UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

GESTIÓN DE UN SISTEMA DE REALIDAD AUMENTADA MEDIANTE MOVIMIENTO DE LA MANO

Borja Cleries Rodríguez

Tutor: José Manuel Mossi García

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

Curso 2018-19

Valencia, 10 de septiembre de 2019

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

VLC/ CAMPUS VALENCIA, INTERNATIONAL CAMPUS OF EXCELLENCE

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

Resumen

En este trabajo final de grado se ha abordado el estudio y desarrollo de una interfaz de Realidad Aumentada presentada sobre un Head-Up Display, dentro de un sistema para proporcionar a los agentes de bomberos la visualización de imágenes térmicas a través de una cámara infrarroja, controlado mediante gestos de los dedos.

Se recopila el proceso realizado desde la búsqueda de información acerca de la tecnología a utilizar hasta los detalles del desarrollo de la interfaz. Se marcan las pautas principales de investigación de cara a una mejora del programa capaz de gestionar el sistema al completo.

Además, también se realiza un estudio de alternativas de ordenadores de placa reducida compatible con el uso de dicha interfaz, dado que este tipo de hardware será el núcleo principal del sistema.

Resum

En aquest treball final de grau s'ha abordat l'estudi i desenvolupament d'una interfície de Realitat Augmentada presentada sota un Head-Up Display, dins d'un sistema per a proporcionar als agents de bombers la visualització d'imatges tèrmiques a través d'una cámara infraroja, controlat mitjançant gestos dels dits.

Es recopila el procés realitzat des de la recerca d'informació sobre la tecnología utilitzada fins els detalles del desenvolupament de la interfície. Es marquen les pautes principals d'investigació de cara a una millora del programa capaç de gestionar el sistema al complet.

A més a més, també es realitza un estudi d'alternatives d'ordinadors de placa reduïda compatible amb l'ús d'aquesta interfície, donat a que aquest tipus de hardware será el nucli principal del sistema.

Abstract

In this final bachelor project, it has been carried out the study and development of an Augmented Reality interface presented on a Head-Up Display, within a system to provide firefighters with the visualization of thermal images through an infrared camera, controlled by finger gestures.

Secondly, the process is collected from the information search about the technology to be used to the details of the interface development. The main research guidelines are set for an improvement of the program capable of managing the entire system.

In addition, a study of alternatives of SBC computers compatible with the use of this interface is also carried out, since this type of hardware will be the main core of the system.

Índice

Índice de Ilustraciones

Índice de Tablas

Capítulo 1. Introducción

En este proyecto se busca diseñar un avanzado sistema de visualización para emergencias en el cual la máscara y el casco del cuerpo de bomberos incorpora una cámara térmica infrarroja, apoyada sobre una interfaz basada en realidad aumentada y controlada mediante gestos de las manos.

Dentro del equipamiento del personal se dispone de cámaras termográficas de gran ayuda, como la FLIR K45 (Véase *Ilustración 1*), pero demasiado voluminosas y que dificultan el trabajo individual, siendo necesaria la colaboración de más trabajadores para una única acción. Es por ello, que se han de encontrar nuevas herramientas capaces de mejorar el trabajo en situaciones extremas de incendios.

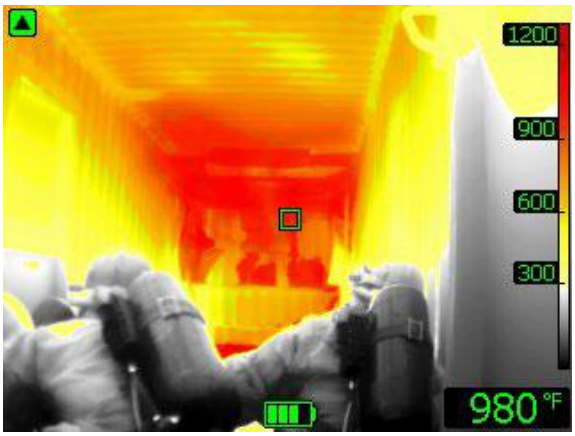

Ilustración 1. Imagen de una cámara termográfica FLIR K45

Dentro del casco del bombero también se incorporará un sistema similar a las gafas inteligentes EPSON Moverio BT-200 (Véase *Ilustración 2*), compatibles con la Realidad Aumentada. La necesidad de interactuar con dichas gafas, más el añadido de funciones en forma de menú, mapas o temperaturas, conllevan el desarrollo de un prototipo de interfaz de usuario controlada por el movimiento de los dedos e integrada mediante acelerómetros en los guantes del agente.

Ilustración 2. Gafas EPSON Moverio BT-200

La Realidad Aumentada podría definirse como la visualización del entorno real con la incorporación de distintos elementos generados por ordenador, ya sea sonido, imágenes, señales generadas por GPS, etc. Dicha realidad es "*aumentada"* puesto que, a la percepción de la persona que está utilizando este procedimiento hay que añadirle, no solo lo que captan sus sentidos, sino

que recibe adicionalmente información generada por distintas posibilidades que comentaremos posteriormente.

Este concepto de RA tiene su origen en investigaciones que se desarrollaron principalmente en la Realidad Virtual (RV a partir de ahora), si bien Thomas Caudell, en 1992, fue quien definió el término mientras trabajaba en una aplicación de RA para Boeing. [1] En dichas investigaciones se patentó una aplicación donde un HUD se utilizaría para marcar dinámicamente la posición de un agujero mediante una flecha verde, junto con el tamaño y la profundidad de la perforación dentro del fuselaje de un avión. Si el trabajador moviese su cabeza, el texto se mantendría exactamente a la misma distancia anterior (Véase *Ilustración 3*).

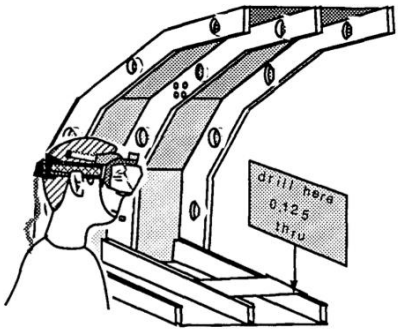

Ilustración 3. Aplicación patentada por Thomas Caudell

Centrándonos en el apartado de la interfaz, la idea principal es aportar a la persona que esté de servicio la información que sea conveniente como, por ejemplo, la temperatura del recinto, el nivel de toxicidad u oxígeno, un mapeado de la zona, etc. La creación de un Head-Up Display (HUD) ayudaría a disponer de dicha información de una forma funcional y precisa, sin requerir de ayuda externa, tal y como se utiliza desde hace años en el mundo del videojuego (Véase *Ilustración 4*) o más recientemente en el del automovilismo.

Ilustración 4. Ejemplo de Head-Up Display en videojuegos (Metroid Prime, 2002)

En último lugar, hay que añadir que el presente trabajo se centra en la gestión de la interfaz de Realidad Aumentada, siendo una parte de un proyecto más amplio y en pleno desarrollo, en el que se encuentran participando más alumnos de la escuela de forma paralela, para disponer de un producto final apto para uso tanto público como privado, competitivo y de coste lo más ajustado posible.

Capítulo 2. Objetivos

Debido a la magnitud de este proyecto, este trabajo aportará una serie de objetivos centrados en la interfaz de la Realidad Aumentada, que serán los siguientes:

- Investigación del mercado de microprocesadores. Se mostrarán las ventajas e inconvenientes de las propuestas analizadas.

- Investigación sobre la tecnología de Realidad Aumentada. Nos marcaremos dos hitos a desarrollar:
  - Estudio de las diferentes posibilidades que se podrían utilizar en el desarrollo de una aplicación de Realidad Aumentada.
  - Observar ventajas e inconvenientes de las propuestas de una forma detallada.

- Realización de pruebas sobre las propuestas anteriores.

- Planteamiento de futuras pruebas y desarrollos como continuación de este trabajo.

Capítulo 3. Estudio de Alternativas

3.1. Ordenador de placa reducida (*Single Board Computer* o *SBC)*

El componente más importante en este trabajo es la elección de un ordenador de placa reducida que se pueda incorporar a la equipación del bombero. Este microprocesador ha de ser capaz de sostener y administrar todas las operaciones y conexiones (back-end) necesarias para que la aplicación final (front-end) funcione correctamente.

Así pues, para el subsistema back-end necesitaremos un SBC con los siguientes requisitos:

- Pequeño y ligero.
- Bajo consumo.
- Altamente modificable.
- Potente.
- Versátil.

El primer problema al que nos encontramos a medida que fue avanzando el proyecto fue el que la placa Raspberry Pi 3B+, aún siendo la opción recomendada debido a su precio/rendimiento y su compatibilidad con Arduino, su sistema operativo Raspbian OS no es compatible con ningún programa que pueda desarrollar programas y aplicaciones de RA. Esto es debido a que el procesador no es de 64 bits ni tampoco tiene opción de instalar Android de forma nativa.

Esto suponía un gran contratiempo, pero a partir de este momento decidimos estudiar posibles alternativas que cumpliesen los requisitos anteriores, pero sin aumentar los gastos de forma excesiva.

A continuación, se detallarán las posibles alternativas a la placa Raspberry Pi 3B+ solventando los problemas de compatibilidad:

	Raspberry Pi 3B+	**Odroid N2**	**UDOO x86 Advanced+**
			
Procesador	Cortex-A53 @1.4GHz	Quad-Core Cortex-A73 @1.8GHz y Dual-Core Cortex-A53 @1.9GHz	Intel Celeron N3160 @2.24GHz
Memoria	1GB LPDDR2 SDRAM	2-4GB DDR4	4GB DDR3
Almacenamiento	Ranura Micro SD	Ranura Micro SD y eMMC	Ranura Micro SD, ranura SSD y 32GB eMMC

Puertos	1 Puerto HDMI 4 Puertos USB 2.0	1 Puerto HDMI 2.0 4K/60Hz 4 Puertos USB 3.0 1 Puerto USB OTG	1 Puerto HDMI 2.0 4 Puertos USB 3.0 1 Puerto USB OTG
Alimentación	Micro USB	Micro USB	Micro USB
Conexiones inalámbricas	WiFi: 2.4GHz y 5GHz 802.11.b/g/n/ac Bluetooth 4.2, BLE	Adaptadores opcionales	Adaptadores opcionales
Red	Gigabit Ethernet (300 Mbps de máximo teórico)	Gigabit Ethernet (1000 Mbps de máximo teórico)	Gigabit Ethernet (1000 Mbps de máximo teórico)
Pines de entrada	40 GPIO	40 GPIO + 7 I2S	20 GPIO + multiplexación con otras interfaces
Conectores	Cámara: CSI / Pantalla táctil: DSI	Opcionales	Compatibilidad con Arduino Leonardo
SO	Raspbian, Linux, Windows 10 IoT Core	Android, Linux	Windows 10, Android, Linux
Precio	36 €	100-120€	150-200€

Tabla 1. Comparativa de SBC

Tanto la placa Odroid N2 como la placa UDOO x86 Advanced + son recomendables, ya que al menos tienen compatibilidad con Android y son mucho más potentes que la Raspberry. Esto nos supone un punto de versatilidad superior, ya que la aplicación RA podría funcionar en múltiples sistemas de una manera mucho más sencilla, simplemente con trasladarla (portear).

Por el contrario, ambas placas no tienen integrada ninguna conexión inalámbrica (se venden por separado) y son sensiblemente más caras. Además, la comunidad de ambos productos no es tan potente como la de Raspberry y esto podría suponer un menor ritmo en el desarrollo de las aplicaciones.

En definitiva, optaríamos por la placa UDOO x86 Advanced+ por su potencia y compatibilidad con Windows 10, debido a la arquitectura de su procesador, además de por todos los requisitos detallados anteriormente. [3]

3.2. Kit de Desarrollo de Software de Realidad Aumentada

Otra de las partes clave en el desarrollo de la aplicación en el sistema de visualización por Realidad Aumentada es, precisamente, encontrar una herramienta capaz de reconocer los gestos de las manos y adaptarla de manera virtual a una interfaz cuyo entorno es real y tridimensional.

Ilustración 5. Prototipo de aplicación AR, Joseph Juhnke

Como vemos en el prototipo diseñado en 2011 por Joseph Juhnke [2] [8] (Véase *Ilustración 5*), la idea principal es que se superponga la información que necesitamos gracias a la incorporación de una interfaz que se pueda visualizar a través de unas gafas de Realidad Aumentada y controlado por los gestos de las manos. Por todo esto, se decidió trabajar con el SDK Vuforia, ya que se trata de una herramienta ampliamente conocida y compatible con muchos sistemas diferentes.

Vuforia es un kit de desarrollo de software de realidad aumentada (SDK) para dispositivos móviles que permite la creación de aplicaciones de RA. Utiliza tecnología de visión por computación para reconocer y rastrear tanto imágenes 2D como objetos 3D en tiempo real. Esto permite posicionar y orientar objetos virtuales en relación con imágenes del mundo real cuando se ven a través de la cámara. El objeto virtual rastrea tanto la orientación como la posición de la imagen para que la perspectiva de la persona sobre el objeto se corresponda con la misma perspectiva del objetivo de la imagen. Así pues, simulamos que el objeto virtual sea parte de la escena del mundo real.

Estas son las características más destacables del SDK Vuforia:

- Reconocimiento de texto e imágenes
- Rastreo: el objetivo fijado no se perderá fácilmente incluso cuando el dispositivo se mueva
- Detección y rastreo simultáneo de objetivos (Targets)

Capítulo 4. Desarrollo

4.1. Instalación de las herramientas y primeros pasos

En primer lugar, hemos de descargar todas las herramientas necesarias para poder trabajar de una forma fluida desde el inicio del desarrollo. Así pues, indicaremos cuáles hemos de descargar y luego entraremos en más detalles con la SDK de Vuforia y el IDE de Unity (Véase *Ilustración 6*) [4] [5].

- Unity Hub:
  - Descarga de la última versión de Unity compatible con Vuforia (Unity 2019.1.12f1).
  - Activación del soporte del SDK de Vuforia, cuyo nombre es Vuforia Augmented Reality Support.
  - Descarga de Android Studio, junto al SDK de Android y al JDK (Java Developer Kit) para la configuración de Android para Windows.
- Visual Studio: Para la programación en C#

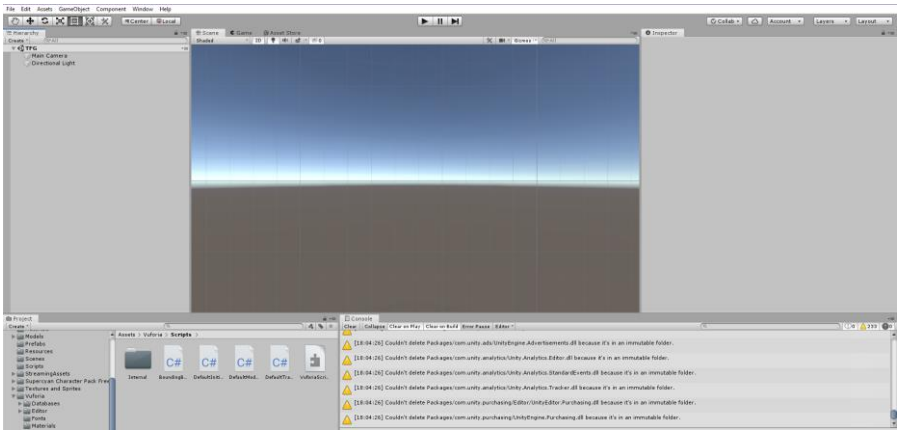

Ilustración 6. Pantalla de inicio de Unity

A continuación, se detallarán los elementos de Vuforia más importantes que serán utilizados en el proyecto:

Image Target

Se trata de la imagen que la tecnología de Vuforia es capaz de rastrear y detectar y que nos da la capacidad de realizar múltiples acciones en Unity, ya sea incorporar botones, imágenes o vídeos sobre dicha imagen. Esta imagen debe ser rica en detalles, ya que utiliza un seguimiento de puntos y, si estos son poco definidos, no será posible realizar un rastreo correcto.

Se realizan las primeras pruebas comparando una mano real con una imagen de un guante, antes de utilizar el guante de bombero con todos los elementos (giroscopios, Arduino y cableado variado) como Image Target del proyecto. Para ello, se ha de subir al Developer Manager de Vuforia (Véase *Ilustración 7*) todas las imágenes que queramos probar y/o utilizar en nuestro trabajo, creando una BBDD con todas imágenes objetivo. [6]

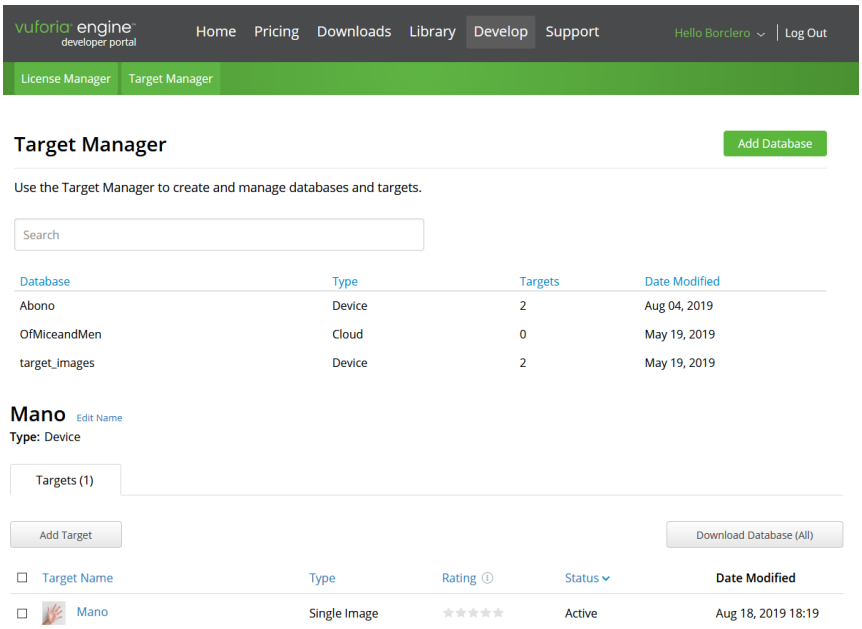

Ilustración 7. Visualización de elementos de la BBDD en Vuforia

Al realizar la primera subida con una mano real (Véase *Ilustración 8*) pudimos observar con una puntuación de 0 de 5 estrellas que la imagen generada no servía de target para el programa, teniendo una mala calidad para su detección.

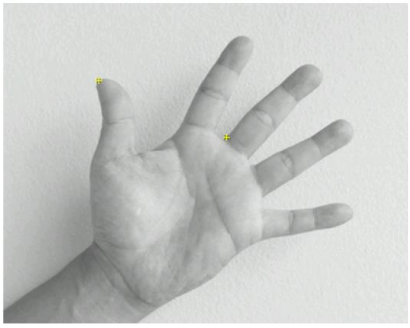

Ilustración 8. Prueba de reconocimiento de mano humana

En el caso de la copia de un guante (Véase *Ilustración 9*), la puntuación fue de 5 estrellas y, con ello, pudimos comenzar a realizar nuestras pruebas con mayores garantías. Para tenerla activa en el proyecto, debemos descargar los elementos de la BBDD (en nuestro caso, el guante) mediante el botón Download Database del Developer, importándolo como Assets de Unity.

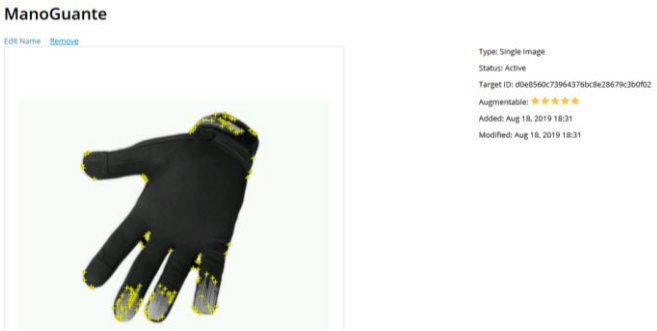

Ilustración 9. Valoración de fotocopia en guante

AR Camera

Se trata de la activación de la cámara para que nuestro proyecto pueda reproducirse en cualquier dispositivo móvil mediante la Realidad Aumentada.

Nada más abrir un nuevo proyecto, tendremos que activar la opción de Vuforia. Nos dirigiremos a la pestaña Field > Build Settings > Player Settings > XR Settings > Activar Vuforia Augmented Reality Support.

Posteriormente, es imprescindible eliminar la Main Camera que nos aparece en la Jerarquía del proyecto y sustituirla por la AR Camera de Vuforia con el click derecho del ratón (Vuforia Engine > AR Camera). Además, tenemos que configurar esta AR Camera, por lo que en la pestaña Inspector de este elemento iremos al Open Vuforia Engine configuration (Véase *Ilustración 10*) y añadiremos la licencia que obtuvimos en el License Manager al registrarnos en Vuforia, tal y como hemos explicado en el punto anterior. [7] [11]

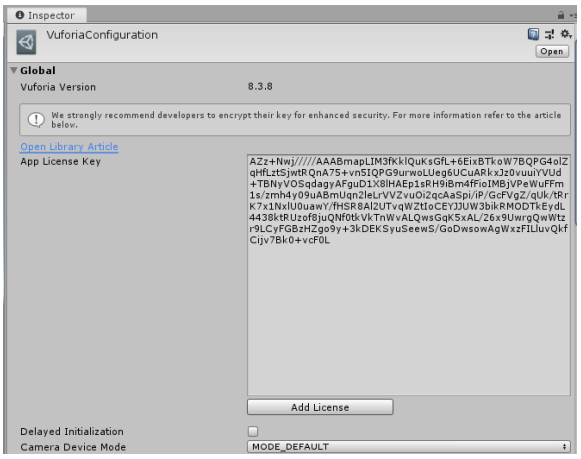

Ilustración 10. Pestaña de configuración de Vuforia en Unity

GameObject

Se trata del componente del editor de Unity más importante. Este componente puede representar, por ejemplo, modelos 3D, escenarios, efectos, cámaras o luces. Por si mismo esto no tendría ninguna funcionalidad, pero sí lo hacen como contenedores de otros elementos llamados **Components,** implementando así las propiedades concretas que queremos darle a cada Object.

Un GameObject siempre tiene activado por defecto el componente “Transform”, que define su posición, rotación y escala y, por tanto, no es posible eliminarlo. El resto de "components” que nos permiten aplicarle su funcionalidad al objeto en cuestión pueden ser añadidos desde el menú habilitado en el editor o desde un script, entre los que se incluyen por defecto elementos como formas 3D y 2D, elementos gráficos como un texto o botones, etc.

Prefabs:
Se trata de objetos reutilizables que permiten crear, configurar y guardar distintos GameObjects con sus distintos Components.

Una vez se ha configurado el entorno de desarrollo y se han detallado los elementos más importantes tanto de Unity como de Vuforia, tenemos que configurar el dispositivo móvil donde vamos a probar la aplicación, activando la depuración USB en dicho dispositivo. Esta configuración puede cambiar en cada dispositivo, por lo que se recomienda realizar una búsqueda en particular y activarlo tal y como se indique.

Por último, la Ilustración 11 sería la configuración inicial en el desarrollo de la aplicación, con el Image Target del guante y la AR Camera ya incluidos en el editor.

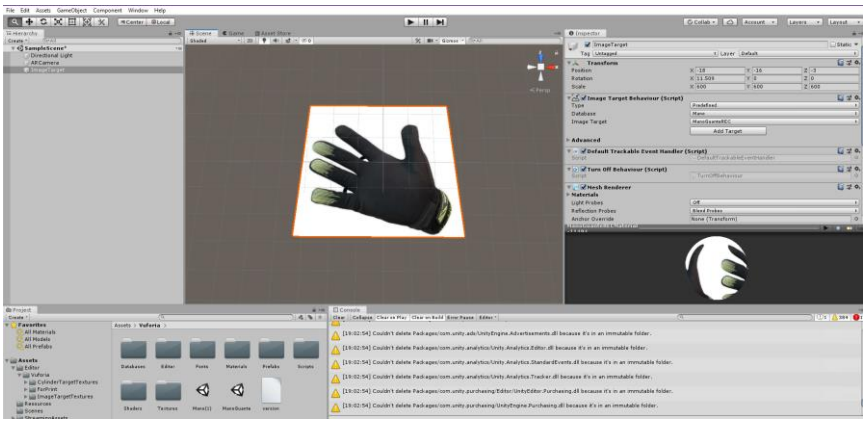

Ilustración 11. Introducción del Image Target en la pestaña de Scene en Unity

4.2. Banco de pruebas

Después de las instalaciones de las herramientas y configuraciones de todos los entornos necesarios para el inicio del desarrollo, comenzamos a realizar las primeras pruebas de diseño.

La idea básica general de este trabajo enlazada con el proyecto global es obtener una aplicación que muestre por pantalla una serie de opciones disponibles que serán controladas mediante los gestos de los dedos de una mano (Véase *Ilustración 12*). Estas opciones estarán disponibles visualmente en todo momento cada vez que el guante entre dentro del campo visual de las gafas de realidad aumentada, por lo que todas las pruebas desarrolladas en este trabajo van dirigidas a encontrar las mejores posibilidades de control y visión, sin poder contar todavía con la unión de esta interfaz con los gestos.

Sin entrar en más detalles de cómo se enlazaría en un futuro la batería de gestos con la interfaz, ya que todavía se encuentra en desarrollo, la idea sería seleccionar una opción dentro de las disponibles juntando la yema del dedo pulgar con el dedo que manejaría dicha opción. Una vez activada esa opción, se manejarían distintas posibilidades como realizar un scroll simulando un roce entre el dedo pulgar e índice en repetidas ocasiones, aumentar un mapa o una imagen tocando el pulgar con el corazón más de 2 veces, etc.

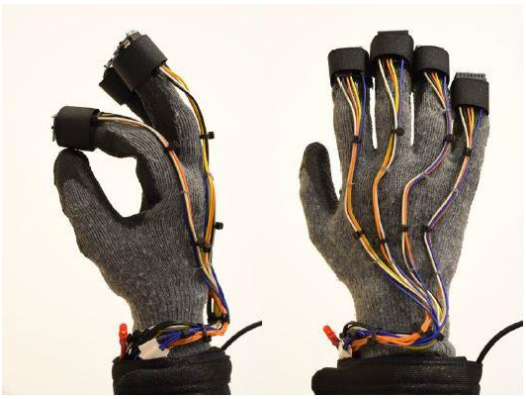

Ilustración 12. Prototipo de guante para reconocimiento de gestos

Estas características, como ya hemos comentado, aún no están disponibles por el momento, y por este motivo, nos vamos a centrar en las pruebas que nos permitan en un futuro poder mostrar todo lo anterior mediante una aplicación de RA, y descartar las que sean menos recomendables.

4.2.1. Reconocimiento en Image Target

En primer lugar, y basándonos en la principal premisa del trabajo, configuramos distintas opciones de texto junto al Image Target. Para ello, añadimos como 3D Object > 3D Text (Véase *Ilustración 13*) dentro de la jerarquía del Target las opciones de Temperatura, Nivel de Toxicidad, Mapa, Radio y Menú.

En este punto es muy importante sabernos manejar correctamente con el apartado de Scene en Unity, ya que tendremos que igualar los textos a la altura de nuestro target en tres dimensiones. Tendremos que utilizar las dimensiones X,Y,Z de la escena para posicionar los textos, tal y como ve se en la siguiente imagen (Véase *Ilustración 14*).

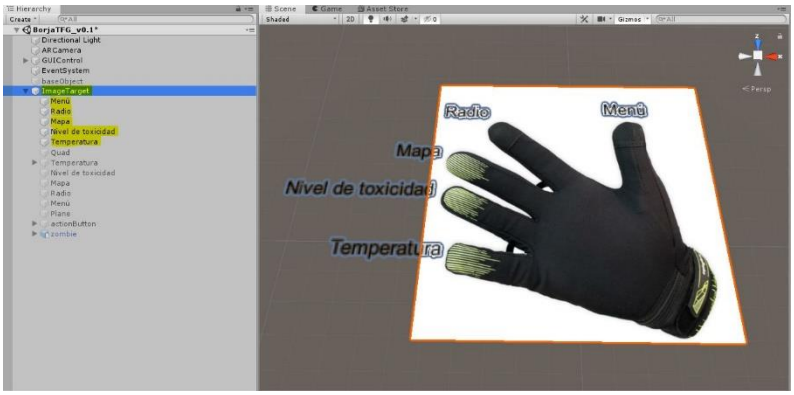

Ilustración 13. Introducción de texto junto al Image Target

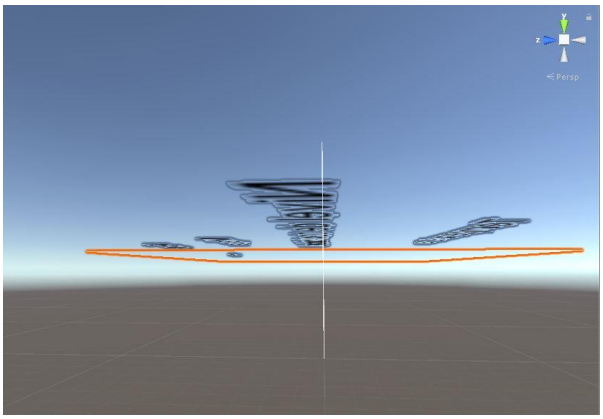

Ilustración 14. Cambio de perspectiva en texto 3D

Por lo tanto, ya estaríamos en disposición de realizar la primera prueba desde el dispositivo móvil, dirigiéndonos al apartado del editor Field > Build Settings > Build, habiendo hecho anteriormente el Switch Platform a Android y activando nuevamente el Vuforia Augmented Reality Support para esta nueva plataforma. Más tarde, introduciremos la APK (Android Application Package) producida por Unity en nuestro procesador, habilitando la opción como desarrollador que comentamos en el apartado 5.1.

Con todo esto, obtenemos la primera versión del diseño de la aplicación (Véase *Ilustración 15*), en el que se puede comprobar que el tracking es correcto y nos permite visualizar las opciones que consideramos oportunas para su presentación.

Ilustración 15. Captura de la aplicación AR (I)

Continuamos haciendo pruebas en esta versión y sustituimos las letras 3D por imágenes. Tanto las imágenes como el resto de componentes no pueden incorporarse automáticamente al proyecto, pero sí pueden hacerlo importándolos como Assets del proyecto. Más tarde, hemos de configurarlas como Sprites 2D para que el proyecto las pueda reconocer, tal y como aparece a continuación (Véase *Ilustración 16*).

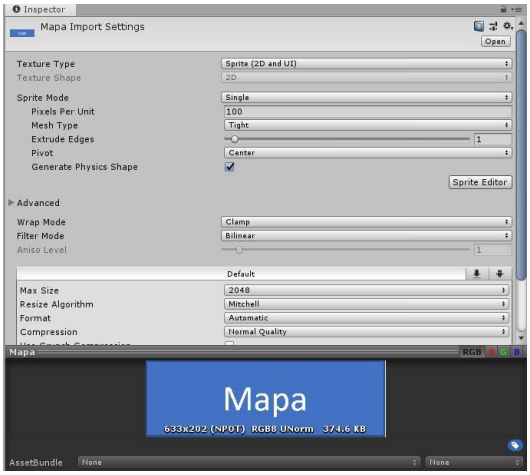

Ilustración 16. Menú Inspector de Unity

4.2.2. Controlador táctil GUI

La forma más efectiva de poder controlar la aplicación táctilmente es utilizar herramientas GUI (Interfaz Gráfica de Usuario) que están disponibles en Unity mediante Canvas.

La GUI es el entorno en el que un usuario puede interactuar con una aplicación o con un Sistema Operativo, basándose en distintos elementos visuales, como por ejemplo, conjuntos de imágenes o iconos.

Para aplicar esta interfaz en el editor de Unity debemos utilizar el GameObject Canvas (Véase *Ilustración 17*), que es el área donde todos los elementos UI (Interfaz de usuario) deben estar presentes. En el menú GameObject > UI se creará un nuevo Canvas si no hay uno presente anteriormente, donde a partir de ahí se podrán crear elementos hijos.

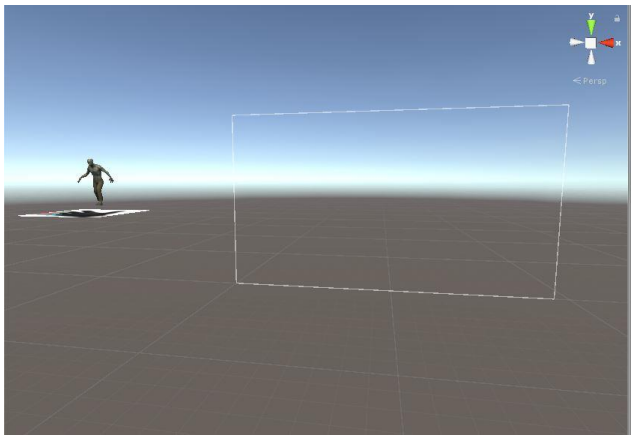

Ilustración 17. Introducción del elemento GUI en la escena de Unity

Una de las opciones propuestas en la creación de la interfaz es habilitar una serie de botones simulando las necesidades que podría tener el bombero en una situación de emergencia pero sin la inclusión, todavía, del reconocimiento de gestos. Así pues, los elementos a desarrollar serán cuatro botones para poder manejar cualquier objeto libremente por la interfaz (izquierda, derecha, arriba y abajo) y dos botones para poder aumentar y reducir cualquier objeto junto a un panel HUD.

Para incorporar estos objetos, primero hemos de importar como Assets del proyecto imágenes que queramos incorporar como Botones en la secuencia UI > Button de la jerarquía del proyecto. Una vez importados hay que añadirlos como objetos 2D/Sprites (Véase *Ilustración 18*).

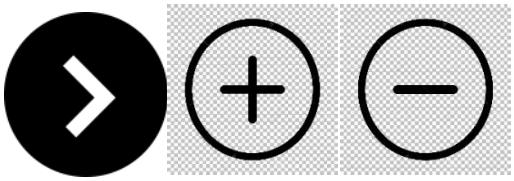

Ilustración 18. Botones utilizados en el GUI de Unity

Así pues, aparecerán como objetos hijos del Canvas, llamado en nuestro caso GUIControl. Al querer incorporar un panel haciendo de HUD como vimos en el primer punto del trabajo, el Panel será hijo del GUIControl y los botones serán hijos a su vez del Panel (Véase *Ilustración 19*).

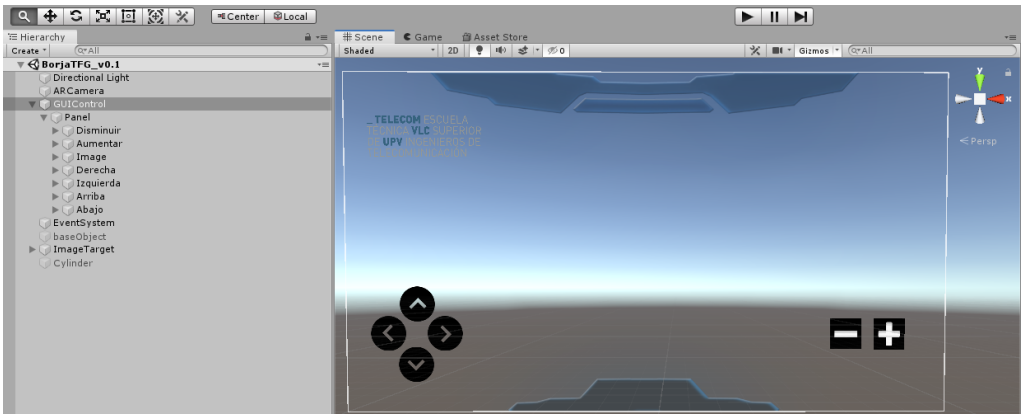

Ilustración 19. Introducción del panel HUD y los botones GUI en la escena de Unity

Incorporados los botones y el panel HUD en la escena del proyecto, debemos crear un script en C# con Visual Studio que pueda realizar los movimientos de los botones en un elemento cualquiera, tal y como hemos descrito en este punto:

- Mover hacia la derecha
- Mover hacia la izquierda
- Mover hacia arriba
- Mover hacia abajo
- Aumentar la escala
- Disminuir la escala

El primer paso que realizaremos para poder trabajar sobre el script es crearlo desde la ventana de Proyecto en Unity con el nombre ControlarObjeto, en la ruta Create > C# Script.

En cualquier código que se vaya a utilizar en Unity, la clase MonoBehaviour ha de utilizarse en todos los casos, ya que se trata de la clase base de la que deriva cada script. Para mover un objeto cualquiera utilizaremos la API (Application Programming Interface) de Unity Transform.Translate, ya que moverá dicho objeto en la dirección y traslación que deseemos.
En el caso de ampliar o disminuir la escala del objeto, usaremos el elemento transform.localScale.x y o z dependiendo en la coordenada que queramos realizar la acción.
Por último, el tiempo, el cual estará el objeto desplazándose vendrá dado por el Time.deltaTime, proporcionándonos en segundos el tiempo entre la imagen (frame) actual y el anterior, siendo esta API únicamente de lectura. [9] [10]

Una vez realizado el script, se añadirá al inspector de Unity del objeto que queramos desplazar o cambiar la escala, en nuestro caso, un cilindro (Véase *Ilustración 20*).

Ilustración 20. Introducción del script en la pestaña de Inspector en Unity

Para poder enlazar los botones con el script tenemos que asignarle un Sistema de Eventos (Event System), que es el encargado de enviar eventos a objetos, basado en los diferentes inputs disponibles, ya sea teclado, ratón, pantalla táctil, etc. En cada sistema tenemos la opción de activar o desactivar los eventos según corresponda mediante los Event Trigger, aplicándole la función del script que necesitemos.

Para todos los botones hemos de asignarle los siguientes triggers, para que el uso de la pantalla táctil sea correcto en el dispositivo móvil:

- Pointer Down: Pulsar el botón.
- Pointer Up: Dejar de pulsar el botón.
- Pointer Enter: Arrastrar el botón desde fuera hacia dentro.
- Pointer Exit: Arrastrar el botón desde dentro hacia fuera.

Por último, solo nos quedará asignarle al GameObject (cilindro) la función adecuada en cada Trigger, poniendo como ejemplo los asignados al botón derecho (Véase *Ilustración 21*). El resto de botones se añadirán análogamente.

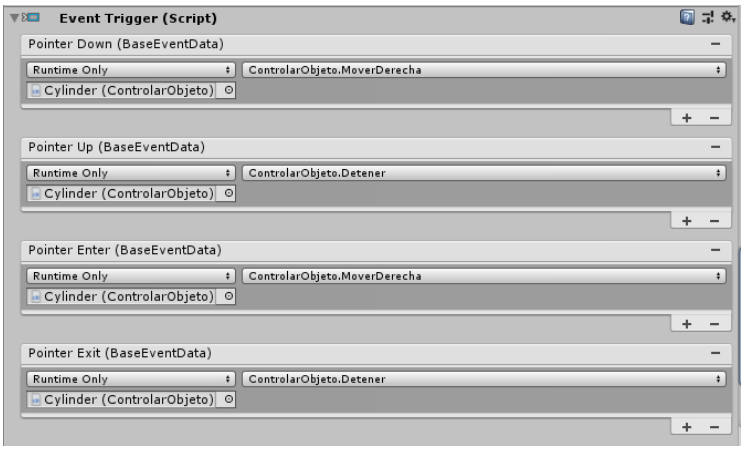

Ilustración 21. Event Trigger en Unity

A continuación, se muestra la aplicación con el controlador táctil, el aumento y disminución de escala, el panel HUD y la imagen de la escuela de Teleco UPV en la siguiente captura (Véase *Ilustración 22*), junto a otra serie de imágenes con la incorporación del cilindro en diferentes estados (Véase *Ilustración 23*).

Ilustración 22. Captura de la aplicación AR (II)

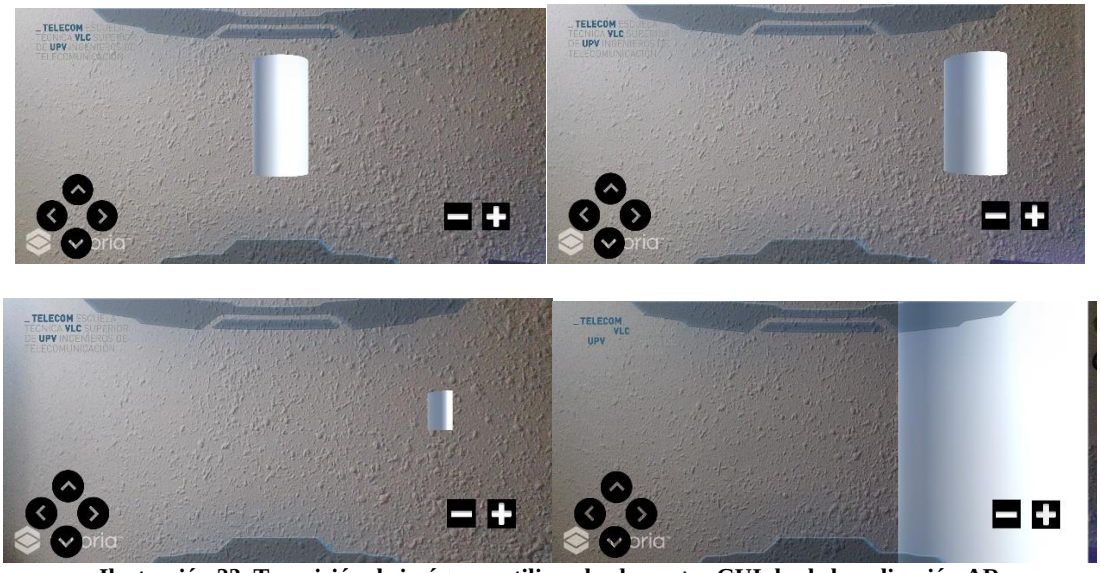

Ilustración 23. Transición de imágenes utilizando elementos GUI desde la aplicación AR

Con todo esto, una primera versión de la aplicación tendría el siguiente aspecto (Véase *Ilustración 24*), mostrando las diferentes opciones en los dedos de las manos y lo explicado en este punto.

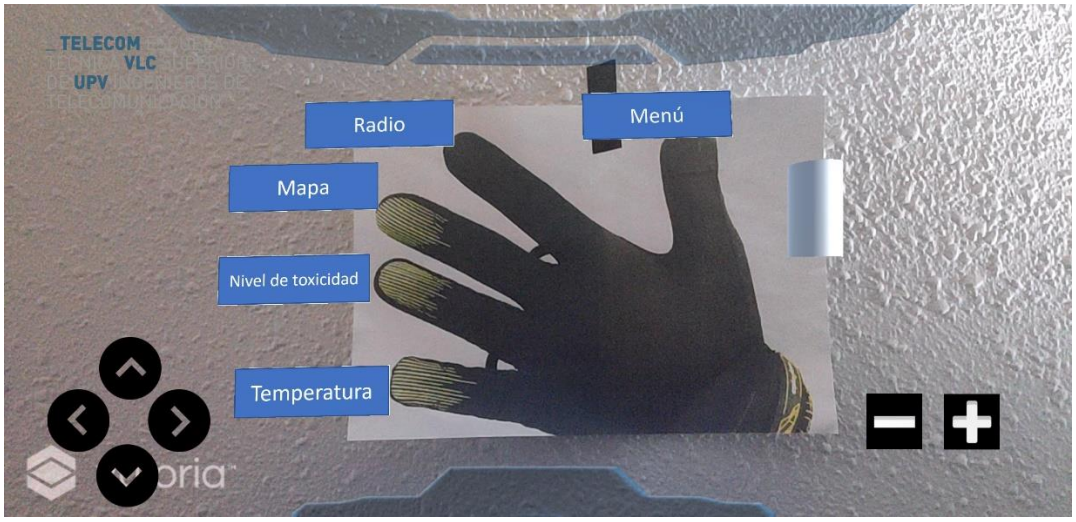

Ilustración 24. Captura de la aplicación AR (III)

Capítulo 5. Conclusiones y líneas futuras de trabajo

En este trabajo se ha conseguido un primer prototipo de aplicación funcional de Realidad Aumentada, pudiendo disponer visualmente de elementos que sean realmente útiles para mejorar en la resolución de determinadas acciones.

La premisa más importante que nos propusimos se ha logrado, ya que aparecen representadas diferentes opciones en el guante, y éstas son capaces de desplazarse al mismo tiempo que lo hace el guante.

Una de las potenciales mejoras en la aplicación sería añadir distintos menús por cada una de las opciones disponibles por defecto, es decir, si seleccionamos la opción Radio, que dicha opción sea capaz de mostrar diferentes radios al mismo tiempo, y que seamos capaces de desplazarnos por cada una de ellas mediante el toque del dedo pulgar con la palma de la mano. En Unity existen opciones para poder realizar lo que comentado.

Llegados a este punto, se bifurcan dos líneas de trabajo futuras que se han de ir desarrollando en paralelo:

1. Enlazar la aplicación AR con los siguientes trabajos:
    - Cambio de microprocesador: de Raspberry Pi a UDOO x86
    - Procesamiento del reconocimiento de gestos, siendo compatible con C#. Este será el punto más delicado, ya que todos los gestos de los dedos de la manos monitorizados y entrenados anteriormente han de verse fielmente representados por la aplicación.
    - Adaptar como cámara AR la cámara térmica, para que se pueda visualizar a través de las gafas.

2. Mejora del prototipo:
    - Realización de un menú con opciones.
    - Incorporación de nuevos elementos.

Dentro de los elementos no incorporados en este trabajo, destacaremos las siguientes opciones:

- Botones virtuales
- Reproducción de vídeo

5.1. Botones virtuales

Una de las alternativas que se plantearon fue la de dar utilidad a los textos/imágenes de cada una de las opciones, en concreto, pulsando virtualmente estos elementos para que la aplicación procesara dichas acciones, teniendo colocado el guante enfrente del dispositivo móvil, marcar la opción con la otra mano y que apareciera lo indicado por pantalla.

La investigación generada por este hecho motivó el estudio de los **Virtual Buttons (botones virtuales)**. [9] [10] Estos botones sí nos proporcionan mecanismos útiles para hacer que los Image Target sean interactivos, en contra de los elementos fijos del punto 5.1. En versiones anteriores de Unity sí existían Virtual Buttons por defecto dentro de la pestaña Prefabs, pero en estos momentos se ha de crear a partir de cada Image Target. Para ello, tendremos que dirigirnos al Inspector de la imagen, abrir el desplegado de Avanzado y pulsar en la opción de Add Virtual Button, incluyendo numerosas opciones por defecto, como es el Virtual Button Behaviour (script de comportamiento en el que podemos modificar el nivel de sensibilidad), añadir texturas y materiales, utilizar manualmente otros scripts que permitan realizar acciones avanzadas, etc.

En este caso nos centraremos en crear un primer script para probar la utilización de cualquier script en C# con Visual Studio.

En esta versión mostraremos la animación de un elemento 3D cada vez que pulsemos virtualmente el botón, que se encontrará en uno de los dedos del guante. Añadiremos un nuevo script sencillo como componente del Image Target, nombrado como vBScript (Anexo X). Cada vez que se pulse el botón se animará el modelo 3D y cuando se deje de pulsar parará automáticamente.

El problema en el uso de estos elementos reside en la poca fiabilidad al virtualizar opciones que han de ser muchos más precisas, aunque sí acabe realizando la interacción tal y como se muestra en la siguiente secuencia de imágenes (Véase *Ilustración 25)*. Una de las motivaciones futuras será la de conseguir mayor fiabilidad en la pulsación de los botones.

Ilustración 25. Transición de imágenes utilizando Botones Virtuales desde la aplicación AR

5.2. Reproducción de vídeo

Para complementar el estudio de las alternativas disponibles, también buscamos diferentes opciones para poder reproducir imágenes en tiempo real y/o vídeos que nos pudieran ayudar en un futuro. Para este trabajo, la reproducción de vídeos no era una prioridad, pero sí puede ser un elemento importante a incorporar ya sea, por ejemplo, imágenes en otros puntos conflictivos, señales de vídeo que nos envíen desde Central u otras aportaciones. En este caso, nos centraremos en la incorporación de vídeo, que nos ayudará a manejar clases específicas de Vuforia.

La clase ITrackableEventHandler nos permite manejar cambios en la interfaz para aquellos estados que puedan ser rastreados (tracking). Para poder editar e incorporar nuevos elementos sobre esta clase, disponemos por defecto del script DefaultTrackableEventHandler, que nos ayudará, tal y como hemos comentado, a incorporar un vídeo en nuestro programa.

En el script tendremos que añadir el siguiente código, que hará referencia a la clase ITrackableEventHandler en primera instancia, y más tarde al inicio y la pausa del vídeo. Además, será necesario incluir dicho script en el Image Target, de la misma forma que lo realizamos en vbScript y en ControlarObjeto.

```
public class DefaultTrackableEventHandler : MonoBehaviour, ITrackableEventHandler
{
       public UnityEngine.Video.VideoPlayer videoPlayer;


protected virtual void OnTrackingFound()
    {
        if (mTrackableBehaviour)
        {
            videoPlayer.Play();

protected virtual void OnTrackingLost()
    {
        if (mTrackableBehaviour)
        {
            videoPlayer.Stop();
            videoPlayer.Pause();
```

Cuando tengamos esto, tendremos que añadir el elemento Quad para simular una superficie plana en la cual aparecerá el vídeo, siguiendo la ruta 3D Object > Quad. Al añadir tanto el código como el Quad, nos aparecerá la opción de añadir un elemento de vídeo en el Video Player del Inspector del Image Target, mostrándose a continuación (Véase *Ilustración 26*):

Ilustración 26. Introducción del script en la pestaña de Inspector en Unity (II)

Teniendo todo los anterior creado, desplazaremos el elemento Quad al lugar donde más nos convenga como, por ejemplo, el centro de la mano, simulando que podamos visualizar en un futuro información que nos sea útil. Aplicando el botón play de la escena, vemos que el vídeo se reproduce correctamente en Unity (Véase *Ilustración 27*).

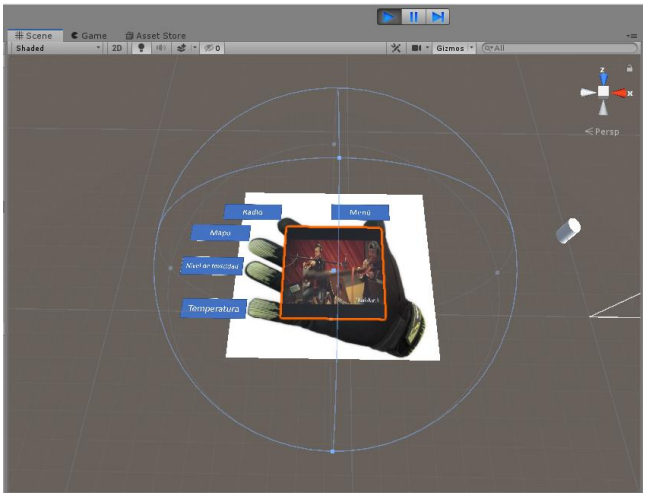

Ilustración 27. Introducción del Vídeo en la pestaña Scene en Unity

Lamentablemente, al importar el proyecto al dispositivo móvil y tratar de reproducirlo cuando reconoce el Image Target, se bloquea automáticamente y se cierra la aplicación, impidiéndonos realizar cualquier otra función con el reproductor de vídeo activado. Esto puede deberse a múltiples factores, ya sea por códec de vídeo, incompatibilidades entre Unity y Vuforia u otros desconocidos. Hemos estado probando varias alternativas de compresión de vídeo, pero el resultado no ha sido el esperado en ninguno de los casos.

Capítulo 6. Bibliografía

[1] Caudell, Thomas P.; Mizell, David W. (1992). *Augmented Reality: An Application of Heads-Up Display Technology to Manual Manufacturing Processes.*

[2] Juhnke, Joseph (2011). *The future of firefighting - A HMD-AR UI concept for first responders.* [En línea] Disponible en: https://www.youtube.com/watch?v=QBAnr2gQTH0

[3] UDOO x86. [En línea] Disponible en: https://www.udoo.org/udoo-x86/

[4] UNITY, <<Manual de Usuario de Unity 2019.1>> 2019. [En línea] Disponible en: https://docs.unity3d.com/Manual/UnityManual.html

[5] VUFORIA, <<Vuforia Developer Library>> 2019. [En línea] Disponible en: h3ttps://library.vuforia.com/getting-started/overview.html

[6] VUFORIA, <<Vuforia Engine Developer Portal>> 2019. [En línea] Disponible en: https://library.vuforia.com/getting-started/overview.html

[7] Greene, Jacob W. (2018). *Creating Mobile Augmented Reality Experiences in Unity*. [En línea] Disponible en: https://programminghistorian.org/en/lessons/creating-mobile-augmented-reality-experiences-in-unity

[8] École polytechnique fédérale de Lausanne (EPFL) (2016). *Firefighters: Augmented Reality in a breathing mask.* [En línea] Disponible en: https://www.face2fire.com/realidad-aumentada-para-bomberos/

[9] Stackoverflow. [En línea] Disponible en: https://stackoverflow.com/

[10] GitHub. [En línea] Disponible en: https://github.com/

[11] García González, Javier (2018). *Tutorial básico AR Unity3D y Vuforia*. [En línea] Disponible en: https://croquetastecnologicas.com/2018/08/06/tutorial-basico-ar-unity3d-y-vuforia/

Agradecimientos

En primer lugar, quisiera agradecer a mi tutor del trabajo, José Manuel Mossi, la dedicación y el trato exquisito que he tenido a lo largo de estos meses, además de su total comprensión en momentos en los que el escaso tiempo y los problemas de diversa índole eran pequeñas piedras en el camino.

También quiero agradecer a mi familia, quién me ha tenido que soportar (un poco) durante estos meses y, en especial, estas últimas semanas. Poder decir que he finalizado el Grado, me reconforta de una forma de la que en estos momentos no soy consciente.

Por último, agradecer a toda la gente que ha estado junto a mí, amigos y compañeros de trabajo.

Sin el apoyo, fuerza y ganas de todos estoy seguro de que no lo habría podido conseguir de la misma manera.

Gracias de nuevo.

Capítulo 7. Anexos

7.1 Script 1: vbScript

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

public class vbScript : MonoBehaviour, IVirtualButtonEventHandler {

    private GameObject vbButtonObject;
    private GameObject zombie;

    void Start()
    {

        vbButtonObject = GameObject.Find("actionButton");
        zombie = GameObject.Find("zombie");

vbButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHandler(this);
    }


    public void OnButtonPressed(VirtualButtonBehaviour vb)
    {

        zombie.GetComponent<Animation>().Play();
    }

    public void OnButtonReleased(VirtualButtonBehaviour vb)
    {
        zombie.GetComponent<Animation>().Stop();
    }

}
```

7.2 Script 2: ControlarObjeto

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

public class ControlarObjeto : MonoBehaviour
{
    // Configuración velocidad
    public float Velocidad = 5.0F;

    private bool right = false;
    private bool left = false;
    private bool up = false;
    private bool down = false;
    private bool increase = false;
    private bool decrease = false;

    void Update()
    {
        if (right)
        {
            // Pulsación botón derecho
            this.transform.Translate(Vector3.right * Time.deltaTime * Velocidad);
        }

        if (left)
        {
            // Pulsación botón izquierdo
            this.transform.Translate(Vector3.left * Time.deltaTime * Velocidad);
        }

        if (up)
        {
            // Pulsación botón arriba
            this.transform.Translate(Vector3.up * Time.deltaTime * Velocidad);
        }

        if (down)
        {
            // Pulsación botón abajo
            this.transform.Translate(Vector3.down * Time.deltaTime * Velocidad);
        }

        if (increase)
        {
            // Pulsación aumento de la escala
            this.transform.localScale = new Vector3(
                this.transform.localScale.x + 1.05F,
                this.transform.localScale.y + 1.05F,
                this.transform.localScale.z + 1.05F);
        }

        if (decrease)
        {
            // Pulsación disminución de la escala
            this.transform.localScale = new Vector3(
                this.transform.localScale.x - 1.05F,
```

```
                this.transform.localScale.y - 1.05F,
                this.transform.localScale.z - 1.05F);
        }
    }


    public void MoverDerecha()
    {
        right = true;
    }

    public void MoverIzqda()
    {
        left = true;
    }

    public void MoverArriba()
    {
        up = true;
    }

    public void MoverAbajo()
    {
        down = true;
    }

    public void Detener()
    {
        right = false;
        left = false;
        up = false;
        down = false;
    }

    public void Aumentar()
    {
        increase = true;
    }

    public void Disminuir()
    {
        decrease = true;
    }

    public void DetenerEscala()
    {
        increase = false;
        decrease = false;
    }
}
```