Diseñando Experiencias Inmersivas con Realidad Aumentada (RA) para la conciencia ampliada en Operaciones Militares

Autor

Caminos Cynthia Elena DNI 23626224

Analista de Sistemas

Director

Ing. Acquesta Alejandro

Doctor en Ingeniería Informática

Buenos Aires, Argentina 2025

RESUMEN

Este proyecto se enfoca en el desarrollo e implementación de experiencias inmersivas mediante realidad aumentada (RA), utilizando la combinación de Unity y Vuforia. Su propósito principal es aplicar la RA en el ámbito militar, permitiendo al personal de la fuerza acceder a información crítica en tiempo real y mejorar su conciencia situacional durante las misiones operativas. El objetivo principal es diseñar una aplicación móvil (APK) basada en RA que integre entornos virtuales y reales.

La solución estará orientada a la detección y seguimiento de objetos y unidades, facilitando la toma de decisiones rápidas y precisas en el campo de operaciones.

Mediante el uso de Imagen Targets y Multi-Targets proporcionados por Vuforia, la aplicación ofrecerá al personal militar una visualización clara y eficiente de datos esenciales, como la ubicación de unidades, equipamiento y rutas tácticas, según las necesidades operativas.

En un contexto militar, la RA representa una herramienta clave para optimizar la eficiencia operativa y reducir riesgos. Permite a los soldados mejorar su coordinación y preparación mediante simulaciones tácticas y acceso instantáneo a información estratégica. Además, esta tecnología se alinea con el modelo de conciencia situacional de (Endsley, 1995), ayudando a percibir, comprender y proyectar eventos en tiempo real.

Las experiencias inmersivas tienen potencial de impacto también en otros sectores:

- En salud, pueden utilizarse para entrenar personal médico y apoyar la rehabilitación de pacientes.
- En educación, las simulaciones inmersivas facilitan la comprensión de conceptos complejos.
- En turismo, enriquecen la experiencia del visitante con información contextual sobre puntos de interés.

Este proyecto busca demostrar que la combinación de RA, Unity y Vuforia es una estrategia efectiva para desarrollar soluciones inmersivas y adaptables a diversas necesidades operativas. Más allá de la defensa, la RA promete revolucionar la forma en que interactuamos con el entorno, mejorando procesos de aprendizaje, toma de decisiones y experiencias de usuario en múltiples sectores.

Palabras claves: Realidad Aumentada, Unity, Vuforia, Conciencia Situacional

ABSTRACT

This project focuses on the development and implementation of immersive experiences through Augmented Reality (AR), utilizing the combination of Unity and Vuforia. Its primary objective is to apply AR in the military domain, enabling personnel to access critical real-time information and enhance situational awareness during operational missions. The main goal is to design a mobile application (APK) based on AR that integrates virtual and real environments.

The proposed solution is oriented towards the detection and tracking of objects and units, facilitating rapid and accurate decision-making in the field. Through the use of Image Targets and Multi-Targets provided by Vuforia, the application will offer military personnel a clear and efficient visualization of essential data, including unit locations, equipment, and tactical routes, according to operational needs.

In a military context, AR serves as a key tool for optimizing operational efficiency and reducing risks. It enhances coordination and preparedness among soldiers through tactical simulations and instant access to strategic information. Furthermore, this technology aligns with Endsley's (1995) situational awareness model, aiding in the perception, comprehension, and projection of real-time events.

Immersive experiences also have significant potential in other sectors:

- In healthcare, AR can be used to train medical personnel and support patient rehabilitation.
- In education, immersive simulations facilitate the understanding of complex concepts.
- In tourism, AR enriches visitor experiences by providing contextual information about points of interest.

This project aims to demonstrate that the integration of AR, Unity, and Vuforia is an effective strategy for developing immersive and adaptable solutions for various operational needs. Beyond defense, AR has the potential to revolutionize how we interact with the environment, enhancing learning processes, decision-making, and user experiences across multiple sectors.

Keywords: Augmented Reality, Unity, Vuforia, Situational Awareness

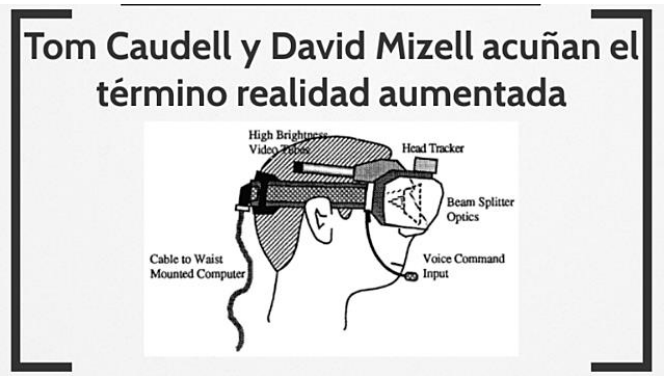
Tabla de contenido

MARCO TEORICO

Este proyecto nace del interés por explorar cómo la realidad aumentada (RA) puede mejorar tanto la preparación como la ejecución de tareas complejas en operaciones militares. La combinación de tecnología y entrenamiento táctico es esencial en un mundo donde la rapidez en la toma de decisiones puede marcar la diferencia entre el éxito y el fracaso. En esta sección, se detallan los fundamentos teóricos que sustentan el desarrollo de aplicaciones inmersivas mediante Unity y Vuforia, vinculándolos con las necesidades actuales del entorno militar. Este marco teórico no solo establece los fundamentos conceptuales del proyecto, sino que también refleja el proceso de aprendizaje que ha acompañado su desarrollo. Al combinar los avances en RA con las necesidades tácticas, se pretende contribuir al diseño de soluciones innovadoras que integren tecnología y estrategia de manera efectiva.

REALIDAD AUMENTADA

(Caudell, T. P., & Mizell, D. W., 1992, págs. 659–669) crean el termino Realidad Aumentada para describir una pantalla que usarían los técnicos electricistas de Boeing que mezclaban gráficos virtuales con la realidad física.



Augmented reality: An application of heads-up display technology to manual manufacturing processes Caudell, T. P., & Mizell, D. W. (1992)

(Azuma Ronald T, 1997)define la Realidad Aumentada (RA) como una tecnología que permite combinar elementos virtuales con el entorno físico en tiempo real. A diferencia de la Realidad Virtual (RV), que sumerge al usuario completamente en un entorno digital, la RA agrega elementos digitales al mundo real sin reemplazarlo. Es decir, en la RA, el mundo real sigue siendo la base, pero con la superposición de información o gráficos generados por computadora.

Azuma Ronald T, 1997 destaca tres características principales que definen la Realidad Aumentada:

- Combina el mundo real con el mundo virtual**:** La RA integra objetos virtuales (como imágenes, sonidos o texto) con la vista del entorno real del usuario.
- Interactúa en tiempo real: Los elementos virtuales se ajustan o se adaptan dinámicamente a las interacciones del usuario con el entorno real.
- Se percibe en 3D: Los objetos virtuales deben parecer parte del espacio tridimensional real, lo que requiere técnicas avanzadas de visualización y seguimiento de posición.

Componentes técnicos de la realidad aumentada (RA)

El artículo también analiza los componentes técnicos esenciales para implementar la Realidad Aumentada:

- Seguimiento o Tracking: Es uno de los principales desafíos, ya que la RA necesita rastrear la posición y orientación del usuario o del dispositivo (por ejemplo, mediante cámaras o sensores de movimiento) en el entorno real. Los métodos de seguimiento incluyen el uso de marcas visuales, cámaras, y sistemas de localización como el *SLAM* (Simultaneous Localization and Mapping).
- Visualización: La RA requiere gráficos en 3D que se superpongan con la escena real, lo que demanda un sistema de visualización eficaz. Esto incluye el diseño de interfaces y la representación gráfica realista de objetos virtuales, considerando iluminación, sombras y perspectiva.

Áreas de aplicación de la RA

En su artículo (Azuma Ronald T, 1997, págs. 355-385) identifica diversas aplicaciones potenciales de la Realidad Aumentada (RA) en distintos campos:

- Entrenamiento y simulación: Para entrenar a pilotos, soldados, cirujanos, entre otros, proporcionando simulaciones inmersivas en un entorno controlado.
- Entretenimiento: Juegos interactivos que combinan el mundo real con el virtual (por ejemplo, *Pokémon Go* puede considerarse un ejemplo moderno de esta categoría).
- Mantenimiento y reparación: Los trabajadores pueden recibir instrucciones superpuestas sobre el equipo o maquinaria en tiempo real, mejorando la eficiencia.
- Medicina: Visualización de datos médicos en tiempo real, lo que ayuda a los cirujanos durante los procedimientos.

Desafíos técnicos

Ronald T. Azuma y, más de una década después, (Zhou, F., Duh, H. B.-L., & Billinghurst, M., 2008, págs. 193-202), analizaron los desafíos técnicos que enfrentaba la RA en sus respectivas épocas, muchos de los cuales continúan siendo relevantes en la actualidad:

- Precisión y rapidez en el seguimiento: El *tracking* de alta precisión es crucial para que los elementos virtuales se alineen correctamente con el mundo real. Azuma (1997) señaló que las tecnologías de seguimiento de su tiempo eran limitadas en cuanto a precisión y desempeño. Posteriormente, Zhou et al. (2008) revisaron los avances en *tracking*, destacando la evolución de

algoritmos más eficientes y sensores mejorados, aunque el desafío de lograr una latencia mínima sigue siendo una preocupación actual.

- Generación de gráficos realistas: Aunque los gráficos 3D habían avanzado en la década de 1990, representar objetos virtuales de forma coherente en un entorno tridimensional real seguía siendo un reto. Azuma identificó la necesidad de mejorar el manejo de iluminación y sombras para una mejor integración visual. Zhou et al. (2008) documentaron cómo los avances en procesamiento gráfico y *rendering* en tiempo real han permitido una mayor fidelidad visual, aunque persisten retos en iluminación dinámica y oclusión entre objetos reales y virtuales.
- Limitaciones del hardware: En los años 90, los dispositivos de RA, como los visores, eran voluminosos, costosos y limitados en poder de procesamiento. Zhou et al. (2008) revisaron cómo la miniaturización y el aumento de la capacidad de cómputo han permitido la expansión de la RA en dispositivos móviles y portátiles, aunque los desafíos en autonomía de batería y rendimiento siguen siendo relevantes.
- Interacción en tiempo real: La interacción entre el usuario y los objetos virtuales debe ser fluida y natural. Azuma planteó la necesidad de desarrollar interfaces intuitivas, un tema que Zhou y sus colegas exploraron a fondo en su revisión de sistemas de interacción basados en gestos, *hand tracking* y dispositivos hápticos, evidenciando una evolución significativa en la usabilidad de la RA.

El artículo de Ronald T. Azuma fue un punto de partida fundamental para el estudio de la realidad aumentada, proporcionando un análisis detallado de sus elementos técnicos, aplicaciones y desafíos. Más de una década después, Zhou et al. (2008) continuaron esta línea de investigación, evaluando los avances logrados en áreas clave y proponiendo direcciones futuras. Aunque la tecnología de RA ha evolucionado considerablemente, muchos de los problemas que ambos estudios identificaron siguen siendo esenciales en el desarrollo actual de esta tecnología.

Azuma Ronald T. también analiza los desafíos técnicos que enfrentaba la realidad aumentada en 1997, muchos de los cuales continúan siendo relevantes en la actualidad:

Precisión y rapidez en el seguimiento: El tracking de alta precisión es crucial para que los elementos virtuales se alineen correctamente con el mundo real. Las tecnologías de seguimiento disponibles en esa época aún eran limitadas en cuanto a precisión y desempeño.

Generación de gráficos realistas: Aunque los gráficos 3D habían avanzado, representar objetos virtuales de forma realista y coherente en un entorno tridimensional real sigue siendo un desafío. Esto involucra el manejo de iluminación, sombras y otras características físicas.

Limitaciones del hardware: En la década de 1990, los dispositivos de RA, como los visores, eran voluminosos, costosos y limitados en cuanto a poder de procesamiento. A medida que la tecnología ha avanzado, estos problemas han disminuido, pero siguen siendo relevantes en dispositivos móviles y portátiles.

Interacción en tiempo real: La interacción entre el usuario y los objetos virtuales debe ser fluida y natural, lo cual requiere el desarrollo de interfaces intuitivas y tecnologías que faciliten la manipulación de objetos en entornos tridimensionales.

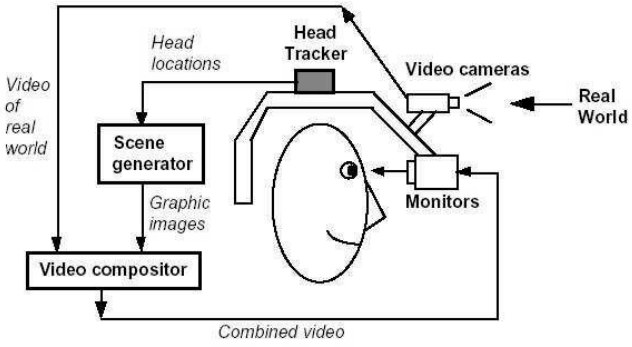
El artículo de Ronald T Azuma fue un punto de partida fundamental para el estudio y la investigación de la realidad aumentada, presenta un análisis detallado de los elementos técnicos, aplicaciones y desafíos asociados con la RA, al tiempo que identifica áreas clave para la investigación futura. Aunque la tecnología de RA ha avanzado considerablemente desde su publicación, muchas de las ideas y problemas que Azuma discute siguen siendo esenciales en el desarrollo de la RA moderna.

En este proyecto se analiza a la RA para operaciones militares en situaciones de combate y entrenamiento, ofreciendo ventajas como:

Conciencia situacional mejorada: Los soldados pueden visualizar la posición de aliados, amenazas y rutas estratégicas directamente en su visor.

Optimización del entrenamiento táctico: Las simulaciones inmersivas permiten a los soldados practicar maniobras en escenarios virtuales sin riesgo.

Toma de decisiones rápida y precisa: La información crítica se presenta en tiempo real, ayudando a reducir errores en situaciones de alta presión.



Survey of Augmented Reality, Ronald T. (Azuma 1997)

CONCIENCIA SITUACIONAL

La conciencia situacional es un concepto central en este proyecto. Se refiere a la capacidad de percibir, comprender y proyectar eventos en el entorno para tomar decisiones acertadas en tiempo real. Basado en el modelo de (Endsley, 1995, págs. 32-64)busco demostrar cómo la RA puede aumentar la conciencia situacional al proporcionar información visual relevante, como mapas, posiciones tácticas y estados del personal militar, integrándola de manera fluida para que el soldado mantenga su enfoque total en la misión.

Aplicar este concepto a través de la RA es particularmente relevante en entrenamientos tácticos y misiones reales. La idea es que un soldado equipado con un casco o dispositivo móvil con RA pueda ver más allá de lo evidente: rutas ocultas, personal militar cercano y amenazas emergentes, todo proyectado directamente en su visor. Esta tecnología reduce el margen de error, haciendo que cada decisión sea más rápida y precisa.

En su artículo titulado *"Toward a Theory of Situation Awareness in Dynamic Systems"* (1995), Mica R. Endsley presenta un modelo teórico de la conciencia situacional (SA). Este modelo describe la habilidad del ser humano para percibir información relevante del entorno, interpretarla en su contexto y anticipar su evolución futura, con el objetivo de tomar decisiones informadas y apropiadas.

El modelo se centra en tres niveles o etapas fundamentales de la conciencia situacional:

Percepción (Nivel 1 de la conciencia situacional):

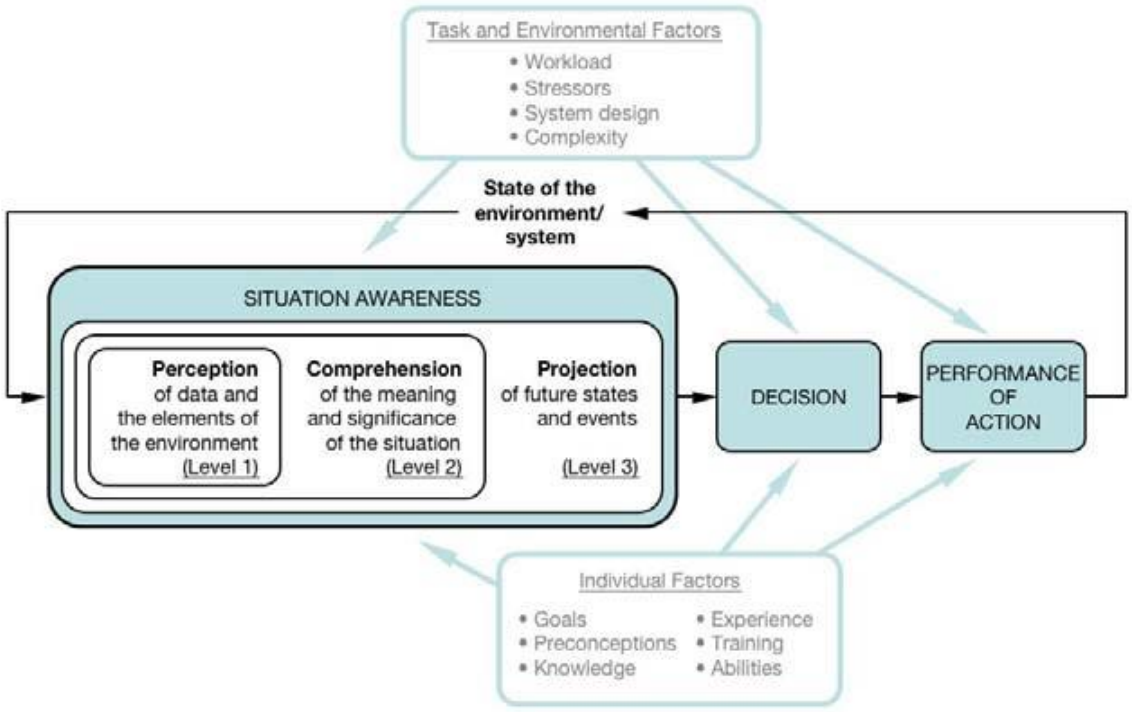
En este primer nivel, el individuo percibe o recoge información sobre los elementos del entorno y las variables relevantes que afectan la situación. La percepción involucra la recopilación de datos de diferentes fuentes sensoriales, como la vista, el oído y otros sensores. Este nivel se refiere a la capacidad de un individuo para identificar y registrar objetos, eventos o condiciones que son críticos para comprender el entorno.

Comprensión (Nivel 2 de la conciencia situacional):

Una vez que se percibe la información, el siguiente paso es la comprensión de esa información dentro del contexto actual. Este nivel implica la interpretación y el análisis de la información percibida para formar una representación coherente y significativa de la situación. La comprensión requiere que la persona tenga conocimiento previo y habilidades cognitivas para interpretar correctamente los datos. Esto incluye comprender la dinámica del sistema, las interacciones entre elementos y los posibles riesgos o oportunidades en el entorno.

Proyección (Nivel 3 de la conciencia situacional):

El tercer nivel se refiere a la capacidad de anticipar o proyectar lo que sucederá en el futuro, basándose en la información percibida y comprendida. Implica hacer predicciones sobre la evolución de la situación y planificar acciones en consecuencia. Este nivel es crucial para la toma de decisiones, ya que permite al individuo anticipar eventos y preparar respuestas apropiadas antes de que ocurran.



Situation Awareness Model (Endsley 1995)

Relación con el desempeño y la toma de decisiones

Mica R. Endsley sostiene que la percepción situacional es fundamental, ya que su ausencia puede derivar en errores críticos en la toma de decisiones o incluso en accidentes graves. Por el contrario, cuando esta habilidad se desarrolla adecuadamente, permite al individuo tomar decisiones rápidas, precisas y efectivas, especialmente en contextos donde el tiempo es crítico. Además, facilita la capacidad de anticiparse a los posibles problemas y ajustar su comportamiento y acciones de acuerdo con las demandas dinámicas del entorno, como:

Aplicaciones del modelo

- Aviación y Tráfico Aéreo:
- Coches Autónomos:
- Realidad Aumentada (RA)

El modelo de conciencia situacional es un marco fundamental para entender cómo las personas perciben, comprenden y responden a su entorno en situaciones complejas y dinámicas. Sus aplicaciones son vastas y se extienden a muchos campos, desde la aviación hasta la RA. El modelo proporciona un enfoque estructurado para abordar la toma de decisiones en situaciones de alta complejidad y sirve como base para el diseño de sistemas que mejoren la interacción humana con la tecnología en entornos de alta exigencia.

UNITY y VUFORIA: Creando soluciones eficaces

Unity es un motor de desarrollo ampliamente utilizado para la creación de aplicaciones interactivas y videojuegos en 2D, 3D y realidad aumentada (RA). Esta plataforma se ha consolidado como un estándar en el desarrollo de soluciones para entornos inmersivos, tanto en realidad virtual, donde los entornos son completamente digitales, como en realidad aumentada, que combina elementos digitales con el mundo real de manera interactiva. Su popularidad se debe a su facilidad de uso, su capacidad para trabajar de manera eficiente con dispositivos móviles y su integración con tecnologías avanzadas, como Vuforia, lo que la convierte en una herramienta ideal para proyectos innovadores y versátiles.En el ámbito militar, Unity permite desarrollar aplicaciones que abarcan:

- **Simulaciones de combate**: Los soldados pueden entrenar en escenarios virtuales que replican condiciones de combate reales.
- **Visualización en tiempo real del entorno operativo**: Los dispositivos móviles o cascos de RA muestran información estratégica durante las misiones.
- **Entrenamiento adaptativo**: La flexibilidad de Unity permite crear experiencias personalizadas según las necesidades de cada unidad militar.

Por otro lado, **Vuforia** ofrece el componente de reconocimiento y seguimiento, esencial para que los soldados puedan identificar objetos y personas en tiempo real. Los Image Targets y Multi-Targets de Vuforia facilitan la integración de marcadores físicos, como insignias o equipos, que activan datos relevantes al instante. Esta tecnología no solo ayuda en la identificación rápida, sino que también mejora la coordinación operativa en escenarios dinámicos

Vuforia es un “Software Development Kit" o "Kit de Desarrollo de Software (SDK)” especializado en realidad aumentada, ampliamente utilizado para reconocimiento de objetos y marcadores. Esta plataforma permite que las aplicaciones identifiquen y sigan elementos físicos en tiempo real, facilitando la superposición de información relevante en la pantalla del usuario.

Funcionalidades claves de Vuforia:

- **Image Targets**: Permiten reconocer imágenes específicas para activar contenido digital.
- **Multi-Targets**: Usados para detectar objetos compuestos por varias superficies, útiles en simulaciones tácticas complejas.
- **Object Recognition**: Facilita la identificación rápida de equipamiento o vehículos en operaciones militares.

Es ideal para desarrollar aplicaciones en entornos militares, donde la identificación rápida de personas, equipos y ubicaciones estratégicas es esencial para el éxito de las misiones.

La combinación de Unity y Vuforia permite crear experiencias inmersivas brindando ventajas operativas y formativas. La integración de estas herramientas incluye los siguientes procesos:

Configuración del entorno de desarrollo: Instalación de Unity y Vuforia, y creación del proyecto.

Implementación de targets: Uso de Image Targets y Multi-Targets para reconocimiento en misiones.

Simulación y pruebas: Validación del rendimiento en entornos simulados y ajustes según los resultados.

Despliegue en dispositivos móviles: Uso de tablets o gafas de RA para facilitar el acceso a la información en campo.

Aprendizajes y oportunidades más allá del uso militar

Si bien el desarrollo se centra en las operaciones militares, el aprendizaje obtenido puede trasladarse a otros campos. En educación, las simulaciones inmersivas pueden transformar la forma en que los estudiantes aprenden conceptos complejos. En salud, esta tecnología permite la visualización precisa de procedimientos médicos y el entrenamiento de profesionales en entornos seguros. Incluso en turismo, la RA enriquece la experiencia del visitante al proporcionar información contextual en tiempo real. Cada uno de estos sectores demuestra que la RA es adaptable y ofrece beneficios que trascienden su aplicación original.

A través de este proyecto, busco explorar cómo estas experiencias inmersivas pueden mejorar procesos operativos en diferentes contextos, manteniendo siempre el foco en optimizar la seguridad y la toma de decisiones. La combinación de RA, Unity y Vuforia ofrece una solución innovadora para aumentar la conciencia situacional y mejorar los resultados operativos en misiones militares. Sin embargo, la verdadera fortaleza de esta tecnología radica en su capacidad para adaptarse a diversos sectores y proporcionar experiencias inmersivas que conectan la teoría con la práctica.

INTRODUCCION

Durante los últimos años, el uso de tecnologías de la información y la comunicación (TIC) ha transformado profundamente la gestión y toma de decisiones en diversos sectores, incluyendo las fuerzas de seguridad. Estas herramientas han optimizado la eficiencia operativa y mejorado tanto la seguridad del personal de la fuerza como la calidad de vida de las personas en contextos críticos. Entre las tecnologías más prometedoras destaca la realidad aumentada (RA), que permite superponer información digital sobre la realidad mediante dispositivos móviles o cascos especializados, proporcionando una experiencia enriquecida y contextual en tiempo real.

Este proyecto de investigación se centra en la intervención y desarrollo de experiencias inmersivas utilizando RA a través de la plataforma Unity. La RA tiene el potencial de cambiar la forma en que interactuamos con el entorno, facilitando el reconocimiento, detección e interacción de objetos y personas en diversos escenarios.

Aunque el enfoque principal de su aplicación está orientado al personal de la fuerza, en el ámbito de la salud, la RA ha demostrado ser eficaz para mejorar la formación médica y apoyar en procesos de rehabilitación, proporcionando entornos interactivos.

Unity permite crear espacios de aprendizaje dinámicos y atractivos, que fomentan una mayor interacción del usuario. En las fuerzas de seguridad y en la seguridad pública, la RA potencia los entrenamientos tácticos y facilita la toma de decisiones en tiempo real.

El proyecto plantea el desarrollo de una aplicación móvil (APK) basada en RA, que permitirá al personal de la fuerza moverse entre entornos virtuales y reales, accediendo a información crítica en tiempo real durante sus tareas. Asimismo, la aplicación funcionará como una herramienta de entrenamiento inmersivo, conectando la teoría con la práctica en simulaciones controladas.

La integración de esta tecnología busca mejorar la preparación operativa y la seguridad del personal de la fuerza, promoviendo nuevas formas de aprendizaje alineadas con los desafíos actuales.

La combinación de RA con Unity representa una estrategia eficaz para optimizar resultados en múltiples áreas. Futuras investigaciones podrían explorar formas de mejorar la accesibilidad y usabilidad de las aplicaciones, optimizar la tecnología para dispositivos móviles y evaluar el impacto a largo plazo de estas herramientas en escenarios reales. A medida que esta tecnología avance, podrá seguir ofreciendo soluciones innovadoras para sectores críticos como la seguridad y la defensa.

CAPITULO 1

1.1 Objetivo

Este capitulo tiene como objetivo detallar el proceso de configuración y uso de Unity en conjunto con el SDK de Vuforia para la implementación de aplicaciones de realidad aumentada.Se abordarán los pasos necesarios para configurar un proyecto en Unity, integrar el SDK de Vuforia, y utilizar ImageTarget para la detección y seguimiento de objetos en tiempo real.

Además, se presentarán los conceptos teóricos fundamentales relacionados con la realidad aumentada, y se discutirá su aplicación en el contexto de un proyecto en desarrollo. El documento hará énfasis en los aspectos técnicos y conceptuales que guían tanto el diseño como la ejecución del proyecto.

1.2 Alcance

El alcance incluye la instalación del software necesario, la configuración del entorno de desarrollo con Unity y Vuforia, así como ejemplos de código para la detección de objetos mediante realidad aumentada. También se cubrirá la visualización de los resultados en un dispositivo móvil o mediante simulación en el editor de Unity.

1.3 Documentos aplicables:

- Documentación oficial de Unity.

https://docs.unity3d.com

- Guía de instalación del SDK de Vuforia.

https://docs.unity3d.com/es/2018.4/Manual/vuforia-sdk-overview.html

- Manual de referencia de C# para scripts en Unity.

https://learn.unity.com/project/scripting-para-principiantes

1.4 Documentos de referencia:

- Tutoriales oficiales de Unity y Vuforia.
  https://www.youtube.com/@vuforia/playlists
- Ejemplos de la comunidad de desarrolladores de Unity y Vuforia.
  https://medium.com/@braulio-camarenah/realidad-aumentada-con-unity-y-vuforia-596266264597.
  https://unity.com/es/solutions/xr/ar.

1.5 Símbolos, abreviaturas y acrónimos

RA: Realidad Aumentada
SDK: Software Development Kit
Vuforia: Plataforma de desarrollo de RA para Unity
Unity: Motor de desarrollo de videojuegos y aplicaciones.
RV: Realidad Virtual
RM: Realidad Mixta
TIC: Tecnología de la información y las comunicaciones
UI: Interfaz de Usuario
iOS: Sistema Operativo Movil
APK: Archivo que contiene los datos de una aplicacion para Android

2. Requerimientos

Para el desarrollo de la aplicación de realidad aumentada usando Unity y Vuforia, se requieren los siguientes elementos:

2.1 Hardware:

- **Dispositivo móvil** (Android o iOS) con soporte para RA, como smartphones o tablets
- **Computadora** con capacidad para ejecutar Unity, con las siguientes especificaciones mínimas:
    - Procesador: Intel Core i5 o superior.
    - Memoria RAM: 8 GB (recomendado 16 GB o más).
    - Tarjeta gráfica: Compatible con DirectX 11.
    - Almacenamiento: 10 GB de espacio libre para Unity y SDKs adicionales.
    - Sistema operativo: Windows 10/11, macOS 10.13+ o distribuciones GNU/Linux compatibles.

2.2 Software:

- **Unity** (versión 2020.3 LTS o superior): Motor de desarrollo necesario para crear y gestionar el proyecto.
- **Vuforia Engine SDK**: Plataforma de RA integrada en Unity para el reconocimiento y seguimiento de objetos.
- **Visual Studio**: Entorno de desarrollo integrado (IDE) para escribir scripts en **C#**, requerido por Unity para funcionalidades personalizadas.
- **Vuforia Developer Account**: Registro en el portal de Vuforia para obtener licencias y acceso a las herramientas de desarrollo.
- **Android Studio o Xcode** (opcional): Requeridos para compilar y desplegar la aplicación en dispositivos Android o iOS, respectivamente.

2.3 Herramientas adicionales:

- **Conexión a Internet** estable: Necesaria para descargar paquetes, plugins y SDKs desde los portales oficiales.
- **Cámara de buena resolución**: Requerida para realizar pruebas de detección y seguimiento precisas en entornos reales.
- **Cable USB o conexión inalámbrica**: Para transferir la aplicación al dispositivo móvil desde la computadora.
- **Cuenta en Google Play Console / App Store** (opcional): En caso de que se busque publicar la aplicación en plataformas móviles.

2.4 Requerimientos adicionales recomendados:

- **Repositorio GitHub o similar**: Para gestionar el control de versiones del proyecto y colaborar en equipo.
- **Dispositivo de RA especializado** (opcional): Como gafas AR (Hololens o similares), para experimentar funcionalidades avanzadas.
- **Baterías adicionales o cargadores portátiles**: Para extender las pruebas en campo sin interrupciones.

CAPITULO 2

Conceptos teóricos de Unity y Vuforia

En este capítulo se desarrollará un marco teórico los conceptos de la Realidad Aumentada mediante el entorno de desarrollo de Unity.

En los últimos años, la realidad aumentada (RA) ha comenzado a transformar las operaciones militares al permitir la superposición de información digital sobre el entorno físico, brindando nuevas posibilidades para mejorar la conciencia situacional y la capacidad de respuesta en misiones tácticas. En este contexto, **Unity** y **Vuforia** se presentan como herramientas clave para el desarrollo de aplicaciones inmersivas que integran elementos virtuales y reales de manera eficaz.

1.Introducción a Unity y Vuforia

Unity es un motor de juegos multiplataformas que permite el desarrollo de aplicaciones interactivas en diversas plataformas como Windows, macOS y GNU/Linux. Esta herramienta ofrece un entorno amigable, donde el proceso de creación es intuitivo, con un enfoque basado en el arrastrar y soltar, simplificando el desarrollo de aplicaciones de Realidad Aumentada (RA) y Realidad Mixta (RM) para dispositivos Android e iOS.

En este contexto, Unity permite el diseño de simulaciones tácticas inmersivas, proporcionando entornos virtuales donde los soldados pueden practicar tácticas y habilidades en escenarios controlados. Estos ejercicios ayudan a anticipar situaciones críticas, reforzando la preparación y coordinación operativa. La capacidad de Unity para combinar elementos 2D, 3D y RA facilita la creación de aplicaciones militares que maximizan la eficiencia y la seguridad del personal en misiones reales.

Además, las aplicaciones desarrolladas con Unity proporcionan:

- Visualización en tiempo real del entorno operativo
- Entrenamiento inmersivo
- Reducción de riesgos

Vuforia complementa las capacidades de Unity al proporcionar herramientas para el reconocimiento de objetos y marcadores en tiempo real.

Gracias a su compatibilidad con dispositivos móviles y gafas de RA, Vuforia ofrece una solución práctica para desplegar información crítica directamente en el campo de visión del soldado. Por ejemplo, un marcador 2D en el casco de un soldado puede proporcionar datos instantáneos sobre su identidad, unidad y estado operativo, mejorando la toma de decisiones bajo presión.

La combinación de Unity y Vuforia permite desarrollar soluciones avanzadas que mejoran tanto el entrenamiento como la ejecución de operaciones en tiempo real.

Esta integración se realiza en varias fases:

1. **Diseño del entorno en Unity**: Se crean escenarios 3D interactivos que simulan las condiciones de las misiones.
2. **Implementación de Vuforia para el reconocimiento**: Se configuran marcadores y objetos para que los soldados puedan ser identificados rápidamente en el entorno.
3. **Pruebas en simulaciones**: Se realizan pruebas en entornos controlados para verificar que el sistema funcione correctamente en distintas condiciones.

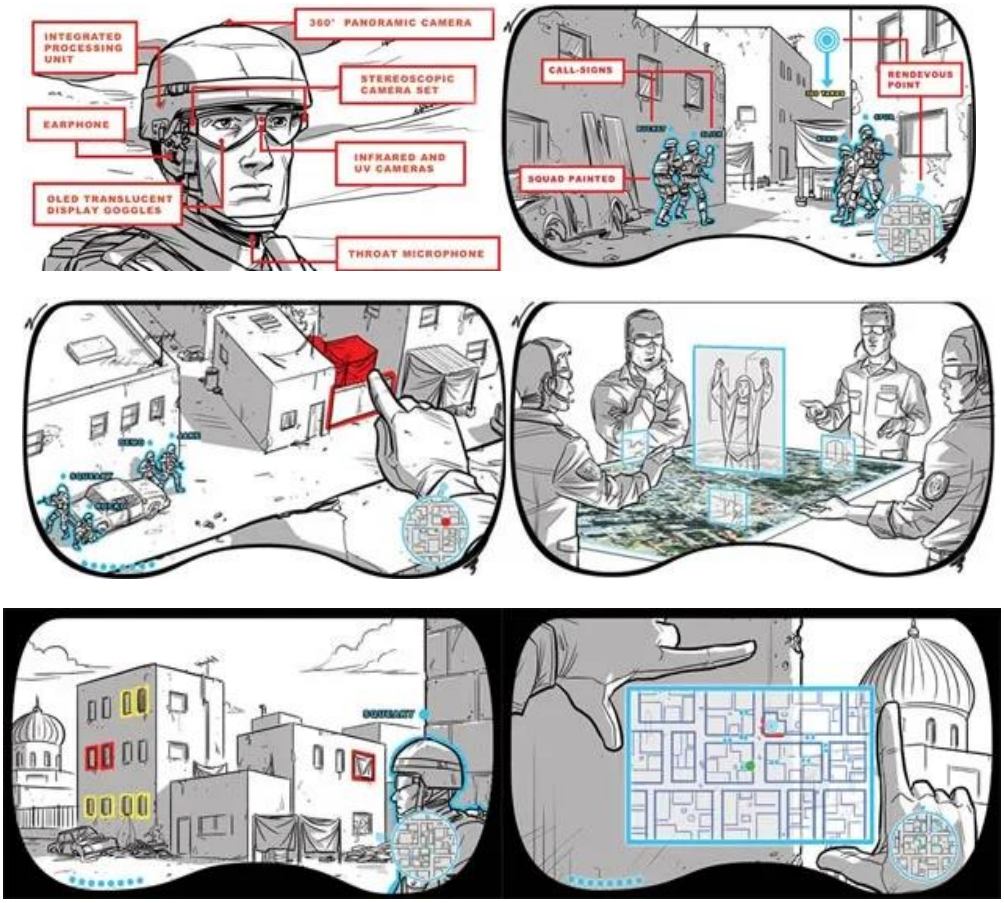
Gracias a esta combinación, los soldados no solo acceden a información relevante en tiempo real, sino que también pueden responder de manera más rápida y coordinada, aumentando las probabilidades de éxito en cada misión.

Aplicación en operaciones militares

Estas tecnologías proporcionan una ventaja estratégica al optimizar tanto la coordinación entre unidades como la respuesta inmediata ante situaciones imprevistas.

Beneficios en el campo de batalla:

- **Mayor precisión operativa:** La información se presenta al instante, minimizando errores y optimizando recursos.
- **Toma de decisiones rápida y segura:** Los soldados tienen acceso inmediato a datos relevantes sin necesidad de informes largos o comunicaciones extensas.
- **Adaptación a múltiples contextos:** Las aplicaciones se ajustan a los distintos escenarios de las misiones, facilitando su uso en entrenamientos y operaciones reales.



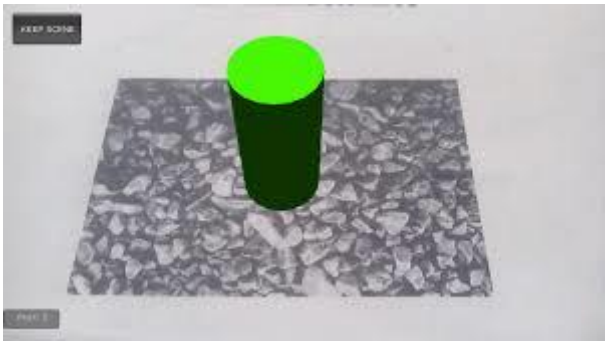
Concepto de RA ,Imagen extraida de la siguiente Fuente:
https://aumentada.net/realidad-aumentada-en-las-fuerzas-militares/

Tipos de Objetivos (Targets) en Vuforia

Vuforia ofrece una variedad de objetivos (targets) que permiten a los desarrolladores implementar diferentes tipos de interacciones de Realidad Aumentada según el tipo de objeto o superficie que desean reconocer. A continuación, se describen los diferentes tipos de objetivos que Vuforia soporta y sus características principales:

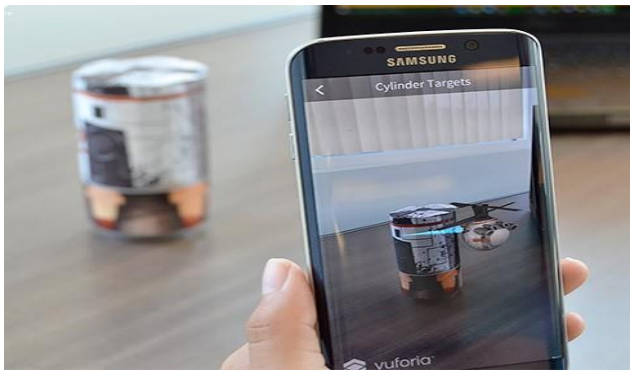
2.1. Objetivos de Imagen

Los **Image Targets** permiten reconocer y rastrear imágenes específicas en el entorno. Estas imágenes pueden ser cualquier tipo de gráfico como fotos, ilustraciones o logotipos. Una vez que la cámara detecta la imagen en el mundo real, el contenido digital se superpone en la posición y orientación de la imagen reconocida. Este tipo de objetivo es ideal para aplicaciones que buscan añadir contenido virtual a medios impresos, como revistas o pósters.



2.2. Objetivos Cilíndricos

Los **Cylindrical Targets** permiten reconocer y rastrear objetos que tienen una forma cilíndrica, como latas o botellas. Este tipo de objetivo es útil para aplicaciones de marketing y publicidad, donde se desea agregar contenido interactivo a productos con superficies cilíndricas.

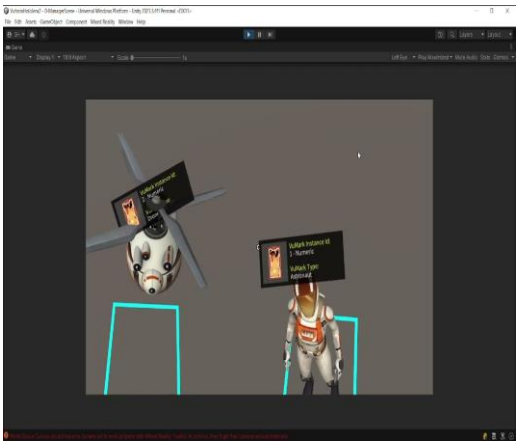


2.3. Objetivos Múltiples

Los **Multi-Targets** son conjuntos de múltiples objetivos de imagen que juntos forman una estructura tridimensional. Este tipo de objetivo permite que los desarrolladores creen experiencias más complejas, donde varios objetivos de imagen colaboran para generar una interacción en tres dimensiones. Un ejemplo común sería una caja o envase con diferentes imágenes en cada lado.

2.4. Objetivos VuMark

Los **VuMarks** son códigos personalizados que pueden contener datos variables como números de serie o información específica de un producto. Estos objetivos combinan la flexibilidad de un código QR con la capacidad de rastreo robusto de una imagen. VuMarks son útiles en aplicaciones donde se necesita rastrear múltiples objetos o identificar productos individualmente.



2.5. Escáner de Código de Barras

El **Barcode Scanner** permite la detección y lectura de códigos de barras dentro de la experiencia de Realidad Aumentada. Esta función es útil en aplicaciones de comercio electrónico o inventarios, donde se puede proporcionar información adicional o interactiva sobre un producto tras escanear su código.

2.6. Reconocimiento de Nubes

El **Cloud Recognition** permite que las aplicaciones RA accedan a una base de datos en la nube para reconocer imágenes almacenadas remotamente. Esto es ideal para aplicaciones que requieren rastrear una gran cantidad de imágenes sin tener que almacenar todas las imágenes localmente en el dispositivo.

2.7. Plano de Tierra

El **Ground Plane** permite rastrear superficies planas como pisos o mesas y colocar objetos virtuales sobre ellos. Esta función es clave para crear experiencias de Realidad Aumentada donde los objetos parecen estar físicamente presentes en el entorno del usuario, como la colocación de muebles en una habitación.

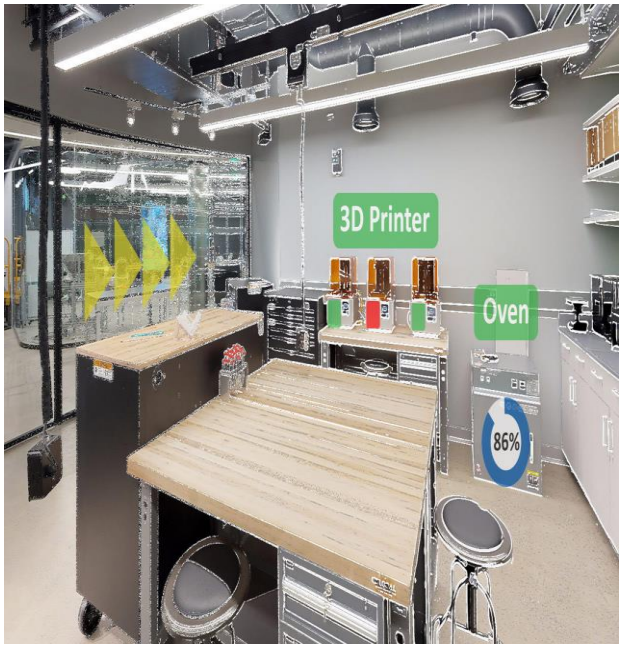
2.8. Objetivos del Modelo

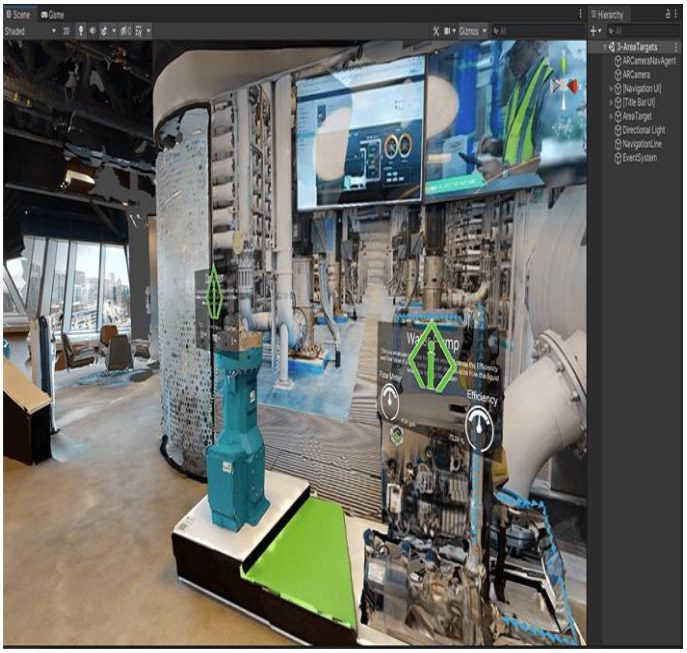
Los **Model Targets** permiten el reconocimiento y seguimiento de objetos tridimensionales físicos, como maquinaria o vehículos, en lugar de imágenes planas. Estos objetivos son especialmente útiles en aplicaciones industriales, donde los usuarios pueden interactuar con versiones digitales de objetos físicos complejos.

2.9. Objetivos del Área

Los **Area Targets** permiten el reconocimiento de entornos físicos completos, como habitaciones o edificios, a partir de un escaneo 3D del lugar. Este tipo de objetivo es ideal para crear experiencias de RA donde el contenido digital interactúa con todo el espacio físico en lugar de un objeto específico.

Esta diversidad de objetivos en Vuforia proporciona flexibilidad a los desarrolladores para implementar aplicaciones de RA en diferentes contextos, ajustándose a las necesidades de cada proyecto y permitiendo una interacción más rica con el entorno del usuario.





En resumen la integración de Vuforia dentro de Unity permite el uso de diversos elementos interactivos de RA como ImageTargets, que son imágenes físicas que Vuforia reconoce y sobre las cuales coloca contenido digital en la aplicación.

Conclusion:

- **Unity** es un motor de desarrollo de videojuegos y aplicaciones que permite la creación de experiencias 2D y 3D interactivas. Su versatilidad lo ha convertido en una de las plataformas más populares para el desarrollo de aplicaciones de realidad aumentada.
- **Vuforia** es una plataforma de realidad aumentada que se integra dentro de Unity y permite a los desarrolladores utilizar RA en sus aplicaciones mediante la detección de objetos y patrones en el mundo real a través de la cámara del dispositivo.

CAPITULO 3

El objetivo de este capítulo es llevar a cabo una práctica introductoria utilizando una de las principales herramientas de Vuforia para obtener un mejor entendimiento de su funcionamiento. En este caso, trabajare con el Image Target. Comenzare con la configuración inicial de Unity con Vuforia, seguido de la creación de un Image Target, para finalmente mostrar un objeto 3D proyectado en el mundo físico.

Configuración inicial

En esta sección se describirá y se incluira captura de pantalla que muestre los siguientes pasos:

- Creación de un nuevo proyecto en Unity.
- Integración del SDK de Vuforia en Unity.
- Configuración inicial de los **ImageTargets**, donde se define una imagen física que será detectada por Vuforia.

1.Creación de un nuevo proyecto en Unity

1) Abre Unity Hub y selecciona la opción **New Project**.
2) Elige una plantilla adecuada, como **3D Core** (para la mayoría de los proyectos RA).
3) Asigna un nombre al proyecto y selecciona la ubicación donde deseas guardarlo.
4) Haz clic en **Create Project**. Esto abrirá el editor de Unity con un nuevo proyecto en blanco.

Captura de pantalla de Unity Hub mostrando la creación de un nuevo proyecto con la Plantilla. **(Figura 3.1) (Figura 3.2).**

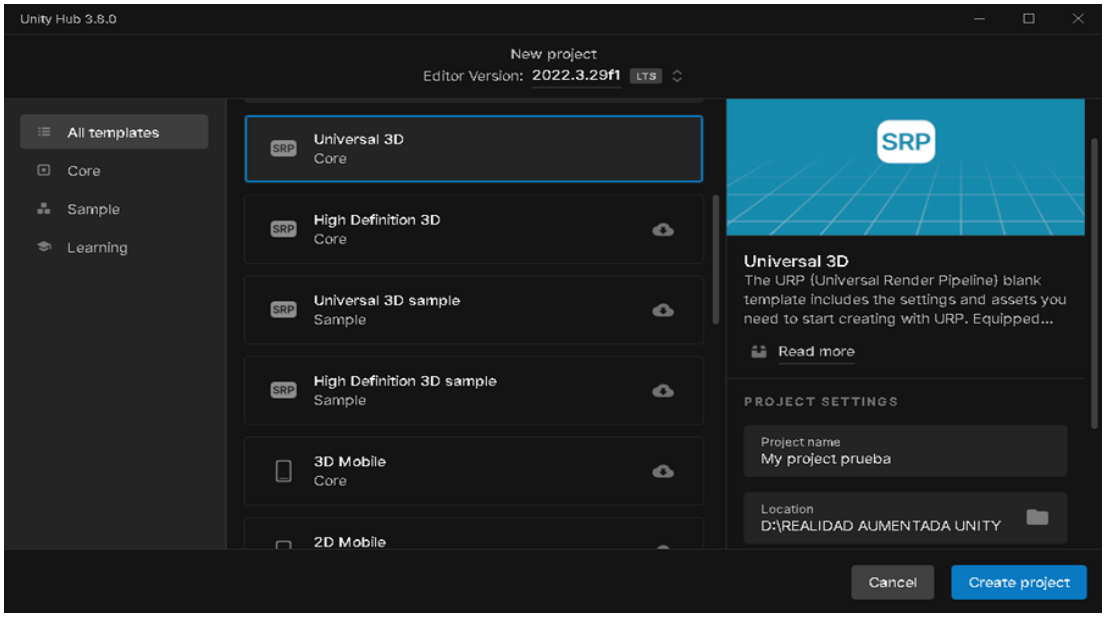


Figura 3.1

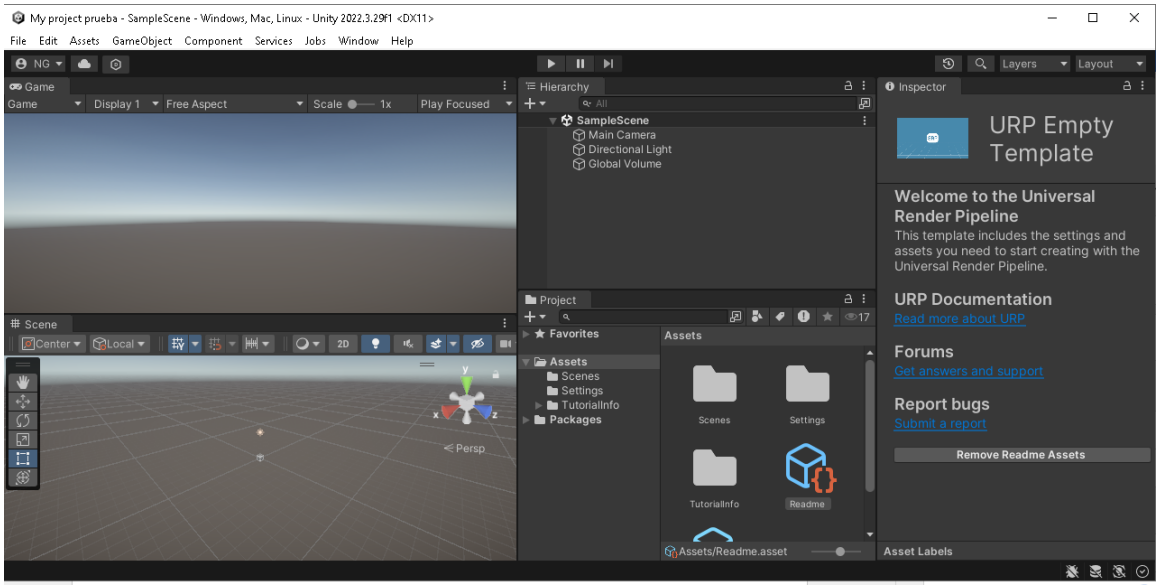


Figura 3.2

4. Integración del SDK de Vuforia en Unity

Una vez instalado Unity y creado el proyecto, es necesario descargar el SDK de VUFORIA para vincularlo. Para eso necesitamos primero crear una cuenta en vuforia para luego poder descargar el SDK.

1. Desde el siguiente link :Download Center SDK | Engine Developer Portal (vuforia.com). **(Figura 4.1).**

2. Una vez logueado puedes descargar el SDK desde el siguiente link Download Center SDK | Engine Developer Portal (vuforia.com) (**Figura 4.2).**

3. Finalizado la descarga se procede a Vincular Vuforia con Unity para eso solo tienes que arrastrar el paquete de Vuforia a la solapa **Project**.**(Figura 4.3)**.

4. Por Último le das click en **Install** para agregar el SDK de Vuforia a tu proyecto.

5. Una vez Integrado en Unity es necesario obtener una licencia de Vuforia para concluir con la Integración del SDK. Cree primero la licencia, dándole clic a “Get Basic” y luego se le da el nombre .Debe aceptar términos y condiciones y dar clic en la opción “Confirmar”.https://developer.vuforia.com/develop/licenses/free/new **(Figura 4.4).(Figura 4.5).**

6. Hago click al nombre de licencia que he generado y copiar la “Licence Key”. **(Figura 4.6).**

7. Ahora en Unity crear un objeto “ARCAMERA” **(Figura 4.7)**.

8. En “ARCAMERA” ir a la solapa “Inspector”, luego “OPEN VUFORIA CONFIGURATION” y pegar la “Licence Key”. ” **(Figura 4.8)(Figura 4.9)**.

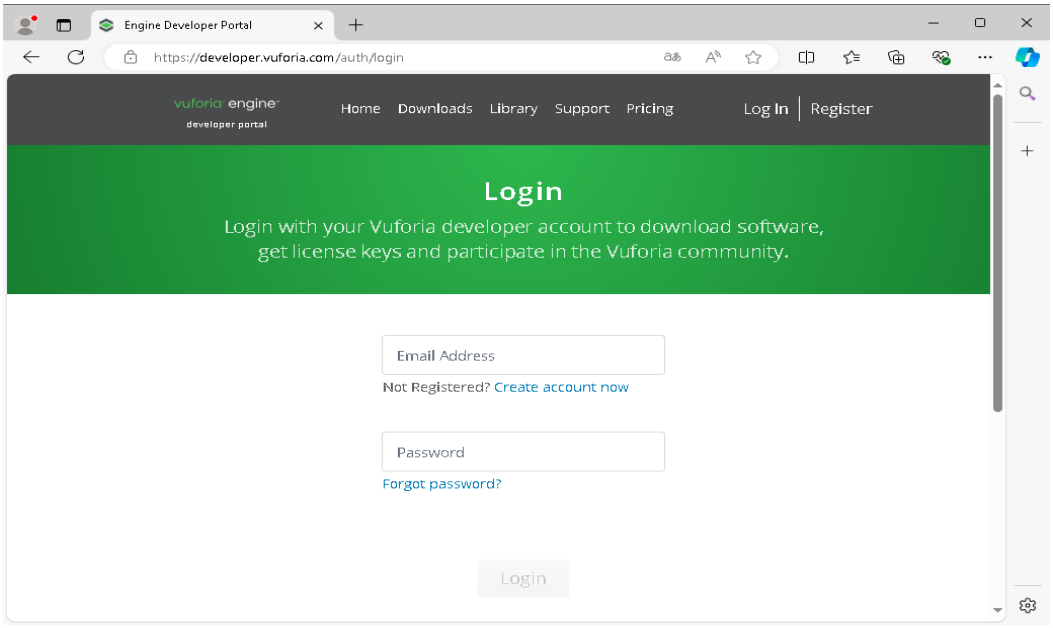


Figura 4.1

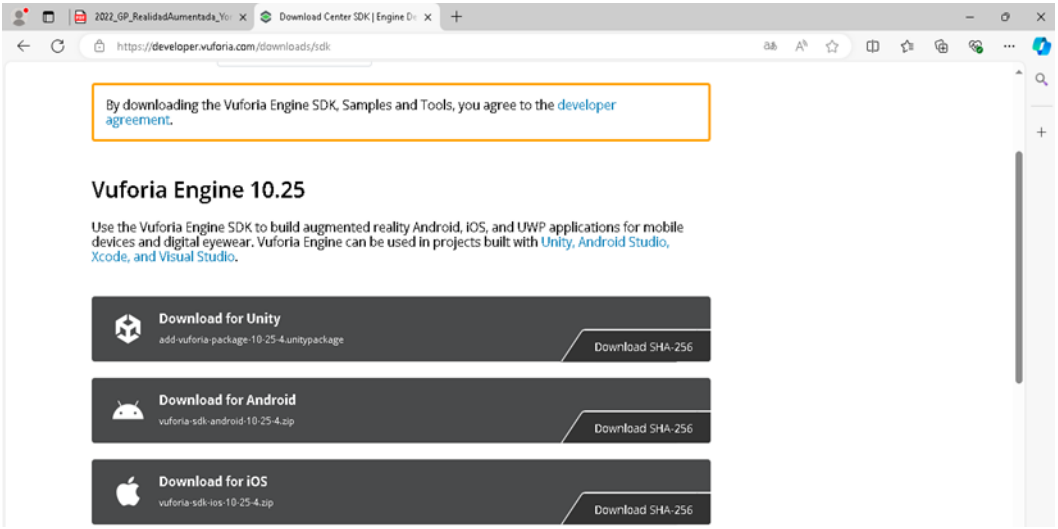


Figura 4.2

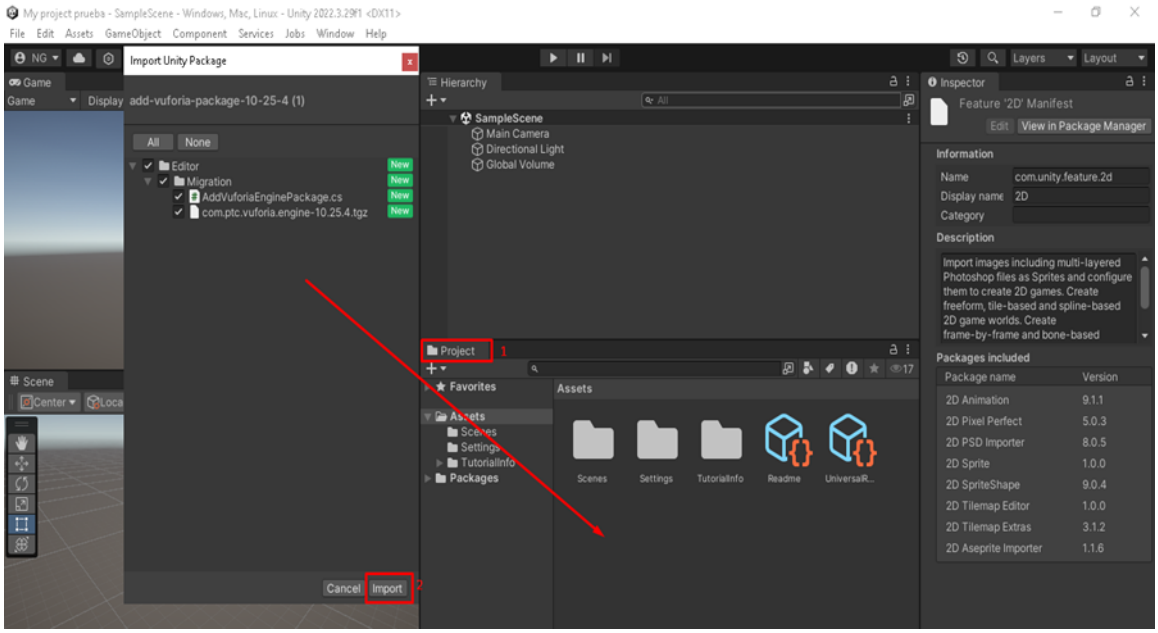


Figura 4.3

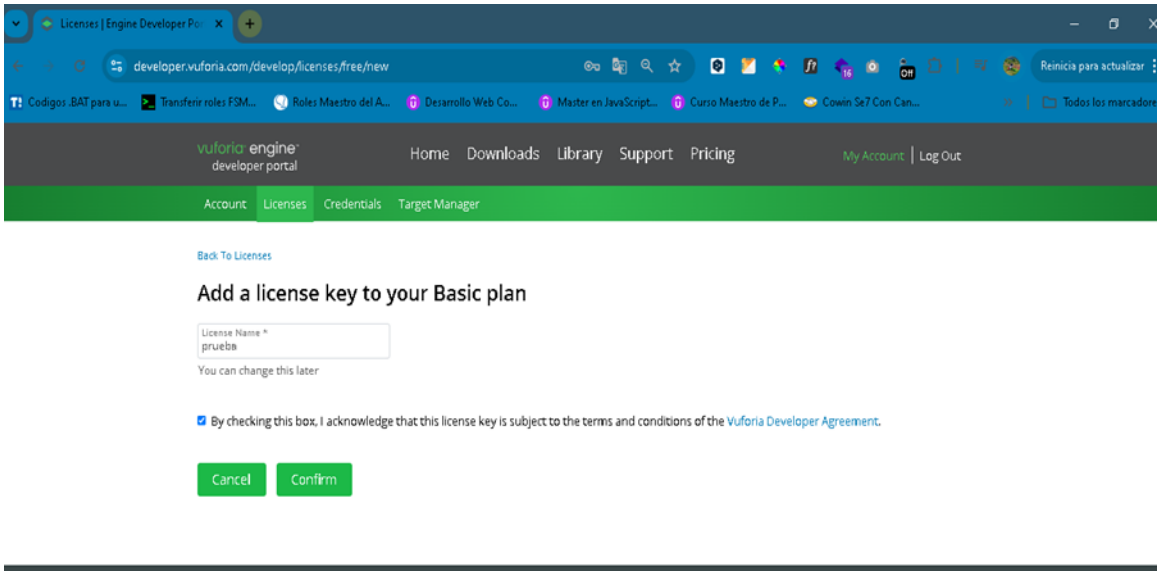


Figura 4.4

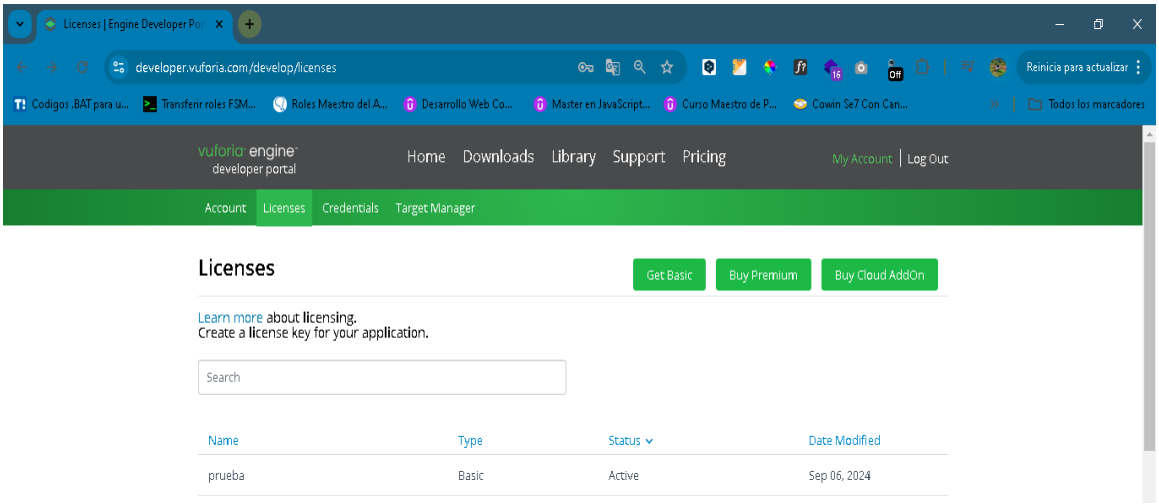


Figura 4.5

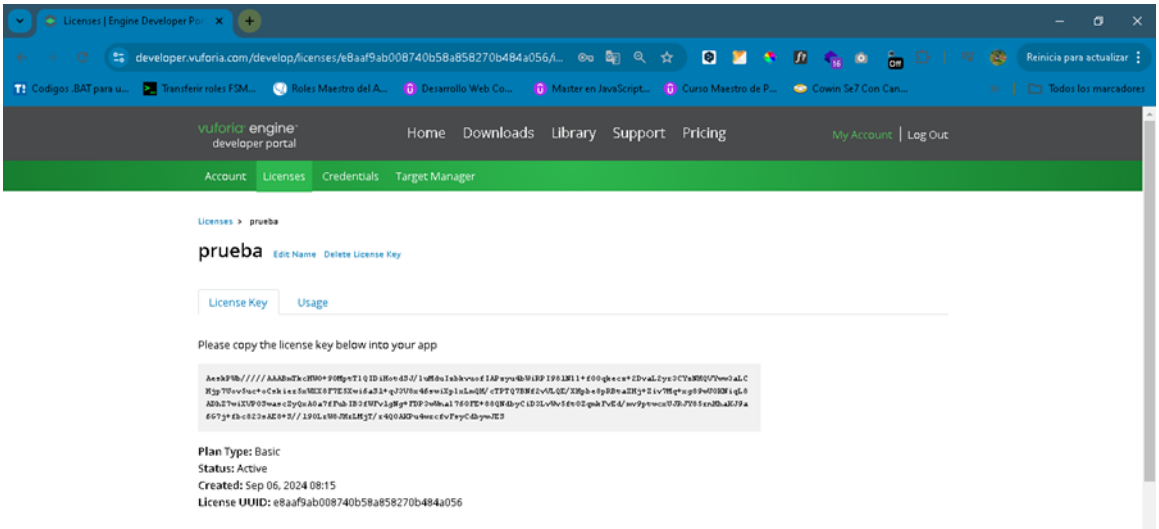


Figura 4.6

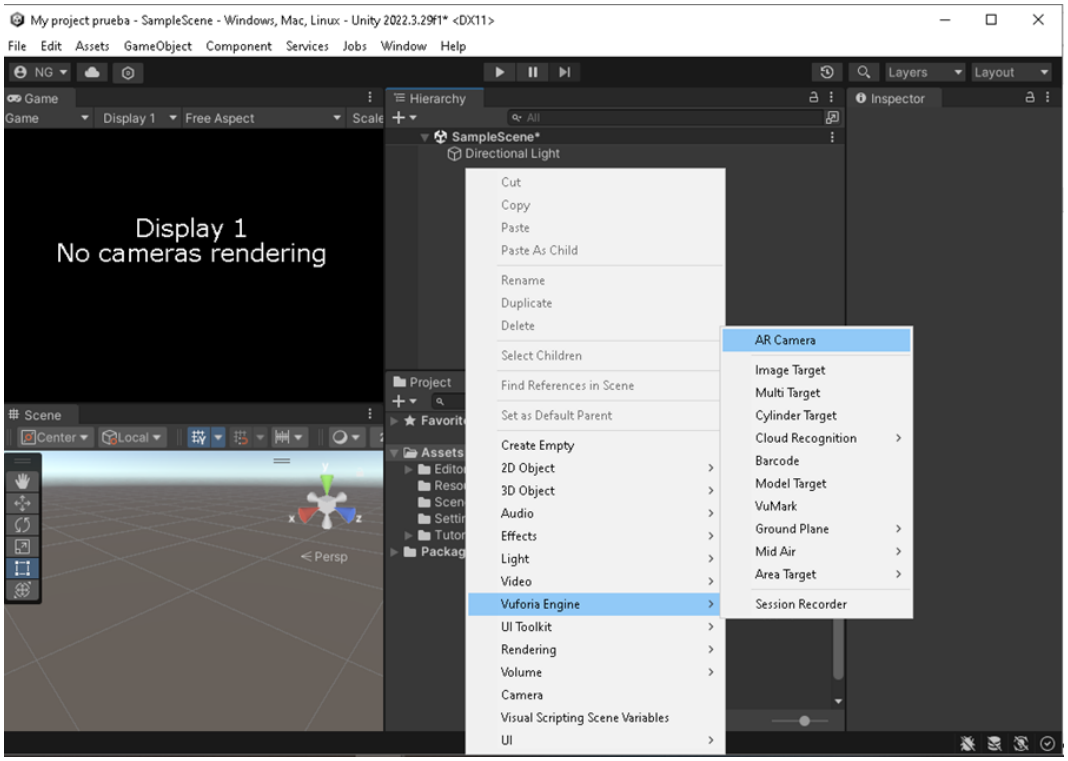


Figura 4.7

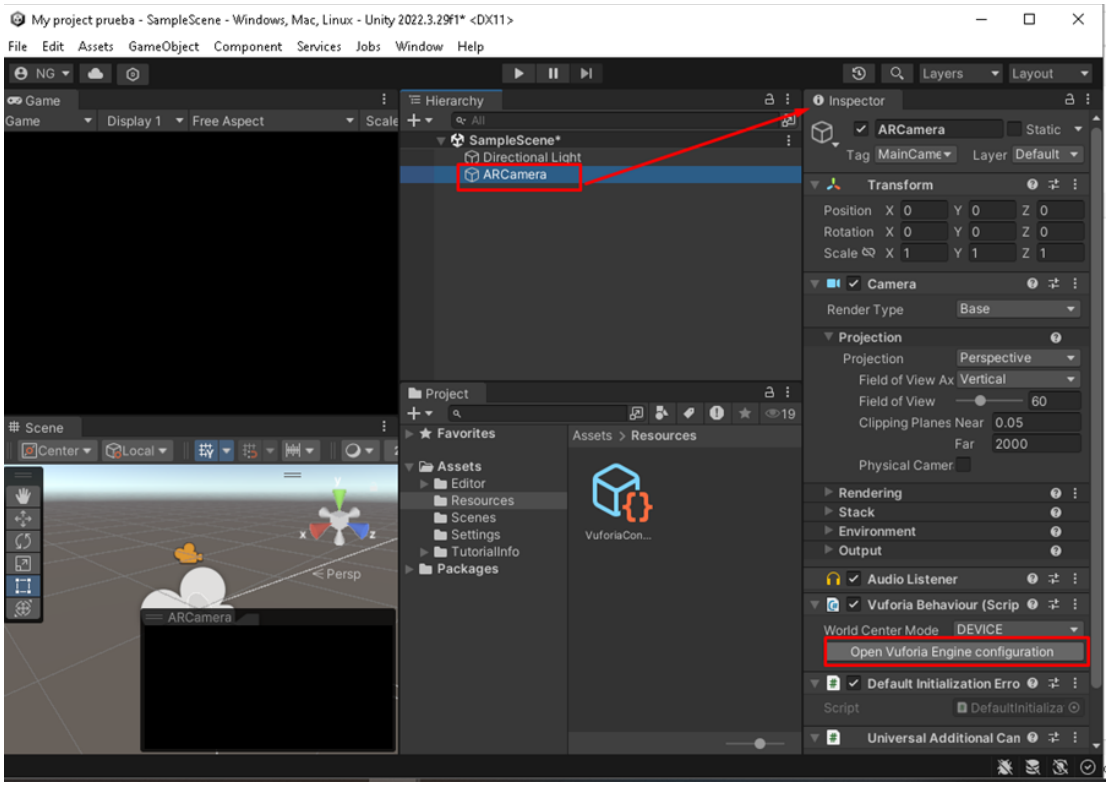


Figura 4.8

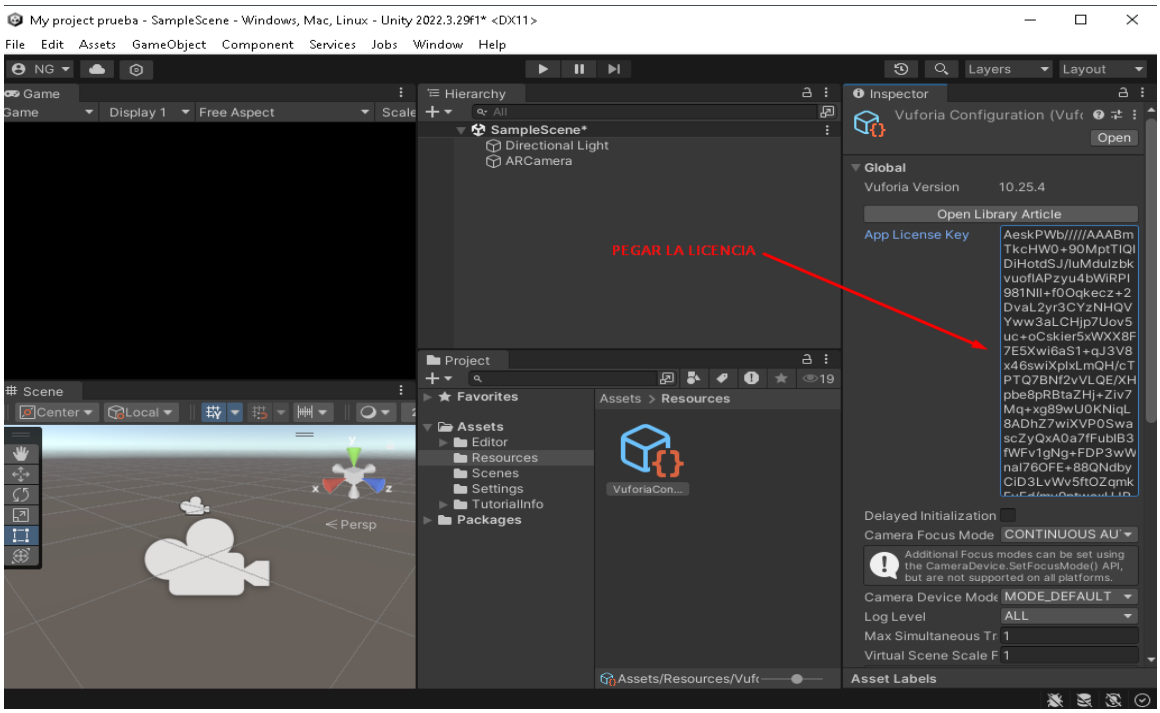

My project prueba - SampleScene - Windows, Mac, Linux - Unity 2022.3.29f1* <DX11>
File Edit Assets GameObject Component Services Jobs Window Help
Hierarchy
SampleScene*
Directional Light
ARCamera
Inspector
Vuforia Configuration
Open
Global
Vuforia Version 10.25.4
Open Library Article
App License Key
PEGAR LA LICENCIA
Delayed Initialization
Camera Focus Mode
Camera Device Mode
MODE_DEFAULT
Log Level
ALL
Asset Labels
Project
Favorites
Assets
Editor
Resources
Scenes
Settings
TutorialInfo
Packages
Assets > Resources
Scene
Persp

Figura 4.9

5. Configuración inicial de los ImageTargets

1. Después de instalar Vuforia, en el menú superior, selecciona **ARCamera > Vuforia Engine > ImageTarget**. **(Figura 5.1)**. Esto agregará un objeto **ImageTarget** a tu escena. En el inspector, verás una sección donde puedes elegir la imagen que deseas utilizar como target. **(Figura 5.2)**
2. Para definir la imagen física que Vuforia debe reconocer, debes subir una imagen a tu **Vuforia Developer Portal**.Se debe de ir a la opción "Target manager" para adicionar la base de datos y la tarjeta que va a ser reconocida por la cámara del dispositivo móvil. Haga clic en el botón "Add Database", asígnele el nombre" y utilice la opción "Type: Device" indicando que el target estará en el dispositivo y luego clic en el botón "Create". y luego descargar la base de datos de imágenes.**(Figura 5.3) (Figura 5.4).**
3. Una vez que ha sido creada la base de datos, debe hacer clic en el botón "Add Target", seleccionar "Type single imagine" y añadir la tarjeta, para lo cual se debe hacer clic en el botón "Browse" y luego buscar la imagen que se va a utilizar como marcador. **(Figura 5.5)**
4. Una vez adicionado el target, debe descargar la base de datos en la opción que dice "Download database all", activar la opción que dice "Unity editor" y hacer clic en el botón "Download". **(Figura 5.6) Figura (5.7).**
5. Importa la base de datos descargada en Unity a través de **Assets > Import Package** y selecciona la imagen en el componente **ImageTarget** o simplemente arrastrando el paquete descargado a la solapa Project
6. El paquete importado que es la imagen se guardará en el editor dentro de Asset en la solapa Project, dentro de esta carpeta se encuentra vuforia y finalmente la imagen que será usada como ImageTarget. **(Figura 6).**
7. Por último se crea un Objeto 3d en la solapa Hierarchy donde creamos el Arcamera ,el ImageTarget y dentro de este el Objeto 3D que mencionamos que en este caso es un "CUBE". **(Figura 6.1).**

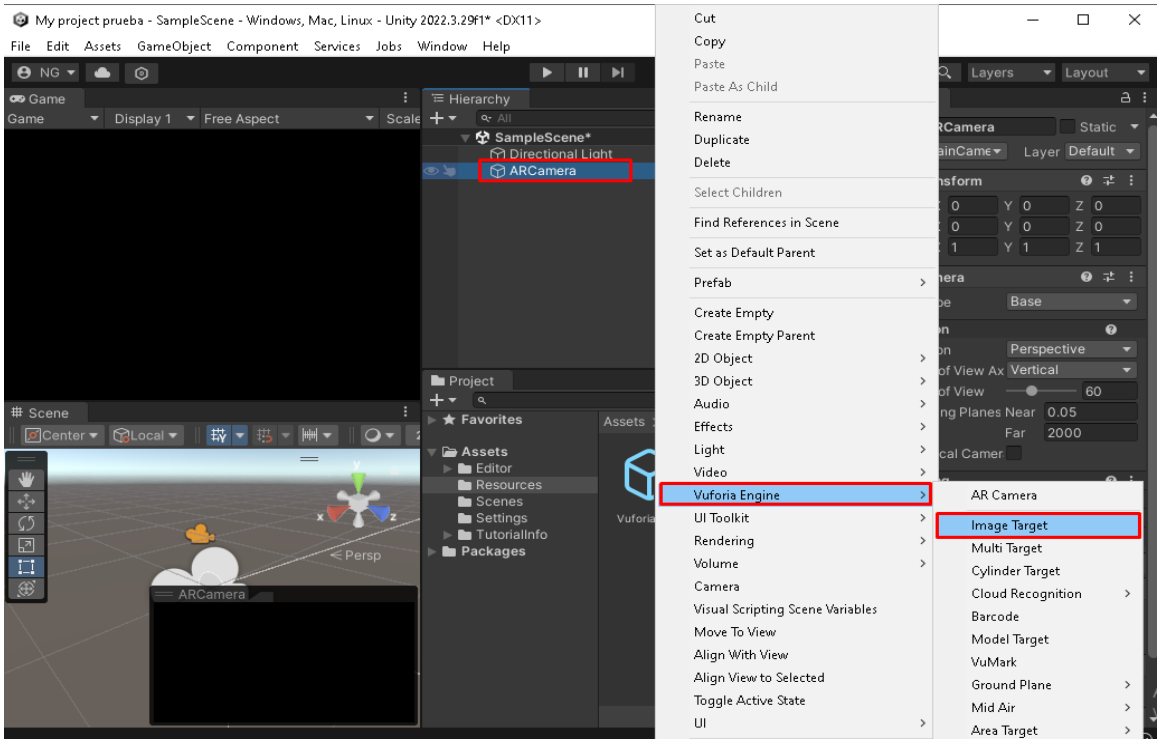


Figura 5.1

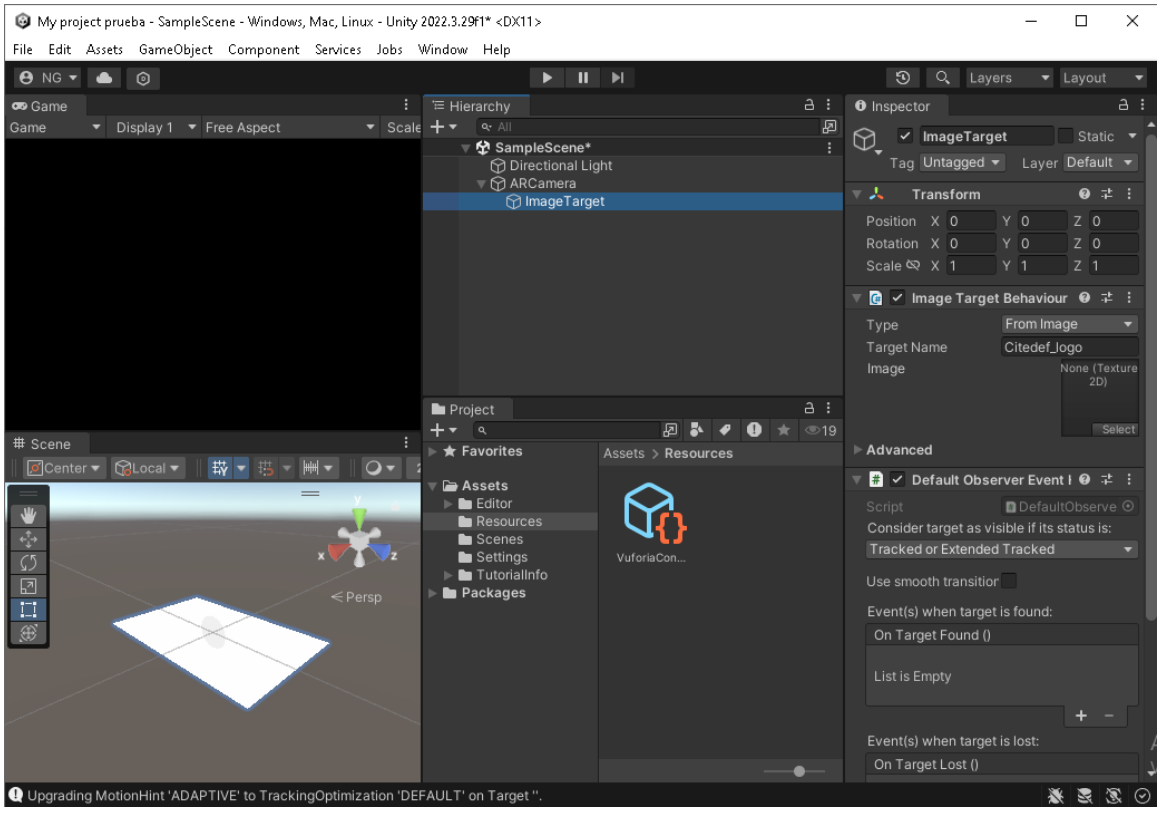


Figura 5.2

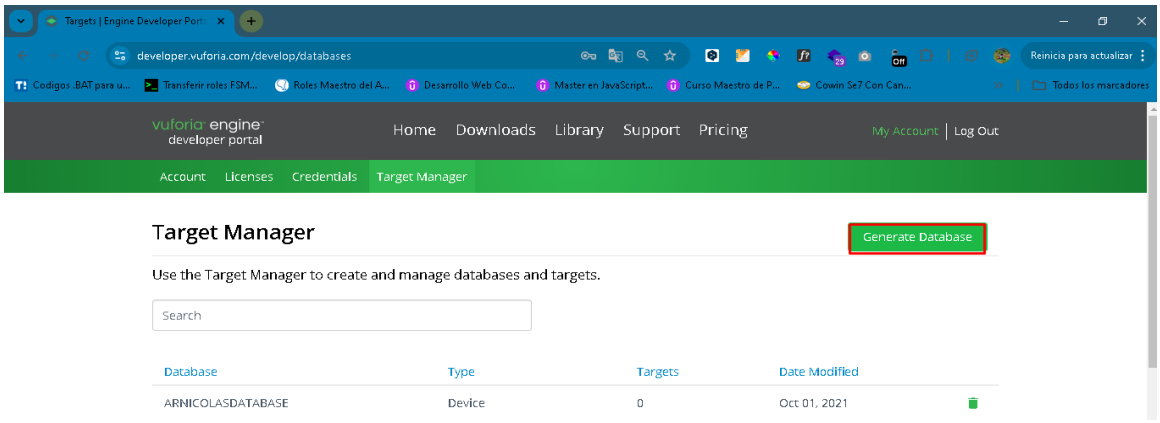


Figura 5.3

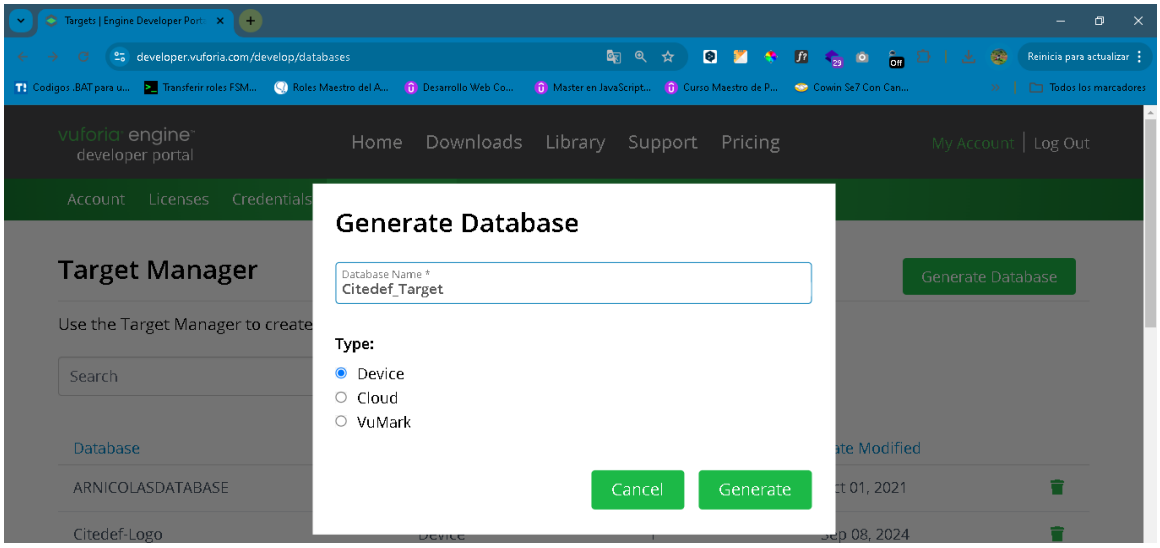


Figura 5.4

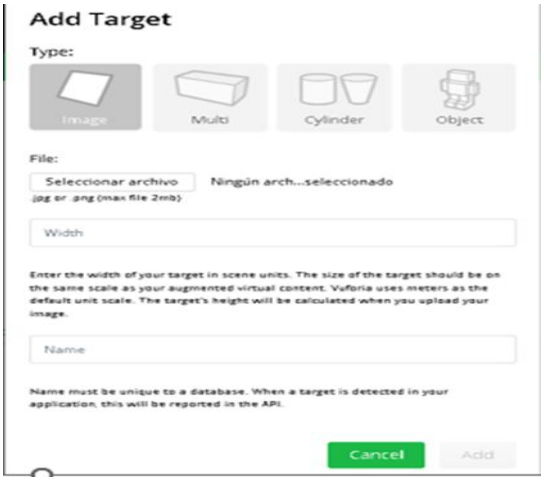


Figura 5.5

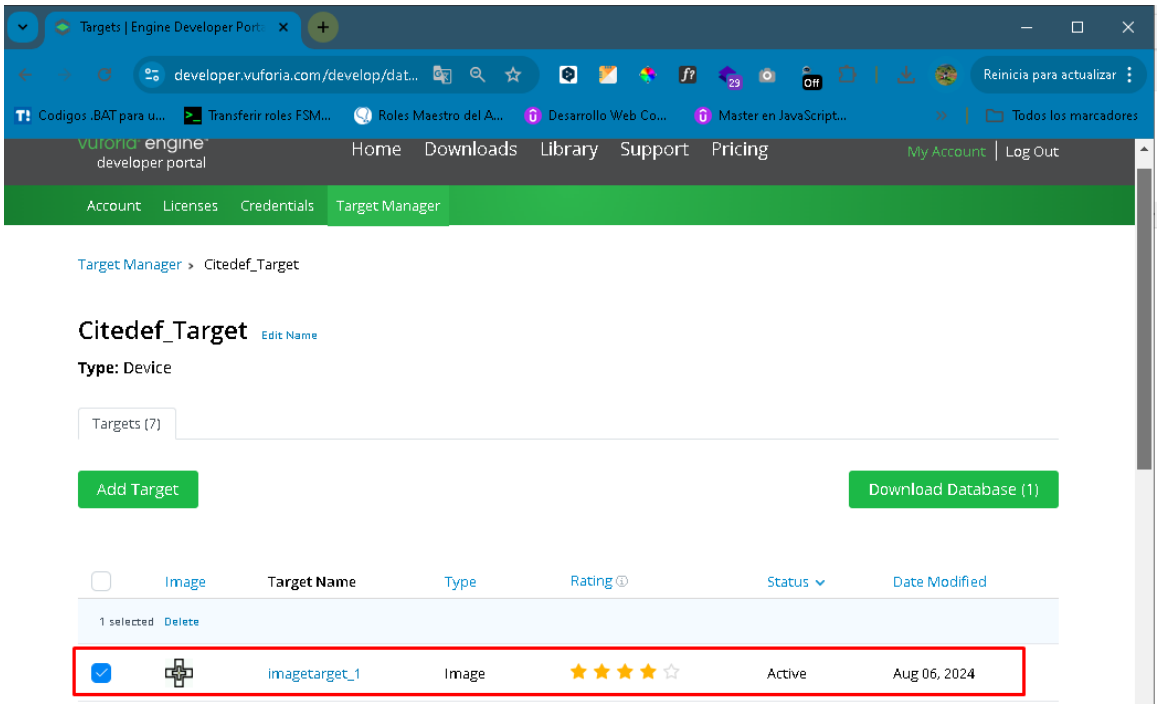


Figura 5.6

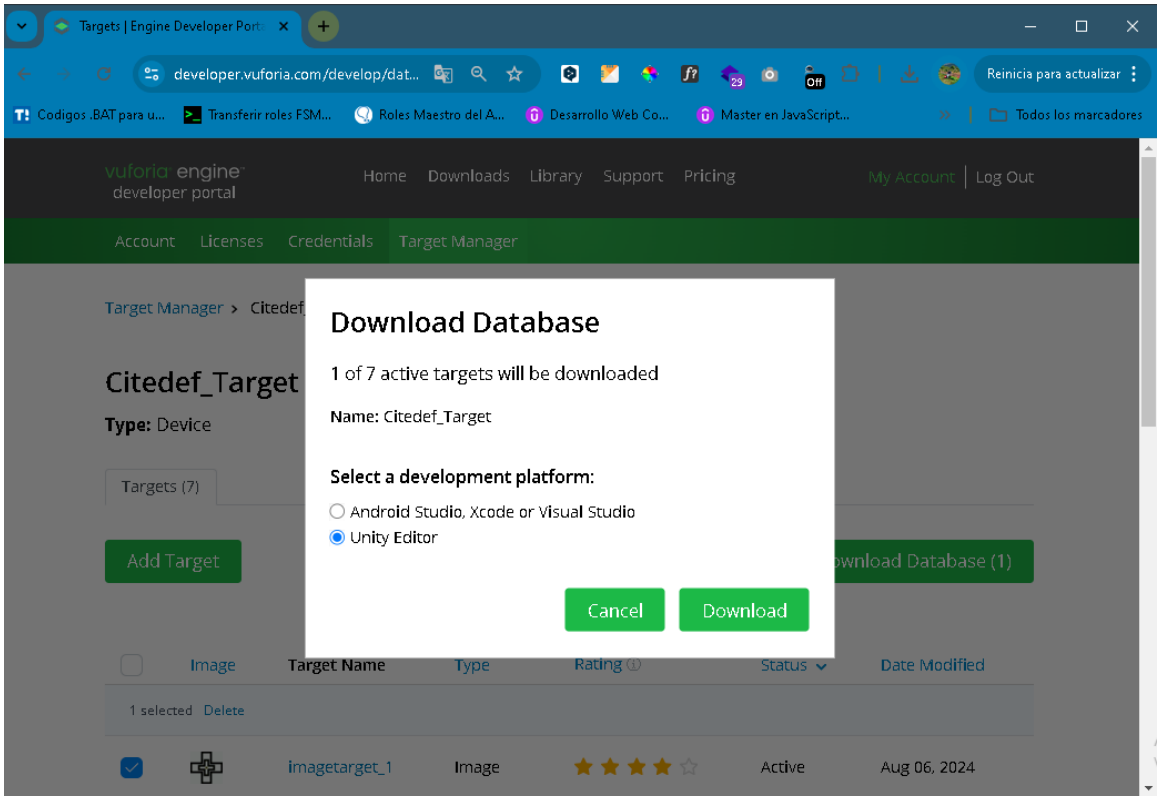


Figura 5.7

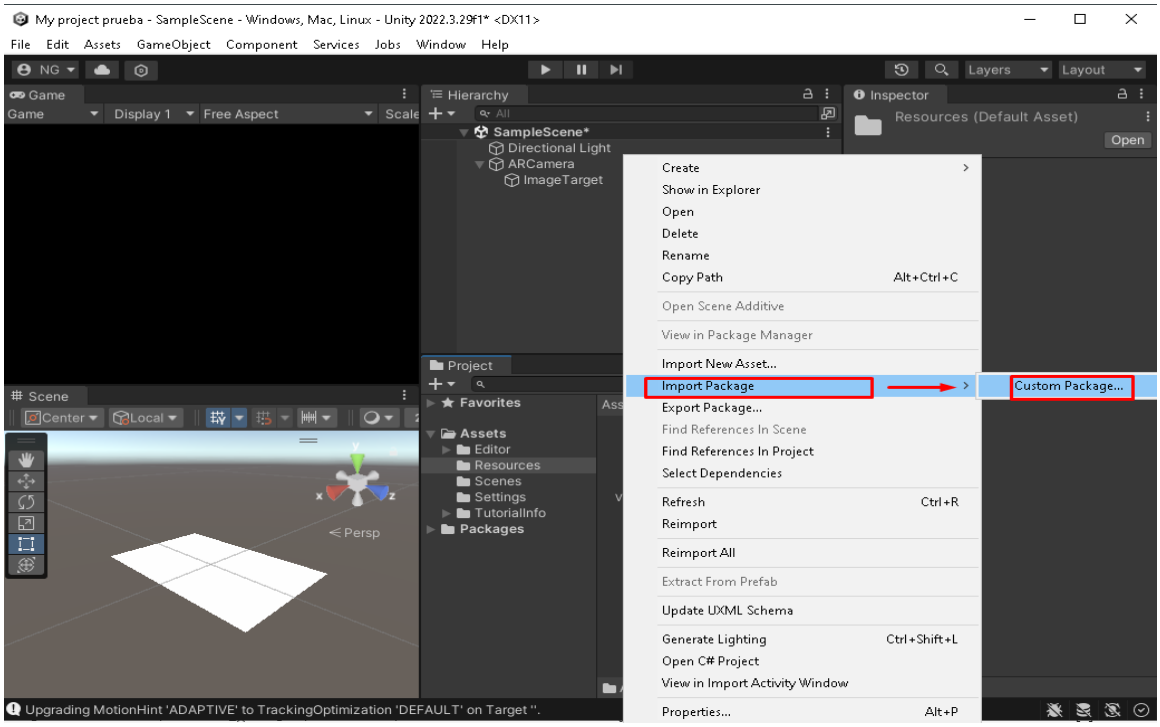


Figura 5.8

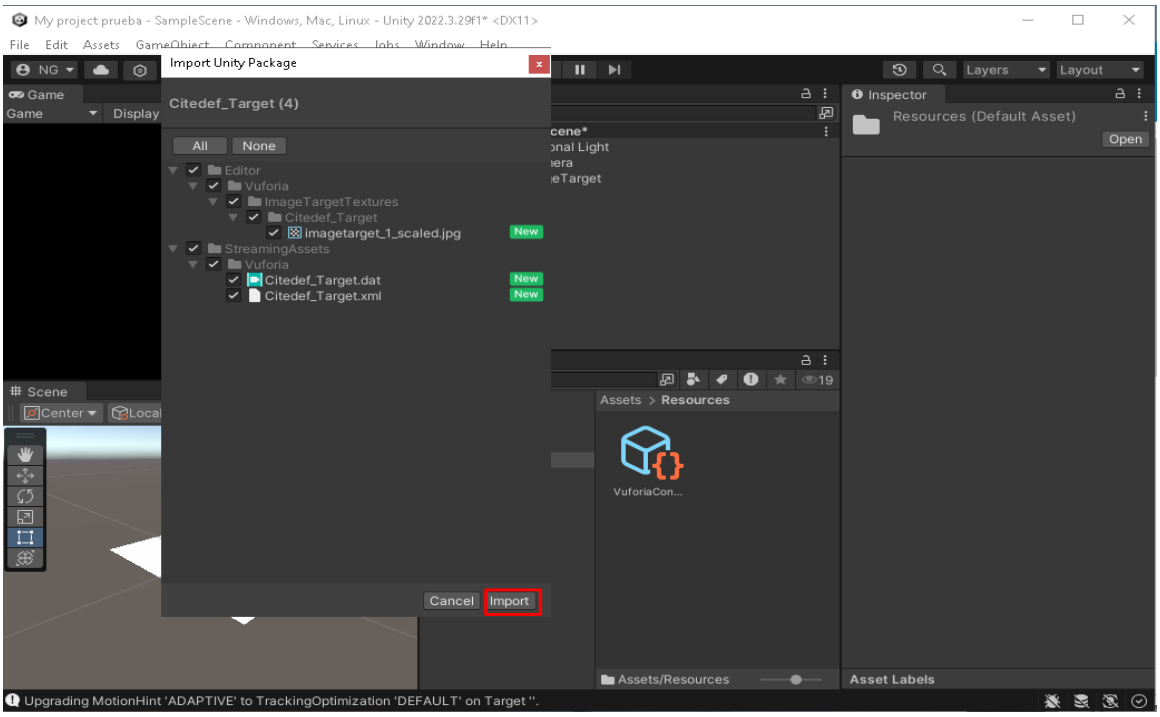


Figura 5.9

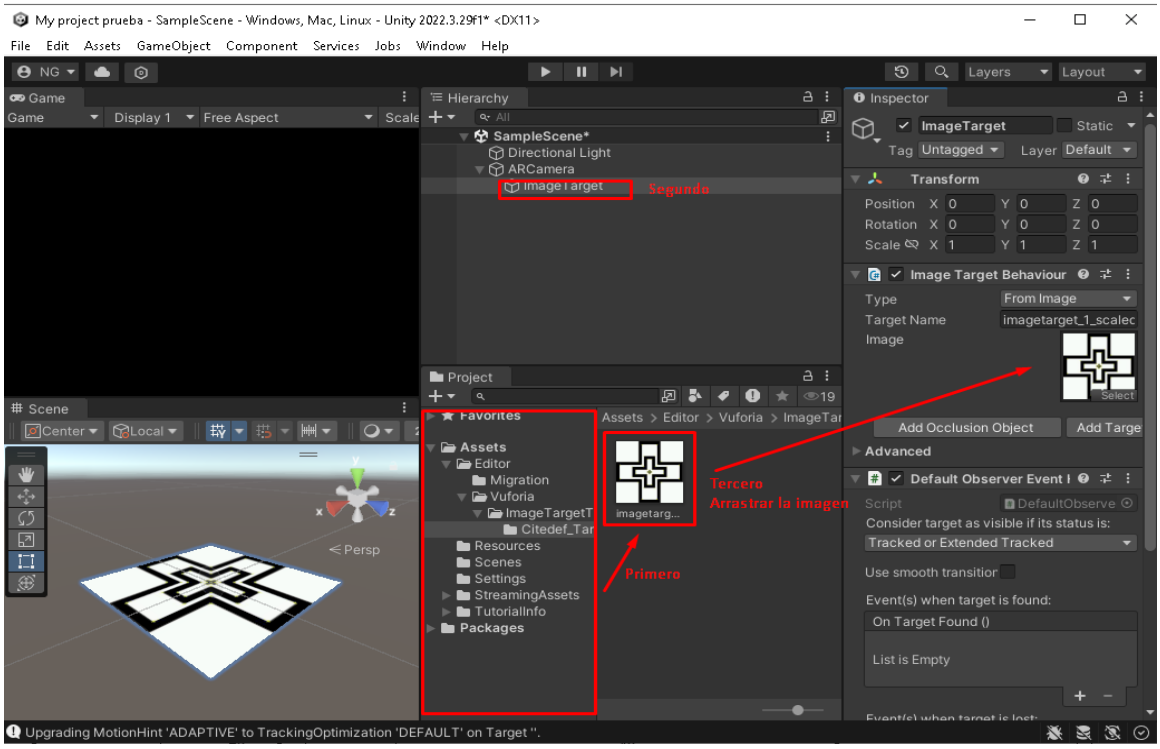


Figura 6

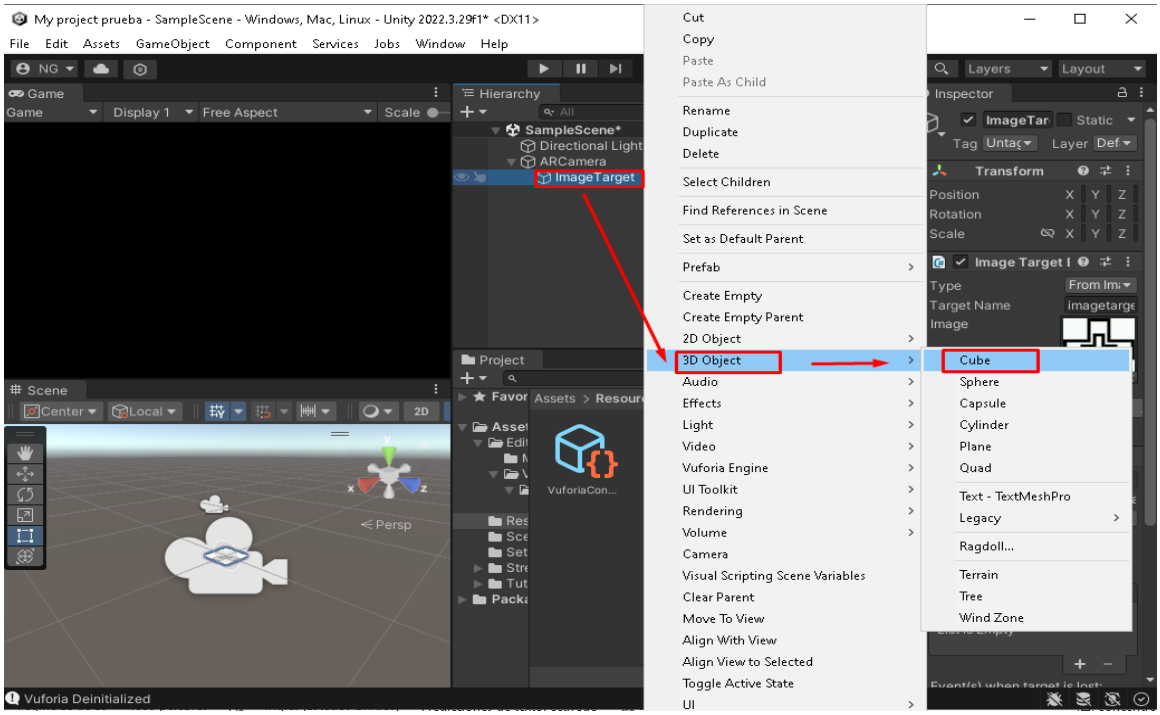


Figura 6.1

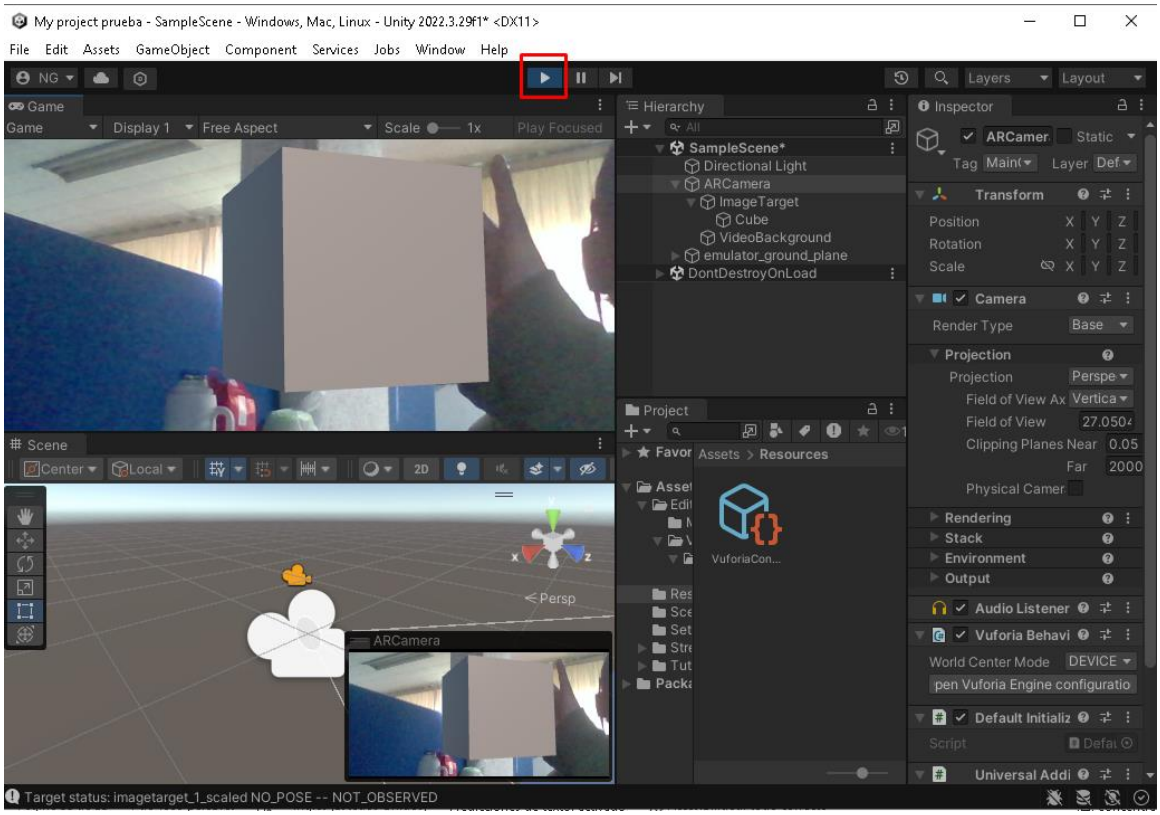


Figura 6.2

CAPITULO 4

A partir de este capítulo, se presentará un extracto de un trabajo en desarrollo, el cual está orientado a la aplicación de la realidad aumentada en un contexto militar. Es importante destacar que lo que se expone aquí no constituye el proyecto completo, sino una sección representativa de su desarrollo actual. Este proyecto tiene como objetivo inicial desarrollar un prototipo de aplicación de RA en dispositivos Android para mejorar la eficacia y la seguridad de las misiones militares conjuntas, con la posibilidad de migrar a dispositivos más avanzados en el futuro.

Lo que se intenta con este proyecto es tener conciencia situacional del soldado en su entorno, mediante una representación aumentada del individuo cuando está realizando una misión. Estos retos incluyen visibilidad limitada, falta de familiaridad con el medio ambiente, mala comunicación y, en general, problemas para la localización de objetivos e identificación de enemigos y fuerzas aliadas. En estas situaciones, saber quién está al lado, qué armamento tiene y cuál es la parte de su misión es de suma importancia, especialmente cuando no todos se conocen en un pelotón.

La realidad aumentada permitirá la ampliación del entorno real mediante la incorporación de simples "targets 2D", marcadores bidimensionales que servirán como puntos de referencia para superponer elementos virtuales en el mundo real. Por ejemplo, el médico del pelotón podría utilizar estos targets para obtener en tiempo real datos vitales sobre los soldados y poder intervenir adecuadamente.

Este capítulo explora cómo la aplicación de la RA puede transformar las operaciones militares, mejorando la conciencia situacional y optimizando la comunicación y coordinación en situaciones críticas, donde tener información en tiempo real puede marcar la diferencia entre el éxito y el fracaso en la misión.

Para abordar con éxito un proyecto de esta índole será necesario dividir y paralelizar el trabajo. Por esta razón el presente documento se centrará exclusivamente en la parte de control de la interfaz. Es cierto que existen numerosas maneras de navegar por una interfaz, mediante el uso de la voz, botones, movimiento ocular, etc. No obstante, una de las formas más naturales y que mejor se adapta a la situación, es mediante la realización de gestos con los dedos de las manos.

Este capítulo se ha estructurado con el propósito de guiar al lector de manera clara y detallada a través del proceso seguido para el desarrollo del proyecto. En primer lugar, en el apartado 4.1, se describirá todo el proceso relacionado con la configuración y selección del hardware. A continuación, en el apartado 4.2, se abordará el diseño y desarrollo de la interfaz de usuario, destacando los aspectos clave de su implementación. Finalmente, en el apartado 4.3, se incluirá un extracto del código desarrollado, proporcionando una visión técnica del funcionamiento del sistema.

4.1 Configuración y selección del hardware

En este apartado se detalla el proceso de selección y configuración del hardware utilizado para el desarrollo del prototipo de realidad aumentada. Para garantizar un rendimiento adecuado, se optó por dispositivos Android debido a su versatilidad, accesibilidad y compatibilidad con las tecnologías de realidad aumentada. El núcleo del sistema de RA se basa en la integración del SDK de Vuforia con Unity, los cuales permiten la detección y el seguimiento de objetos en tiempo real mediante el uso de la cámara del dispositivo móvil.

Además, se consideraron los requisitos de procesamiento gráfico y velocidad de respuesta, evaluando distintas opciones de dispositivos para asegurar que el prototipo funcione de manera eficiente. Se eligió una plataforma de bajo costo para hacer el proyecto más accesible y viable a nivel de desarrollo, manteniendo la posibilidad de migrar a hardware más avanzado en futuras versiones.

Ejemplo de selección de hardware: En este proyecto, se utilizó un dispositivo Samsung Galaxy A04s, cuya elección se debió a su procesador Octa-core de 2.0 GHz y su GPU Mali-G52, que proporcionan un equilibrio adecuado entre rendimiento y precio. Este dispositivo cuenta con una pantalla de 6.5 pulgadas y una resolución de 720x1600 píxeles, lo que resulta adecuado para aplicaciones de realidad aumentada que requieren una visualización clara y detallada de los objetos superpuestos. Además, su cámara de 50 MP se ajusta a los requisitos del SDK de Vuforia, permitiendo una detección precisa de los targets 2D.

La configuración inicial del hardware incluyó el uso de **Unity** como plataforma principal de desarrollo, permitiendo la integración directa con el SDK de Vuforia. Para realizar las pruebas y la depuración en tiempo real, se utilizó el dispositivo Android como plataforma de pruebas conectada directamente a Unity, lo que facilitó el proceso de ajuste del proyecto y optimización de los tiempos de respuesta.

Posibles mejoras en hardware: Si bien el uso de dispositivos Android es una opción accesible para el desarrollo y pruebas iniciales, existen opciones más avanzadas para mejorar la experiencia de usuario en futuras versiones del proyecto. Entre las posibles mejoras de hardware se encuentra la incorporación de lentes de realidad aumentada como las **DreamGlass** o las **Microsoft HoloLens**.

1. **DreamGlass**: Estas gafas de realidad aumentada ofrecen una pantalla transparente con una resolución de 2.5K, lo que permite una visualización clara de los objetos digitales superpuestos en el mundo real. Su campo de visión de 90 grados y la capacidad de conexión a dispositivos móviles las convierten en una opción ideal para mejorar la interacción en aplicaciones de RA, especialmente en entornos donde se requiere un alto nivel de precisión visual. Además, su portabilidad y diseño ergonómico permiten una mayor comodidad para el usuario durante misiones prolongadas. **(Figura 1)**

2. **Microsoft HoloLens 2**: Esta es otra opción avanzada para la migración del proyecto a un hardware más potente. HoloLens 2 combina una interfaz de RA avanzada con la capacidad de interactuar con el entorno físico a través del reconocimiento de gestos, seguimiento ocular y comandos de voz. Con una pantalla holográfica que proyecta imágenes tridimensionales en el espacio, este dispositivo podría mejorar significativamente la experiencia inmersiva y la interacción del usuario con los elementos virtuales del proyecto. **(Figura 2)**.

La incorporación de estos dispositivos de realidad aumentada más avanzados permitiría al proyecto expandirse en términos de funcionalidades y precisión, proporcionando una experiencia más rica y completa. Además, estos equipos pueden integrarse sin problemas con plataformas como Unity y Vuforia, lo que facilitaría la transición a hardware de mayor rendimiento sin grandes modificaciones en el desarrollo actual.

4.2 Diseño y desarrollo de la interfaz de usuario

El diseño de la interfaz de usuario (UI) para este proyecto de realidad aumentada (RA) se centró en ofrecer una experiencia de usuario clara y fluida, permitiendo la interacción con los elementos virtuales de manera natural y eficiente. Se implementó un sistema de paneles dinámicos y controles visuales que se despliegan y actualizan en función de la detección de objetivos en el entorno real.

El proceso comenzó con la creación de un canvas **(Figura 1)** que actuara como base para los elementos de la UI, utilizando Unity como la herramienta principal. El sistema se diseñó para mostrar información crítica, como los nombres y rangos de los soldados, un logotipo institucional y un panel dinámico que muestra el estado de los objetos detectados por la cámara. Se utilizaron prefabs para facilitar el modularidad de los elementos visuales, como los botones, switches, y círculos que cambian de color dependiendo de si los objetivos de imagen son detectados o no.

Además de los elementos visuales, se integró una funcionalidad de reconocimiento de audio que proporciona retroalimentación sonora cuando un ImageTarget es detectado o perdido. Este componente de audio mejora la interacción y la experiencia inmersiva del usuario, ya que permite reconocer, mediante señales auditivas, cuándo un objeto del entorno ha sido identificado por el sistema de RA. Este reconocimiento de audio también está vinculado a los eventos de activación y desactivación de los elementos visuales en la interfaz, aportando una dimensión adicional a la interacción.

La interfaz fue diseñada para adaptarse a distintas resoluciones de pantalla y dispositivos, garantizando una experiencia de usuario coherente en plataformas móviles. Se integraron funciones que permiten a los usuarios interactuar con los objetos en pantalla, actualizando la información en tiempo real cuando un ImageTarget es encontrado o perdido.

Además, se añadió un control para activar o desactivar los objetos virtuales mediante un interruptor (toggle), lo que otorga mayor control al usuario sobre la visualización de los elementos. Estos mecanismos se construyeron teniendo en cuenta la simplicidad y funcionalidad, evitando sobrecargar al usuario con elementos innecesarios y garantizando una navegación intuitiva. **(Figura 2)**

Figura 1

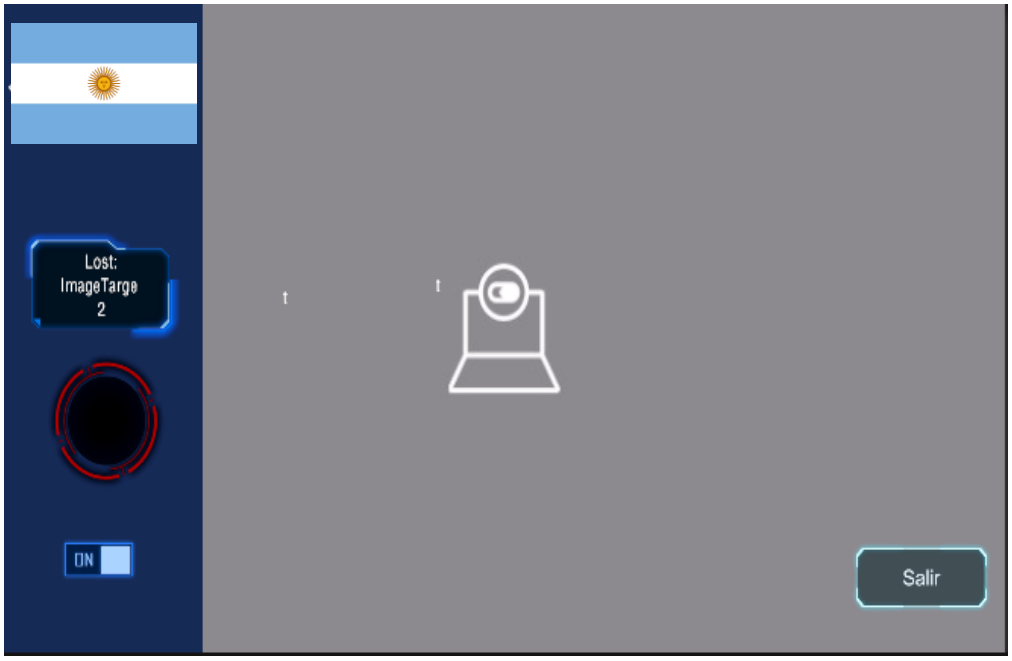


Figura 2

1.3 Extracto del codigo

En este apartado se presenta un extracto del código desarrollado para la implementación del proyecto de realidad aumentada. Este código integra Unity y Vuforia para la detección de targets y la superposición de objetos virtuales sobre el entorno real. Además, se han realizado mejoras en la estructura y organización del código para mejorar su legibilidad y modularidad.

Uno de los aspectos clave del desarrollo fue la creación de una interfaz de usuario dinámica que se actualiza en función de los ImageTargets detectados. La clase principal, CrearCanvas1, es responsable de gestionar la creación de elementos UI como paneles, botones y switches, que interactúan con los objetos del mundo real. El código ha sido diseñado para ser modular, reutilizable y fácil de escalar.

CODIGO:

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;



public class CrearCanvas1 : MonoBehaviour

{

    public Font customFont;

    public Sprite logoImage;


    public GameObject circlePrefab;


    public GameObject PanelPrefab;


    // Referencias a los prefabs de los switches

    public GameObject switchPrefabOn;

    //public GameObject switchPrefabOff;


    public GameObject BotonPrefab;


    public GameObject MostrarPrefab;
```

```
// Utilizar arrays para los Textos de nombre y rango
public Text[] soldierNameTexts = new Text[5];
public Text[] soldierRankTexts = new Text[5];

private Soldier[] soldiers = new Soldier[5]; // Array de soldados

private GameObject myCanvasObject; // Referencia al objeto Canvas para acceso global
private bool isImageTargetVisible = false;

// public GameObject imageTargetsContainer;
public GameObject[] imageTargets;



void Start()
{


    // Inicializar cada soldado y actualizar la información
    soldiers[0] = new Soldier("Pedro", "Cabo");
    soldiers[1] = new Soldier("Ruben", "Suboficial");
    soldiers[2] = new Soldier("Pablo", "Teniente");
    soldiers[3] = new Soldier("Jose", "Capitan");
    soldiers[4] = new Soldier("Ignacio", "Mayor");

    UpdateSoldierInfo();

    MiPanel();

}
```

```
void MiPanel()
{
    //GameObject myCanvasObject = new GameObject("CanvaElectronica");
    myCanvasObject = new GameObject("CanvaElectronica");

    Canvas myCanvas = myCanvasObject.AddComponent<Canvas>();
    myCanvas.renderMode = RenderMode.ScreenSpaceOverlay;
    CanvasScaler canvasScaler = myCanvasObject.AddComponent<CanvasScaler>();
    canvasScaler.uiScaleMode = CanvasScaler.ScaleMode.ScaleWithScreenSize;
    canvasScaler.referenceResolution = new Vector2(1920, 1080);
    myCanvasObject.AddComponent<GraphicRaycaster>();

    RectTransform canvasRect = myCanvasObject.GetComponent<RectTransform>();
    canvasRect.anchorMin = new Vector2(0, 0);
    canvasRect.anchorMax = new Vector2(1, 1);
    canvasRect.offsetMin = new Vector2(0, 0);
    canvasRect.offsetMax = new Vector2(0, 0);

    GameObject blueBackground = new GameObject("BlueBackground");
    blueBackground.transform.SetParent(myCanvasObject.transform, false);
    Image blueImage = blueBackground.AddComponent<Image>();
    blueImage.color = new Color(18f / 255f, 43f / 255f, 87f / 255f);
    RectTransform blueRect = blueImage.GetComponent<RectTransform>();
    blueRect.anchorMin = new Vector2(0, 0);
    blueRect.anchorMax = new Vector2(0.2f, 1);
    blueRect.offsetMin = new Vector2(0, 0);
    blueRect.offsetMax = new Vector2(0, 0);
    Logo(myCanvasObject.transform, new Vector2(10, 850));//Vector3(-923.210999,262.950989,0)
    Cartel(myCanvasObject.transform, new Vector2(180, 850));
    Switches(myCanvasObject.transform, new Vector2(10, 120));
    //Botones(myCanvasObject.transform, new Vector2(80, 120));
```

```
    //SliderVertical(myCanvasObject.transform, new Vector2(120, 250));
    BotonSalir(myCanvasObject.transform, new Vector2(1610, 50));
    EstadoPanel(myCanvasObject.transform, new Vector2(40, 520), "Texto Inicial");
    EstadoCirculo(myCanvasObject.transform, new Vector2(160, 350));

}



void EstadoCirculo(Transform parent, Vector2 anchoredPosition)
{
    GameObject circleInstance = Instantiate(circlePrefab, parent, false);
    RectTransform circleRect = circleInstance.GetComponent<RectTransform>();
    circleRect.anchorMin = new Vector2(0, 0);
    circleRect.anchorMax = new Vector2(0, 0);
    circleRect.pivot = new Vector2(0, 0);
    circleRect.anchoredPosition = anchoredPosition;

    // Obtiene todos los componentes Image en el prefab y sus hijos
    Image[] images = circleInstance.GetComponentsInChildren<Image>();
    if (images.Length > 0)
    {
        foreach (Image img in images)
        {
            img.color = isImageTargetVisible ? Color.green : Color.red;
        }
    }
    else
    {
        Debug.LogError("No Image components found on circlePrefab!");
    }
}
```

```
    // Este método se llamará cuando un ImageTarget sea encontrado.
    public void OnImageTargetFound(GameObject imageTarget)
    {
        isImageTargetVisible = true;
        EstadoCirculo(myCanvasObject.transform, new Vector2(160, 350)); // Asume que quieres cambiar el estado del círculo también.
        string targetName = imageTarget.name; // Obtiene el nombre del ImageTarget detectado
        EstadoPanel(myCanvasObject.transform, new Vector2(40, 520), "Detected: " + targetName); // Actualiza el panel con el nombre del target detectado
    }

    // Este método se llamará cuando un ImageTarget sea perdido.
    public void OnImageTargetLost(GameObject imageTarget)
    {
        isImageTargetVisible = false;
        EstadoCirculo(myCanvasObject.transform, new Vector2(160, 350)); // Asume que quieres cambiar el estado del círculo también.
        string targetName = imageTarget.name; // Obtiene el nombre del ImageTarget detectado
        EstadoPanel(myCanvasObject.transform, new Vector2(40, 520), "Lost: " + targetName);

        //EstadoPanel(myCanvasObject.transform, new Vector2(40, 520), "Lost: " + imageTarget.name); // Actualiza el panel indicando que el target fue perdido
    }

    /*
    public void OnImageTargetFound()
    {
        isImageTargetVisible = true;
        EstadoCirculo(myCanvasObject.transform, new Vector2(160, 350)); // Pasando los parámetros necesarios
        string targetName = imageTarget.name; // Obtiene el nombre del ImageTarget
        EstadoPanel(myCanvasObject.transform, new Vector2(160, 350), "Detected: " + targetName); // Actualiza el panel con el nombre
    }
```

```
    public void OnImageTargetLost()
    {
        isImageTargetVisible = false;
        EstadoCirculo(myCanvasObject.transform, new Vector2(160, 350)); // Pasando los parámetros necesarios
            EstadoPanel(myCanvasObject.transform, new Vector2(160, 350), "Lost: " + imageTarget.name); // Opcionalmente actualiza el panel al perder el target
    }
*/
    void EstadoPanel(Transform parent, Vector2 anchoredPosition, string textToShow)
    {
        // Crear instancia del panel
        GameObject panelInstance = Instantiate(PanelPrefab, parent, false);
        RectTransform panelRect = panelInstance.GetComponent<RectTransform>();
        panelRect.sizeDelta = new Vector2(300, 180);
        panelRect.anchorMin = new Vector2(0, 0);
        panelRect.anchorMax = new Vector2(0, 0);
        panelRect.pivot = new Vector2(0, 0);
        panelRect.anchoredPosition = anchoredPosition;

        // Crear objeto de texto como hijo del panel
        GameObject textGameObject = new GameObject("Text");
        textGameObject.transform.SetParent(panelInstance.transform, false);

        // Configurar el componente Text
        Text textComponent = textGameObject.AddComponent<Text>();
        textComponent.font = customFont;
        textComponent.text = textToShow; // Usa el nombre del ImageTarget
        textComponent.fontSize = 35;
        textComponent.alignment = TextAnchor.MiddleCenter;
        textComponent.color = Color.white;
```

```
    // Configurar RectTransform del texto para que llene el panel
    RectTransform textRect = textGameObject.GetComponent<RectTransform>();
    textRect.anchorMin = new Vector2(0, 0);
    textRect.anchorMax = new Vector2(1, 1);
    textRect.offsetMin = new Vector2(0, 0);
    textRect.offsetMax = new Vector2(0, 0);
}



/*
    void EstadoPanel(Transform parent, Vector2 anchoredPosition)
    {
        // Crear instancia del panel
        GameObject panelInstance = Instantiate(PanelPrefab, parent, false);
        RectTransform panelRect = panelInstance.GetComponent<RectTransform>();
        panelRect.sizeDelta = new Vector2(300, 90);
        panelRect.anchorMin = new Vector2(0, 0);
        panelRect.anchorMax = new Vector2(0, 0);
        panelRect.pivot = new Vector2(0, 0);
        panelRect.anchoredPosition = anchoredPosition;

        // Crear objeto de texto como hijo del panel
        GameObject textGameObject = new GameObject("Text");
        textGameObject.transform.SetParent(panelInstance.transform, false);

        // Configurar el componente Text
        Text textComponent = textGameObject.AddComponent<Text>();
        textComponent.font = customFont;
        textComponent.text = "Texto de Ejemplo";
        textComponent.fontSize = 40;
        textComponent.alignment = TextAnchor.MiddleCenter;
        textComponent.color = Color.white;
```

```
        // Configurar RectTransform del texto para que llene el panel
        RectTransform textRect = textGameObject.GetComponent<RectTransform>();
        textRect.anchorMin = new Vector2(0, 0);
        textRect.anchorMax = new Vector2(1, 1);
        textRect.offsetMin = new Vector2(0, 0);
        textRect.offsetMax = new Vector2(0, 0);
    }

*/

void Logo(Transform parent, Vector2 anchoredPosition)
{
    GameObject logoObject = new GameObject("Logo");
    logoObject.transform.SetParent(parent, false);
    Image logo = logoObject.AddComponent<Image>();
    logo.sprite = logoImage;
    RectTransform logoRect = logo.GetComponent<RectTransform>();
    logoRect.sizeDelta = new Vector2(170, 180);
    logoRect.anchorMin = new Vector2(0, 0);
    logoRect.anchorMax = new Vector2(0, 0);
    logoRect.pivot = new Vector2(0, 0);
    logoRect.anchoredPosition = anchoredPosition;

}

void Cartel(Transform parent, Vector2 anchoredPosition)
{
    GameObject TextObject = new GameObject("Cartel");
    TextObject.transform.SetParent(parent, false);
    //TextObject.transform.SetParent(buttonObject.transform, false);
    Text CartelText = TextObject.AddComponent<Text>();
```

```
    CartelText.text = "Instituto de Investigaciones Científicas y Tecnicas para la Defensa";
    CartelText.font = customFont;
    CartelText.fontSize = 28;
    CartelText.color = Color.white;
    CartelText.alignment = TextAnchor.MiddleCenter;
    RectTransform textRect = CartelText.GetComponent<RectTransform>();
    textRect.sizeDelta = new Vector2(185, 180);
    textRect.anchorMin = new Vector2(0, 0);
    textRect.anchorMax = new Vector2(0, 0);
    textRect.pivot = new Vector2(0, 0);
    textRect.anchoredPosition = anchoredPosition;

}
/*
void Cartel(Transform parent, Vector2 anchoredPosition)
{
    GameObject TextObject = new GameObject("Cartel");
    TextObject.transform.SetParent(parent, false);

    TextMeshProUGUI CartelText = TextObject.AddComponent<TextMeshProUGUI>();
    CartelText.text = "Instituto de Investigaciones Científicas y Técnicas para la Defensa";
    CartelText.font = customFont; // Asegúrate de cambiar esto
    CartelText.fontSize = 20;
    CartelText.color = Color.white;
    CartelText.alignment = TextAlignmentOptions.Center;
    RectTransform textRect = CartelText.GetComponent<RectTransform>();
    textRect.anchorMin = new Vector2(0.5f, 0.5f);
    textRect.anchorMax = new Vector2(0.5f, 0.5f);
    textRect.pivot = new Vector2(0.5f, 0.5f);
    textRect.anchoredPosition = anchoredPosition;
    textRect.sizeDelta = new Vector2(300, 50); // Define el tamaño según sea necesario
}
```

```
*/

    void Botones(Transform parent, Vector2 anchoredPosition)
    {
        GameObject buttonObject = new GameObject("MiBoton");
        buttonObject.transform.SetParent(parent, false);
        Button button = buttonObject.AddComponent<Button>();
        Image buttonImage = buttonObject.AddComponent<Image>();
        buttonImage.color = new Color(255f / 255f, 0f / 255f, 0f / 255f);
        RectTransform buttonRect = button.GetComponent<RectTransform>();
        buttonRect.sizeDelta = new Vector2(250, 70);
        buttonRect.anchorMin = new Vector2(0, 0);
        buttonRect.anchorMax = new Vector2(0, 0);
        buttonRect.pivot = new Vector2(0, 0);
        buttonRect.anchoredPosition = anchoredPosition;
        Shadow buttonShadow = buttonImage.gameObject.AddComponent<Shadow>();
        buttonShadow.effectColor = new Color(0, 0, 0, 0.7f);
        buttonShadow.effectDistance = new Vector2(6, -6);
        GameObject buttonTextObject = new GameObject("TextoBoton");
        buttonTextObject.transform.SetParent(buttonObject.transform, false);
        Text buttonText = buttonTextObject.AddComponent<Text>();
        buttonText.text = "ACTIVAR IMAGE TARGET";
        buttonText.font = customFont;
        buttonText.fontSize = 24;
        buttonText.color = Color.white;
        buttonText.alignment = TextAnchor.MiddleCenter;
        RectTransform textRect = buttonText.GetComponent<RectTransform>();
        textRect.anchorMin = new Vector2(0, 0);
        textRect.anchorMax = new Vector2(1, 1);
        textRect.pivot = new Vector2(0.5f, 0.5f);
        textRect.sizeDelta = new Vector2(0, 0);
        textRect.anchoredPosition = new Vector2(0, 0);
```

```
    }

    void BotonSalir(Transform parent, Vector2 anchoredPosition)
    {
        // Instanciar el BotonPrefab
        GameObject buttonObject = Instantiate(BotonPrefab, parent, false);

        // Obtener el componente RectTransform y configurar sus propiedades
        RectTransform buttonRect = buttonObject.GetComponent<RectTransform>();
        buttonRect.sizeDelta = new Vector2(300, 150);
        buttonRect.anchorMin = new Vector2(0, 0);
        buttonRect.anchorMax = new Vector2(0, 0);
        buttonRect.pivot = new Vector2(0, 0);
        buttonRect.anchoredPosition = anchoredPosition;

        // Configurar el componente Button y asignar la función que se ejecutará al hacer clic
        Button button = buttonObject.GetComponent<Button>();
        button.onClick.AddListener(delegate { Salir(); });

        // Configurar el texto del botón si es necesario
        Text buttonText = buttonObject.GetComponentInChildren<Text>(); // Asumiendo que el prefab ya tiene un componente Text
        if (buttonText != null)
        {
            buttonText.text = "Salir";
        }
    }

    // Función a ejecutar cuando se presione el botón Salir
    void Salir()
    {
        Debug.Log("Salir del juego o aplicación");
```

```
    // Aquí puedes agregar código para cerrar la aplicación, por ejemplo:
    // Application.Quit();
}



/*    void Switches(Transform parent, Vector2 anchoredPosition)
    {
        GameObject switchToggleObject = Instantiate(switchPrefabOn, parent, false);
        RectTransform switchToggleRect = switchToggleObject.GetComponent<RectTransform>();
        switchToggleRect.sizeDelta = new Vector2(350, 70);
        switchToggleRect.anchorMin = new Vector2(0, 0);
        switchToggleRect.anchorMax = new Vector2(0, 0);
        switchToggleRect.pivot = new Vector2(0, 0);
        switchToggleRect.anchoredPosition = anchoredPosition;

        // Configurar el Toggle para controlar los ImageTargets
        Toggle toggle = switchToggleObject.GetComponent<Toggle>();
        if (toggle != null)
        {
            // Asegúrate de que el estado inicial del Toggle refleje la visibilidad actual de los ImageTargets
            toggle.isOn = imageTargetsContainer.activeSelf;

            // Añade el listener que manejará el cambio de estado del Toggle
            toggle.onValueChanged.AddListener(delegate { ToggleImageTargets(toggle.isOn); });
        }
        else
        {
            Debug.LogError("Toggle component not found on the switch prefab");
        }
    }

    // Función para activar o desactivar los ImageTargets
```

```
    private void ToggleImageTargets(bool isOn)
    {
        if (imageTargetsContainer != null)
        {
            imageTargetsContainer.SetActive(isOn);
        }
        else
        {
            Debug.LogError("ImageTargets container not assigned or found.");
        }
    }
*/

// Esta función crea e instancia el switch en la UI
public void Switches(Transform parent, Vector2 anchoredPosition)
{
    GameObject switchToggleObject = Instantiate(switchPrefabOn, parent, false);
    RectTransform switchToggleRect = switchToggleObject.GetComponent<RectTransform>();
    switchToggleRect.sizeDelta = new Vector2(350, 70);
    switchToggleRect.anchorMin = new Vector2(0, 0);
    switchToggleRect.anchorMax = new Vector2(0, 0);
    switchToggleRect.pivot = new Vector2(0, 0);
    switchToggleRect.anchoredPosition = anchoredPosition;

    // Configurar el Toggle para controlar los ImageTargets
    Toggle toggle = switchToggleObject.GetComponent<Toggle>();
    if (toggle != null)
    {
        // Inicialmente, asume que todos los ImageTargets están activos
        toggle.isOn = CheckImageTargetsActive();

        // Añade el listener que manejará el cambio de estado del Toggle
```

```
        toggle.onValueChanged.AddListener(delegate { ToggleImageTargets(toggle.isOn); });
    }
    else
    {
        Debug.LogError("Toggle component not found on the switch prefab");
    }
}

// Función para activar o desactivar los ImageTargets
private void ToggleImageTargets(bool isOn)
{
    foreach (GameObject target in imageTargets)
    {
        if (target != null)
        {
            target.SetActive(isOn);
        }
    }
}

// Verifica si todos los ImageTargets están activos
private bool CheckImageTargetsActive()
{
    foreach (GameObject target in imageTargets)
    {
        if (target != null && !target.activeSelf)
        {
            return false; // Retorna falso si alguno está desactivado
        }
    }
    return true; // Retorna verdadero si todos están activados
}
```

```
void SliderVertical(Transform parent, Vector2 anchoredPosition)
{
    GameObject sliderObject = new GameObject("VerticalSlider");
    sliderObject.transform.SetParent(parent, false);
    Slider slider = sliderObject.AddComponent<Slider>();
    slider.direction = Slider.Direction.BottomToTop;

    RectTransform sliderRect = slider.GetComponent<RectTransform>();
    sliderRect.sizeDelta = new Vector2(20, 300);
    sliderRect.anchorMin = new Vector2(0, 0);
    sliderRect.anchorMax = new Vector2(0, 0);
    sliderRect.pivot = new Vector2(0, 0);
    sliderRect.anchoredPosition = anchoredPosition;
    GameObject background = new GameObject("Background");
    background.transform.SetParent(sliderObject.transform, false);
    Image backgroundImage = background.AddComponent<Image>();
    backgroundImage.color = new Color(0.8f, 0.8f, 0.8f, 0.5f);
    RectTransform bgRect = backgroundImage.GetComponent<RectTransform>();
    bgRect.anchorMin = new Vector2(0, 0);
    bgRect.anchorMax = new Vector2(1, 1);
    bgRect.sizeDelta = new Vector2(0, 0);
    GameObject handle = new GameObject("Handle");
    handle.transform.SetParent(sliderObject.transform, false);
    Image handleImage = handle.AddComponent<Image>();
    handleImage.color = Color.red;
    slider.handleRect = handleImage.GetComponent<RectTransform>();
    slider.handleRect.sizeDelta = new Vector2(10, 10);
    slider.minValue = 0;
    slider.maxValue = 100;
```

```
        slider.value = 50;
    }

    private void UpdateSoldierInfo()
    {
        for (int i = 0; i < soldiers.Length; i++)
        {
            if (soldierNameTexts[i] != null && soldierRankTexts[i] != null)
            {
                soldierNameTexts[i].text = "Nombre: " + soldiers[i].Name;
                soldierRankTexts[i].text = "Rango: " + soldiers[i].Rank;
            }
        }
    }

}
```

Principales mejoras

Uso de prefabs(Elementos prediseñado): Para garantizar la modularidad y reutilización, los elementos de la UI, como paneles, botones y switches, se generan a partir de prefabs. Esto simplifica la modificación y actualización de los elementos UI sin necesidad de duplicar el código.

Detección de ImageTargets: Se incluyen métodos como OnImageTargetFound y OnImageTargetLost, que gestionan la detección y pérdida de los objetivos visuales (ImageTargets). Estas funciones permiten cambiar dinámicamente los elementos visuales, como el color de un círculo o el texto en un panel, dependiendo de si un objetivo ha sido detectado.

Mejora en la legibilidad y organización del código: Para mejorar la claridad del código, se implementaron métodos separados para cada funcionalidad específica, como la creación de logos (Logo), paneles de estado (Estado Panel) y switches (Switches). Esto facilita la comprensión y el mantenimiento del código.

Optimización para dispositivos móviles: El código ha sido optimizado para funcionar eficientemente en dispositivos móviles, priorizando la fluidez en la interacción. Esto se ha logrado mediante la gestión adecuada de la actualización de los elementos UI solo cuando es necesario, como al detectar o perder un ImageTarget.

Ejemplo de método OnImageTargetFound:

Este método se activa cuando se detecta un ImageTarget, actualizando el color del círculo y el panel de estado.

```
public void OnImageTargetFound(GameObject imageTarget)

{

isImageTargetVisible = true;

EstadoCirculo(myCanvasObject.transform, new Vector2(160, 350)); // Cambiar el estado del círculo

string targetName = imageTarget.name; // Obtiene el nombre del ImageTarget detectado

EstadoPanel(myCanvasObject.transform, new Vector2(40, 520), “Detected: “ + targetName); // Actualiza el panel con el nombre del target detectado

}.
```

Manejo de eventos UI:

El código también permite la creación dinámica de botones y switches que responden a los cambios en la detección de ImageTargets.

```
void BotonSalir(Transform parent, Vector2 anchoredPosition)

{

GameObject buttonObject = Instantiate(BotonPrefab, parent, false);

RectTransform buttonRect = buttonObject.GetComponent<RectTransform>();

buttonRect.sizeDelta = new Vector2(300, 150);

buttonRect.anchoredPosition = anchoredPosition;


Button button = buttonObject.GetComponent<Button>();

button.onClick.AddListener(delegate { Salir(); });

Text buttonText = buttonObject.GetComponentInChildren<Text>();
buttonText.text = “Salir”;

}


void Salir()

{

Debug.Log(“Salir del juego o aplicación”);

}
```

Uso de prefabs para modularidad:

Los prefabs permiten la creación rápida de componentes UI como switches y paneles. A continuación, se muestra un ejemplo de cómo se crea un switch que controla los ImageTargets.

```
public void Switches(Transform parent, Vector2 anchoredPosition)
{
    GameObject switchToggleObject = Instantiate(switchPrefabOn, parent, false);
    RectTransform switchToggleRect = switchToggleObject.GetComponent<RectTransform>();
    switchToggleRect.sizeDelta = new Vector2(350, 70);
    switchToggleRect.anchoredPosition = anchoredPosition;

    Toggle toggle = switchToggleObject.GetComponent<Toggle>();
    toggle.isOn = CheckImageTargetsActive();
    toggle.onValueChanged.AddListener(delegate { ToggleImageTargets(toggle.isOn); });
}
```

Integración de JSON para configuración dinámica

Una de las primeras mejoras implementadas fue la incorporación de **JSON** para la carga dinámica de datos. Esto permite cambiar las configuraciones de los soldados, la interfaz de usuario (UI) y otros elementos sin necesidad de modificar el código directamente, lo que aporta flexibilidad y facilita la personalización.

Por ejemplo, el archivo JSON puede contener la información de los soldados y la configuración de la UI, lo que permite a los desarrolladores o usuarios modificar estos datos externamente y adaptarlos a distintos contextos. El código a continuación carga la configuración desde un archivo JSON:

Archivo C# para llamar al JSON:

```
// Soldier.cs

public class Soldier

{

public string Name { get; set; }

public string Rank { get; set; }


public Soldier(string name, string rank)

{

Name = name;

Rank = rank;

}

}
```

ARCHIVO JSON:

```
//SoldierInfo.Json

{

   "Name": "John Doe",

   "Rank": "Sargento"

}
```

CAPITULO 5

En este capítulo se detalla posibles mejoras a futuros con la integración de Opencv si se desea usar todavía esta plataforma.

El trabajo ha demostrado un gran potencial al integrar tecnologías de realidad aumentada mediante Unity y Vuforia, permitiendo mejorar la conciencia situacional en contextos militares. Sin embargo, existen posibilidades para potenciar aún más el sistema, aprovechando tecnologías adicionales como el uso del algoritmo ArUco de OpenCV.

Casos en los que Unity y OpenCV pueden trabajar Juntos

En proyectos donde Unity es indispensable, se puede aprovechar OpenCV para tareas específicas de visión computacional, como:

- Procesamiento Previo de Imágenes:
  - Realizar la detección de marcadores con OpenCV en un backend y luego enviar los datos de posición y orientación a Unity.
- Sistemas de Detección Auxiliar:
  - Usar OpenCV como un sistema complementario para mejorar la precisión de la detección realizada por Vuforia.

Recomendación para una posible implementación en Unity. Implementación en un sistema híbrido

En lugar de forzar una integración completa de OpenCV y Unity, se puede diseñar un sistema híbrido:

- Backend Independiente con OpenCV:
  - Implementar la detección de marcadores y el cálculo de poses en un backend independiente usando Python o C++ con OpenCV.
- Comunicación entre Procesos:
  - Utilizar protocolos de comunicación (como sockets o APIs REST) para enviar los datos procesados a Unity.
- Visualización en Unity:
  - En Unity, usar estos datos para superponer los objetos virtuales, reduciendo la carga de procesamiento en tiempo real dentro del entorno.

Conclusión del trabajo en desarrollo de realidad aumentada para dispositivos móviles

El trabajo tuvo como objetivo desarrollar un prototipo funcional de Realidad Aumentada para dispositivos móviles Android, orientado a mejorar la eficacia y seguridad en entornos operativos. Este desarrollo representó un avance innovador en la incorporación de tecnologías de RA, explorando soluciones avanzadas para escenarios de alta exigencia táctica.

Logros alcanzados

1. **Prototipo funcional adaptable**:
   La aplicación permite detectar objetivos 2D y ofrecer información personalizada según el perfil del usuario. Los perfiles están diseñados para diferentes roles operativos, como médicos, personal en campo o líderes de equipo, con acceso a datos específicos adaptados a sus responsabilidades.

   Por ejemplo:
   - Un médico puede visualizar datos de salud en tiempo real para intervenciones adecuadas.
   - Un líder de equipo puede acceder a información estratégica como armamento disponible y objetivos asignados.
2. **Flexibilidad operativa mediante menú dinámico**:
   Se implementó un sistema de selección en pantalla táctil que permite cambiar la información y contenidos multimedia asociados a cada objetivo (símbolo o imagen), adaptándose dinámicamente a las necesidades del usuario y la misión.

3. **Pruebas exitosas en capacidades avanzadas**:
   En evaluaciones iniciales, se validó el potencial del sistema para reconocimiento facial, objetivos 3D y objetos reales, mostrando resultados prometedores para futuras implementaciones.
4. **Fortalecimiento del equipo técnico**:
   El proyecto permitió adquirir experiencia práctica con herramientas como Unity y Vuforia, consolidando las capacidades técnicas del equipo para futuros desarrollos.

Aspectos pendientes y oportunidades

1. **Hardware especializado**:
   Aunque el prototipo fue exitoso en dispositivos Android, es mucho mas fiable en hardware especifico de realidad aumentada o sistemas embebidos.
2. **Reconocimiento avanzado**:
   Es necesario implementar tecnologías más robustas para mejorar el reconocimiento facial, biométrico y de objetos, automatizando y enriqueciendo las funcionalidades actuales.
3. **Validación en campo**:
   Aún se requieren pruebas integrales en escenarios reales para evaluar completamente la funcionalidad del sistema bajo condiciones operativas exigentes.

CONCLUSION

Este trabajo ha demostrado que la realidad aumentada, en combinación con herramientas como Unity y Vuforia, representa un avance innovador en el entrenamiento y la operatividad militar, mejorando significativamente la conciencia situacional y la eficiencia en misiones tácticas. La capacidad de la RA para superponer información digital en el entorno físico en tiempo real ha permitido optimizar la toma de decisiones, mejorar la coordinación entre unidades y reducir los riesgos operativos en escenarios críticos.

El desarrollo de una aplicación móvil basada en RA ha evidenciado su potencial como una herramienta flexible y adaptable a distintos entornos operacionales, desde la defensa hasta sectores como la salud, la educación y el turismo. En el contexto militar, la integración de Image Targets y Multi-Targets ha facilitado la identificación de equipamiento, personal y rutas tácticas, fortaleciendo la formación del personal y permitiendo entrenamientos más dinámicos, realistas y efectivos.

Además, este trabajo introduce un enfoque innovador en la forma en que se combinan entornos físicos y virtuales para el entrenamiento y la planificación de misiones. La posibilidad de simular escenarios de combate con un alto nivel de realismo, sin exponer a los soldados a riesgos innecesarios, representa una revolución en los métodos de preparación militar.

Contribución e innovación tecnológica

Más allá de su aplicación en defensa, este estudio abre nuevas oportunidades para la implementación de la RA en sectores estratégicos. La flexibilidad de plataformas como Unity y Vuforia permite seguir explorando mejoras tecnológicas en distintas áreas, tales como:

- **Optimización del rendimiento en dispositivos móviles**, asegurando un uso prolongado y eficiente en campo.
- **Desarrollo de interfaces más intuitivas y ergonómicas**, reduciendo la sobrecarga cognitiva y facilitando la adopción por parte del personal militar.
- **Integración con inteligencia artificial y big data**, permitiendo el análisis predictivo de situaciones y la generación de estrategias adaptativas.
- **Uso de sensores avanzados y redes 5G** para mejorar la precisión y la capacidad de procesamiento en tiempo real.
- **Aplicaciones en operaciones conjuntas y multinacionales**, facilitando la interoperabilidad entre distintas fuerzas y unidades.

El futuro de la realidad aumentada en el ámbito militar

La realidad aumentada está redefiniendo la forma en que los militares entrenan, planifican y ejecutan misiones, proporcionando herramientas avanzadas para visualizar, analizar y tomar decisiones con una rapidez y precisión sin precedentes. En un entorno cada vez más dinámico y tecnológico, la adopción de soluciones inmersivas permitirá maximizar la eficiencia operativa y mejorar la seguridad en el campo de batalla.

Este trabajo representa un primer paso hacia la evolución de la conciencia situacional mediante RA, demostrando que la innovación tecnológica no solo optimiza los métodos actuales, sino que abre la puerta a una nueva era en la estrategia y la defensa. A medida que esta tecnología continúe avanzando, se espera que sus aplicaciones rompan barreras y transformen los sectores más críticos de la sociedad, consolidándose como un pilar clave en las operaciones del futuro.

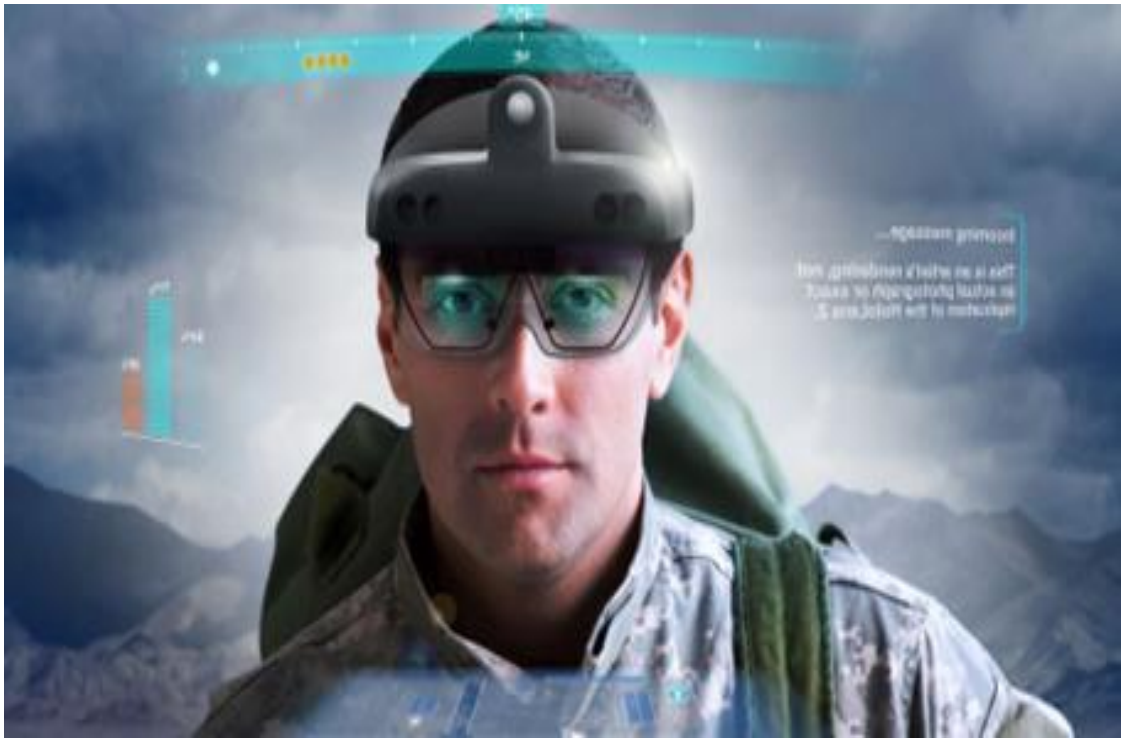


Imagen extraida del siguiente link : https://www.grupoftp.com/noticias/ivas-las-gafas-de-realidad-aumentada-militares-basadas-en-las-hololens-2-de-microsoft/

1. Creación de Proyectos de Realidad Aumentada en Unity

Unity Technologies. (n.d.). *Guía para el desarrollo de aplicaciones de realidad aumentada con Unity.* Unity Learn. Recuperado el [fecha de consulta], de https://learn.unity.com

Resumen del contenido:
Este recurso oficial de Unity ofrece una guía paso a paso para crear proyectos de realidad aumentada (RA) desde cero. Cubre desde la configuración inicial del entorno de desarrollo en Unity, la integración de elementos de RA con el SDK de Vuforia y la optimización del proyecto para dispositivos Android e iOS. Incluye ejemplos prácticos y una introducción al uso de **C#** para personalizar las interacciones.

2. Configuración de Image Targets en Unity con Vuforia

Vuforia Developer. (n.d.). *Cómo implementar Image Targets en aplicaciones de RA.* Vuforia Developer Portal. Recuperado el [fecha de consulta], de https://developer.vuforia.com

Resumen del contenido:
Este tutorial técnico del portal de desarrolladores de Vuforia explica cómo configurar y utilizar **Image Targets** en proyectos de RA. Detalla los pasos para subir imágenes al portal de Vuforia, generar bases de datos de objetivos, importar estas bases a Unity y programar interacciones con los targets. También incluye recomendaciones para optimizar el rendimiento en dispositivos móviles.

3. Tutorial Adicional sobre Desarrollo Avanzado de RA

Referencia:
Coursera. (n.d.). *Augmented Reality Development with Unity: A Practical Guide.* Coursera. Recuperado el [fecha de consulta], de https://www.coursera.org

Resumen del contenido:
Este curso práctico en línea, disponible en Coursera, cubre el desarrollo de aplicaciones de realidad aumentada utilizando Unity. Además de los fundamentos, aborda temas avanzados como la optimización de gráficos 3D, la integración de múltiples targets y el despliegue de aplicaciones en entornos reales. Este recurso es útil para quienes buscan perfeccionar habilidades avanzadas en RA.

4. Documentación Oficial de Unity y Vuforia

a. Unity:

Unity Technologies. (n.d.). *Unity User Manual (2020.3 LTS): Guía del usuario.* Unity Documentation. Recuperado el [fecha de consulta], de https://docs.unity3d.com

Descripción:
El manual del usuario de Unity incluye secciones específicas sobre la integración de plugins, optimización de proyectos de RA y ejemplos de cómo crear experiencias inmersivas interactivas.

b. Vuforia:

Vuforia Developer. (n.d.). *Base de conocimientos de Vuforia Engine.* Vuforia Developer Portal. Recuperado el [fecha de consulta], de https://developer.vuforia.com

Descripción:
La documentación oficial de Vuforia ofrece guías técnicas sobre el reconocimiento de objetos, la configuración de Multi-Targets y las mejores prácticas para integrar RA en proyectos móviles. Incluye tutoriales para principiantes y desarrolladores avanzados.

Normas técnicas y manuales

1. Unity Technologies. (2023). *Unity User Manual (2020.3 LTS).* Unity Documentation. Recuperado de https://docs.unity3d.com
   Manual oficial que cubre las funcionalidades de Unity, incluyendo la integración con Vuforia y las mejores prácticas para el desarrollo de proyectos de RA.
2. Vuforia Developer Guide. (2023). *Vuforia SDK Documentation.* Vuforia Developer Portal. Recuperado de https://developer.vuforia.com
   Guía técnica para implementar Vuforia en proyectos de RA, incluyendo el uso de Image Targets y Multi-Targets.
3. International Organization for Standardization. (2019). *ISO 9241-210: Ergonomía de la interacción persona-sistema: Diseño centrado en el usuario.*
   Norma que detalla principios de diseño centrados en el usuario, esenciales para desarrollar interfaces efectivas en aplicaciones de RA.

Bibliografia academica:

Azuma Ronald T. (1997). A Survey of Augmented Reality",Presence: Teleoperators and Virtual Environments, 6(4).

Caudell, T. P., & Mizell, D. W. (1992). Augmented reality: An application of heads-up display technology to manual manufacturing processes. *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences (HICSS),* (1). Retrieved from https://doi.org/10.1109/HICSS.1992.183317

Endsley, M. R. (1995). Toward a theory of situation awareness in dynamic systems. Human Factors.

Zhou, F., Duh, H. B.-L., & Billinghurst, M. (2008). Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR. Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR).

Bibliografía complementaria

1. Craig, A. B. (2013). *Understanding Augmented Reality: Concepts and Applications*. Morgan Kaufmann.
   Este libro explica en profundidad los fundamentos técnicos y prácticos de la realidad aumentada, destacando sus aplicaciones en distintos sectores, incluyendo educación, salud y defensa.
2. Billinghurst, M., Clark, A., & Lee, G. (2015). *Augmented Reality: Principles & Practice*. Addison-Wesley.
   Proporciona una guía técnica detallada sobre cómo crear aplicaciones de RA, incluyendo conceptos avanzados y su integración con herramientas como Unity y Vuforia.
3. Furht, B. (2011). *Handbook of Augmented Reality*. Springer.
   Este manual ofrece una visión detallada sobre las tecnologías emergentes de RA, sus desafíos técnicos y aplicaciones prácticas.
4. Jerald, J. (2015). *The VR Book: Human-Centered Design for Virtual Reality*. Morgan & Claypool.
   Aunque centrado en la realidad virtual, este libro aborda principios de diseño centrado en el usuario que son directamente aplicables a la RA y al desarrollo de interfaces intuitivas.