unity Technologies. Es utilizado para crear juegos para múltiples plataformas, como Windows, macOS, Linux, iOS, Android, PlayStation, Xbox, entre otros. Permite a los desarrolladores crear gráficos en 3D, animaciones, físicas, iluminación, sonido e interfaz de usuario. Además, cuenta con una amplia comunidad y una gran cantidad de recursos y herramientas en línea para ayudar a los desarrolladores. Es una herramienta que permite el desarrollo de videojuegos para diversas plataformas como videoconsolas, PC e incluso móviles. Con Unity, los creadores pueden crear aplicaciones para diferentes sectores como arquitectura, ingeniería y construcción, sector automotriz, del transporte y la producción, medios y entretenimiento, marketing y publicidad... Pero también cuenta con la opción de aplicaciones para teléfonos móviles tanto Android como Apple. Una de las ventajas de este software es que cuenta como ya hemos dicho con una gran comunidad y esto a la hora de emprender un proyecto siempre facilitará el encontrar ayuda en Internet si tenemos alguna duda o si nos hemos quedado estancados en alguna parte del trabajo y no sabemos continuar.



Ilustración 3 Logotipo de la compañía Unity

En el caso que nos concierne, Unity cuenta también con un apartado de desarrollo de aplicaciones de realidad aumentada. De nuevo estas serían unas de las ventajas de Unity para el desarrollo de aplicaciones:

1. Facilidad de uso: Unity es fácil de aprender y utilizar, lo que permite a los desarrolladores crear aplicaciones de realidad aumentada sin tener que ser expertos en programación.

2. Compatibilidad multiplataforma: Unity permite desarrollar aplicaciones para varias plataformas, incluyendo iOS, Android, Windows y Mac, lo que permite a los desarrolladores llegar a un público más amplio.
3. Amplia comunidad: Unity cuenta con una gran comunidad de desarrolladores que comparten recursos y conocimientos, lo que permite a los nuevos desarrolladores aprender y obtener ayuda fácilmente.
4. Amplio conjunto de herramientas: Unity cuenta con un amplio conjunto de herramientas, como el editor de escena, el sistema de iluminación, el sistema de física, el sistema de animación, entre otros, lo que permite a los desarrolladores crear aplicaciones de realidad aumentada con un alto grado de realismo.
5. Amplia gama de plugins: Unity tiene una amplia gama de plugins disponibles, como Vuforia, ARKit, ARCore, entre otros, lo que permite a los desarrolladores utilizar las últimas tecnologías de realidad aumentada.

Unity utiliza el lenguaje de programación C#. Las principales ventajas de usar C# en comparación con otros lenguajes es su potencia como lenguaje, así como su flexibilidad. C# permite a los desarrolladores crear muchos tipos de aplicaciones seguras y sólidas que se ejecutan en .NET. También cuenta con una sintaxis sencilla similar a Java, facilitando su uso para programadores nuevos. Además, C# ofrece declaraciones en el espacio de nombres, un rango amplio de tipos de datos, atributos de acceso, pase de parámetros, métodos virtuales y redefiniciones, propiedades y un inicializador. Sin embargo, los usuarios tienen que contar con una versión reciente de Visual Studio .NET, además de los requerimientos mínimos del sistema para poder trabajar adecuadamente con C#. Por lo tanto, C# es un lenguaje de programación muy potente y fácil de usar con muchas ventajas, pero también hay algunas desventajas a tener en cuenta como la dicha anteriormente. Así mismo precisamos contar con Windows NT 4 o superior, requeriremos de 4Gb solo para la instalación, además de necesitar espacio extra para descargar extensiones etc. Aparte de todo esto hay que tener en cuenta que para quien no está familiarizado con ningún lenguaje de programación, le costará más trabajo iniciarse en su uso, y si se quiere consultar algún tutorial más explícito sobre la programación en C# se tendría que contar además con una conexión a Internet.

2.4.3

Unreal Engine

Unreal Engine es un motor de juego y desarrollo de realidad virtual desarrollado por Epic Games. Es uno de los motores de juego más populares y utilizados en la industria del juego, y ha sido utilizado en una amplia variedad de juegos para consolas, PC y dispositivos móviles.



Ilustración 4 logo de Unreal

Unreal Engine cuenta con una serie de características y herramientas que facilitan el desarrollo de juegos y aplicaciones:

1. Gráficos de gran calidad: Unreal Engine cuenta con un motor de renderizado avanzado que permite crear gráficos de alta calidad, con iluminación dinámica, sombras y efectos de partículas.
2. Herramientas de edición de niveles: Unreal Engine cuenta con una interfaz visual intuitiva para crear y editar niveles, lo que facilita el trabajo de los desarrolladores.
3. Desarrollo para diferentes plataformas: Unreal Engine permite desarrollar juegos y aplicaciones para varias plataformas, incluyendo PC, consolas, dispositivos móviles y realidad virtual.

Unreal Engine es utilizado en una amplia variedad de juegos y aplicaciones, desde juegos AAA hasta juegos independientes, y también se utiliza en otras industrias como la arquitectura, el cine y la televisión, la educación y la formación, la ingeniería y la manufactura, y la realidad virtual y la realidad aumentada.

El lenguaje que utiliza Unreal Engine es C++. Este es un lenguaje de programación de alto nivel con características de programación orientada a objetos y diseño genérico. Fue desarrollado como una extensión del lenguaje C original con la intención de añadir características de programación orientada a objetos. C++ es ampliamente utilizado en la creación de sistemas operativos, juegos, motores de juegos, aplicaciones de escritorio y móviles, programación de sistemas embebidos y en aplicaciones de inteligencia artificial y aprendizaje automático. Es uno de los lenguajes de programación más populares y utilizado en la industria debido a su rendimiento, flexibilidad y gran cantidad de bibliotecas y recursos disponibles.

Además, Unreal Engine ofrece el uso de blueprints. Las Blueprints de Unreal son un sistema de programación visual que permite a los desarrolladores de juegos crear lógica y comportamientos complejos sin necesidad de escribir código. En lugar de escribir líneas de código en un lenguaje de programación tradicional, los desarrolladores pueden crear scripts visuales utilizando nodos que representan funciones y eventos del motor de juego. Esto es un factor a tener en cuenta pues hace que la programación sea mucho más sencilla.

Comparativa de Plataformas

Android Studio, Unity y Unreal son tres plataformas de desarrollo de software diferentes con enfoques y usos únicos.

^{2.44}Android Studio es un entorno de desarrollo integrado (IDE) de código abierto específico para el desarrollo de aplicaciones para dispositivos Android. Utiliza el lenguaje de programación kotlin o java y proporciona herramientas para diseño de interfaz de usuario, depuración y pruebas. La desventaja de Android es que a la hora de diseñar es poco eficaz ya que no ofrece ningún tipo de interfaz que muestre de manera gráfica el proceso. Solo permite introducir el código, no ofrece ningún tipo de diseño desde la plataforma. Esto hace que pierda atractivo para este tipo de proyectos

Unity, por otro lado, es un motor de juegos y una plataforma de desarrollo de aplicaciones de realidad virtual y aumentada. Utiliza C# como lenguaje de programación y proporciona un entorno de desarrollo visual para crear juegos y aplicaciones interactivas en 2D y 3D. Lo mas atrayente y cómodo de Unity es que ofrece varias herramientas de diseño intuitivas y entretenidas.

Por último, Unreal Engine es otro motor de juegos y plataforma de desarrollo de aplicaciones interactivas. A diferencia de Unity, utiliza C++ como lenguaje de programación y se enfoca en proporcionar gráficos de alta calidad y rendimiento para juegos y aplicaciones de realidad virtual y aumentada. Es un software complejo de usar y que sobre todo se centra en el desarrollo de videojuegos.

Característica	 Android Studio	 Unity®	 UNREAL ENGINE
Lenguaje de programación	Java/kotlin	C#	C++ y blueprints
Plataforma principal	Android principalmente	Multi-plataforma (iOS, Windows, MacOS, Linux, entre otros)	Multi-plataforma (iOS, Windows, MacOS, Linux, entre otros)
Usado principalmente para	Desarrollo de aplicaciones móviles	Desarrollo de juegos y experiencias interactivas	Desarrollo de juegos y experiencias interactivas
Grado de complejidad	Bajo, es un entorno de desarrollo integrado fácil de usar	Medio, ofrece una gran cantidad de funcionalidades pero puede ser complejo para principiantes	Medio-bajo con la utilización de blueprints
Comunidad y recursos disponibles	Amplia, son dos plataformas muy conocidas y tienen una gran comunidad, extensiones y recursos de libre uso fáciles de encontrar en Internet		no tiene una gran comunidad en cuanto a AR se refiere
Licencia	Los tres tienen una licencia gratuita. Tienen versiones pro de pago, pero con la versión gratuita es más que suficiente.		

Tabla 2-1 comparativa de los 3 softwares

Como conclusión de la Tabla 2-1, he decidido utilizar Unity puesto que cuenta con recursos de diseño intuitivos que nos servirán en la realización de este proyecto. El lenguaje C# es un lenguaje con el que estoy más familiarizado que el kotlin o el java, puesto que vimos nociones básicas cuando estábamos en primero. Es relativamente fácil de usar y sobre todo es más atractivo que Android Studio. Unreal por otro lado es francamente más difícil de utilizar y considero que me hacen falta muchas nociones para poder utilizar esta plataforma con fluidez.

En cuanto a Unreal, los Blueprints son una herramienta muy poderosa que permite a los desarrolladores crear lógica y comportamientos complejos de manera visual, sin tener que escribir código tradicional. Esta herramienta es muy intuitiva y puede ser utilizada por cualquier persona, independientemente de su nivel de experiencia en programación. Esto hace que sea una opción a tener en cuenta

Por otro lado, es cierto que Unreal tiene una comunidad menos desarrollada en el ámbito de AR en comparación con Unity. Por eso, si el proyecto está enfocado en AR, puede ser más conveniente utilizar Unity. A pesar de que Unity no soporta Blueprints, cuenta con una amplia gama de herramientas para el

desarrollo de juegos y pluggins de AR, lo que lo convierte en una excelente opción para proyectos enfocados en este ámbito.

2.5 Librerías de realidad aumentada

ARCore

ARCore es una plataforma de realidad aumentada desarrollada por Google, diseñada para ser utilizada en el sistema operativo Android. Esta plataforma permite a los desarrolladores crear aplicaciones de realidad aumentada que pueden ser utilizadas en una amplia variedad de dispositivos Android. La finalidad principal de ARCore es ofrecer una experiencia de realidad aumentada más accesible y fácil de usar, lo que permite a los usuarios interactuar con su entorno de una manera más natural e intuitiva. [6]



Ilustración [6] Logo de ARCore

ARCore utiliza tecnologías avanzadas para entender y hacer seguimiento del entorno del usuario. Por ejemplo, la cámara del dispositivo es utilizada para capturar imágenes y vídeos del entorno, mientras que sensores como el giroscopio y el acelerómetro ayudan a determinar la posición y orientación del dispositivo. ARCore también utiliza la tecnología de detección de superficies para identificar y hacer seguimiento de objetos y de su entorno, lo que permite a los usuarios colocar objetos virtuales en el mundo real. Se puede integrar en las plataformas de Android studio, Unity y Unreal por lo que es destacable su utilidad.

Además de estas tecnologías de detección de entorno, ARCore incluye una amplia variedad de características y herramientas para mejorar la experiencia de realidad aumentada. Así, ARCore ofrece soporte para luces ambientales y sombras dinámicas, lo que permite que los objetos virtuales se integren de manera más realista. También incluye características de seguimiento de mano y de interacción, que permiten a los usuarios controlar dichos objetos con gestos naturales y fáciles de usar.

ARCore ofrece así mismo una amplia variedad de herramientas de desarrollo para facilitar la creación de aplicaciones de realidad aumentada. Por ejemplo, incluye una biblioteca de funciones integrada que permite a los desarrolladores agregar características avanzadas como la detección de superficies y la interacción con objetos virtuales. También incluye una serie de herramientas de desarrollo en línea, como un editor de escena y herramientas de prueba, que permiten a los desarrolladores probar y mejorar sus aplicaciones antes de lanzarlas al mercado.

ARCore posee las siguientes características:

1. Detección de zonas urbanas: como ya sabemos, Google durante los últimos años se ha dedicado a registrar fotográficamente (de manera panorámica) las calles de todo el mundo. ARCore utiliza esto para realizar un reconocimiento exacto de la situación del usuario. Mediante la cámara de nuestro teléfono ARCore reconoce la calle en la que se encuentra y en caso de haber error en el GPS, ARCore se encargaría de corregirlo. Es una herramienta más para el geoposicionamiento.
2. Interacción con objetos virtuales: ARCore ofrece características de seguimiento de mano y de interacción que permiten a los usuarios controlar objetos virtuales con gestos naturales. Gracias a esto podemos encontrar a un usuario de Internet que hizo una aplicación en la que podía empujar un globo virtual con sus manos por poner un ejemplo.
3. Integración realista con el entorno: ARCore incluye soporte para luces ambientales y sombras dinámicas, lo que permite que los objetos virtuales se integren de manera más realista en el entorno.
4. Herramientas de desarrollo: ARCore ofrece una amplia variedad de herramientas de desarrollo para facilitar la creación de aplicaciones de realidad aumentada, incluyendo una biblioteca de funciones integrada y herramientas en línea.

Estas características hacen que ARCore sea una plataforma altamente avanzada y accesible para el desarrollo de aplicaciones de realidad aumentada en Android.

Ofrece una API (Application Programming Interface) que permite a los desarrolladores crear aplicaciones de realidad aumentada al utilizar la cámara y los sensores del dispositivo para detectar y rastrear el entorno del usuario. La API de ARCore proporciona herramientas para detectar superficies, como pisos, mesas y paredes, identificar planos en el espacio real, y realizar un seguimiento en tiempo real de la posición y orientación de la cámara. Además, la API de ARCore es compatible con una amplia gama de dispositivos Android, lo que significa que los desarrolladores pueden llegar a una audiencia más amplia con sus aplicaciones de realidad aumentada.

ARKit

ARKit es una plataforma de realidad aumentada desarrollada por Apple que permite a los desarrolladores crear experiencias de realidad aumentada inmersivas para dispositivos iOS y iPadOS. La plataforma utiliza la cámara, los sensores de movimiento y la potencia de procesamiento del dispositivo para detectar y rastrear la posición y orientación de objetos en el mundo real, lo que permite superponer objetos virtuales de manera precisa. [7]

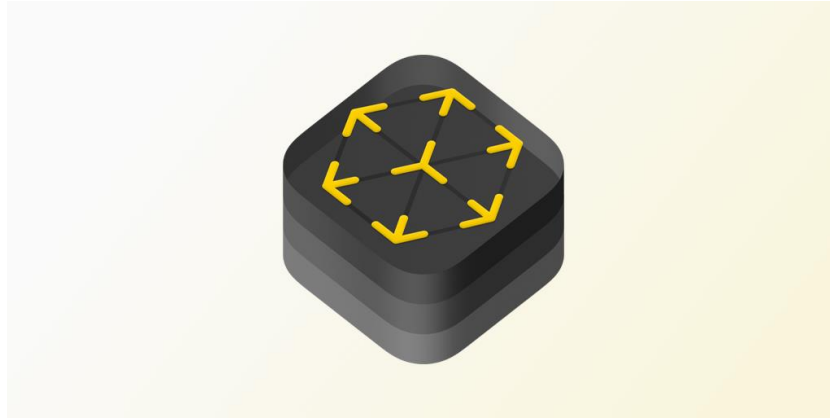


Ilustración [7] logotipo ARkit

ARKit admite varias características avanzadas de AR, como el seguimiento de rostros y la detección de planos, lo que permite la creación de experiencias de AR más realistas y precisas. Además, la plataforma admite la integración con herramientas de desarrollo de juegos como Unity y Unreal Engine, lo que facilita la creación de juegos de AR y aplicaciones interactivas.

Las aplicaciones de ARKit abarcan una amplia gama de industrias, desde el entretenimiento y los juegos hasta la educación y la arquitectura. Por ejemplo, las aplicaciones de ARKit permiten a los usuarios ver cómo quedaría un mueble en su hogar antes de comprarlo, o incluso explorar una ciudad en realidad aumentada para obtener información sobre los puntos de interés cercanos.

En general, se parece bastante a ARcore de Google. No obstante tiene dos características en las que se especializa más. En primer lugar la integración con herramientas de desarrollo de juegos. ARKit se integra con las herramientas de desarrollo de juegos populares como Unity y Unreal Engine, lo que permite a los desarrolladores crear fácilmente juegos y aplicaciones de realidad aumentada. El segundo es el seguimiento de rostros. ARKit admite el seguimiento de rostros y la detección de expresiones faciales, lo que permite la creación de aplicaciones de realidad aumentada avanzadas que utilizan las expresiones faciales del usuario.

ARKit es una plataforma de realidad aumentada poderosa y versátil que ha permitido a los desarrolladores crear algunas de las experiencias de AR más innovadoras y emocionantes en dispositivos móviles.

Vuforia Engine

Vuforia es una plataforma de realidad aumentada desarrollada por PTC que permite a los desarrolladores crear experiencias de realidad aumentada para dispositivos móviles, así como para dispositivos inteligentes como gafas de realidad aumentada. La plataforma utiliza tecnología de detección de objetos y seguimiento de imágenes para superponer objetos virtuales en el mundo real de manera precisa.



Ilustración [8] logotipo de Vuforia

Una de las características principales de Vuforia es su capacidad para detectar objetos específicos en el mundo real, lo que permite la creación de aplicaciones de realidad aumentada que interactúan con objetos específicos. Vuforia también admite la detección de marcadores, que son imágenes o patrones predefinidos que se utilizan para superponer objetos virtuales en el mundo real.

Además de la detección de objetos y marcadores, Vuforia admite la detección de planos y la creación de experiencias de realidad aumentada basadas en la ubicación. Este es su punto fuerte. La plataforma también admite la integración con herramientas de desarrollo entre las que se encuentra Unity, lo que permite a los desarrolladores crear fácilmente aplicaciones de realidad aumentada para dispositivos móviles y gafas inteligentes.

Vuforia es una plataforma de realidad aumentada poderosa y versátil que ha permitido a los desarrolladores crear algunas de las experiencias de realidad aumentada más innovadoras y emocionantes en dispositivos móviles y gafas inteligentes.

2.5.4

Tabla resumen

A continuación, se muestra la Tabla 2, donde se hace un resumen de las tres plataformas y una conclusión final.

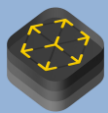
Característica	 ARCore		 vuforia™
Detección de objetos y marcadores	✗ ARCore no cuenta con una detección de objetos y marcadores tan precisa como Vuforia.	✗ ARKit no cuenta con una detección de objetos y marcadores tan precisa como Vuforia.	✓ Vuforia es excelente en la detección de objetos y marcadores en el mundo real, lo que permite superponer información o contenido relacionado en tiempo real.
Detección de planos	✓ ARCore se enfoca en la detección y seguimiento de planos en el mundo real, lo que permite una superposición más precisa de objetos virtuales.	✓ ARKit también es capaz de detectar y seguir planos en el mundo real, lo que permite una superposición más precisa de objetos virtuales.	✓ Vuforia es capaz de detectar planos en el mundo real con precisión, lo que permite una superposición de objetos virtuales más precisa.
Integración con entorno real	✓ ARCore se enfoca en la integración con el entorno real, lo que permite una superposición más natural de objetos virtuales.	✓ ARKit también se enfoca en la integración con el entorno real, lo que permite una superposición más natural de objetos virtuales.	✗ Vuforia no cuenta con la misma integración con el entorno real que ARCore o ARKit

Tabla 2 comparativa de librerías

Finalmente decido utilizar ARcore. La principal razón es porque está diseñada para dispositivos Android que no requieren muchas prestaciones, por lo que será más sencillo llevar a cabo este tipo de proyectos con la plataforma seleccionada. Se centra en la geolocalización y en la integración con el entorno, por lo que es perfecto para lo que estamos buscando. Además, cuenta con un gran número de herramientas y una gran comunidad que utiliza esta plataforma.

Conclusión final

Con el fin de concluir mi estado del arte voy a realizar un breve resumen de lo anteriormente explicado y cuales son las conclusiones finales.

En primer lugar, se ha evaluado la posibilidad de utilizar las Hololens, sin embargo, debido a su elevado costo, esta opción no resultó viable para el proyecto. Asimismo, se descartó la idea de emplear gafas de VR para AR como las Meta Quest, ya que estas no cuentan con características fundamentales

como GPS, lo que las hace inadecuadas para el tipo de aplicación que se pretende desarrollar. Considerando las limitaciones mencionadas, se decidió utilizar un dispositivo móvil con sistema operativo Android como base para la aplicación de realidad aumentada. Esta opción resulta conveniente por su accesibilidad, bajo costo y gran popularidad. En este caso se va a utilizar un Xiaomi redmi note 11.

En cuanto a la plataforma de desarrollo. Se analizó la opción de utilizar Unreal Engine para el desarrollo de la aplicación de realidad aumentada. Esta plataforma de desarrollo cuenta con características como los blueprints, que facilitan el proceso de programación y permiten desarrollar aplicaciones de alta calidad. Sin embargo, se observó que existe una comunidad y un soporte limitado en el ámbito de la realidad aumentada, lo que puede afectar la capacidad de resolver problemas y obtener ayuda en caso de requerirlo.

Así mismo, Android Studio Es una plataforma que cuenta con pocas herramientas de diseño. Lo que dificultaría mucho este proyecto debido a su uso de imágenes y elementos gráficos.

Finalmente, se decidió utilizar Unity en combinación con ARCore como la tecnología más adecuada para el desarrollo de la aplicación de realidad aumentada. Unity es una plataforma de desarrollo de juegos y aplicaciones ampliamente utilizada y que cuenta con una gran comunidad de desarrolladores. Además, ARCore es un SDK de Google diseñado específicamente para el desarrollo de aplicaciones de realidad aumentada en dispositivos móviles con Android, lo que permite aprovechar las capacidades del dispositivo móvil de manera óptima.

3 DESARROLLO DEL TFG

3.1 Descripción del apartado

Configuración y primeros pasos con Unity

Tras descargar Unity el primer paso es configurar adecuadamente el proyecto. Para empezar a trabajar con AR, el usuario tiene que instalar las extensiones de AR, como AR Foundation y ARCore. Para hacerlo, debe asegurarse de tener AR Foundation instalada y configurada en su entorno de desarrollo. Si no la tiene, debe buscar en el Package Manager de Unity y hacer clic en el botón "Install". Luego, debe abrir o crear un nuevo proyecto de Unity y seleccionar la plataforma de compilación de destino deseada, que en su caso será Android. En el menú File, debe seleccionar Build Settings y elegir Android como plataforma de destino deseada. Finalmente, debe hacer clic en el botón "Switch Platform" para cambiar la plataforma de compilación.

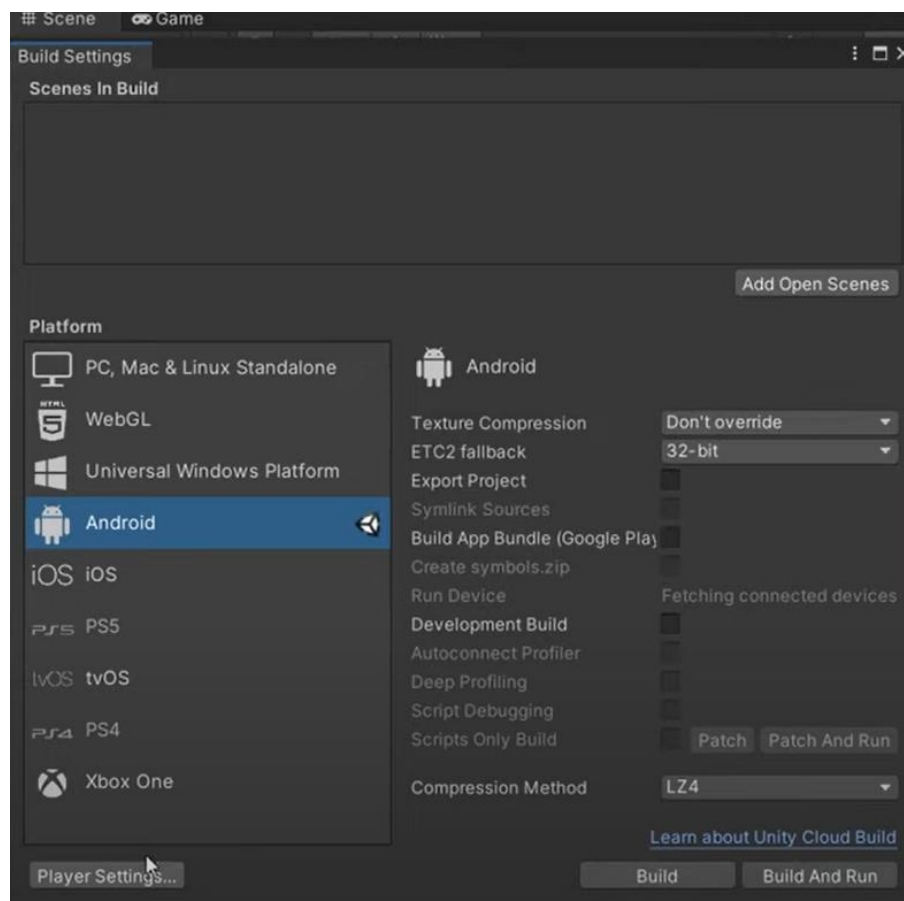


Ilustración 6 cambio a Android

Para importar el paquete de extensiones de ARCore más reciente, debe ir a Window → Package Manager en Unity. Luego, hacer clic en el botón "+" y seleccionar la opción "Add package from git URL..." en el menú desplegable. En el campo de texto, pegue la siguiente URL: <https://github.com/google-ar/arcore-unity-extensions.git>. Esta URL es la dirección del repositorio Git

donde se almacena el código fuente del paquete de extensiones de ARCore para Unity. Al pegar esta URL en el Package Manager de Unity, indique a Unity que descargue el código fuente del paquete desde ese repositorio, compile los archivos necesarios y agregue el paquete al proyecto de Unity. Luego, espere a que se descargue e instale el paquete de ARCore y ya está listo para utilizarlo en el proyecto de Unity.

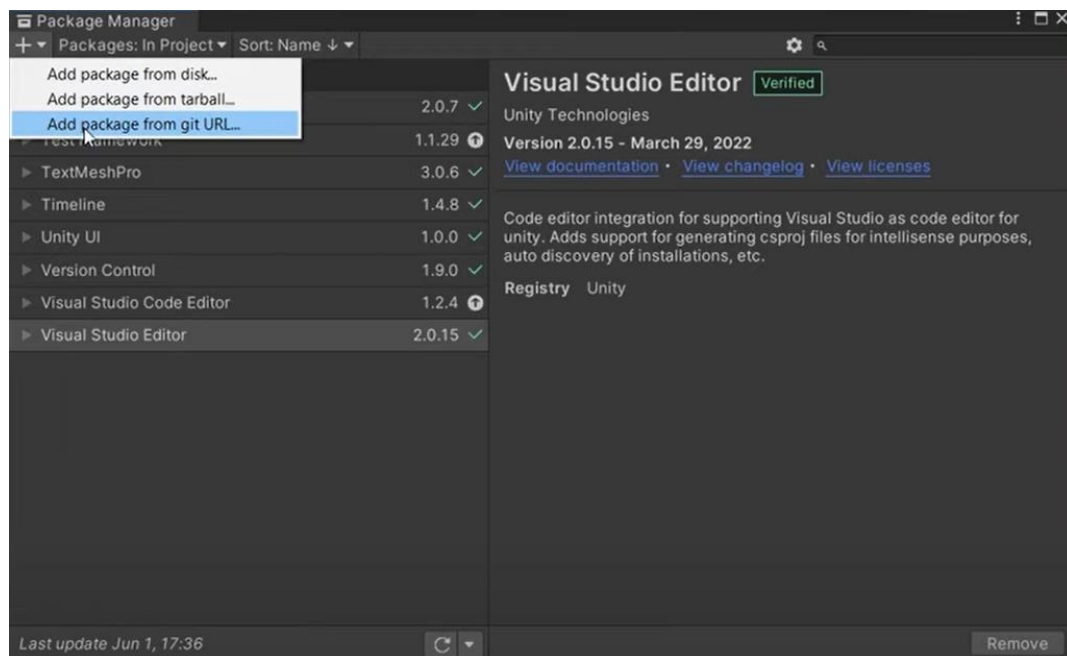


Ilustración 7 Añadir paquete de git

Cuando instale ARCore Extensions en el proyecto de Unity, también se instalará automáticamente una versión adecuada de las dependencias requeridas: AR Foundation y ARCore XR Plugin.

Es importante destacar que estas dependencias pueden no ser visibles en la vista Package Manager > Package: In project en Unity. Sin embargo, se pueden ver los paquetes instalados en la ventana Project en la sección de Packages. Al instalar ARCore Extensions, asegure que todas las dependencias necesarias también se instalarán y configurarán automáticamente para el proyecto. Esto hace que el proceso de integración de ARCore en el proyecto de Unity sea mucho más fácil, ya que no tiene que preocuparse por instalar y configurar manualmente cada una de las dependencias necesarias.

Para configurar las extensiones de ARCore, siga los siguientes pasos: En el panel Hierarchy, haga clic con el botón derecho y agregue los siguientes objetos del juego: XR → AR Session, XR → AR Session Origin y XR → ARCore Extensions. Luego, seleccione el paquete ARCore Extensions en el panel Hierarchy.

En el panel Inspector, para cada uno de los siguientes campos, haga clic en el botón de destino y conecte cada campo con el objeto de juego correspondiente de la siguiente manera:

Session: Use el AR Session de su escena.

Session Origin: Use el AR Session Origin de su escena.

Camera Manager: Use el AR Camera de su escena.

Después, cree y conecte los siguientes elementos para los Cloud Anchors y los filtros de configuración de cámara: En el panel Project → Assets, haga clic con el botón derecho y seleccione Create → XR → ARCore Extensions Config. En el panel Project → Assets, haga clic con el botón derecho y seleccione Create → XR → Camera Config Filter.

Vuelva a seleccionar el paquete ARCore Extensions en el panel Hierarchy. En el campo ARCore Extensions Config del panel Inspector, haga clic en el botón de destino y conéctelo al elemento ARCoreExtensionsConfig. En el campo Camera Config Filter del panel Inspector, haga clic en el botón de destino y conéctelo al elemento ARCoreExtensionsCameraConfigFilter. Siguiendo estos pasos y conectando correctamente todos los campos de las extensiones de ARCore, el panel de extensiones de ARCore Inspector se ve como en la Ilustración 8:

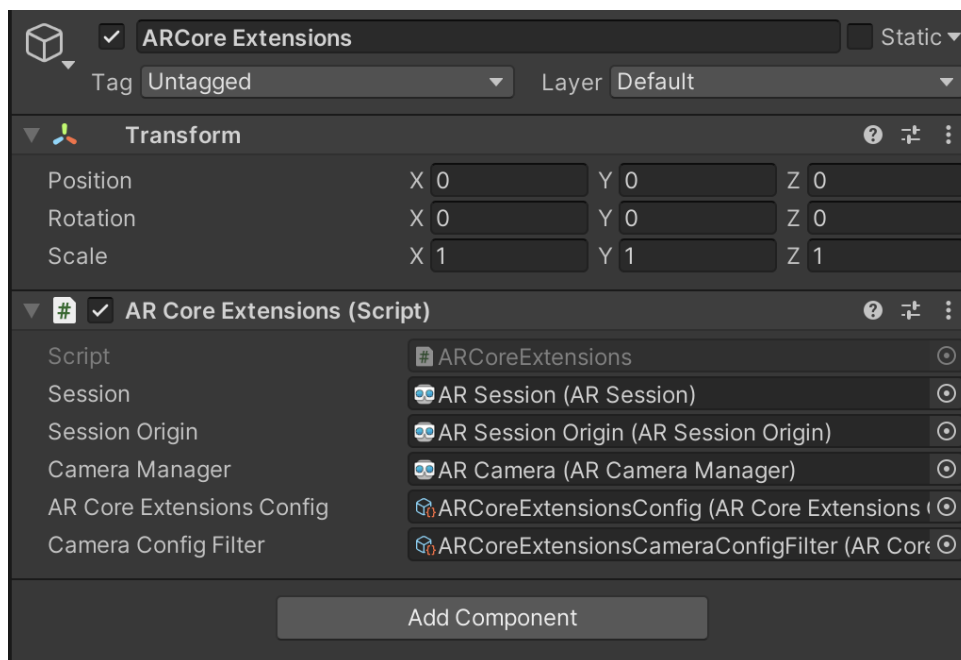


Ilustración 8 interfaz Unity

Por último solo queda descargar el proyecto muestra de aplicaciones de AR geoespacial de Google a partir de la API. Es tan sencillo como entrar en la Google cloud platform console y desde ahí crear un nuevo proyecto. Una vez creado, lo único que hay que hacer es ir a la parte de credenciales y desde ahí crear una API privada. Copiamos la API y nos vamos a la sección de Project settings → ARCore Extensions y seleccionamos API Key en la estrategia de autenticación de Android. Es entonces cuando nos abre una pestaña donde pegaremos nuestra clave API anteriormente creada. El resultado queda como en la Ilustración 9:

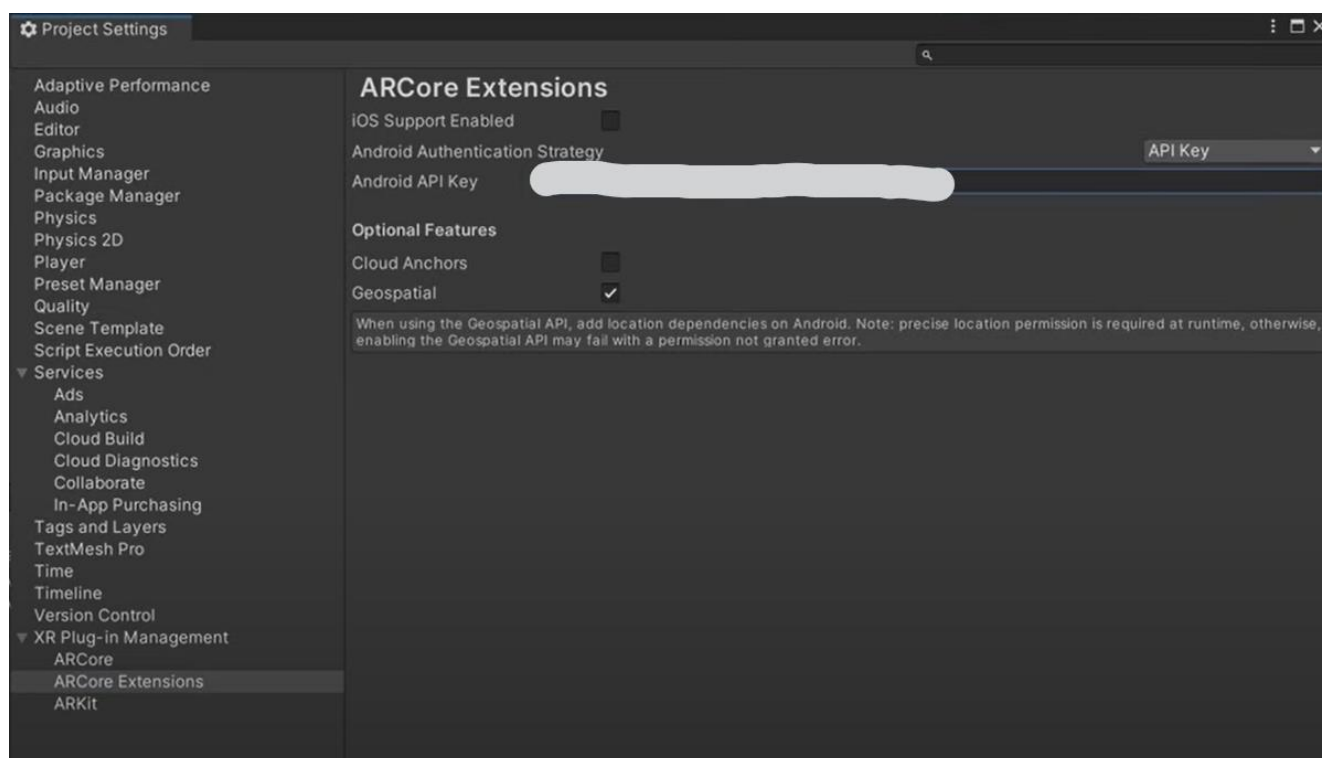


Ilustración 9 API key

Y con esto ya tiene completamente inicializado y configurado el proyecto.

3.1.2 *Script VPSManager*

Tras haber configurado el nuevo proyecto es hora de crear el primer script. En Unity, un script es un programa que se utiliza para agregar interactividad y lógica a los objetos del juego. Los scripts en Unity se adjuntan a los objetos del juego y se ejecutan mientras el juego está corriendo.

El objetivo de este script es gestionar la colocación de objetos en realidad aumentada (AR) utilizando el modo geoespacial en la plataforma AR Foundation de Unity. El script verifica si el modo geoespacial es compatible, y si lo es, coloca los objetos en la posición correspondiente en la Tierra en función de sus coordenadas de latitud, longitud y altitud. El script utiliza el componente AREarthManager para obtener la posición y el estado de seguimiento de la Tierra, y el componente ARAnchorManager para agregar anclas de AR a la escena.

En este caso va a crear un script llamado VPSManager. En la Ilustración 10 podemos ver una captura completa del código y a continuación se explicará parte por parte que es lo que hace.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.XR.ARFoundation;
5 using UnityEngine.XR.ARSubsystems;
6 using Google.XR.ARCoreExtensions;
7 using System;
8
9 public class VPSManager : MonoBehaviour
10 {
11     [SerializeField] private AREarthManager earthManager;
12
13     [Serializable]
14     public struct GeospatialObject
15     {
16         public GameObject ObjectPrefab;
17         public EarthPosition EarthPosition;
18     }
19
20     [Serializable]
21     public struct EarthPosition
22     {
23         public double Latitude;
24         public double Longitude;
25         public double Altitude;
26     }
27
28     [SerializeField] private ARAnchorManager arAnchorManager;
29     [SerializeField] private List<GeospatialObject> geospatialObjects = new List<GeospatialObject> ();
30     // Start is called before the first frame update
31     void Start()
32     {
33         VerifyGeospatialSupport();
34     }
35
36     private void VerifyGeospatialSupport()
37     {
38         var result = earthManager.IsGeospatialModeSupported(GeospatialMode.Enabled);
39
40         switch (result)
41         {
42             case FeatureSupported.Supported:
43                 Debug.Log("Ready to VPS");
44                 PlaceObjects();
45                 break;
46             case FeatureSupported.Unknown:
47                 Debug.Log("Unknown...");
48                 Invoke("VerifyGeospatialSupport", 5.0f);
49                 break;
50             case FeatureSupported.Unsupported:
51                 Debug.Log("Unsupported");
52                 break;
53         }
54     }
55
56     private void PlaceObjects()
57     {
58         if (earthManager.EarthTrackingState == TrackingState.Tracking)
59         {
60             var geospatialPose = earthManager.CameraGeospatialPose;
61
62             foreach (var obj in geospatialObjects)
63             {
64                 var earthPosition = obj.EarthPosition;
65                 var objAnchor = arAnchorManagerExtensions.AddAnchor(arAnchorManager, earthPosition.Latitude, earthPosition.Longitude, earthPosition.Altitude, Quaternion.identity);
66                 Instantiate(obj.ObjectPrefab, objAnchor.transform);
67             }
68         }
69         else if (earthManager.EarthTrackingState == TrackingState.None)
70         {
71             Invoke("PlaceObjects", 5.0f);
72         }
73     }
74 }
75
76
77
78
79
```

Ilustración 10 código VSPManager completo

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.XR.ARFoundation;
5  using UnityEngine.XR.ARSubsystems;
6  using Google.XR.ARCoreExtensions;
7  using System;
8
9  public class VPSManager : MonoBehaviour
10 {

```

Ilustración 11 código VPSManager (1)

En estas líneas, se están importando varios módulos necesarios para el funcionamiento del script. "ARFoundation" y "ARCoreExtensions" son bibliotecas que permiten la creación de aplicaciones de realidad aumentada. "System" y "System.Collections.Generic" son bibliotecas comunes de C# para la gestión de colecciones de datos.

```

11 [SerializeField] private AREarthManager earthManager;
12
13 [Serializable]
14 public struct GeospatialObject
15 {
16     public GameObject ObjectPrefab;
17     public EarthPosition EarthPosition;
18 }
19
20 [Serializable]
21 public struct EarthPosition
22 {
23     public double Latitude;
24     public double Longitude;
25     public double Altitude;
26 }

```

Ilustración 12 código VPSManager (2)

En estas líneas primero se declara un objeto de tipo *AREarthManager* llamado "earthManager" que se serializa y se puede asignar desde el inspector de Unity. A continuación, se define un objeto geoespacial que consiste en un prefab de *GameObject* y una posición de la Tierra *EarthPosition*. Un prefab en Unity es un objeto que contiene una configuración predefinida de componentes y propiedades que se pueden reutilizar fácilmente en múltiples escenas o proyectos. Por ultimo define *EarthPosition* como una posición de la Tierra que consta de una latitud, una longitud y una altitud.

```

28 [SerializeField] private ARAnchorManager aRAnchorManager;
29 [SerializeField] private List<GeospatialObject> geospatialObjects = new List<GeospatialObject> ();
30 // Start is called before the first frame update
31 void Start()
32 {
33     VerifyGeospatialSupport();
34 }
35

```

Ilustración 13 código VPSManager (3)

Estas líneas declaran dos objetos que se serializan y se pueden asignar desde el inspector de Unity. "aAnchorManager" es un objeto de ARAnchorManager que se utilizará para crear y administrar anclas geoespaciales en la escena. "geospatialObjects" es una lista de objetos geoespaciales que se utilizarán para colocar los objetos en la posición deseada en la Tierra. Luego se ejecuta *Start* cuando se inicia el script. Llama al método *VerifyGeospatialSupport()* para verificar si el dispositivo es compatible con la función geoespacial.

```
36
37 private void VerifyGeospatialSupport()
38 {
39     var result = earthManager.IsGeospatialModeSupported(GeospatialMode.Enabled);
40
41     switch (result)
42     {
43         case FeatureSupported.Supported:
44             Debug.Log("Ready to VPS");
45             PlaceObjects();
46             break;
47         case FeatureSupported.Unknown:
48             Debug.Log("Unknown...");
49             Invoke("VerifyGeospatialSupport", 5.0f);
50             break;
51         case FeatureSupported.Unsupported:
52             Debug.Log("Unsupported");
53             break;
54     }
55 }
56
57 }
```

Ilustración 14 código VPSManager (4)

En esta parte del código de la Ilustración 14 *VerifyGeospatialSupport* se encarga de verificar si el modo geoespacial es compatible con el dispositivo.

Primero, la función llama al método *IsGeospatialModeSupported* de *AREarthManager* y pasa el modo geoespacial habilitado (*GeospatialMode.Enabled*) como parámetro. El resultado se almacena en la variable *result*.

A continuación, utilice una estructura de control switch para manejar el resultado del método *IsGeospatialModeSupported*. Si el modo geoespacial es compatible (*FeatureSupported.Supported*), se muestra un mensaje de depuración *Debug.Log* que indica que el dispositivo está listo para utilizar la realidad aumentada geoespacial, y luego se llama a la función *PlaceObjects* para colocar los objetos en el espacio geoespacial.

Si el resultado es desconocido (*FeatureSupported.Unknown*), significa que el dispositivo aún está verificando si el modo geoespacial es compatible. En este caso, se muestra un mensaje de depuración y se utiliza el método *Invoke* para llamar a la función *VerifyGeospatialSupport* nuevamente después de 5 segundos. Si el resultado es no compatible (*FeatureSupported.Unsupported*), se muestra un mensaje de depuración que indica que el dispositivo no es compatible con el modo geoespacial.

```

58 private void PlaceObjects()
59 {
60     if (earthManager.EarthTrackingState == TrackingState.Tracking)
61     {
62         var geospatialPose = earthManager.CameraGeospatialPose;
63         foreach (var obj in geospatialObjects)
64         {
65             var earthPosition = obj.EarthPosition;
66             var objAnchor = ARAnchorManagerExtensions.AddAnchor(arAnchorManager, earthPosition.Latitude, earthPosition.Longitude, earthPosition.Altitude, Quaternion.identity);
67             Instantiate(obj.ObjectPrefab, objAnchor.transform);
68         }
69     }
70     else if (earthManager.EarthTrackingState == TrackingState.None)
71     {
72         Invoke("PlaceObjects", 5.0f);
73     }
74 }
75
76
77
78

```

Ilustración 15 código VPSManager (4)

Por ultimo, esta es una función llamada *PlaceObjects* que se encarga de colocar los objetos en la posición correcta en el espacio geoespacial.

Comienza con una declaración de control de flujo *if* que verifica si el estado de seguimiento de la tierra (*EarthTrackingState*) es igual a *Tracking*. Si es así, la función continúa con la lógica para colocar los objetos. Si no es así, significa que la cámara no ha podido rastrear la ubicación de la Tierra, por lo que la función espera 5 segundos y luego se llama a sí misma utilizando el método *Invoke*.

Dentro del *if*, la función recorre cada uno de los objetos en la lista *geospatialObjects* usando un bucle *foreach*. Para cada objeto, se obtiene la información de la posición terrestre almacenada en la estructura *EarthPosition*.

A continuación, se utiliza el método *AddAnchor* de *ARAnchorManagerExtensions* para crear un objeto de anclaje (*anchor*) en la posición especificada por *Latitude*, *Longitude*, *Altitude*, y *Quaternion.identity*. Este anclaje representa la posición en el espacio geoespacial donde se colocará el objeto 3D.

3.1.3 Finalmente, se instancian los objetos 3D usando *Instantiate* y se colocan en la posición del objeto de anclaje creado anteriormente.

Creación del marcador

Para la creación del marcador, ha sido creado un proyecto aparte donde poder hacer cambios de forma rápida y sin tener que compilar el programa en el teléfono que es más tedioso.

Desde Unity se puede cambiar fácilmente un texto, ya que la propia interfaz de Unity me deja editar el texto que yo quiera. El objetivo primero es poder representar un texto donde apareciese el nombre de un barco, su nacionalidad y su velocidad desde un script. Para ello introduce el siguiente código:

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using TMPro;
5
6
7  public class demo : MonoBehaviour
8  {
9
10     public TMP_Text Buque;
11     public TMP_Text Nacionalidad;
12     public TMP_Text velocidad;
13
14
15     // Start is called before the first frame update
16     void Start()
17     {
18
19         Buque.text = "barrutia";
20         Nacionalidad.text = "Españita";
21         velocidad.text = "8Kts";
22
23     }
24
25     // Update is called once per frame
26     void Update()
27     {
28
29     }
30 }
31
```

Ilustración 16 código texto

Este código lo que hace es definir tres textos en TextMeshPro. TextMeshPro es una mejora significativa sobre el sistema de texto predeterminado de Unity, por eso en un principio se ha decidido usarlo. Estos tres textos los llamaremos Buque, Nacionalidad y Velocidad. Más adelante les daremos el nombre barrutia, Española y 8Kts respectivamente.

Ejecutando la aplicación queda como en la Ilustración 17.

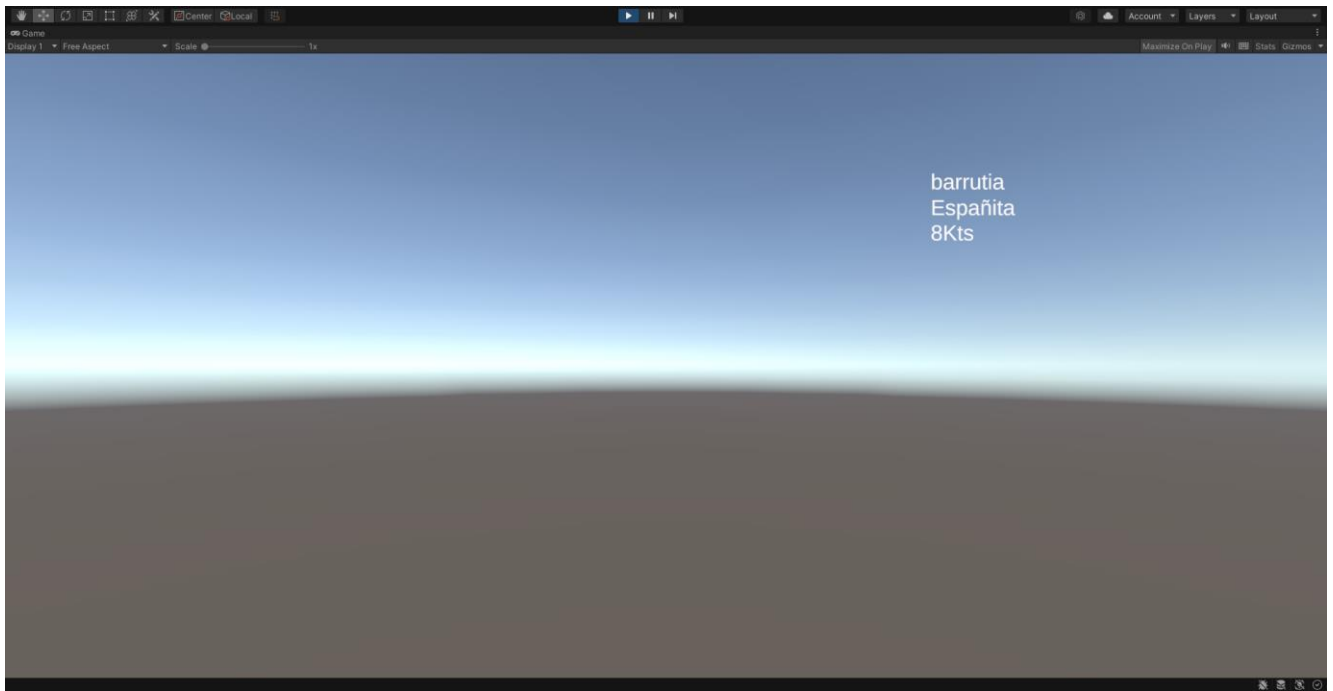


Ilustración 17 textos Unity

Una vez hecho esto realice un diseño sencillo para poner los textos. En este caso, a pesar de incluir texto es solo una imagen (realizada con la app Goodnotes):

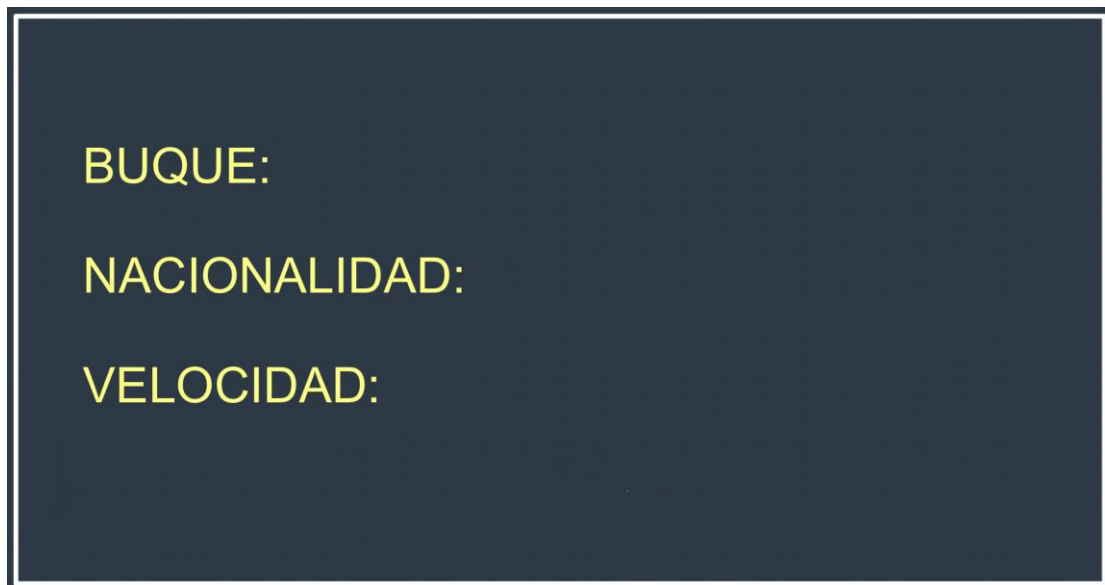


Ilustración 18 primer diseño de marcador

A continuación, cree una imagen UI de interfaz de usuario. Una imagen UI es un elemento de la interfaz de usuario que se utiliza para mostrar gráficos o imágenes en 2D superpuestas en el juego o la aplicación de Unity.

En Unity, las imágenes UI se pueden agregar como objetos de la interfaz de usuario a través de la ventana "Canvas". Una vez agregadas, las imágenes UI se pueden ajustar y personalizar para mostrar la imagen deseada, ya sea una imagen estática o animada. En Unity, un Canvas es un componente que define un área de la pantalla donde se pueden colocar elementos de la interfaz de usuario (UI). Es como una hoja en blanco que sirve de contenedor para los objetos de la interfaz de usuario, como botones, texto, imágenes, etc.

En este caso coloque una plantilla en blanco y agregue la Ilustración 18 en forma de sprite (formato utilizado en Unity para agregar imágenes). Este cambio se puede ver en la pestaña *inspector* mostrada en la Ilustración 19. En esta seleccionamos el *texture type* y la ponemos como sprite.

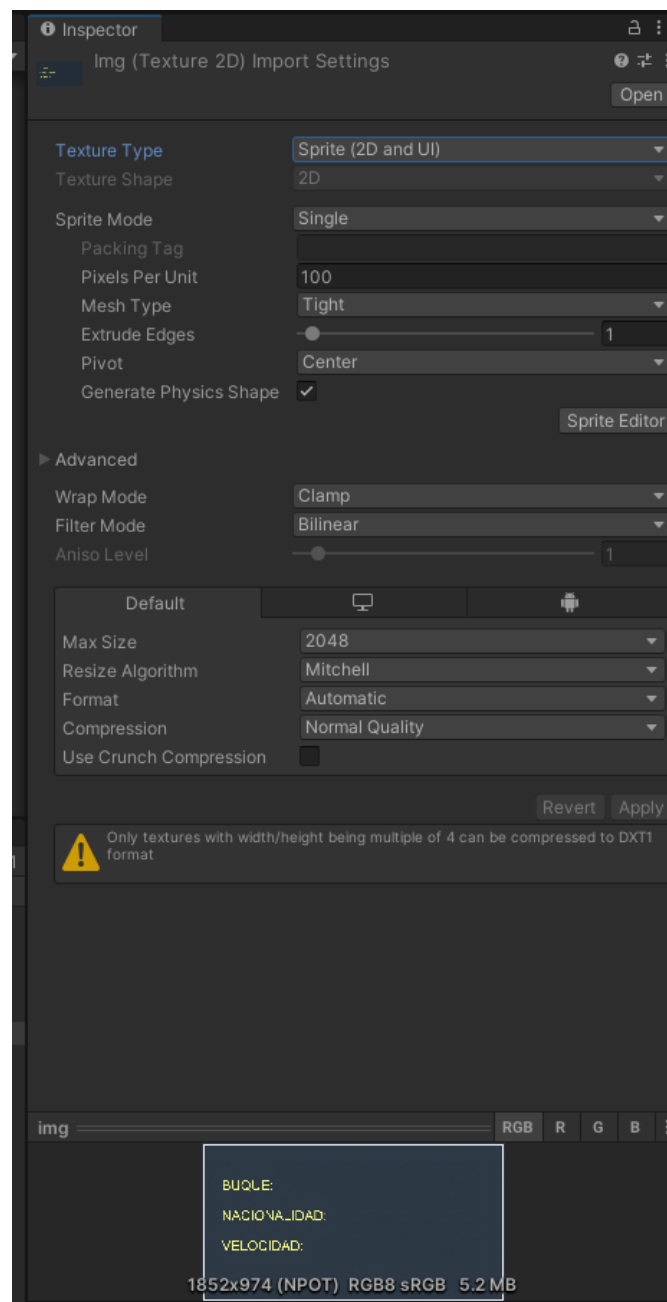


Ilustración 19 Cambio a Sprite

A la imagen le hemos añadido los tres textos de la Ilustración 17. Para ello le añadimos el script demo a nuestro prefab de marcador. Simplemente arrastramos los textos a su cambio correspondiente. Viendose como en la Ilustración 20

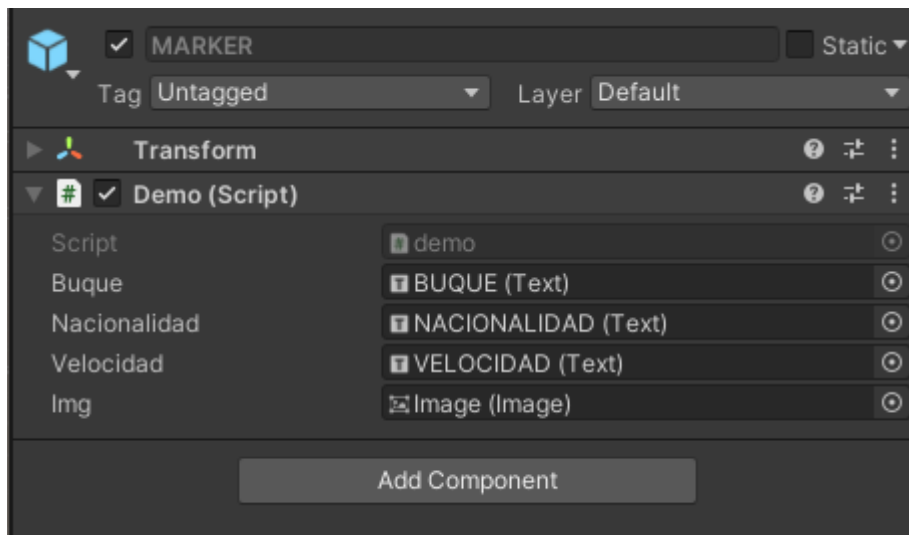


Ilustración 20 interfaz Unity

No obstante al ejecutar el programa los textos y la imagen no son compatibles. Por lo que sustituimos los textos de TextMeshPro por textos de interfaz de usuario (UI). En el código de la Ilustración 16 únicamente hay que cambiar *TMP_Text* por *Text* para que funcione. Volvemos a ejecutar y el resultado se puede ver en la Ilustración 21.

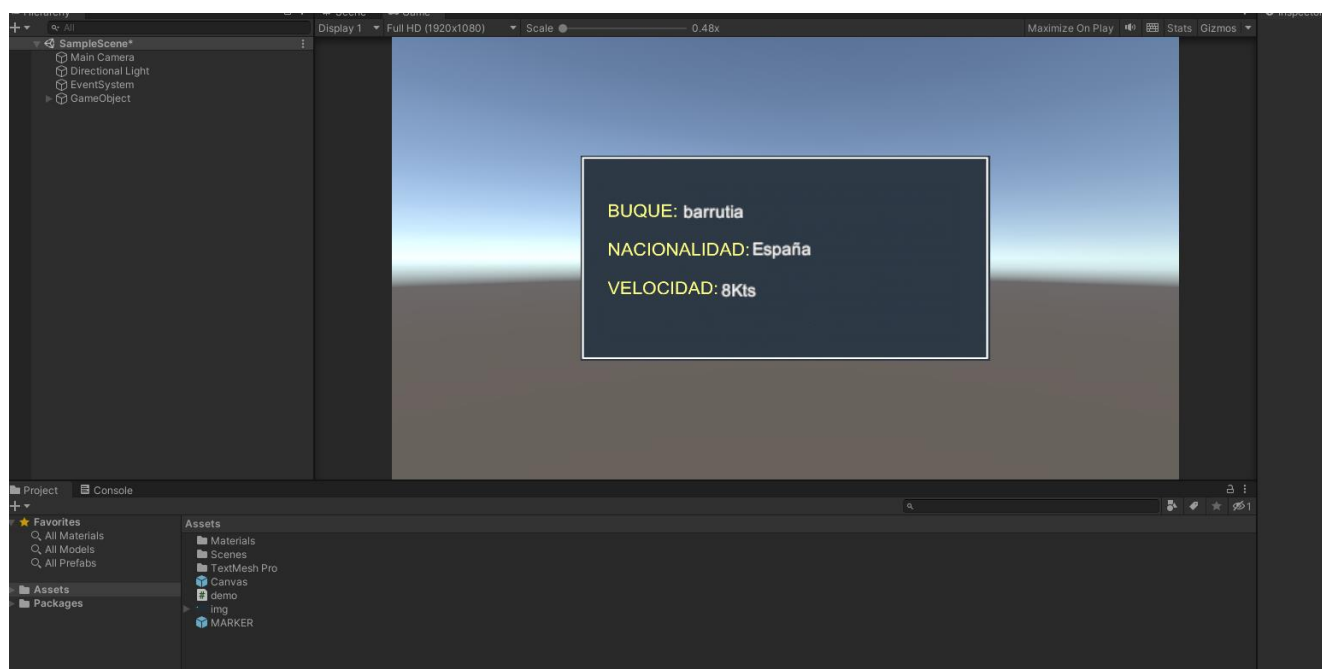


Ilustración 21 marcador con textos

3.1.4 *Poner el marcador en AR*

Una vez tiene el marcador hecho, ahora tenemos que probarlo en el proyecto de AR. Para ello meta el prefab del marcador en el objeto que quiera representar en AR, donde pone *object prefab* como se ve en la Ilustración 22:

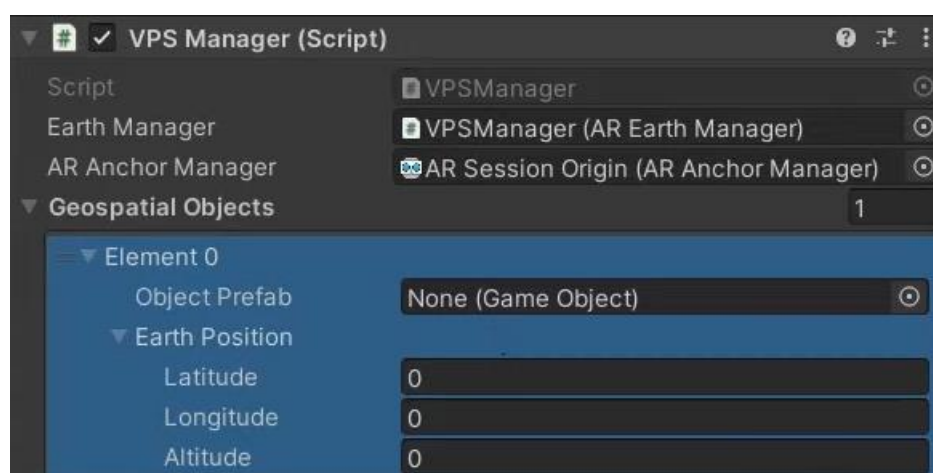


Ilustración 22 captura de proyecto

A esto también debemos añadirle una posición GPS en donde pone *Earth Position* en la Ilustración 22. En este caso se ha introducido la posición del laboratorio puesto que es donde se está realizando el proyecto. El resultado al ejecutar se puede ver en la Ilustración 23.

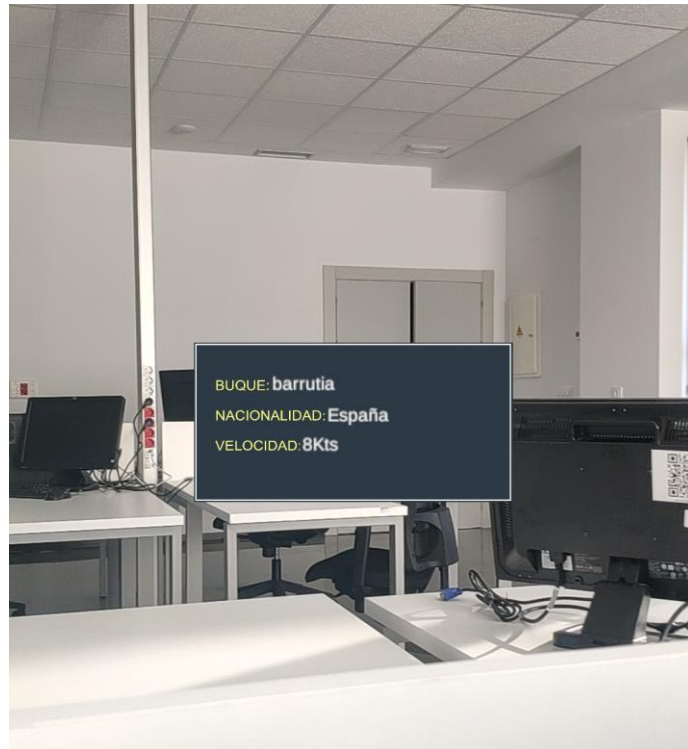


Ilustración 23 prueba AR con marcador

El resultado puede parecer bueno. No obstante, el marcador que aparece en pantalla al iniciar la app no se coloca en la posición GPS sino que se queda en el centro de la pantalla. Por lo que este primer intento tiene un error que debemos resolver.

3.1.5

Solución al error

La solución a este error se ha realizado cambiando el script Demo de la Ilustración 16. Lo que se ha averiguado es que las imágenes UI necesitan una referencia, no vale con darle un posicionamiento GPS. Por ello se ha modificado el script Demo de la siguiente manera:

```
15 public Transform target;
```

Ilustración 24 declaracion del Target

Esta línea de código de la Ilustración 24 declara una variable llamada "target" de tipo Transform. Un componente Transform en Unity almacena la posición, la rotación y la escala de un objeto en el mundo 3D. Por lo tanto, esta variable se utilizará para almacenar una referencia al objeto en el mundo 3D que se usará para posicionar elementos en la interfaz de usuario en la pantalla.

```
41 Vector2 pos = Camera.main.WorldToScreenPoint(target.position)
```

Ilustración 25 código posición del taget

En esta línea de código se convierte la posición del objeto "target" en coordenadas del mundo a coordenadas de pantalla utilizando la función "WorldToScreenPoint" de la cámara principal de la escena (Camera.main).

El método "WorldToScreenPoint" toma la posición en coordenadas del mundo (que se especifica como el argumento "target.position") y la convierte en coordenadas de pantalla. El resultado se almacena en la variable "pos", que es un Vector2 que representa la posición del objeto en la pantalla.

Una vez hecho esto, introduzca un objeto invisible en el campo de la Ilustración 22 de *object prefab*. Que quede como en la Ilustración 26.

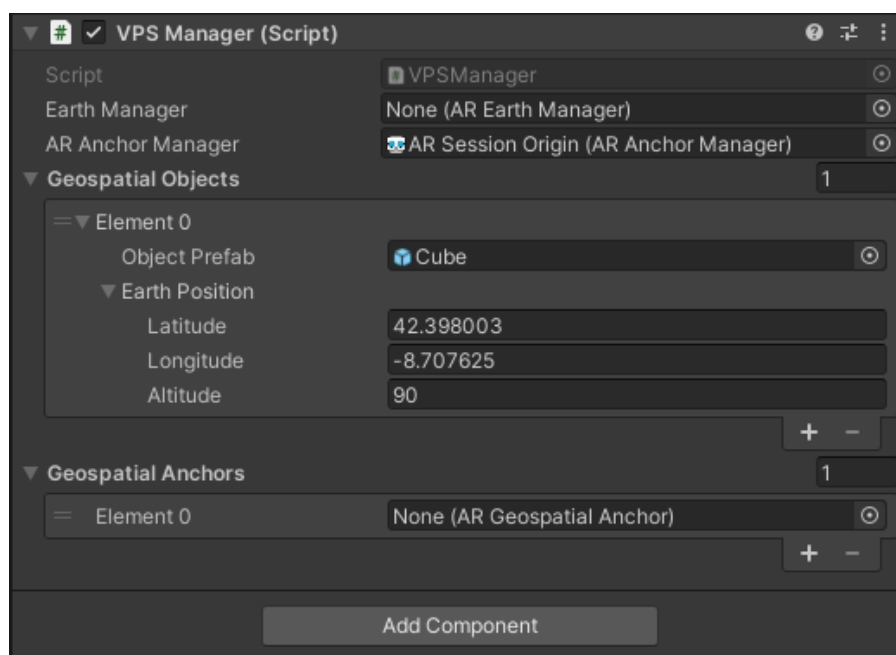


Ilustración 26 captura Cube en VPSManager

Este objeto es un simple cubo invisible que es el que va a hacer de target. Por lo que también meta ese mismo cubo en el campo de Target del marcador. Como en la Ilustración 27.

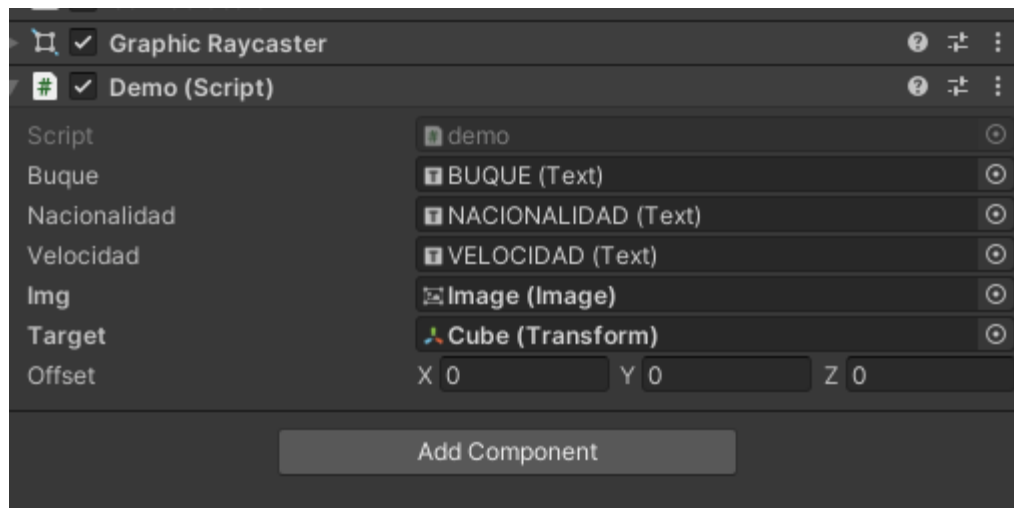


Ilustración 27 campos del marcador

Con esto lo que hemos conseguido es hacer que nuestra Ilustración 18 se coloque justo sobre el cubo desde el punto de vista de la cámara. Al ejecutar el resultado queda como en la Ilustración 28.



Ilustración 28 segunda prueba con marcador en AR

Lo que ocurre aquí es que imagen UI sí consigue que se coloque en la posición que desea. No obstante, los textos siguen centrados como ocurría en la Ilustración 23. Para solucionar esto introduje el siguiente código:

```
63  
64     img.transform.position = pos;  
65     Buque.transform.position = new Vector2(pos.x + 32.6f, pos.y + 47f);  
66     Nacionalidad.transform.position = new Vector2(pos.x + 119.2f, pos.y - 2.2f);  
67     velocidad.transform.position = new Vector2(pos.x + 80.1f, pos.y - 48.1f);  
68
```

Ilustración 29 código para posicionar

En estas líneas de código simplemente hace que la posición de la imagen y los textos sean la de *pos* que ya calculó en la Ilustración 25. Los textos en este caso si los deja en esa posición lo que ocurre es que saldrán por defecto justo en el centro de la imagen. Para solventar esto simplemente ajustamos la posición en el eje x,y según donde queramos que aparezcan.

Al ajustar las posiciones e indicar que la imagen estará justo en el centro quedando como en la Ilustración 30.

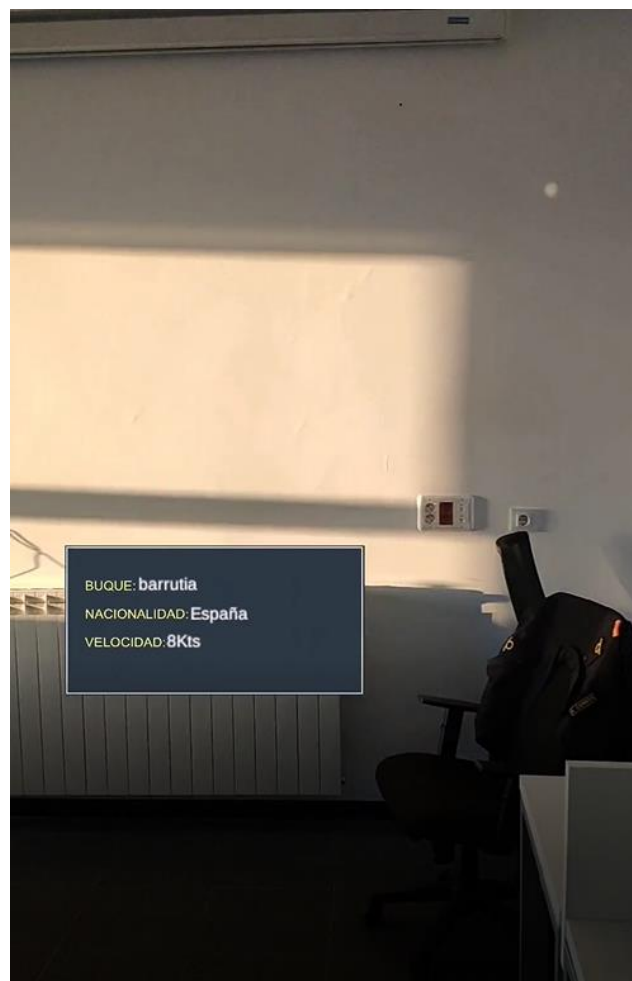


Ilustración 30 tercera prueba del marcador en AR

Con esto ya puede representar nuestro marcador con los textos en la posición GPS que queramos. No obstante, vamos a hacer unos cambios en el script Demo para mejorar la representación.

Mejoras del script Demo

Lo que ocurría anteriormente era que el marcador se salía de la pantalla si no lo estábamos apuntando. Las siguientes líneas de código tienen como objetivo establecer límites para la posición de los elementos de la interfaz de usuario, para evitar que salgan de la pantalla y mantenerlos siempre en el rango de visión.

```

29 void Update()
30 {
31
32     float minX = img.GetPixelAdjustedRect().width / 2;
33
34     float maxX = Screen.width - minX;
35
36
37     float minY = img.GetPixelAdjustedRect().height / 2;
38
39     float maxY = Screen.height - minY;
40
41     Vector2 pos = Camera.main.WorldToScreenPoint(target.position) + offset;
42
43
44     if (Vector3.Dot((target.position - transform.position), transform.forward) < 0)
45     {
46
47         if (pos.x < Screen.width / 2)
48         {
49
50             pos.x = maxX;
51         }
52         else
53         {
54
55             pos.x = minX;
56         }
57     }
58
59
60     pos.x = Mathf.Clamp(pos.x, minX, maxX);
61     pos.y = Mathf.Clamp(pos.y, minY, maxY);

```

Ilustración 31 mejora script Demo

```
29 void Update()
30 {
31
32     float minX = img.GetPixelAdjustedRect().width / 2;
33
34     float maxX = Screen.width - minX;
35
36
37     float minY = img.GetPixelAdjustedRect().height / 2;
38
39     float maxY = Screen.height - minY;
```

Ilustración 32 captura 1 Script Demo

En estas líneas de código de la Ilustración 32 calculo la posición mínima en el eje X,Y para la imagen img. La posición mínima es la mitad del ancho de la imagen dividido por 2. La posición máxima es igual al ancho de la pantalla menos la posición mínima calculada anteriormente.

```
41 Vector2 pos = Camera.main.WorldToScreenPoint(target.position) + offset;
42
43
44 if (Vector3.Dot((target.position - transform.position), transform.forward) < 0)
45 {
46
47     if (pos.x < Screen.width / 2)
48     {
49
50         pos.x = maxX;
51     }
52     else
53     {
54
55         pos.x = minX;
56     }
57 }
```

Ilustración 33 captura 2 Script Demo

Estas líneas de código de la Ilustración 33 comprueban si el objeto referenciado por target está detrás de la cámara Camera.main comparando la dirección del vector entre la posición del objeto y la posición de la cámara (target.position - transform.position) con la dirección hacia adelante de la cámara (transform.forward). Si el objeto está detrás de la cámara, entonces se comprueba si la posición en X de pos está a la izquierda de la mitad de la pantalla. Si es así, se establece la posición en X de pos en la posición máxima. De lo contrario, se establece la posición en X de pos en la posición mínima.

```
60 pos.x = Mathf.Clamp(pos.x, minX, maxX);
61 pos.y = Mathf.Clamp(pos.y, minY, maxY);
```

Ilustración 34 captura 3 Script Demo

Estas líneas de código de la Ilustración 34 aseguran que la posición pos no se salga de los límites de la pantalla. La posición en X se ajusta para que esté dentro del rango entre minX y maxX, mientras que la posición en Y se ajusta para que esté dentro del rango entre minY y maxY.

De esta forma conseguimos el efecto de la Ilustración 35.

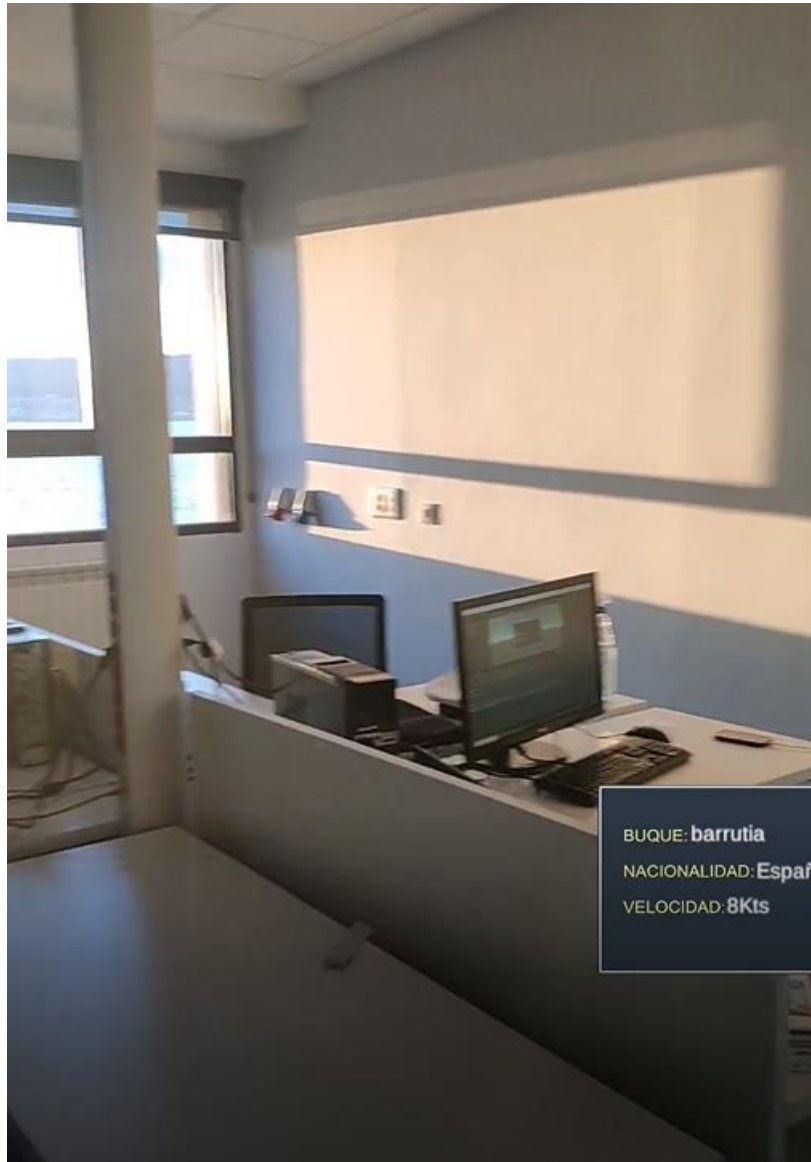


Ilustración 35 prueba de mejora de marcador en AR

Como vemos en el marcador en la Ilustración 35 queda en a la derecha de la pantalla, indicando que se encuentra en esta dirección. Esto nos sirve de guía para ver donde está colocando exactamente el marcador.

Mejora del marcador

Tras haber llevado a cabo la mejora del script Demo, se procede a mejorar la apariencia visual del marcador. Para ello, se ha decidido utilizar la herramienta de edición de imágenes Photoshop junto con Goodnotes.

3.1.7

Este proceso de mejora estética tiene como objetivo mejorar la presentación del marcador, de tal forma que los usuarios puedan disfrutar de una experiencia visual más agradable y atractiva mientras utilizan la aplicación. Se ha tenido en cuenta que una buena estética puede tener un impacto significativo en la percepción general de la calidad del producto por parte de los usuarios.

Por lo tanto, se ha tenido en cuenta factores como la legibilidad, la claridad y la coherencia visual para asegurarse de que el marcador sea más atractivo para los usuarios siendo este el resultado:

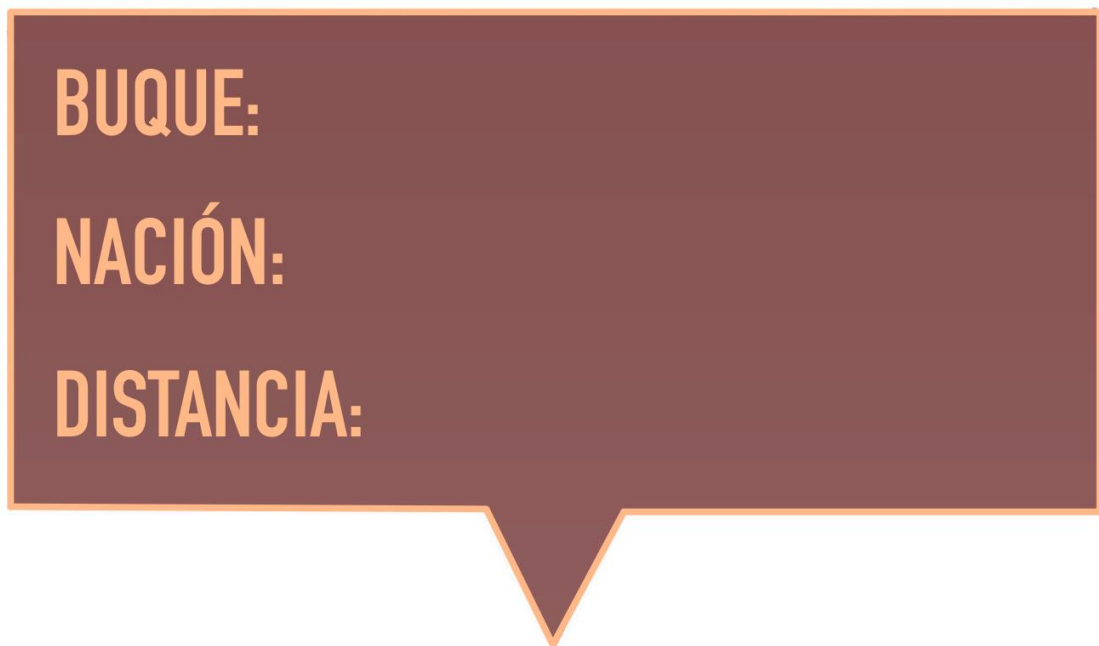


Ilustración 36 Nuevo marcador



Ilustración 37 marcador nuevo en AR

Como se aprecia en la Ilustración 36 se han cambiado los colores y se cambiado el parámetro de velocidad por el de distancia. Esta distancia se actualiza en tiempo real a medida que nos acercamos o alejamos del punto marcado y se ha calculado con las siguientes líneas de código de la Ilustración 38

```
distancia.transform.position = new Vector2(pos.x + 80.1f, pos.y - 50f);  
distancia.text = ((int)Vector3.Distance(target.position, Camera.main.transform.position)).ToString() + "m";
```

Ilustración 38 código sobre distancia

Este código Ilustración 38 está estableciendo la posición de un elemento de texto en la interfaz de usuario (UI) llamado "distancia". La primera línea establece la posición del elemento de texto en un nuevo Vector2 que se calcula sumando 80.1 a la coordenada x de la posición actual y restando 50 de la coordenada y de la posición actual. Esto es simplemente para ajustar bien su posición en el marcador.

La siguiente línea de código establece el texto del elemento de texto "distancia". El texto muestra la distancia entre la posición actual de la cámara y la posición del objeto de destino (target). La distancia se convierte en un entero mediante la función "(int)", y se agrega la unidad de medida "m" al final del texto.

Además de la distancia. He añadido una lista de banderas. En función de que país sea representa una bandera u otra. Esto se ha conseguido con el código de la Ilustración 39.

```
void Start()
{
    Buque.text = "John Paul K.";
    Nacionalidad.text = "Francia";

    // Inicializa el diccionario de banderas
    banderas = new Dictionary<string, Sprite>();
    banderas["España"] = banderaEspanaSprite;
    banderas["Francia"] = banderaFranciaSprite;
    banderas["Italia"] = banderaItaliaSprite;
    banderas["Alemania"] = banderaAlemaniaSprite;
    banderas["Estados Unidos"] = banderaUSA;
    banderas["China"] = banderaChina;
    banderas["Inglaterra"] = banderaInglaterra;
    banderas["Rusia"] = banderaRusia;
    banderas["Australia"] = banderaAustralia;
    banderas["Grecia"] = banderaGrecia;
    banderas["Egipto"] = banderaEgipto;
    banderas["México"] = banderaMexico;
```

Ilustración 39 diccionario de banderas

Podemos ver un código que define un objeto de tipo Dictionary<string, Sprite>, que es una estructura de datos que permite almacenar una colección de elementos de tipo clave-valor. En este caso, la clave es un string que representa el nombre del país y el valor es un objeto Sprite que representa la imagen de la bandera del país.

Luego, se agregan elementos al diccionario usando el método Add y se asigna a cada clave un valor correspondiente que representa la imagen de la bandera del país. En este ejemplo, se agregan banderas de España, Francia, Italia, Alemania, Estados Unidos, China, Inglaterra, Rusia, Australia, Grecia, Egipto y México.

```
// Obtén la nación del texto de Nacionalidad
string nacion = Nacionalidad.text;

// Si la nación tiene una bandera en el diccionario, muéstrala en la imagen de la bandera
if (banderas.ContainsKey(nacion))
{
    banderaImage.sprite = banderas[nacion];
}
```

Ilustración 40 código identificación bandera

Este es otro fragmento (véase Ilustración 40) de código en lenguaje de programación que se encarga de asignar una imagen de bandera a un objeto Image en función de la nacionalidad ingresada por el usuario.

Primero, se obtiene la nacionalidad ingresada por el usuario y se almacena en la variable nación.

Luego, se verifica si el diccionario banderas contiene una clave que coincide con la nacionalidad ingresada por el usuario usando el método ContainsKey(). Si existe una clave en el diccionario que coincide con la nacionalidad ingresada, se obtiene el valor correspondiente (que es un objeto Sprite) y se asigna a la propiedad sprite del objeto Image banderaImage, lo que cambia la imagen de la bandera que se muestra en la pantalla.

Si la nacionalidad ingresada no se encuentra en el diccionario, la imagen de la bandera no se cambia y se mantiene la que estaba previamente.

En la interfaz de unity, se crean los siguientes campos, como podemos ver en la Ilustración 41.

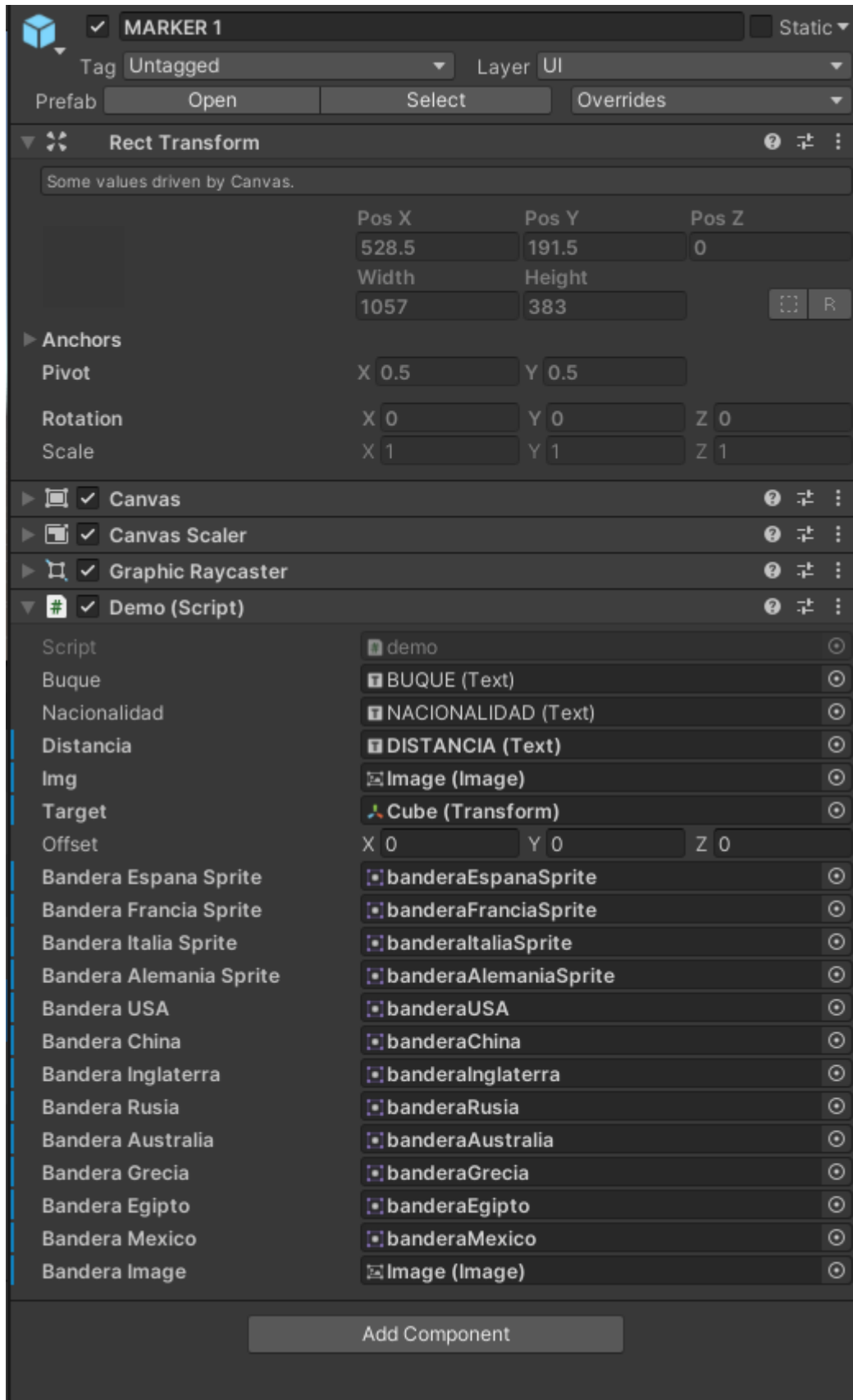


Ilustración 41 interfaz de unity con banderas

Estos campos simplemente se han asociado con las banderas correspondientes. Es un trabajo tedioso pero se podrían cargar todas las banderas del mundo. No obstante, ahora mismo no procede y simplemente como prueba se han cargado en el programa las banderas de los países más comunes.

Marcadores de faros y peligros

Además de la del marcador previamente mencionado, se ha desarrollado un prototipo adicional de marcador que tiene la capacidad de señalar los faros y peligros a la navegación. Este nuevo marcador ha sido diseñado específicamente para mejorar la seguridad de la navegación, y proporciona información crucial sobre los obstáculos potenciales que los navegantes pueden encontrar en su camino.

El objetivo principal de este marcador es alertar a los navegantes sobre la presencia de faros y otros peligros en la zona de navegación. La información proporcionada por este marcador incluye la ubicación exacta del faro o del peligro, la distancia aproximada desde la ubicación actual del usuario y una imagen simple del objeto señalado.



Ilustración 42 nuevos marcadores

Estos serían dos ejemplos (vease Ilustración 42). El marcador de la izquierda serviría para señalar un faro y el de la derecha para señalar un pecio o un bajo. La idea es que dependiendo de que sea lo que se quiera señalar se utilice una imagen u otra. De esta forma se pueden señalar todo tipo de elementos (faros, boyas, cables submarinos, bateas...)

A continuación, veremos el código de estos marcadores en la Ilustración 43.

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

public class faros : MonoBehaviour
{
    public Text distancia;
    public Image img;
    public Transform target;
    public Vector3 offset;
    public Image icono;
    // Start is called before the first frame update
    void Start()
    {
        // Update is called once per frame
    }

    void Update()
    {
        Vector2 pos = Camera.main.WorldToScreenPoint(target.position + offset);

        img.transform.position = pos;
        icono.transform.position = new Vector2(pos.x + 69f, pos.y + 5f);
        distancia.transform.position = new Vector2(pos.x + 43f, pos.y - 6f);
        distancia.text = ((int)Vector3.Distance(target.position, Camera.main.transform.position)).ToString() + "m";
    }
}
```

Ilustración 43 código marcadores nuevos

Este código tiene una estructura parecida al que se ha visto anteriormente. En las primeras líneas se declaran cinco variables públicas: *distancia*, *img*, *target*, *offset* e *icono*. La variable *distancia* es de tipo *Text* y se utiliza para mostrar la distancia entre el objeto asociado y la cámara. La variable *img* es de tipo *Image* y se utiliza para mostrar una imagen de UI que se moverá para estar sobre el objeto asociado. La variable *target* es de tipo *Transform* y se utiliza para almacenar el objeto asociado al script. La variable *offset* es de tipo *Vector3* y se utiliza para almacenar el desplazamiento de la imagen de UI con respecto al objeto asociado. La variable *icono* es de tipo *Image* y se utiliza para mostrar el símbolo de lo que el marcador señala (faro, pecio...).

En el método *Update()*, primero se llama a la función *WorldToScreenPoint()* de la cámara principal para obtener la posición en la pantalla del objeto asociado con el desplazamiento almacenado en la variable *offset*. Se guarda la posición en la variable *pos* de tipo *Vector2*. Luego, se actualiza la posición de la imagen de UI asociada a la variable *img* utilizando la posición *pos*.

También se actualiza la posición de la segunda imagen de UI asociada a la variable *icono*. Se agrega un valor de 69 unidades a la posición *x* y un valor de 5 unidades a la posición *y* para desplazar la imagen hacia la derecha y hacia arriba con respecto a la imagen principal.

Se actualiza la posición del texto en la imagen de UI asociada a la variable *distancia*. Se agrega un valor de 43 unidades a la posición *x* y un valor de -6 unidades a la posición *y* para desplazar el texto hacia la derecha y hacia abajo con respecto a la imagen principal.

Finalmente, se actualiza el texto de la variable distancia utilizando la función *Distance()* de la clase *Vector3*, que calcula la distancia entre la posición del objeto asociado y la posición de la cámara principal. La distancia se convierte a un número entero y se agrega la "m" para mostrar la unidad de medida en metros (que es la unidad por defecto de unity). El resultado se asigna como texto en la variable distancia.

El funcionamiento es igual que con el marcador principal. Se le debe asociar un target y a este se le añade la posición GPS que queramos. Este sería un ejemplo de como uno de estos marcadores señalaría el bajo de Morrazán (véase Ilustración 44):



3.1.9

Ilustración 44 bajo de Morrazán

Problema con el Street view

ARCore es una plataforma de realidad aumentada desarrollada por Google que permite a los desarrolladores crear aplicaciones de realidad aumentada para dispositivos móviles. Aunque ARCore es una herramienta muy útil para crear aplicaciones de realidad aumentada, no está diseñada específicamente para ser utilizada en entornos marítimos. El uso de ARCore en este tipo de entorno puede presentar varios problemas.

Durante la ejecución de este trabajo se ha detectado que los objetos no se sitúan exactamente en las posiciones GPS que se introducen sino que se colocan en las posiciones transform de unity. Estas son coordenadas XYZ que se miden en metros. Inicialmente se consideró que la ubicación era correcta ya que el origen de las coordenadas (0,0,0) coincidía con la ubicación del laboratorio. No obstante, se notó que al ingresar otra posición, el marcador seguía colocándose en el mismo punto, lo que indicaba que el sistema de referencia utilizado no correspondía al sistema de coordenadas GPS.

Tras conversar con Camilo Medina, propietario del canal de Youtube "Unity Adventure" (canal que ha sido de gran ayuda por sus numerosos tutoriales), se llegó a la conclusión de que el VPSManager utilizado en el trabajo no funciona adecuadamente, ya que este no utiliza la función Street View. El Street view es un mapeo de imágenes panorámicas realizado por Google en casi todo el mundo. Para crear Street View, los vehículos de Google equipados con cámaras de alta resolución capturan imágenes a medida que se desplazan por las calles. Esto implica que, si la aplicación no se utiliza en calles donde haya Street view, las coordenadas GPS no funcionarán de forma adecuada. Como la aplicación se desarrolla en un entorno marítimo, las coordenadas GPS no serán válidas.

La solución empleada consistió en utilizar la técnica de proyección mercatoriana, la cual se caracteriza por ser una proyección cilíndrica modificada que se aplica sobre la superficie terrestre. En este método, la tierra se proyecta sobre un cilindro que circunda el ecuador, lo que permite una representación plana de la misma. Cabe destacar que las proyecciones mercatorianas son ampliamente utilizadas en la actualidad para la elaboración de cartas náuticas y otros mapas, gracias a su capacidad para mostrar áreas extensas de la superficie terrestre de manera precisa y proporcional.

A continuación se va a mostrar el código que he utilizado para conseguir usar proyecciones mercator (véase Ilustración 45):

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Posicionador : MonoBehaviour
6  {
7      // Establece las coordenadas de latitud y longitud del punto en el que se colocará el prefab
8      float latitude = 42.3821514774857f;
9      float longitude = -8.775349251516671f;
10
11     // Establece la escala de conversión de latitud / longitud a coordenadas X-Y
12     float latitudeScale = 111320f;
13     float longitudeScale = -68960f;
14
15     // Establece el punto de origen de las coordenadas XY
16     Vector2 originPoint = new Vector2(42.39776854844385f, -8.706614083417586f);
17
18     // Prefab a ser posicionado
19     public GameObject prefab;
20
21     void Start()
22     {
23         // Calcula las coordenadas XY del punto de latitud y longitud especificado
24         float deltaX = (longitude - originPoint.y) * longitudeScale;
25         float deltaY = (latitude - originPoint.x) * latitudeScale;
26         Vector2 newPosition = new Vector2(deltaX, deltaY);
27
28         // Establece la posición del prefab en las coordenadas XY calculadas
29         prefab.transform.position = new Vector3(newPosition.x, 250f, newPosition.y);
30     }
31

```

Ilustración 45 proyección mercator

Con este código se consigue calcular la distancia en metros en las coordenadas XYZ a partir de dos posiciones GPS. Cabe destacar del código las líneas 12 y 13. Estas líneas corresponden con el factor por el que hay que multiplicar la diferencia de latitudes y longitudes. Simplemente debemos saber que un grado de latitud es aproximadamente 111,32 Km. No obstante, un grado de longitud cambia dependiendo del punto de la tierra en el que nos encontremos. De esta forma si estamos en el ecuador seguirán siendo 111,32 Km. Si no es así, seguirá la siguiente fórmula:

$$\text{longitud en metros} = \pi * \text{radio de la Tierra} * \cos(\text{latitud}) / 180$$

En nuestro caso, para la latitud donde nos encontramos, un grado de longitud son 68,96 Km en proyección mercatoriana.

En la línea 19 se introduce el prefab que queremos posicionar. Este prefab debe ser evidentemente el mismo Target visto en la Ilustración 27. El resultado a la hora de representar es muy bueno (véase Ilustración 44).

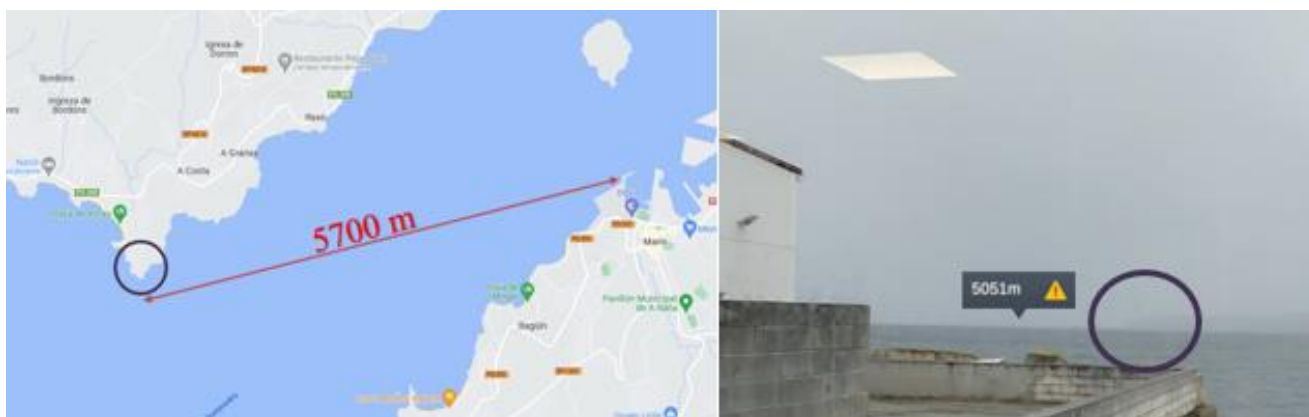


Ilustración 46 foto comparativa morrazán

Como se puede ver en la Ilustración 46 la posición es bastante precisa en demora. La distancia tiene un error de unos 650m. Este error se atribuye a una sucesión de errores a la hora de escoger los puntos GPS, en donde han sido cogidos de forma aproximada sin ser muy exhaustivo.

Como último paso, para optimizar el código se ha sido fusionar el código de la Ilustración 45 y el de la Ilustración 10. Se busca de esta manera mayor simplicidad.

```

public class VPSManager : MonoBehaviour
{
    [SerializeField] private AREarthManager earthManager;

    [Serializable]
    public struct GeospatialObject
    {
        public GameObject ObjectPrefab;
        public EarthPosition EarthPosition;
    }

    // Establece las coordenadas de latitud y longitud del punto en el que se colocará el prefab
    float latitude = 42.3821514774857f;
    float longitude = -8.775349251516671f;

    // Establece la escala de conversión de latitud / longitud a coordenadas X-Y
    float latitudeScale = 111320f;
    float longitudeScale = -68960f;

    // Establece el punto de origen de las coordenadas XY
    Vector2 originPoint = new Vector2(42.39776854844385f, -8.706614083417586f);

    [Serializable]
    public struct EarthPosition
    {
        public double Latitude;
        public double Longitude;
        public double Altitude;
    }

    [SerializeField] private ARAnchorManager aRAnchorManager;
    [SerializeField] private List<GeospatialObject> geospatialObjects = new List<GeospatialObject>();
    // Start is called before the first frame update
    void Start()
    {
        VerifyGeospatialSupport();
        Posicionador();
    }
}

```

Ilustración 47 VPSManger nuevo Parte 1

```
private void VerifyGeospatialSupport()
{
    var result = earthManager.IsGeospatialModeSupported(GeospatialMode.Enabled);

    switch (result)
    {
        case FeaturesSupported.Supported:
            Debug.Log("Ready to VPS");
            PlaceObjects();
            break;
        case FeaturesSupported.Unknown:
            Debug.Log("Unknown...");
            Invoke("VerifyGeospatialSupport", 5.0f);
            break;
        case FeaturesSupported.Unsupported:
            Debug.Log("Unsupported");
            break;
    }
}

private void PlaceObjects()
{
    if (earthManager.EarthTrackingState == TrackingState.Tracking)
    {
        var geospatialPose = earthManager.CameraGeospatialPose;

        foreach (var obj in geospatialObjects)
        {
            var earthPosition = obj.EarthPosition;
            var objAnchor = ARAnchorManagerExtensions.AddAnchor(arAnchorManager, earthPosition.Latitude, earthPosition.Longitude, earthPosition.Altitude, Quaternion.identity);
            Instantiate(obj.ObjectPrefab);
        }
    }
    else if (earthManager.EarthTrackingState == TrackingState.None)
    {
        Invoke("PlaceObjects", 5.0f);
    }
}

private void Posicionador()
{
    foreach (var obj in geospatialObjects)
    {
        // Calcula las coordenadas XY del punto de latitud y longitud especificado
        float deltaX = (longitude - originPoint.y) * longitudeScale;
        float deltaY = (latitude - originPoint.x) * latitudeScale;
        Vector2 newPosition = new Vector2(deltaX, deltaY);
        // Establece la posición del prefab en las coordenadas XY calculadas
        obj.ObjectPrefab.transform.position = new Vector3(newPosition.x, 250f, newPosition.y);
    }
}
```

Ilustración 48 VPSManger nuevo Parte 2

El principal cambio ha sido que el objeto que se posiciona en la función *posicionador()* es el mismo que se instancia en la función *PlaceObjects()*. Esto hace que se simplifique el programa, ya que antes había que declarar un objeto dos veces, primero a instanciar y ese mismo también luego a posicionar. Ahora solo hay que declararlo una vez.

Mejora de aspecto e implementación de tipo de buque

Se ha realizado algunas modificaciones en el marcador que se veían necesarias. En particular, se ha cambiado el color para hacerlo más oscuro y se ha eliminado los textos, ya que lo consideramos redundante. Este es el resultado (véase Ilustración 49):



Ilustración 49 marcador definitivo

También se aprecia un icono de un barco de guerra. Esta nueva mejora permite colocar una imagen del tipo de barco que se está marcando. A continuación, se va a explicar como lo he realizado:

```
public Dictionary<string, Sprite> tipos;

public Sprite tipopesquero;
public Sprite tipocarguero;
public Sprite tipovelero;
public Sprite tipowarship;

public Image tipoimage;
```

Ilustración 50 código de iconos de barco parte 1

La Ilustración 50 muestra en su primera línea de código la creación de un diccionario público llamado "*tipos*", que contiene pares clave-valor de tipo string y Sprite, respectivamente. Luego, las siguientes cuatro líneas de código crean variables públicas de tipo Sprite llamadas "*tipopesquero*", "*tipocarguero*", "*tipovelero*" y "*tipowarship*", respectivamente, y les asignan cada una una imagen de sprite. Por último, se crea una variable pública de tipo *Image* llamada "*tipoimage*". Decir que anterior a esto, se declara un texto llamado *Tipo*.

```
Tipo.text = "Carguero";

// Inicializa el diccionario de tipos de barco
tipos = new Dictionary<string, Sprite>();
tipos["Pesquero"] = tipopesquero;
tipos["Carguero"] = tipocarguero;
tipos["Velero"] = tipovelero;
tipos["Warship"] = tipowarship;
```

Ilustración 51 código de iconos de barco parte 2

El primer código de la Ilustración 51 establece el texto del componente de texto llamado "*Tipo*" a "*Carguero*". El segundo código inicializa el diccionario de tipos de barco creando un nuevo objeto *Dictionary<string, Sprite>()*. Luego, se asignan las imágenes de sprite correspondientes a cada tipo de

barco en el diccionario usando los nombres de clave "Pesquero", "Carguero", "Velero" y "Warship" y sus respectivas variables de sprite. En un principio estos son los tipos que se van a incluir. No obstante ocurre como con las banderas, puede haber muchos más tipos de buques. No obstante, por ahora servirá.

```
// Obtén la nación del texto de Nacionalidad y de tipo
string nacion = Nacionalidad.text;
string tipo = Tipo.text;

// Si la nación tiene una bandera en el diccionario, muéstrala en la imagen de la bandera. Lo mismo para el tipo
if (banderas.ContainsKey(nacion))
{
    banderaImage.sprite = banderas[nacion];
}

if (tipos.ContainsKey(tipo))
{
    tipoImage.sprite = tipos[tipo];
}
```

Ilustración 52 código de iconos de buque parte 3

Esta última parte, (Ilustración 52) el código primero almacena el texto actual del componente de texto llamado *Tipo* en la variable *tipo*. Luego, el código verifica si el diccionario de tipos de barco llamado *tipos* contiene una clave que coincide con el valor de la variable *tipo* usando el método *ContainsKey()*. Si la clave existe en el diccionario, entonces se asigna el sprite correspondiente a la variable *tipoImage.sprite* usando el valor de la clave del diccionario *tipos[tipo]*.

Como se puede apreciar, estas líneas de código siguen el mismo funcionamiento que la ilustración de la bandera. En definitiva, el código identifica el tipo de buque a marcar. Si este tipo se encuentra en el diccionario de tipos entonces la aplicación coloca un icono del mismo en la parte de abajo a la derecha.

4 RESULTADOS Y VALIDACIÓN

4.1 Pruebas de la aplicación

Embarcaciones

Este apartado consiste en mostrar como funciona la aplicación y en como se visualiza el marcador de embarcaciones.

4.1.1



Ilustración 53 Tabarca

En la Ilustración 53 podemos ver el marcador a una distancia cercana, señalando el Patrullero Tabarca, buque de guerra de la Armada española. La distancia real era de 160m aproximadamente, por lo que el error es de 120m. El tiempo que tomó en posicionarse fue de aproximadamente 4 minutos y no permaneció mucho tiempo en esa posición.

Peligros y faros



Ilustración 54 Tambo

En la Ilustración 54 podemos ver dos marcadores. El de la derecha marcando el faro de Tambo y el de la izquierda un marcador para los cables submarinos al oeste de la isla. La distancia real al faro es de 1154m por lo que esta vez ha sido bastante más preciso. Los cables submarinos también han sido fielmente representados. El tiempo que tomó para colocarse adecuadamente fue casi instantáneo, destacando que permaneció en esa posición sin descuadrarse.

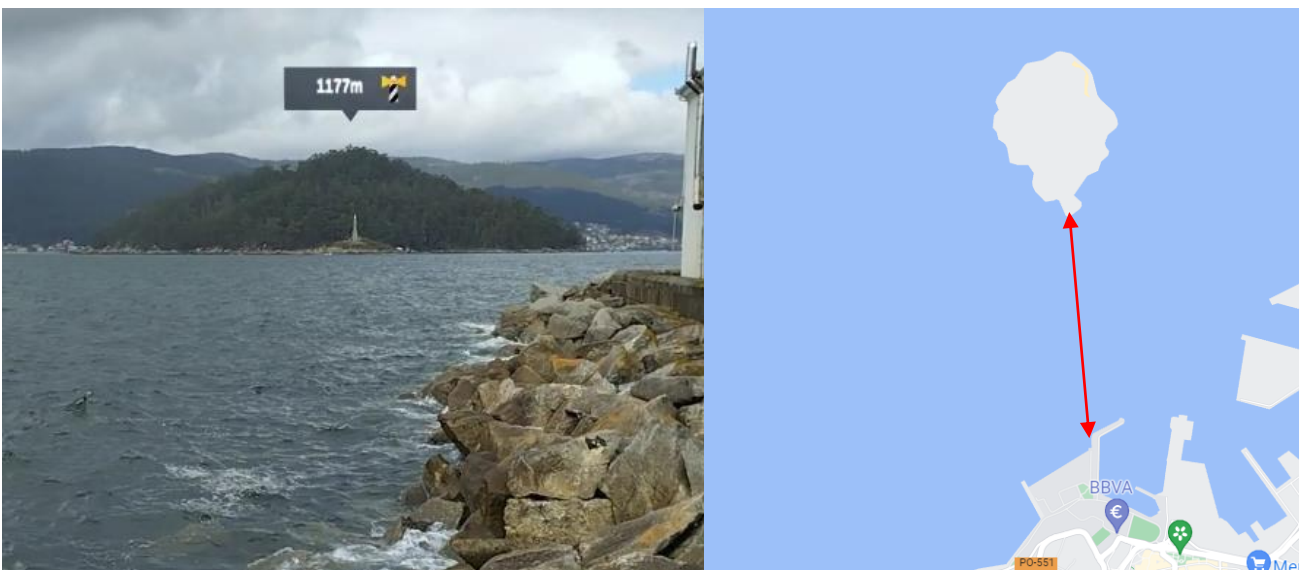


Ilustración 55 tambo comparativa

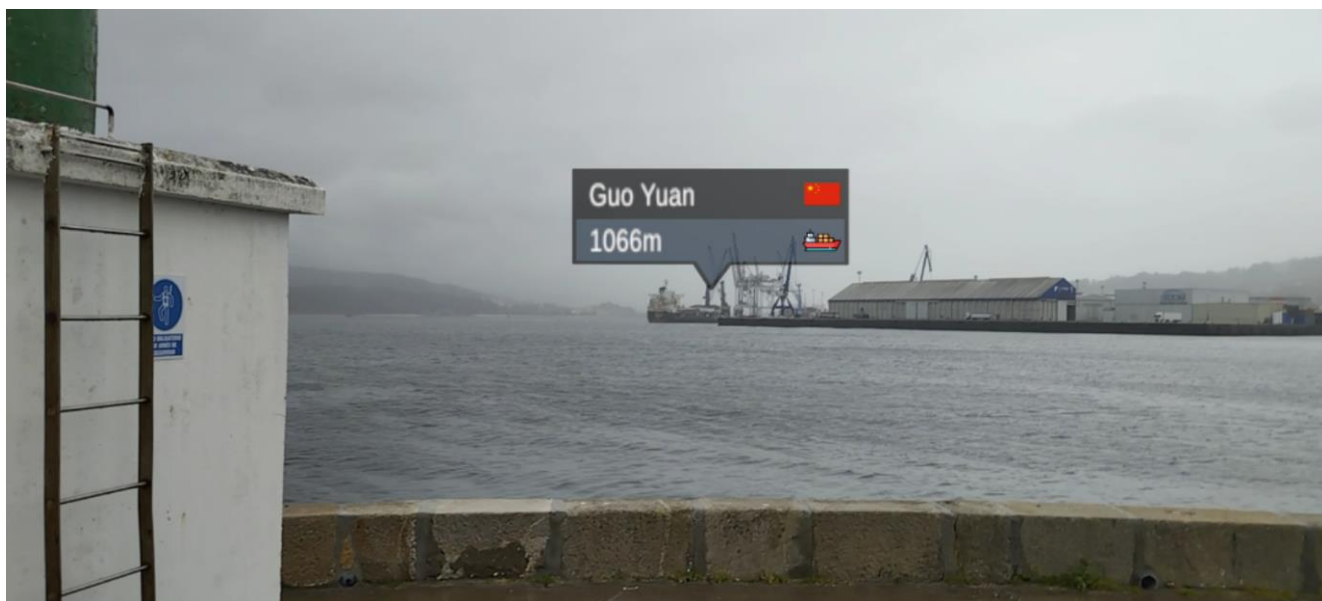


Ilustración 56 carguero atracado

En la Ilustración 56 podemos ver el marcador del carguero chino Guo Yuan. Se encuentra atracado en el puerto de Marín a una distancia de 960m, por lo que el error en distancia es de 106m. en la Ilustración 57 se ve que en demora también está fielmente representado. El tiempo que tomó para colocarse adecuadamente el marcador fue de 3 minutos, destacando que no permaneció en su posición durante mucho tiempo, volviéndose a descuadrarse pasado un minuto aproximadamente.

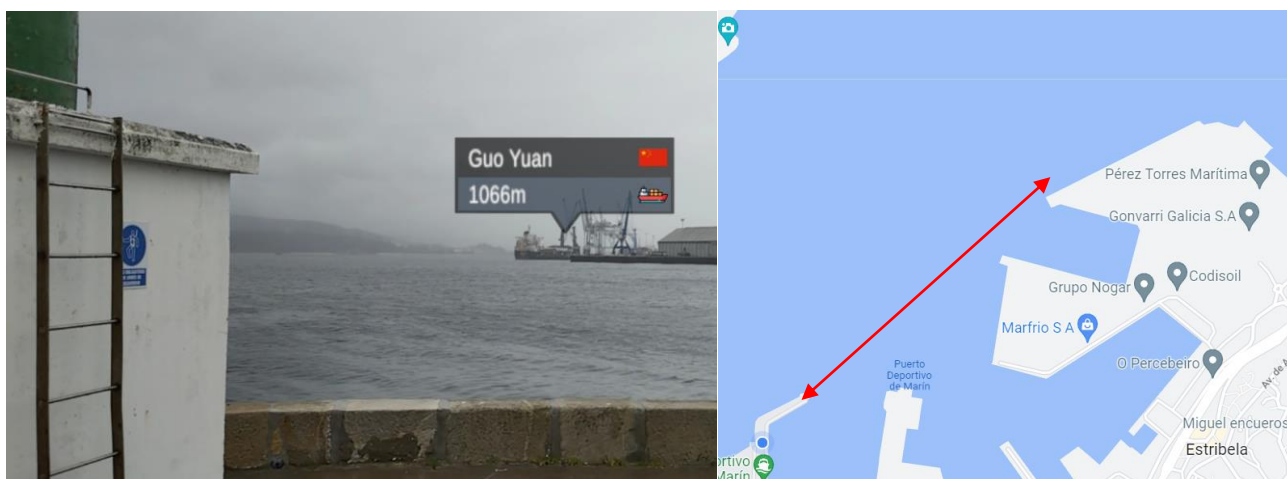


Ilustración 57 comparativa carguero



Ilustración 58 Giralda atracado

En la Ilustración 58 se puede ver el velero Giralda atracado en el muelle de la Escuela. Se encuentra a 100m por lo que el error a sido mayor (265m). En demora se encuentra bien representado. No obstante, tardó unos 4 minutos en posicionarse adecuadamente permaneciendo unos 2 minutos hasta volver a descolocarse.

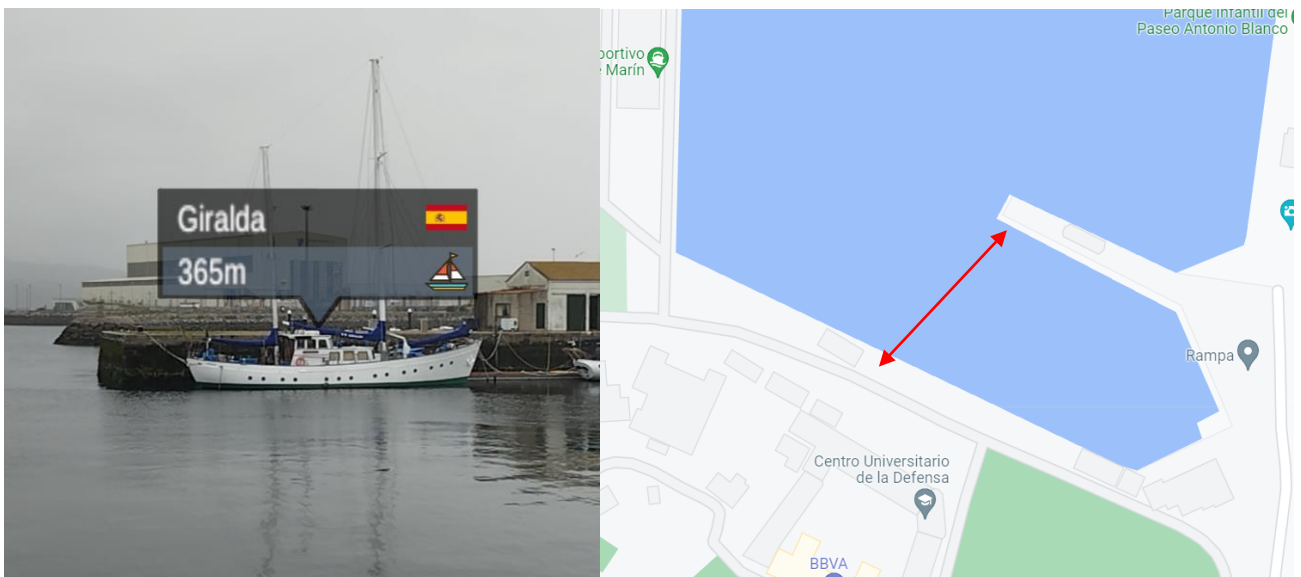


Ilustración 59 comparativa Giralda

4.2 Validación

El presente texto tiene como objetivo analizar los problemas que surgen al ejecutar la aplicación. Uno de los principales inconvenientes es la demora al colocar correctamente el marcador donde el usuario desea. En algunos casos, este proceso puede tardar unos pocos segundos, sin embargo, lo más frecuente es que el tiempo de espera se extienda a varios minutos.

Se ha observado también que la aplicación funciona mejor con blancos lejanos. Cuando el marcador debe colocarse a menos de 200m tarda considerablemente más. Además, tiene tendencia a no permanecer adecuadamente colocado mucho tiempo. Esto con blancos lejanos no ocurre.

Asimismo, otro problema que afecta el desempeño de la aplicación es la necesidad de reiniciarla en ocasiones para que funcione correctamente. Esta situación resulta incómoda y poco eficiente para el usuario.

Otro factor limitante de la aplicación es que requiere de una conexión a Internet para su correcto funcionamiento. En este sentido, si el usuario no dispone de acceso a wifi o una red de datos móviles, no podrá ejecutar la aplicación, lo que limita su uso en ciertas circunstancias y lugares. Siendo la mar un lugar crítico pues hay que disponer de medios especiales para el acceso a la red.

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Conclusiones

En este proyecto se establecieron tres objetivos principales que buscaban investigar y desarrollar una aplicación de realidad aumentada para su uso en la Armada. A continuación, se presentarán las conclusiones obtenidas en cada uno de estos objetivos, destacando las principales contribuciones y limitaciones del proyecto.

5.1.1 *Estado del arte*

Se ha cumplido con éxito el objetivo de realizar un estado del arte sobre la realidad aumentada, donde se han presentado diferentes plataformas de desarrollo que existen en el mercado y se ha analizado su pertinencia para este proyecto en cuestión. Además, se ha evaluado los dispositivos capaces de soportar aplicaciones de este tipo, en donde también se ha llegado a conclusiones claras. Este análisis ha permitido sacar conclusiones sobre cuál es la plataforma más adecuada para el desarrollo de esta aplicación de AR.

Además, durante el proceso de investigación se han identificado diferentes áreas de aplicación del AR en la armada, lo que ha permitido tener una visión más amplia de las posibilidades que ofrece esta tecnología en el ámbito militar. Se ha explorado cómo el AR puede ser utilizado en la formación de buceadores, en la simulación de situaciones de combate, en la ejecución de ejercicios y en otras aplicaciones relevantes.

Todo este proceso de investigación ha permitido aportar una visión más completa sobre las aplicaciones del AR, lo que sin duda puede ser de gran utilidad para futuros proyectos en este ámbito. Además, se ha expuesto algunas de las múltiples aplicaciones que puede tener en la Armada. Algunas pueden sonar ambiciosas, otras sin embargo tras este proyecto se puede concluir en que son más alcanzables. Por ejemplo, la posible aplicación para hacer canales dragadas.

La aplicación

Otro objetivo de este proyecto fue el desarrollo de una aplicación capaz de identificar buques y pecios en tiempo real mediante la conexión con un servidor de Elastic Search. Aunque el resultado final de la aplicación no cumple completamente con este requisito, se logró crear una versión offline de la aplicación que permite al usuario introducir los datos y representar información sobre buques, pecios y faros en unas imágenes de interfaz de usuario (2D) y crean una apariencia muy práctica a la hora de mostrar los datos al usuario.

La versión offline de la aplicación es capaz de mostrar información sobre los buques y pecios introducidos por el usuario, incluyendo su posición, la distancia a la misma, nacionalidad y tipo de buque. Pudiéndose mostrar muchos más datos de interés como la velocidad, su origen y destino...

Evaluación de la aplicación

5.1.3

La aplicación funciona correctamente. Es capaz de representar los marcadores en las posiciones predefinidas, así como también permite al usuario colocar varios marcadores al mismo tiempo. Las distancias en general son aproximadas a la realidad.

Sin embargo, se ha observado que la aplicación tarda un tiempo en posicionarse correctamente, lo que puede generar cierta demora en su funcionamiento. En algunos casos, se ha reportado que el proceso de posicionamiento puede llevar varios minutos, incluso requiriendo reiniciar la aplicación. Es importante considerar esta situación para mejorar la eficiencia y la experiencia del usuario al utilizar la aplicación en unas líneas futuras.

Como conclusión, podemos dar por completado el objetivo puesto que la aplicación cumple con los requerimientos iniciales. No obstante, se considera que lo ideal sería que su eficacia a la hora de posicionar funcionase de manera más adecuada. Hay que tener en cuenta que no se busca una aplicación que sea capaz de incorporarse en las unidades de la armada, sino que se busca solo que sea capaz de posicionar marcadores en el lugar indicado. Como esto si que lo cumple, damos como completado el objetivo.

5.2 Líneas futuras

Como líneas futuras se pone en primer plano que la aplicación sea capaz de conectarse al servidor de Elastic Search para que sea capaz de representar en tiempo real los marcadores sobre un buque en movimiento. Se debe tener especial cuidado en tener una buena conexión a Internet, puesto que se especula que el nivel exigencia de tráfico de datos será alto.

Como línea futura también se podría realizar una unificación de este proyecto y el de mi compañero Jesus Moreno “Aplicacion de tecnicas de aprendizaje profundo a la localizacion y seguimiento de objetos

en entornos marítimos empleando cámaras de video portátiles”. Consiste en la localización positiva de un buque mediante inteligencia artificial y a través de una cámara digital. La idea es que la futura aplicación reconozca un buque y muestre sus datos mediante el servidor de Elastic Search. De tal forma que la futura aplicación podría hacer una localización, un seguimiento y una identificación.

Como última propuesta para las líneas futuras, se propone el desarrollo de una aplicación para los ejercicios de canales dragadas, descrito en el estado del arte de este proyecto.

6 BIBLIOGRAFÍA

- [1] J. Rodríguez y V. Fernández, Cómo redactar el estado del arte de un trabajo, Editorial Genios, 2010.
- [2] P. Martínez y A. García, Cómo escribir una buena memoria de TFG, Publicaciones del 2000, 2013.
- [3] A. Pérez, Cómo escribir una bibliografía, Nuevas publicaciones.
- [4] «Web de La Moncloa,» [En línea]. Available: <http://www.lamoncloa.gob.es>. [Último acceso: 13 enero 2015].
- [5] «new york times-pokemon go,» [En línea]. Available: <https://www.nytimes.com/2020/01/01/world/canada/pokemon-go-canada-military.html>.
- [6] «ARcore,» [En línea]. Available: <https://developers.google.com/ar?hl=es-419>.
- [7] «apple,» [En línea]. Available: <https://developer.apple.com/augmented-reality/>.
- [8] «Vuforia,» [En línea]. Available: <https://developer.vuforia.com/>.
- [9] «Armada Española,» [En línea]. Available: <https://armada.defensa.gob.es/ArmadaPortal/page/Portal/ArmadaEspañola/buquessuperficie/prefLang-es/16UBC--03unidadbuceoferrol>.
- [10] «visión del AJAL,» [En línea]. Available: file:///C:/Users/angel/Downloads/15122022_VISION%20DE%20AJAL%20ARSENAL%20INTELIGENTE_VONLINE%20firmado.pdf.
- [11] «MarineTraffic,» [En línea]. Available: MarineTraffic.com.
- [12] «Neptuno,» [En línea]. Available: <https://www.neptuno.es/que-es-un-sistema-ais/>.
- [13] «ONNAUTIC,» [En línea]. Available: <https://onnautic.com/blog/que-es-el-sistema-de-identificacion-automatica-o-ais/>.

ANEXO I: IMPLICACIONES SOCIALES, Y/O ECONÓMICAS, Y/O AMBIENTALES

Por un lado, las implicaciones sociales de esta aplicación de AR para posicionar marcadores sobre buques, faros, pecios o naufragios pueden ser muy positivas al proporcionar información valiosa a las personas que trabajan en la industria marítima, ya que podrán identificar fácilmente los objetos cercanos y conocer más detalles sobre ellos. Además, también puede ser muy útil para los turistas interesados en el patrimonio marítimo y la navegación, ya que les permitiría conocer más sobre el entorno marítimo.

Es importante tener en cuenta que cualquier aplicación tecnológica puede fallar. En el caso de la aplicación de AR, podría haber problemas técnicos que afecten su funcionamiento, como fallos en el posicionamiento de objetos o problemas de conectividad. Además, también pueden surgir problemas de precisión en la información proporcionada por la aplicación, lo que puede tener implicaciones graves para los navegantes y otros usuarios. Debe considerarse como una ayuda a la navegación la cual puede tener errores.

En cuanto a las implicaciones económicas, la aplicación podría tener un impacto positivo en la industria turística y de entretenimiento, ya que puede ayudar a promover la exploración de sitios marítimos y la realización de excursiones. Por ejemplo, para señalar un naufragio turístico. Además, la aplicación puede ser una herramienta útil para los trabajadores de la industria marítima, ya que puede ayudar a mejorar la eficiencia y la productividad en sus trabajos, señalando por ejemplo una zona de pesca.

Por último, las implicaciones ambientales de la aplicación también son importantes. Si bien la aplicación puede ser una herramienta útil para promover la conciencia y la comprensión del patrimonio marítimo, también podría desencadenar el aumento de la actividad humana en áreas marinas frágiles, lo que podría tener efectos perjudiciales sobre los ecosistemas marinos. Por ejemplo, si los turistas y otros visitantes no respetan los límites establecidos en ciertas áreas, pueden dañar los arrecifes de coral, la vida marina y otros recursos naturales.

En definitiva, aunque la aplicación puede ser una herramienta muy útil para promover la conciencia y la comprensión del patrimonio marítimo, es importante tener en cuenta las implicaciones sociales, económicas y ambientales que pueden derivarse de su uso. Es crucial trabajar para minimizar cualquier impacto negativo y asegurarse de que la aplicación se utilice de manera responsable.

ANEXO II: REFLEXIONES ÉTICAS Y SOCIALES

Una aplicación de Realidad Aumentada como esta en el entorno marítimo puede tener diversos usos éticos y sociales al ayudar a identificar buques que están realizando actividades ilícitas, como el tráfico de drogas o el contrabando.

Por ejemplo, una aplicación de AR podría ser utilizada para identificar rápidamente los buques que se encuentran en una zona determinada, lo que facilitaría la labor de los agentes de seguridad en el control de las actividades ilegales. La aplicación podría mostrar información sobre el buque en tiempo real, como su nombre, bandera, propietario, historial de navegación y otros datos relevantes. Además, podría tener acceso a bases de datos internacionales para verificar la legalidad de la actividad del buque.

Concluyendo, una aplicación de AR en el entorno marítimo puede tener un impacto positivo en la seguridad y el bienestar social al ayudar a identificar y detener actividades ilegales, como el tráfico de drogas y el contrabando. La tecnología de AR puede proporcionar una experiencia inmersiva y mejorar la capacidad de los agentes de seguridad para combatir estas actividades, lo que podría salvar vidas y mejorar la calidad de vida de las personas.