UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

APLICACIÓN DE ASISTENCIA BASADA EN REALIDAD AUMENTADA PARA LA INDUSTRIA

MEMORIA PRESENTADA POR:

Diego Coloma Bravo

TRABAJO FINAL DE GRADO

GRADO DE INGENIERÍA INFORMÁTICA

Tutor: Jordi Linares Pellicer

Convocatoria de defensa: febrero/marzo, 2019

RESUMEN

En el presente trabajo se pretende desarrollar un sistema de asistencia para la industria con Realidad Aumentada. La idea principal es, mediante la ayuda de la Realidad Aumentada, desarrollar una aplicación que sea capaz de ayudar a la reparación o instalación de maquinaria en la industria y que cuente con un sistema para poder crear casos de asistencia de forma sencilla, sin necesidad de utilizar código.

Con esta aplicación, los operarios que tengan que instalar o reparar una máquina o aparato, con la ayuda de algún dispositivo que disponga de una cámara y una pantalla o lugar donde poder mostrar el contenido (en el que esté instalada nuestra aplicación) tendrán una guía basada en Realidad Aumentada que les ayudará a realizar dicha instalación o reparación. Irán apareciendo indicaciones en tiempo real y combinando el mundo real con la Realidad Aumentada de las acciones que el operario debe realizar para completar la reparación. Cada vez que el operario realice una de ellas, aparecerá la siguiente, así sucesivamente hasta completar el trabajo. El objetivo de este proyecto es que el operario optimice el tiempo de trabajo, y pueda realizar dicho trabajo de forma más efectiva, dado que al visualizar la solución le será más fácil solucionar el problema.

Palabras clave: Industria, Realidad Aumentada, Asistencia, Unity.

RESUM

En el present treball es pretén desenvolupar un sistema d'assistència per a la indústria amb Realitat Augmentada. La idea principal és, mitjançant l'ajuda de la Realitat Augmentada, desenvolupar una aplicació que siga capaç d'ajudar a la reparació o instal·lació de maquinària en la indústria i que compte amb un sistema per poder crear casos d'assistència de manera senzilla, sense necessitat d'utilitzar codi.

Amb aquesta aplicació, els operaris que hagen d'instal·lar o reparar una màquina o aparell, amb l'ajuda d'algun dispositiu que dispose d'una càmera i una pantalla o lloc on poder mostrar el contingut (en el qual estiga instal·lada la nostra aplicació) tindran una guia basada en Realitat Augmentada que els ajudarà a realitzar aquesta instal·lació o reparació. En el mateix dispositiu, aniran apareixent indicacions en temps real i combinant el món real amb la Realitat Augmentada de les accions que l'operari ha de realitzar per a completar la reparació. Cada vegada que l'operari realitze una d'elles, apareixerà la següent, així successivament fins a completar el treball. L'objectiu d'aquest projecte és que l'operari optimitze el temps de treball, i puga fer aquest treball de forma més efectiva, ja que en visualitzar la solució li siga més fàcil solucionar el problema.

Paraules clau: Indústria, Realitat Augmentada, Asistència, Unity.

ABSTRACT

This paper aims to develop an assistance system for the industry with Augmented Reality. The main idea is, through the aid of Augmented Reality, developing an application to be able to support the repair or installation of machinery in the industry, that also has a system to create assistance cases easily, without using code.

With this application, the operators who have to install or repair a machine or device, with the help of a device with a camera and a screen or place where you can display the content (whereby our application is installed) it will have a guide based on Augmented Reality that will help you to complete the installation or repair. On the same device, indications will appear in real-time and they will combine the real world with the Augmented Reality actions that the operator has to perform to complete the repair, and each time the operator performs one of them, the following will appear, and so on, until the work is completed. The aim of this project is to optimise the operator's work time, and also to be able to carry out the work more effectively, as it is easier for him to solve the problem when he visualises the solution.

Keywords: Industry, Augmented Reality, Assistance, Unity.

"Podría parecer que hemos llegado a los límites alcanzables por la tecnología informática, aunque uno debe ser prudente con estas afirmaciones, pues tienden a sonar bastante tontas en cinco años".

John Von Neumann.

"Puesto que solo hay un mundo real, mientras que el número de mundos virtuales potenciales es infinito, la probabilidad de que habitemos el único mundo real es casi nula. Ninguno".

Yuval Noha.

"La clave del éxito en los negocios está en detectar hacia dónde va el mundo y llegar ahí primero".

Bill Gates.

AGRADECIMIENTOS

Escribo estas líneas para dar mi más sincero agradecimiento a todas esas personas que de un modo u otro me han acompañado durante esta etapa de mi vida. A mis padres y mi hermana en especial, por estar a mi lado y prestarme su ayuda en todo lo que han podido. Como aconsejarme y darme sus opiniones, pero sobre todo por su paciencia y apoyo constante a lo largo de este período.

Tampoco me olvido de mi tutor de proyecto Jordi Linares, a quien agradezco la dedicación, los consejos y el aprendizaje que me ha proporcionado, porque sin todo esto la elaboración del presente proyecto no hubiera sido posible.

Por último, decir que la elaboración de este trabajo exigía una gran dedicación y compromiso por parte del autor, pero también por parte de todas esas personas a las que dedico estos agradecimientos, ya que han dedicado parte de su tiempo y han estado siempre que las he necesitado ayudando a que esto fuera posible.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

GLOSARIO DE TÉRMINOS Y ABREVIACIONES

- **App:** Aplicación.
- **AR:** Augmented Reality. En castellano, Realidad Aumentada.
- **Caso de asistencia:** En este caso se define como una solución creada basándose en la Realidad Aumentada para dar soporte a un trabajo que necesita una asistencia.
- **IOT:** Internet of Things.
- **JDK:** Java Development Kit.
- **RA:** Realidad Aumentada.
- **RV:** Realidad Virtual.
- **SDK:** Software Development Kit.
- **Target:** En castellano, objetivo. En este caso se define como el elemento que debe reconocer el dispositivo para mostrar el contenido en Realidad Aumentada.
- **UI:** User Interface. En castellano, Interfaz de Usuario.
- **VR:** Virtual Reality. En castellano, Realidad Virtual.
- **XR:** Extended Reality. En castellano, Realidad Extendida.

1. INTRODUCCIÓN Y MOTIVACIÓN

Esta idea surge de un problema que se plateó en la empresa donde estoy realizando las prácticas del Grado en Ingeniería Informática. El problema que planteó la empresa al departamento TIC fue que, al tener varias sedes repartidas por el mundo, y los expertos estar en la sede de Alcoy (España), cada vez que hay una rotura o atasco en una de las máquinas de producción que hay en las distintas sedes, se presenta muy difícil la tarea de explicar a los técnicos de cada sede, como hay que reparar dicha rotura, como desatascar la máquina o como solucionar dicha anomalía. Esto supone que las máquinas estén mucho tiempo paradas, e incluso, que muchas veces los expertos que residen en Alcoy se tengan que desplazar a otros países para hacer estas reparaciones y formar a los técnicos de la sede a la que se desplazan. Esto supone un coste muy elevado tanto por el desplazamiento como por el tiempo de paro, y para solucionar esto, la respuesta de nuestro departamento fue clara, la teleasistencia remota. En el momento en el que investigamos un poco nos dimos cuenta qué, la mejor opción era la de una aplicación de teleasistencia remota que utilice Realidad Aumentada.

El funcionamiento de este tipo de aplicaciones es sencillo. El técnico realiza una llamada al experto con un dispositivo compatible (gafas de AR, *smartphone*…), y una vez el experto coge la llamada, este podrá ver todo lo que esté viendo el técnico, y en la misma imagen podrá hacer marcas y enseñarle otras imágenes o descripciones. Todo esto lo verá el técnico en su dispositivo, superpuesto a la imagen real que ven sus ojos. Con esta técnica se consigue una comunicación fluida y que presta mucha ayuda al técnico. Como ejemplo de este tipo de aplicación podemos nombrar: Teamviewer Pilot, ARSoft o ATR.

Desde hace unos años, la Realidad Aumentada es una tecnología en auge, que presenta muchas oportunidades y grandes retos. Muchas son las empresas que están apostando por esta tecnología y muchas más lo harán en un futuro próximo. La llamada Industria 4.0 o Cuarta Revolución Industrial, que combina técnicas avanzadas de producción y operaciones con tecnologías inteligentes[1] ya es una realidad, y la Realidad Aumentada será un pilar fundamental en esta revolución.

Juntando la curiosidad que me surgió al investigar un poco sobre el tema de la Realidad Aumentada durante las prácticas, las oportunidades y posibilidades que nos ofrece esta tecnología, y el crecimiento exponencial que está teniendo su utilización tanto en el mundo empresarial como en la vida cotidiana, tuve ganas de ir un poco más

[1] *https://www2.deloitte.com/es/es/pages/manufacturing/articles/que-es-la-industria-4.0.html*

allá. Gracias a todo esto tuve la idea de profundizar un poco más, y utilizar todo lo aprendido en el Grado de Ingeniería Informática para desarrollar una aplicación en Realidad Aumentada como Trabajo Fin de Grado.

Éste Trabajo Fin de Grado se centra en una aplicación que utilizando Realidad Aumentada pretende ofrecer un servicio de asistencia desatendida, donde los usuarios puedan, con un dispositivo compatible, ver en su dispositivo indicaciones, y siguiendo unos pasos, realizar una reparación de un dispositivo, cambiar una pieza etc. En concreto, para este TFG nos vamos a centrar en la reparación de una máquina industrial, pero esto mismo se podría hacer para cualquier otro dispositivo u objeto.

1.1 OBJETIVOS

El objetivo que se pretende conseguir con la realización de este Trabajo Fin de Grado es completar el diseño y desarrollo de una aplicación de asistencia para la industria que sirva para ayudar a la reparación o instalación de distintos tipos de dispositivos o maquinaria.

Para conseguir este objetivo final, se proponen los siguientes subobjetivos:

a) Investigación y elección de las herramientas a utilizar para el desarrollo de la aplicación.
b) Integración de la Realidad Aumentada con la aplicación.
c) Conexión con una base de datos para controlar el *login* de los usuarios y guardar las valoraciones que los usuarios hagan de la aplicación.
d) Desarrollo de un sistema con el que se puedan realizar casos de asistencia sencillos sin necesidad de utilizar código, para que no sea necesario un programador para crear dichos casos de asistencia.
e) Desarrollo de un caso de asistencia con necesidades específicas que necesitan de programación.
f) Implementación de la Interfaz de Usuario.
g) Implementación de añadidos que pueden ser útiles para el usuario, tales como guardar capturas de pantalla o encender el flash del dispositivo.

Consiguiendo todos estos objetivos se pretende acabar desarrollando una aplicación que ayude a los usuarios finales con las tareas de reparación o instalación de dispositivos o maquinaria que tengan que realizar, mostrándoles los pasos a seguir de una forma clara e interactiva. También se pretende obtener un sistema para que los casos en que los pasos a seguir sean sencillos, se pueda crear esta ayuda de forma sencilla y sin necesidad de programar. Así mismo, también se desarrolla un caso en el

que dicha asistencia es más compleja, y se necesita de programación para lograr implementar este caso.

1.2 IMPORTANCIA DEL TEMA

Con el pasar de los años, la industria ha ido evolucionando y se ha logrado avanzar tanto en los procesos de fabricación como en la ejecución de estos.

A mediados del siglo XVIII, con las máquinas de vapor, llegó la Primera Revolución Industrial, nacida en Gran Bretaña. En este cambio se pasó del trabajo manual al comienzo de la utilización de maquinaria.

A mediados del siglo XIX se inicia la Segunda Revolución Industrial. El gran invento de esta fue la cinta transportadora, con la que se empezó a fabricar en serie.

La Tercera Revolución Industrial o Revolución Científico-Técnica es un término aprobado por el Parlamento Europeo en 2007.[2] Este proceso se asienta sobre distintos factores, de los cuales estos son los cinco principales:

- Innovaciones en el almacenamiento de energía.
- Expansión de las energías renovables.
- Formas de transporte más eficientes y menos contaminantes.
- Conversión de edificios en plantas de energía.
- Tecnología *Smart Grid*.

Ahora, todo apunta a que estamos delante, o que incluso hemos llegado a la Cuarta Revolución industrial. Como ejemplo, se puede ver lo que dice Klaus Schwab en su libro 'La Cuarta Revolución Industrial': *"Estamos al borde de una revolución tecnológica que modificará fundamentalmente la forma en que vivimos, trabajamos y nos relacionamos. En su escala, alcance y complejidad, la transformación será distinta a cualquier cosa que el género humano haya experimentado antes*"[3]. Esta revolución está marcada por tecnologías como la Inteligencia Artificial, Internet de las Cosas (IOT), computación cuántica o nanotecnología entre otras. La fusión de todas estas tecnologías con el mundo biológico, digital y físico, en conjunto con su impacto en todas las disciplinas, es lo que denominamos cómo Cuarta Revolución Industrial. Este cambio representa, cómo la tecnología va a integrarse con el ser humano de aquí en adelante.

---

[2] *https://www.satel-iberia.com/la-tercera-revolucion-industrial-o-la-revolucion-cientifico-tecnica/*

[3] *Klaus Schwab. La cuarta Revolución Industrial*

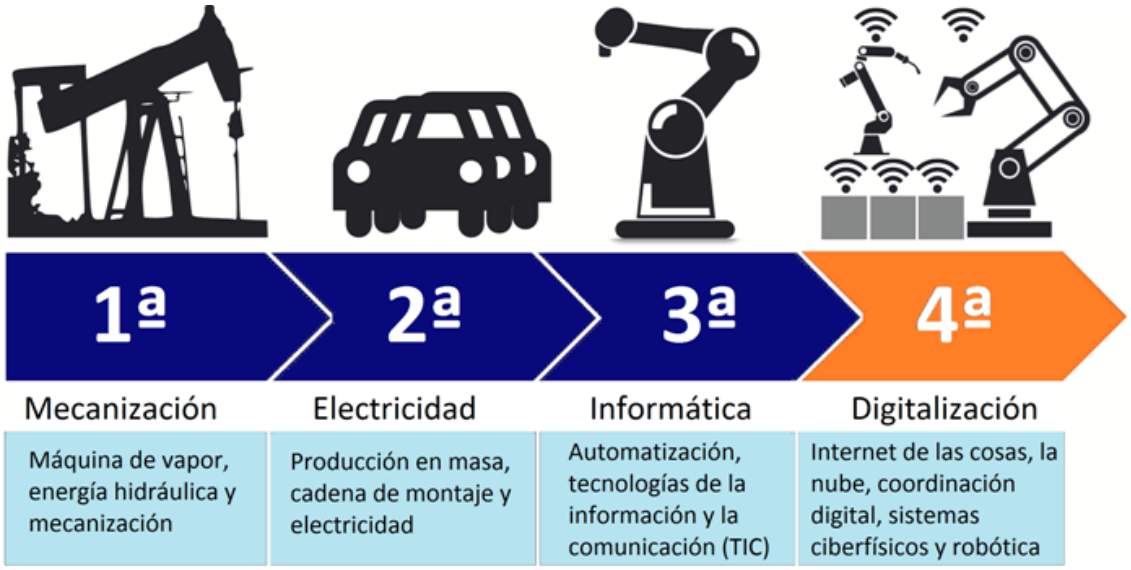


Figura 1. Niveles RA. Fuente: https://economipedia.com/definiciones/cuarta-revolucion-industrial.html

La importancia de esta Cuarta Revolución es que afectará a toda la sociedad, no solo al sector industrial, ya que mejorará los procesos de producción, pero también cambiará por completo el modo en el que los clientes vean los productos que deseen, como se diseñen dichos productos o incluso que aparezcan productos o servicios completamente nuevos. El impacto de esta revolución será a todos los niveles: a nivel de grandes y pequeños ecosistemas; a nivel de empleados y clientes; a nivel individual y organizativo.

Es muy importante adaptarse a estos nuevos cambios y no quedarse atrás, ya que estas tecnologías que parecen el futuro ya están aquí y de forma inmediata empezaran a implantarse tanto en la vida diaria del ser humano, como en su trabajo, por ello es muy conveniente investigar y conocer sobre estas tecnologías y estar preparado para cuando estas tecnologías se acaben implantando para quedarse.

1.3 ¿QUÉ PIENSA LA GENTE DE LA REALIDAD AUMENTADA?

Parece interesante, ya que esta tecnología es una tecnología nueva, y no conocida por todos, preguntar a personas cercanas al autor sobre la Realidad Aumentada para ver su reacción y que piensan sobre dicha tecnología.

La pregunta que se ha realizado a todos los entrevistados es la siguiente:

“El Trabajo Fin de Grado que se está realizando es una aplicación para dispositivos Android que utiliza la Realidad Aumentada. ¿Me podría decir que piensa sobre la Realidad Aumentada? Gracias.”

Y estas son las respuestas:

a. *"No sé lo que es la Realidad Aumentada"*

Roberto García, amigo del autor.

b. *"Sé que es algo parecido a la Realidad Virtual, pero no podría definirlo con claridad"*

Jaume Espí, amigo y compañero del autor en el Grado de Ingeniería Informática.

c. *"¿Eso es lo de las Google Glass no? Me suena haber escuchado en las noticias que las gafas esas que presentaba Google funcionaban con Realidad Aumentada"*

Francisco Juan, tío político del autor.

d. *"Me suena escucharlo en la tele, pero no sé lo que es"*

David Llinares, amigo del autor.

Como se puede ver, de las personas que han sido preguntadas la gran mayoría no conocen esta tecnología, ya que es relativamente nueva y aún no está instaurada en la vida cotidiana de la gente. Incluso la respuesta de un estudiante del Grado de Ingeniería Informática es que no sabe con claridad qué es este término. Esto da evidencias de que la Realidad Aumentada es un campo con infinitas posibilidades y aplicaciones, pero con mucho por explorar y mucho camino por recorrer.

2. ESTADO DEL ARTE

2.1 INTRODUCCIÓN A LA REALIDAD AUMENTADA

Aunque desde principios del siglo XX, ya algunos (como Morton Heiling, pionero de la Realidad Virtual[4]) tenían algunas ideas parecidas a lo que ahora se conoce como Realidad Aumentada, la primera vez que se utiliza este término es en 1992, y Tom Caudell es el que lo nombra[5]. Caudell utilizaba este término para referirse a dispositivos que aportaban más información a los operarios de la que podían captar por sí mismos.

Esta fusión mencionada anteriormente se consigue superponiendo imágenes virtuales que proporcionen alguna información adicional sobre la imagen real. Con esta superposición se consigue obtener gran cantidad de información que mediante los sentidos que posee el ser humano no se podrían ver, pero a la vez, y aquí reside la clave de esta tecnología, no perder de vista la realidad.

Para hacer uso de esta tecnología se necesita un sistema que, según Edgar Mozas Fenoll[6], debe estar formado en la mayoría de los casos por los elementos siguientes:

- Cámara.
- Procesador.
- Pantalla o proyector.
- Activador.
- Marcador.

Además de estos elementos de hardware, también necesitaremos un software y estos son algunos ejemplos:

- Unity.
- Blender.
- ARToolKit.
- HP-Reveal.

En la siguiente figura [Figura 2] podemos ver un ejemplo de lo que es la Realidad Aumentada. Esta *app* muestra información adicional del entorno que rodea al usuario que esté utilizando dicha aplicación. En este caso, se trata de unos textos superpuestos

---

[4] *Pimentel, K., & Teixeira, K. (1993). Virtual reality. New York, NY: McGraw-Hill*

[5] *Caudell & Mizel, Thomas & David (7 de febrero de 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. doi:10.1109/HICSS.1992.183317. Consultado el 31 de octubre de 2019*

[6] *Edgar Mozas Fenoll (2016). TIC 4.1 Qué es la realidad aumentada*

a los objetos que hay en la imagen. Se pueden diferenciar así los elementos que vemos en la imagen:

a) **Elementos virtuales**: Los cuadros superpuestos que aparecen con iconos y texto, que están dando información sobre los establecimientos cercanos del lugar desde donde se está utilizando la aplicación. Un ejemplo es el cuadro en el que podemos ver escrito 'Shopping center' junto con un icono amarillo con una cesta de la compra.

b) **Elementos reales**: Como elementos reales se pueden ver las calles, los edificios, las personas, los vehículos etc.

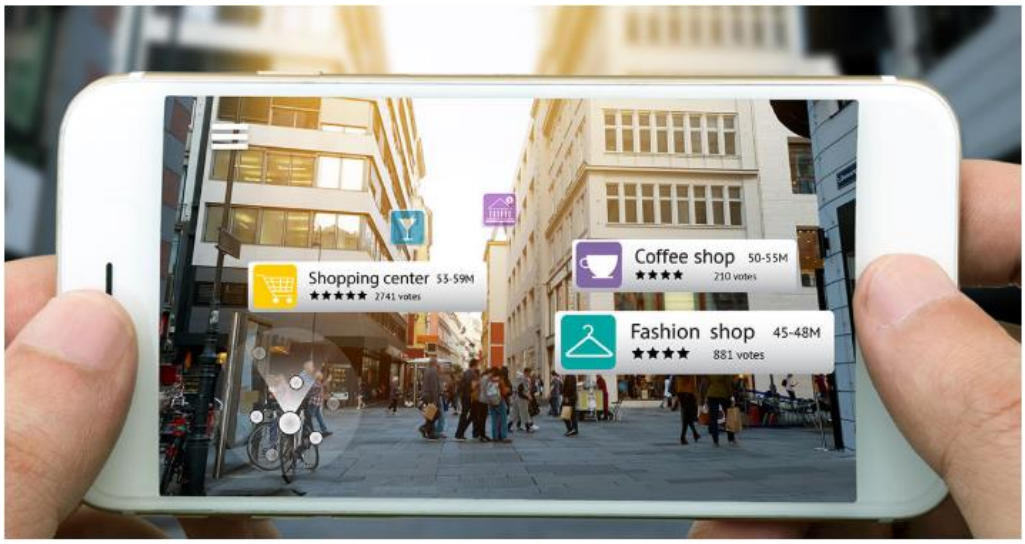


Figura 2. App Realidad Aumentada. Fuente: https://www.ionos.es/digitalguide/online-marketing/vender-en-internet/app-de-realidad-aumentada/

2.1.1 Niveles de la Realidad Aumentada

Lens-Fitzgerald en 2009, en su artículo *Augmented Reality Hype Cycle*[7] propone una nueva clasificación para las aplicaciones de RA. Esta clasificación se puede entender como una posible forma de medir dichas aplicaciones por el elemento que activa la experiencia en RA de la aplicación. La clasificación que se propone en este artículo es una clasificación de cuatro niveles (0-3):

a) **Nivel 0** – Hiperenlazando el mundo físico (*physical world hyper linking*). En este nivel, las aplicaciones están basadas en códigos de barras, códigos QR… Estos códigos son hipervínculos a contenidos externos, es decir que no existe registro

[7] *LENS-FITZGERALD, M. (2009). Augmented Reality Hype Cycle. Recuperado de http://acdc.sav.us.es/pixelbit/images/stories/p46/12.pdf*

ninguno en 3D. Estas aplicaciones son a grandes rasgos hiperenlaces en los que no es necesario escribir.

Figura 3. Código de barras. Fuente: https://hdnh.es/inventos-codigo-de-barras/

Figura 4. Código QR. Fuente: https://es.qr-code-generator.com/

b) **Nivel 1** – Basado en marcadores *(marker based* AR). Los activadores de AR en este nivel son llamados marcadores. Una buena definición de marcador podría ser la siguiente: "*Los marcadores son unas imágenes en blanco y negro, generalmente cuadradas, con dibujos sencillos y asimétricos.*"[8] Cuando la cámara escanea estos marcadores aparecen superpuestos objetos 3D a la imagen real.

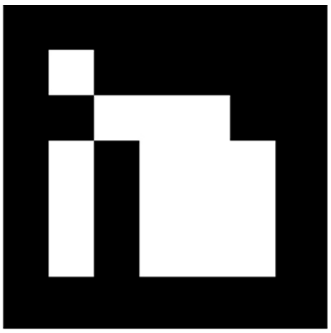


Figura 5. Marcador. Fuente: http://www.aumentaty.com/community/es/pin/ficha/marcador-ra-2/

Figura 6. Marcador activado. Fuente: https://ardev.es/realidad-aumentada/

c) **Nivel 2** – Realidad Aumentada sin marcadores (*markerless* AR). Los activadores en este caso no son marcadores, sino que pueden ser desde imágenes, objetos o localizaciones (están capturadas gracias al GPS y brújula del dispositivo que utilicemos) del mundo real.

[8] *Estebanell, M., Ferrés, J., Cornellà, P. & Codina, D. (2012). Realidad aumentada y códigos QR en educación. En J. Hernández, M. Pennesi, D. Sobrino & A. Vázquez (Coords) Tendencias emergentes en educación con TIC. (pp. 277-320). Barcelona: Editorial espiral.*

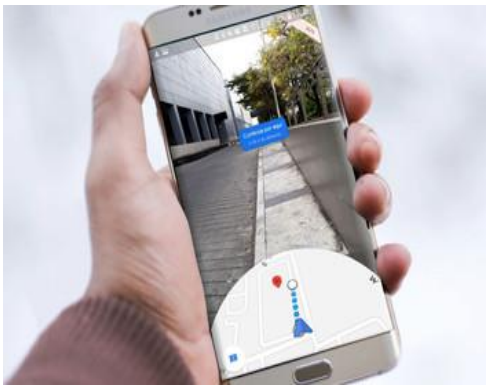


Figura 7. RA activada con GPS. Fuente: https://www.xatakandroid.com/navegacion-y-mapas/google-maps-realidad-aumentada-asi-ultima-prueba-para-seguir-direcciones-usando-camara

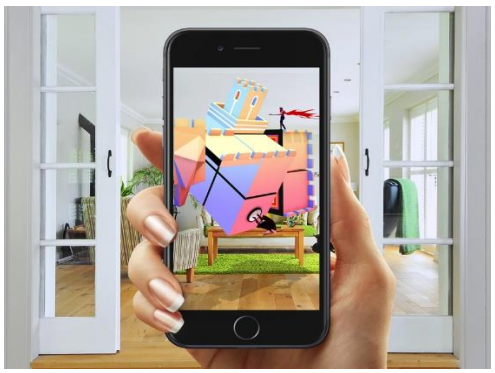


Figura 8. RA sin marcadores. Fuente: https://www.nobbot.com/pantallas/mejores-apps-realidad-aumentada/

d) **Nivel 3** – Visión Aumentada (*Augmented Vision*). Este nivel está representado por los dispositivos concebidos específicamente para Realidad Aumentada, las *smartglasses*. Con estos dispositivos se puede conseguir una experiencia inmersiva, que se fusione con el mundo real de una forma muy natural.

Figura 9. Nivel 3 Realidad Aumentada. Fuente: https://www.macworld.es/articulos/apple/gafas-realidad-aumentada-apple-3696297/

2.1.2 Dispositivos para Realidad Aumentada

Como ya se ha comentado, para superposición de elementos en los que se basa la Realidad Aumentada se necesita un dispositivo que permita hacer uso de esta tecnología. Estos son los diferentes tipos de dispositivo que se encuentran en el mercado, y con los que se puede utilizar dicha tecnología:

a) **Ordenador:** Un ordenador (tanto un PC de sobremesa como un ordenador portátil) es un dispositivo perfectamente válido siempre que cuente con una cámara y una pantalla, ya sean integradas, como suele ser el caso de los ordenadores portátiles, o sean periféricos de un ordenador de sobremesa.

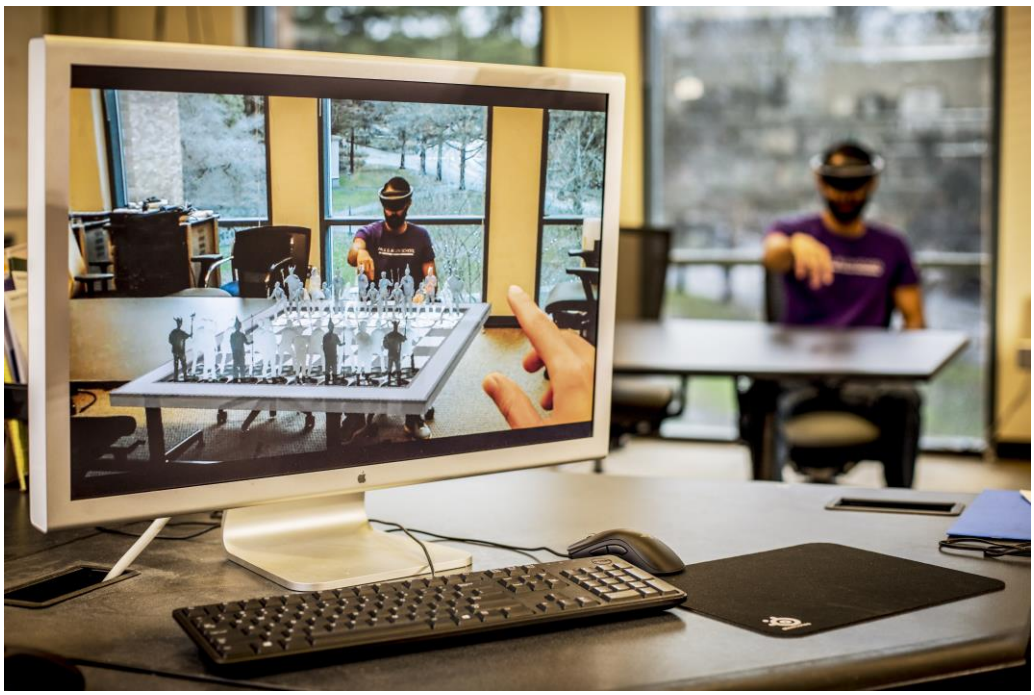


Figura 10. Ordenador ejecutando RA. Fuente: https://www.washington.edu/news/2018/01/08/uw-reality-lab-launches-with-6m-from-tech-companies-to-advance-augmented-and-virtual-reality-research/

b) **Smartphone:** Smartphone (‘Teléfono Inteligente’ si lo traducimos literalmente del inglés) es un teléfono con una pantalla táctil, y un software que permite a los usuarios realizar muchas actividades parecidas a las que puede realizar un ordenador, como por ejemplo conectarse a internet, crear archivos o instalar aplicaciones[9]. El ejemplo más claro de utilización de la Realidad Aumentada en un dispositivo móvil es el fenómeno en forma de juego, que invadió los móviles de la mayoría de los usuarios el verano del 2016, Pokémon Go. Este juego

[9] *https://www.whistleout.com.mx/CellPhones/Guides/que-es-un-smartphone*

integra personajes virtuales con los que el usuario puede interactuar mediante la pantalla táctil de su teléfono móvil o Tablet en ubicaciones del mundo real.

Figura 11. Smartphone ejecutando Pokémon Go. Fuente: https://www.digitaltrends.com/mobile/best-augmented-reality-apps/

c) ***Headset* o Gafas de Realidad Aumentada (del inglés Smartglasses):** Estos dispositivos, se colocan en la cabeza del usuario, y por medio de cristales en forma de gafas o visores, se proyectan imágenes sobre estos cristales par que funcionen como si de una pantalla se tratase. Esta es la forma más útil e intuitiva para el usuario de integrar la realidad aumentada, ya que percibe la información como si de su propia vista se tratase haciendo uso de sensores integrados en el dispositivo para correlacionar las imágenes que se muestran con la propia vista y tener ambas manos libres para trabajar[10].

Estos son algunos de los ejemplos más conocidos:

- **Google Glass**

  Las Google Glass son las smartglasses de la empresa Google. Estas fueron lanzadas en junio del año 2012. Salieron a la venta para el público general, el 15 de mayo de 2014, aunque el 15 de enero de 2015 Google anunció su retirada del mercado. Desde julio de 2017 vuelven a estar a la venta.

[10] *https://pandorafms.com/blog/es/realidad-aumentada/*

La presentación de estas gafas alzó mucho revuelo ya que la empresa Google fue la primera que creó un dispositivo que utilizaba la Realidad Aumentada, y estaba enfocado al uso diario. Era un *wearable* que buscaba, mostrando información en los ojos de las personas que lo utilizaran, mejorar o potenciar la vida de estas.

- **Microsoft Hololens**

Las Microsoft Hololens son las smartglasses de la empresa Microsoft. Estas fueron lanzadas el 30 de marzo de 2016 en EEUU y Canadá, y el 12 de octubre del mismo año anunció la expansión mundial del producto. El 24 de febrero de 2019 se anunciaron las Hololens 2, en el Mobile World Congres de Barcelona. El coste de estas gafas es de unos $3000 la versión estándar, y unos $5000 la *Suite*.

- **Epson Moverio**

En este caso se habla de la alternativa que ofrece la empresa Epson. En 2016 se lanzó el primer modelo de estas, y ahora contamos con distintos modelos. Desde las BT-350, un modelo más cómodo y para un uso más diario hasta las BT-2200, rugorizadas y preparadas para su uso en un entorno industrial. La clave del auge de estas smartglasses es su precio y es que podemos encontrar unas Epson BT-350 por unos 750€. [Figura 12]

Figura 12. Imagen Epson Moverio BT-300. Fuente: http://www.inavateonthenet.net/news/article/epson-now-offering-moverio-smart-glasses-to-wider-audience

2.1.3 Oportunidades y aplicaciones de la realidad aumentada

La Realidad Aumentada tiene infinitas aplicaciones en muchos ámbitos distintos. Esto se debe en gran parte a algo que ya se ha comentado, y es que mientras la se está utilizando no se deja de percibir la realidad. Comparando esta tecnología con la Realidad Virtual (VR), se puede ver como la Realidad Virtual se basa en una inmersión total del individuo que la está utilizando, por el contrario, la Realidad Aumentada consigue una inmersión parcial, con lo que se puede interactuar con el mundo real y el virtual a la vez.

Figura 13. Realidad Virtual / Realidad Aumentada. Fuente: http://www.iamwire.com/wp-content/uploads/2017/07/Augmented-reality-vs-virtual-reality.jpg

Esta interacción de la que se habla hace que la Realidad Aumentada sea tan versátil y esté presente en tantos ámbitos tanto profesionales como sociales. Estos son ejemplos de algunos de ellos:

a) **Educación:** En la educación, una de las posibilidades que ofrece la Realidad Aumentada es la de incluir marcadores en los libros de texto, i que sobre estos aparezcan imágenes en 3D de objetos o representaciones que ayuden a los alumnos a asimilar y reforzar los conocimientos aprendidos. Hirokazu Kato, desarrollador de ArToolKit: "*Creo que la realidad aumentada es la mejor forma de conexión entre el mundo real y los contenidos digitales, esta característica*

permite al usuario reforzar el aprendizaje de los contenidos educativos mediante su asociación al mundo real"[11].

b) Medicina: En el sector médico, las herramientas que proporciona esta tecnología han permitido nuevas formas de realizar pruebas y desempeñar el trabajo médico. Uno de los ejemplos puede ser obtener datos de forma no inmersiva en las resonancias magnéticas, o unas gafas de AR que permiten distinguir las células sanas de las cancerígenas. Además de esto también existen aplicaciones desarrolladas para este sector, aplicaciones como por ejemplo: AccuVein, que ayuda a localizar las venas de los pacientes para que las inyecciones de los profesionales sean de mayor precisión; Brain Power, aplicación que está orientada a las personas con autismo y les ayuda a aprender habilidades que les sirvan en su día a día y a la vez se evalúa su proceso; y por último vamos a nombrar SNAP, una aplicación que ayuda a los cirujanos a preparar las cirugías mostrando distintos puntos de vista y perspectivas en 3D de las intervenciones[12].

c) Turismo: En este sector se ofrece la posibilidad de herramientas que permiten hacer las funciones de una guía turística de una forma sencilla, rápida y eficaz. Como ejemplo se pueden poner aplicaciones que permiten visitar lugares, y con un simple teléfono móvil. Saber toda la historia del lugar (por ejemplo, WikiTude o La Ciudad de México). También hay aplicaciones que se utilizan en los museos, para que los visitantes puedan, desde su propio teléfono móvil ver toda la información sobre las obras de arte expuestas[13].

2.1.4 ¿En qué plataforma hemos realizado el proyecto de Realidad Aumentada?

El sistema operativo que se ha elegido para desarrollar el proyecto es el sistema de código abierto lanzado el 23 de septiembre de 2008, Android. Se ha elegido este tema por dos puntos en concreto. Primero, cuando se decidió desarrollar este proyecto, se pensó en hacerlo en la plataforma que permitiera llegar a más cantidad de dispositivos, y cómo se puede ver en la siguiente figura [Figura 14], Android es el líder del sector y la tendencia sigue al alza. El segundo punto que se tuvo en cuenta es, que tiene mucho

[11] *Realidad Aumentada en la Educación: una tecnología emergente X. Basogain, M. Olabe, K. Espinosa, C. Rouèche* y J.C. Olabe*

[12] *https://realidadenaumento.es/realidad-aumentada-en-la-medicina/*

[13] *https://es.calameo.com/read/0047274995ef80ff0f551*

sentido, trasladar esta aplicación para utilizarla con gafas de RA, ya que la gran mayoría de estas, funcionan sobre un sistema operativo basado en Android.

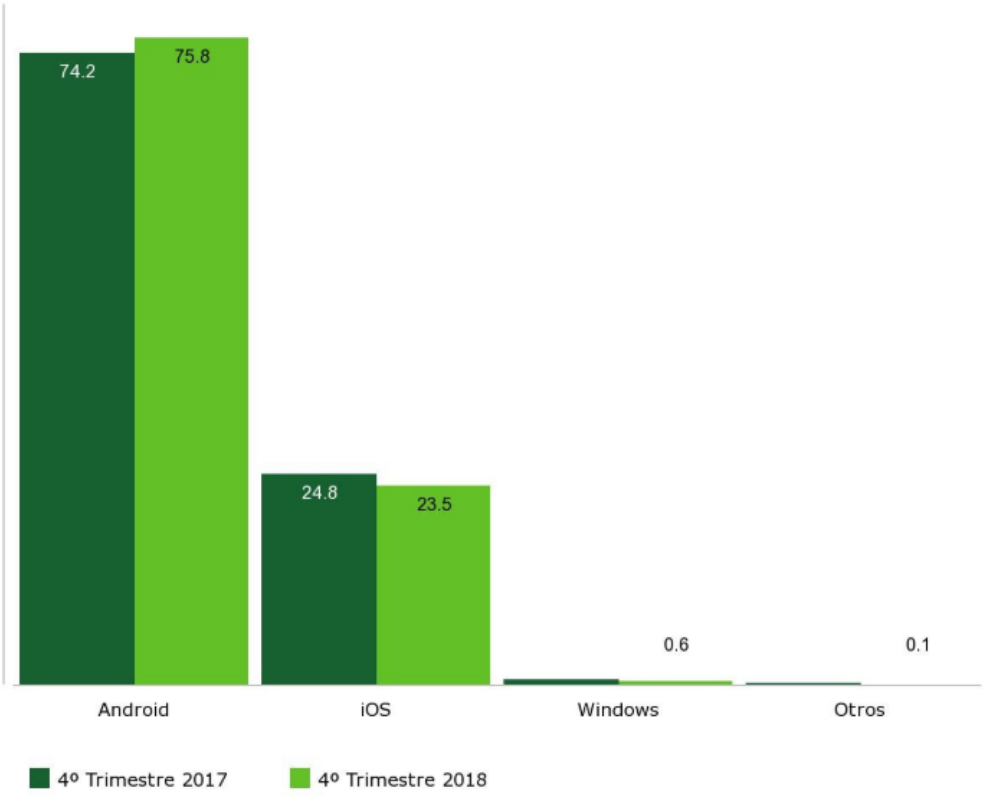


Figura 14. Gráfico cuota de mercado de los sistemas operativos en smartphones. Fuente: https://www.xatakandroid.com/moviles-android/android-termino-2018-rozando-90-cuota-mercado-espana-kantar

2.2 HERRAMIENTAS PARA EL DESARROLLO DE RA

2.2.1 Unity como estándar

Unity, es una herramienta creada por la empresa Unity Technologies (empresa fundada por David Helgason (CEO), Nicholas Francis (CCO) y Joachim Ante (CTO). Esta herramienta es conocida por ser una de las más utilizadas en el desarrollo de videojuegos, de hecho, algunos de los juegos más famosos para la plataforma móvil como son 'Hollow Knight', 'Monument Valley' e incluso videojuegos hechos para ser jugados en Pc como 'Ori and the Blind Forest' están hechos utilizando esta plataforma. Pero Unity no es solo una herramienta de desarrollo de videojuegos ya que, por ejemplo, también es la herramienta de desarrollo más utilizada para los creadores de realidad extendida (XR) al tener soporte tanto par a Realidad Virtual (RV) como Realidad Aumentada (AR).

Utiliza distintos motores gráficos según la plataforma, como por ejemplo OpenGL (para Mac, Linux y Windows), Open GL ES (para Android y iOS) y tiene soporte para sombras dinámicas, mapeado de reflejos o mapeado de relieve[14].

Uno de los puntos fuertes de Unity, es que es multiplataforma. La herramienta de desarrollo tiene soporte para la compilación en una gran cantidad de plataformas (Android, iOS, Windows, SteamOS, PlayStatio4…) y permite crear un juego o aplicación, y de forma muy sencilla, compilarlo para distintas plataformas.

Además de esto, en este entorno existe la opción de utilizar una licencia gratuita, con la que no se puede hacer uso de todos sus componentes y utensilios, pero que es suficiente para una gran cantidad de tareas, y para realizar este TFG.

2.2.2 Vuforia

Vuforia es un SDK (*Software Development Kit*), para dispositivos móviles que permite la construcción de aplicaciones basadas en Realidad Aumentada.[15] Las aplicaciones desarrolladas con este SDK utiliza la pantalla de un dispositivo para superponer imágenes, objetos virtuales etc. Con objetos del mundo real cuando, siendo captadas estas imágenes reales por una cámara.

Vuforia permite el reconocimiento de marcadores, imágenes, objetos 2D, objetos 3D y una forma de marcador propia llamada VuMark. VuMark es un tipo de marcador diseñado específicamente para Vuforia y que en la propia web de Vuforia definen así: *"VuMark es el código de barras de próxima generación. Permite la libertad de un diseño personalizado y consciente de la marca mientras codifica datos simultáneamente y actúa como un objetivo AR. Los diseños de VuMark son completamente personalizables, por lo que puede tener un VuMark único para cada objeto único."*[16].

2.2.2.1 RECONOCIMIENTO DE IMÁGENES DE VUFORIA

Al utilizar imágenes como *targets* se necesita que dichas imágenes sean fáciles de reconocer para la herramienta que se va a utilizar (Vuforia en nuestro caso), para que la aplicación funcione correctamente.

Vuforia, para rastrear y reconocer los targets, se basa en el contraste de dicho *target* que es captado por la cámara del dispositivo.

---

[14] *https://docs.unity3d.com/es/530/Manual/*

[15] *https://www.ptc.com/en/products/augmented-reality/vuforia*

[16] *https://library.vuforia.com/articles/Training/VuMark*

Como se puede ver en la siguiente imagen, al subir la imagen a la web de Vuforia, para poder descargarla posteriormente en la base de datos, en la misma web se puede ver un *Rating* que va de 0 a 5 estrellas de la viabilidad de la imagen para ser reconocida por la herramienta.

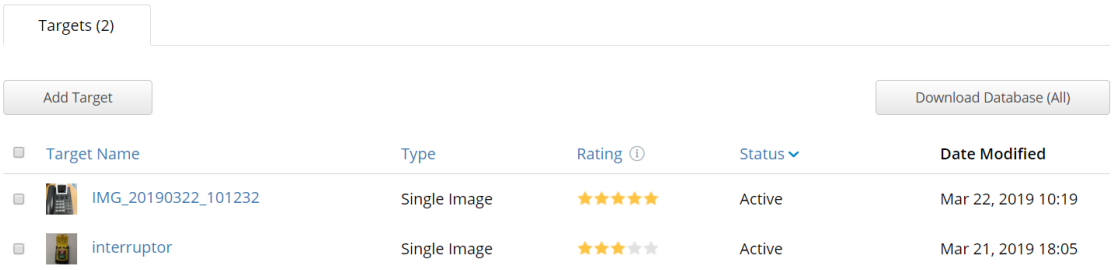


Figura 15. Targets en Vuforia. Fuente: Autor

Para que la imagen pueda ser reconocida con facilidad por Vuforia se recomienda que sea una imagen rica en detalles, tenga un gran contraste con regiones brillantes y oscuras, esté bien iluminada y no tengan patrones repetitivos[17]. Además de esto, para poder subir la imagen a Vuforia Developer Portal, debe de ser una imagen .jpg o .png y tener un peso inferior a los 2MB.

Figura 16. Ejemplo buen target. Fuente: https://library.vuforia.com/articles/Solution/Optimizing-Target-Detection-and-Tracking-Stability

[17] *https://library.vuforia.com/articles/Solution/Optimizing-Target-Detection-and-Tracking-Stability*

2.2.3 Otras herramientas para la creación de aplicaciones utilizando RA

La elección para realizar este TFG ha sido la de utilizar la herramienta Unity unido con Vuforia para la creación de la aplicación, pero en el mercado existen muchas alternativas, y estas son algunas de las más interesantes:

a) **LayAR.** Es una de las más conocidas y su principal característica es que permite visualizar el terreno a través de capas. Se utiliza principalmente para crear imágenes interactivas que pueden servir para fines como promociones o publicidad.

b) **Blippar.** La principal característica de Blippar es que es una herramienta web. Es una herramienta muy sencilla, pero con la que con pocos conocimientos sobre el tema se pueden crear apps sencillas sin necesidad de descargar ningún programa, simplemente registrándonos en su web.

c) **Wikitude.** Esta es una herramienta también muy sencilla de utilizar, y que permite crear aplicaciones con gran facilidad pero que a su vez permite reconocimiento tanto 2D como 3D, rastreo de localización, representación y animación de modelos 2D y 3D etc.

d) **Metaio.** Otra de las más utilizadas y que ofrece un gran número de funcionalidades. Es una herramienta muy completa que ofrece características enfocadas a sectores como la industria, el marketing o la automoción. Empresas como Ikea o Audi han desarrollado distintos proyectos utilizando Metaio.

3. PLANIFICACIÓN DEL PROYECTO

La planificación se puede definir como: “un proceso bien meditado y con una ejecución metódica y estructurada, con el fin el obtener un objetivo determinado.”[18]

Para el desarrollo de este TFG y con el objetivo de cumplir satisfactoriamente todos los objetivos planteados anteriormente se ha propuesto la siguiente planificación a seguir para lograr completar el desarrollo de una forma satisfactoria.

FASE	TAREAS POR REALIZAR
FASE DE PLANIFICACIÓN	**1.** Escoger la idea a realizar. **2.** Realizar una búsqueda de información y un análisis del tema escogido. **3.** Planificación de las prioridades y el orden de desarrollo de los objetivos deseados.
FASE DE DESARROLLO	**1.** Implementación y proceso de pruebas de cada uno de los subdesarrollos necesarios para implementar los objetivos.
FASE FINAL	**1.** Realización de las últimas pruebas. **2.** Entrega del proyecto.

Tabla 1. Planificación del proyecto. Fuente: Elaboración propia.

3.1 DIAGRAMA DE PROCESOS

Se ha creado un diagrama de procesos a seguir para el desarrollo de los objetivos planteados en los apartados anteriores. Con este diagrama, se puede ver de forma gráfica el proceso paso de desarrollo para cada uno de los objetivos paso a paso.

[18] https://www.webyempresas.com/que-es-la-planificacion/

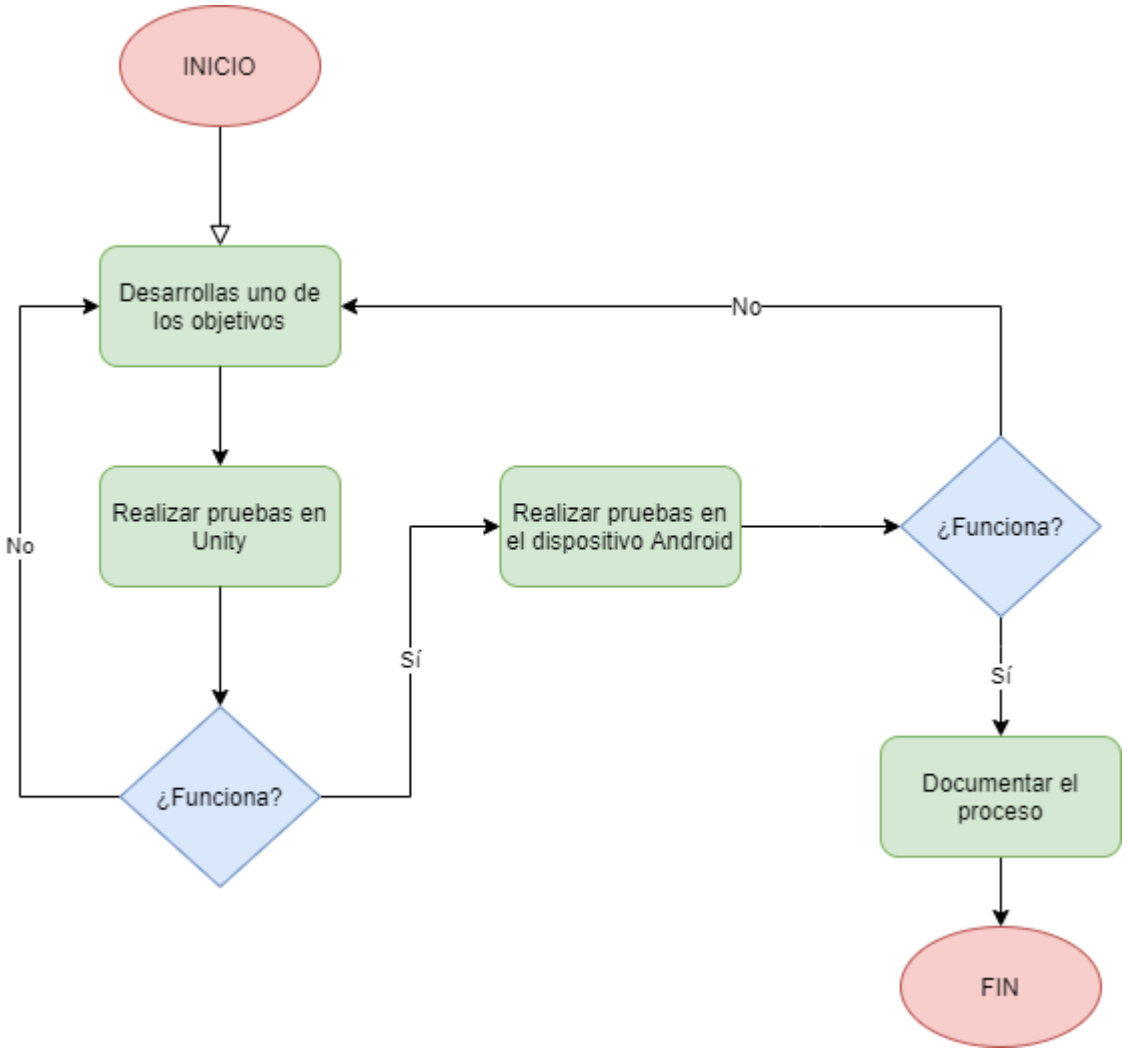


Figura 17. Diagrama de flujo del desarrollo de los objetivos propuestos. Fuente: Autor.

4. DESARROLLO DEL PROYECTO

4.1 PRIMEROS PASOS

Una vez se ha elegido el entorno de desarrollo con el que se va a trabajar, se pasa a nombrar los equipos a los que se tiene acceso para realizar este trabajo. El equipo con el que se va a desarrollar la aplicación es el propio ordenador portátil, un Lenovo Ideapad 700-15isk y estas son sus especificaciones.

Nombre del equipo:	LAPTOP-INB796OK
Sistema operativo:	Windows 10 Home 64 bits (10.0, compilación 18362)
Idioma:	español (configuración regional: español)
Fabricante del sistema:	LENOVO
Modelo del sistema:	80RU
BIOS:	E5CN53WW
Procesador:	Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz (8 CPUs), ~2.6GHz
Memoria:	8192MB RAM
Archivo de paginación:	7775MB usados, 3130MB disponibles
Versión de DirectX:	DirectX 12

Figura 18. Características del PC. Fuente: Autor

Para realizar las pruebas, se utiliza un smartphone, y este será el Xiaomi MiA2 Lite. En este caso la versión de la que se dispone es la que cuenta con 3GB de memoria RAM y 32GB de almacenamiento. Estas son sus especificaciones.

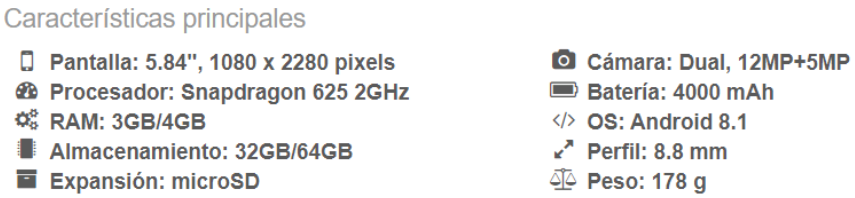


Figura 19. Características del smartphone. Fuente: Autor

4.1.1 Instalación de Unity

Una vez nombrados los equipos con los que se va a trabajar, el primer paso que se realizará será instalar la herramienta elegida, en este caso, Unity. Se descargará el *Unity Hub* en el siguiente enlace: https://unity3d.com/es/get-unity/download

Una vez instalado, lo primero que se hace es crear un usuario en la plataforma, y activar una licencia, en este caso una licencia *Unity Personal* que es una licencia

gratuita que puede ser utilizada como uso no profesional, o en entidades que no superen los 100.000$ de ingresos anuales. Ahora que ya se tiene la licencia se va a instalar la versión de Unity que se desea, en este caso se instalará Unity 2017.4.36f1.

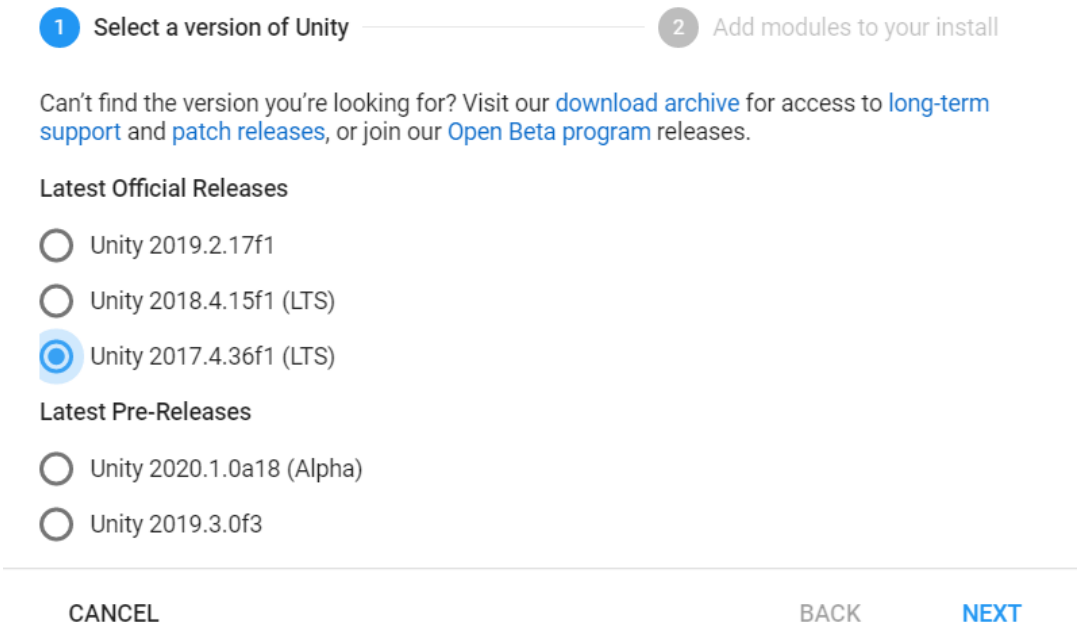


Figura 20. Selección de la versión de Unity. Fuente: Autor

Además de esto, también se selecciona que plataformas se quieren instalar junto a la versión de Unity y en este caso se selecciona 'Android Build Support' y 'Vuforia Augmented Reality Support' que son las que necesitaremos para la realización del proyecto.

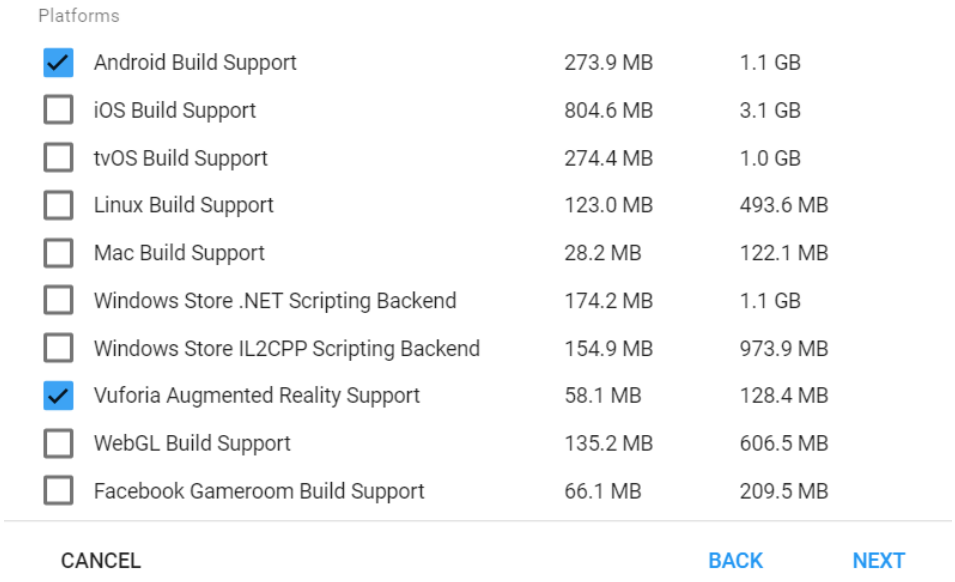


Figura 21. Instalación Vuforia Augmented Reality Support y Android Build Support. Fuente: Autor

4.1.2 Instalación de Android SDK y Java Development Kit (JDK)

Para poder crear proyectos que sean utilizados en dispositivos Android, se necesita también instalar tanto Android SDK como Java Development Kit (JDK). En este paso se instalará el SDK de Android, aunque se puede descargar e instalar por separado. La opción más sencilla es hacerlo mediante Android Studio, así que el primer paso será instalar Android Studio. Para eso lo primero que se hará es ir al siguiente enlace para descargar Android Studio: https://developer.android.com/studio.

Al instalar Android Studio, se puede ver en la siguiente imagen como Android SDK Plataform se instala a la vez que el programa. Una vez instalado solo faltará instalar las APIs de las versiones de Android para las que se quiera programar.

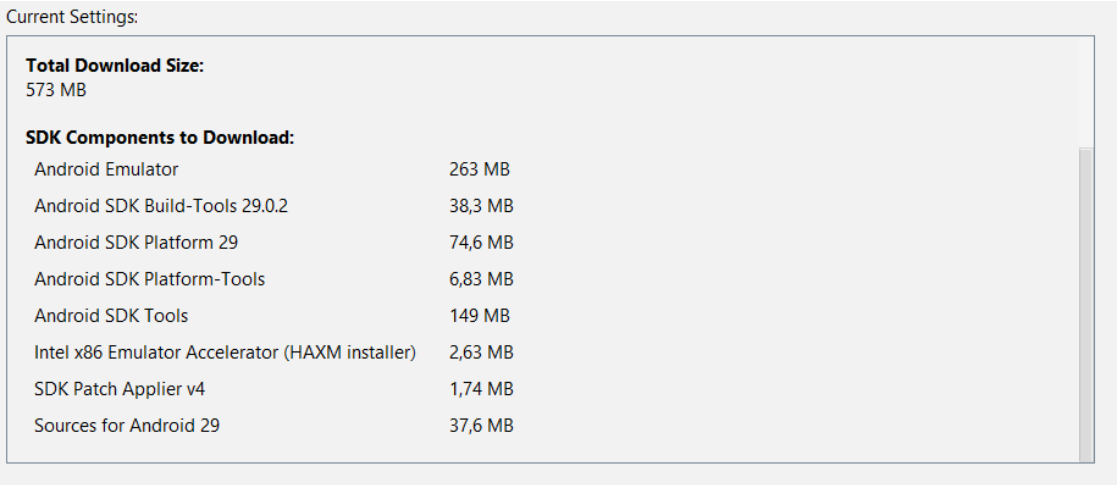


Figura 22. Instalación Android Studio. Fuente: Autor

Una vez instalado el SDK en el paso anterior, se procede a instalar el JDK. Para realizar la descarga se accede al siguiente enlace:

https://www.oracle.com/technetwork/java/javase/downloads/jdk13-downloads-5672538.html

En este caso se instalará la última versión disponible en la web, que es la 1.8.0.

Una vez están instalados estos dos componentes, el siguiente paso a seguir es poner dentro de Unity, donde se encuentran, tanto el JDK como el SDK de Android para que este pueda hacer uso de ellos. Para esto se va a *Edit >> Preferences >> External Tools* y aquí se indican las rutas donde se encuentren estos componentes.

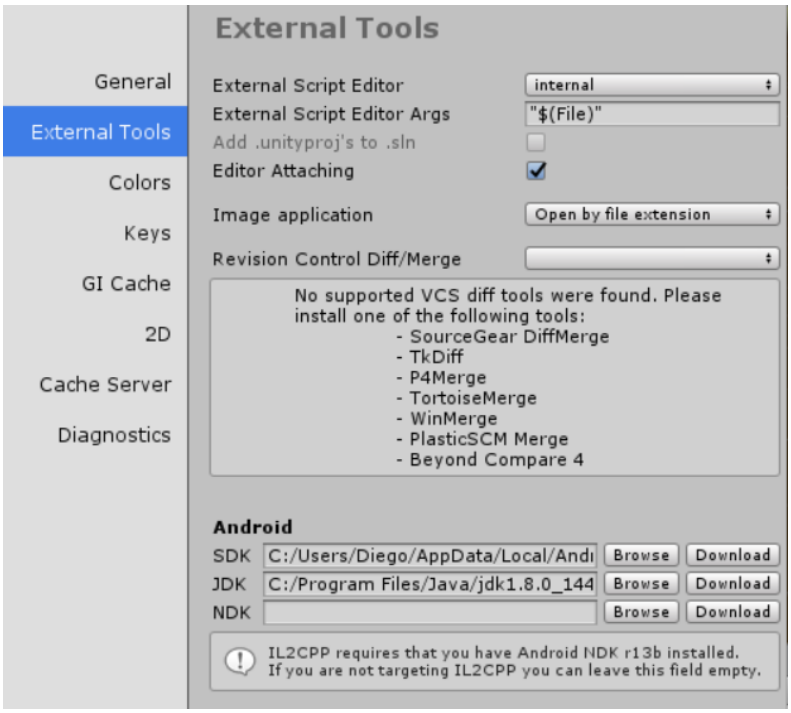


Figura 23. Indicando a Unity donde se encuentran el SDK y JDK. Fuente: Autor

4.1.3 Añadir targets al proyecto con Vuforia Development Portal

Para añadir los targets que se van a utilizar, Vuforia ofrece la posibilidad de hacerlo, accediendo a https://developer.vuforia.com/ . Una vez se ha hecho el registro en este portal, lo primero que se hace es elegir el tipo de licencia que se quiere utilizar. Vuforia ofrece dos tipos de licencia: Development Key y Deployment Key. Development Key es una licencia gratuita que ofrece Vuforia con el inconveniente de tener una marca de agua. Así mismo, Deployment Key es la versión de pago, con la que se eliminaría dicha marca de agua. En este caso se utilizará la versión gratuita, Development Key. Con esta versión se puede añadir una o varias bases de datos, con varios targets dentro de cada base de datos.

Una vez se tiene la licencia, se crea una base de datos y añade los targets que se quieran. Como se puede ver en la siguiente imagen, los targets pueden ser imágenes, cuboides, cilindros u objetos 3D.

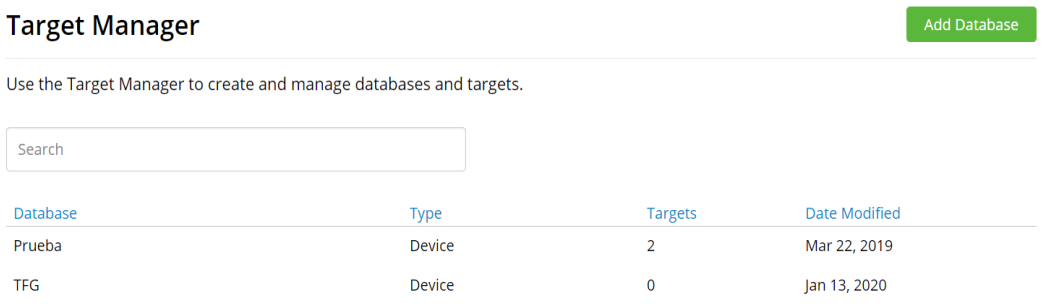


Figura 24. Creación de la Base de Datos. Fuente: Autor

Después de haber añadido los targets que se querían añadir, se descarga la base de datos y se selecciona Unity Editor como plataforma de desarrollo, para que así esta base de datos sirva en el proyecto.

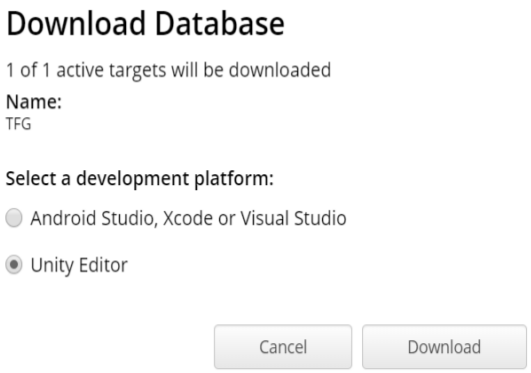


Figura 25. Descarga de la Base de Datos. Fuente: Autor

4.1.4 Elección del tipo de target

Como se ha comentado anteriormente, para activar los contenidos en Realidad Aumentada se necesitan targets. Estos pueden ser marcadores, objetos, geolocalización etc. Y más en concreto Vuforia permite añadir imágenes, objetos 3D, cilindros y cubos.

Figura 26. Añadir *target* desde Vuforia. Fuente: Autor.

En este caso se han seleccionado imágenes como targets y es por los siguientes motivos. Son los tipos de target más fáciles de obtener dentro de Vuforia y su funcionamiento es muy bueno. Además, ya que se pretende desarrollar un sistema para que un técnico, sin necesidad de conocimientos de programación pueda implementar casos de asistencia de forma sencilla, también se pretende que los targets sean sencillos de obtener para facilitar dicha implementación.

4.1.5 Primer ejemplo y toma de contacto

Teniendo instalado todo lo necesario, lo primero que se hace es crear un proyecto. Se crea un nuevo proyecto de nombre 'TFG'. Este será un proyecto de prueba, para crear un ejemplo sencillo, con el que comprobar que todos los elementos que se ha instalado funcionan correctamente, y que al ejecutar la aplicación en el smartphone la app funciona como se espera. Además, también servirá como toma de contacto con la herramienta Unity.

Con el proyecto creado, lo primero que se hace es activar Vuforia Augmented Reality en *XR Settings*, se llega aquí a través de esta ruta: *File>>Build Settings>>Player Settings.*

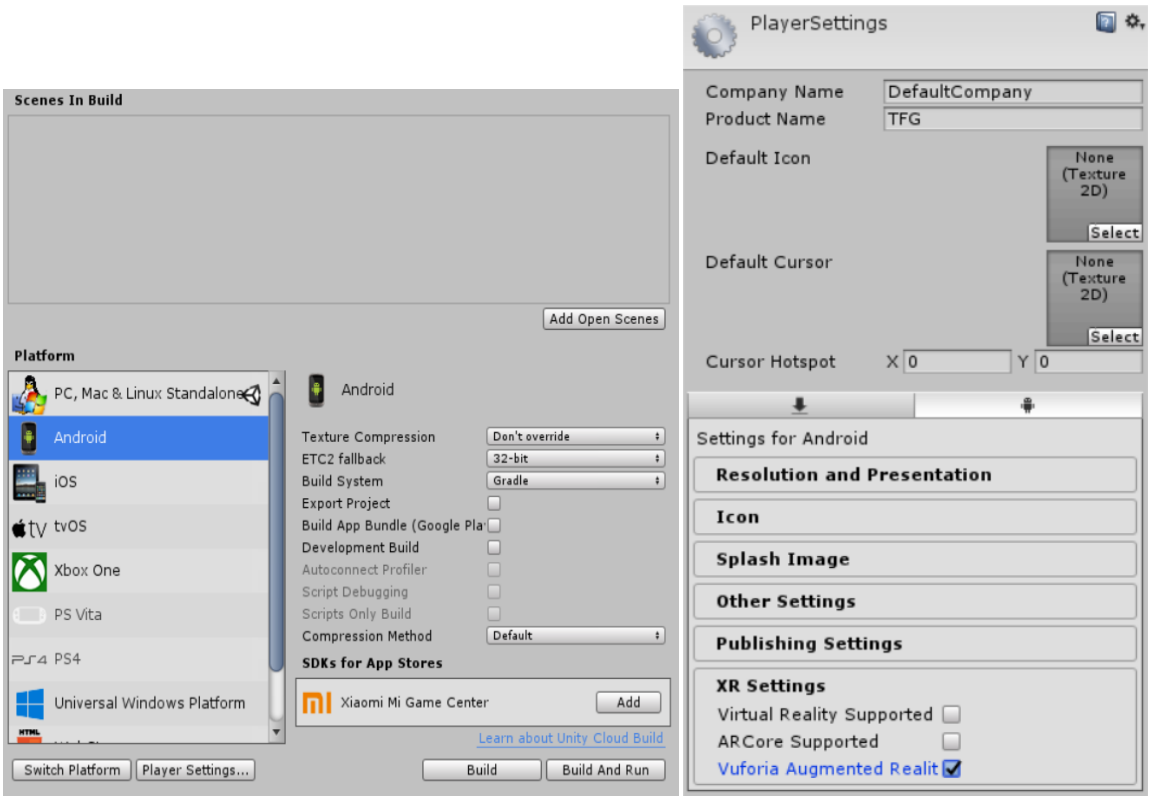


Figura 27. Activación de Vuforia Augmented Reality en Unity. Fuente: Autor

Ahora a través de *Assets>>Import Package>>Custom Package*, se importa la base de datos que se ha creado anteriormente. Como se puede ver en la siguiente imagen, al importar la base de datos se importan los archivos .jpg o .png que se hayan puesto previamente en la base de datos ya también .dat y .xml que son los que Vuforia utilizará para el reconocimiento de los targets, en este caso, imágenes. [Figura 28]

Ahora se elimina la *Main Camera* que aparece en la ventana *Hierarchy* y se añade la AR Camera que se encuentra en *GameObject>>Vuforia*. Además, también se añade un Imagen Target (que es donde más tarde se pondrá la imagen que se ha importado en la base de datos). Todo lo que se añade se puede ver en la ventana Hierarchy.

Seguidamente, en la configuración de Vuforia, se añade la licencia (Developer Key) que se ha creado previamente en la web de Vuforia y se activa base de datos que se habían importado. [Figura 29]

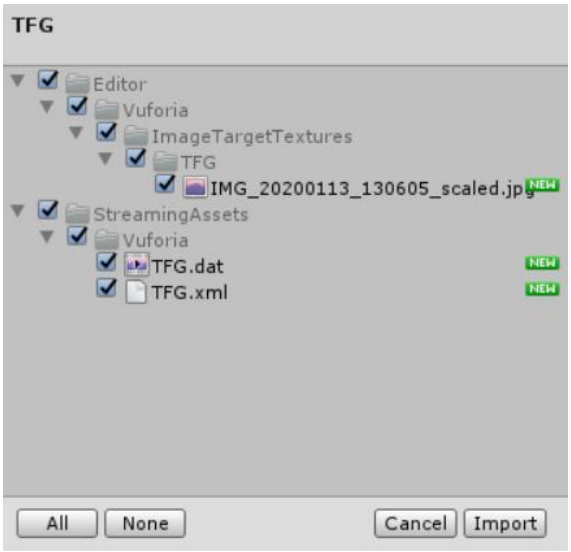


Figura 28. Importación Base de Datos. Fuente: Autor

Figura 29. Licencia de Vuforia en Unity. Fuente: Autor

Una vez está todo preparado, se añade la imagen que se tienen en la base de datos a Imagen Target y también un modelo 3D (en este caso de un muñeco) que se arrastra dentro del Imagen Target, para que aparezca cuando se reconoce esta imagen.

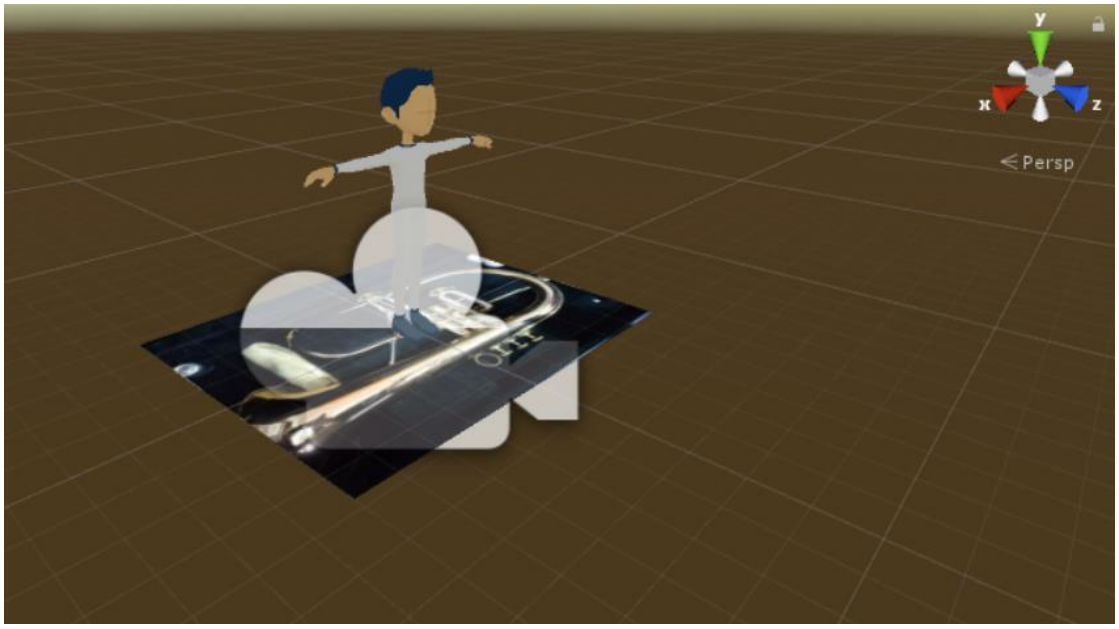


Figura 30. Editor de Unity con los componentes de la aplicación ya añadidos. Fuente: Autor

Después de esto, se compila la aplicación y una vez compilada, se copia el .apk al dispositivo móvil, se instala y se ejecuta.

Como se puede observar, la aplicación funciona como se espera, y nada más ejecutarla y dirigir la cámara hacia el objeto del que se había hecho la fotografía, aparece superpuesto el modelo 3D.

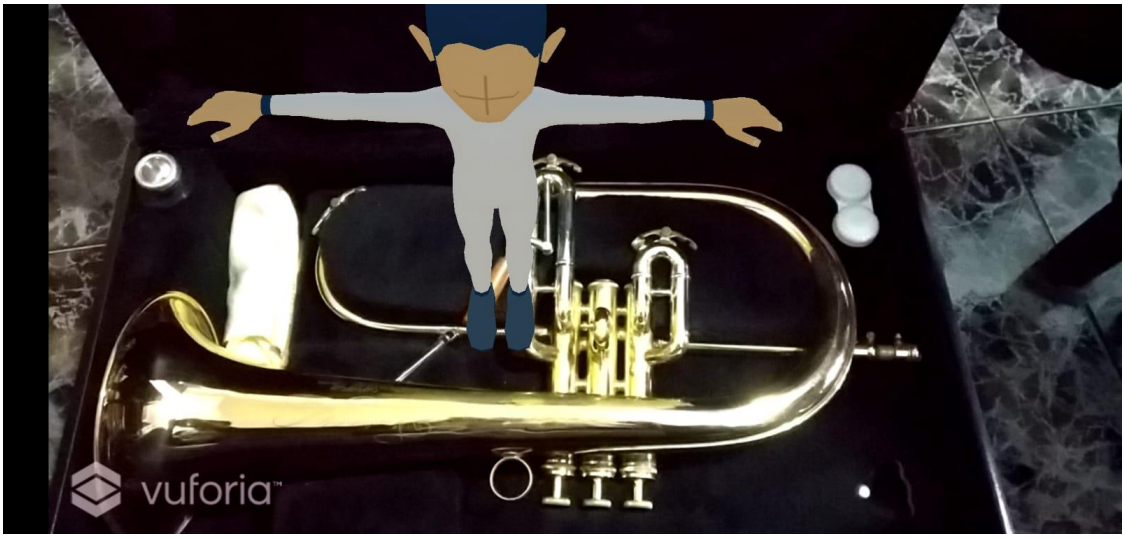


Figura 31. Primera ejecución de la aplicación de prueba. Fuente: Autor

4.1.6 Preparación del smartphone

Para poder utilizar el smartphone para ejecutar la aplicación, sin tener que compilarla, instalarla y ejecutarla en él cada vez, Unity ofrece la posibilidad de hacerlo conectando el teléfono móvil al PC donde se esté desarrollando la aplicación.

Primero se habilita la depuración USB en el terminal.

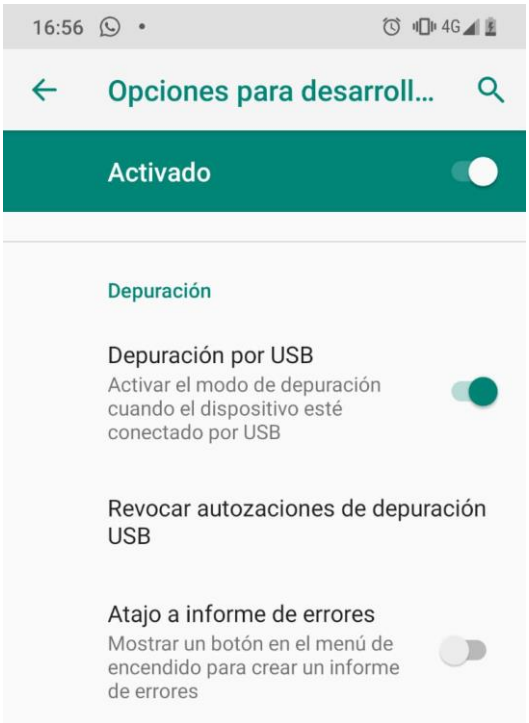


Figura 32. Activación depuración USB en smartphone. Fuente: Autor

Una vez se tiene esto se descarga la aplicación ‘Unity Remote 5’ que se encuentra en la Play Store de Android.

Figura 33. Instalación Unity Remote 5. Fuente: Autor

Con la aplicación ya instalada, lo único que se hace en el dispositivo móvil es ejecutarla.

Por la parte de Unity para poder realizar esta ejecución, se selecciona en ‘Unity Remote’ el dispositivo de ejecución. Se selecciona en *Edit* >> *Project Settings* >> *Editor* >> *Unity Remote* >> *Device* ‘Any Android Device’.

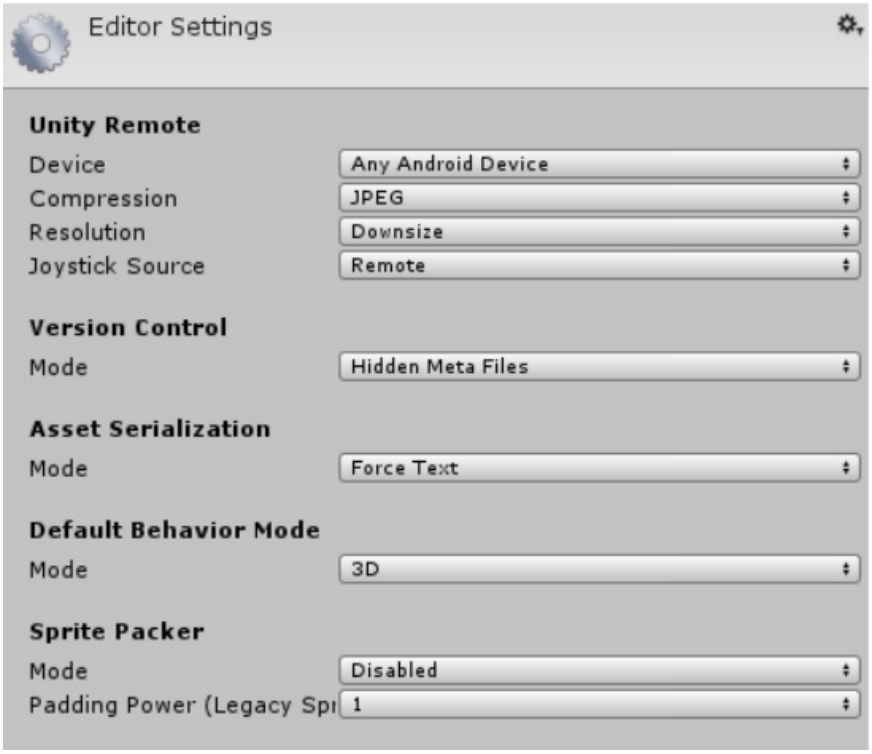


Figura 34. Seleccionar dispositivo en Unity. Fuente: Autor

Con todo lo anterior hecho y el dispositivo móvil conectado al ordenador mediante USB (en este caso con un cable USB Tipo B a MicroUSB) si se clica en el icono de 'Play' la aplicación se ejecutará en el smartphone.

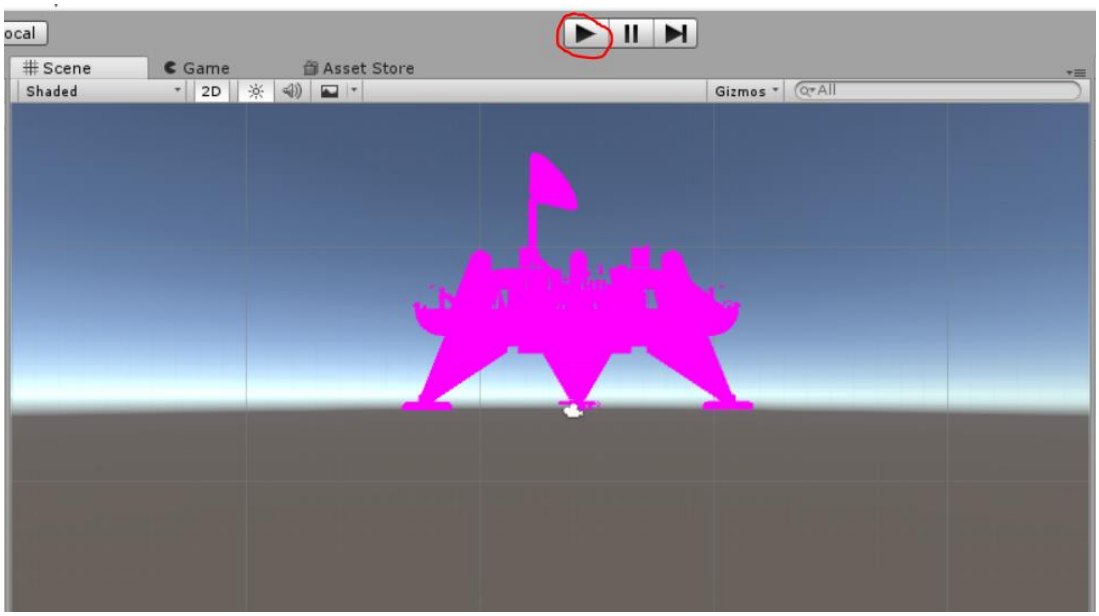


Figura 35. Botón Play en Unity. Fuente: Autor

4.1.6.1 Requisitos del dispositivo

El requisito que debe cumplir el smartphone en el que se quiera instalar la aplicación es tener una versión de Android igual o superior a la versión Android 4.1 'Jelly Bean'

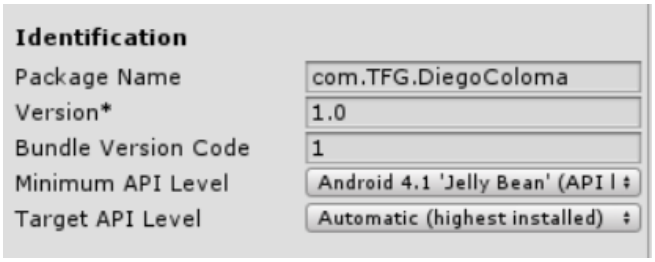


Figura 36. Versión mínima de Android. Fuente: Autor

4.2 IMPLEMENTACIÓN DEL SISTEMA PARA LA CREACIÓN DE CASOS DE ASISTENCIA SENCILLOS SIN LA NECESIDAD DE UTILIZAR CÓDIGO

Una vez finalizada toda la preparación del entorno de trabajo, y acabado el primer ejemplo que sirve como una toma de contacto con este entorno de trabajo, se procede a la implementación de la aplicación en sí misma. Se utiliza el ejemplo que antes se ha desarrollado para empezar la implementación propiamente dicha ya que en este ejemplo

está todo preparado para poder empezar. Se borran los elementos que se habían añadido (target y modelo 3D) y ya se puede empezar a trabajar sobre este proyecto como si de uno nuevo se tratase.

Uno de los objetivos de este TFG es desarrollar un sistema para que un técnico sin conocimientos de programación pueda crear un caso de asistencia sencillo de una forma rápida y fácil.

Para empezar esta implementación, lo primero que se ha hecho es crear un GameObject en Unity para poner todos los elementos propios de este apartado y que así estén separados de las otras partes de la aplicación. Seguidamente, se ha importado una base de datos con un target para usar de ejemplo.

Para interactuar con la aplicación durante la asistencia se ha decidido poner unos botones, en concreto, un botón para empezar la ejecución y dos botones de adelante y atrás para poder iterar entre los distintos pasos con los que cuente el caso de asistencia que se vaya a ejecutar.

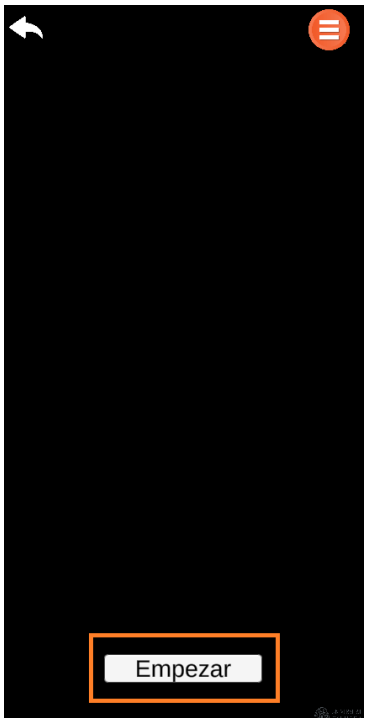


Figura 37. Botón Empezar. Fuente: Autor

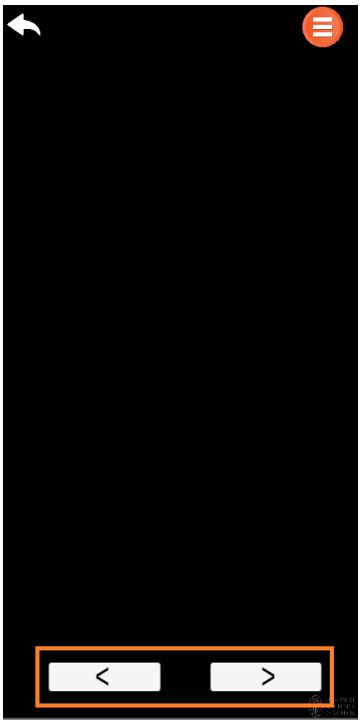


Figura 38. Botones adelante y atrás. Fuente: Autor

Ya que los casos de asistencia que se van a resolver utilizando este sistema son casos sencillos, se ha decidido que los elementos que se puedan añadir para realizar las explicaciones necesarias en cada paso sean: un número ‘n’ de flechas y un texto.

Para la flecha se ha elegido un modelo 3D que se ha descargado del *Asset Store* de Unity, en concreto ‘Arrow WayPointer’. Además del modelo, al hacer esta importación se

cuenta con distintos materiales para poder cambiar visualmente las flechas si así se desea. Siempre que se hable de una flecha en este punto se hará referencia a un objeto de este tipo. Para el texto se ha elegido un objeto tipo *Text* al cual se puede cambiar el color para poner el que se desee.

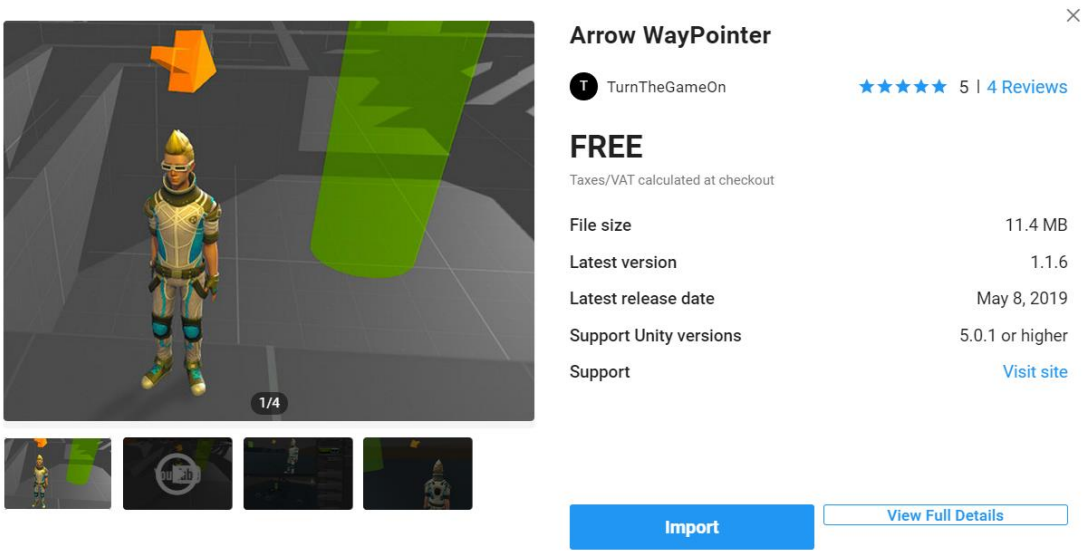


Figura 39. Arow WayPointer en el Asset Store. Fuente: Autor.

Una vez seleccionados los elementos que se van a utilizar, se añaden tanto un objeto Text como el modelo 3D de la flecha que se acaba de importar. Se añaden en el Hierarchy haciendo que dependan del target que se ha añadido anteriormente para que así, al enfocar ese objeto con la cámara, aparezcan los elementos que ahora se han añadido.

Figura 40. Jerarquía de los elementos. Fuente: Autor.

Como siguiente paso se crea un *script* desde donde controlar e implementar el funcionamiento de este punto de la aplicación. El script se ha llamado Pasos.cs y se puede ver por completo en el apartado de Anexos.

En este script lo primero que se ha hecho es crear las variables y las clases que se van a necesitar.

FRAGMENTO DE CÓDIGO	FUNCIONAMIENTO

`public class Pasos : MonoBehaviour`
`{`
`  public TrackableBehaviour target;`
`  public GameObject flechageneral;`
`  public TextMesh variableTexto;`
`  private int contador = -1;`
`  private string nombre = "";`
`  private Button b_anterior;`
`  private Button b_siguiente;`
`  private Button b_empezar;`
`  private GameObject iniciar;`
`  public Text aviso;`
`  public List<Instruccion> Paso = new List<Instruccion>();`
	Esta es la creación de todas las variables que se utilizan tanto para relacionar con los elementos anteriormente añadidos (como por ejemplo el b_empezar) y las que se utilizarán para emplear el funcionamiento de esta parte.

`[Serializable] public class Flechas`
`{`
`  public Vector3 posicion = new Vector3();`
`  public Vector3 rotacion = new Vector3();`
`  [NonSerialized] public GameObject flecha;`
`}`
`[Serializable]`
`public class Instruccion`
`{`
`  public string texto;`
`  public List<Flechas> Flechas = new List<Flechas>();`
`}`
	Flechas: Esta clase sirve para indicar la posición y rotación de las flechas que se añadan.
	Instrucción: En esta clase hay una lista de objetos de tipo Flechas y un *string*. Estos serán los elementos que se añadirán con cada instrucción.

Tabla 2. Variables y clases del script Pasos.cs. Fuente: Elaboración propia.

Para que los botones creados anteriormente estén vinculados con sus correspondientes variables se les ha añadido un *tag* a cada uno de ellos con el que poder acceder a estos desde el script con la función *FindGameObjectWithTag*. Con esta función en el *Start()* del script se han vinculado los tres botones añadidos anteriormente con sus variables correspondientes.

```
boton_empezar = GameObject.FindGameObjectWithTag("startButton").GetComponent<Button>();
boton_adelante = GameObject.FindGameObjectWithTag("rightButton").GetComponent<Button>();
boton_atras = GameObject.FindGameObjectWithTag("leftButton").GetComponent<Button>();
```

Figura 41. Vinculación de los botones con sus variables en el script Pasos.cs. Fuente: Autor

Para que las instrucciones se puedan añadir de una forma interactiva lo que se ha hecho es crear la clase “Instrucción” y hacerla *Serializable*. Con esto se consigue que en el *Inspector* de Unity al añadir el script al que pertenece la clase se puedan dar valores a sus variables desde el mismo Inspector, sin necesidad de hacerlo dentro del script. Esta clase tiene como variables un string de nombre texto, y una lista de objetos tipo “Flechas”. “Flechas” es otra clase que se ha creado (también definida como Serializable) que contiene dos *Vector3* llamados “posición” y “rotación” y un GameObject llamado

“flecha”. Los dos vectores serán los valores de posición y rotación que se podrán dar a la flecha que se creará. El GameObject llamado “flecha” y declarado *NonSerialized* (para que no aparezca en el Inspector), será una variable que servirá para la creación de cada flecha que se quiera añadir para cada paso de la instrucción. Para poder añadir más de un paso a cada caso de asistencia se crea una lista de objetos tipo ‘Instrucción’ de nombre ‘Lista’.

Con todo esto al añadir el script a un objeto en Unity y pinchar sobre él, en el Inspector aparece la posibilidad de añadir sobre un target tantos pasos como se quieran añadir, y en cada paso, se puede añadir un texto y tantas flechas como se quieran.

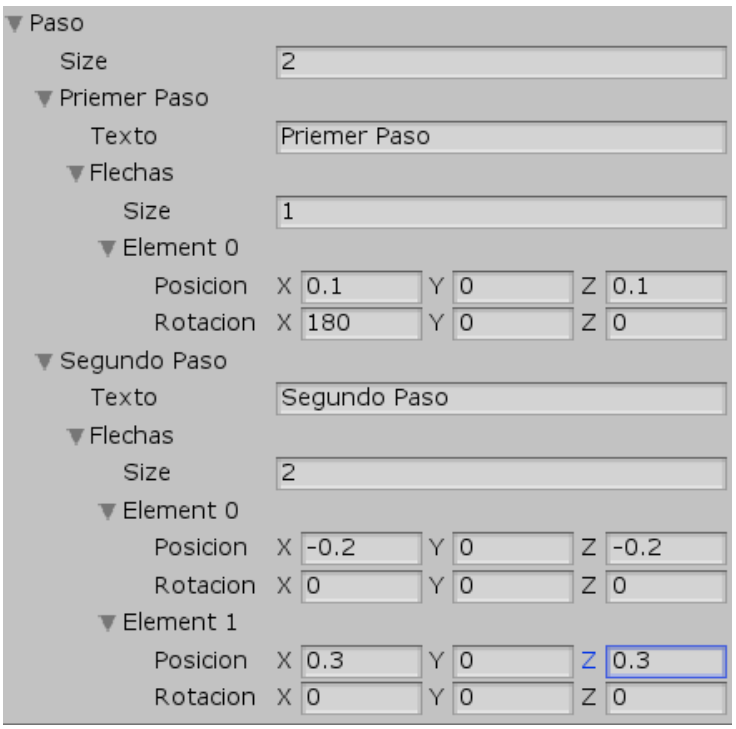


Figura 42. Añadido de instrucciones interactivo en el Inspector. Fuente: Autor

Para crear todas las flechas que se añadan en cada paso se ha creado en el método Start() del script un bucle anidado que recorre la lista de objetos ‘Flechas’ de la lista “Pasos” que crea una copia de la flecha añadida en un principio y se guarda en la variable “flecha” que tiene cada objeto tipo ‘Flechas’.

Se puede ver en el siguiente fragmento de código como en el método *Update()* se guarda en la variable “nombre” el nombre del target *trackeado* (si lo hay) en cada instante. Esto sirve ya que, si en el mismo proyecto se han añadido otros targets, las indicaciones solo deben salir cuando la cámara enfoque el target que se haya elegido objetivo, y teniendo guardado el nombre de dicho target en la variable, se puede controlar esto.

```
private void Update()
{
    StateManager sm = TrackerManager.Instance.GetStateManager();
    IEnumerable<TrackableBehaviour> tbs = sm.GetActiveTrackableBehaviours();

    foreach (TrackableBehaviour tb in tbs)
    {

        if (tb.CurrentStatus == TrackableBehaviour.Status.TRACKED)
        {
            nombre = tb.TrackableName;

        }

    }
}
```

Figura 43. Update() del script Pasos.cs. Fuente: Autor.

Para controlar como se cambia entre los distintos pasos de la instrucción se ha creado la corutina *CambiarPaso().* Lo primero que se hace es comprobar el nombre del objeto trackeado actualmente para comprobar que es sobre el que se han creado las instrucciones. Una vez hecho esto, utilizando condicionales se comprueba el valor de la variable "contador" que es la que determinara que paso se debe mostrar. Para que se muestren en las instrucciones del paso al que se cambia al ejecutar la corutina es, mediante la función *setActive()*, activar todas las flechas y el texto del paso al que se avanza y desactivar los de los pasos anterior y posterior (si es el primer paso solo se desactiva el posterior, y si es el último, el anterior). A su vez, utilizando el atributo *interactable* (que sirve para habilitar o deshabilitar la capacidad de ser seleccionado o pulsado de un botón) se actúa sobre los botones que sirven para iterar entre los pasos y se deshabilita esta capacidad en el botón b_anterior cuando estamos en el primer paso, y se deshabilita del botón b_siguiente cuando estamos en el último.

Fragmento de código	Funcionamiento

`if (contador == 0)`
`{`
`  for (int j = 0; j < (Paso[contador + 1].Flechas.Count); j++)`
`  {`
`  Paso[contador + 1].Flechas[j].flecha.SetActive(false);`
`  }`
`  for (int j = 0; j < Paso[contador].Flechas.Count; j++)`
`  {`
`  Paso[contador].Flechas[j].flecha.SetActive(true);`
`  variableTexto.text = Paso[contador].texto;`
`  }`
`  b_anterior.interactable = false;`
`  b_siguiente.interactable = true;`
`  yield return null;`
`}`
	Fragmento de código de la corutina CambiarPaso.cs del script Pasos.cs. En concreto, como se muestran las indicaciones cuando llegamos al primer paso del caso de asistencia.

Tabla 3. Fragmento de código del script Pasos.cs. Fuente: Elaboración propia.

También se tratan los casos de que haya un solo paso, o que no se haya creado ninguno. En el caso que no se haya creado ningún paso se muestra el mensaje: “No hay ningún paso creado” y si hay solo uno, se muestran sus flechas y su texto y se deshabilita la capacidad de ser pulsados de los dos botones, tanto b_siguiente como b_anterior. Con las funciones *BotonSiguiente()* y *BotonAnterior()* se cambia el valor de la variable “contador” con la que se controla que paso es el que se tiene que mostrar, y se llama a la corutina CambiarPaso*()*. Estas funciones se ejecutan al pulsar el b_siguiente y b_anterior respectivamente.

Modificando el script *DefaultTrackableEventHandler.cs* que es un script que todos los objetos *trackeables* tienen por defecto, se ha añadido una parte de código para que el b_empezar pueda ser seleccionable (con el atributo interactable) ya que en un principio este está deshabilitado y a su vez que cuando el target no sea detectado por la cámara, aparezca un mensaje que indique que no se encuentra dicho target i que se vuelva a enfocar. Además de esto, también se esconden los botones que se encargan de la iteración entre los pasos (b_siguiente y b_anterior) cuando la cámara pierde el target.

Fragmento de código	Objetivos

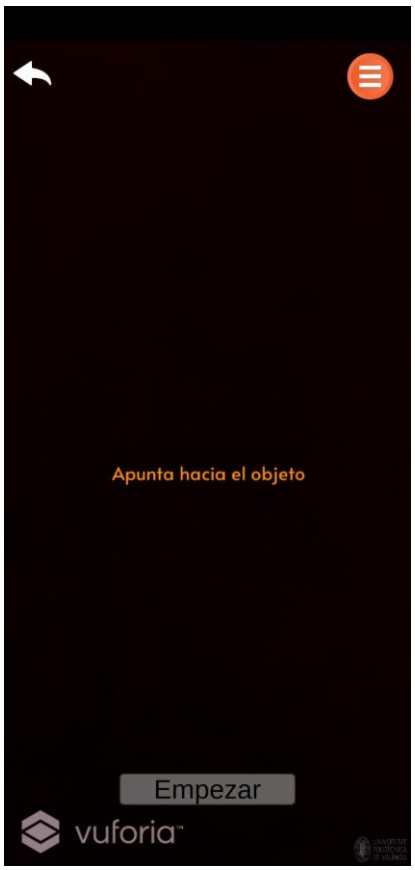
`public Text aviso;`
`public Button b_empezar;`
`public GameObject botones;`
 	
`public void OnTrackableStateChanged(`
`  TrackableBehaviour.Status previousStatus,`
`  TrackableBehaviour.Status newStatus)`
`{`
`  m_PreviousStatus = previousStatus;`
`  m_NewStatus = newStatus;`
 	
`  Debug.Log("Trackable " + mTrackableBehaviour.TrackableName +`
`  " " + mTrackableBehaviour.CurrentStatus +`
`  " -- " + mTrackableBehaviour.CurrentStatusInfo);`
 	
`  if (newStatus == TrackableBehaviour.Status.DETECTED ||`
`  newStatus == TrackableBehaviour.Status.TRACKED ||`
`  newStatus == TrackableBehaviour.Status.EXTENDED_TRACKED)`
`  {`
`  aviso.text = "";`
`  b_empezar.interactable = true;`
`  botones.SetActive(true);`
`  OnTrackingFound();`
`  }`
`  else if (previousStatus == TrackableBehaviour.Status.TRACKED &&`
`  newStatus == TrackableBehaviour.Status.NO_POSE)`
`  {`
`  aviso.text = "Vuelve a enfocar el objeto";`
`  b_empezar.interactable = false;`
`  botones.SetActive(false);`
`  OnTrackingLost();`
 	
`  }`
`  else`
`  {`
`  // For combo of previousStatus=UNKNOWN + newStatus=UNKNOWN|NOT_FOUND`
`  // Vuforia is starting, but tracking has not been lost or found yet`
`  // Call OnTrackingLost() to hide the augmentations`
`  OnTrackingLost();`
`  }`
`}`
	Realizar las acciones de mostrar y esconder los botones deseados al encontrar o perder la cámara el target.
	
	Para realizar dichas acciones se han creado tres variables. Una variable tipo Text, una variable tipo ‘Button’ y otra variable tipo GameObject.
	
	El método que se encuentra bajo la declaración de las variables *OnTrackableStateChanged()* permite saber si se tiene o no trackeado un target a través del parámetro *Status* de la clase *TrackableBehaviour.*

Tabla 4. Fragmento de código del script DefaultTrackableEventHandler.cs. Fuente: Elaboración propia.

Con todo esto ya se tiene preparado el sistema para la creación de casos de asistencia de forma interactiva desde el Inspector de Unity sin necesidad de tener conocimientos de programación. Como ejemplo se ha cogido una imagen del cuadro de mando de una máquina industrial. Esta imagen ha sido cogida de la página web de Ingeniarías Eléctricas Saubi S.L.[19]. Se ha elegido utilizar una imagen cogida de una web con el objetivo de demostrar que este sistema permite de una forma muy sencilla, con una imagen y especificando las indicaciones que se deseen en a través del Inspector, crear un caso de asistencia real.

Como se puede ver en las siguientes imágenes, dando valores a los elementos que se desea poner en el Inspector se logra crear un caso de asistencia de forma satisfactoria y funcional.

En la tabla siguiente se muestra un ejemplo de todos los pasos de un caso de asistencia creado con el sistema que se acaba de desarrollar.

1. Primera ejecución sin haber trackeado el target	**2. Antes de pulsar el botón Empezar**
	

19 *http://ingenieriasaubi.es/armarios*

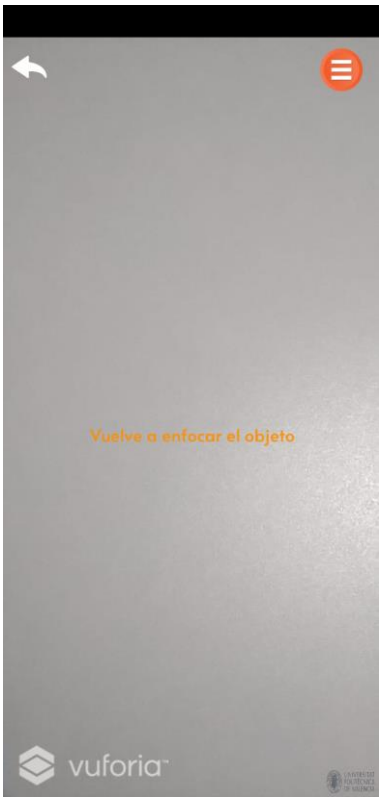
3. Primer Paso	4. Segundo Paso
	
5. Perdida del target	**6. Tercer Paso**
	

Tabla 5. Pasos del ejemplo creado con el sistema de creación de casos de asistencia sin necesidad de código. Fuente: Elaboración propia.

4.2.1 Botón *reset*

Ya que se pueden añadir tantos pasos como se desee, se ha decidido añadir un botón *reset* con el que poder volver al principio de la asistencia, sin necesidad de pasar por todos los pasos uno a uno para retroceder hasta ese punto. Pulsando este botón, estando en cualquiera de los pasos de la asistencia que se esté realizando, se resetea su estado y se vuelve al principio, donde se muestra el botón Empezar para volver a empezar la asistencia. Para lograr este funcionamiento se ha añadido un botón, que al pulsarlo ejecuta el método *Resetear()* que se encuentra en el script Pasos.cs.

Fragmento de código	`public void Resetear()`
	`{`
	`  if (contador != -1)`
	`  {`
	`  for (int j = 0; j < (Paso[contador].Flechas.Count); j++)`
	`  {`
	`  Paso[contador].Flechas[j].flecha.SetActive(false);`
	`  }`
	`  }`
	`  variableTexto.text = "";`
	`  contador = -1;`
	`  b_anterior.gameObject.SetActive(false);`
	`  b_siguiente.gameObject.SetActive(false);`
	`  b_empezar.gameObject.SetActive(true);`
	`  b_empezar.interactable = false;`
	`  aviso.text = "Vuelve a envfocar el objetivo";`
	`}`

Objetivo	Con este método se pretende volver al principio de la asistencia sin tener que retroceder por cada paso que se ha avanzado. Para lograr esto se desactivan todas las flechas además del texto que se muestra. El contador con el que se controla en que paso se está en cada momento se pone con valor '-1' que es con el que se inicializa en un principio. Se desactivan los botones b_siguiente y b_anterior, y se activa el botón b_empezar.

Tabla 6. Fragmento de código del script Pasos.cs. Método Resetear(). Fuente: Elaboración propia.

Para que se ejecute este método al pulsar el botón se ha añadido un evento en el *On Click()* de dicho botón (esto se puede encontrar en el Inspector), a este evento se le ha añadido el objeto que tiene contiene el script Pasos.cs y se ha seleccionado el método Resetear() de este script.

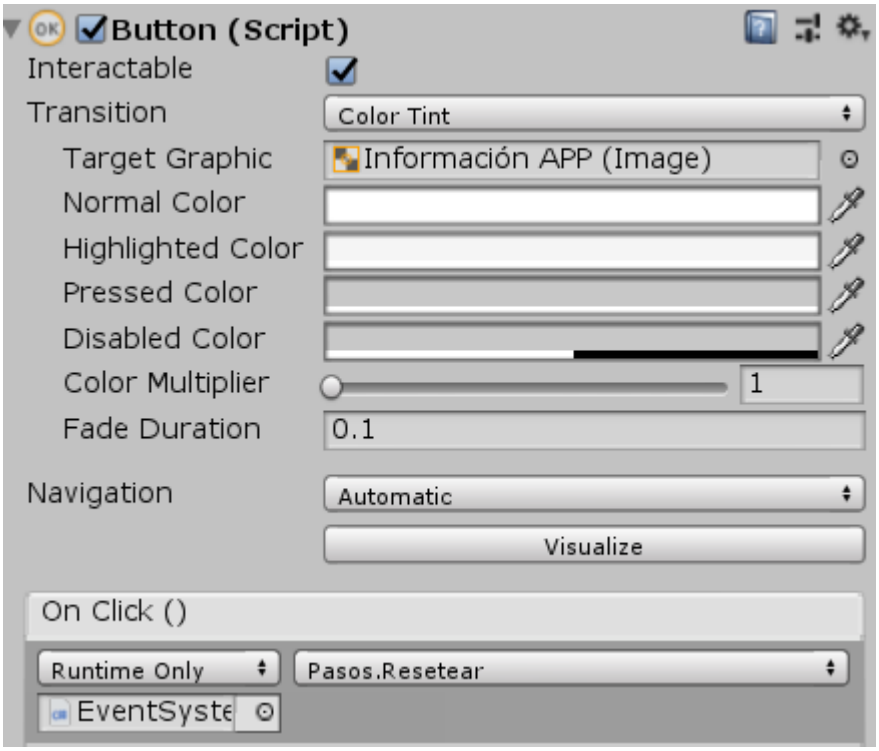


Figura 44. Función *On Click()* del botón reset. Fuente: Autor.

4.3 IMPLEMENTACIÓN DE UN CASO DE ASISTENCIA CON UN GRADO DE COMPLEJIDAD QUE NECESITA DE LA UTILIZACIÓN DE CÓDIGO

Otro de los objetivos del proyecto es desarrollar un caso de asistencia que, por su grado de complejidad o necesidad de personalización, necesite por ejemplo el reconocimiento de dos targets distintos y con el que sea necesaria la utilización de código para su correcta implementación. Para el desarrollo de este caso se ha escogido crear esta asistencia para hacer la sustitución de la cámara de una Tablet de la marca Szenio, concretamente el modelo Szenio PC 2032QC. Para crear este caso de asistencia será necesario la utilización de dos targets distintos además de indicaciones más precisas, con distintos modelos y con animaciones que ayudan a la comprensión de la tarea que se ha de realizar.

Lo primero que se ha hecho es añadir los dos targets que se van a utilizar en esta implementación. En este caso los dos targets son los que se muestran en las siguientes figuras. La primera corresponde a la tapa del dispositivo, y la segunda al interior del mismo, donde ya se pueden ver sus componentes.

Figura 45. Primer target de la Tablet Szenio. Fuente: Autor.

Figura 46. Segundo target de la Tablet Szenio. Fuente: Autor.

Después se han añadido todos los elementos necesarios para crear las instrucciones deseadas. Sumado al asset 'Arrow WayPointer' importado en el paso anterior, también se ha importado para esta implementación 'Workplace Tool' que como el anterior está disponible para descargar en el Asset Store de forma gratuita.

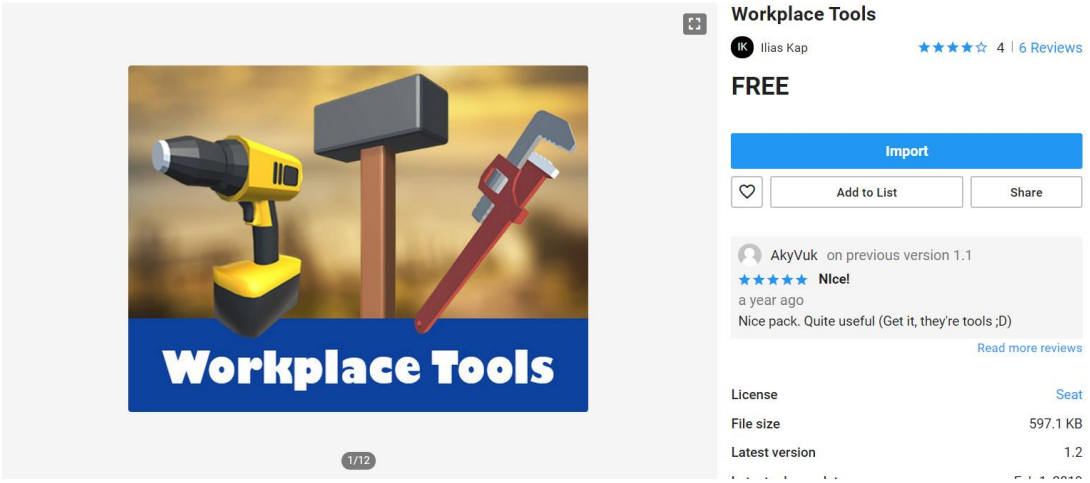


Figura 47. Elemento Workplace Tools. Fuente: Autor.

Una vez añadidos los elementos, se pueden empezar a crear las instrucciones. Primero se ha creado un GameObject por cada paso que se quiere crear y se han puesto dentro de cada target para que cada paso dependa del target sobre el que tendrá que aparecer. Seguidamente se añaden todos los elementos que se utilizan para dar las instrucciones dentro de cada paso.

Figura 48. Jerarquía de la implementación. Fuente: Autor.

Este tipo de implementación permite (además de utilizar varios targets) combinar varios tipos de indicadores, además de añadir animaciones y personalizar todo lo deseado las instrucciones que se van a dar en cada paso. Por esto, se han creado animaciones para algunos de los elementos que intervienen en cada uno de los pasos.

Al igual que con los casos de asistencia creados con el sistema implementado en el punto anterior, para iterar entre los distintos pasos se utilizarán los botones creados anteriormente con este fin.

Para implementar esta iteración se ha creado el script PasosConCodigo.cs. Para controlar el paso en el que se encentra la asistencia, este script funciona de una forma muy parecida. Primero se comprueba el target que se está trackeando y seguidamente mediante una variable se sabe que paso hay que mostrar. Una vez sabido el paso a mostrar, se activa el objeto que contiene todos los elementos de dicho paso, y se desactivan los restantes. Los botones siguen el mismo comportamiento que en el paso anterior.

Fragmento de código	Objetivos

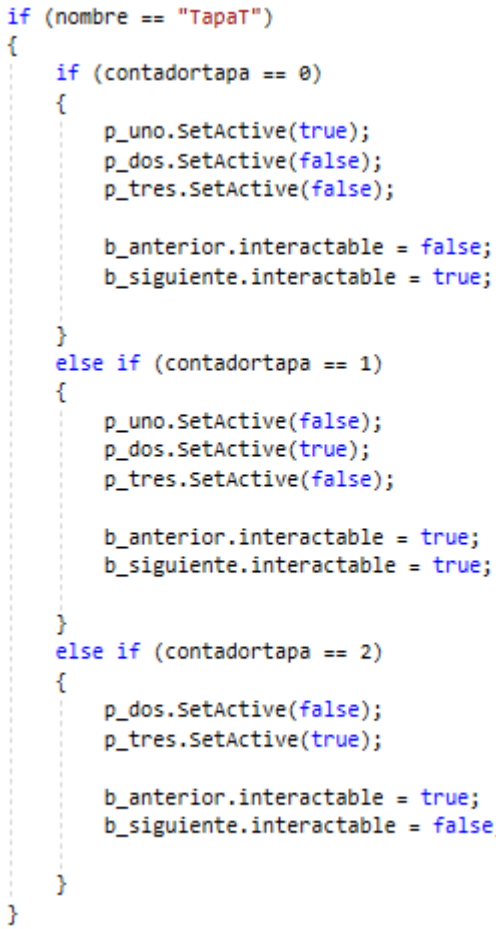

`if (nombre == "TapaT")`
`{`
`  if (contadortapa == 0)`
`  {`
`  p_uno.SetActive(true);`
`  p_dos.SetActive(false);`
`  p_tres.SetActive(false);`
`  b_anterior.interactable = false;`
`  b_siguiente.interactable = true;`
`  }`
`  else if (contadortapa == 1)`
`  {`
`  p_uno.SetActive(false);`
`  p_dos.SetActive(true);`
`  p_tres.SetActive(false);`
`  b_anterior.interactable = true;`
`  b_siguiente.interactable = true;`
`  }`
`  else if (contadortapa == 2)`
`  {`
`  p_dos.SetActive(false);`
`  p_tres.SetActive(true);`
`  b_anterior.interactable = true;`
`  b_siguiente.interactable = false;`
`  }`
`}`
	Mostrar y esconder las indicaciones según el paso en el que se encuentre el caso de asistencia.
	Además de esto controlar el estado de los botones.

Tabla 7. Fragmento de código del script PasosConCodigo.cs. Fuente: Elaboración propia.

La diferencia entre los dos scripts es que esta iteración, en el script Pasos.cs se controla en una corutina llamada desde los métodos que se ejecutan al pulsar los botones que iteran entre los pasos, sin embargo, en PasosConCodigo.cs esta iteración es controlada en el método Update(). Esta diferencia se debe a que este segundo caso

de asistencia es más complejo y cuenta con dos targets. El control se hace desde el método Update() para que al cambiar del primer al segundo target, automáticamente aparezca el primer paso relacionado con este segundo target, sin tener que pulsar ningún botón haciendo así el proceso más sencillo para el usuario. A su vez, también se ajusta el estado de los botones adelante y atrás. De esta forma, los métodos *BotonSiguiente()* y *BotonAnterior()* son muy parecidos a los métodos con el mismo nombre de Pasos.cs con la particularidad que no llaman a ninguna corutina, y que al comprobar el nombre del objeto trackeado lo comparan con los nombres de los targets de este caso de asistencia (‘TapaT’ y ‘Tablet’ concretamente).

Para que esto métodos se accionen, se añaden a los botones de interacción (botón Anterior y botón Siguiente).

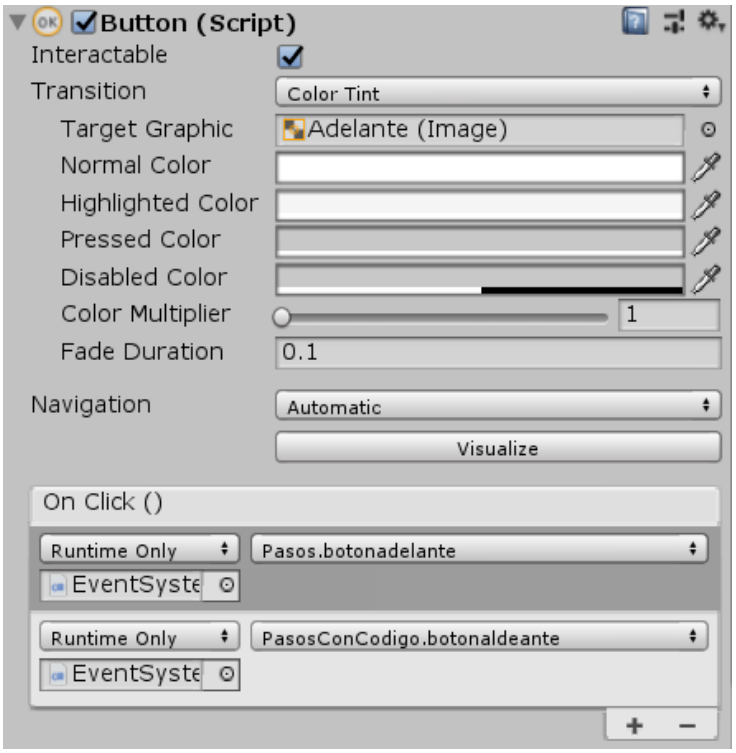


Figura 49. Función *On Click()*. Fuente: Autor.

Al hacer clic sobre estos botones se ejecutarán los métodos de ambos scripts, pero una vez se ejecutan, cada método comprueba el nombre del target que está visualizando la cámara en este momento, y solo realiza las acciones pertinentes si este nombre es el del target al que está dirigido el script donde se encuentra el método.

El script DefaultTrackableEventHandler.cs funciona también sobre cualquier target que es añada, por lo que con los que se han añadido también saldrán los avisos cuando la cámara los pierda de vista.

1. Target 1 - Paso 1	2. Target 1 - Paso 2
	
3. Target 1 - Paso 3	**4. Target 2 - Paso 1**
	

5. Target 2 - Paso 2	6. Target 2 - Paso 3
	

Tabla 8. Pasos del caso de asistencia implementado con código. Fuente: Elaboración propia.

4.3.1 Botón reset

Al igual que en el punto anterior se ha creado un método llamado Resetear() dentro del script PasosConCodigo.cs. Este método tiene un funcionamiento muy parecido al creado en el punto anterior con el mismo nombre. Actualiza las variables y les da el valor que tenían en su definición.

Fragmento de código	Objetivo

`public void Resetear()`
`{`
`contadortapa = -1;`
`contadortablet = 0;`
`b_anterior.gameObject.SetActive(false);`
`b_siguiente.gameObject.SetActive(false);`
`b_empezar.gameObject.SetActive(true);`
`b_empezar.interactable = false;`
`p_uno.SetActive(false);`
`p_dos.SetActive(false);`
`p_tres.SetActive(false);`
`p_unotablet.SetActive(false);`
`p_dostablet.SetActive(false);`
`p_trestablet.SetActive(false);`
`p_cuatrotablet.SetActive(false);`
`aviso.text = "Vuelve a enfocar el objetivo";`
`empezado = false;`
`}`
	Reiniciar las variables dándoles el valor con el que se inicializaron para volver al principio.

Tabla 9. Fragmento método Resetear() del script Pasos.cs. Fuente: Elaboración propia.

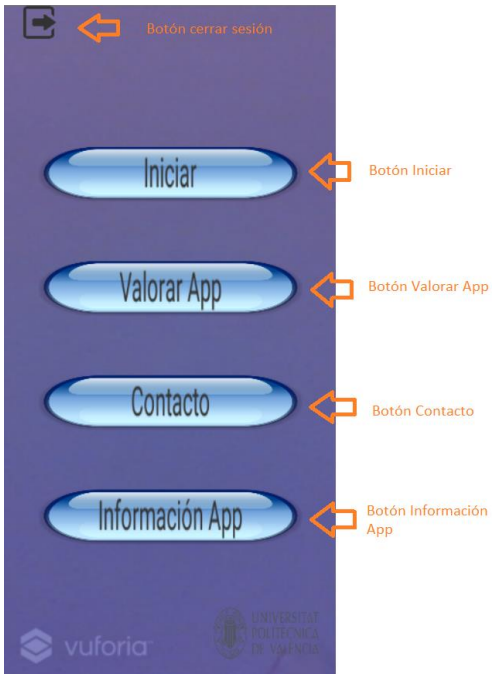
De la misma forma que se ha hecho para que el botón reset ejecute el método Resetear() del script Pasos.cs, se repite en este caso para el método con el mismo nombre del script PasosConCodigo.cs. De esta forma al pulsar el botón se resetearán ambos casos de asistencia, y los dos volverán al principio.

4.4 SISTEMA DE MENÚS

Para interactuar con la aplicación, se ha creado un sistema de menús que muestran al usuario en pantalla las distintas opciones que ofrece la aplicación y permite al mismo navegar por dentro de dicha app.

4.4.1 Menú Principal

El menú principal es la ventana a la cual se accede después de haber hecho el login y desde él, se accede al resto de submenús de la aplicación que son los siguientes:



- **Iniciar**: Es el botón que se utiliza para acceder al contenido propio de la aplicación.

- **Valorar App**: A través de este botón accedemos al submenú valorar app.

- **Contacto**: Con este botón se accede a un submenú con información de contacto.

- **Información App**: Con este botón se accede a un submenú con información sobre la aplicación.

- **Botón cerrar sesión**: Con este botón cerramos la sesión y salimos a la escena del login.

Figura 50. Menú Principal. Fuente: Autor.

Como *background* de este sistema de menús se ha elegido poner una imagen de color (azul en este caso) con cierto grado de transparencia para que deje ver el contenido que está captando la cámara la cual se deja activa desde el momento en el que se ejecuta la aplicación.

Dentro del Hierarchy de Unity se han agrupado todos los elementos del sistema de menús en un *Canvas* para tenerlos separados de los elementos de Realidad Aumentada propios de la aplicación.

Figura 51. Jerarquía de la aplicación. Fuente: Autor

Además de esto, como este sistema de menús solo está preparado para la utilización con el dispositivo móvil orientado verticalmente, se ha forzado a que la orientación de la aplicación siempre sea vertical y no cambie al girar el dispositivo. Para esto se ha seleccionado la opción *Portrait* en el campo *Default Orientation* al cual se accede en *Build Settings >> Player Settings >> Resolution and Presentation.* Seleccionando Portrait ser refiere a que el dispositivo estará en posición vertical con el botón *Home* en la parte inferior.

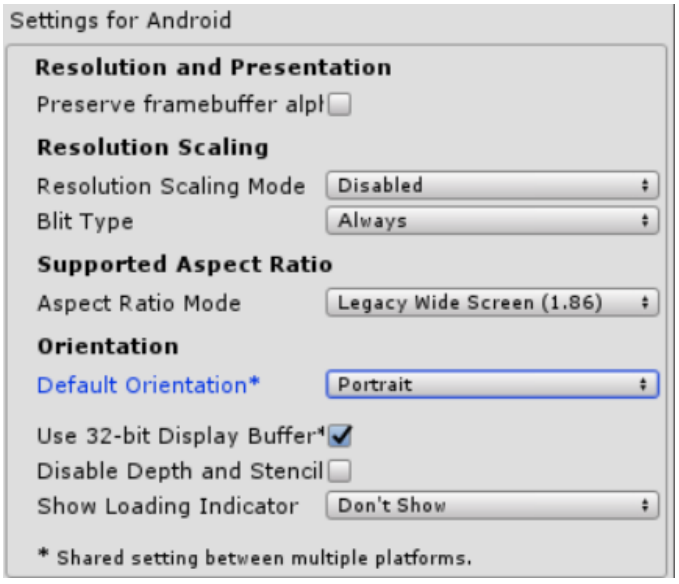


Figura 52. Orientación. Fuente: Autor

Para navegar entre los diferentes menús, se han añadido funciones que ocultan y muestran los diferentes menús al pulsar los botones. Esto se ha hecho desde el apartado On Click() de cada botón.

4.4.2 Submenús

Todos los submenús (excepto el submenú Iniciar) tienen el mismo background que se ha comentado anteriormente y también tienen cosas en común todos ellos.

Todos los paneles a los que se acceda a través del menú principal tienen en la parte superior izquierda el Botón Atrás. Desde cualquiera de estos submenús, pulsando sobre este botón se regresará al menú principal. Además de esto todos tienen el logo de la Universidad Politécnica de Valencia en la esquina inferior derecha.

4.4.2.1 Iniciar

En el submenú Iniciar es en el que se puede acceder al contenido propio de la aplicación. En este panel están los rasgos comunes que hay en todos los submenús (el logo de la UPV y el botón atrás para volver al Menú Principal) y un menú desplegable en la esquina superior derecha. Al desplegar este menú se pueden ver los botones que permiten tomar una captura de pantalla y encender/apagar el flash del dispositivo móvil. Dicho menú se puede plegar y desplegar pulsando sobre el botón 'Menú Desplegable'.

Como se ha comentado anteriormente, en este submenú el background es completamente transparente ya que aquí es donde se ejecutará el contenido de la aplicación y se le ha dado un grado de transparencia total para no ensuciar la visión al usuario final.

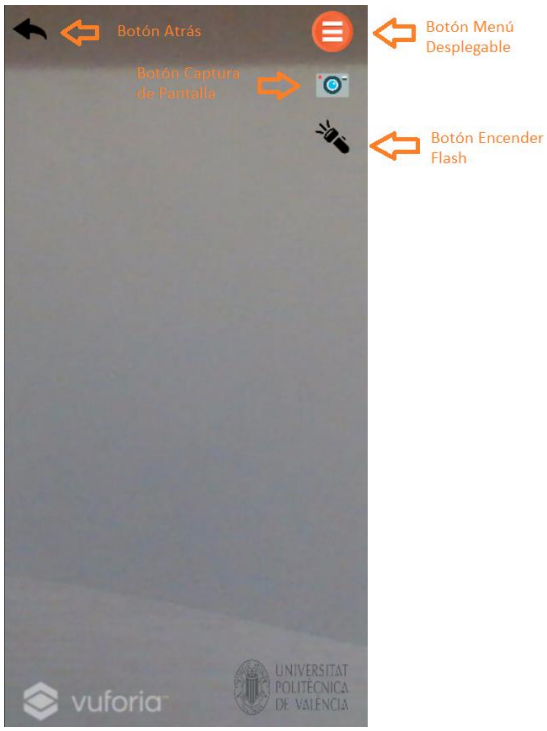


Figura 53. Submenú Iniciar. Fuente: Autor.

4.4.2.2 Valorar App

En este panel se muestra un *slider* y un botón desde el que se puede valorar la aplicación y esta valoración se guarda en la base de datos. Con ello se puede saber la opinión que tienen los usuarios sobre la aplicación

Figura 54. Submenú Valorar App. Fuente: Autor

Se ha configurado el slider para que los valores que devuelva sean números enteros del cero al cinco, y el incremento de estos números sea de izquierda a derecha como se puede ver en la siguiente figura. [Figura 55]

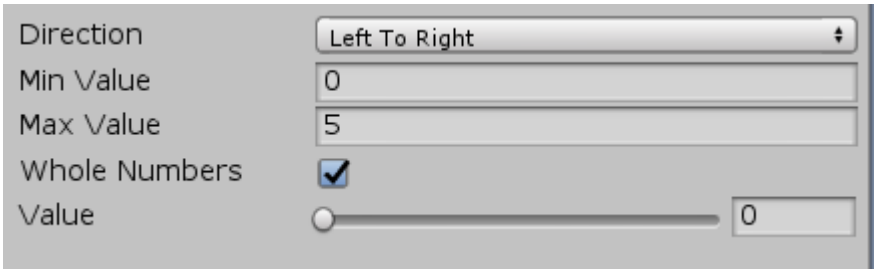


Figura 55. Configuración *slider.* Fuente: Autor.

4.4.2.3 Contacto

En el panel contacto se puede ver información referente al contacto del desarrollador de la aplicación: Email, número de teléfono, lugar de residencia y estudios que ha realizado.

Figura 56. Submenú Contacto. Fuente: Autor

4.4.2.4 Información App

En el submenú de 'Información App' se muestra información de la aplicación que estamos ejecutando.

Figura 57. Submenú Información App. Fuente: Autor

En este panel se muestra la versión de la aplicación. Para mostrar dicho dato en el script Log.cs se ha añadido el siguiente código.

Fragmento de código	`public class Log : MonoBehaviour {`
	`  public Text version;`
	`  // Use this for initialization`
	`  void Start () {`
	`  version.text = "Version: " + Application.version;`
	`  }`

Objetivo	Con este fragmento del script Log logramos retener la versión de la aplicación en una variable tipo Text.

Tabla 10. Fragmento script Log para almacenar la versión de la aplicación. Fuente: Elaboración propia.

Para mostrarla en el panel se ha añadido el script a *EventSystem*. Una vez añadido, en el Inspector aparece el script Log.cs (al seleccionar EventSystem*)* y se arrastra el elemento Text que se tiene preparado para mostrar dicha información, dentro del campo 'Version' que aparece en el apartado del script Log.

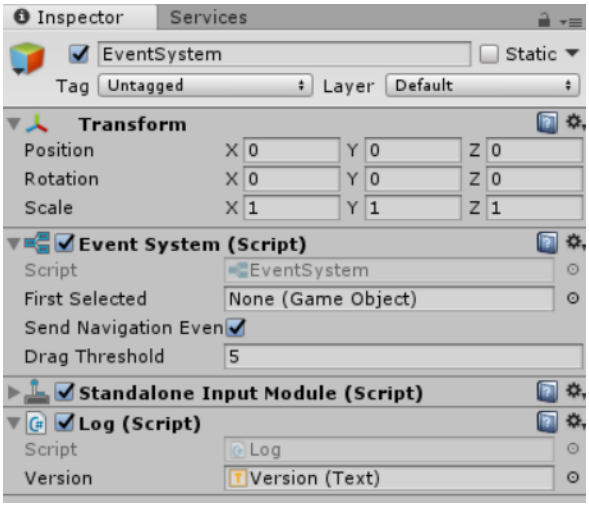


Figura 58. Inspector del elemento EventSystem. Fuente: Autor

4.5 ICONO DE LA APLICACIÓN

Para mejorar la experiencia de usuario, ayudar a reconocer la aplicación y dar personalidad a la misma, se ha añadido un icono para que al instalar la aplicación en un dispositivo aparezca representada por dicho icono. Se ha elegido un icono con las letra 'AR' haciendo referencia a 'Augmented Reality'.

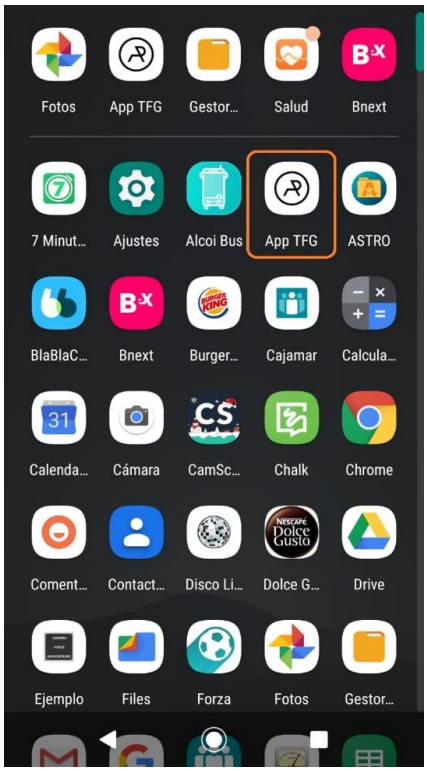


Figura 59. Menú de aplicaciones del smartphone. Fuente: Autor

Para ello se ha descargado una imagen .png y se ha añadido en el apartado *Icon* en Unity accesible a través de *Build Settings >> Player Settings >> Icon.*

Figura 60. Apartado Icon en Unity. Fuente: Autor

4.6 IMPLEMENTACIÓN DE LA CAPTURA DE PANTALLA

Para mejorar la experiencia de usuario dentro de la aplicación se ha configurado un botón para poder almacenar capturas de lo que se está viendo en pantalla en el momento. Con esta funcionalidad se pueden almacenar imágenes de las que el usuario tenga dudas, o quiera guardar para poder ver posteriormente.

Figura 61. Botón Capturar Pantalla. Fuente: Autor

Para accionar la captura de pantalla se ha puesto una imagen (de una cámara) en la pantalla en la que se ejecuta la aplicación a la que se ha añadido un componente botón para que tenga la funcionalidad de este. Se ha puesto 'Capturar Pantalla' de nombre a la imagen y en el Hierarchy está dentro de 'Iniciar'.

La funcionalidad de captura de pantalla se ha implementado a través del script TomarCapturaPantalla.cs. En este script se han creado dos funciones: TomarCaptura() y CapturarPantalla().

a) El método TomarCaptura() es un método público que llama a la corutina CapturaPantalla().

b) La corutina CapturarPantalla() contiene un *yield* en el que se espera a que acabe el *frame* para que se capture una imagen quieta y no en movimiento. Después de esto se guarda en una variable string la hora y fecha actuales que se pondrán como nombre del archivo a guardar. Con este nombre y mediante la función *CaptureScreeenshot()* se hace la captura y se guarda. La elección de la hora y

fecha actuales como nombre del archivo es para asegurar que no se repetirá el nombre del archivo en ningún momento.

```
public void TomarCaptura()
{
    StartCoroutine("CapturaPantalla");
}
private IEnumerator CapturaPantalla()
{
    yield return new WaitForEndOfFrame();
    string nombre = System.DateTime.Now.ToFileTime().ToString();
    ScreenCapture.CaptureScreenshot(nombre + ".png");
}
```

Figura 62. Fragmento de código del script TomarCapturaPantalla.cs. Fuente: Autor.

Para que esta acción se efectúe al pulsar el botón 'Capturar Pantalla' se arrastra el script dentro del objeto EventSystem y posteriormente se añade en el elemento On Click() de la imagen 'Capturar Pantalla' una nueva función. Se arrastra el objeto EventSystem y se elige la función 'TomarCaptura' del script TomarCapturaPantalla.cs.

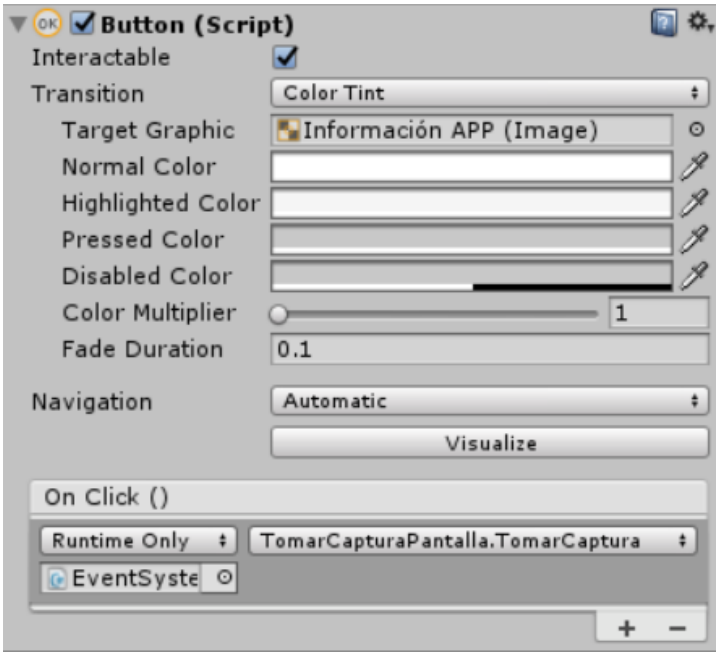


Figura 63. Elemento On Click() del botón Capturar Pantalla. Fuente: Autor

Una se vez se hacen las capturas, si la aplicación se está ejecutando en el ordenador a través del editor de Unity las imágenes se guardan en la siguiente ruta: *C:\Users\Diego\Documents\App TFG* donde 'App TFG' es el nombre de la aplicación.

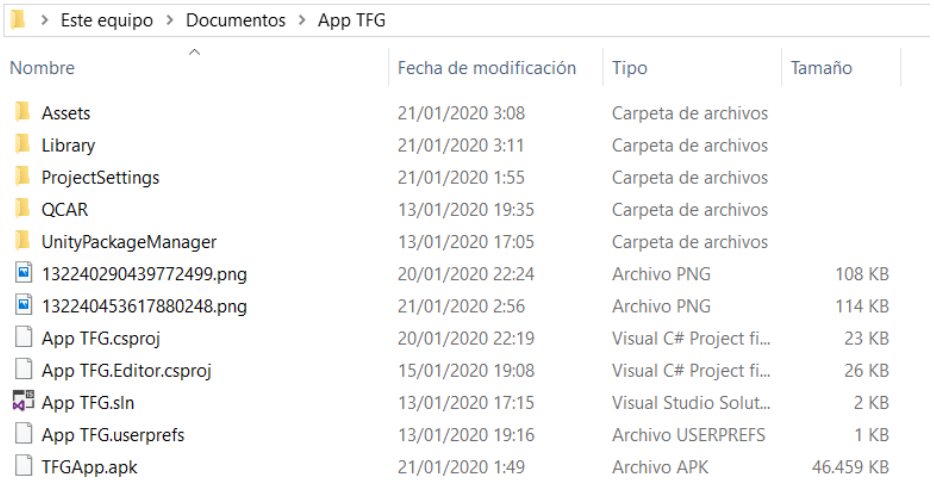


Figura 64. Ruta guardado capturas en ordenador. Fuente: Autor

En cambio, si la aplicación se está ejecutando en un dispositivo Android las imágenes se guardarán en la siguiente ruta: *Este equipo\Mi A2 Lite\Almacenamiento interno compartido\ Android\data\com.TFG.DiegoColoma\files*. La penúltima carpeta (com.TFG.DiegoColoma) tendrá el nombre que se ponga en el apartado *Identification* de *Build Settings >> Player Settings >> Other Settings.*

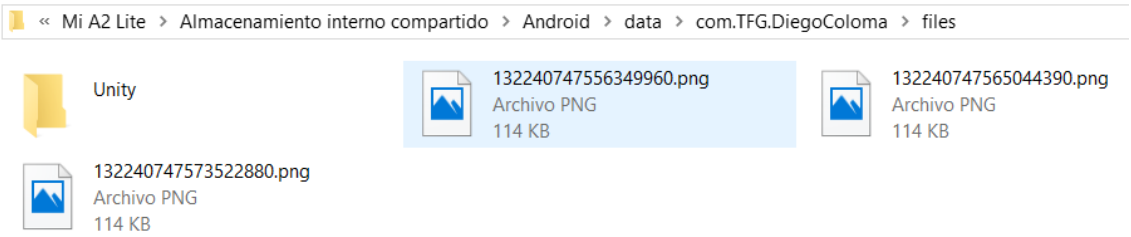


Figura 65. Ruta guardado capturas en dispositivo Android. Fuente: Autor

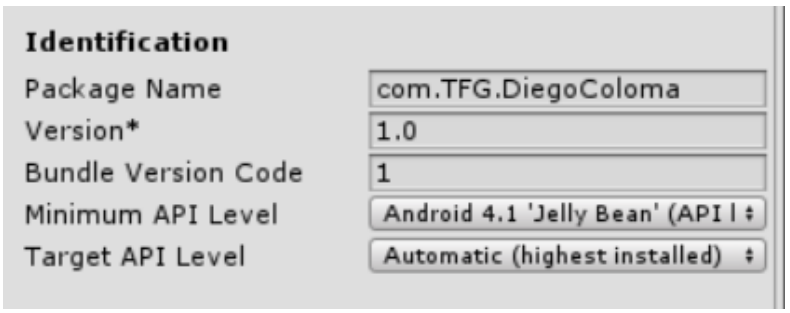


Figura 66. Package Name. Fuente: Autor

4.7 IMPLEMENTACIÓN DEL ENCENDIDO Y APAGADO DEL FLASH

Se ha querido añadir la función de encender y apagar el flash del dispositivo que esté ejecutando la aplicación ya que, dada la naturaleza de la aplicación, si en algún momento no se tiene buena luz en el lugar donde se esté utilizando el dispositivo y la aplicación, se puede encender el flash e iluminar mejor la zona deseada para que sea más fácil para la cámara reconocer los objetos y para el mismo usuario verlos.

Para ejecutar esta función, dentro de la pantalla donde se ejecuta la aplicación (submenú Iniciar) se ha puesto una imagen con componente botón de forma igual que en el apartado de Captura de Pantalla. Se ha puesto 'Encender Flash' de nombre a este botón.

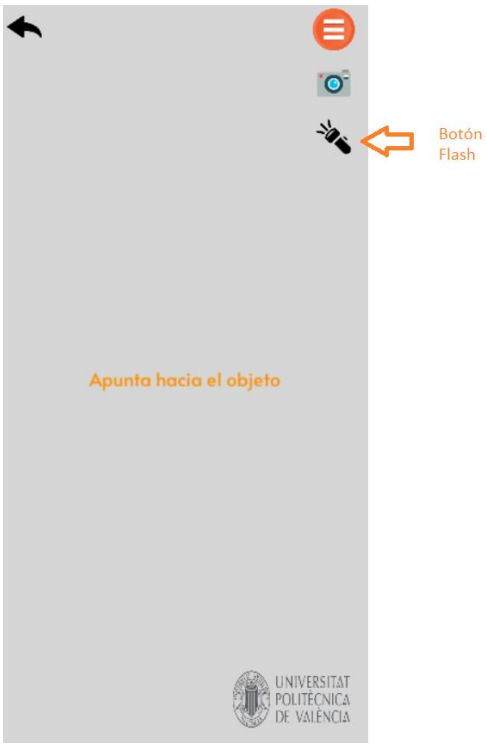


Figura 67. Botón Encender Flash. Fuente: Autor

Para implementar la funcionalidad del encendido y apagado del flash se ha utilizado el script escrito en C#, EncenderFlash.cs.

En este script se controla mediante un booleano (de nombre 'encendido') si el flash está o no encendido y para encender y apagar el flash se utiliza *CameraDevice.Instance.SetFlashMode()*. Si se pone esta instancia a *true* se enciende el flash y si se pone en *false* se apaga.

Dentro del script EncenderFlash.cs podemos encontrar los siguientes métodos que sirven para implementar su funcionalidad:

a) ***Start():*** Este método se utiliza para hacer acciones en la inicialización y en este caso se pone *CameraDevice.Instance.SetFlashMode()* a *false* y también se pone false el booleano que se utiliza para saber si está encendido o no el flash.

b) ***EncenderF():*** En este método se comprueba el estado del booleano 'encendido', si este está true se apaga el *flash* y se pone dicho booleano a false. Si en cambio ya está en false se enciende el flash y se pone el booleano a true.

Fragmento de código	Funcionamiento

`public void EncenderF()`
`{`
`  if (!encendido){`
`  CameraDevice.Instance.SetFlashTorchMode(true);`
`  encendido = true;`
`  // Debug.Log("encendido");`
`  }`
`  else`
`  {`
`  CameraDevice.Instance.SetFlashTorchMode(false);`
`  encendido = false;`
`  // Debug.Log("apagado");`
`  }`
`}`
	Fragmento del script EncenderFlash.cs en el que se utiliza la función *SetFlashTorchMode* cambiándola a *true* o a *false* para controlar el estado del flash del dispositivo.

Tabla 11. Fragmento script EncenderFlash.cs. Fuente: Elaboración propia.

Con este condicional se controla el estado del flash e implementamos su encendido y apagado mediante el botón que hay preparado.

Para relacionar el clic del botón con esta funcionalidad se arrastra el script EncenderFlash.cs dentro del objeto *EventSystem* y posteriormente dentro del apartado *On Click()* del botón 'Encender Flash' se arrastra el objeto *EventSystem* y se selecciona la función EncenderF() del script EncenderFlash.cs.

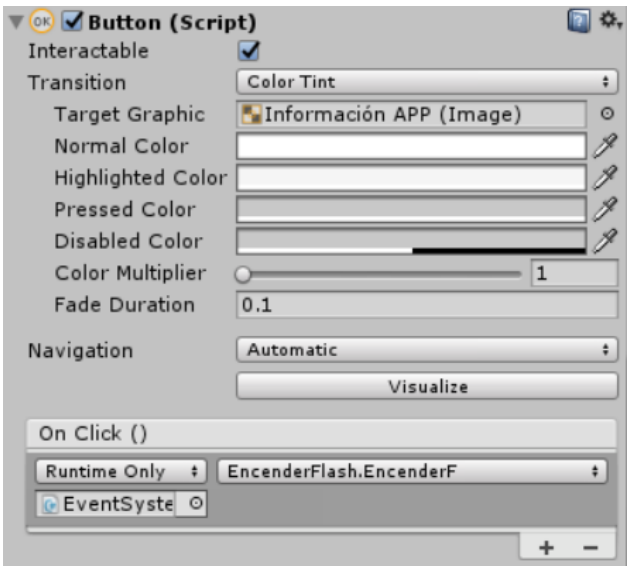


Figura 68. Elemento On Click() del botón Encender Flash. Fuente: Autor

4.8 IMPLEMENTACIÓN DEL MENÚ DESPLEGABLE

Dado que se han añadido varios botones con el fin de dotar de funcionalidades extra a la aplicación, dichos botones quedaban fijos en la pantalla mientras se ejecutaba la aplicación y esto podía empeorar la experiencia de uso de un usuario al disminuir la visibilidad de las instrucciones, que son la parte importante, al tener demasiados botones en pantalla. Para solventar este contratiempo, se ha decidido implementar un menú desplegable, para que los botones de 'Reset', 'Captura de Pantalla' y 'Encendido y Apagado de flash', se muestren y escondan al pulsar sobre dicho menú desplegable.

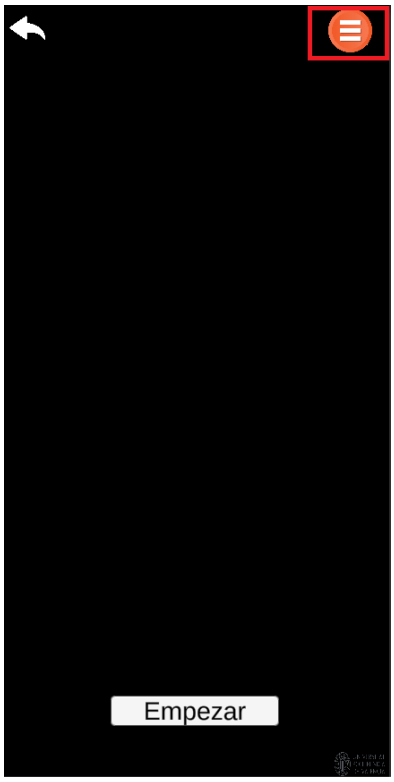


Figura 69. Menú desplegable escondido. Fuente: Autor.

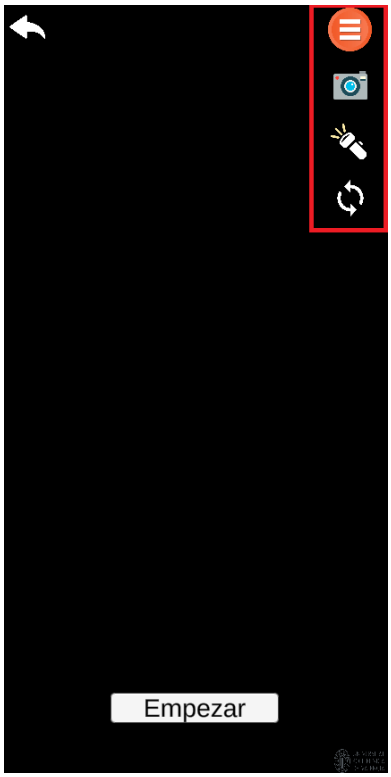


Figura 70. Menú desplegable desplegado. Fuente: Autor.

Con el objetivo de crear este menú desplegable se han creado dos animaciones para cada uno de los botones que se quieren añadir al menú. Para todos los botones se ha creado una animación en la cual el botón queda en la misma posición que el botón del menú desplegable, pero por detrás escondido. La otra animación que tiene cada botón los sitúa a cada uno de ellos en la posición que tendrán una vez se haya desplegado el menú. Estas posiciones se pueden observar en las figuras anteriores.

Para cambiar entre una animación y otra, en la pestaña *Animator* de Unity se han creado transiciones entre las dos animaciones y se ha puesto que la animación por defecto sea en la que el objeto está escondido.

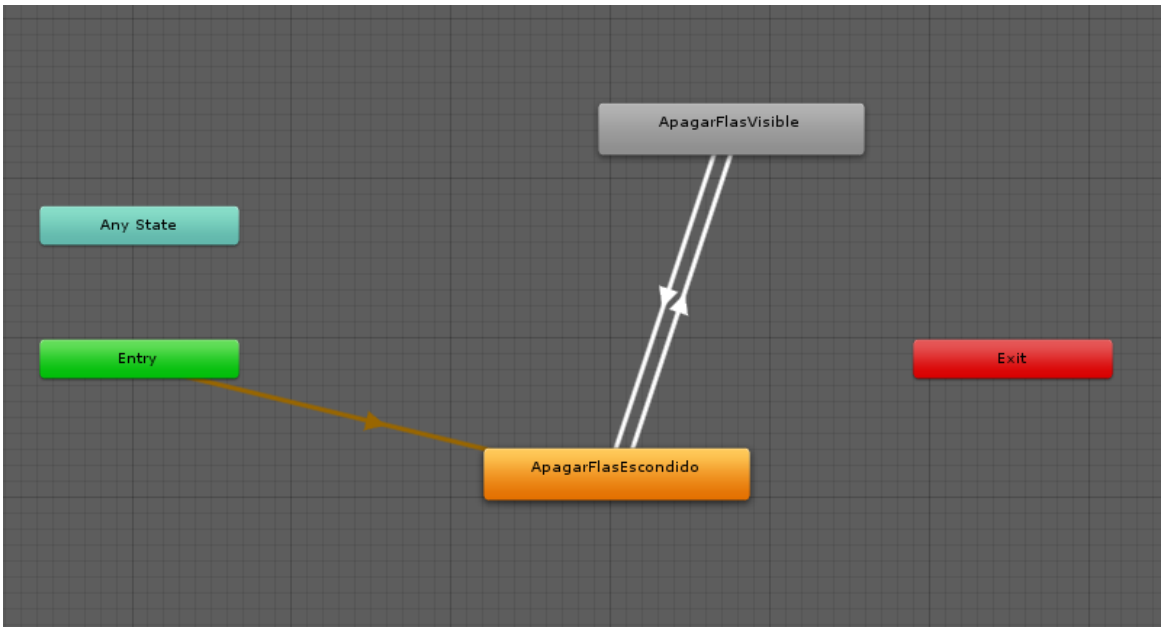


Figura 71. Animaciones del botón de encendido del flash. Fuente: Autor.

Se ha añadido el parámetro menú a todos los botones y a cada una de sus transiciones se ha añadido una condición que depende de este parámetro para poder controlar a través de este que animación debe de ejecutarse.

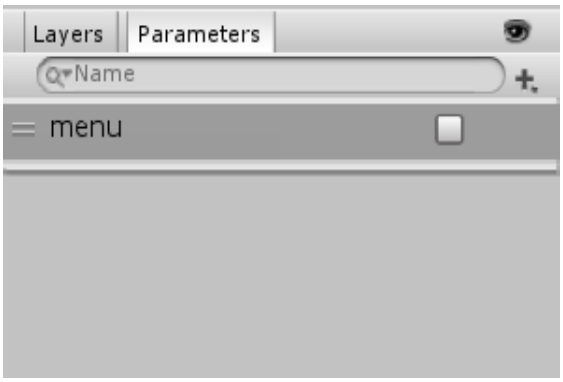


Figura 72. Parámetro "menu" de los botones del menú desplegable. Fuente: Autor.

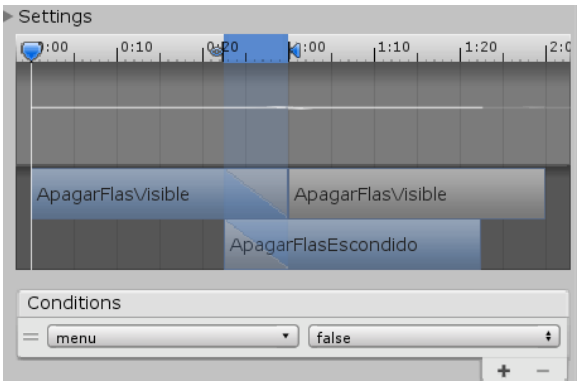


Figura 73. Condición de una de las transiciones del botón de apagar y encender el flash. Fuente: Autor.

Para controlar este menú desplegable se han creado los scripts MenuDesplegable.cs y MostrarMenu.cs.

Con el método *Mostrar()* que se ejecuta al pulsar el botón del menú desplegable se controla el valor de un booleano que es el que en el script MenuDesplegable.cs determina si como cambia el parámetro 'menu' para transicionar entre las animaciones en los que los botones están escondidos, y las animaciones en las que los botones están desplegados.

Fragmento del script	Objetivo

`private Animator animacion;`
`private MostrarMenu mostrarmenu;`
 	
`private void Start()`
`{`
`  animacion = GetComponent<Animator>();`
`  mostrarmenu = GameObject.Find("EventSystem").GetComponent<MostrarMenu>();`
`}`
 	
`private void Update()`
`{`
`  if(mostrarmenu.mostrar == false)`
`  {`
`  animacion.SetBool("menu", false);`
`  }`
`  if(mostrarmenu.mostrar == true)`
`  {`
`  animacion.SetBool("menu", true);`
`  }`
`}`
	Saber si se ha pulsado el botón del menú desplegable para desplegar este o contraerlo según lo que se desee.

`public bool mostrar;`
 	
`public void Mostrar()`
`{`
`  if (mostrar == true)`
`  {`
`  mostrar = false;`
`  }`
`  else if (mostrar == false)`
`  {`
`  mostrar = true;`
`  }`
`}`
	Controlar el valor del booleano “mostrar” al que se accederá desde el script MenuDesplegable.cs para saber si se desea que el menú este escondido o desplegado.

Tabla 12. Scripts MostrarMenu.cs y MenuDesplegable.cs con los que se controla el funcionamiento del menú desplegable. Fuente: Elaboración propia.

4.9 CONEXIÓN CON LA BASE DE DATOS

4.9.1 Qué es Xampp

Xampp es el nombre de una herramienta que permite realizar pruebas con un servidor local, sin necesidad de tener un servidor en la nube. Esta herramienta es multiplataforma y de software libre el desarrollo viene por parte de Apache Friends y tiene una licencia GNU.

El nombre Xampp es en realidad un acrónimo de X (para cualquiera de los distintos sistemas operativos), Apache, MySQL, PHP, Perl[20].

[20] *https://www.ecured.cu/XAMPP*

En el paquete básico de Xampp se encuentra:

a) Apache: Servidor web multiplataforma y de código abierto, que fue creado a mitad de los años 90. Soporta los lenguajes Python, PHP y Perl y es uno de los más utilizados del mercado.
b) MySQL: Desarrollado en 1994 desarrollado en su origen por MySQL AB, MySQL es un gestor de base de datos. Este gestor sigue un modelo cliente-servidor y es un gestor de datos relacionales.
c) PHP: Lenguaje de código abierto diseñado por Ramsus Lerdford en 1994. Suele ser utilizado para la programación en la parte del servidor.
d) Perl: Lenguaje multiplataforma y de uso libre diseñado por Larry Wall en 1987. Este lenguaje es interpretado al igual que por ejemplo Javascript y tiene una mezcla de características tomadas de los lenguajes C, AWK, Bourne Shell, Lisp, sed y otros. Su nombre significa Practical Extracting and Reporting Language (PERL).

4.9.2 Creación de la base de datos

Se ha creado una base de datos en Xampp. La finalidad de la creación de esta base de datos es poder controlar el acceso a la aplicación y poder guardar las valoraciones de los usuarios. Es una base de datos MySQL en la se almacenan los usuarios, sus contraseñas y sus valoraciones. Se puede acceder a la base de datos a través de un servidor Apache que el mismo Xampp lleva incorporado. Al estar alojada en local para acceder y administrar a la base de datos se accede desde un navegador web a este enlace http://localhost/phpmyadmin/.

Se ha creado una base de datos llamada 'tfg', y dentro de esta se han creado dos tablas en la base de datos, 'usuarios' y 'valoracion'.

Figura 74. Tablas de la base de datos. Fuente: Autor

Para guardar los datos en estas tablas se utilizan dos ficheros .php: Login.php y Valoración.php. Con estos ficheros se accede a los datos almacenados para comprobar si el usuario que intenta acceder a la aplicación existe, y también se guardan las valoraciones.

La tabla de usuarios contiene estos campos:

a) **Id**: Es la clave primaria de la tabla, y es entero autoincremental.
b) **User**: el nombre del usuario del tipo *varchar* y longitud máxima 15. Es un valor único para que no pueda haber dos nombres de usuario iguales
c) **Password**: la contraseña del usuario del tipo varchar y longitud máxima 15.

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
☐	1	**id**	int(11)			No	*Ninguna*		AUTO_INCREMENT	Cambiar Eliminar Más
☐	2	**user**	varchar(15)	utf8_spanish_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	3	**password**	varchar(15)	utf8_spanish_ci		No	*Ninguna*			Cambiar Eliminar Más

Figura 75. Campos de la tabla usuarios. Fuente: Autor

La tabla valoración tiene contiene estos campos:

a) **Usuario:** Nombre del usuario que envía la valoración del tipo varchar y de longitud máxima 15.
b) **Valoración:** un número entero que guarda la valoración del usuario.

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
☐	1	**usuario**	varchar(15)	utf8_spanish_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	2	**valoracion**	int(2)			No	*Ninguna*			Cambiar Eliminar Más

Figura 76. Campos de la tabla valoracion. Fuente: Autor

Se ha establecido una relación entre la columna usuario de la tabla valoración y la columna usuario de la tabla usuarios, ya que todos los usuarios que envíen una valoración serán usuarios almacenados en la base de datos.

Se puede en la siguiente imagen como el envío de la valoración y su almacenamiento en la base de datos se aplica correctamente.

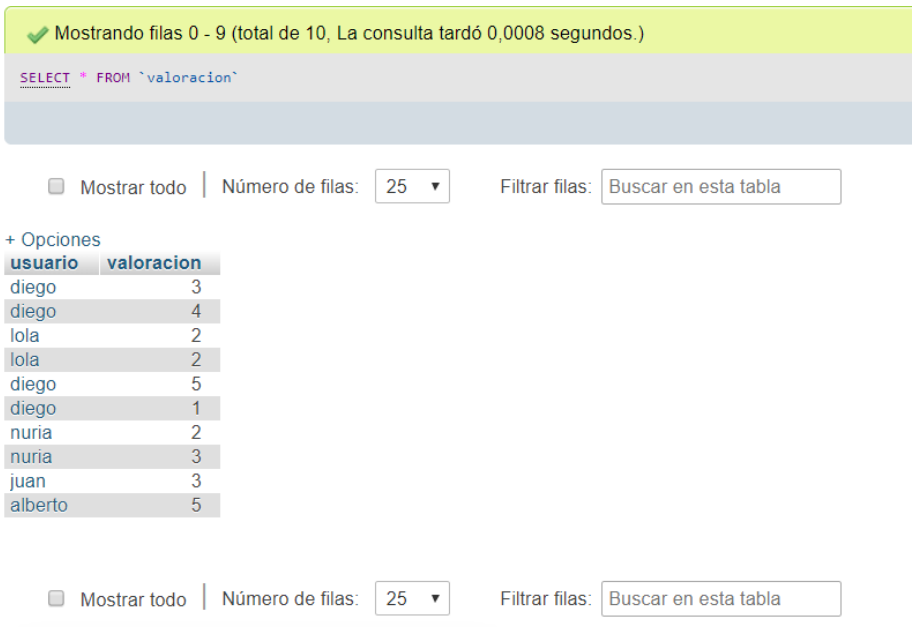


Figura 77. Valoraciones guardadas en la base de datos. Fuente: Autor

4.9.3 Comunicación entre Unity y la base de datos

Para verificar que el usuario que intenta acceder a la aplicación está guardado en la base de datos, se ha creado este menú de login.

Figura 78. Pantalla de Login. Fuente: Autor

A través del script Log.cs que recoge el nombre de usuario y la contraseña de los *Input Fields* que hay en la pantalla de login, se envían con el método Enviar() al script Login.php. Con el resultado de la consulta SQL que se ejecuta con el fichero .php si el resultado es que existe dicho usuario, y la contraseña es la correcta, se cambia de escena y se avanza al menú principal. Si los datos no son correctos aparece un mensaje de acceso denegado y si falta por algún campo por rellenar un mensaje para recordar que todos los campos deben contener datos.

```
$userlogin = $_POST["userName"];
$passlogin = $_POST["pass"];

$sql = "SELECT user FROM usuarios WHERE user = '$userlogin' AND password = '$passlogin'";


$result = $conn->query($sql);
```

Figura 79. Fragmento del script Login.php. Fuente: Autor.

Fragmento de código	Funcionamiento

`public void Entrar()`
`{`
`  if(inputUser.text == "" || inputPass.text == "" ){`
`  textResult.text = "Rellena todos los campos";`
`  return;`
`  }`
`  Login(inputUser.text, inputPass.text, delegate (Reply reply)`
`  {`
`  textResult.text = reply.response;`
`  if (reply.ok == true)`
`  {`
`  user = inputUser.text;`
`  SceneManager.LoadScene("TFG");`
`  }`
`  });`
`}`
`public void Login(string userName, string pass, Action<Reply> reply)`
`{`
`  StartCoroutine(checkUser(userName, pass, reply));`
`}`
	Fragmento de código del script Log.php. Se recogen los datos y se ejecuta la corutina chechUser. Se utiliza una corutina para esperar la contestación del script Login.php y poder permitir o denegar el acceso a la aplicación.

Tabla 13. Fragmento de código del script Log.php. Fuente: Elaboración propia.

Para recoger las valoraciones de los usuarios se ha creado un submenú llamado Valoración App al cual se accede desde el botón Valoración App del Menú Principal.

Figura 80. Pantalla de Valoración App. Fuente: Autor

A través del fichero Valoracion.cs se envía la valoración el usuario sobre la aplicación. La valoración se recoge a través de un slider y se envía con el botón Enviar de debajo. Desde este fichero con la función Valorar() se envían los datos al fichero valoracion.php que ejecuta la consulta SQL que añade la valoración del usuario a la base de datos.

```
$user = $_POST["user"];
$valor = $_POST["valor"];

$sql = "INSERT INTO valoracion VALUES  ('$user' , '$valor')" ;


$conn->query($sql) or die ("Error al enviar la valoración");
```

Figura 81.Fragmento del script Valoracion.php. Fuente: Autor.

Fragmento de código	Definición

`public void Valorar()`
`{`
`  user = log.Usuario();`
`  valor = (int)slider.value;`
`  StartCoroutine(EnviarValoracion(user, valor));`
`  gracias.text = "Gracias por tu valoración";`
`}`
	Fragmento del script Valoracion.php, en concreto método valorar en el que se recogen los datos en variables y se llama a la corutina EnviarValoracion().

Tabla 14. Fragmento de código del script Valoracion.php. Fuente: Elaboración propia.

4.10 INTEGRACIÓN DE UNITY ANALYTICS EN LA APLICACIÓN

Dada la naturaleza de la aplicación, es muy importante el buen funcionamiento de la misma. Para ayudar a controlar esto, se ha decidido utilizar Unity Analytics. Esta herramienta permite recopilar datos de uso de la aplicación, que se pueden analizar posteriormente para saber cual es el volumen de utilización de la aplicación, mejorar algunos errores y realizar mejoras.

Esta herramienta de análisis devuelve datos por defecto como por ejemplo: DAU, usuarios activos por día; MAU, usuarios activos por mes; número total de sesiones; tiempo activo en la aplicación.

Para integrarla en nuestra aplicación, en el apartado *Services* de Unity, se ha habilitado el servicio *Analytics.* Una vez activado esto ya se pueden ver los datos que la herramienta devuelve por defecto en la web.

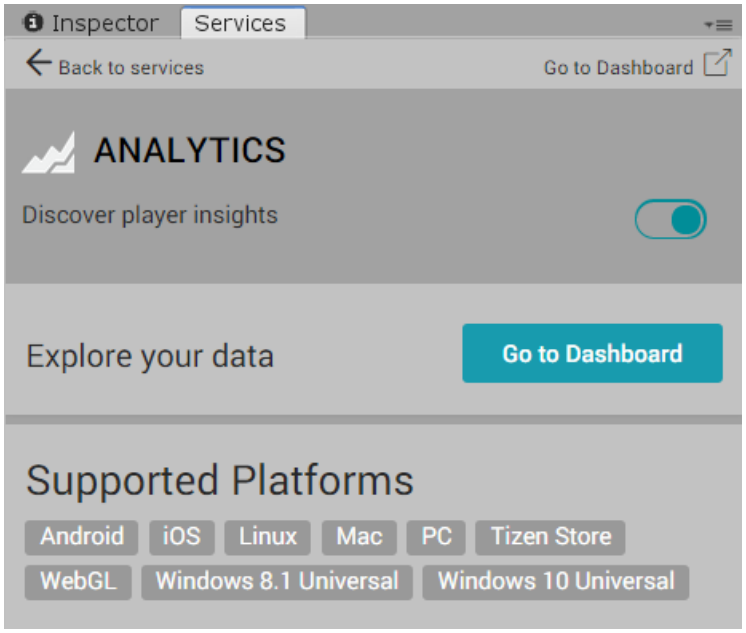


Figura 82. Integración de Unity Analytics. Fuente: Autor.

Para poder mejorar la aplicación, una de las cosas que parece interesante es saber qué pasos de los casos de asistencia son más difíciles de comprender por el usuario y hacen que este los tenga que repetir para asegurarse de su que su ejecución haya sido correcta. Para saber esto, se ha implementado un *Custom Event.* Los *Custom Events* son eventos que Unity Analytics permite crear para poder recoger datos de una forma personalizada.

Para saber cuándo un usuario repite un paso, se ha decidió implementar un método, que se enlazará con el botón atrás, y cada vez que se pulse, lanzará un evento que envíe el nombre del target que se está enfocando en ese momento, además del número del paso al que se regresa al pulsar el botón. Este método se puede encontrar en los scripts Pasos.cs y PasosConCodigo.cs. El método está implementado en ambos scripts para que envíe los datos de cualquiera de los casos de asistencia.

```
public void Analisis()
    {
        if (nombre == target.TrackableName)
        {
            Analytics.CustomEvent("Analisis",
            new Dictionary<string, object>
            {
                {"Target", nombre},
                {"Paso", contador }
            }
                );
        }
    }
```

Figura 83. Método Analisis() en el script Pasos.cs Fuente: Autor.

En la siguiente figura se muestran algunos de lo resultados que se han obtenido con el evento que se ha creado. Esos resultados se pueden ver a través del *Dashboard* de la web de Unity.

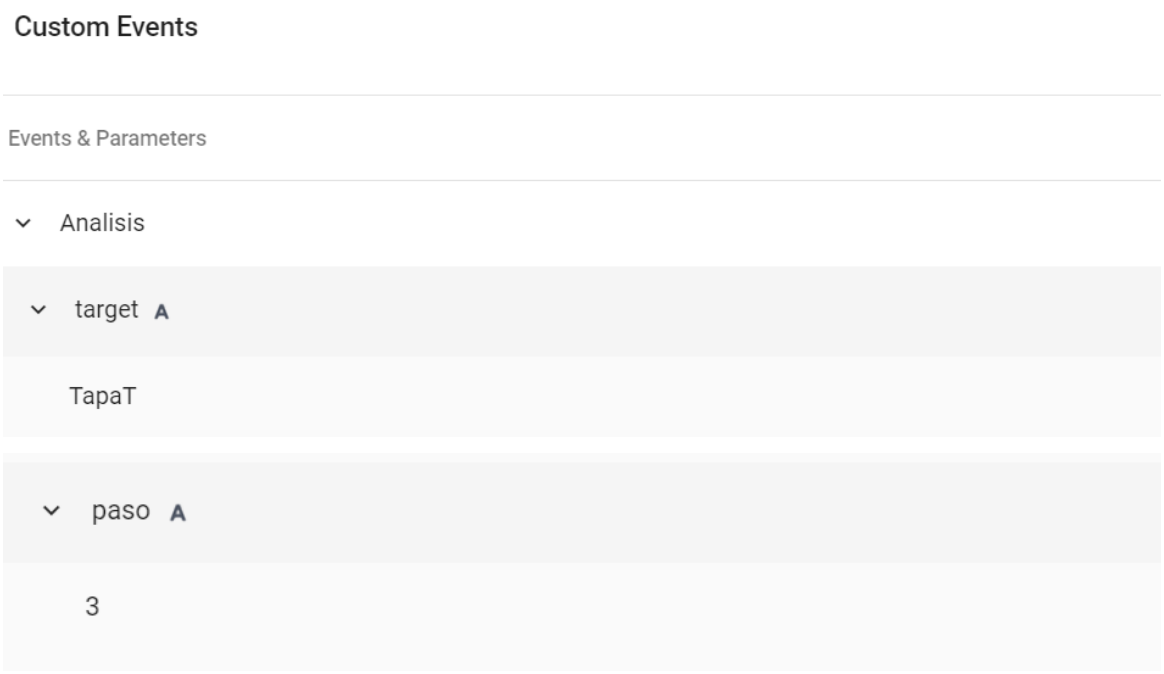


Figura 84. Datos de Unity Analytics en el Dashboard. Fuente: Autor.

5. CONCLUSIONES

La Realidad Aumentada es una tecnología en auge, es el futuro, pero también el presente tanto de la industria como de la vida cotidiana. Esta tecnología permite mostrar a los ojos de los usuarios información de una forma totalmente nueva, hace que el usuario pueda ver de manera natural la información delante de sus ojos, como si de objetos reales se tratara. Esto abre el camino a infinidad de nuevas posibilidades que es el momento de estudiar y explotar.

Utilizando esta tecnología se ha desarrollado una aplicación de asistencia para la industria con el que ofrecen una asistencia desatendida a los usuarios que la utilicen y ayudarlos a realizar las tareas programadas mostrándoles los pasos que tienen que ejecutar y ofreciéndoles información sobre dicha tarea.

Se han completado todos los objetivos planteados y se han añadido algunas características a la aplicación para hacerla más completa y funcional. Los objetivos principales han sido implementar un sistema para poder crear casos de asistencia sin necesidad de trabajar con código e implementar un caso de asistencia más complejo, esta vez sí, con la utilización de código. El primer objetivo se ha desarrollado para que no sea necesario un experto en la plataforma o un programador para crear casos de asistencia, que un técnico, con unos conocimientos básicos y de forma muy sencilla pueda hacerlo. El segundo, es un ejemplo de que algunos casos de asistencia son más complejos, y sí necesitan que un experto los cree. Por todo esto, la aplicación cumple con las pretensiones que se tenían sobre ella, es funcional y supera todas las pruebas a las que se le ha sometido.

Por último, también cabe destacar que esta aplicación es muy adaptable a distintos ámbitos. En este caso se ha centrado para dar asistencia a los usuarios, pero también es adaptable a la formación, utilizando así los usuarios la aplicación como una ayuda en el aprendizaje. También se podría explotar en un entorno comercial, dando la opción al vender un producto que necesite un montaje, de descargar esta aplicación y tener un manual de instrucciones interactivo y mucho más claro para los compradores que el clásico manual en papel.

6. POSIBLES MEJORAS FUTURAS

Una vez acabado con todo el desarrollo y finalizado el proyecto se proponen implementaciones que en un futuro se pueden añadir a la aplicación que permitirán tener una aplicación más completa y funcional.

- Cifrado en el login.

  Ya que el *login* de la aplicación está vinculado a una base de datos local, no se ha visto necesario realizar ningún cifrado, pero con el cambio de la base de datos a una base de datos externa sería muy interesante cifrar los datos para proteger los datos de los usuarios ya que en las comunicaciones online es importante tener los datos cifrados ya que serán mucho menos vulnerables y hay mucho menos riesgos de ser interceptados.

- Exportación de la aplicación para gafas de Realidad Aumentada.

  Dado el uso que se va a dar a esta aplicación, tiene mucho sentido que esta aplicación se utilizada con *smartglasses.* La utilización de la aplicación con gafas permitiría al usuario tener ambas manos libres y seguir trabajando mientras no pierde de vista las indicaciones que se le van dando a través del programa. Además de esto, la integración de la aplicación con los trabajos que se van a realizar sería mucho más sencilla para los usuarios ya que ya que simplemente tendrían que llevar puestas unas ‘gafas’, que ya muchos utilizaran por problemas de visión o para protegerse de los rayos solares. Para los usuarios será mucho más sencillo acostumbrarse a trabajar con unas gafas puestas, que teniendo que sujetar un dispositivo con la mano.

7. BIBLIOGRAFÍA Y REFERENCIAS

7.1 PUBLICACIONES

1. Schwab, Klaus (2016). “La cuarta Revolución Industrial”. Barcelona, España. Editorial: Debate.
   Recuperado de:
   http://40.70.207.114/documentosV2/La%20cuarta%20revolucion%20industrial-Klaus%20Schwab%20(1).pdf
2. Pimentel, K., & Teixeira, K (1993). “Virtual reality”. New York, NY: Editorial: McGraw-Hill.
3. Caudell & Mizel, Thomas & David (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. Recuperado de:
   https://www.researchgate.net/publication/3510119_Augmented_reality_An_application_of_heads-up_display_technology_to_manual_manufacturing_processes
4. Mozas Fenoll, Edgar (2016). “TIC 4.1 Qué es la realidad aumentada”.
   Recuperado de:
   https://www.youtube.com/watch?reload=9&v=jpd5E_Dh9PU
5. Lens-Fitzgerald, M (2009). “Augmented Reality Hype Cycle”.
   Recuperado de:
   http://acdc.sav.us.es/pixelbit/images/stories/p46/12.pdf
6. J. Hernández, M. Pennesi, D. Sobrino & A. Vázquez (2012). “Tendencias emergentes en educación con TIC”. Barcelona, España. Editorial: Espiral. Recuperado de:
   https://dialnet.unirioja.es/servlet/articulo?codigo=4230624
7. X. Basogain, M. Olabe, K. Espinosa, C. Rouèche* y J.C. Olabe (2007). “Realidad Aumentada en la Educación: una tecnología emergente”. Bilbao, España. Recuperado de:
   http://files.trendsandissues.webnode.com/200000010-3884839004/educamadrid-2007.pdf

7.2 PÁGINAS WEB

1. Página web Unity. *www.unity.com.* [Consulta: 08/01/2020]
2. Página web Vuforia. *developer.vuforia.com.* [Consulta: 08/01/2020]
3. Página web Aumentaty. *www.aumentaty.com.* [Consulta: 15/12/2019]
4. Página web ARDev. *www.ardev.com.* [Consulta: 15/12/2019]
5. Página web Xataka. *www.xataka.com.* [Consulta: 18/12/2019]
6. Página web Wikipedia *www.wikipedia.com.* [Consulta: 29/12/2019]
7. Página web Economipedia. *www.economipedia.com.* [Consulta: 14/12/2019]
8. Página web PTC. *www.ptc.com.* [Consulta: 08/01/2020]
9. Página web Epson. *www.epson.com*. [Consulta: 15/12/2019]
10. Repositorio de la UPV, Riunet. *riunet.upv.es*. [Consulta: 15/12/2019]
11. YouTube. *www.youtube.com*. [Consulta: 16/01/2020]

8. ANEXOS

ANEXO 1. SCRIPT LOGIN.PHP

```
<?php
    // Database Things
==========================================================
      //These values can be found in the email.
    $host = "localhost"; //<--put your host here
    $user = "diecobra"; //<-- put username here
    $password = "1234"; //<--put your password here
    $dbname = "tfg"; //<-- put your database name here
    $conn = new mysqli($host, $user, $password, $dbname) or
die("Error en la conexión a la base de datos");



      $loginUsuario = $_POST["nombreUsuario"];
    $loginContrasenya = $_POST["contrasenya"];

      $sql = "SELECT user FROM usuarios WHERE user = '$loginUsuario'
AND password = '$loginContrasenya'";


       $result = $conn->query($sql);

      if($result->num_rows > 0)
      {
            $m = array('ok' => true , 'response' => "Acceso
permitido");
            Header('Content-Type: application/json');
            echo json_encode($m);

            exit();
      }
      else
      {
            $m = array('ok'=> false , 'response' => "Acceso
denegado");
            Header('Content-Type: application/json');
            echo json_encode($m);

            exit();
      }
      $conn->close();

       ?>
```

ANEXO 2. VALORACIÓN.PHP

```
<?php
    // Database Things
=============================================================
      //These values can be found in the email.
    $host = "localhost"; //<--put your host here
    $user = "diecobra"; //<-- put username here
    $password = "1234"; //<--put your password here
    $dbname = "tfg"; //<-- put your database name here
    $conn = new mysqli($host, $user, $password, $dbname) or
die("Error en la conexión a la base de datos");



      $user = $_POST["user"];
    $valor = $_POST["valor"];

      $sql = "INSERT INTO valoracion VALUES  ('$user' , '$valor')" ;


       $conn->query($sql) or die ("Error al enviar la valoración");


      $conn->close();
       ?>
```

ANEXO 3. EXIT.CS

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

public class Exit : MonoBehaviour {

       //Se utiliza para salir de la aplicación
       public void ExitApp()
    {
        Application.Quit();
    }
}
```

ANEXO 4. VALORACION.CS

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


public class Valoracion : MonoBehaviour
{
    //Variables
    [SerializeField] private Slider slider;
    private string user;
    private int valor;
    private Log log = new Log();
    [SerializeField] private Text gracias;

    //Método publico que llama a EnviarValoracion
    public void Valorar()
    {

        user = log.Usuario();

        valor = (int)slider.value;

        StartCoroutine(EnviarValoracion(user, valor));

        gracias.text = "Gracias por tu valoración";

    }

    //Envía los datos a la base de datos
    private IEnumerator EnviarValoracion(string user, int valor)
    {

        WWWForm form = new WWWForm();
        form.AddField("user", user);
        form.AddField("valor", valor);
        Debug.Log(user);
        Debug.Log(valor);

        WWW w = new WWW("http://localhost/TFG/Valoracion.php", form);
        yield return w;

    }
   }
```

ANEXO 5. LOG.CS

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



public class Log : MonoBehaviour
{
    //Variables
    [SerializeField] private InputField inputUser;
    [SerializeField] private InputField inputPass;
    [SerializeField] private Text textResult;
    public Text version;
    private static string user;


    public class Reply
    {
        public bool ok;
        public string response;
    }

    //Devuelve el nombre del usuario
    public string Usuario()
    {
        return user;
    }


    // Use this for initialization
    void Start()
    {
        version.text = "Version: " + Application.version;
    }


    //Método público que llama a Login y si el nombre y contraseña son
corectos, cambia de escena
    public void Entrar()
    {
        if (inputUser.text == "" || inputPass.text == "")
        {
            textResult.text = "Rellena todos los campos";
            return;

        }
        Login(inputUser.text, inputPass.text, delegate (Reply reply)

        {
            textResult.text = reply.response;
            if (reply.ok == true)
            {
                user = inputUser.text;
                SceneManager.LoadScene("TFG");
            }

        });
```

```
    }

    //Llama la corutina chechUser
    public void Login(string nombreUsuario, string contrasenya, Action<Reply>
reply)
    {
        StartCoroutine(checkUser(nombreUsuario, contrasenya, reply));

    }

    //Envía los datos a la base de datos, y recoge el resultado de esta
    private IEnumerator checkUser(string nombreUsuario, string contrasenya,
Action<Reply> reply)
    {
        WWWForm form = new WWWForm();
        form.AddField("nombreUsuario", nombreUsuario);
        form.AddField("contrasenya", contrasenya);

        WWW w = new WWW("http://localhost/TFG/Login.php", form);

        yield return w;
        reply(JsonUtility.FromJson<Reply>(w.text));
    }

    //Se utiliza para salir del Menú principal al Login, cambiando de escena
    public void Salir()
    {
        SceneManager.LoadScene("Login");
    }
   }
```

ANEXO 6. MOSTRARMENU.CS

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

public class MostrarMenu : MonoBehaviour
{
    //Variable
    public bool mostrar;

    //Se utiliza para mostrar u ocultar el menú desplegable
    public void Mostrar()
    {
        if (mostrar == true)
        {
            mostrar = false;
        }
        else if (mostrar == false)
        {
            mostrar = true;
        }
    }

   }
```

ANEXO 7. MENUDESPLEGABLE.CS

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

public class MenuDesplegable : MonoBehaviour
{
    //Variables
    private Animator animacion;
    private MostrarMenu mostrarmenu;

    // Use this for initialization
    private void Start()
    {
        animacion = GetComponent<Animator>();
        mostrarmenu =
GameObject.Find("EventSystem").GetComponent<MostrarMenu>();
    }

    // Update is called once per frame
    private void Update()
    {
        if (mostrarmenu.mostrar == false)
        {
            animacion.SetBool("menu", false);
        }
        if (mostrarmenu.mostrar == true)
        {
            animacion.SetBool("menu", true);
        }
    }

   }
```

ANEXO 8. TOMARCAPTURAPANTALLA.CS

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

public class TomarCapturaPantalla : MonoBehaviour
{
    //Llama a la corutina CapturaPantalla
    public void TomarCaptura()
    {
        StartCoroutine("CapturaPantalla");
    }

    //Se utiliza para tomar una captura de pantalla y guardarla en local
    private IEnumerator CapturaPantalla()
    {
        yield return new WaitForEndOfFrame();
        string nombre = System.DateTime.Now.ToFileTime().ToString();
        ScreenCapture.CaptureScreenshot(nombre + ".png");
    }

   }
```

ANEXO 9. ENCNDERFLASH.CS

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

public class EncenderFlash : MonoBehaviour
{

    //Variable
    private bool encendido;


    // Use this for initialization
    void Start()
    {
        encendido = false;
        CameraDevice.Instance.SetFlashTorchMode(false);

    }

    //Se utiliza para apagar y encender el flash del dispositivo
    public void EncenderF()
    {
        if (!encendido)
        {

            CameraDevice.Instance.SetFlashTorchMode(true);
            encendido = true;
        }
        else
        {
            CameraDevice.Instance.SetFlashTorchMode(false);
            encendido = false;
        }

    }


   }
```

ANEXO 10. PASOS.CS

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

public class Pasos : MonoBehaviour
{
    //Variables
    public TrackableBehaviour target;
    public GameObject flechageneral;
    public TextMesh variableTexto;
    private int contador = -1;
    private string nombre = "";
    private Button b_anterior;
    private Button b_siguiente;
    private Button b_empezar;
    private GameObject iniciar;
    public Text aviso;
    public List<Instruccion> Paso = new List<Instruccion>();


    //Clase que guarda la posicion y rotación de las flechas que se añaden
    [Serializable]
    public class Flechas
    {
        public Vector3 posicion = new Vector3();
        public Vector3 rotacion = new Vector3();
        [NonSerialized] public GameObject flecha;


    }

    //Clase que almacena el texto y la lista de flechas que se ha añadido para
cada Paso
    [Serializable]
    public class Instruccion
    {
        public string texto;
        public List<Flechas> Flechas = new List<Flechas>();

    }


    // Use this for initialization
    private void Start()
    {

        b_empezar =
GameObject.FindGameObjectWithTag("StartButton").GetComponent<Button>();
        b_siguiente =
GameObject.FindGameObjectWithTag("rightButton").GetComponent<Button>();
        b_anterior =
GameObject.FindGameObjectWithTag("leftButton").GetComponent<Button>();
        iniciar = GameObject.FindGameObjectWithTag("CanvasIniciar");
```

```
        for (int i = 0; i < Paso.Count; i++)
        {
            for (int j = 0; j < Paso[i].Flechas.Count; j++)
            {
                Paso[i].Flechas[j].flecha = Instantiate(flechageneral);
                Paso[i].Flechas[j].flecha.transform.position =
(flechageneral.transform.position + Paso[i].Flechas[j].posicion);

Paso[i].Flechas[j].flecha.transform.Rotate(Paso[i].Flechas[j].rotacion);
                Paso[i].Flechas[j].flecha.SetActive(false);
            }



        }

        flechageneral.SetActive(false);
        b_siguiente.gameObject.SetActive(false);
        b_anterior.gameObject.SetActive(false);
        b_empezar.interactable = false;
        iniciar.SetActive(false);

    }


    //Se utiliza para resetear el valor de las variables, y volver al principio
del caso de uso
    public void Resetear()
    {
        if (contador != -1)
        {
            for (int j = 0; j < (Paso[contador].Flechas.Count); j++)
            {
                Paso[contador].Flechas[j].flecha.SetActive(false);

            }
        }
        variableTexto.text = "";
        contador = -1;
        b_anterior.gameObject.SetActive(false);
        b_siguiente.gameObject.SetActive(false);
        b_empezar.gameObject.SetActive(true);
        b_empezar.interactable = false;
        aviso.text = "Vuelve a envfocar el objetivo";

    }


    //Corutina que se utiliza para controlar el cambio de paso
    public IEnumerator CambiarPaso()
    {



        if (nombre == target.TrackableName)
        {

            if (Paso.Count > 1)
            {

                if (contador == 0)
                {
```

```
for (int j = 0; j < (Paso[contador + 1].Flechas.Count); j++)
                    {
                        Paso[contador + 1].Flechas[j].flecha.SetActive(false);

                    }

                    for (int j = 0; j < Paso[contador].Flechas.Count; j++)
                    {
                        Paso[contador].Flechas[j].flecha.SetActive(true);
                        variableTexto.text = Paso[contador].texto;

                    }

                    b_anterior.interactable = false;
                    b_siguiente.interactable = true;
                    yield return null;
                }
                else if (contador > 0 && contador < (Paso.Count - 1))
                {
                    for (int j = 0; j < (Paso[contador - 1].Flechas.Count);
j++)
                    {
                        Paso[contador - 1].Flechas[j].flecha.SetActive(false);

                    }
                    for (int j = 0; j < (Paso[contador + 1].Flechas.Count);
j++)
                    {
                        Paso[contador + 1].Flechas[j].flecha.SetActive(false);

                    }

                    for (int j = 0; j < Paso[contador].Flechas.Count; j++)
                    {
                        Paso[contador].Flechas[j].flecha.SetActive(true);
                        variableTexto.text = Paso[contador].texto;

                    }

                    b_anterior.interactable = true;
                    b_siguiente.interactable = true;
                    yield return null;
                }
                else if (contador == (Paso.Count - 1))
                {

                    for (int j = 0; j < (Paso[contador - 1].Flechas.Count);
j++)
                    {
                        Paso[contador - 1].Flechas[j].flecha.SetActive(false);

                    }


                    for (int j = 0; j < Paso[contador].Flechas.Count; j++)
                    {
                        Paso[contador].Flechas[j].flecha.SetActive(true);
                        variableTexto.text = Paso[contador].texto;

                       }
```

```
b_anterior.interactable = true;
                    b_siguiente.interactable = false;
                    yield return null;

                }

            }
            else if (Paso.Count == 1)
            {
                for (int j = 0; j < Paso[contador].Flechas.Count; j++)
                {
                    Paso[contador].Flechas[j].flecha.SetActive(true);
                    variableTexto.text = Paso[contador].texto;

                }



                b_anterior.interactable = false;
                b_siguiente.interactable = false;
                yield return null;

            }
            else
            {
                b_anterior.interactable = false;
                b_siguiente.interactable = false;
                variableTexto.text = ("No hay ningún paso creado");
            }
        }


    }


    //Se utiliza para avanzar al siguiente paso, llamando a la corutina
CambiarPaso
    public void BotonSiguiente()
    {

        if (nombre == target.TrackableName)
        {

            if (contador == -1)
            {
                b_siguiente.gameObject.SetActive(true);
                b_anterior.gameObject.SetActive(true);
                b_empezar.gameObject.SetActive(false);

                contador++;

                StartCoroutine("CambiarPaso");

            }
            else if (nombre == target.TrackableName && contador < Paso.Count -
1)
            {
                contador++;
                StartCoroutine("CambiarPaso");
            }
        }

       }
```

```
    //Se utiliza para retroceden al paso anterior, llamando a la corutina
CambiarPaso
    public void BotonAnterior()
    {
        if (nombre == target.TrackableName)
        {
            if (contador > 0)
            {
                contador--;
                StartCoroutine("CambiarPaso");

            }
        }
    }

    //Update is called once per frame. Se utiliza para saber el nombre del
objeto trackeado
    private void Update()
    {
        StateManager sm = TrackerManager.Instance.GetStateManager();
        IEnumerable<TrackableBehaviour> tbs =
sm.GetActiveTrackableBehaviours();

        foreach (TrackableBehaviour tb in tbs)
        {

            if (tb.CurrentStatus == TrackableBehaviour.Status.TRACKED)
            {
                nombre = tb.TrackableName;

            }

        }
    }

    //Se utiliza para enviar datos a Unity Analytics
    public void Analisis()
    {
        if (nombre == target.TrackableName)
        {
            Analytics.CustomEvent("Analisis",
            new Dictionary<string, object>
            {
                {"Target", nombre},
                {"Paso", contador }
            }
                );
        }
    }



   }
```

ANEXO 11. PASOSCONCODIGO.CS

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

public class PasosConCodigo : MonoBehaviour
{
    //Variables
    private int contadortapa = -1;
    private int contadortablet = 0;
    public GameObject p_uno;
    public GameObject p_dos;
    public GameObject p_tres;
    public GameObject p_unotablet;
    public GameObject p_dostablet;
    public GameObject p_trestablet;
    public GameObject p_cuatrotablet;
    public Button b_anterior;
    public Button b_siguiente;
    public Button b_empezar;
    public Text aviso;
    private bool empezado = false;
    private string nombre = "";

    //Se utiliza para resetear el valor de las variables, y volver al principio
del caso de uso
    public void Resetear()
    {

        contadortapa = -1;
        contadortablet = 0;
        b_anterior.gameObject.SetActive(false);
        b_siguiente.gameObject.SetActive(false);
        b_empezar.gameObject.SetActive(true);
        b_empezar.interactable = false;
        p_uno.SetActive(false);
        p_dos.SetActive(false);
        p_tres.SetActive(false);
        p_unotablet.SetActive(false);
        p_dostablet.SetActive(false);
        p_trestablet.SetActive(false);
        p_cuatrotablet.SetActive(false);
        aviso.text = "Vuelve a enfocar el objetivo";
        empezado = false;

    }

    // Use this for initialization
    void Start()
    {
        p_uno.SetActive(false);
        p_dos.SetActive(false);
        p_tres.SetActive(false);
        p_unotablet.SetActive(false);
        p_dostablet.SetActive(false);
        p_trestablet.SetActive(false);
        p_cuatrotablet.SetActive(false);

       }
```

```
// Update is called once per frame
    void Update()
    {

        StateManager sm = TrackerManager.Instance.GetStateManager();
        IEnumerable<TrackableBehaviour> tbs =
sm.GetActiveTrackableBehaviours();

        foreach (TrackableBehaviour tb in tbs)
        {

            if (tb.CurrentStatus == TrackableBehaviour.Status.TRACKED)
            {
                nombre = tb.TrackableName;

            }
        }


        if (nombre == "TapaT")
        {
            if (contadortapa == 0)
            {
                p_uno.SetActive(true);
                p_dos.SetActive(false);
                p_tres.SetActive(false);
                b_anterior.interactable = false;
                b_siguiente.interactable = true;

            }
            else if (contadortapa == 1)
            {
                p_uno.SetActive(false);
                p_dos.SetActive(true);
                p_tres.SetActive(false);
                b_anterior.interactable = true;
                b_siguiente.interactable = true;

            }
            else if (contadortapa == 2)
            {
                p_dos.SetActive(false);
                p_tres.SetActive(true);
                b_anterior.interactable = true;
                b_siguiente.interactable = false;

            }
        }

        if (nombre == "Tablet")
        {
            if (contadortablet == 0)
            {
                p_unotablet.SetActive(true);
                p_dostablet.SetActive(false);
                p_trestablet.SetActive(false);
                p_cuatrotablet.SetActive(false);
                b_anterior.interactable = false;
                b_siguiente.interactable = true;


               }
```

```
else if (contadortablet == 1)
            {
                p_unotablet.SetActive(false);
                p_dostablet.SetActive(true);
                p_trestablet.SetActive(false);
                b_anterior.interactable = true;
                b_siguiente.interactable = true;

            }
            else if (contadortablet == 2)
            {
                p_dostablet.SetActive(false);
                p_trestablet.SetActive(true);
                p_cuatrotablet.SetActive(false);
                b_anterior.interactable = true;
                b_siguiente.interactable = true;

            }
            else if (contadortablet == 3)
            {
                p_trestablet.SetActive(false);
                p_cuatrotablet.SetActive(true);
                b_anterior.interactable = true;
                b_siguiente.interactable = false;

            }
        }

    }


    //Se utiliza para avanzar al siguiente paso, cambiando el valor de la
variable para que el método Update ejecute el cambio
    public void BotonSiguiente()
    {


        if (nombre == "TapaT" && contadortapa < 2)
        {
            if (contadortapa == -1)
            {
                b_siguiente.gameObject.SetActive(true);
                b_anterior.gameObject.SetActive(true);
                b_empezar.gameObject.SetActive(false);

                contadortapa++;

            }
            else
            {
                contadortapa++;
            }
        }
        if (nombre == "Tablet" && contadortablet < 3)
        {
            contadortablet++;

        }


       }
```

```
//Se utiliza para retroceder al paso anterior, cambiando el valor de la
variable para que el método Update ejecute el cambio
    public void BotonAnterior()
    {
        if (nombre == "TapaT")
        {
            if (contadortapa > 0)
            {
                contadortapa--;

            }
        }

        if (nombre == "Tablet")
        {
            if (contadortablet > 0)
            {
                contadortablet--;

            }
        }

    }

    //Se utiliza para enviar datos a Unity Analytics
    public void Analisis()
    {
        if (nombre == "TapaT")
        {
            Analytics.CustomEvent("Analisis",
            new Dictionary<string, object>
            {
                {"Target", nombre},
                {"Paso", contadortapa }
            }
                );
        }
        else if (nombre == "Tablet")
        {
            Analytics.CustomEvent("Analisis",
           new Dictionary<string, object>
           {
                {"Target", nombre},
                {"Paso", contadortablet }
           }
               );
        }
    }


   }
```

ANEXO 12. DEFAULTTRACKABLEEVENTHANDLER.CS

```
/*============================================================================
=
Copyright (c) 2019 PTC Inc. All Rights Reserved.

Copyright (c) 2010-2014 Qualcomm Connected Experiences, Inc.
All Rights Reserved.
Confidential and Proprietary - Protected under copyright and other laws.
============================================================================*
/

using UnityEngine;
using UnityEngine.UI;
using Vuforia;

/// <summary>
/// A custom handler that implements the ITrackableEventHandler interface.
///
/// Changes made to this file could be overwritten when upgrading the Vuforia
version.
/// When implementing custom event handler behavior, consider inheriting from
this class instead.
/// </summary>
public class DefaultTrackableEventHandler : MonoBehaviour,
ITrackableEventHandler
{
    #region PROTECTED_MEMBER_VARIABLES

    protected TrackableBehaviour mTrackableBehaviour;
    protected TrackableBehaviour.Status m_PreviousStatus;
    protected TrackableBehaviour.Status m_NewStatus;

    #endregion // PROTECTED_MEMBER_VARIABLES

    #region UNITY_MONOBEHAVIOUR_METHODS

    protected virtual void Start()
    {
        mTrackableBehaviour = GetComponent<TrackableBehaviour>();
        if (mTrackableBehaviour)
            mTrackableBehaviour.RegisterTrackableEventHandler(this);
    }

    protected virtual void OnDestroy()
    {
        if (mTrackableBehaviour)
            mTrackableBehaviour.UnregisterTrackableEventHandler(this);
    }

    #endregion // UNITY_MONOBEHAVIOUR_METHODS

    #region PUBLIC_METHODS

    /// <summary>
    ///     Implementation of the ITrackableEventHandler function called when
the
    ///     tracking state changes.
    /// </summary>

    ///
```

```
//Variables
    public Text aviso;
    public Button b_empezar;
    public GameObject botones;

    public void OnTrackableStateChanged(
        TrackableBehaviour.Status previousStatus,
        TrackableBehaviour.Status newStatus)
    {
        m_PreviousStatus = previousStatus;
        m_NewStatus = newStatus;

        Debug.Log("Trackable " + mTrackableBehaviour.TrackableName +
                  " " + mTrackableBehaviour.CurrentStatus +
                  " -- " + mTrackableBehaviour.CurrentStatusInfo);

        if (newStatus == TrackableBehaviour.Status.DETECTED ||
            newStatus == TrackableBehaviour.Status.TRACKED ||
            newStatus == TrackableBehaviour.Status.EXTENDED_TRACKED)
        {
            aviso.text = "";
            b_empezar.interactable = true;
            botones.SetActive(true);
            OnTrackingFound();
        }
        else if (previousStatus == TrackableBehaviour.Status.TRACKED &&
                 newStatus == TrackableBehaviour.Status.NO_POSE)
        {
            aviso.text = "Vuelve a enfocar el objeto";
            b_empezar.interactable = false;
            botones.SetActive(false);
            OnTrackingLost();

        }
        else
        {
            // For combo of previousStatus=UNKNOWN +
newStatus=UNKNOWN|NOT_FOUND
            // Vuforia is starting, but tracking has not been lost or found yet
            // Call OnTrackingLost() to hide the augmentations
            OnTrackingLost();
        }
    }

    #endregion // PUBLIC_METHODS

    #region PROTECTED_METHODS

    protected virtual void OnTrackingFound()
    {
        if (mTrackableBehaviour)
        {
            var rendererComponents =
mTrackableBehaviour.GetComponentsInChildren<Renderer>(true);
            var colliderComponents =
mTrackableBehaviour.GetComponentsInChildren<Collider>(true);
            var canvasComponents =
mTrackableBehaviour.GetComponentsInChildren<Canvas>(true);
```

```
// Enable rendering:
            foreach (var component in rendererComponents)
                component.enabled = true;

            // Enable colliders:
            foreach (var component in colliderComponents)
                component.enabled = true;

            // Enable canvas':
            foreach (var component in canvasComponents)
                component.enabled = true;
        }
    }


    protected virtual void OnTrackingLost()
    {
        if (mTrackableBehaviour)
        {
            var rendererComponents =
mTrackableBehaviour.GetComponentsInChildren<Renderer>(true);
            var colliderComponents =
mTrackableBehaviour.GetComponentsInChildren<Collider>(true);
            var canvasComponents =
mTrackableBehaviour.GetComponentsInChildren<Canvas>(true);

            // Disable rendering:
            foreach (var component in rendererComponents)
                component.enabled = false;

            // Disable colliders:
            foreach (var component in colliderComponents)
                component.enabled = false;

            // Disable canvas':
            foreach (var component in canvasComponents)
                component.enabled = false;
        }
    }

    #endregion // PROTECTED_METHODS
   }
```