Escuela Politécnica Superior de Linares

Grado en Ingeniería de Tecnología de Telecomunicación

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

SISTEMA DE REALIDAD AUMENTADA (AR) PARA SERVICIO DE INFORMACIÓN TURÍSTICA EN PASAJE COMERCIAL

Alumno: **David García Ortiz**

Tutor: Prof. D. Raúl Mata Campos
Depto.: Ingeniería de Telecomunicación

Septiembre, 2017

AGRADECIMIENTOS

En primer lugar, quisiera agradecer a toda mi familia los ánimos y el apoyo que han aportado a lo largo de la realización de este proyecto.

Sin olvidarme de mis amigos, los cuales han sido mis mayores críticos en cuanto a diseño, y también han proporcionado mucho apoyo.

Por último, quisiera dar las gracias a mi tutor de Trabajo de Fin de Grado, Raúl Mata Campos, por permitirme trabajar en este proyecto.

RESUMEN

Resumen

La realidad aumentada es una tecnología que permite aumentar el mundo real que percibimos con elementos virtuales. En esta memoria se describe el uso de esta tecnología, entre otras, para obtener información a tiempo real sobre tiendas del pasaje del comercio de Linares.

La aplicación que describimos es capaz de proporcionar información comercial mediante posicionamiento GPS, también enfocando con la cámara un marcador o nombre de la tienda, cuenta además con reconocimiento de códigos QR y utiliza diferentes funciones y aplicaciones que se encuentran dentro del dispositivo. Este sistema proporciona información de carácter estático, como videos o audios, al que se le suma también información de carácter dinámico, mediante acceso a enlaces web.

En este trabajo observaremos de manera detallada como ha sido el proceso evolutivo de esta aplicación.

Abstract

Augmented Reality is a technology which allows to increase he real world we perceive with virtual elements. In this report, we describe the use of this technology, among others, for obtain real-time information about shops in the commercial passage of Linares.

The application we describe i sable to provide commercial information using GPS positioning, also focusing on the camera a markers or store name, it also has QR code recognition and uses different functions and applications that are inside the device. This system provides static information, such as videos or audio, which also adds dynamic information, through access to web links.

In this work we will observe in detail as has been the evolutionary process of this application.

INDICE

INDICE DE FIGURAS

Figuras del capítulo 2

Figuras del capítulo 3

Figuras del capítulo 5

CAPITULO 1

INTRODUCCIÓN

La realidad aumentada (RA) ha estado creciendo estos últimos años a grandes pasos siendo una tecnología emergente, que combina la visión que tenemos del mundo real con elementos virtuales, dando la oportunidad de crear experiencias visuales con grandes utilidades en el ámbito social y comercial.

Hasta hace poco tiempo, no más de 6 años, esta tecnología resultaba muy costosa y complicada para poder hacer uso de ella diariamente y mucho menos por el usuario medio. Sin embargo, a día de hoy, cualquier usuario que disponga de un dispositivo móvil inteligente (Smartphone), los cuales se comercializan a nivel mundial y que prácticamente todas las personas cuentan con uno o más de estos dispositivos, cuentan con la capacidad computacional que requiere esta tecnología, algo impensable hace tan solo unos pocos años.

Estos últimos años se ha oído halar cada vez más de los conceptos de "*Realidad Virtual"* y "*Realidad Aumentada"* (RV/RA). Estas tecnologías tenían el concepto de ser "curiosas" y "futuristas", pero como podemos comprobar por las diferentes empresas que apuestan por esta tecnología, han tomado un carácter de "necesario" y "posible".

Entre estas empresas podemos destacar las compañías de entretenimiento visual, como *PlayStation* (Sony), *Xbox* (Microsoft) o *Wii* (Nintendo); así como los gigantes del software, *Facebook* y *Google.* Han proporcionado un salto generacional muy destacado en la tecnología virtual y aumentada, que ha sido aprovechada por la comunidad de desarrolladores, dando lugar a la creación de nuevas experiencias y herramientas que se pueden usar en otros campos.

El propósito de este proyecto es adquirir nuevos conocimientos en sistemas de realidad aumentada y diseñar e implementar una aplicación con esta tecnología. Esta aplicación debe suministrar información de carácter turístico/comercial sobre una calle comercial de una ciudad, en este caso se escogió el pasaje del comercio de Linares ya que me encontraba en esta ciudad en el proceso de la aplicación. Los requisitos que debíamos de cumplir era la utilización de marcadores y posicionamiento GPS, ya que cuentan con un gran valor a día de hoy en el mundo de las aplicaciones de esta tecnología.

La finalidad del proyecto es que un usuario que se encuentre de visita en la ciudad para la que se creó la aplicación, en este caso Linares, tenga la posibilidad de llegar hasta este pasaje del comercio sin necesidad de conocerse la ciudad y además le aporte información relacionada con las tiendas a través de la realidad aumentada.

CAPITULO 2

MARCO TEORICO

En este apartado vamos a conocer una serie de conocimientos básicos necesarios para el correcto entendimiento de nuestro proyecto. Por tanto, vamos a explicar que es una aplicación móvil, cuando surgieron, sus características y como se distribuyen, acto seguido, intentaremos aclara el concepto de realidad aumentada, su historia, su tecnología y a que hace referencia. Para terminar, explicaremos las distintas plataformas que pueden servir para realizar una aplicación como la que se ha llevado a cabo en este proyecto.

2.1 Aplicación móvil (App)

Una aplicación móvil está diseñada para ser ejecutada en teléfonos inteligentes (Smartphone), tabletas y otros dispositivos como smartwatch, las cuales permiten al usuario realizar alguna tarea específica, ya sea de ámbito profesional o de ocio.

Estas Apps nos las encontramos en diversas plataformas de distribución, aunque también las podemos descargar desde alguna página de Internet, asumiendo el riesgo de si es maliciosa. En general, cada sistema operativo tiene su propia plataforma de distribución.

Dependiendo de los desarrolladores o compañías, nos encontramos con aplicaciones gratuitas, de pago o las denominadas "*freemium"*, las cuales inicialmente son gratuitas, pero dan la posibilidad de realizar compras dentro de la App, normalmente este tipo de aplicación está orientada al entretenimiento.

2.1.1 HISTORIA DE LAS APLICACIONES MÓVILES

A ciencia cierta no se sabe cuál fue el inicio de las Apps, lo que si es cierto es que llevan con nosotros muchos años, más de lo que comúnmente se pueda pensar.

Todo empezó en la década de los 90 cuando se popularizó el uso de los teléfonos móviles, los cuales traían precargadas aplicaciones muy básicas como agenda, contactos, juegos y en algunos casos e-mail. Estos dispositivos se denominaban "*Featurephones"*, como los que se muestran en la siguiente figura. [1]

Figura 2.1 *Featurephones.*

El ejemplo de App que más fama obtuvo fue el juego “Snake”, perteneciente a la compañía Nokia, este juego tuvo tal éxito que muchas personas preferían comprarse un dispositivo de esta marca solo por poseer dicho videojuego.

En el año 2000 se introdujo a los dispositivos móviles la tecnología WAP (Wireless Application Protocol), la transmisión de datos EDGE (lo que conocemos como la tecnología 3G) y su conexión a internet permitiendo un mayor desarrollo de las aplicaciones ya existentes, aunque con las restricciones de los fabricantes de sistemas operativos, que no permitían involucrar a desarrolladores externos, estancaban esta industria.

Fue en el año 2008 cuando Apple lanza su plataforma de distribución y, además, publica su primer SDK (Software Development Kits) para darle acceso a las personas interesadas, es aquí cuando el abanico de aplicaciones se incrementa rápidamente. Android no tardó mucho tiempo en subirse al carro del desarrollo de aplicaciones móviles, creando su propia plataforma de distribución y liberando su SDK, para un uso más factible por parte de los desarrolladores independientes. [2]

Desde entonces las App se han convertido en herramientas que forman parte de nuestras vidas cotidianas, enfocadas a todo tipo de tareas, con lo que tienen un mercado muy amplio.

2.1.2 CARACTERISTICAS

Las aplicaciones están escritas por necesidad en algún lenguaje de programación compilado (normalmente es distinto para cada sistema operativo) y su funcionamiento, diseño y recursos están encaminados para aportar funcionalidades tales como: [3]

- Acceso rápido y sencillo a las funciones de los dispositivos móviles sin necesidad de autentificar cada vez que se use por el usuario, para ello se suelen pedir los permisos necesarios en la instalación de la App.

- Que funcione en todos los sistemas operativos, para así abarcar todo el mercado.

- Satisfacer alguna necesidad o problema.

- Que se pueda vincular con las redes sociales.

- Respecto al diseño, la App debe ser atractiva a la vista.

- Funciones fuera de línea, que no requieran conexión a internet para poder disfrutar de la aplicación.

- Seguridad y protección de datos, para ganarte la confianza de los usuarios.

- Actualizaciones regulares, ya sea para solucionar problemas que te indican los usuarios o para mejorar la aplicación añadiendo nuevos contenidos. [4]

2.1.3 DISTRIBUCIÓN

En cuanto a su distribución, existen diferentes plataformas de distribución o tiendas enfocadas normalmente a cada sistema operativo, las cuales suelen ser gestionadas por la misma empresa que desarrolla el sistema. Las tiendas organizan las aplicaciones y cada una tiene normas diferentes de retribución y publicación. Las más conocidas son:

- **Google Play**

  Es una plataforma de distribución de software en línea desarrollado por Google Inc. para dispositivos con sistemas operativos Android. Fue lanzada en octubre de 2008.

En la plataforma se encuentran disponibles tanto aplicaciones gratuitas como de pago. Su interfaz es rápida y sencilla de utilizar.

- **App Store**

Fue el primer servicio de distribución de aplicaciones, concretamente se lanzó el 10 de julio de 2008, contando con un catálogo de 500 aplicaciones y en tan solo cuatro días consiguió 10 millones de descargas. En esta plataforma se encuentran todas las aplicaciones disponibles para el sistema iOS, ya sea Smartphone o PC convencional.

Además de esta versión, en 2012 se creó la *App Store Volume,* la cual solo está disponible en EEUU y para empresas, esta versión permite la compra en grandes volúmenes de aplicaciones, las cuales se usan para mejorar el negocio o para regalarlas a modo de promoción.

- **Windows Store**

Windows Phone Store es la plataforma de distribución de Microsoft, para los dispositivos que cuentan con el sistema operativo Windows. Fue lanzada en octubre de 2010. Actualmente se ha unificado con Windows 10 en todas sus plataformas (Smartphone, PC y consolas) haciendo posible el enlace entre todos los dispositivos con este sistema operativo. [1]

- **Amazon AppStore**

Se trata de una aplicación móvil, que a su vez realiza la función de distribuidor de otras aplicaciones móviles. Actualmente solo es compatible para dispositivos con sistema operativo Android. Esta plataforma es la primera alternativa de Google Play, ya que tiene bastantes aplicaciones y cuenta con el respaldo de una empresa potente detrás. [5]

- **F-Droid**

Es un repositorio de aplicaciones para Android, donde únicamente se alojan aplicaciones de software libre y código abierto, siendo en este ámbito una alternativa a Google Play, donde encontramos aplicaciones con un denominador común: tienen licencia libre, por lo que su código fuente es accesible y puedes editarlo para adaptar la aplicación a tus necesidades. [6]

2.1.4 TIPOS DE APLICACIONES

A la hora de desarrollar una App, debemos tener en cuenta cómo va a estar construida técnicamente y ver qué tipo de aplicación es la que nos conviene para desarrollarla. De manera general, podemos dividir las aplicaciones móviles en tres tipos distintos: Nativas, Hibridas y Web.

2.1.4.1 Nativas

Este tipo de aplicaciones están diseñadas para ejecutarse en un dispositivo y sistema operativo especifico, por tanto, para cada sistema operativo utilizaremos un tipo de lenguaje de programación. En el caso de iOS se utilizan los lenguajes *Swift,* aunque hasta hace poco se usaba *Objective C*. Las aplicaciones desarrolladas para el sistema Android lo hacen con lenguaje *java* o *C#.* Para las aplicaciones con sistema Windows se utiliza el lenguaje *C# y .NET*.

A la hora de desarrollar nuestra aplicación en nativo, tendremos que programar en los diferentes lenguajes, para cada sistema operativo, en caso de que queramos abarcar un mercado mayor, ya que, si solo lo hiciéramos en un lenguaje, nos estaríamos limitando a los dispositivos que soporten ese sistema operativo.

Este tipo de aplicaciones suele proporcionar unos resultados más robustos y fluidos, puesto que, al estar programado en el lenguaje nativo, saca mejor partido a las cualidades de cada dispositivo. Según la aplicación que vayamos a desarrollar, nos encontramos con algunas ventajas y también con algunos inconvenientes, que podemos resumir en la siguiente figura. [7]

Algunos ejemplos de aplicaciones nativas son *WhatsApp o Facebook.*

Ventajas	Inconvenientes
• Acceso completo al dispositivo • Mejor experiencia del usuario • Visibilidad en APP Store • Envío de notificaciones o "avisos" a los usuarios • La actualización de la app es constante	• Diferentes habilidades / idiomas / herramientas para cada plataforma de destino • Tienden a ser más caras de desarrollar • El código del cliente no es reutilizable entre las diferentes plataformas

Figura 2.2. *Ventajas y desventajas de aplicaciones nativas*. [8]

2.1.4.2 Web

Este tipo de aplicaciones son las más sencillas y económicas de desarrollar, ya que son un tipo de software que es codificado en un lenguaje soportado por todos los navegadores web y cuya ejecución se hace a través del navegador en internet o a través de una intranet (de ahí que reciban el nombre de App web) y adaptan el formato de su contenido según las dimensiones de la pantalla del Smartphone en cuestión.

Las aplicaciones web están íntimamente relacionadas con el almacenamiento de datos en la nube, es decir, toda la información se guarda de manera permanente en servidores web, también nos envía esta información a nuestros dispositivos cuando sea requerida y nos realiza copias cada cierto periodo de estos envíos dentro de los dispositivos que utilicemos. [9]

Este tipo de aplicaciones utiliza el lenguaje de programación puramente web, es decir, se puede programar con los lenguajes *HTML, CSS (sirve para organizar la presentación y aspecto de una página web) y JavaScript.* Además, se puede portar a cualquier tipo de plataforma (Android, iOS, Windows).

Por el contrario, estas aplicaciones ofrecen una peor experiencia de uso, ya que ignora las características del dispositivo y una menor seguridad, puesto que ésta depende de la seguridad del navegador. Otro inconveniente es que, al tratarse de una web modificada, necesita de una conexión a internet, ya que sin internet la app no funcionará. [10]

En la siguiente tabla se muestra algunas ventajas e inconvenientes de este tipo de aplicaciones.

Ventajas	Inconvenientes
• El mismo código base reutilizable en múltiples plataformas • Proceso de desarrollo más sencillo y económico • No necesitan ninguna aprobación externa para publicarse (a diferencia de las nativas para estar visibles en app store) • El usuario siempre dispone de la última versión • Pueden reutilizarse sitios "responsive" ya diseñados	• Requiere de conexión a internet • Acceso muy limitado a los elementos y características del hardware del dispositivo • La experiencia del usuario (navegación, interacción..) y el tiempo de respuesta es menor que en una app nativa • Requiere de mayor esfuerzo en promoción y visibilidad

Figura 2.3. *Ventajas y desventajas de aplicaciones web.* [8]

2.1.4.3 Hibridas

Este tipo de aplicaciones engloba lo mejor de las aplicaciones nativas y las aplicaciones web, ya que aprovecha al máximo la versatilidad de un desarrollo web y tiene la capacidad de adaptación al dispositivo como una app nativa. Son desarrolladas mediante frameworks externos, basados en lenguajes de programación web como *HTML, C# y JavaScript*, los cuales realizan las conversiones y llamadas al sistema, aprovechando las funcionalidades del dispositivo tales como la cámara, el GPS o la función de llamada. [8]

Una de sus puntos fuertes es que comporta un menor coste que una aplicación nativa y una mejor experiencia de uso que una aplicación web. Dando la posibilidad de agrupar los códigos y distribuirlas en las plataformas de distribución. [10]

Los desarrolladores corporativos utilizan este tipo de aplicaciones para hacer que el soporte del número de dispositivos móviles en la empresa lleve menos tiempo y sea menos costoso. Los desarrolladores privados utilizan este tipo de aplicación para llegar a la mayoría de clientes potenciales al subir sus apps a múltiples plataformas de distribución sin tener que reescribir los códigos para cada tipo de dispositivos. [11]

Sin embargo, este tipo de aplicación es una opción menos robusta y fluida que una aplicación nativa, ya que la conversión al lenguaje nativo desde el lenguaje base no es tan buena como programar íntegramente en aplicaciones nativas, esto se debe a que cada página debe ser renderizada desde el servidor y supone una mayor dificultad de desarrollo.

En la siguiente tabla podemos observar las principales ventajas e inconvenientes de este tipo de aplicaciones.

Ventajas	Inconvenientes
• Es posible distribuirla en las tiendas de iOS y Android. • Instalación nativa pero construida con JavaScript, HTML y CSS • El mismo código base para múltiples plataformas • Acceso a parte del hardware del dispositivo	• Experiencia del usuario más propia de la aplicación web que de la app nativa • Diseño visual no siempre relacionado con el sistema operativo en el que se muestre

Figura 2.4. *Ventajas y desventajas de aplicaciones hibridas*. [8]

2.2 Realidad Aumentada.

En este apartado explicaremos todos los conceptos relacionados con la realidad aumentada. Haremos un estudio de los tipos de realidades que existen, el concepto que se tiene de realidad aumentada, su historia, los niveles que existen, así como la tecnología necesaria entre otras cosas.

2.2.1 INTRODUCCIÓN

Para tener un concepto más claro sobre la realidad aumentada, debemos explicar qué tipos de “realidades” existen a nivel computacional.

En el contexto actual de las telecomunicaciones, con significativos avances en hardware y flujo de datos, se hace posible la creación de una nueva dimensión virtual, en el cual la realidad sea el soporte de la expresión de la información digitalizada. Estando ambos elementos, realidad e información, en un mismo dispositivo que se encuentra en constante interacción con el usuario. El desarrollo teórico de esta técnica está basado en los trabajos de Paul Milgran y Fumio Kishino (1994) quienes definieron una taxonomía basada en el “continuo de la virtualidad” (*virtuality continuum*), el cual explica que la realidad se podría definir como una escala continua que abarca desde el entorno real, hasta un entorno completamente virtual.

Figura 2.5. *Grafico del continuo de virtualidad.*

De izquierda a derecha va aumentando el grado de estímulos generados por ordenadores. De este modo, se reemplaza el mundo real por uno distinto e irreal, creado totalmente con contenidos virtuales.

El extenso campo que se encuentra entre el mundo real y el mundo virtual es lo que denominamos realidad mixta (MR). Dentro de esta clasificación existen varios tipos de realidad mixta. Partiendo desde el mundo real, nos encontramos con los distintos tipos: [12]

2.2.1.1 TUI

Las *Tangible User Interfaces (TUI)* o Interfaces Tangibles de Usuario permite que una persona interactúe con la información digital a través de la física del entorno. Tal vez el ejemplo más significativo de esta tecnología sea *Reactable*, que están orientados a DJs, músicos profesionales, escuelas, museos o centros públicos. Se trata de una mesa con pantalla digital, la cual genera diferentes sonidos según el objeto que pongas. Se puede generar hasta pequeñas piezas musicales con la conjunción de los distintos objetos que se coloquen. [13]

Figura 2.6. *Reactable*.

2.2.1.2 Realidad Aumentada

La *Augmented Reality (AR)* o Realidad Aumentada, la podemos situar en el extremo de continuo más cercana a la realidad propiamente dicha. Esto se debe a que su funcionalidad radica en incluir elementos virtuales en el mundo real. Explicaremos con más detalle esta tecnología a continuación.

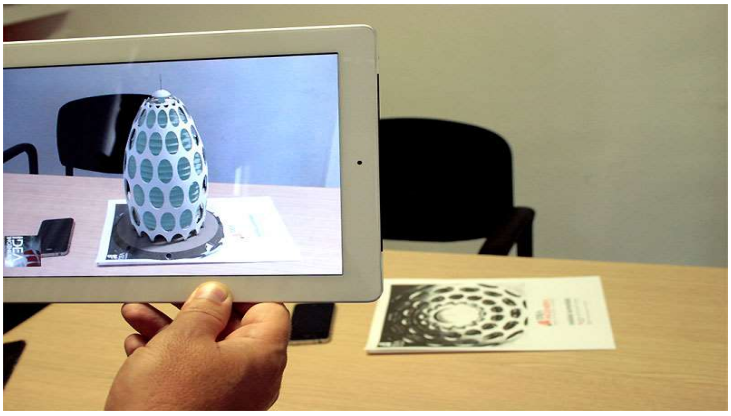


Figura 2.7. *Aplicación de Realidad Aumentada*.

2.2.1.3 Virtualidad Aumentada

El termino de Virtualidad Aumentada se aplica cuando la realidad virtual se nutre de elementos físicos reales como las mano, rostro u objetos. Aunque este concepto no es tan famoso como los anteriores, dado que suele confundirse con la realidad virtual.

Uno de los ejemplos más claros de esta tecnología es cuando salió al mercado el EyeToy, el complemento de PlayStation de Sony. Con este complemento el usuario era captado por una cámara e incluido en el videojuego, de tal manera que sus movimientos interactuaban con el videojuego. [14]

Figura 2.8. *EyeToy.*

2.2.1.4 Realidad Virtual

La Realidad Virtual o *Virtual Reality (VR)*, es el extremo opuesto en el continuo a nuestra realidad sensorial. Consiste en un entorno generado completamente por un ordenador, en el que podemos diferenciar tres grados básicos: [15]

- Realidad inmersiva: se basa en la simulación de un ambiente tridimensional en el que el usuario percibe a través de estímulos sensoriales y se siente dentro del mundo virtual que está explorando. Actualmente está muy de moda, gracias a los últimos dispositivos sacados al mercado como las PlayStation VR, Oculus rift, etc.

Figura 2.9. *Diferentes dispositivos de VR inmersiva.*

- Realidad No inmersiva: son los más conocidos, ya que se trata de la proyección en una pantalla del mundo virtual. Los videojuegos típicos se pueden categorizar dentro de este tipo.

Figura 2.10. *Videojuego.*

- Realidad semi-inmersiva: se caracteriza por ser cuatro pantallas en forma de cubo, tres para las paredes y otra para el suelo, que rodean al usuario proyectando el mundo virtual. A veces se pueden acompañar las imágenes mediante otros dispositivos que enriquezcan la experiencia, tales como olores, aire, sonidos… pero el usuario siempre notará la diferencia entre realidad y mundo virtual.

Figura 2.11. *VR Semi-inmersiva.*

2.2.2 REALIDAD AUMENTADA

En este apartado vamos a definir con mayor profundidad el concepto de Realidad Aumentada, ya que es el que nos involucra en este proyecto.

La Realidad Aumentada (AR) se define como la idea de visualizar a través de un dispositivo tecnológico, el entorno físico, junto con elementos virtuales, creando así una realidad mixta en tiempo real. Se le asigno este nombre ya que añade/aumenta la realidad con contenido virtual. [16]

Se puede decir que el nacimiento de la Realidad Aumentada está ligado al de la Realidad Virtual desde los comienzos de ambas. Será más adelante, cuando la tecnología esté lo suficientemente perfeccionada como para separar las dos ramas.

El termino Realidad Aumentada es acuñado en 1990 por los investigadores de la Boeing, Tom Caudell y David Mizell, a propósito de referirse a la superposición de una pantalla digital que mezcla gráficos virtuales de alta tecnología y proyectarlos a las tablas de usos múltiples y reutilizables, a fin de tener lecturas complementarias a la realidad física.

Aunque quizás la definición más aceptada por el colectivo es la que dio Ronald Azuma en 1997. Esta definición proporciona un punto de partida para cualquier persona interesada en la investigación o el uso de la Realidad Aumentada, y es que se fundamenta en tres conceptos básicos que ha de tener todo aquello que denominemos realidad aumentada:

- Combina elementos virtuales y reales. Es indispensable en esta tecnología.
- Es interactiva en tiempo real. Viene a significar que los contenidos son generados en el mismo instante en el que el usuario los visualiza en el mundo real.
- Esta registrada en 3D. Hace referencia a que solo puede mostrar modelos en 3D, aunque por razones obvias no se toma mucho en cuenta este concepto, ya que, si así fuera, quedarían excluidas las muestras de video, imágenes, audios, etc. [17]

Un esquema básico de un sistema de realidad aumentada suele estar compuesto por una cámara que captura las imágenes del mundo real, conectada a un ordenador que realiza los cálculos necesarios para poder introducir los elementos virtuales dentro de la escena real, proyectando el resultado final hacia una pantalla o unas lentes especiales.

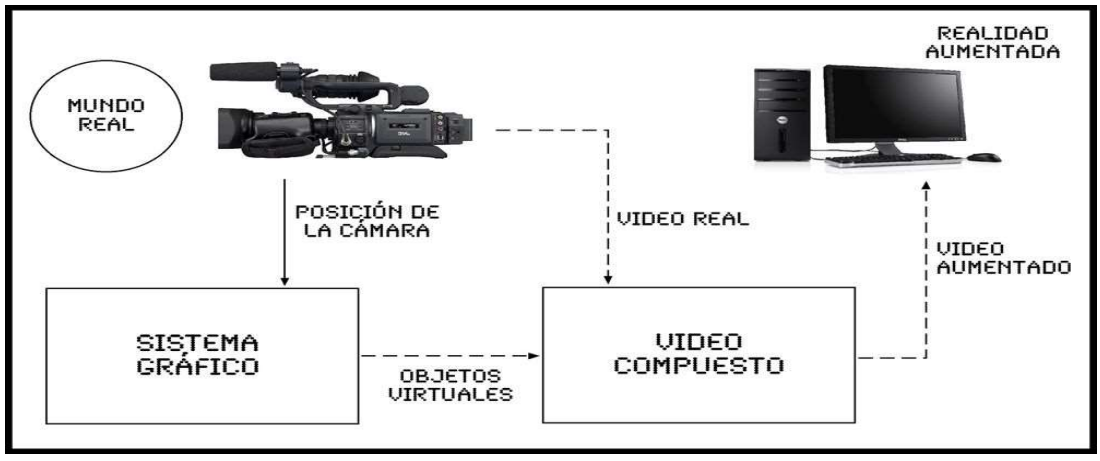


Figura 2.12. *Esquema de un sistema de Realidad Aumentada.*

2.2.2.1 Historia de la Realidad aumentada

Hace ya mucho tiempo que inventos fabricados por el hombre pueden relacionarse con el entorno y proporcionar información adicional a los usuarios. La protohistoria de la realidad aumentada comienza con una curiosa maquina inventada por el filósofo, visionario y realizador de cine Morton Heilig. En 1957 comienza a construir un prototipo con un aspecto muy similar a una máquina de videojuegos arcade, que en aquello tiempos aun no existían. La llamo Sensorama, este nombre pretendía condensar la experiencia que proporcionaba este prototipo, ya que proyectaba imágenes en 3D, a lo que se le sumaba un sonido envolvente, hacia vibrar el asiento y creaba viento lanzando aire al espectador.

A primera vista y siguiendo la clasificación antes mencionada, el prototipo Sensorama tiene más similitudes con la realidad virtual, y por ello se considera a Heilig el padre de la realidad virtual, sin embargo, se podría discutir que se trataba de realidad aumentada, ya que el video que mostraba la pantalla no estaba editado. El contenido adicional (vibraciones, aire, sonido, etc.) se añadían a lo largo del metraje, dependiendo de en qué momento se encontrase, por lo tanto, era información adicional al video mostrado. [18]

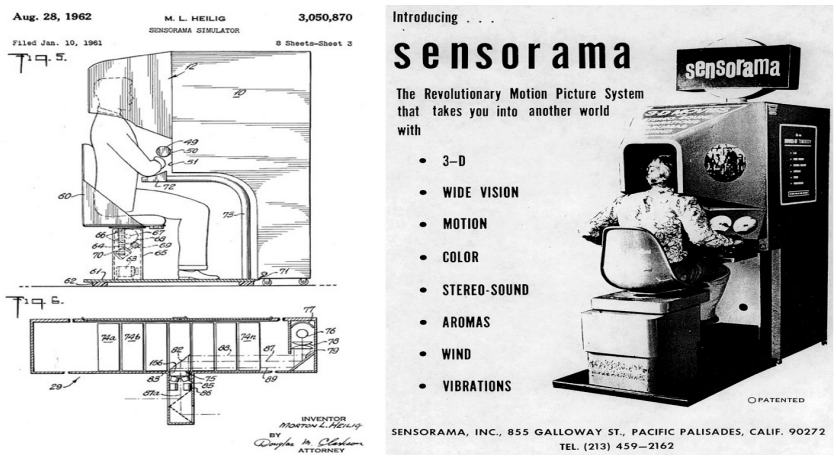


Figura 2.13. *Sensorama.*

Por lo que podemos comprobar como en sus comienzos, la realidad virtual y la realidad aumentada han ido de la mano, compartiendo sus orígenes.

El siguiente dispositivo de realidad aumentada que nos encontramos a lo largo de su historia se diseñó 9 años más tarde, en 1966. El profesor de Ingeniería Eléctrica de Harvard, Ivan Sutherland, creo un dispositivo que podríamos definir como el antecesor de las actuales Oculus Rift y compañía. Sutherland lo llamó "*Human Mounted Display"* (HDM). Pero, al contrario que los dispositivos que tenemos hoy en día, esta máquina no era algo que pudiese manejar con tus propias manos, ya que era tan pesada que se tenía que colgar

del techo y el usuario adaptarse a ella. Sin embargo, supuso un gran avance, a pesar de que solo mostraba modelos básicos en 3D. [18]

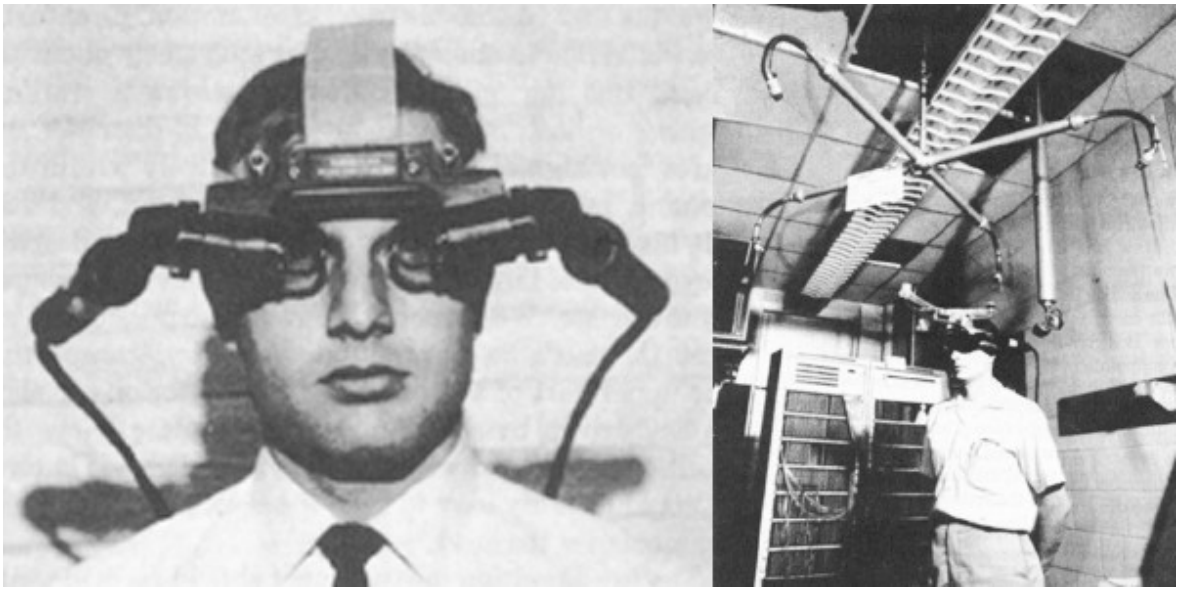


Figura 2.14. *Imágenes del primer HDM.*

En 1985, Myron Krueger creó VideoPlace. Esta es la tecnología predecesora del famoso Kinect de Microsoft, ya que era el usuario el que interactuaba con el contenido digital proyectado en una pantalla. Fue la primera vez que los dos mundos se unían sobre el mismo plano y aunque a día de hoy vemos imágenes y videos de esta tecnología como algo bastante básico, cabe destacar que, para la época, fue un gran hito, ya que daba la oportunidad de interactuar con los dos mundos en tiempo real, dando total libertad al usuario. [19]

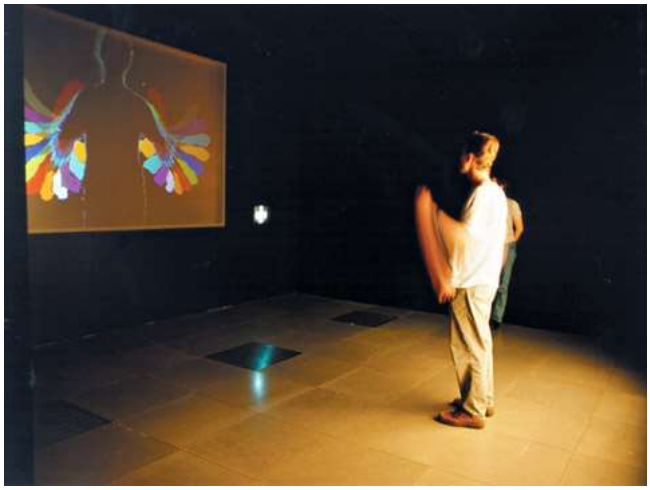


Figura 2.15. *VideoPlace.*

En el año 1990, el investigador en Boeing, Tom Caudell, crea el termino de Realidad Aumentada. Caudell estaba implicado en los desarrollos que la compañía realizaba para

mejorar el proceso de fabricación, donde usaba un software para mostrar los planos de cableado sobre las piezas producidas. Por esta misma fecha, L-B Rosenberg desarrollaba para las fuerzas aéreas de los Estado Unidos, la que se conoce como el primer sistema de realidad aumentada. Se trataba de un dispositivo que daba consejos al usuario sobre cómo realizar ciertas tareas a medida que estas se presentaban, es decir, como una especie de guía virtual.

Figura 2.16. *Tom Caudell.*

El otro gran trabajo de realidad aumentada realizado ese mismo año surgía en la Universidad de Columbia, donde Steven Feiner, Blair MacIntyre y Doree Seligmann, creaban KARMA *(Knowledge-based Augmented Reality for Maintenance Assitance)*, un HDM que interactuaba con una impresora. El dispositivo proyectaba una imagen en 3D para dar instrucciones al usuario sobre como recargar la impresora, en lugar de acudir al manual de uso. [18]

Figura 2.17. *KARMA.*

A partir de entonces, los avances en la realidad aumentada y virtual fueron incrementándose a grandes pasos. En 1995 la compañía Nintendo lanza su Virtual Boy, un HDM que compatibilizaba su consola de sobremesa. En 1999 se crea ARToolkit, a manos de Hirokazu Kato, la gran librería de realidad aumentada. Tan solo un año después, Bruce H. Thomas desarrolla el primer juego al aire libre con dispositivos móviles de realidad aumentada, llamado ARQuake.

Figura 2.18. *ARQuake.*

En 2008 sale a la venta AR Wikitude para los primeros dispositivos Android, que te permite descubrir qué hay a tu alrededor usando la cámara de tu Smartphone. [20]

Ya en el año 2012, Google presentaba sus gafas de AR, llamadas Google Glass. Al año siguiente, Niantic (con la colaboración de Google), sacaba a la venta Ingress, el juego más exitoso para dispositivos móviles con tecnología aumentada hasta la fecha, llamado Pokemon Go.

En la actualidad, varias compañías de prestigio están apostando fuerte por este tipo de tecnología, como Facebook con sus Oculus Rift, Microsoft con Holo Lens, Sony con PlayStation VR o Nintendo con Pokemon Go.

La realidad aumentada y virtual están viviendo sus inicios dorados, ya que a día de hoy contamos con innumerables aplicaciones, abarcando casi todos los sectores del mercado, por lo que la empresa que se posicione como líder en esta industria, obtendrá una considerable ventaja en el futuro próximo.

2.2.3 NIVELES

En función del tipo de activadores de la información asociada a los elementos, podemos distinguir varios niveles.

Dichos niveles fueron propuestos por Lens-FitzGerald, cofundador de Layar, uno de los navegadores de realidad aumentada más extendidos:

- **Nivel 0** *(Physical World Hyper Linking):* Para este nivel se utiliza el reconocimiento en códigos de barra o códigos QR, estos códigos solo sirven como enlaces a páginas web o contenido alojado en la nube u otro servidor, sin mostrar ningún contenido en 3D.
- **Nivel 1** *(Marker Based AR):* En este nivel, las aplicaciones se apoyan en elementos físicos de tamaño pequeño para mostrar el contenido virtual. En un principio solo se contaba con los llamados FrameMarkers, imágenes cuadradas con bordes específicos, que tenían gran similitud en su funcionamiento con un código QR típico. Poco después se añadió el reconocimiento de imágenes, objetos rectangulares, cilíndricos y finalmente cualquier tipo de objeto.

  Cabe destacar que la tecnología actual prefiere un determinado tipo de imagen u objeto para asegurar el correcto funcionamiento de la aplicación, en el desarrollo de la aplicación se explica cómo se han desarrollado los marcadores utilizado en este proyecto.
- **Nivel 2** *(Markless AR):* Este nivel hace uso del hardware del dispositivo, en concreto hace uso del GPS, giroscopio o acelerómetro. Detectan la posición en la que se encuentra el usuario para mostrar el contenido aumentado y superponerlo en los objetos o lugares reales.

  Este tipo de aplicaciones suelen fallar dependiendo de las condiciones meteorológicas y el entorno, ya que son muy variables y debido a que el GPS de nuestro dispositivo tiene un margen de error de entre 8 y 11 metros.
- **Nivel 3** *(Augmented Vision):* Es el menos habitual de ver en el mercado, ya que está en un periodo de integración, debido a que tiene que incorporar todo lo necesario en unas gafas o casco. Estarían formados por una tecnología personalizada que, sin necesidad de marcadores, pudiera entender el entorno, proporcionando a la vez un contenido aumentado específico para cada usuario. Es lo que pretendía hacer Google con el lanzamiento de las Google Glass. [21]

Habiendo explicado los distintos tipos de niveles que existen, podemos comprobar que la mayoría de las aplicaciones de realidad aumentada que se encuentran en el mercado a día de hoy, están comprendidas entre los niveles 1 y 2, debido a que, o bien hacen uso de marcadores (imágenes, objetos, etc.) como muchas aplicaciones de educación, o bien utilizan la posición del dispositivo para mostrarnos el contenido, como guías turísticas que nos marcan el camino o nos dicen la distancia hasta nuestro objetivo.

Por el contrario, los productos que se vendían con realidad aumentada hace unos años, eran con códigos QR. Este nivel esta ya casi obsoleto ya que apenas proporciona contenido virtual, tan solo no abre el navegador. En nuestro proyecto se quería trabajar con los tres primeros niveles de realidad aumentada, ya que para el cuarto nivel necesitaríamos unas Oculus Rift o unas Holo Lens para poder desarrollarlo.

2.2.4 TECNOLOGÍA

En este apartado trataremos de explicar los contenidos físicos y el conjunto de niveles, programas que deben tener los dispositivos, ya sean Smartphone, tabletas u ordenadores. Existen requisitos que debemos tener en cuenta, estos son que cuenten con una pantalla con una buena resolución, una tarjeta gráfica que pueda procesar la información, un procesador potente, un gigabyte mínimo de memoria RAM y con suficiente espacio en el disco duro para no colapsar el dispositivo.

2.2.4.1 Hardware

En general, todos los dispositivos de Realidad Aumentada deben constar de dos partes básicas. Una primera parte que es el “headset”, la cual incorpora el sistema GPS, indispensable para localizar la posición del usuario y una segunda parte que es un “display”, necesario para mostrar al usuario el contenido generado virtualmente que se añade a la imagen real.

Como “display” se suelen emplear dos tipos de pantallas:

- **Pantallas ópticas transparentes** (Optical See-through Display): estas proyectan el contenido virtual sobre una lente transparente a través de la cual, el usuario puede ver el contenido virtual sin necesidad de un ordenador o Smartphone. Un ejemplo de este tipo de pantallas son las Google Glass.

Figura 2.19. *Google Glass.*

- **Pantallas de mezcla de imágenes** (Video-mixed Display): en estas pantallas el contenido virtual se proyecta sobre una pantalla digital. El ejemplo más claro y del

que se dispone hoy en día es el Smartphone, donde podemos ver la realidad física a través de la cámara del dispositivo.

Figura 2.20. *Smartphone de última generación.*

De hecho, los Smartphone se están convirtiendo en el medio más usado por la Realidad Aumentada, gracias a la comodidad que ello implica. Hasta hace pocos años, la tecnología aumentada requería un gran coste computacional, ya que se necesitaba de un ordenador lo bastante potente como para moverla sin problemas. Los Smartphone con los que contamos hoy en día ya son ordenadores de bolsillo más potentes que muchos de los ordenadores de sobremesa de hace 10 años y, además, disponen de todo el hardware necesario para hacer funcionar una aplicación de realidad aumentada en ellos (display, cámara, GPS, etc.) y cuentan con aún más funciones, como acelerómetro, giroscopio, sensores, etc. Por no mencionar las capacidades de la CPU moderna, ya que cuentan con suficiente espacio de disco duro y unas memorias RAM con mucha capacidad.

Tal y como se encuentra el mercado en estos momentos, en pleno crecimiento en este ámbito, muchas compañías han optado por gafas con la tecnología suficiente como para poder representar el contenido aumentado en tiempo real, algunas de ellas utilizan como soporte el Smartphone, otras empresas crean su propio sistema (los cuales suelen ser más voluminosas y pesadas), pero siendo soportables para el uso cotidiano. [22]

2.2.4.2 Software

Es cierto que el hardware es una parte fundamental para esta tecnología, pero es en el software donde radica la magia para fusiones coherente de imágenes del mundo real con imágenes virtuales en 3D.

Teniendo en cuenta lo mencionado anteriormente, las aplicaciones de realidad aumentada actuales dividen sus tareas principales en 3 pasos:

- Reconocimiento de objetos.
- Tracking, o seguimiento de objetos.
- Iluminación y renderizado.

2.2.4.2.1 *Reconocimiento de objetos*

Esta tarea, como ya se ha comentado, depende del nivel de nuestra aplicación. Si estamos ante el caso de una aplicación del nivel 2, se hará uso de procesos de registro de imágenes, tales como detección de esquinas o bordes, de umbral, detección de puntos importantes, etc.

Si por el contrario se trata de una aplicación de nivel 3, se utilizarán los distintos componentes de posicionamiento, para comprobar si existen las coordenadas donde se encuentra el dispositivo en la base de datos, mostrando en caso afirmativo el contenido de la aplicación.

En nuestra aplicación se tienen en cuenta estos dos niveles, haciendo que salte un menú cuando te encuentras en alguna de las entradas del pasaje del comercio de Linares, utilizando el nivel 3, además de utilizar el reconocimiento de texto e imágenes que se corresponde con el nivel 2. [22]

2.2.4.2.2 *Tracking*

Cuando nos referimos a Tracking, estamos hablando del seguimiento del objeto reconocido y su correcto posicionamiento dentro del entorno físico de acuerdo a la posición del usuario, teniendo en cuenta la posibilidad de que el usuario se esté moviendo. Para ello se usan las coordenadas del objeto y las del dispositivo, comparándolas cada frame de la ejecución y actualizando el contenido según las variaciones. [23]

Las aplicaciones actuales son capaces de reconocer vehículos en movimiento, o guardando la posición de un objeto “trackeado”, de tal forma que, si mueves tu dispositivo, el objeto seguirá con sus mismas coordenadas y permanecerá estático, viendo parcialmente si los bordes del dispositivo lo cortan.

En enero de este año, Microsoft ha comprado la patente de un sistema de seguimientos de objetos, con el que será posible localizar objetos cotidianos como las llaves, cartera o teléfonos móviles, gracias a este tracking integrado en las Holo Lens. [24]

2.2.4.2.3 *Iluminación y renderizado*

En esta tarea se introducen los elementos virtuales a los objetos de la escena que ya están identificado y analizado, mediante programas de representación gráfica, pero esto no es suficiente, ya que se debe intentar adaptar a las condiciones del entorno con el objetivo de hacer lo más realista posible la experiencia de realidad aumentada. Esto se consigue mediante dos vías:

- Coherencia geométrica o de posición: esta vía trata de hacer que un objeto virtual se introduzca en una posición determinada, mimetizándose en la escena del entorno real detectado, de forma que parezca que realmente pertenece a ella.
- Coherencia en la iluminación: esta parte trata de la adaptación de la iluminación al objeto virtual, adaptándolo a la iluminación del entorno real. De modo que en una noche oscura podamos ver con claridad los objetos virtuales.

2.2.4.3 Técnicas de visualización

Una vez explicado cómo funcionan las aplicaciones de realidad aumentada, debemos distinguir las tres técnicas principales de visualización:

2.2.4.3.1 Display en la cabeza

Se trata de las ya mencionadas HDM, se componen de una pantalla transparente localizada entre el mundo real y los ojos del usuario, gracias a las nuevas tecnologías podemos incorporarlas a gafas o cascos.

Cuando el usuario está usando este dispositivo, está viendo en todo momento el mundo físico a través de la lente sobre la cual aparecen los contenidos virtuales, siendo su mayor pega que la información gráfica se ve condicionada al campo visual del usuario. [25]

2.2.4.3.2 Display de mano

En este campo el dispositivo más utilizado es el Smartphone, siendo el claro ganador ya que, el contenido virtual se superpone sobre un video grabado en tiempo real, tienen un coste computacional algo elevado, pero con la tecnología actual no supone ningún problema. Las ventajas que tiene son la portabilidad y la multifuncionalidad. [25]

2.2.4.3.3 Display espacial

También conocido como realidad aumentada espacia (SAR), es un término poco conocido, pero es uno de los más prometedores. En este tipo de visualización, el Display no está asociado a ningún usuario, de tal manera que varios usuarios pueden utilizarlo al mismo

tiempo. Esta técnica se consigue gracias a los proyectores, que proyectan el contenido virtual sobre un objeto real, ya sean paredes, mesas, etc.

Este modelo soluciona muchos de los problemas de los anteriores tipos, y el usuario no se ve obligado a llevar un dispositivo, pudiendo interactuar con otros usuarios. [25]

2.3. Estado del arte

En este apartado haremos un estudio de las aplicaciones más destacadas en el año 2016, dentro del ámbito que ocupa nuestra aplicación. Haremos mención de aquellas que utilizan algunas de las funciones de las que dispone nuestra aplicación.

Layar

El uso de imágenes interactivas tiene cada vez más presencia en nuestras vidas, sobre todo en los comercios y anuncios publicitarios que vemos a nuestro alrededor. Esta aplicación te permite crear y accede a contenido virtual desde carteles, revistas, anuncios y códigos QR impresos en los productos, además podrás contemplar contenido extra como videos, cupones de descuento, etc., solo deber enfocar tu cámara del Smartphone o Tablet. También cuenta con Geo Layers, esta función te permitirá encontrar eventos, restaurantes y tiendas que estén a tu alrededor. [26]

Google Translate

Esta aplicación creada por los desarrolladores de Google, permite traducir texto y voz de manera instantánea gracias al reconocimiento de texto, al igual que en nuestra aplicación.

Solo tienes que acceder a la aplicación desde el Smartphone, seleccionar la opción de capturar video y apuntar hacia la imagen que quieras traducir, Google Translate mostrará el mensaje en el idioma que hayas seleccionado, ya sea un anuncio publicitario, una señal de tráfico o el menú de un restaurante. [26]

Field Trip

Se trata de la aplicación pionera de Niantic Inc., que funciona como un guía turístico personal, ya que permite buscar entre diversas categorías los lugares que te interesan durante tu recorrido por una determinada ciudad, como museos, sitios históricos o restaurantes. Solo hay que enfocar con tu Smartphone el lugar del que desees obtener más información y Field Trip desplegará una ficha de datos sobre ese sitio, pudiendo guardar la ubicación del lugar por si se quiere visitar de nuevo. [26]

Wikitude

En esta aplicación solo tienes que introducir un término de búsqueda, como por ejemplo “kebab” y apuntar tu Smartphone a la calle, la aplicación desplegará automáticamente todos los lugares cercanos que estén relacionados con este término en cuestión, así como la distancia a la que te encuentras. [26]

Star Walk

Esta aplicación utiliza la tecnología GPS y los sensores del dispositivo para señalarte en la bóveda celeste las constelaciones, los planetas y las galaxias con información actualizada en tiempo real. Seleccionando cualquier objeto en el cielo, Star Walk te desplegará una ficha con información detallada sobre él. Otras de sus funciones es que puedes ver cómo se veía el cielo en una fecha pasada o cómo se verá en el futuro. [26]

Turismo de Galicia

“Turismo de Galicia” es una aplicación que funciona como guía tanto para los turistas potenciales que quieren conocer la Comunidad gallega como para los que ya se encuentran en ella. La Xunta ha renovado esta aplicación con el objetivo de que sea más fácil localizarla y descargarla.

Gracias a esta aplicación, los viajeros que deseen visitar Galicia podrán conocer de antemano la localización y la información acerca de los recursos y servicios turístico de Galicia, como alojamientos, restaurantes, artesanía, museos o las oficinas de turismos entre otros muchos recursos. [27]

CAPITULO 3

TRABAJO DESARROLLADO

3.1 Estudio del posible contenido utilizado

En este apartado realizaremos un estudio de las posibles plataformas que podemos usar en nuestra aplicación, de este modo haremos una comparativa con las referentes competencias para cada tecnología, plataforma y sistema operativo. En el último apartado, haremos una selección de cada una de ellas.

3.1.1 SISTEMAS OPERATIVO DE APLICACIONES

Este proyecto se basa en la creación de una aplicación para dispositivos móviles, ya sean Smartphone o tablets, por lo tanto, debemos tener en cuenta las diferentes plataformas existentes y qué presencia tienen en el mercado. Las plataformas más destacadas en cuanto a volumen de mercado son iOS y Android, la siguiente plataforma más robusta es Windows Phone, aunque se encuentra a un nivel muy inferior que las antes mencionadas.

A continuación, se muestra un gráfico en el cual podemos observar el porcentaje de ventas de los sistemas operativos móviles desde principios de 2017 a junio de 2017, según el servicio de estadísticas NetMarkertShare. [28]

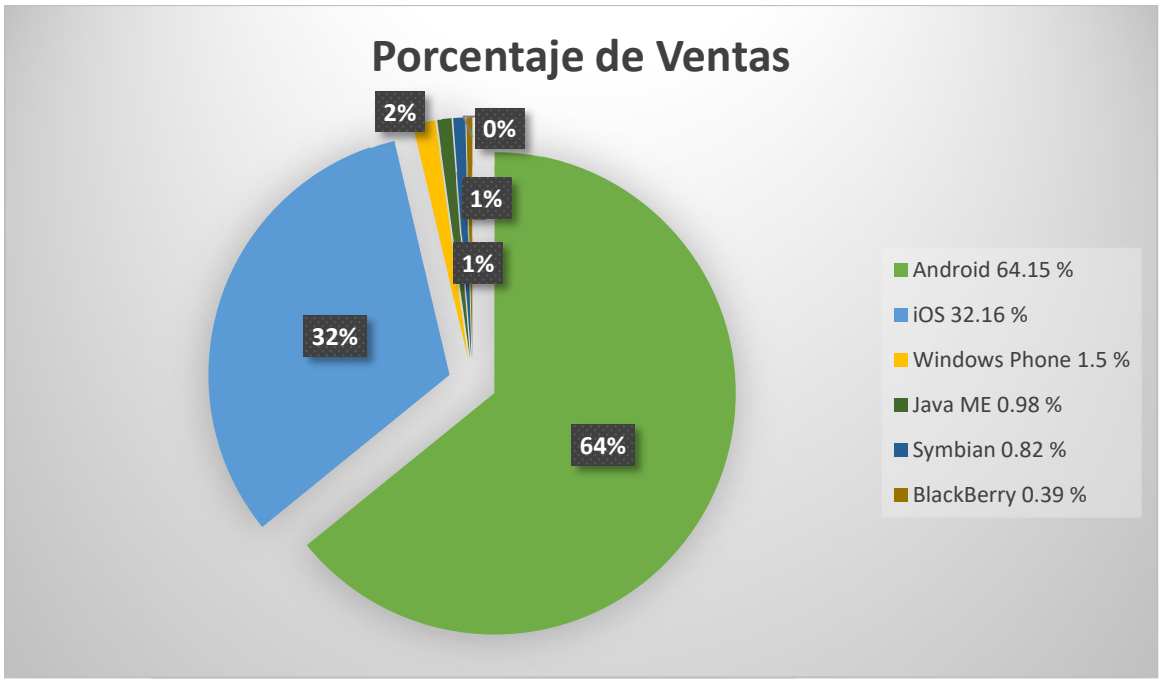


Figura 3.1. *Gráfico de sistemas operativos.*

Como podemos observar en el gráfico, Android e iOS abarcan el 96 % de las ventas de dispositivos móviles a nivel mundial, por lo que son las dos plataformas con mayor interés por parte de los desarrolladores. A continuación, realizaremos un estudio de los tres sistemas operativos más utilizados.

1. **Sistema operativo iOS**

Figura 3.2 *Logotipo de Apple.*

Este sistema operativo pertenece a la multinacional Apple Inc. Originalmente fue desarrollado para el iPhone, bajo el nombre de iPhone OS (en el año 2007). En esta versión las únicas aplicaciones que se podían tener eran las que hoy en día vienen preinstaladas, como Mapas, Reloj, Calculadora, Fotos o iPod. No fue hasta su segunda versión (iPhone OS 2.0) en la cual ya se permitía instalar aplicaciones de tercero y lo que dio lugar al nacimiento de AppStore.

Aunque los cimientos de iOS se crearon mucho antes, pues iOS deriva de OS X (lo que conocemos hoy en día como MacOS), que está basado en Darwin BSD, por lo que podemos decir que iOS es un sistema operativo de tipo Unix.

El sistema se puede dividir en cuatro capas de abstracción: el núcleo del sistema operativo, la capa de Servicios, la capa de Medios y la capa de Cocoa Touch, esta última capa es la que diferencia a iOS con OS X, ya que para OS X (Cocoa) está pensado para usarse con ratón y teclado, mientras que la capa de iOS (Cocoa Touch) está pensado para manejarse mediante una interfaz táctil.

Es el segundo sistema operativo más importante del mercado, en su plataforma de distribución, la AppStore, cuenta con unos 2 millones de aplicaciones, siendo cierto que en cuanto a ganancias mediante estas aplicaciones supera a Google en aproximadamente 100 millones. Esto se debe al elitismo característicos de los usuarios de Apple, que están dispuestos a pagar algo más por las aplicaciones de esta compañía. Por no contar con el hecho de que Apple obtiene ganancias por parte de los desarrolladores, como de los usuarios, ya que para poder desarrollar en este sistema operativo es necesario pagar una cuota anual próxima a los 100 €, por no mencionar que para desarrollar necesitas de un

Mac, ya que Apple solo deja desarrollar y probar las aplicaciones en dispositivos de su propia casa.

No obstante, se trata de un sistema operativo muy potente y fluido, ya que cuenta con actualizaciones inmediatas, ya sea por mejoras en la versión, un error que afecte a ciertos dispositivos o un nuevo tipo de ataque que se aprovecha de los fallos comunes en los navegadores web del mundo. Sea lo que sea, Apple lo arreglará a la mayor brevedad posible, estando la actualización disponible para todos los usuarios desde el minuto cero en que esté disponible su descarga.

Otro punto fuerte de Apple es la integración total con el resto de dispositivos de la casa, ya tengas un iPhone, un iPad o un Apple TV, la experiencia es total y fluida gracias a que Apple diseña, desarrolla y optimiza su sistema operativo para un solo hardware. Siendo estos dispositivos versiones reducidas de iOS. La base es la misma y eso hace que funcionen como uno solo.

Otra de las ventajas de iOS con respecto a otros sistemas operativos móviles es su seguridad y la facilidad de uso. Con este sistema operativos tendrás mayor seguridad ante amenazas que puedan proceder de fuentes no fiables. [29]

2. **Sistema Operativo Android**

Figura 3.3. *Logotipo de Android.*

Figura 3. Logotipo Android.

En octubre de 2003 se funda la empresa llamada Android Inc. cuyo objetivo era desarrollar un sistema operativo para móviles basado en Linux, pero en julio de 2005 fue comprada por la multinacional Google por un precio estimado de cincuenta millones de dólares. Una suma importante ya que Android Inc. apenas era conocido. Al siguiente año se empieza a trabajar en un sistema operativo basado en el kernel de Linux, con una interfaz dinámica.

En 2007 Google firmaba la Open Handset Alliance (un conglomerado de empresas, como fabricantes de chips, de terminales, operadoras o desarrolladores de software). Se trataba

de una alianza comercial cuyo principal propósito era el desarrollo de estándares abiertos para dispositivos móviles. En el mismo día en que se firmó la alianza anunciaron su primera versión, Android 1.0 Apple Pie, que se presentaba como un sistema operativo móvil totalmente gratuito y Open Source a diferencia de iOS. Esta primera versión tenía mucho margen de mejora y apenas inquietó a la competencia, pero ya disponía de conceptos que a día de hoy son un estándar de los sistemas operativos móviles. A partir de entonces hasta día de hoy muchos fabricantes de teléfonos móviles se han decantado por Android como base para sus modelos de Smartphone, tablets y otros dispositivos. [30]

Al estar basado en Open Source, todo el código del sistema y el software que lo compone puede ser consultado y modificado libremente por cualquier usuario. Esta es una ventaja para los desarrolladores ya que pueden crear todo tipo de aplicaciones usando todas las funciones del teléfono, como usar la función de llamada, o la función de mensajería de texto, la cámara de fotos, etc. Además, Android optimiza los recursos internos del teléfono (memoria, hardware, etc.) con el objetivo de hacer funcionar lo mejor posible la aplicación. Uno de los puntos fuertes de Android es que no se limita a utilizar las aplicaciones y funciones que vienen de serie en un teléfono, sino que permite usar todas aquellas que vayamos incorporando y descargando desde Internet. [31]

Android cuenta con la ventaja de que no hace falta pagar ninguna cuota para hacerte desarrollador, tan solo tendrás que pagar un precio no superior a los 30 € para poder subir tu aplicación a la plataforma de distribución Google Play. Las aplicaciones subidas a esta plataforma son controladas con mucho detalle para evitar el mal uso de ellas, Google trata con mano dura a los desarrolladores que se saltan las reglas de comportamiento, siendo una de las plataformas con mayor contenido de aplicaciones en el mercado, con más de 2,2 millones de aplicaciones. [32]

3. **Sistema Operativo Windows Phone**

Figura 3.4. *Logotipo de Windows Phone.*

Es el sistema operativo desarrollado por Microsoft, que comenzó a desarrollarse bajo el nombre en clave 'Photon' a principios de 2004, pero fue cancelado. No fue hasta 2008 cuando se volvió a reorganizar el grupo de Windows Mobile y comenzó el desarrollo de un nuevo sistema operativo para móviles. Se esperaba que este sistema operativo estuviese terminado para 2009, pero hubo demoras y solo fue lanzada una versión intermedia. A finales de 2010 se lanzó Windows Phone 7, pocos meses después Microsoft llego a un acuerdo con Nokia para usar este sistema operativo y así reemplazar el sistema Symbian, sin llegar a tener mucho éxito. Después de varios intentos fallidos, Microsoft optó por una línea distinta de lo habitual, sacando a la luz 'Windows 10 Mobile', el cual permite enlazar todas sus plataformas en un mismo sistema operativo, esto quiere decir que su diseño y funcionamiento será el mismo que el ordenador de tu casa, proporcionando un mismo entorno para todos sus dispositivos, y si además eres usuario de 'Xbox' (la video consola de Microsoft) mejor aún, pues dicha consola también cuenta con la misma interfaz. [33]

3.1.2 TECNOLOGÍAS DE REALIDAD AUMENTADA

Existen una gran variedad de tecnologías que trabajan con realidad aumentada, aunque no todas proporcionan una fácil difusión y otras no permiten trabajar con código abierto, por lo que las tecnologías que vamos a estudiar son las más robustas y las más usadas en estos últimos años, ya que podemos encontrar una gran variedad de tecnologías aplicadas a la realidad aumentada, aunque muchas de ellas sólo nos permiten crear aplicaciones sencillas y no podremos explotar al máximo los objetivos de nuestro proyecto.

Debemos tener en cuenta que la realidad aumentada crece a grandes pasos y por lo tanto siempre aparecerá un nuevo software que se ajuste mejor a la aplicación que vayamos a realizar.

ARToolkit

Figura 3.5 *Logotipo AR Toolkit.*

ARToolkit fue desarrollado originalmente por Hirokazu Kato en 1999. Actualmente se mantiene como un proyecto de código abierto alojado en SourceForge con licencias comerciales en ARToolWorks. [35]

ARToolkit es un conjunto de librerías para C/C++ que se utilizan para desarrollar aplicaciones de realidad aumentada. Para crear estas aplicaciones, ARToolkit cuenta con una serie de funciones para la captura de video y para la búsqueda de ciertos patrones, en las imágenes deseadas, utilizando técnicas de visión por computador. [34]

En lo que respecta a la categoría de Realidad Aumentada para web, la librería de programación ARToolkit ha sido portada a Flash (FLARToolkit) y a Silverlight (SLARToolkit), lo que ha permitido el lanzamiento de múltiples campañas de promoción. En estas, el usuario imprime un marcador, la presenta a una webcam y esta superpone algún tipo de mensaje, presentación 3D o algún juego. [36]

Vuforia

Figura 3.6 *Logotipo de Vuforia.*

Vuforia es una plataforma de desarrollo de software que facilita a los programadores de aplicaciones móviles un motor de reconocimiento de imágenes muy potente y una gran variedad de herramientas sin que se vean obligados a preocuparse por las limitaciones de índole técnica. [37]

Originalmente fue desarrollado por Qualcomm hasta 2015 que fue vendida a PTC, una compañía estadounidense, la cual mantendrá el objetivo de inició Qualcomm, proponer un nexo entre el mundo real y el digital a través de la realidad aumentada. Su SDK está

constantemente evolucionando y es compatible con Android, iOS, UWP y Unity. Vuforia no es de código abierto, pero su rango de precio es razonable, afortunadamente no hay coste inicial para desarrollo o educación. Vuforia usa la visión de la computadora para entender lo que “ve” en la cámara y crea un modelo del entorno, cuando procesa los datos, el sistema puede ubicarse en el mundo, sabiendo sus coordenadas. [38]

AR-Media

Figura 3.7 *Logotipo de AR-Media.*

La plataforma AR-Media en un framework de desarrollo estructurado y modular que incluye diferentes módulos de software organizados de acuerdo a una arquitectura especifica. Los módulos incluyen: seguimiento en tiempo real, representación en tiempo real e interfaces. [39]

Total Immersion

Figura 3.8 *Logotipo Total Immersion.*

Total Immersion es líder en probar soluciones de realidad virtual y aumentada para e-commerce, comercio móvil, retail y marketing de la marca.

Esta empresa fue creada por Valentin Lefevre y Bruno Uzzan en 1998, su primer proyecto estaba orientado a deportistas, ya que consistía en una cinta de correr y un video para permitir correr a los deportistas por las calles de diferentes ciudades. La experiencia de consumo de alta calidad fue lo que dio a la empresa la posibilidad de ampliar su visión.

Total Immersion fue el pionero del D'Fusion, el software que permite la fusión de realidad y animación, que llegó a ser conocido como realidad aumentada (AR). Esta fusión convirtió a la empresa en un fenómeno mundial, atrayendo a clientes como Disney y FedEx. [40]

Tango

Figura 3.9 *Logotipo de Tango.*

Este sistema de realidad aumentada fue creado por Google y ha sido desarrollado para dar un salto de calidad a las aplicaciones de AR con un mejor funcionamiento e interacción con el entorno real. El conjunto de cámaras y sensores que usa un dispositivo con Tango permite calcular el relieve, el movimiento y la profundidad con mayor precisión que con una sola cámara, como en el resto de plataformas de AR, ya que estas se basan en los datos que proporciona el dispositivo móvil y las imágenes de la cámara para simular los elementos. Por tanto, para realizar una aplicación AR se necesita un dispositivo con varias cámaras, como la tableta Lenovo Phab2 Pro, que cuenta con 3 cámaras. [41]

Metaio

Figura 3.10. *Logotipo de Metaio.*

Metaio fue fundada en 2003 en Munich, Alemania por el actual director general, Thomas Alt y el actual director Peter Meier, la compañía surgió de un proyecto interno de Volkswagen. En 2005 Metaio lanzó la primera aplicación de AR de consumidor final llamada KPS, que permitía al usuario poner muebles virtuales en una imagen de su sala de estar. [42]

A pesar de que la licencia fuera de pago, el coste no era excesivo. También disponía de una herramienta gratuita llamada "Creator" la cual permitía hacer aplicaciones pequeñas de forma intuitiva. Y de otra herramienta, llamada "Junaio", la cual se basaba en el GPS para mostrar contenidos cercanos a la posición del usuario.

Su mayor punto fuerte era su tecnología SLAM, la cual estaba optimizada para seguir funcionando incluso con variaciones del entorno.

No obstante, Metaio fue comprada por la empresa Apple en 2015 y retirada del mercado, dejándola solo para uso interno.

Layar

Figura 3.11. *Logotipo Layar.*

Fundada en 2009, fue una de las primeras plataformas que nos permitía desarrollar aplicaciones de realidad aumentada basadas en posicionamiento. Lo que significa que tan solo usaba el GPS y el acelerómetro, al poco tiempo añadió el reconocimiento de códigos QR. [43]

En un principio solo estaba disponible para Android, aunque no se tardó mucho en dar soporte a iOS. El auge de layar fue disminuyendo al comprobarse que sus competidores daban más posibilidades de desarrollo.

Layar fue comprada por la compañía BlippAR, la cual se dedica a ofrecer a distintos clientes soluciones comerciales basadas en aplicaciones de realidad aumentada. [44]

AR Crowd

Figura 3.12. *Logotipo de AR Crowd.*

ARCrowd proporciona al usuario una herramienta totalmente online y gratuita que no requiere de ningún tipo de aplicación para pc, Tablet o Smartphone, usando cualquier navegador actual podrás disfrutar de la realidad aumentada. Este software permite desarrollar ARbooks, los cuales se pueden compartir en las redes sociales más populares de hoy en día, como Facebook, Google Plus, Twitter o enviar por correo. [45]

3.1.3 PLATAFORMA DE DESARROLLO

Para el desarrollo de aplicaciones en la mayoría de los dispositivos tendremos la opción de hacerlo en cada sistema operativo que deseemos abarcar o bien, a través de aplicaciones hibridas que nos permitan abarcar dispositivos con diferentes sistemas operativos con el mismo desarrollo, aunque nos pueden limitar en cuanto a uso interno del dispositivo. Debemos tener en cuenta que nuestra aplicación hará uso de la realidad aumentada, la cual requiere, generalmente, de uso del hardware interno de los dispositivos. No obstante, existen ciertas plataformas que permiten desarrollar en un lenguaje intermedio que cuando compilan generan un código nativo para cada sistema operativo. Otro de los factores a tener en cuenta es que nos proporcione soporte para trabajar con las herramientas de desarrollo de la mayoría de tecnologías de realidad aumentada.

UNITY

Figura 3.13. *Logotipo de Unity.*

Unity es un motor gráfico orientado a la creación de videojuegos, que con el paso de los años ha ido aumentando el número de plataformas para las que permite desarrollar (Windows, Xbox, PlayStation, Wii, etc.), que con el paso del tiempo y el auge de las aplicaciones móviles también dio soporte para Android, iOS y Windows Phone. Esta plataforma fue desarrollada por la empresa Unity Technologies en el año 2004. El éxito de Unity se debe al enfoque en las necesidades de los desarrolladores independientes que les era imposible crear su propio motor del juego ni las herramientas o licencias necesarias.

La primera versión de Unity se lanzó en el año 2005 y fue construida exclusivamente para funcionar en equipos de la plataforma Mac, consiguiendo el éxito suficiente como para continuar con el desarrollo del motor y herramientas. En el año 2010 se lanzó la versión 3, dedicada a introducir más herramientas que los estudios de alta gama, con el fin de captar el interés de los desarrolladores más grandes, mientras que proporciona herramientas para equipos independientes y más pequeñas. La última versión, Unity 5, fue lanzada en 2015, permitiendo crear aplicaciones íntegramente en la plataforma, haciendo uso de interfaces de usuario (UI), los cuales permiten añadir botones, textos, imágenes, videos, audios, etc. [46]

Unity cuenta con 3 modelos de licencias, Unity Personal es la versión gratuita, tiene un tope de ingresos de $100 mil dólares, al llegar a dicho toe será necesario subscribirse a la licencia Plus o Pro. Unity Plus (enfocado a desarrolladores móviles) con un tope de ingresos de $200 mil dólares, cuenta con más prestaciones que la versión gratuita, para subscribirse a esta licencia se debe pagar una mensualidad de $35 dólares con un compromiso de suscripción de 12 meses. Unity Pro es la versión completa con acceso a todos los servicios y herramientas que proporciona Unity, no tiene límite de ingresos y se puede adquirir por un precio de $125 dólares al mes con un mínimo de 12 meses. [47]

Otro dato importante de Unity es, al igual que Google, Apple o Windows, que dispone de su propia plataforma de distribución (Asset Store), donde se pueden encontrar desde modelos 3D, interfaces gráficas, audios e incluso código (scripts), que los usuarios proporcionan gratuitamente o los venden, según la calidad y el esfuerzo que conlleve el contenido creado. [48]

Todo este desarrollo en los últimos años ha hecho posible que muchas tecnologías diversas opten por dar soporte en esta plataforma. La realidad aumentada (AR) no es una excepción. Unity es compatible con la mayoría de softwares de realidad aumentada existentes en el mercado (Vuforia, Metaio, ARToolkit, etc.).

ANDROID STUDIO

Figura 3.14. *Logotipo de Android Studio.*

Android Studio es un entorno de desarrollo integrado (IDE) que se utiliza para la creación de aplicaciones. Este entorno de desarrollo está basado en la tecnología IntelliJ IDEA, esta plataforma ha sido creada para sustituir a Eclipse, que ha sido una de las mayores plataformas hasta llegar Android Studio. Además, cuenta con un gran editor de códigos y proporciona más funciones que aumentan tu productividad en el proceso de compilación de aplicaciones para Android como un rápido emulador con varias funciones, un entorno unificado que permite desarrollos para todos los dispositivos Android, puedes aplicar cambios mientras tu aplicación se está ejecutando sin necesidad de compilar un nuevo APK y una gran variedad que hace que Android Studio sea una de los entornos de desarrollo más utilizado. [49]

La primera versión estable fue lanzada en el año 2014, creado por Google, desde un primer momento ha sido publicado como un entorno gratuito a través de la Licencia Apache y está escrito en Java. Aunque en un primer momento solo se desarrolló para la plataforma de Microsoft Windows, hoy en día podemos desarrollar aplicaciones Android en Mac OS y GNU/Linux. [50]

Este entorno de desarrollo contiene el Software Development Kit (SDK) de Android, indispensable para poder desarrollar en esta plataforma, ya que cuenta con las plataformas de Android y las APIs de la versión de Android para la que queramos programar.

3.1.4. SELECCIÓN DE LOS CONTENIDOS UTILIZADOS

En este apartado haremos un estudio de las tecnologías que escogeremos para realizar nuestro proyecto. Puesto que cada uno cuenta con sus ventajas y desventajas, o simplemente se adapta mejor al entorno de trabajo que queremos realizar.

Para el sistema operativo la decisión fue fácil, en nuestro caso se optó por elegir Android, ya que para desarrollar en iOS debíamos pagar una cuota elevada solo por desarrollar y además debíamos disponer de un IPhone y un Mac para poder desarrollarlo.

Tampoco se optó por el sistema operativo Windows Phone debido a la falta de dispositivos con los que probar nuestro proyecto, ya que lo más común en el mercado es el sistema operativo Android como se ha mencionado anteriormente.

Para la selección de la plataforma de desarrollo nos basamos en que tuviese una fácil interfaz para crear el contenido, además de una buena documentación para poder buscar toda la información necesaria, ya que debíamos aprender desde cero como desarrollar en estos entornos y sin una buena base este proyecto se haría muy tedioso. También era necesario que diese soporte para la mayoría de tecnologías de realidad aumentada. Por ello se escogió la plataforma de Unity.

A la hora de seleccionar una tecnología de realidad aumentada, y sabiendo que Unity es compatible con la mayoría de estas tecnologías, nos decantamos por elegir Vuforia, puesto que contaba con documentación para su implementación, también es una de las tecnologías más potentes a día de hoy. Las otras tecnologías no contaban con la potencia que nos proporciona Vuforia, puesto que ARCrowd se basa en una tecnología online y nuestro interés era enlazarlo con la plataforma de desarrollo. Layar no cuenta con los mismos servicios y se quedaba corta en comparación con Vuforia. En el caso de Metaio solo nos proporcionaba de forma gratuita herramientas que no permiten explotar todos los

servicios de realidad aumentada. Tampoco se escogió Tango ya que no disponíamos de dispositivos con más de dos cámaras. Para el caso de Total Immersion, no dispone de paquete de enlace con Unity, por tanto, no se pudo probar su funcionamiento. La tecnología AR-Media nos proporciona la tecnología ARToolkit, el cual, si se puede enlazar con Unity, de hecho, se enlazó y se probó el funcionamiento y a la hora de empezar a crear marcadores, ARToolkit solo nos permitía escoger un modelo que venía por defecto llamado Hiro, y nuestro interés era poder realizar nuestros propios marcadores. Tampoco contaba con mucha información de ayuda a la hora de enlazarlo con Unity.

Gracias a la información proporcionada por Vuforia y al gran número de tutoriales que trabajan con esta tecnología enlazada con Unity, no dudamos en utilizarlo para nuestro proyecto.

3.2 Desarrollo de la aplicación

En este apartado se explica el proceso de instalación de todos los contenidos necesarios para lograr con éxito todos los objetivos propuestos en este proyecto. De esta manera se explicará cómo han de descargarse e instalarse los programas necesarios, las herramientas que se utilizan y como se ha diseñado la interfaz de la aplicación hasta su descarga.

Lo realizamos de una manera detallada para facilitar futuros trabajos relacionados con este tipo de proyectos de forma que resulte de utilidad al lector.

3.2.1 DESCARGA E INSTALACIÓN DE UNITY

En este apartado veremos con detenimiento la correcta descarga e instalación de la plataforma de desarrollo. Detallamos las paginas oficiales de donde se descarga y como se han desarrollado los diferentes procesos de instalación para el correcto funcionamiento de la tecnología de realidad aumentada

En la página oficial de Unity, “ http://unity3d.com/es “, nos encontramos las últimas noticias de la empresa, sus productos, información de la plataforma de desarrollo, etc. y en su esquina superior derecha tenemos la opción ‘Obtener Unity’. Al darle click, se carga la siguiente página, donde se puede elegir el tipo de licencia, en nuestro caso elegimos la licencia Personal, ya que como comentamos anteriormente es la versión gratuita.

Figura 3.15. *Página de descargar Unity.*

Una vez presionemos en la licencia Personal, nos saldrá la opción de descargar el instalador que utiliza un Asistente de Descarga, el cual tiene instrucciones detalladas paso a paso. Desde la versión 5.0 de Unity en adelante, el Asistente de Descarga le permite seleccionar que componentes del editor de Unity quiere descargar e instalar. Este asistente es un pequeño ejecutable con un tamaño inferior a 1 MB.

Tras abrir el instalador debemos aceptar los términos de licencia para poder continuar con la instalación, a continuación, nos da la opción de elegir entre la arquitectura que tiene nuestro PC, en este caso es de 64 bit. Llegamos a la parte más importante del instalador, elegir los componentes que deseamos descargar e instalar, de entre ellos podemos elegir para que plataforma vamos a desarrollar nuestra aplicación, tal y como se muestra en la figura. En caso de no estar seguro de qué componentes quiere instalar, dejar las selecciones predeterminadas. Este instalador también tienes la opción de descargar e instalar Visual Studio, en nuestro caso se instaló previamente la versión Visual Studio 2017, el cual es un entorno de desarrollo integrado para sistemas operativos Windows, soporta múltiples lenguajes de programación, tales como C++, C#, Java, etc.

Visual Studio permite a los desarrolladores crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles y consolas, entre otros.

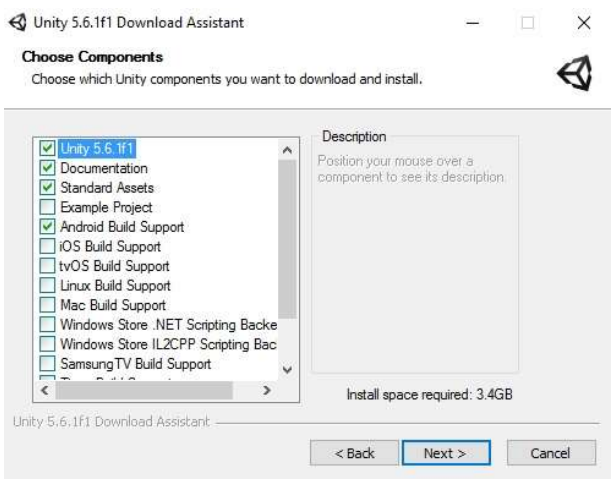


Figura 3.16. *Instalador de Componentes.*

3.2.2 INICIANDO UNITY

Para poder utilizar la plataforma Unity debemos crear una cuenta en la página oficial. Para crear esta cuenta necesitamos poner nuestro nombre, el Nick que queramos que nos aparezca, nuestro país de origen, una cuenta de Email y una contraseña para poder acceder a nuestro perfil. Una vez que le demos a crear, nos mandarán un correo electrónico para confirmar nuestro Email, confirmamos y ya podremos acceder a nuestra cuenta.

Siempre que lance el editor de Unity, se mostrará la pantalla de inicio, si no existe ningún proyecto de Unity en su ordenador, o Unity no sabe dónde se encuentran, pedirá que se cree un proyecto nuevo dándole a la pestaña NEW. Desde esta pantalla podemos nombrar, establecer opciones, especificar la localización del nuevo proyecto, seleccionar en 3D o 2D para el tipo de proyecto (si no se está seguro de qué elegir, lo dejamos en 3D, ya que se puede cambiar esta configuración más tarde) y también hay una opción para seleccionar Asset packages, estos paquetes son contenido pre-elaborado, como imágenes, estilos, efectos de iluminación, entre otras útiles herramientas de creación. Estos assets también pueden ser añadidos vía el editor de Unity, por si en un primer momento no sabemos cuál poner.

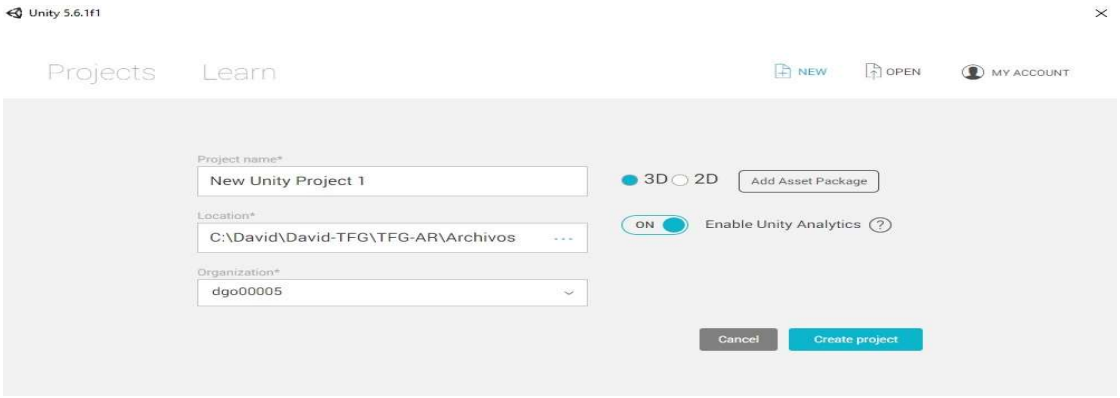


Figura 3.17. *Creando un nuevo proyecto.*

En caso de que ya se tengan proyectos de Unity en nuestro ordenador, la pantalla de inicio se mostrará de esta manera. Podemos elegir de entre los que tengamos para abrir el editor de Unity de cada proyecto.

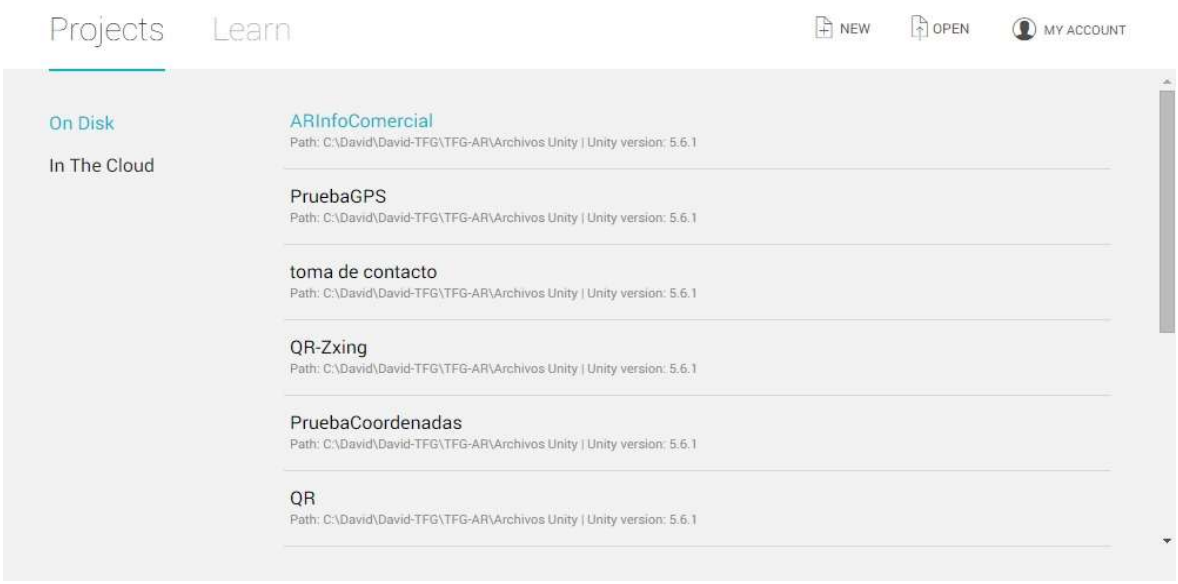


Figura 3.18. *Proyectos existentes.*

Una vez abierta la interfaz del editor de Unity, la ventana principal se compone de ventanas con pestañas que pueden ser re-arregladas, agrupadas o des-adjuntadas y minimizadas. Esto significa que el aspecto del editor puede ser diferente de un proyecto a otro, dependiendo de la preferencia personal. Las ventanas más comunes y útiles se muestran al iniciar el editor.

Figura 3.19. *Interfaz Unity (imagen obtenida de la documentación de Unity).*

The Project Window (ventana del proyecto) muestra los assets de librería que están disponibles para ser usados. Cuando importamos los assets a nuestro proyecto, estos aparecen aquí.

Scene View (vista de escena) nos permite una navegación visual para editar la escena, puede mostrarse en 2D o 3D. También está la pestaña Game View (vista de juego), la cual simula la representación de nuestro juego o aplicación ya terminado.

The Hierarchy Window (ventana de jerarquía) es una representación de texto jerárquico de cada objeto en la escena. Cada elemento en la vista de escena tiene una entrada en la jerarquía, por lo que las dos ventanas están vinculadas. La jerarquía nos muestra la estructura de cómo los objetos están agrupados.

The Inspector Window (ventana del inspector) nos permite visualizar y editar todas las propiedades de las que cuentan cada objeto, para editar este objeto debe estar seleccionado. Ya que cada objeto tiene diferentes propiedades, el contenido de la ventana del inspector va a variar dependiendo de ese objeto. [48]

3.2.3 CONFIGURACIÓN DE UNITY

En este apartado explicaremos con detalle las configuraciones que debemos hacer a Unity, para el correcto funcionamiento de la aplicación.

Para poder trabajar con Android, necesitamos configurar las preferencias de Unity. Estas preferencias son el SDK de Android y el JDK de java, indispensables para poder probar los juegos en los dispositivos móviles.

3.2.3.1 SDK

Para obtener el SDK debemos descargar e instalar Android Studio, desde su página oficial https://developer.android.com/studio/index.html, que cuenta con la herramienta SDK Manager.

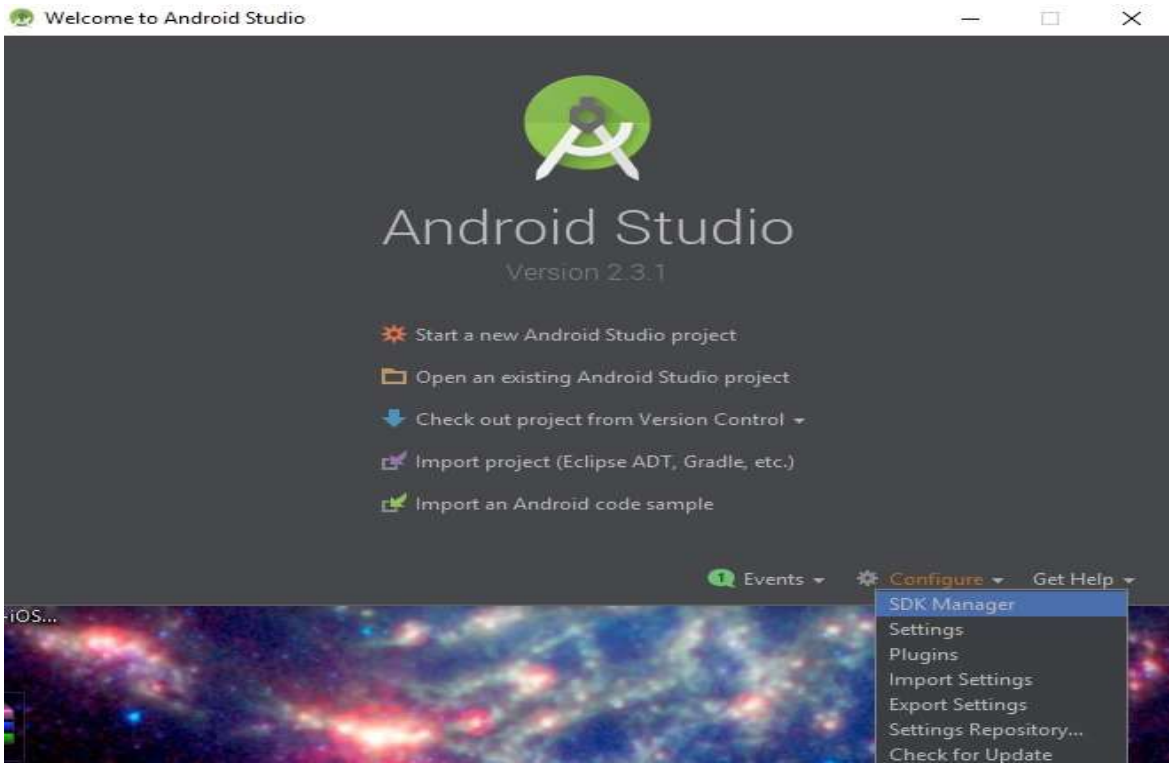


Figura 3.20. *Interfaz de Android Studio.*

Dentro de SDK Manager, debemos descargar las últimas plataformas Android, que se encuentran en la pestaña SDK Platforms, para que nuestra aplicación soporte la mayoría de versiones que hay en el mercado Android a día de hoy. La documentación de Unity recomienda descargar una plataforma Android con un nivel de API igual o superior a 19, por tanto, hemos instalado la versión 4.4 (KitKat) y cuatro de las versiones más modernas, con niveles de API de 22, 23, 24 y 25. Otros de los paquetes que debemos instalar se

encuentran dentro de la pestaña SDK Tools, estos paquetes son: Android SDK Build-Tools (descargamos las versiones más modernas ya que muchas de ellas están obsoletas), Android Emulator, Android SDK Platform-Tools, Android SDK Tools, Documentation for Android SDK, Google Play services, Google USB Driver, Android Support Repository y Google Repository. Muchas de ellas ya vienen marcadas por defecto, una vez las tengamos todas seleccionadas le daremos a Apply y seguidamente a OK.

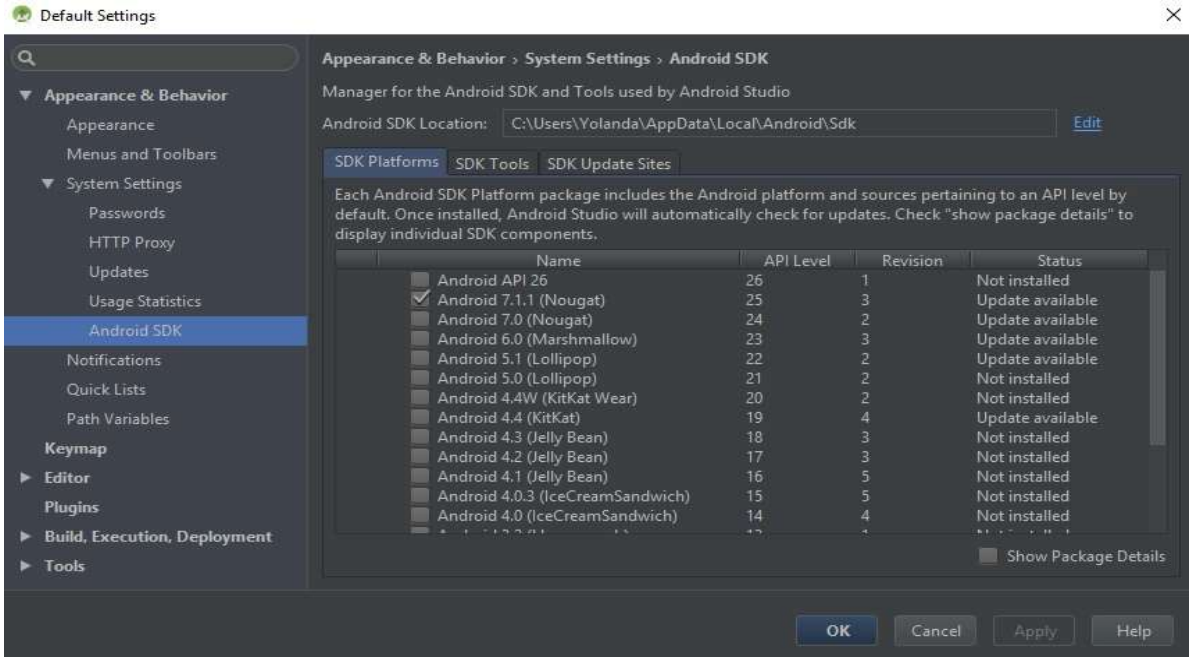


Figura 3.21. *Interfaz SDK Manager.*

3.2.3.2 JDK

Ahora procedemos a descargar e instalar el JDK de Java, desde su página oficial http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html. JDK es un software que provee herramientas de desarrollo para que Unity pueda trabajar con Java. Una vez descargado, lo instalaremos en nuestro PC haciendo doble click en el ejecutable, esta instalación proporcionará una carpeta llamada Java en nuestros archivos de programa, dentro estará el JDK. Lo único que necesitaremos de Java es su ruta de directorio.

Para la configuración de Unity debemos tener la interfaz del editor abierta, desde esta interfaz le damos a la pestaña 'Edit' y dentro de esta le daremos a la opción "*Preference…*" tal y como se muestra en la figura.

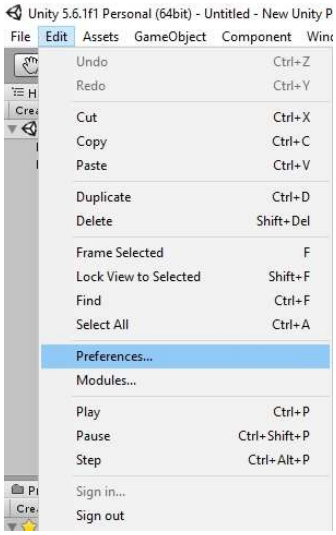


Figura 3.22. *Indicación de dónde encontrar las preferencias.*

Dentro de la ventana de preferencias tenemos que seleccionar la opción "*External Tools*", aquí podemos seleccionar el editor de Script, ya sea Visual Studio o Monodevelop (este editor viene por defecto con Unity). En la sección de Android, debemos especificar la ruta de nuestro SDK y JDK.

Tenemos la opción de poner la ruta directamente, si sabemos dónde se encuentran, o dándole a "*browse*" para abrir el explorador de archivos. La ruta del JDK se encuentra en archivos de programas > Java y seleccionamos el JDK.

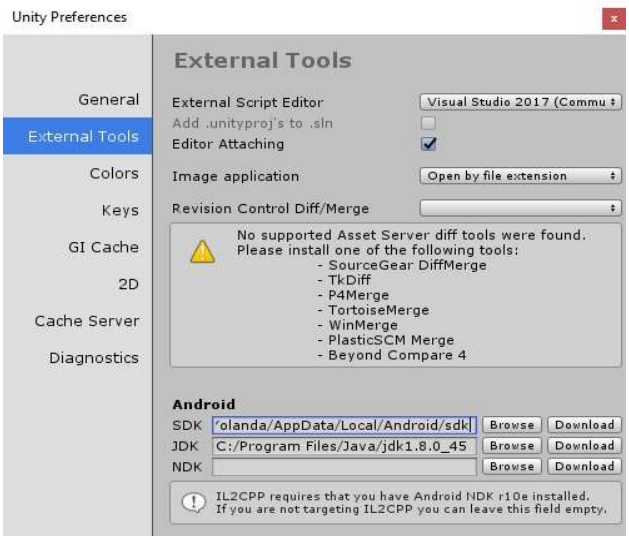


Figura 3.23. *Preferencias de Unity.*

La ruta del SDK se encuentra en Disco local (C:) > Usuarios > nuestro usuario (en mi caso es Yolanda); dentro de nuestro usuario debemos buscar en la barra de navegador, la carpeta oculta "*AppData*", ya dentro de esta carpeta le damos a Local > Android y ya nos aparecerá la carpeta SDK que es la que debemos seleccionar.

3.2.3.3 Selección de entorno

Para tener configurado completamente Unity, debemos elegir la plataforma en la que queremos hacer nuestra aplicación. En nuestro caso se trata de Android.

Para realizar esta configuración, dentro del editor de Unity le damos a la pestaña "*File*" y dentro de este le damos a *"Build Settings*…", tal y como se muestra en la figura.

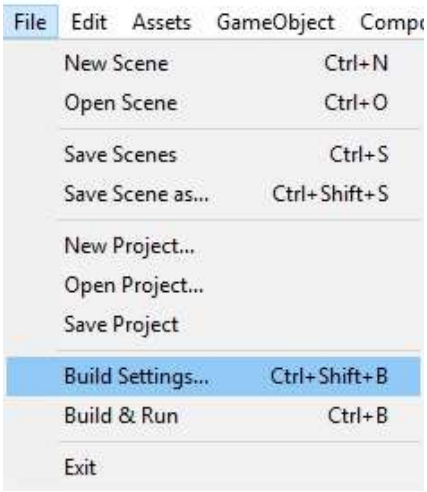


Figura 3.24. *Accediendo a las plataformas.*

Se nos abrirá una nueva ventana con todas las plataformas que soporta Unity, en este momento solo nos centraremos en cambiar de plataforma, más adelante explicaremos que hace cada botón y para qué sirve el panel que aparece encima de las plataformas.

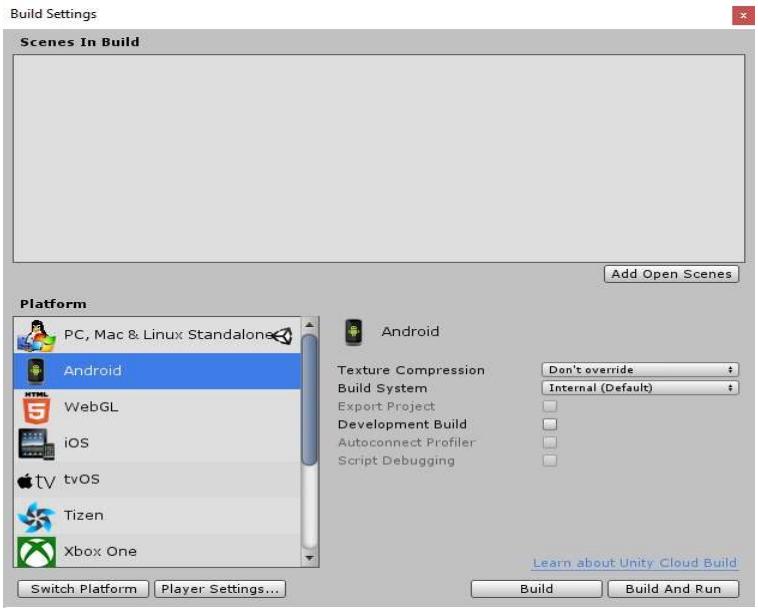


Figura 3.25. *Cambiando de plataforma.*

Por defecto Unity trabaja en la plataforma para PC, MAC y Linux, para poder cambiar de plataforma, debemos seleccionar la que deseemos, que para este caso se trata de Android y a continuación debemos darle al botón de "*Switch Platform*". Una vez pulsado este botón, debemos esperar a que el logotipo de Unity nos aparezca al lado de Android. Ya tenemos configurado Unity para desarrollar aplicaciones en Android y podemos cerrar esta ventana.

3.2.4 INCORPORANDO VUFORIA

En este apartado veremos cómo instalar la herramienta principal para el funcionamiento de la tecnología aumentada, se trata de Vuforia. Además de estudiar su contenido, explicaremos que carpetas se han utilizado en este proyecto.

3.2.4.1 Descarga e Instalación en Unity

Para trabajar con realidad aumentada necesitamos introducir la librería de Vuforia, ésta se descarga desde la página oficial, en el apartado "*Dev Portal*" y en la pestaña "*Downloads*", https://developer.vuforia.com/downloads/sdk. En esta página podemos descargar Vuforia para Android Studio, iOS, UWP (plataforma universal de Windows) y para Unity, dependiendo de la plataforma de desarrollo que tengamos. En nuestro caso descargaremos Vuforia para Unity, con extensión unitypackage, una vez descargado y colocado en la carpeta que deseemos en nuestro PC, volvemos a la interfaz de edición de Unity, ahora accedemos a la pestaña "*Assets*", dentro buscamos "*Import Package*", se desplegará un panel en donde tendremos que seleccionar "*Custom Package*"

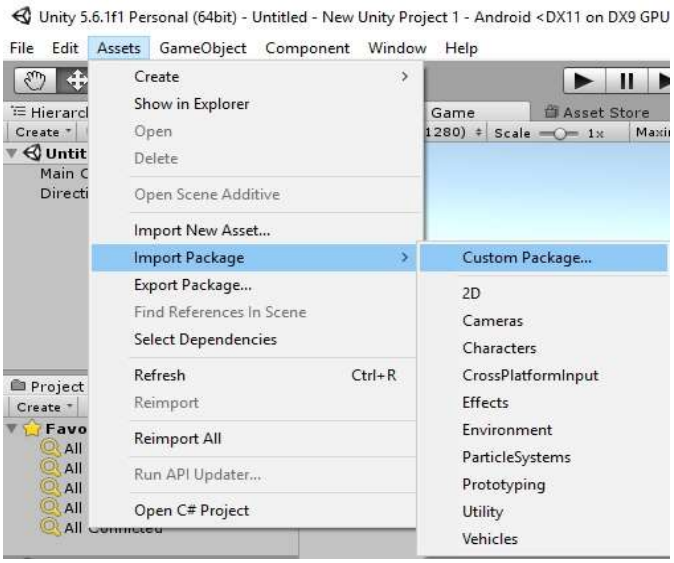


Figura 3.26. *Importando paquetes de Unity.*

Una vez seleccionado "*Custom Package*" se nos abrirá el explorador de archivos, ahora buscamos la carpeta donde hayamos guardado la descarga de Vuforia y lo seleccionamos, tal y como muestra la siguiente figura.

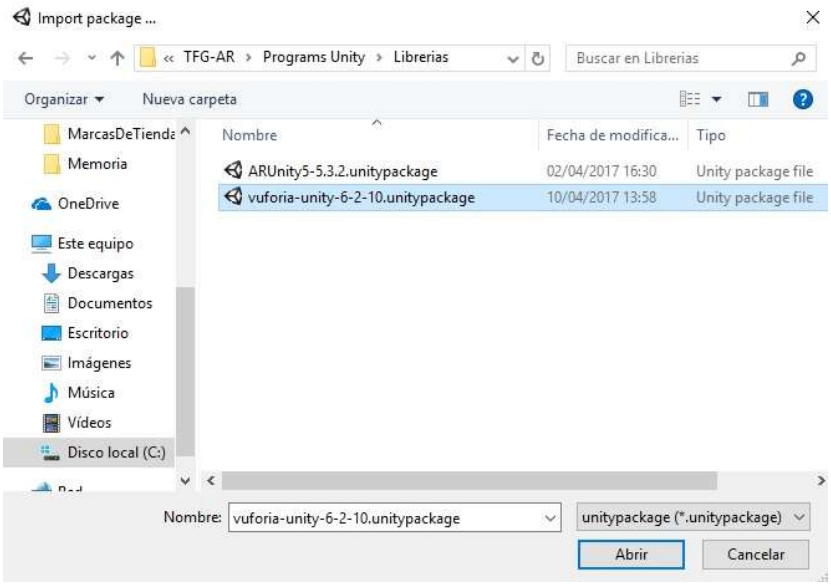


Figura 3.27. *Selección del paquete de Vuforia.*

Tras haberlo seleccionado y abierto, en la interfaz de Unity nos aparecerá una barra de progreso, la cual sirve para descomprimir todo el contenido de la librería de Vuforia. Cuando esta barra de progreso se encuentre por la mitad, nos saltará una nueva ventana con todo el contenido del paquete de Vuforia.

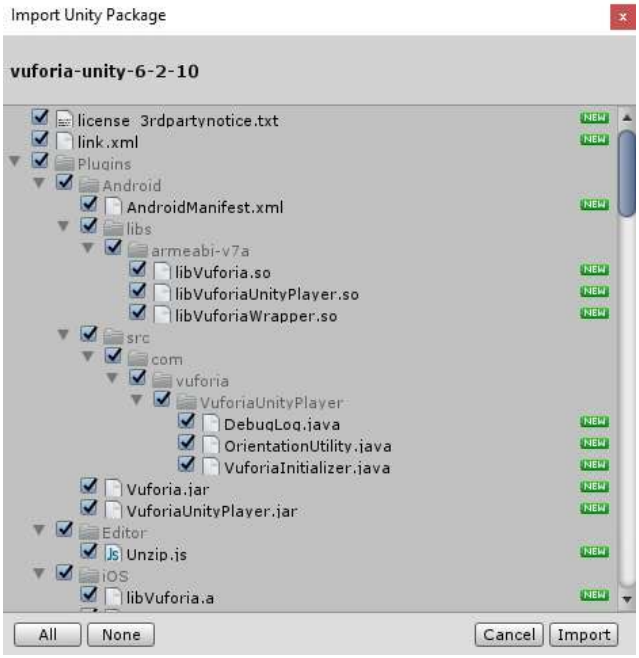


Figura 3.28. *Contenido del paquete de Vuforia.*

Debemos asegurarnos de que todas las pestañas están marcadas, ya que, de lo contrario, podríamos provocar un error que haría que la realidad aumentada no funcionase. Podemos seleccionar todas las pestañas de una sola vez dándole al botón "*All*", y posteriormente le daremos al botón "*Import*". Se cerrará esta ventana y la barra de progreso continuará hasta terminar.

Al terminar la importación de paquetes, Unity nos mostrará el siguiente mensaje:

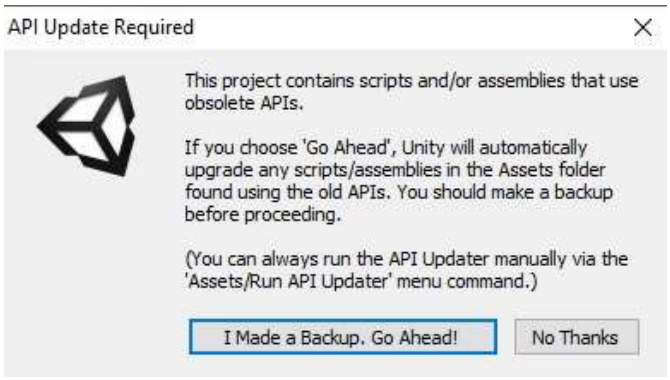


Figura 3.29. *Actualización de APIs obsoletas.*

Este mensaje viene a decir que algunos de los script y ensamblaje de paquetes están usando APIs (Interfaz de Programación de Aplicaciones) antiguas. Para que se actualice con las últimas versiones debemos presionar el botón que pone "*I Made a Backup. Go Ahead!*". En caso de presionar el botón de "*No Thanks*" deberemos actualizar manualmente todos los scripts y ensamblaje de paquetes.

3.2.4.2 Contenido de Vuforia

Tras haber instalado el paquete de Vuforia, en la parte de 'Project Window', que sólo contaba con una carpeta vacía llamada Assets, nos aparece el contenido de este paquete, que son tres carpetas, pero la de mayor interés para nosotros es la de Vuforia, ya que las otras contienen los Plugins para cada plataforma (Android, iOS, etc.) y la configuración del paquete de Vuforia.

Dentro de la carpeta de Vuforia nos encontramos con texturas, materiales y una fuente de texto diferentes a las que proporciona Unity, aunque las dos carpetas que más vamos a utilizar son la de "*Prefabs*" y la de "*Scripts*".

Los "*Prefabs*" son un tipo de Asset que permite almacenar objetos tipo GameObject con componentes y propiedades. Estos actúan como una plantilla a las cuales se les pueden

crear nuevas instancias del objeto en la escena. Los componentes y propiedades de cada Prefab se pueden anular o modificar individualmente.

En esta carpeta contamos con diferentes Prefabs, el más importante de ellos es 'AR Camera', ya que es la que nos proporciona la función de realidad aumentada, indispensable en todas las escenas para realizar nuestra aplicación.

3.2.4.3 Implementando Realidad Aumentada

El Prefab AR Camera contiene un apartado llamado "*Transform*" donde podemos modificar la posición, rotación y escala de la cámara, dos scripts y un "*AudioListener"*. El primer script se llama VuforiaBehaviour, éste se encarga del seguimiento y representación del video de fondo. El segundo script se encarga de mostrar mensajes de error cuando se produce algún fallo, ya sea por no reconocer la cámara o por no tener la clave de licencia de Vuforia.

Para introducir AR Camera en nuestra aplicación tan solo debemos arrastrar este Prefab a la ventana de jerarquía (Hierarchy), todos los Prefabs introducidos en esta ventana se pondrán de color azul. Para que no haya doble sonido o interrupción de la visualización, tenemos que eliminar la perspectiva que viene por defecto, esta se llama "*Main Camera"*, tan sólo tenemos que pulsar el botón derecho del ratón y darle a "*Delete*".

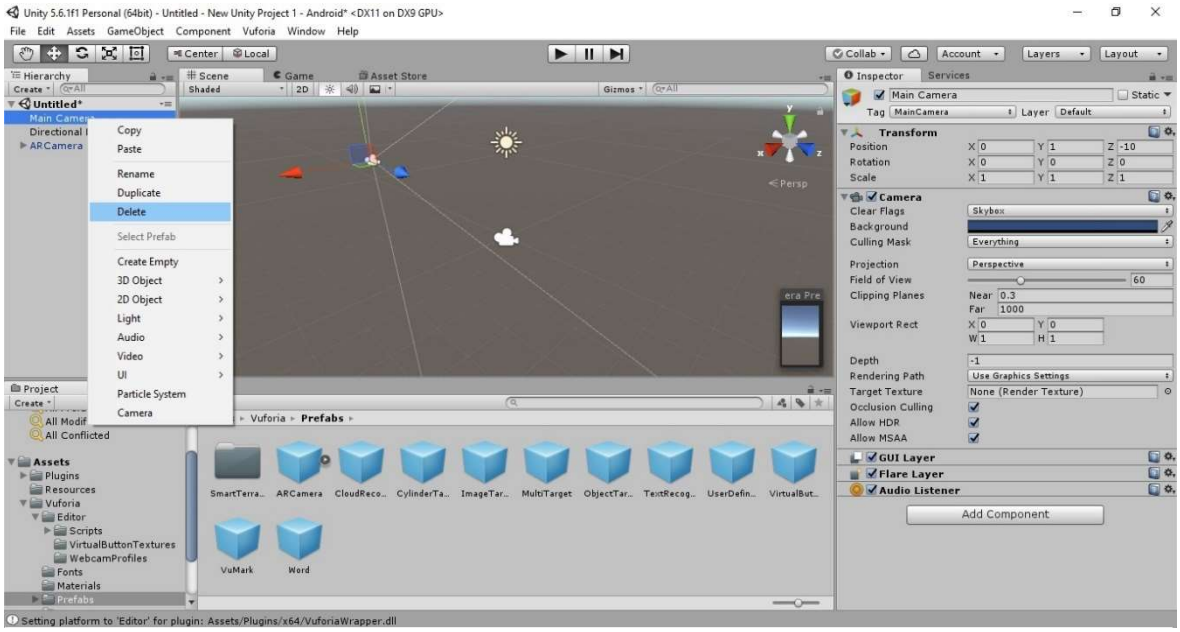


Figura 3.30. *Eliminación de Main Camera.*

Una vez eliminado procederemos a guardar la escena. Las escenas contienen los objetos de nuestra aplicación. Pueden ser usadas para crear un menú principal o niveles

individuales. Cuando creamos un nuevo proyecto, la escena nos aparece con el nombre untitled (sin título) y unsaved (sin guardar).

Para guardar la escena en la cual estamos trabajando, debemos escoger File desde el menú y dentro de este le damos a Save Scene, también se puede guardar presionando Ctrl + S.

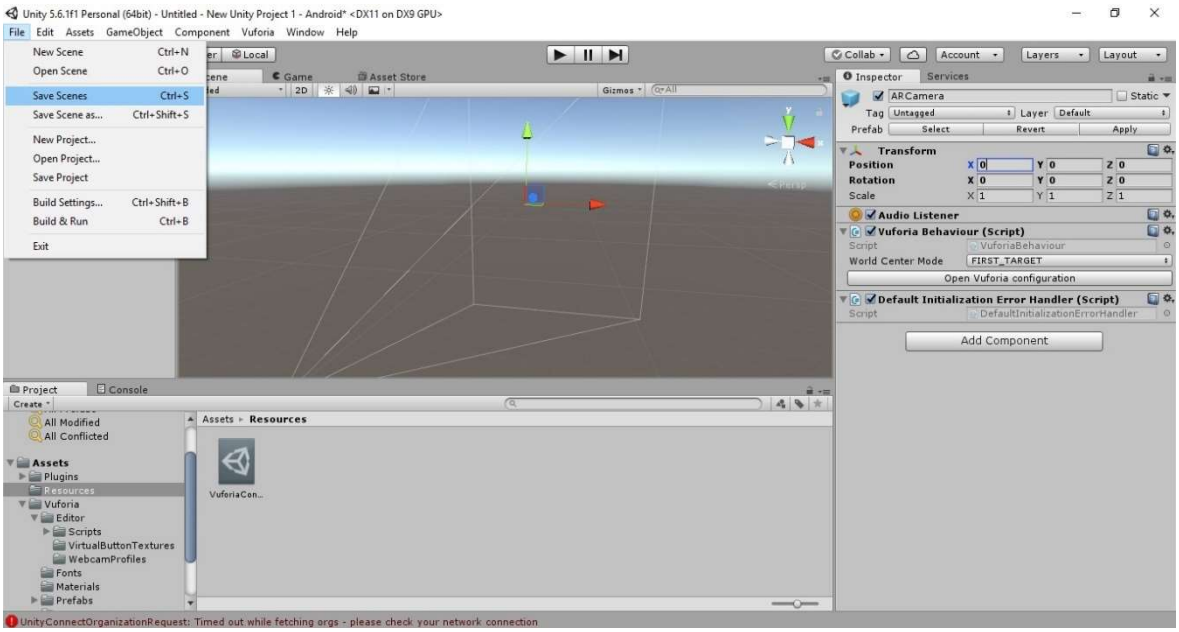


Figura 3.31. *Guardando la escena.*

Las escenas son guardadas como assets, en la carpeta de Assets de nuestro proyecto. Por lo tanto, aparecen en la ventana del Proyecto, también podemos crear una carpeta dentro de la carpeta Assets, en nuestro caso hemos creado una carpeta llamada Scenes, ya que para nuestro proyecto se han necesitado 27 escenas. De esta forma queda más organizado a la hora de cambiar de escena.

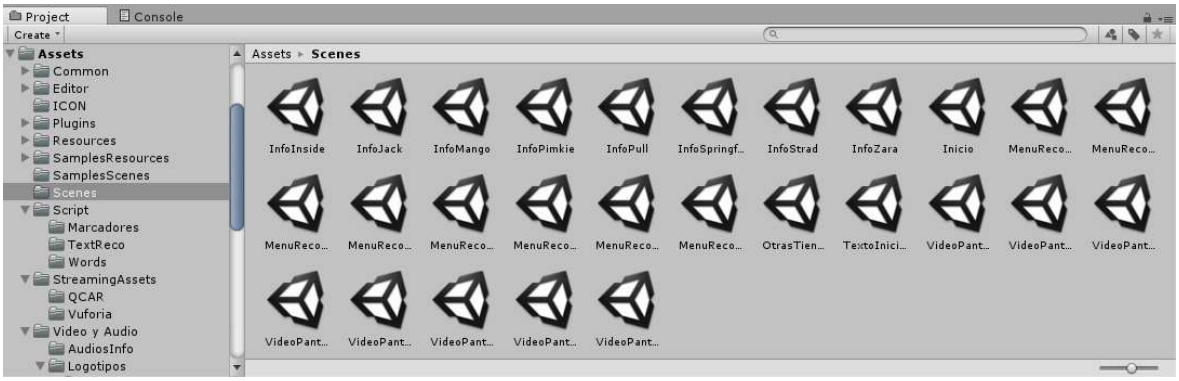


Figura 3.32. *Carpeta con las escenas de nuestra aplicación.*

Todavía no tenemos incorporada la realidad aumentada en nuestra aplicación, ya que, si intentamos darle al Play, Unity nos mostrará este mensaje.

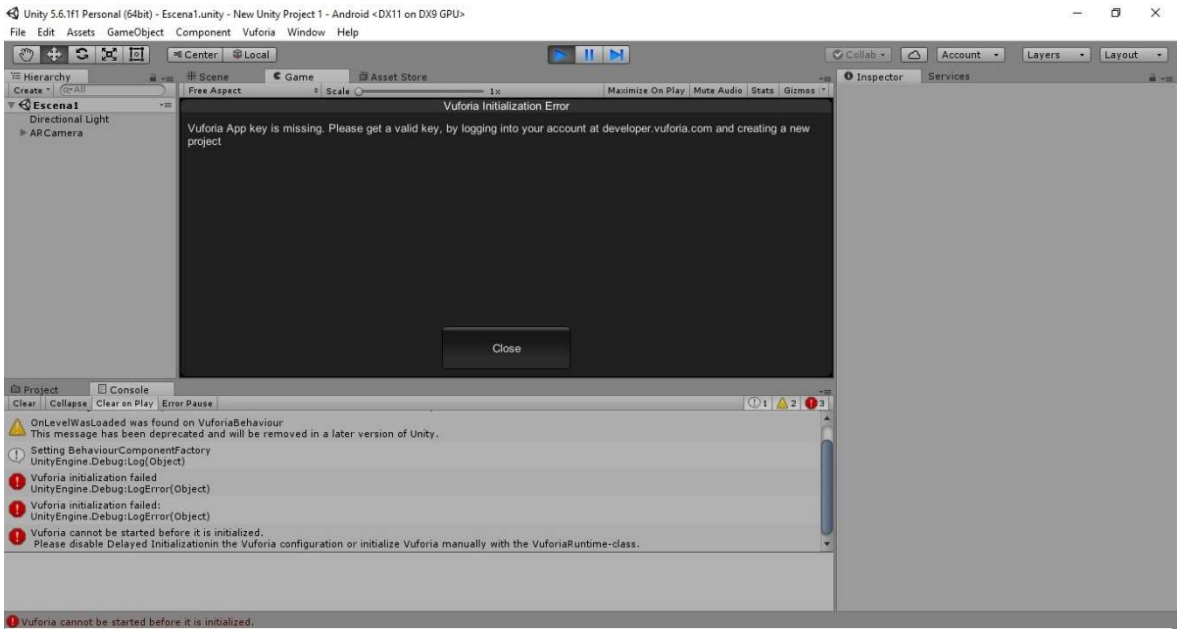


Figura 3.33. *Mensaje de error Vuforia-Unity.*

Para poder trabajar con Vuforia necesitamos crearnos una cuenta de usuario desde la página oficial, https://developer.vuforia.com/, en la parte superior de la ventana nos aparecen las opciones 'Log In' (en caso de tener cuenta acceder aquí) y "*Register*", en la cual nos pedirán nuestro nombre, apellido, compañía, nuestro país, el email donde queremos tener la cuenta y una contraseña para poder acceder a la cuenta. Una vez creada nuestra cuenta, para quitar el mensaje de error que nos muestra Unity, debemos añadir una clave de licencia. Para ello desde la página oficial antes mencionada, buscamos la pestaña "*Develop"*, dentro de ella tenemos dos opciones, License Manager y Target Manager. En este momento nos centraremos sólo en License Manager, para añadir la clave de licencia pulsamos el botón llamado "*Add License Key*". A continuación, debemos escoger que tipo de proyecto estamos realizando, en nuestro caso es la opción de Development, siendo esta opción solo para desarrollo; hay que tener en cuenta que las otras opciones, "*Consumer*" o "*Enterprise*", no son gratuitas, tal y como muestra la siguiente figura.

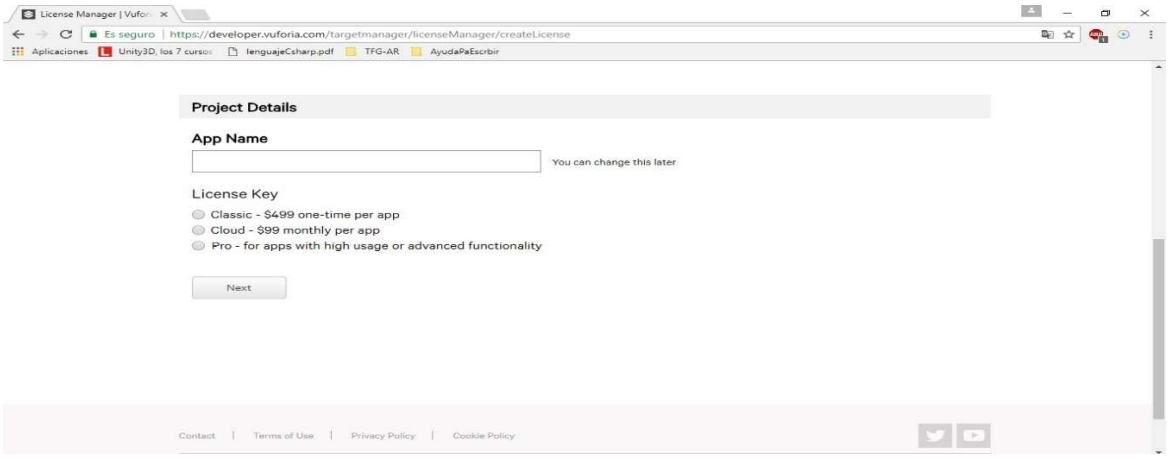


Figura 3.34. *Precio de licencia "Consumer" para marketing y juegos.*

Una vez seleccionada la opción de Development, nos aparecerá en la misma página los detalles de nuestro proyecto, aquí debemos ponerle un nombre a la licencia y pulsamos Next, aceptamos los términos y condiciones de Vuforia y confirmamos. Tras crear la licencia, nos aparecerá en una lista, donde tenemos todas nuestras licencias, seleccionamos la licencia creada y se nos abrirá una nueva ventana, en ella está la clave de licencia.

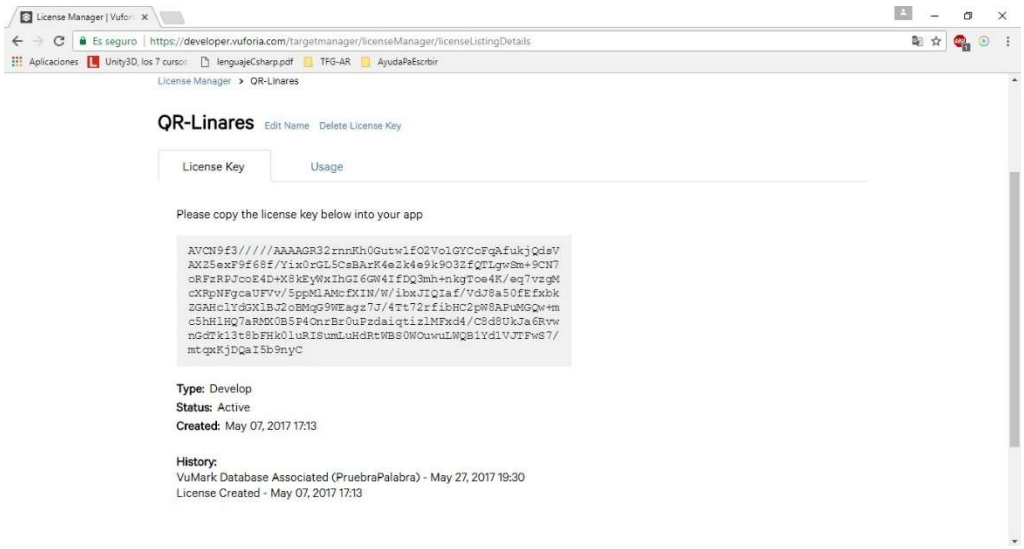


Figura 3.36. *Añadir clave de licencia.*

Una vez añadida la clave de licencia, ya tenemos enlazado Vuforia con Unity y podemos trabajar con la realidad aumentada. Este proceso sólo lo tenemos que hacer una vez, ya que esta clave de licencia se guardará para todo nuestro proyecto, por tanto, podemos añadir el Prefab de AR Camera a otras escenas sin necesidad de volver a repetir este proceso.

3.2.4.4 Contenidos utilizados de Vuforia

En este apartado haremos mención de las funciones que realiza nuestra aplicación. Estas funciones pueden ser que vengan por defecto en las librerías de Vuforia, o que se tengan que descargar un nuevo paquete para añadirlo en estas librerías.

3.2.4.4.1 *Marcadores*

Otras de las funciones que vamos a utilizar en nuestra aplicación es el reconocimiento de marcadores y texto. Estos marcadores son los denominados targets. Los Targets son utilizados por el rastreador (Tracker: procesa las imágenes captadas por la cámara) para reconocer un objeto del mundo real; los targets pueden ser de diferentes tipos, los principales son Image Targets (compuesta por fotos, páginas de revistas, cubiertas de libros, tarjetas, etc.) y Word Targets (elementos textuales que representen palabras simples o compuestas).

Para poder utilizar el Prefab de Image Target también debemos configurarlo desde la página oficial de Vuforia, ya que requiere de la creación de una base de datos que contiene las imágenes que vamos a emplear como marcadores.

Accedemos a la misma página donde hemos creado la clave de licencia, pero esta vez seleccionamos la pestaña de Target Manager. Tras acceder a ella tenemos que presionar el botón llamado "*Add Database*" para la creación de nuestra base de datos. Una vez presionado nos aparecerá una nueva ventana.

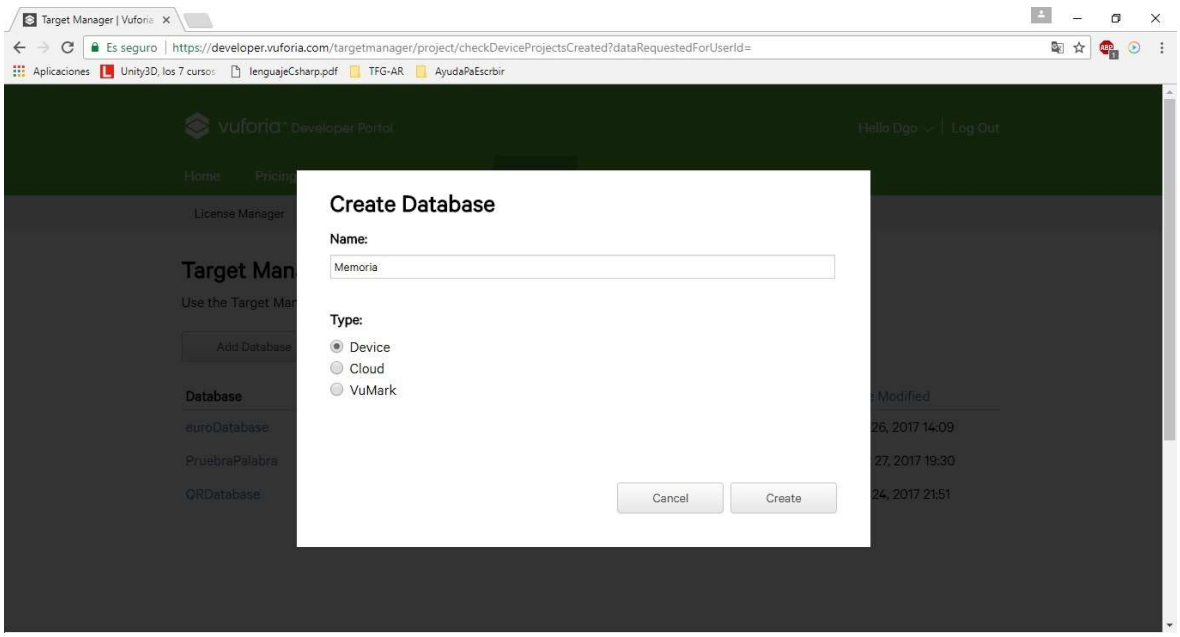


Figura 3.37. *Creación de la Base de Datos.*

En esta ventana debemos poner un nombre a nuestra base de datos y escoger la opción de Device, tal y como se muestra en la imagen superior. A continuación, presionamos Create. Tras crear la base de datos nos aparecerá un listado con todas las que hayamos creado. Ahora debemos introducir nuestras imágenes en la base de datos, para ello seleccionamos en este listado la base de datos creada, para acceder a ella. Al ser nueva, esta aparecerá vacía, para añadir las imágenes que queremos como marcadores debemos presionar el botón que aparece llamado "*Add Target*".

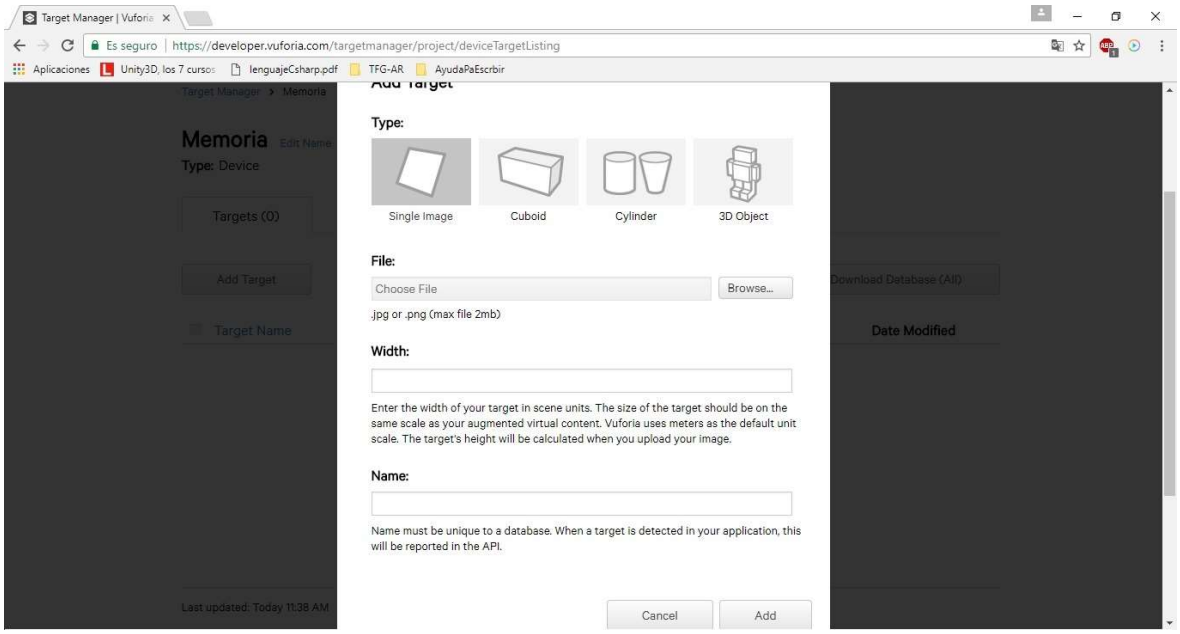


Figura 3.38. *Añadiendo imagen a nuestra Base de Datos.*

Tal y como muestra la figura, debemos seleccionar el tipo de imagen como "*Single Image"*, buscamos la imagen que vamos a poner como marcador en el explorador de archivos de nuestro PC dándole a "*Browse…*", el siguiente paso será el de ponerle un tamaño a nuestro marcador, en nuestro caso hemos puesto a todos los marcadores un tamaño de 5, para que sea más fácil de detectar por la cámara. El último paso es ponerle un nombre a nuestro marcador. En nuestra aplicación hemos creado ocho marcadores, cada uno correspondiente a una tienda del Pasaje del Comercio de Linares.

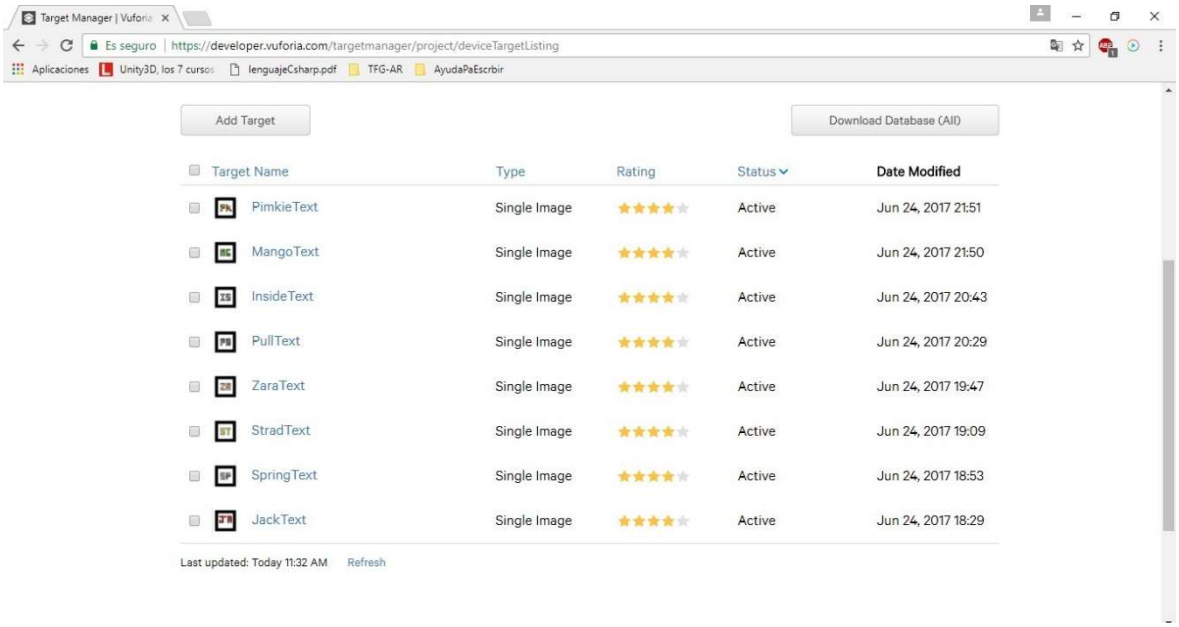


Figura 3.39. *Contenido de la Base de Datos.*

Como podemos observar en la figura, cada marcador cuenta con una clasificación, llamado "*Rating*", este cuenta con una valoración de un máximo de 5 estrellas. Vuforia nos dará esta valoración dependiendo de cuantos puntos de referencia tiene nuestra imagen. Estos los podemos observar accediendo a uno de los marcadores y presionando "*Show Features*" que se encuentra debajo de la imagen, obteniendo las referencias que toma Vuforia para reconocer el marcador.

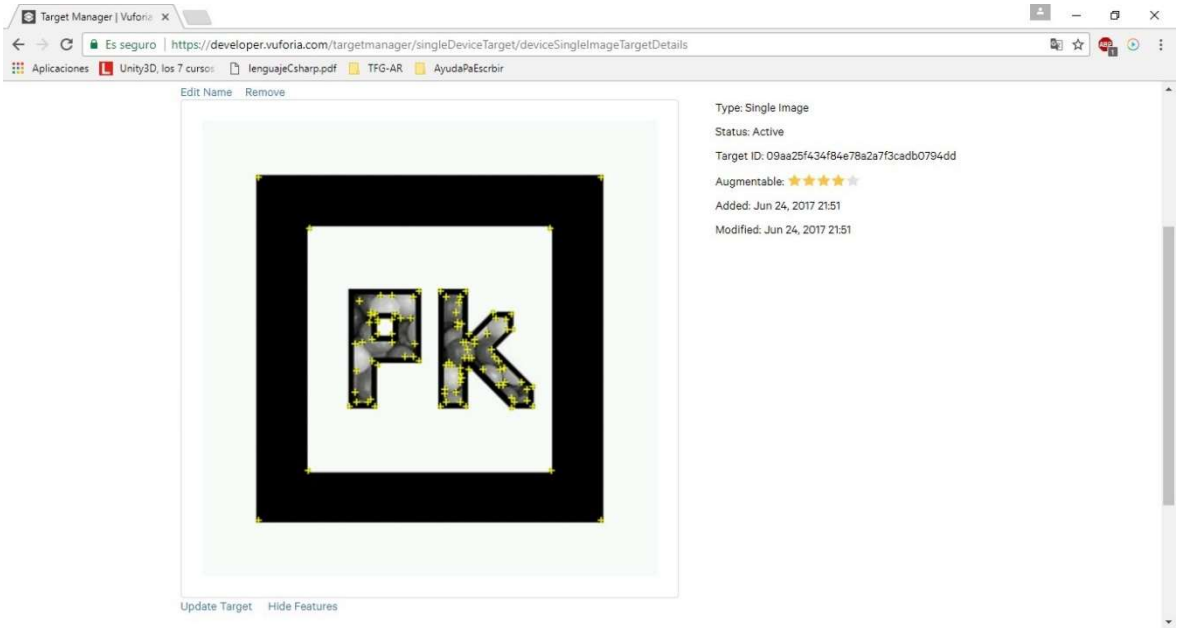


Figura 3.40. *Puntos de referencia de un marcador.*

Según especifica la documentación de Vuforia, debemos tener una valoración de como mínimo 3 estrellas, en caso de que sea inferior, significa que no tiene muchos puntos de referencia y puede producirse un error, ya sea que no detecte el marcador o que lo confunda con otro. En nuestro caso hemos conseguido obtener una valoración de 4 estrellas.

Una vez estemos satisfechos con la valoración de nuestros marcadores, debemos descargar la base de datos, presionando el botón que aparece en la parte superior derecha del listado de imágenes, llamado "*Download Database (All)*". Al presionarlo se nos abrirá una pequeña ventana donde podremos elegir la plataforma de desarrollo, en nuestro caso escogemos la opción de 'Unity Editor' y la descargamos.

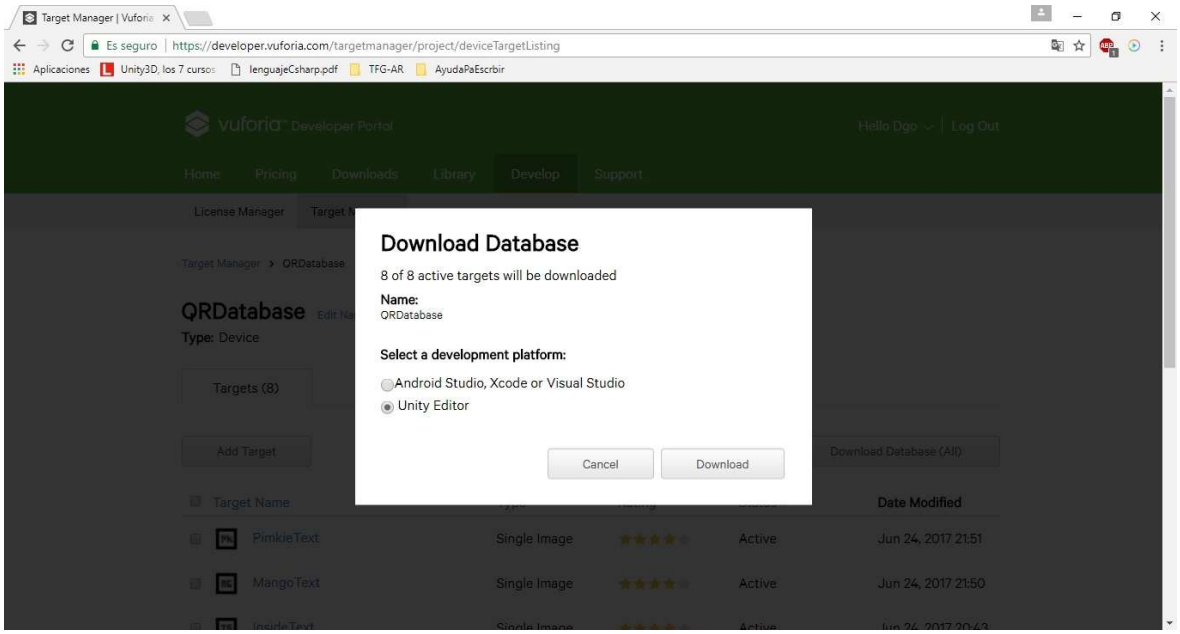


Figura 3.41. *Descarga de la Base de Datos*.

Esta base de datos tiene la extensión del tipo "*unitypackage"*, para añadirla a nuestra aplicación seguimos el mismo procedimiento que para añadir Vuforia a nuestro proyecto. Desde la interfaz de Unity, presionamos Assets en la barra de menú, buscamos Import Package y dentro de este seleccionamos Custom Package, se nos volverá a abrir el explorador de archivos y buscamos la base de datos descargada. Nos aparecerá la barra de progreso y la ventana con el contenido de la base de datos. Seleccionamos todas las opciones y presionamos importar. Tras haberse cargado correctamente, es hora de enlazarlo con Unity para ello nos situamos en la ventana del inspector de AR Camera, en la misma ventana donde copiamos nuestra clave de licencia, aquí encontraremos la pestaña Datasets, la desplegamos y aparecerá la opción de marcar nuestra base de datos,

una vez marcada aparecerá justo debajo la opción para activarla que también debemos marcar.

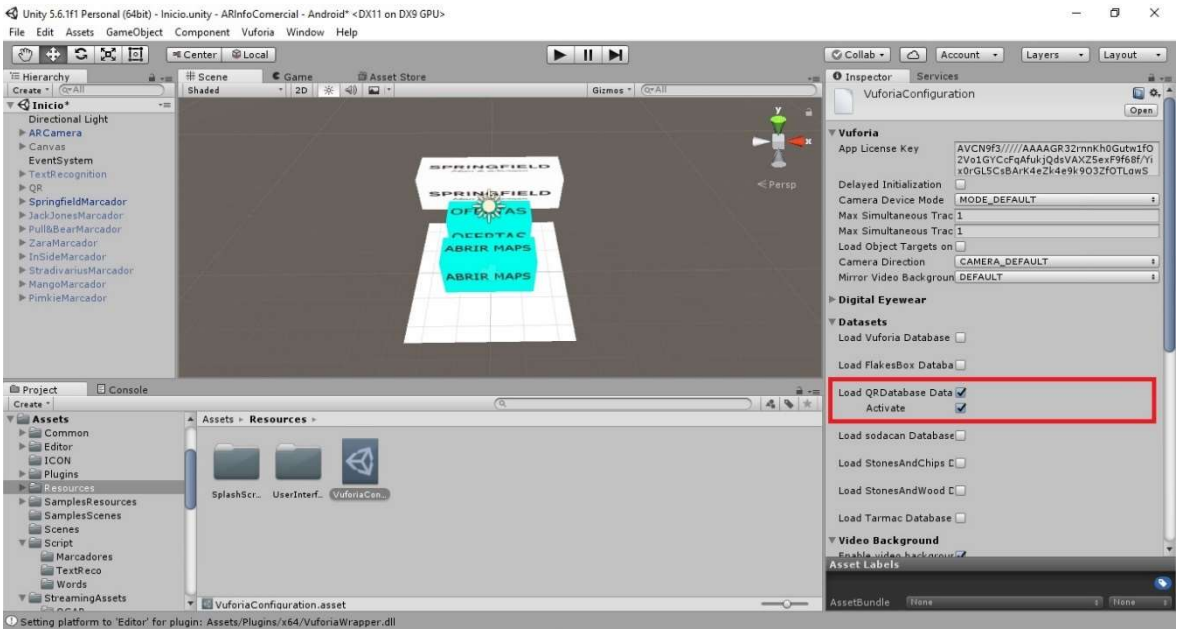


Figura 3.42. *Cargando la Base de Datos en Unity.*

Para seleccionar cada imagen que se encuentre dentro de nuestra base de datos, debemos seleccionar el ImageTarget, en la ventana del inspector nos aparecerán sus propiedades, tan solo debemos centrarnos en el Script llamado "*Image Target Behaviour*".

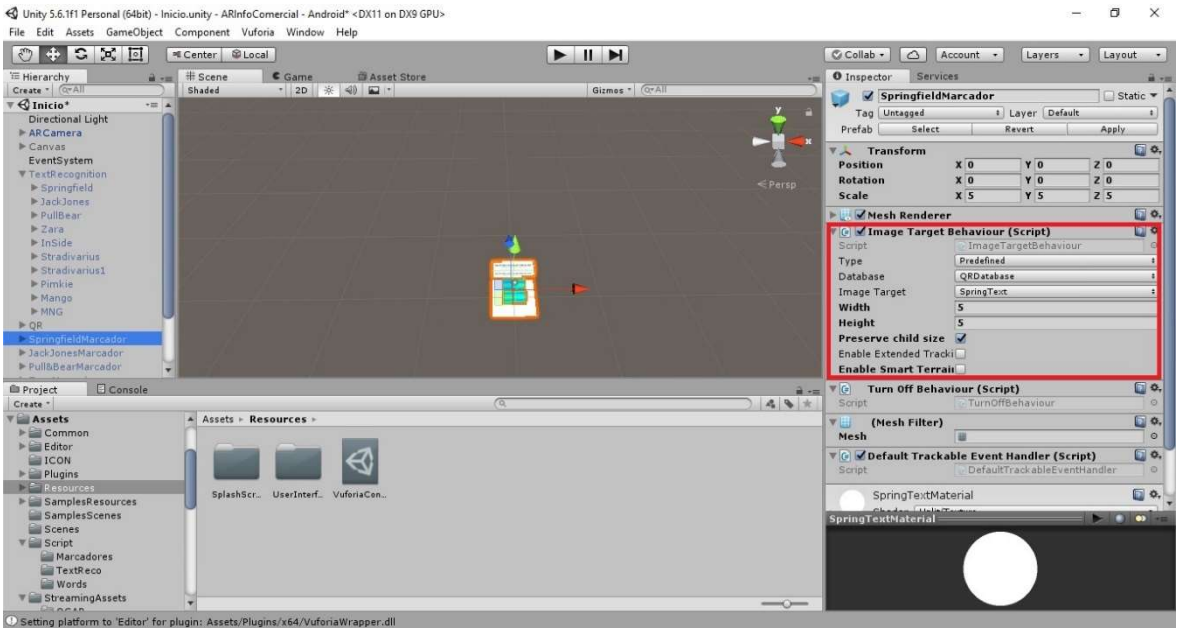


Figura 3.43. *Selección de marcador.*

Como podemos observar, en la opción de "*Database*" se encuentra una pestaña desplegable, en ella debemos buscar nuestra base de datos y seleccionarla. Una vez que

la tengamos seleccionada, en la opción de Image Target podremos escoger el marcador que deseemos de nuestra base de datos, tenemos que asegurarnos de poner el mismo tamaño que le pusimos a la hora de añadirlo en la base de datos, aunque generalmente aparece por defecto.

3.2.4.4.2 *Reconocimiento de texto*

Otra de las funciones que nos proporciona Vuforia es la de reconocimiento de texto, esta se encuentra en la carpeta de Prefabs y se llama TextRecognition. Este prefab requiere de una descargar de un paquete que contiene más de cien mil palabras en inglés, a través de la página oficial, accediendo a la pestaña de Downloads y dentro de esta en la pestaña Samples. Desde aquí debemos buscar el archivo llamado Vuforia-samples-core-unity-6-2-10.zip, y lo descargamos para Unity.

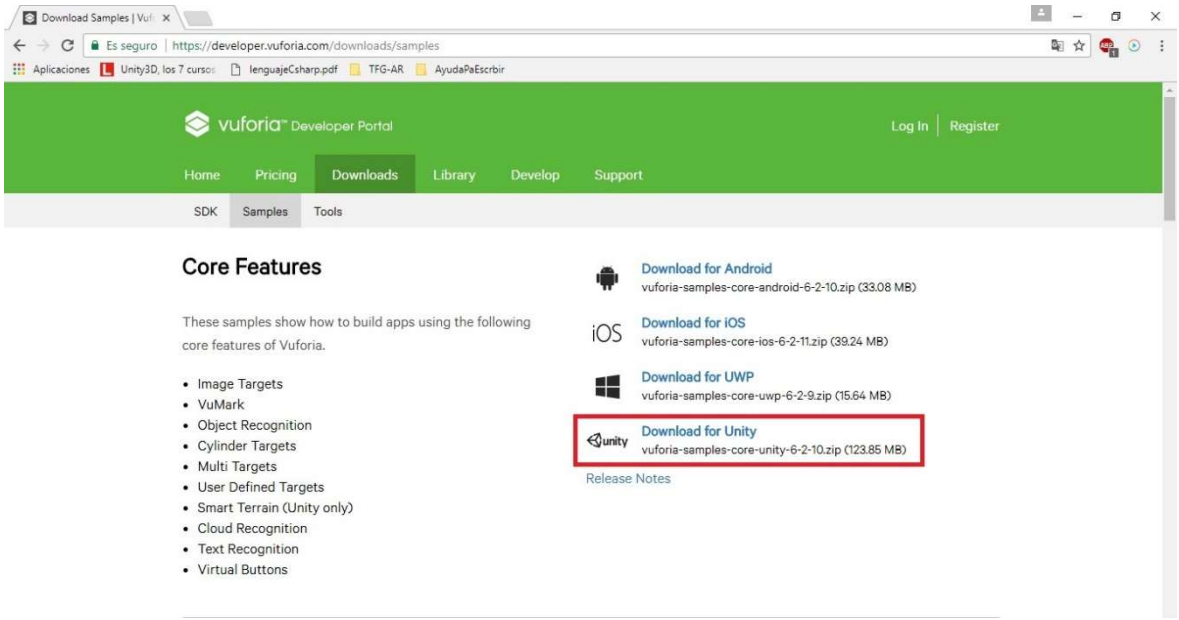


Figura 3.44. *Descarga el paquete de palabras.*

Este archivo comprimido contiene dos paquetes de Unity, el único que vamos a necesitar para obtener el listado de palabras es el llamado VuforiaSamples-6-2-10.unitypackage, para introducirlo a Unity seguimos el mismo proceso que realizamos anteriormente, en la barra de menú le damos a Assets, dentro de este seleccionamos Import Package, le damos a Custom Package y buscamos en el explorador de archivos el paquete de VuforiaSamples y le damos a importar.

Tras haber importado el paquete, ya podemos añadirlo a nuestro Text Recognition, para ello, seleccionamos el Prefab de Text Recognition en la ventana de jerarquía, por lo que en la ventana del inspector nos aparecen sus propiedades, en el script llamado Text Reco

Behaviour, podemos observar la opción de ‘Word List’ con una pestaña desplegable, aquí nos saldrá el listado, llamado Vuforia-English-word.

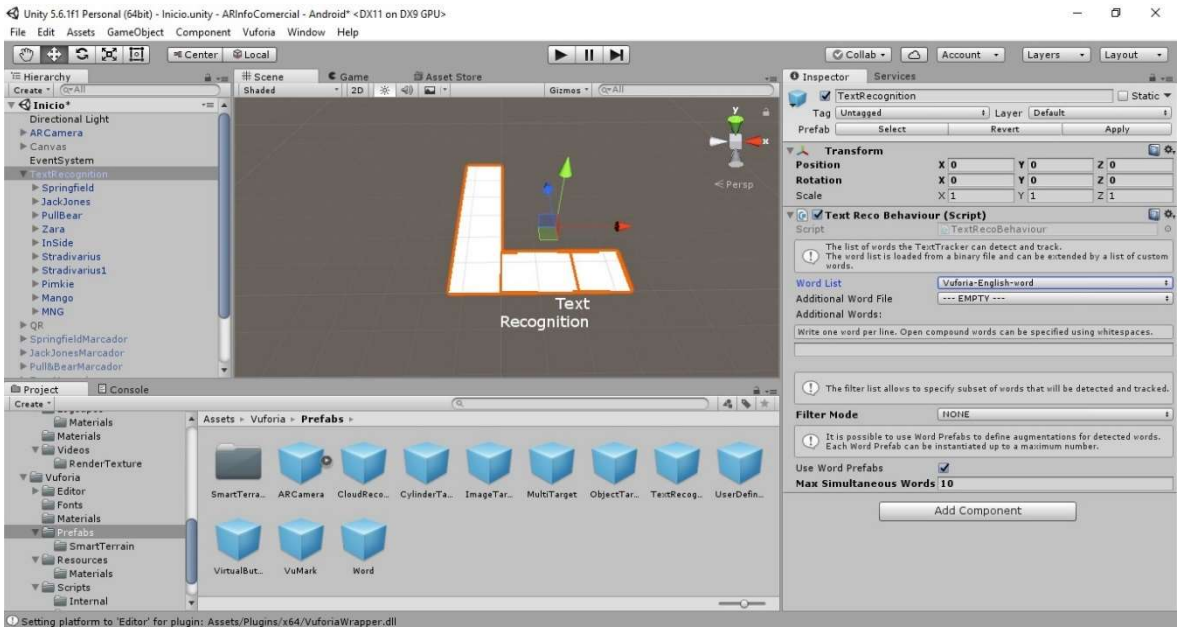


Figura 3.45. *Seleccionando el listado de palabras.*

Como podemos observar en la figura, debemos seleccionar la pestaña llamada “*Use Word Prefabs*” que aparece más abajo en el inspector, una vez seleccionado nos muestra el máximo de palabras simultaneas, aquí ponemos el número total de palabras que va a reconocer nuestra aplicación. En nuestro caso se han implementado 10 palabras, aunque tenemos 8 tiendas disponibles, el nombre de algunas tiendas no están implementadas en el listado de palabras lo que ha hecho necesario que estas tiendas tengan dos reconocimientos de texto con palabras compuestas o una palabra dentro del nombre de la tienda que Vuforia sí reconoce. En el Capítulo 4 se explica con detalle este inconveniente.

Dentro del Prefab Text Recognition vamos a añadir otro Prefab, llamado Word, este sirve para introducir la palabra que queremos que nuestra aplicación reconozca, insertaremos tantos Word como palabras contenga nuestra aplicación. Para introducir cada palabra tan solo debemos arrastrar este Prefab a la ventana de jerarquía, dentro de Text Recognition.

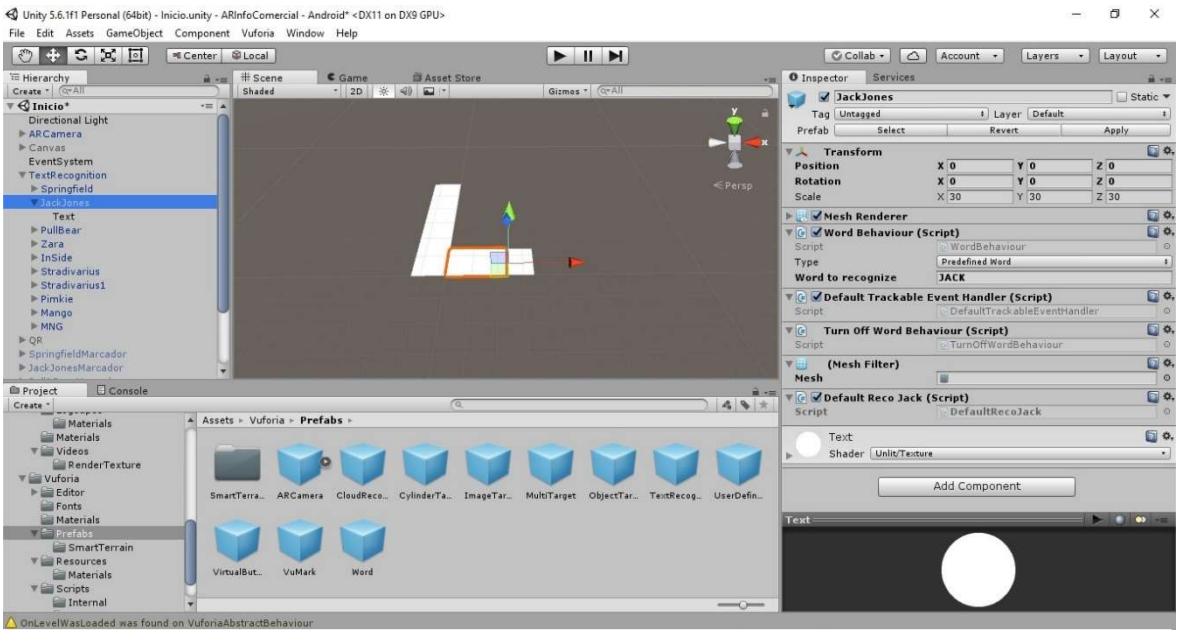


Figura 3.46 *Introducción de palabra para reconocer.*

Como podemos observar en la figura, en el script llamado Word Behaviour debemos de poner la palabra que queremos que reconozca, dejando el tipo como "*Predefined Word*". Este Prefab contiene en su interior un texto el cual debemos dejar en blanco, ya que cuando ponemos la palabra que queramos que reconozca también se escribirá sola en el apartado de Text.

3.2.4.4.3 *Reconocimiento de códigos QR*

Otra de las funciones de las que cuenta nuestra aplicación es la detección de códigos QR. El nivel de realidad aumentada es el más simple, ya que cada código QR está asociado a una función específica, puede ser una URL, un texto, E-mail, SMS, imágenes, etc. En nuestra aplicación está asociada a cada URL de las diferentes marcas comerciales.

Para introducir el lector de códigos QR en nuestro proyecto, hemos utilizado la herramienta llamada ZXing.Net, esta herramienta la podemos obtener desde su página oficial https://zxingnet.codeplex.com/, una vez que nos encontremos en esta página debemos buscar la pestaña de descarga "*Download*". Tras acceder a esta pestaña, nos aparece la descarga recomendada para nuestro PC. Seleccionamos esta descarga y comenzara a descargarse un archivo comprimido, en nuestro caso se trata de la versión ZXing.Net.0.15.0.0. Dentro de este archivo debemos buscar la carpeta de Unity.

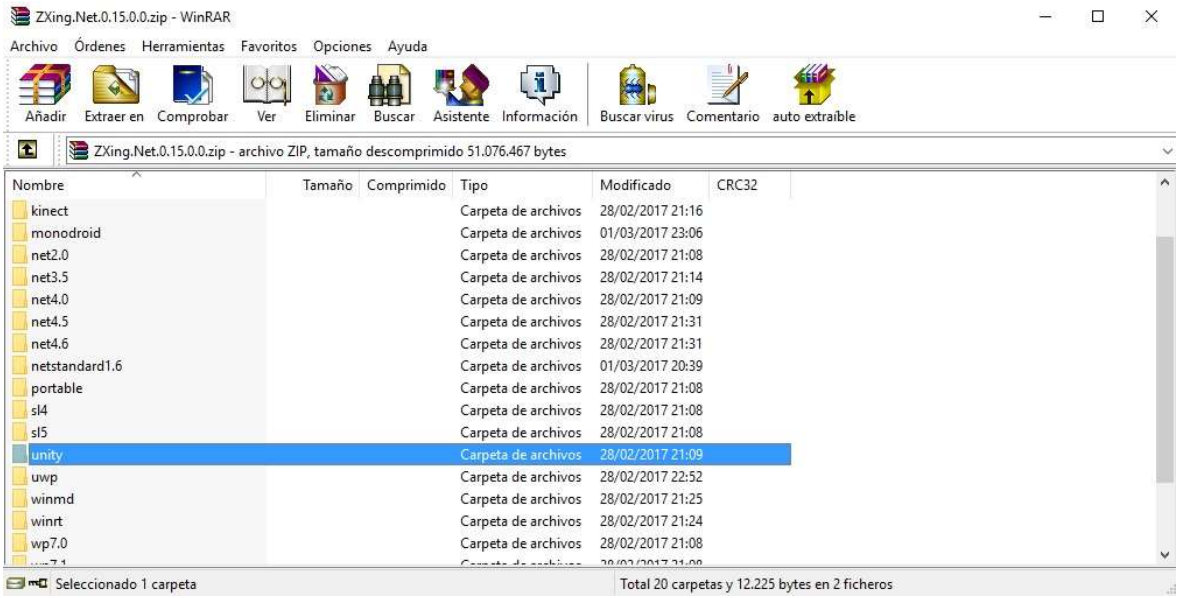


Figura 3.47. *Contenido archivo comprimido.*

Extraeremos el contenido de esta carpeta, que contiene un archivo con la extensión .dll (biblioteca de enlace dinámico), otro archivo con extensión .pdb (se utilizan en el lenguaje de programación C++, es un formato de base de datos) y otro archivo con extensión .xml (se trata de un metalenguaje, un lenguaje que se utiliza para decir algo acerca de otro). Para nuestra aplicación solo es necesario el archivo con extensión .dll, para implementarla, debemos arrastrar este archivo a la carpeta llamada plugins, quedando de la siguiente manera.

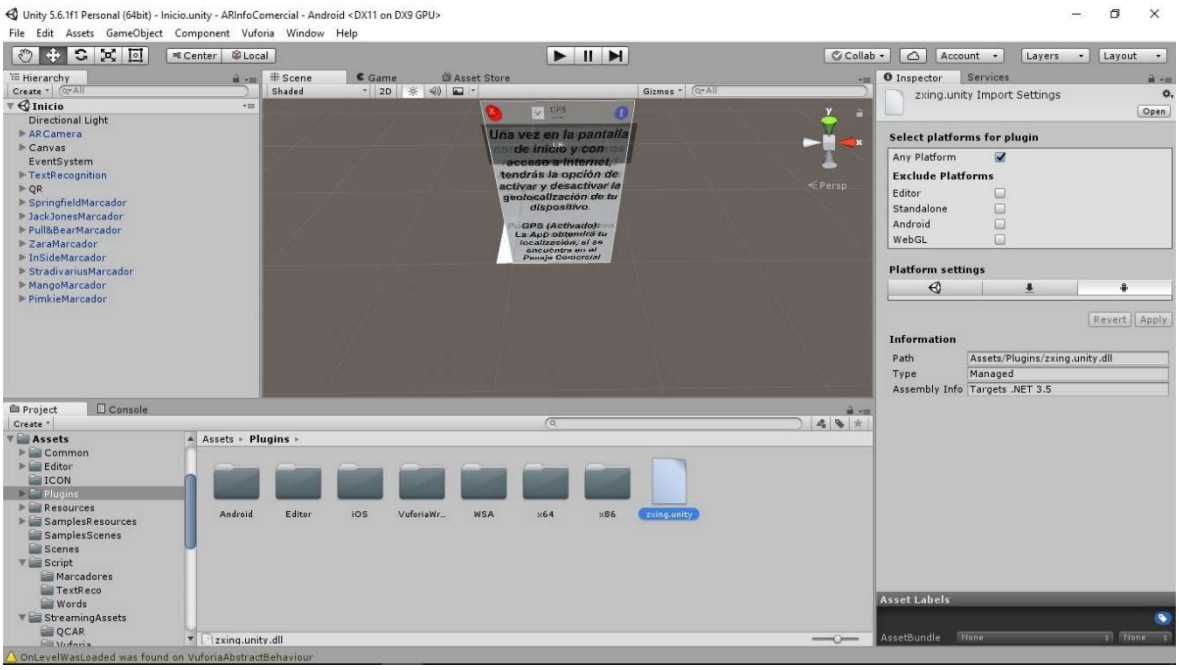


Figura 3.48. *Implementación de archivo en Unity.*

Ya hemos añadido todas las librerías y herramientas para hacer funcionar esta aplicación. En el siguiente apartado vamos a describir como se han realizado los menús, marcadores, creación de nuestro propio banner y otras funciones.

3.2.5 INTERFAZ DE CONTENIDOS DE REALIDAD AUMENTADA

En este apartado veremos cómo se ha realizado cada escena de la aplicación, ya que hemos utilizado diferentes técnicas para realizar un estudio más completo sobre Unity.

3.2.5.1 Contenido de la primera escena

Comenzamos con la escena que aparece nada más arrancar la aplicación. Esta escena la hemos llamado 'Texto Inicial', en ella aparece un texto de bienvenida a la aplicación y que hacer en caso de necesitar más información para sacarle todo el partido a nuestra aplicación. Este texto desaparecerá al pasar un cierto tiempo, este periodo de tiempo se ha creado a partir de un collider, estos definen la forma de un objeto para los propósitos de colisiones físicas. En nuestra aplicación emplearemos esta acción para hacer que se cargue la escena principal, llamada 'Inicio'. Para crear este collider tenemos que añadirle a la escena dos cubos, pulsando el botón de GameObject de la barra de menú, desplegamos la opción "*3D Object*" y elegimos el objeto 'Cube'. A este cubo lo llamamos CubeStop, que será un colisionador estático, esto quiere decir que nunca se mueven o cambian cuando el otro cubo colisiona, para darle la función de collider debemos configurarlo como Trigger (utilizando la propiedad Is Trigger, que nos aparece en la ventana del inspector), este no se comportará como un objeto sólido y por tanto le permitirá a otros colliders pasar a través de él. Cuando un collider entra en su espacio, un trigger va a llamar a la función OnTriggerEnter en los scripts del trigger objeto. Por tanto, añadiremos un script a CubeStop con esta función. Una vez tengamos este cubo configurado, vamos a duplicarlo presionando las teclas Ctrl + D, a este nuevo cubo lo llamaremos CubeStart, a diferencia del otro cubo, este no lleva la propiedad "*Is Trigger*", por lo que debemos quitarla y también quitaremos el script con la función OnTriggerEnter. Para poder identificarlo desde el script del otro cubo, le añadiremos un nuevo Tag, desde la ventana del inspector, desplegamos la pestaña de Tag y presionamos en "*Add Tag…"*, en nuestro caso le hemos puesto el nombre de "*TextInicio*", tras crearlo, lo seleccionamos para nuestro CubeStart. Para permitir que este objeto tenga el comportamiento físico, como por ejemplo la gravedad, debemos añadirle un nuevo componente desde la ventana del inspector llamado

"*Rigidbody*", presionando el botón "*Add Component*" que aparece en la parte inferior de esta ventana. Una vez añadido le marcamos la pestaña de '*Use Gravity'*, con lo que este objeto caerá por el eje Y al iniciar la escena, cuanto más lo subamos por este eje, más tiempo tardará en colisionar. Por tanto, las posiciones de los cubos quedan de la siguiente manera: tanto el CubeStop como el CubeStart se colocan en la misma posición en el eje X y fuera de la posición del canvas, para que no se muestre en la aplicación, en nuestro caso X = -78, para el eje Y tenemos que colocar el CubeStop en la posición 0 y el CubeStart lo subimos hasta obtener el tiempo deseado, en nuestro caso tiene un valor de Y = 300.

Para la creación del panel en el cual aparece el texto debemos de crear un canvas, dando botón derecho del ratón en la ventana de jerarquía, en el menú de UI, o también presionando el botón de GameObject que aparece en la barra del menú. De esta forma se crearán la mayoría de objetos y menús de las que cuenta nuestra aplicación.

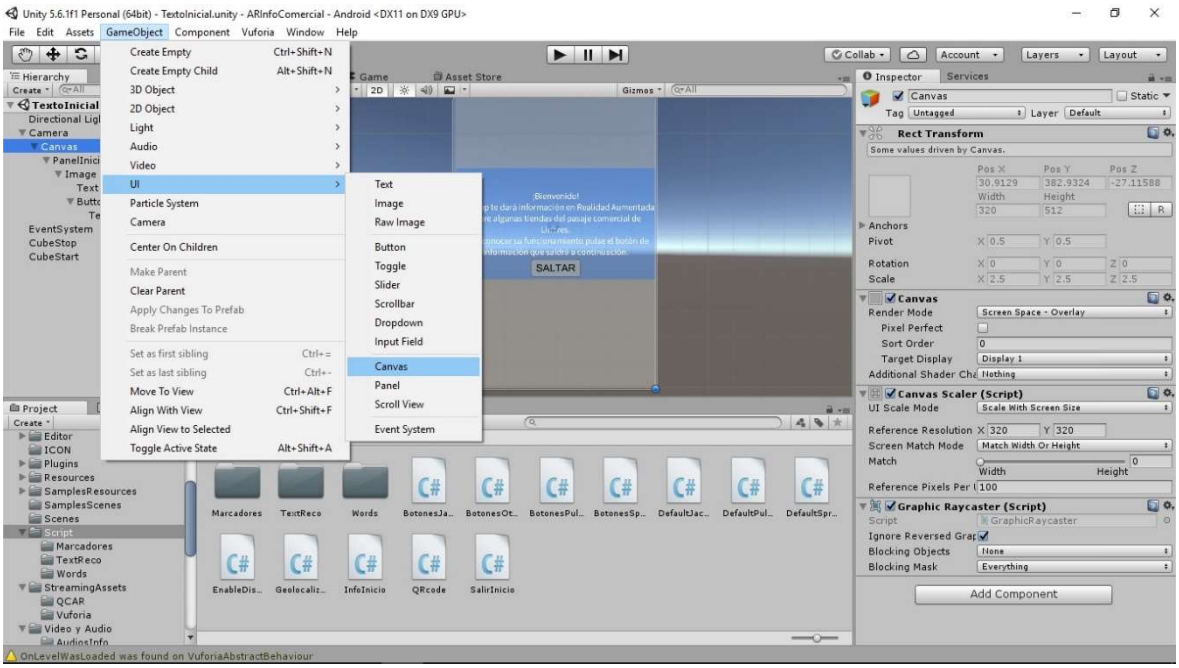


Figura 3.49. *Desplegando el menú UI.*

Este canvas tendrá las dimensiones que queramos según el dispositivo, en nuestro caso lo hemos ajustado al tamaño de la pantalla, con una referencia de resolución de 320 en el eje X y 320 en el eje Y, siendo la resolución que mejor se ajusta a los dos dispositivos de prueba. En el canvas no es necesario configurar nada más. Ahora dentro del canvas crearemos un panel, que se ajustará al tamaño del canvas, en este panel solo vamos a configurar la transparencia, presionando en 'Color' dentro del script 'Image' y arrastrando la última barra en la pequeña ventana que se nos abrirá.

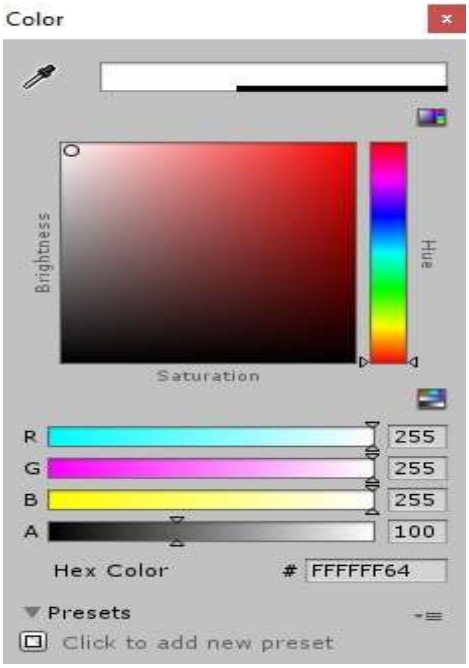


Figura 3.50. *Configuración de Color y Transparencia.*

Como podemos observar en la figura, hemos puesto el panel de color blanco y una transparencia de más de la mitad de la barra, por lo que nuestro fondo se verá del color de fondo que tenga nuestra Main Camera.

Dentro de este panel crearemos una imagen, a la cual le pondremos el tamaño que deseemos y le ajustaremos el color y la transparencia con el mismo procedimiento que el anterior. Con el ajuste preestablecido de anclaje la imagen quedaría siempre centrado en el panel que se está trabajando con el inconveniente de que se pueda mover cuando cambias de dispositivo, por lo que tenemos que establecerlo en el menú de "*Rect Transform*". Cada objeto que introduzcamos en la escena cuenta con estos anclajes, también podemos anclar el objeto manualmente moviendo unos triángulos que nos aparecen por defecto en el centro del canvas.

Dentro de la imagen creamos un "*Text*", también en el menú UI, en este objeto podremos escribir lo que necesitemos, cambiar la fuente de texto, alinearlo a nuestro gusto, cambiar el color del texto y ponerle el tamaño deseado. En este caso se ha dejado en color blanco, alineado en el centro y con un tamaño de fuente de 14. También se ha creado un botón con el cual tendremos la opción de saltar esta escena en caso de que ya la hayamos leído y no se haya acabado del tiempo del collider. Este botón se crea desde el menú UI, ajustamos el tamaño, posición, anclaje y color según queramos. Cada botón cuenta con una función en la ventana del inspector llamada "*On Click ()*", para realizar una acción cuando se pulse el botón. Para añadir esta opción debemos crear una variable pública en un script. Este script puede ir en un objeto vacío, o como en nuestro caso, añadido al script

de CubeStop. Tras crear la variable pública en el script, presionamos el botón '+' dentro de "*On Click ()*" para poder añadirle la variable, debemos arrastrar el objeto donde tengamos el script en la parte que pone "*None (Object)*" y desplegar la pestaña de "*Function*" para seleccionarla, como muestra la figura.

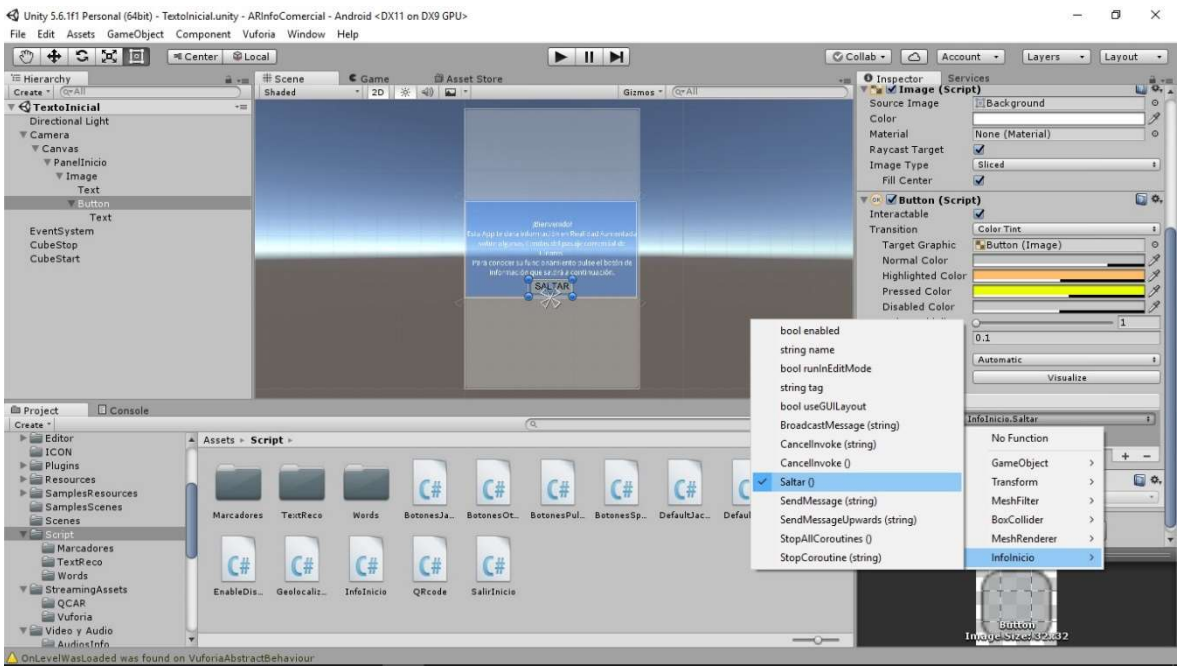


Figura 3.51. *Seleccionando la acción del botón.*

La función que tendrá este botón será la de cargar la escena 'Inicio' cuando se presione. Debemos asegurarnos de que añadimos la escena a nuestra aplicación, del mismo modo que cambiamos de plataforma, seleccionamos la opción File de la barra del menú, accedemos a Build Setting, una vez abierta esta ventana tenemos que introducir la escena a la aplicación, ya que si no lo hacemos nuestra aplicación no contará con esta escena. Hay que tener en cuenta que la escena que tengamos como número 0 será la primera que salte cuando arranque la aplicación, de modo que la escena 'TextoInicio' es la primera que saltará, como se muestra en la siguiente figura.

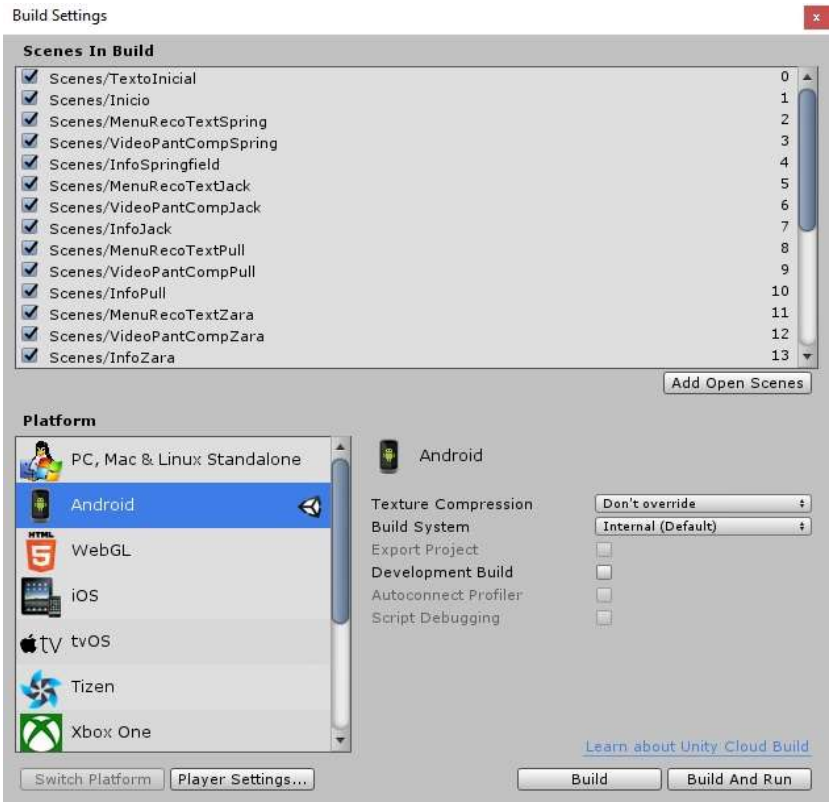


Figura 3.52. *Añadiendo escenas a nuestra aplicación.*

3.2.5.2 Contenido de la escena del panel principal

Seguimos con la siguiente escena, es la más importante de la aplicación ya que contiene todo el reconocimiento de marcadores, el reconocimiento de texto, la detección de los códigos QR, un menú de información y la geolocalización, se trata de la escena llamada 'Inicio'. Explicaremos cada función y panel con los que cuenta esta escena. Al terminar la escena de "*TextoInicio*" nos saltará automáticamente esta escena, activándose la cámara del dispositivo, también contará con un botón para cerrar la aplicación, otro para desactivar y activar el GPS y otro botón de información con el que activaremos unos paneles. A este panel lo llamaremos panel principal.

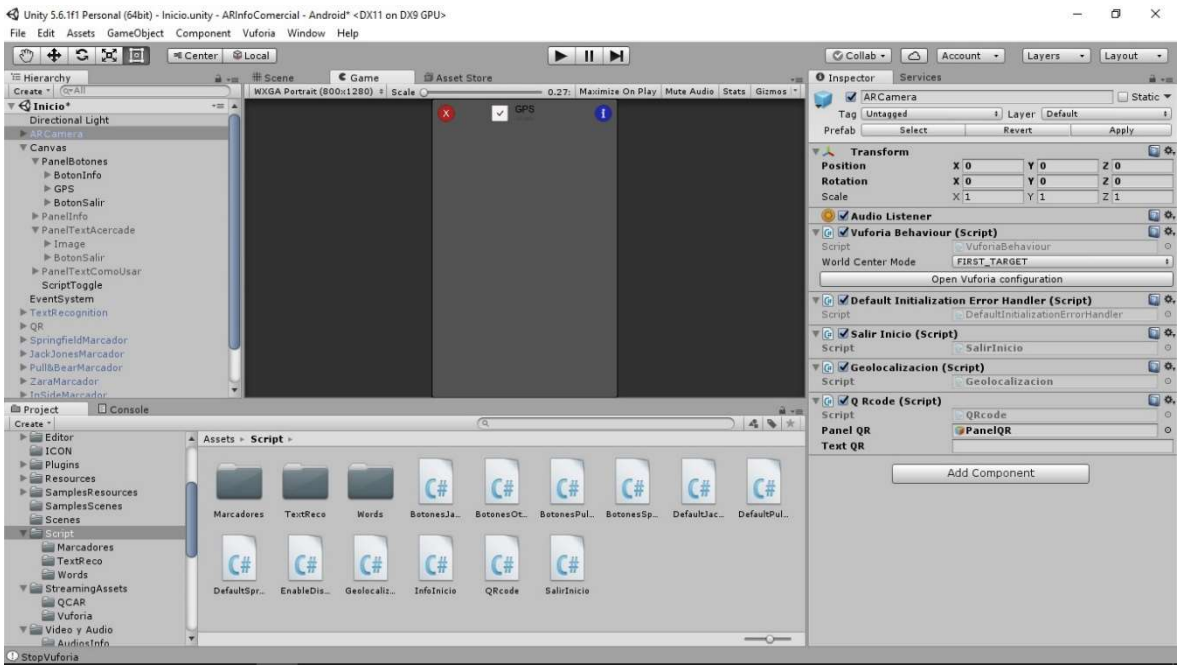


Figura 3.53. *Panel principal de escena de inicio.*

Como podemos observar en la figura, dentro de AR Camera tenemos implementado unos scripts, uno para el botón de salir, otro para utilizar el GPS del dispositivo y otro para la detección de los códigos QR. Los botones de salir y de información tienen una fuente de imagen llamada Knob, que hace que estos botones tengan una forma circular, seleccionamos esta fuente de imagen en la ventana del inspector. Para la 'i' de información se ha descargado desde el Asset Store un Asset que contiene una variedad de iconos.

Cuando pulsemos el botón de información, se nos activará un panel que cuenta con dos botones centrales los cuales se llaman "*Acerca de…"* y "*Como usar"*, además de un botón para volver al panel principal.

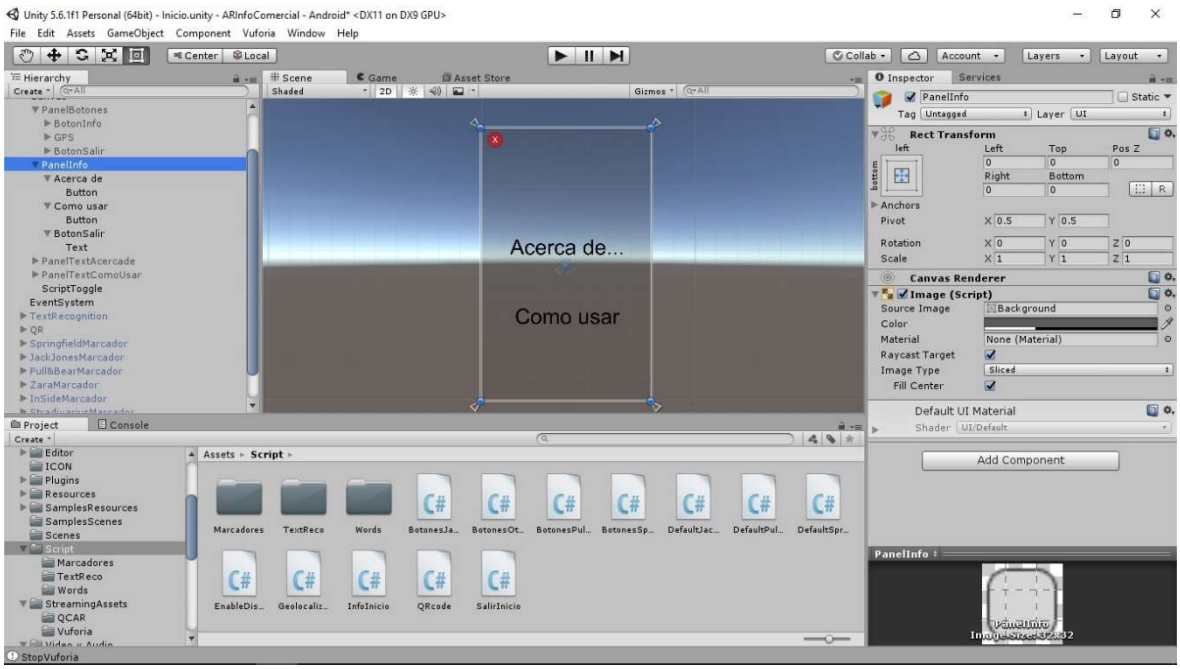


Figura 3.54. *Panel de información.*

Como podemos observar en la ventana de jerarquía, los textos de que nos aparecen tienen un botón, en este caso transparente, para poder pulsar sobre ellos. Cuando pulsamos uno de estos, se desactiva este panel y se activa otro explicando en qué consiste la aplicación o si pulsamos el otro nos dará información de cómo usar esta aplicación. Estos textos explicativos cuentan con sus propios paneles, de esta forma podemos activarlos y desactivarlo con unas pocas líneas de código en los scripts. Estos dos paneles cuentan con los mismos componentes, pero difieren en el texto explicativo.

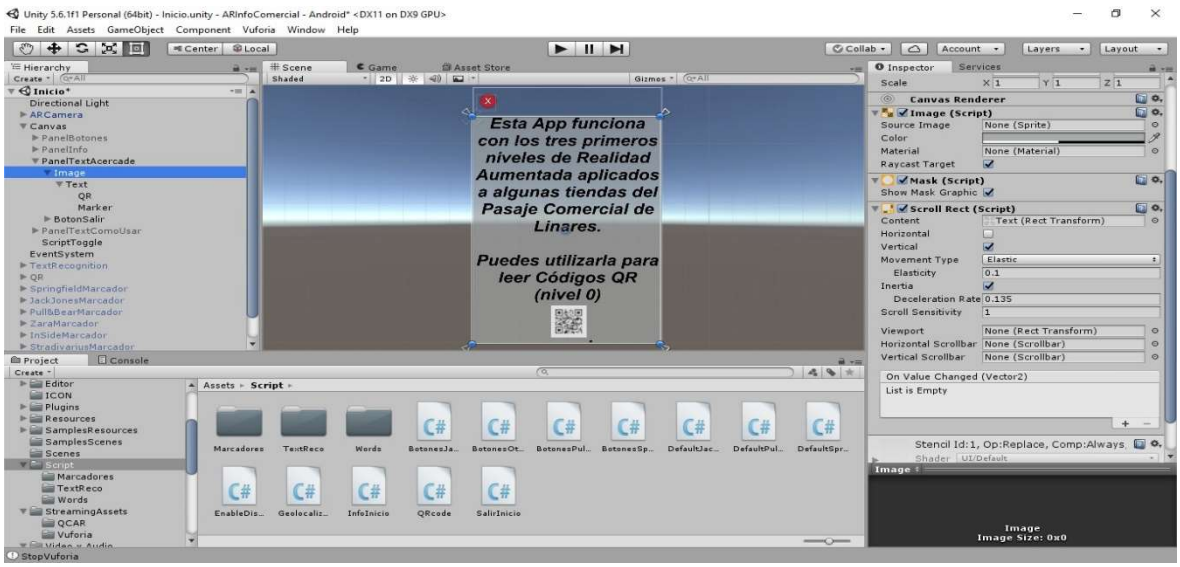


Figura 3.55. *Panel Acerca de…*

En la ventana de jerarquía podemos observar el contenido de este panel. Dentro del panel creamos una imagen, del menú de UI, le ajustamos el tamaño deseado y le añadimos los componentes 'Mask' y 'Scroll Rect' dándole en la ventana del inspector a 'Add Component' y buscándolos. El componente 'Mask' sirve para modificar la apariencia de los elementos hijos, en este caso el GameObject 'Text', por lo que, si el hijo es más grande que el padre (Image) entonces solamente la parte que encaja dentro del padre será visible. El componente 'Scroll Rect' proporciona la funcionalidad de desplazarse sobre este contenido. Cuando se combinan estos dos componentes se crea una vista de desplazamiento, donde solamente el contenido desplazable dentro del 'Scroll Rect' es visible, por tanto, debemos añadirle el texto, que es hijo del objeto Image, arrastrándolo desde la ventana de jerarquía a la ventana del inspector para poder desplazar el texto eligiendo la opción de desplazarlo en horizontal o en vertical. Al objeto 'Text' le añadimos un componente llamado 'Content Size Fitter', que es un ajustador del tamaño del contenido. A este componente le dejamos la opción de 'Horizontal Fit' como 'Unconstrained', y la opción de 'Vertical Fit' como 'Preferred Size', ya que solo nos interesa como el ancho es controlado. Una vez tengamos esto ajustado podremos desplazar el texto en vertical y así mantener un tamaño de fuente cómodo para ser leído. Para añadir una imagen como la que se muestra en la figura anterior, tan solo debemos añadir un GameObject del menú de UI llamado 'Raw Image', a este objeto le arrastramos la imagen con extensión .JPG en la ventana del inspector dentro del apartado de 'Raw Image' donde aparece la opción de 'Texture', este es el objeto que utilizaremos para añadir imágenes a nuestra aplicación.

Dentro de este canvas podemos observar un objeto llamado ScriptToggle, se trata de un objeto vacío en el cual añadiremos el script que controla todos estos paneles y el botón de activado y desactivado del GPS. Este objeto vacío se crea presionando el botón derecho del ratón en la ventana de jerarquía y seleccionando 'Create Empty'. El GameObject para la activación del GPS se llama Toggle, dentro del menú UI, este objeto controla el script de 'Geolocalización', que se encuentra en AR Camera activándolo y desactivándolo dentro de este Prefab.

Otra de las funciones con las que cuenta esta escena es la de reconocimiento de texto explicado anteriormente, la función que realiza cuando reconoce un texto es hacer saltar la escena con la que está relacionada, si por ejemplo reconocemos la palabra 'Springfield' la aplicación saltará a la escena correspondiente de la tienda, que cuenta con un menú que explicaremos más adelante.

Para la detección de los códigos QR hemos creado el siguiente panel, en el cual aparece el texto de la dirección URL de la que corresponde cada código QR, presionando donde

nos aparece el texto, la aplicación abre el navegador del dispositivo con esa dirección de URL, de esta forma accedemos a las paginas oficiales de las tiendas. Para crear esta interfaz, creamos un nuevo canvas, dentro de este añadimos un panel, en el interior del panel le añadimos una imagen para ponerle un fondo de color y dentro de la imagen un texto del mismo tamaño que la imagen, a este texto le hemos añadido un nuevo Tag, llamado URL, para que el script de reconocimiento de códigos QR encuentre este texto indicándoselo en el script. También cuenta con un botón para cerrar este panel en caso de que no queramos acceder a la página oficial de la tienda. El script que realiza las funciones de detección de códigos QR, llamado 'QR Code', se encuentra dentro de AR Camera. Para generar los códigos QR hemos utilizado la página online llamada QR-Code-Generator, con el siguiente enlace podremos acceder a esta página, 'http://es.qr-code-generator.com/', donde podremos seleccionar la función que queremos que cumpla nuestros códigos, en nuestro caso se trata de direcciones URL.

La siguiente función que realiza esta escena es la de reconocimiento de marcadores. La función que cumplirán estos marcadores en la aplicación es la de situarse en algún sitio de interés de Linares, ya sea la estación de autobuses o en otro lugar que sea transitado por turistas, ya que estos marcadores cuentan con la función de abrir la aplicación de Maps, que traen por defecto los dispositivos móviles y tabletas, al abrirse nos muestra la ruta que debemos seguir para llegar hasta el pasaje del comercio de Linares.

Estos marcadores los hemos creado con el programa gratuito llamado GIMP 2 y con la ayuda del Paint que viene por defecto en el PC. Desde Paint hemos creado una plantilla, de este modo hemos generado el borde del marcador y las letras.

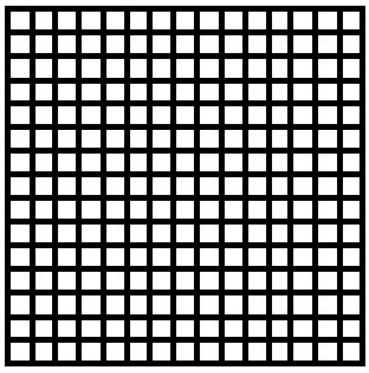


Figura 3.56. *Plantilla marcadores.*

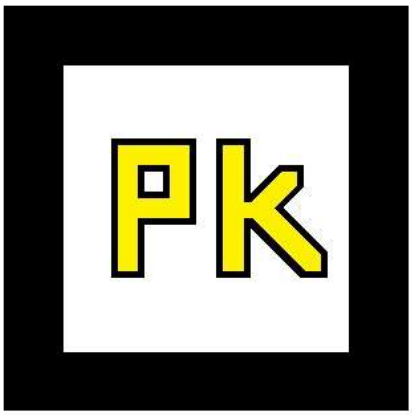


Figura 3.57. *Creación de borde y letras.*

Ahora es turno del programa GIMP 2, el cual nos permite añadirle capas a la imagen para poner otra imagen de fondo, en nuestro caso este fondo se ha implementado en las letras. A cada marcado se le ha asignado un fondo diferente.

Figura 3.58. *Marcador de la tienda Pimkie.*

Para estos marcadores se han empleado objetos 3D que podemos encontrar en la barra de menú pulsando en GameObject y eligiendo alguno de los objetos 3D que nos

proporciona Unity. Para nuestra aplicación se han usado los cubos, crearemos tres cubos eligiendo su tamaño y posición, uno para el logotipo de la tienda y otros dos que actuarán como botones. Para añadirles estas imágenes no basta con arrastrar la imagen con extensión .JPG, debemos crear un nuevo material. Este material se crea pulsando botón derecho del ratón en la ventana de proyecto, de ahí buscaremos la opción de 'Create' y posteriormente le daremos a material, tal y como muestra la figura

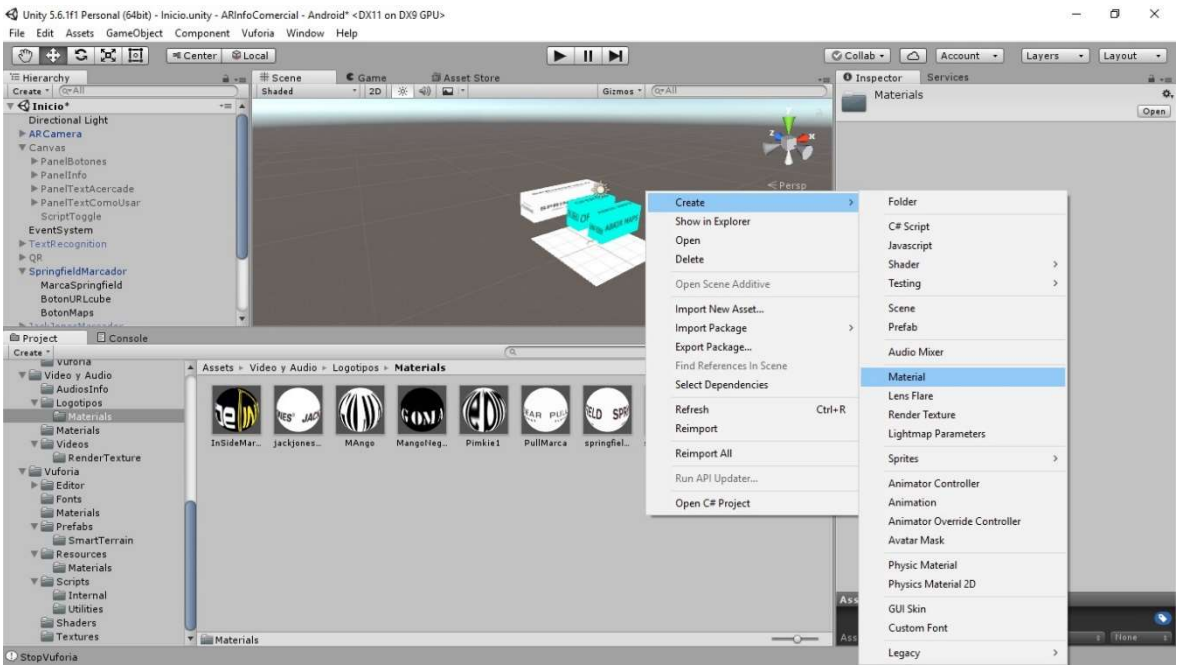


Figura 3.59. *Creando un nuevo material*.

Tras crearlo tendremos que elegir en la ventana del inspector que shader le pondremos a nuestro material, para poder añadirle una imagen .JPG debemos seleccionar el shader denominado Unlit/Texture. Una vez seleccionado arrastraremos la imagen .JPG al recuadro de textura.

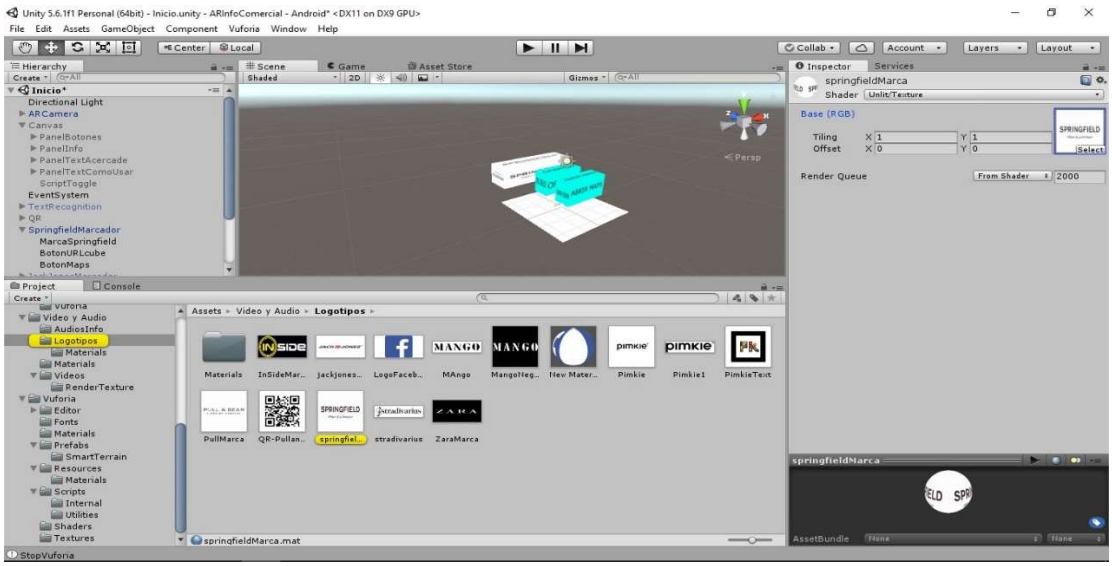


Figura 3.60. *Añadiendo imagen a nuestro material.*

Para los otros dos cubos seguimos el mismo proceso introduciendo la imagen que deseamos. Esto dos cubos cuentan con una función en su script que les hace actuar como botones, se trata de la variable 'OnMouseDown', que al presionar los cubos esta variable llama a otra con la cual realizaremos la función del botón. Cada uno de estos cubos debe tener su propio script, ya que la función 'OnMouseDown' solo permite que se presione un objeto. El botón 'Ofertas' te abre el navegador con la página oficial de la tienda, mientras que el botón 'Abrir Maps' te abre la aplicación de Maps con la ruta hacia la tienda. Estos cubos solo aparecen cuando la cámara está detectando el marcador, si se pierde de vista, los cubos desaparecen también.

Dado que contamos con ocho tiendas, se ha repetido este proceso para cada una de ellas, cambiando en los scripts la URL de cada tienda, las coordenadas para cada localización y el logotipo de cada tienda.

Una vez explicado cómo se ha realizado la escena de 'Inicio' vamos a explicar el proceso para realizar el menú donde nos aparece un listado con las tiendas de la aplicación, llamada 'OtrasTiendas', esta escena saltará en la aplicación cuando nos encontremos cerca del pasaje del comercio gracias a la geolocalización de nuestro dispositivo, ya que el script de geolocalización viene definido para estas coordenadas, aunque también podremos acceder a esta escena a través de los menús de cada tienda con el reconocimiento de texto. Para la creación de esta escena le añadimos el prefab de AR Camera (borrando la Main Camera que viene por defecto), también se le añadirá un canvas para crear el panel donde nos muestra las tiendas. Dentro de este panel le añadimos un botón para volver a la escena de 'Inicio', una imagen con un texto indicando el título de esta escena y una imagen con un texto y un botón para cada tienda.

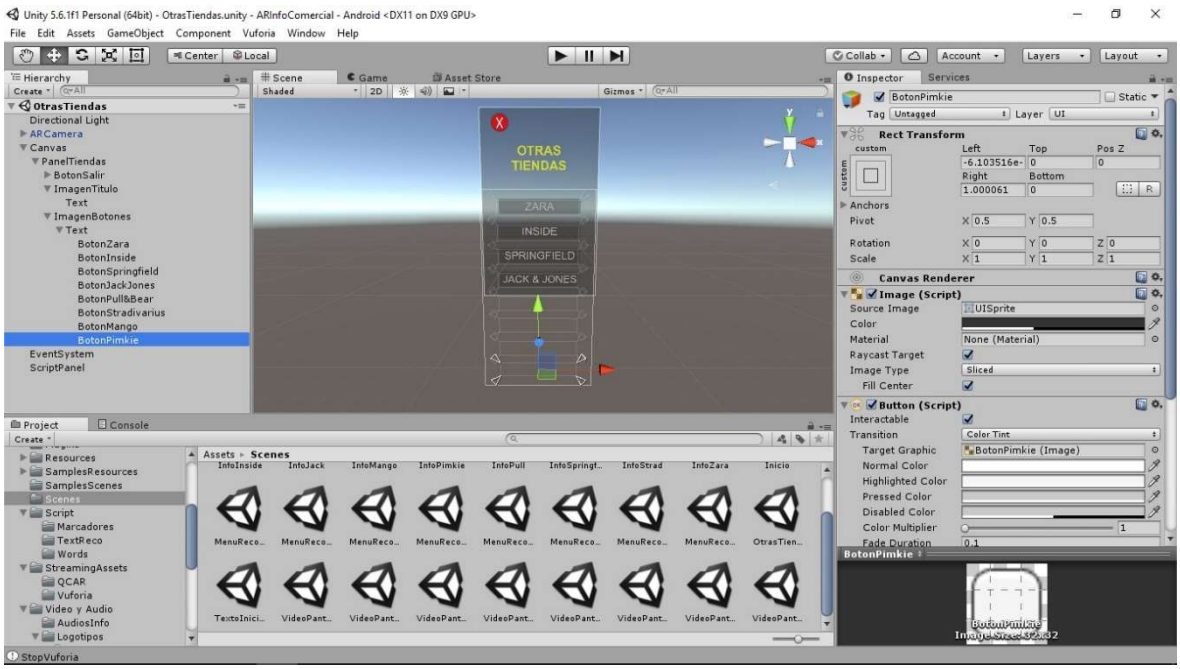


Figura 3.61. *Escena de Otras Tiendas.*

Como podemos observar, el texto que contiene los botones de las tiendas se ha realizado de la misma manera que el panel de información para hacer una vista de desplazamiento, añadiendo los componentes 'Mask' y 'Scroll Rect' en la imagen y el componente 'Content Size Fitter' en el texto, marcando las mismas opciones que en el panel de información. Para poner los nombres de las diferentes tiendas en el mismo objeto de texto hemos separado con dos espacios el nombre de cada tienda, en la opción donde se escribe el texto, para poder hacer la vista de desplazamiento, por tanto, estos botones serán ajustados a la posición de cada nombre de las tiendas. Cada botón hace que salte la escena correspondiente a cada tienda. La acción que cumple cada botón se encuentra en el script que está dentro del objeto vacío llamado ScriptPanel.

3.2.5.3 Contenido del menú de la tienda

Continuamos con la siguiente escena, que corresponde al menú de cada tienda, podemos acceder a este menú a través del reconocimiento de texto o mediante la escena de Otras Tiendas pulsando en los botones. Tenemos ocho escenas iguales, una para cada tienda, por tanto, explicaremos tan solo una de ellas. En este caso nos situaremos en la escena correspondiente a la tienda Jack & Jones, llamada 'MenuRecoTextJack'.

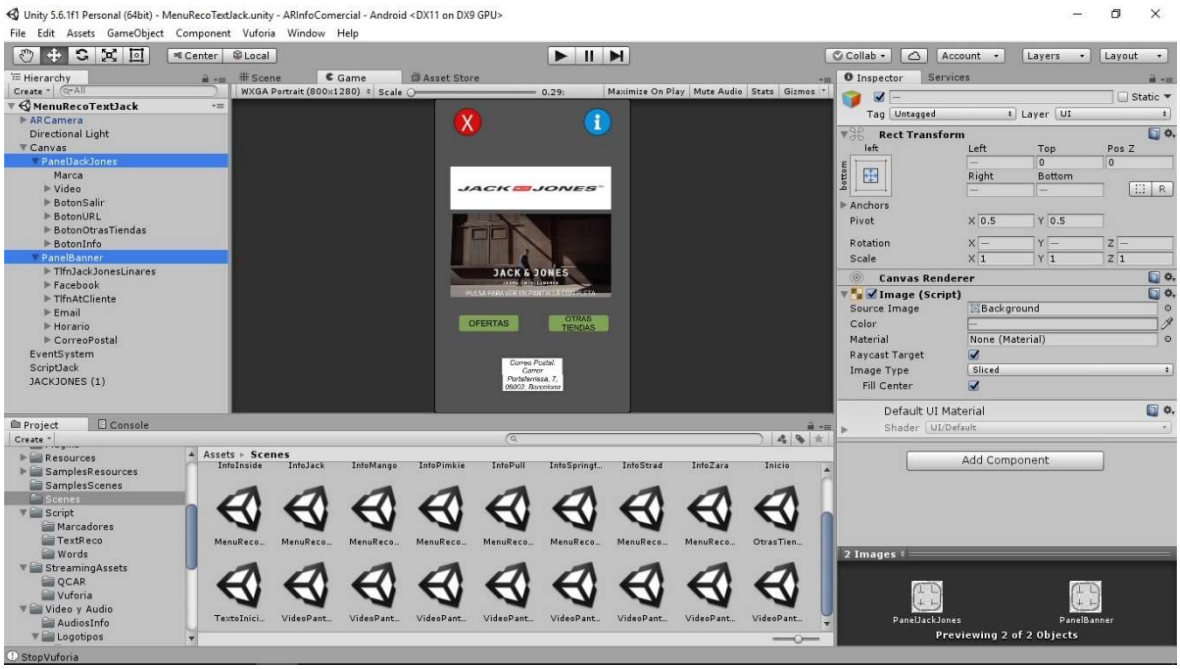


Figura 3.62. *Menú de la tienda Jack & Jones.*

Estas escenas cuentan con dos paneles, uno para el contenido del menú, llamado PanelJackJones y el otro con el contenido del banner. También cuentan con un objeto vacío llamado ScriptJack, que contendrá el script con las funciones de los botones y del banner, y un video de uno de los anuncios de la tienda.

Para la creación del panel con el contenido del menú de cada tienda debemos añadir un objeto ‘Raw Image’ para mostrar el logotipo de la tienda, mediante la creación de un material como se ha realizado anteriormente y arrastrándolo a la ventana del inspector. Crearemos otro objeto ‘Raw Image’ para añadir el video y que se reproduzca sin sonido, para poder realizar esta función debemos tener el video en la ventana de jerarquía, debe ser de formato .mp4 o .mov para que Unity lo detecte. Estos videos están guardados en una carpeta creada en el Asset llamada ‘Videos y Audios’, ya que en caso de que el dispositivo no cuente con acceso a Internet estos videos no harán falta descargarlos. Dentro de esta carpeta crearemos otra para los ‘Render Textures’, que son utilizados para poder darles textura a los objetos ‘Raw Image’. Los ‘Render Textures’ se crean pulsando botón derecho del ratón en la ventana de proyecto. Crearemos tantos como videos tengamos, en este caso son ocho, debemos configurarlos con el mismo tamaño que contiene el video. Una vez creado, pulsamos el video en la ventana de jerarquía para que nos aparezcan las opciones de configuración en la ventana del inspector, buscamos en esta ventana la opción ‘Render Mode’ que cambiaremos a ‘Render Texture’, por lo que nos aparecerá la opción de ‘Target Texture’, aquí elegimos o arrastramos nuestro ‘Render Texture’ creado, con lo que quedará enlazado con el video. Ahora en la ventana del

inspector del objeto ‘Raw Image’ tenemos el script llamado ‘Raw Image’ donde en la opción de ‘Texture’ le pondremos nuestro ‘Render Texture’ creado. Dentro del objeto ‘Raw Imagen’ le añadiremos un botón del mismo tamaño para poder saltar a la escena en la que aparece el video en pantalla completa y con audio, este botón también cuenta con un pequeño texto en la parte inferior que pone ‘Pulsa para ver en pantalla completa’ y así aclarar que se puede pulsar. Los siguientes botones siguen los mismos patrones que antes hemos explicado, el botón para volver a la escena ‘Inicio’, el botón de información para acceder a la escena llamada ‘InfoJack’ donde tenemos información sobre la tienda, el botón de ofertas que te envía a la página oficial desde el navegador, el botón de otras tiendas que abre la escena de ‘Otras Tiendas’ para interactuar con todas las tiendas. El otro panel con el que cuenta esta escena se trata de un banner creado manualmente para cada tienda, este banner contiene información de contacto para cada tienda, como el número de teléfono de la tienda de Linares, el número de teléfono de atención al cliente, para visitar su página de Facebook, el correo electrónico, el horario que tiene en Linares y su correo postal. Toda esta información pertenece a un bucle dentro del script para que vaya rotando cada 5 segundos. Para implementar este banner en la escena debemos añadir al panel un objeto imagen, ajustando su tamaño, posición y color, dentro de esta imagen añadiremos un texto, con el mismo tamaño que la imagen, escribiendo alguna de la información antes mencionada y un botón transparente, también con el mismo tamaño, para poder pulsarlo. Para generar los otros objetos con la diferente información tan solo deberemos duplicar la imagen creada, pulsando sobre el objeto y presionando Ctrl + D, cambiando el título de la imagen en la ventana de jerarquía y el contenido del texto, tenemos que dejar las imágenes en la misma posición que la primera para que a la hora de activar el bucle, éste vaya activando y desactivando las imágenes creando el efecto de banner. Este banner se ha creado para activar otras aplicaciones del dispositivo, de modo que si pulsamos en él cuando aparece la información del teléfono de la tienda de Linares, nuestro dispositivo abrirá la aplicación de llamada con el número de teléfono marcado, al igual que si pulsamos cuando aparece el teléfono de información al cliente, también se nos abrirá la aplicación de Facebook o la aplicación de Mail cuando pulsemos en sus correspondientes imágenes, para el horario comercial y el correo postal no hay ningún botón asociado. Al igual que los botones del panel anterior, las funciones de los botone del banner se encuentra en el script situado en el objeto vacío ‘ScriptJack’.

Al pulsar el botón de información en este menú, se nos abrirá la escena llamada ‘InfoJack’.

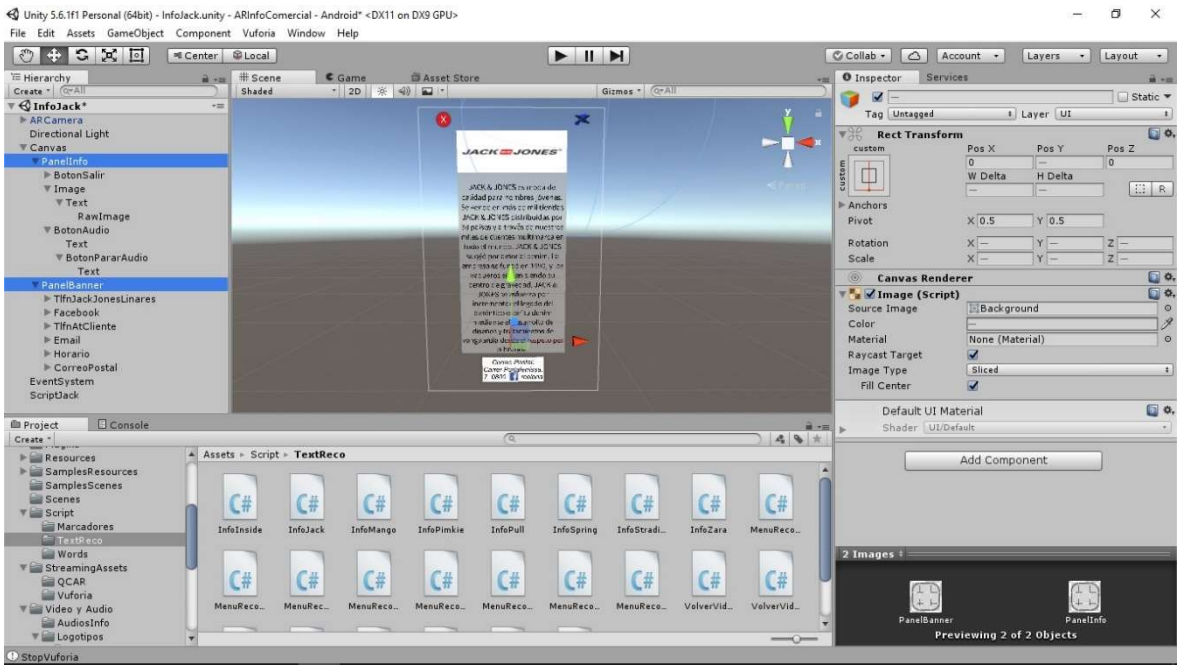


Figura 3.63. *Escena InfoJack.*

Esta escena contiene una vista deslizante con información relacionada con cada tienda, por lo que lo crearemos del mismo modo que los anteriores, también contiene un botón para volver al menú de la tienda y otro botón para reproducir todo el texto. Para generar estos audios hemos utilizado el programa TTSReader, que nos permite seleccionar el tipo de voz que deseamos, además del idioma, tan solo debemos introducir el texto y este programa nos genera un archivo mp3.

Al pulsar el icono de la corchea, se reproducirá este audio y nos aparecerá una X sobre la corchea, si lo pulsamos el audio se parará. Para añadir el audio a esta escena, debemos seleccionar en la ventana de jerarquía el botón al cual le hemos puesto este icono, dentro de la ventana del inspector le añadiremos el componente ‘Audio Source’, arrastramos el audio, que debemos tener guardado en una carpeta dentro de assets en la ventana de proyecto, en la opción de ‘Audio Clip’ y le pondremos las funciones de los botones. Para asignarle las opciones de los botones vamos a poner una orden para reproducir el audio arrastrando el objeto ‘BotonAudio’ a la ventana del inspector, en la opción de ‘On Click ()’, ahora seleccionamos la función de ‘Audio Source’, buscando la opción ‘Play ()’. Para la otra orden, arrastramos el script generado para esta escena, esta función sirve para activar el objeto que nos tacha la corchea, así al pulsarlo una vez, realizamos dos funciones. Dentro de este botón tenemos otro botón con la función de para el audio, que es cuando tenemos la corchea tachada, en este botón también hay dos funciones, una arrastrando el objeto ‘BotonAudio’, con la función de ‘Audio Source’ y poniendo la opción ‘Stop ()’ para detener el audio y la otra desde el script para desactivar el tachado de la corchea.

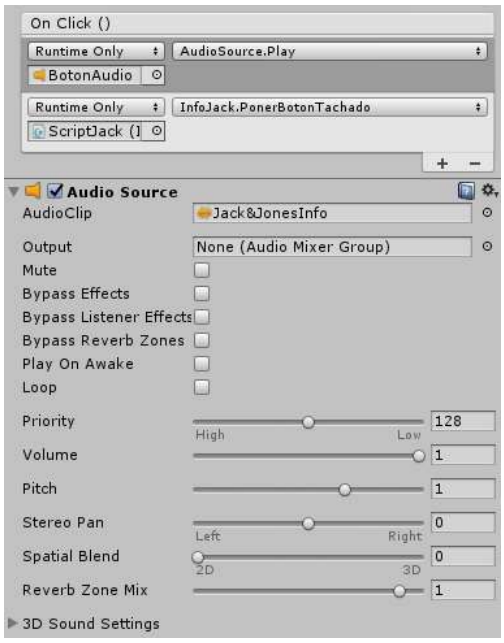


Figura 3.64. *BotonAudio*.

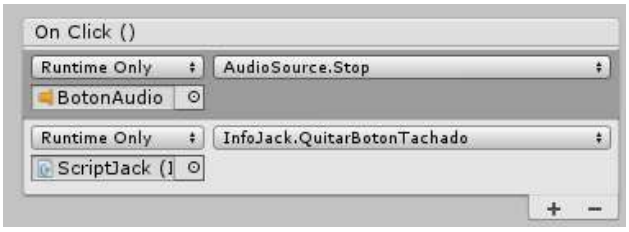


Figura 3.65. *BotonPararAudio*.

En esta escena también se ha creado el mismo panel de banner que en la anterior, con las mismas funciones y el mismo patrón.

La última escena que queda por explicar es la llamada 'VideoPantCompJack', esta se encarga de reproducir el video del anuncio, que hay en el menú de cada tienda, en pantalla completa y con sonido. Por tanto, tendremos una escena de video a pantalla completa para cada tienda.

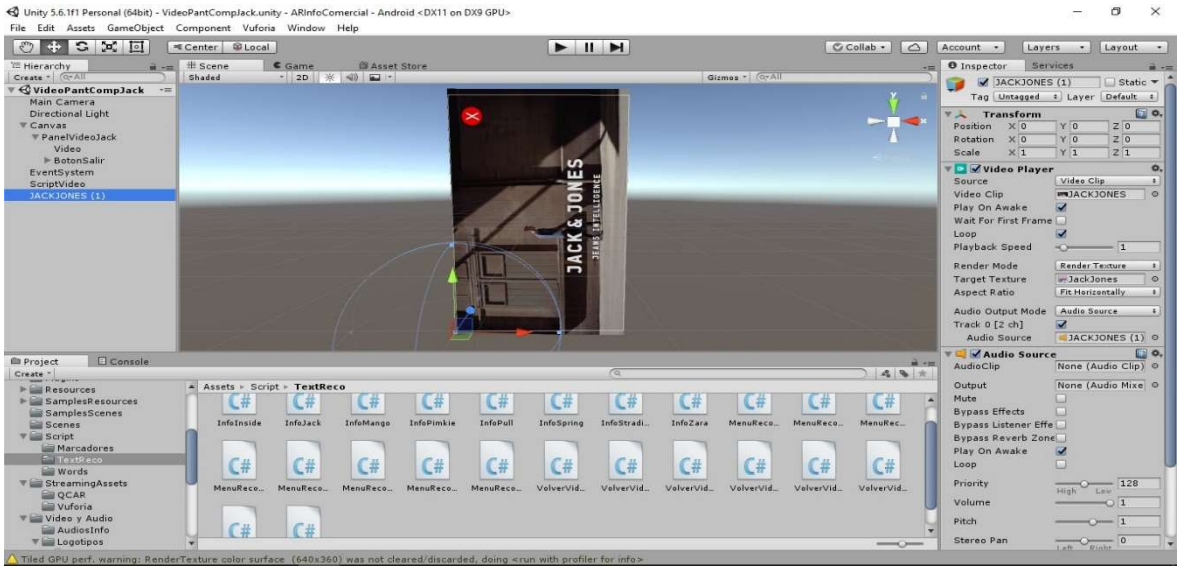


Figura 3.66. *Escena de video a pantalla completa.*

Para generar esta escena debemos añadir el video a la ventana de jerarquía y configurarlo como realizamos anteriormente, con el 'Render Texture', solo que en este caso le

añadimos el componente de ‘Audio Source’ para introducir el sonido en el video y le pondremos la opción de ‘Loop’ en ‘Video Player’ para que se vuelva a reproducir cuando termine. En la figura superior podemos ver cómo está configurado. Para que nos muestre el video debemos añadirle el objeto ‘Raw Image’ configurado con el ‘Render Texture’ que hemos creado anteriormente. Nuestra aplicación se ha diseñado para que muestre todas las escenas en vertical, por lo que debemos configurar el objeto ‘Raw Image’ rotándolo en el eje Z 90 grados y ajustando el tamaño al del canvas para ocupar el tamaño de la pantalla, debido a esta rotación el video se nos mostrará en el dispositivo en horizontal. Esta escena cuenta también con un botón para volver a la escena del menú de la tienda que rotaremos también para tener una interfaz en horizontal.

3.2.5.5 Descargando la aplicación

Una vez explicada todas las escenas y añadidas en ‘Build Setting’ es el momento de descargar el archivo .apk. Estos archivos son los que se utilizan para las aplicaciones de los dispositivos Android. Para poder descargar este archivo debemos realizar unos ajustes que son fundamentales para el correcto funcionamiento de la aplicación, también añadiremos la identificación de nuestra aplicación, el logotipo que tendrá, la versión a la que pertenece y deberemos de crear una carpeta de claves para poder publicar la aplicación. Para realizar estos ajustes debemos acceder mediante la barra de menú en la pestaña de ‘Edit’, buscar la opción ‘Player Setting’ y escoger la opción de ‘Player’, como muestra la figura.

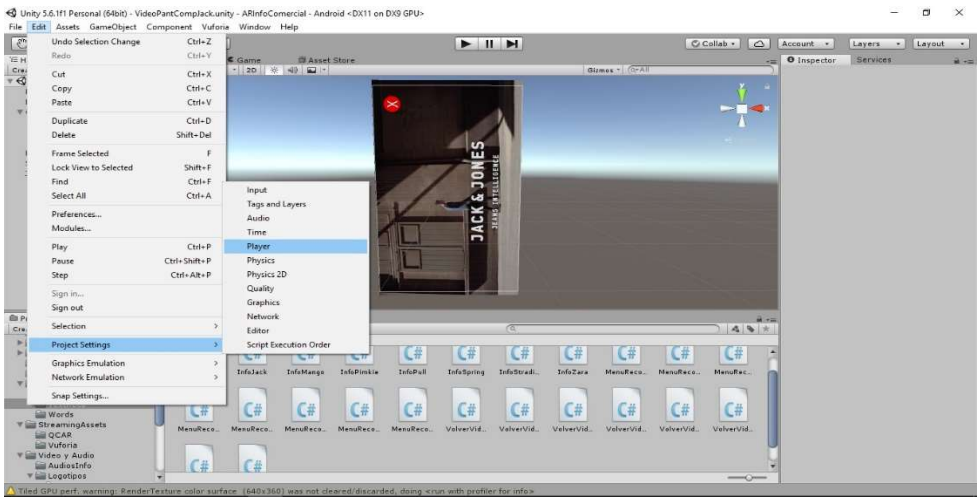


Figura 3.67. *Acceso a los ajustes de la aplicación.*

También podemos acceder mediante la ventana de ‘Build Setting’ pulsando en el botón ‘Player Setting’. Tras acceder a esta configuración nos aparecerá en la ventana del

inspector diferentes apartados en los cuales configuraremos algunas cosas importantes para nuestra aplicación.

Figura 3.68. *Ajustes de la aplicación.*

Los apartados que vamos a configurar son los llamados 'Resolution and Presentation', 'Other Settings' y 'Publishing Setting'. Como podemos observar en la figura le añadimos el nombre de la compañía (en este caso mi nombre), el nombre del producto y el logotipo que tendrá nuestra aplicación, al cual le hemos puesto el escudo de la universidad. En el apartado de resolución y presentación tenemos la opción de configurar que orientación tendrá nuestro dispositivo, en nuestro caso se ha escogido la opción 'Portrait' que es para mantener todas las escenas en vertical.

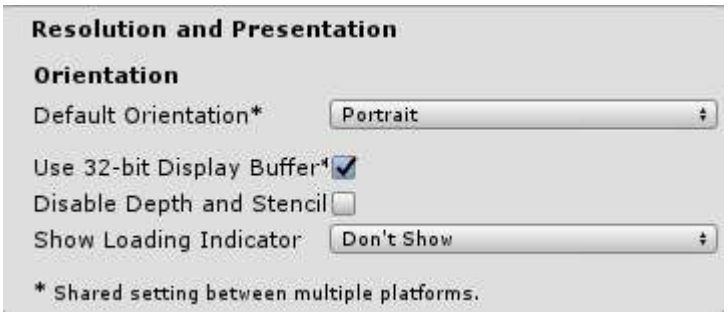


Figura 3.69. *Configuración de la orientación.*

El siguiente apartado que vamos a configurar es el de 'Other Setting', aquí configuraremos el nombre del paquete siguiendo el mismo patrón que en la siguiente imagen, también debemos de tener en cuenta las opciones de 'Version' y de 'Bundle Version Code', ya que estos valores los deberemos de ir aumentando cada vez que vayamos a actualizar la

aplicación, o para descargarla de nuevo, como podemos observar en la figura esta aplicación se ha descargado 12 veces, cada vez para comprobar el funcionamiento en el proceso de desarrollo. En la opción de 'Version' empezaremos por el número 1.0, y en el 'Bundle Version Code' empezaremos por el número 1, en cada descarga le aumentaremos un número a cada opción. Para el resto de opciones ya vienen por defecto como deben de estar.

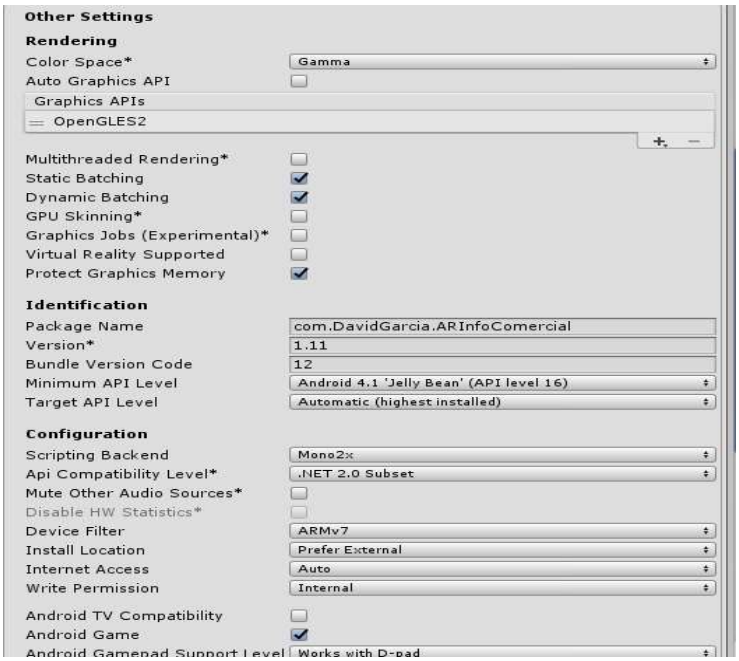


Figura 3.70. *Configuración de identificación.*

El último apartado que vamos a configurar es el más importante, se trata del apartado llamado 'Publishing Setting', en este apartado es donde tendremos que generar una carpeta de claves y asegurarnos de que, a la hora de descargar el archivo, estas claves han sido introducidas, ya que de lo contrario no nos permitirá descargar la aplicación.

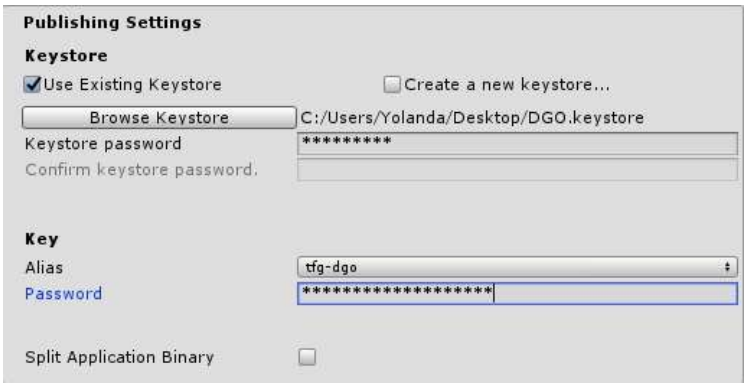


Figura 3.71. *Configuración de publicación.*

Para generar esta carpeta de claves debemos seleccionar la opción ‘Create a new keystore…’, debemos seleccionar el lugar donde se guardará nuestro keystore en el PC, seleccionando el botón de ‘Browse Keystore’ y eligiendo el lugar en el explorador de archivos, escribimos la contraseña que tendrá y la confirmamos, una vez realizado, deberemos de crear un nuevo alias desplegando la pestaña y presionando ‘Create a new key’, esto abrirá una nueva ventana en la cual deberemos rellenar los siguientes datos. Para mayor seguridad debemos escoger una contraseña diferente que la carpeta de claves.

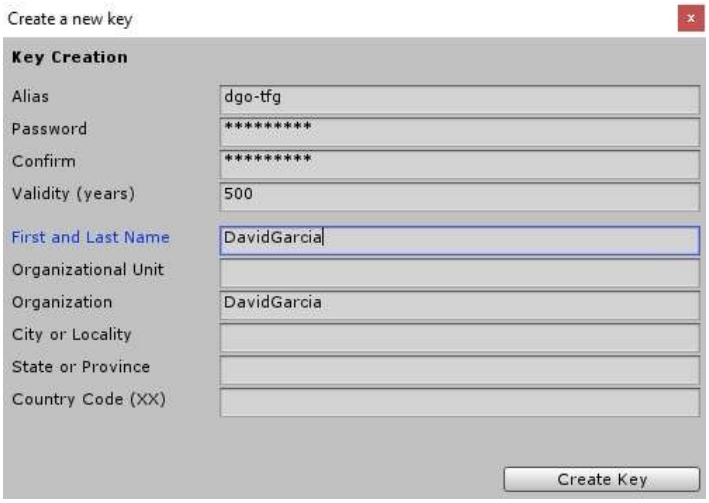


Figura 3.72. *Creación de un nuevo alias.*

Tras crearlo lo seleccionaremos en la opción de ‘Alias’ e introduciremos su clave. Ya estamos listo para poder descargar el archivo de aplicación, tan solo debemos irnos a File > Build Setting y en esta ventana presionaremos el botón de ‘Build’, nos aparecerá el explorador de archivos y lo guardaremos en el lugar deseado, en nuestro caso hemos escogido la misma ruta donde guardamos nuestro proyecto, creando una nueva carpeta llamada App. Una vez en el PC tan solo debemos trasladarlo a un dispositivo Android, este dispositivo necesita de un gestor de archivos, que podemos descargar gratuitamente desde ‘Google Play Store’ para poder buscar el archivo dentro del móvil e instalarlo.

3.3 Desarrollo de los Scripts

En este apartado explicaremos como se han realizado los scripts pertenecientes a cada escena, dentro de cada uno de ellos se realizan las funciones de los botones, bucles de tiempos, abrir otras aplicaciones, carga de escenas, etc.

Antes de ilustrar los contenidos de los scripts debemos tener un conocimiento básico de como realiza las funciones para Unity.

El comportamiento de los GameObjects es controlado por los Components que están adjuntos. Unity nos permite crear nuestros propios Componentes mediante el desarrollo de scripts. Unity soporta dos lenguajes nativamente:

- **C#** (pronunciado C-Sharp), un lenguaje estándar de la industria similar a Java o C++;
- **UnityScript**, un lenguaje diseñado específicamente para uso con Unity y modelado tras JavaScript.

Estos scripts se crean directamente desde la interfaz de Unity, podemos crearlo desde el menú Create en la parte superior izquierda del panel del Proyecto o seleccionando Assets > Create > C# Script (o JavaScript) desde el menú principal. Otra forma de crearlo es situándonos en la carpeta donde queramos crearlo dentro de la ventana del Proyecto y pulsando botón derecho del ratón, presionamos Create y posteriormente buscamos C# Script o JavaScript. El nuevo script será creado en esta carpeta y nos pedirá que ingresemos un nuevo nombre, debemos ingresarle el nuevo nombre en este punto en vez de editarlo después, ya que el nombre que le pongamos será utilizado para crear el texto inicial dentro del archivo. Un script hace sus conexiones con el funcionamiento interno de Unity al implementar una clase que deriva desde la clase integrada llamada MonoBehaviour. Debemos tener en cuanto dos funciones definidas dentro de la clase, y que nos aparecerán por defecto al crear un script. La función **Update** se encargará de la actualización por frame para el GameObject, puede incluir movimiento, bucles, acciones de trigger, básicamente cualquier cosa que vaya a ser manejado durante la aplicación. La función **Start** va a ser llamada por Unity antes de que la función Update sea llamada por primera vez, es un lugar ideal para hacer cualquier inicialización.

Una vez explicado cómo se crea un script y sus dos funciones principales, vamos a ver en detalle los scripts utilizados en la aplicación.

La primera escena que salta cuando arrancamos la aplicación es ‘TextoInicial’, en la que solo contamos con un script llamado ‘InfoInico’. El contenido de este script es el siguiente.

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

public class InfoInicio : MonoBehaviour {
    public GameObject PanelInicio;

    private void Start()
    {
        PanelInicio.gameObject.SetActive(true);
    }

   public void Saltar()
    {
        SceneManager.LoadScene("Inicio");
    }
   private void OnTriggerEnter(Collider other)
        {
        if (other.gameObject.tag == "TextInicio")
            SceneManager.LoadScene("Inicio");
        }
}
```

Script InfoInicio

La palabra *using* tiene diversos usos en el lenguaje C#, como definir un alias, adquirir y liberar recurso y para la importación de espacios de nombres, que es como lo vamos a utilizar en nuestro proyecto. La palabra *using* actúa como una directiva del compilador, haciendo que se puedan utilizar tipos definidos en el espacio de nombre importado sin necesidad de especificarlos de forma explícita en el código. [1]

Cuando se crea un script nos aparece por defecto los espacios de nombre de *System.Collection; System.Collection.Generic; y UnityEngine* con el contenido de interfaces y clases que definen colecciones genéricas y para poder utilizar los componentes de trabajo de Unity. A parte de estos espacios de nombre, le hemos añadido *UnityEngine.SceneManagement* para poder utilizar la herramienta de cambio de escena en Unity. Lo primero que hacemos es crear un GameObject público, este GameObject nos aparecerá en la ventana del Inspector y tan solo deberemos arrastrar el panel deseado para aplicarle la función deseada.

Figura 3.73. *Seleccionando el panel para el Script.*

En este caso nada más arrancar la escena vemos como en la función **Start** el panel que vamos a manipular se activa, con la línea de código *PanelInicio.gameObject.SetActivate(true)*, de este modo le indicamos que el GameObject

que se encuentre dentro de este objeto público sea activado, con la función *SetActivate(true)* al arrancar, en caso de que entre paréntesis tuviésemos escrito *false*, el objeto estaría desactivado, de esta forma podemos manipular la activación y desactivación de los paneles. Lo siguiente que nos encontramos es la variable que hará funcionar el botón de 'Saltar', como podemos ver su acción es la de cargar la escena de 'Inicio', gracias a la función *SceneManager.LoadScene("nombre de escena")*, esta variable debemos hacerla pública para poder seleccionarla en la interfaz de Unity. Por último, nos encontramos con la variable privada *OnTriggerEnter(Collider other)* esta se encarga de la función del colisionador, dentro de esta variable le añadimos la condición de que si el cubo al que le pusimos el Tag de 'TextoInicio' colisiona con el otro cubo al cual se le ha añadido este script, se cargue la escena de 'Inicio'.

La siguiente escena en saltar es 'Inicio', como ya comentamos anteriormente, esta escena es la más importante de la aplicación. En ella se encuentran los scripts que hacen posible las funciones de los diferentes niveles de la realidad aumentada, a estos scripts los hemos llamado 'Geolocalización', 'QRcode', 'EnableDisableGPS' y otros cuya función sirven para realizar acciones con los botones. Empecemos por los scripts que se encuentran dentro de 'AR Camera'.

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

public class SalirInicio : MonoBehaviour {

    public void Salir()
    {
        Application.Quit();
    }

    void Update () {
        if (Input.GetKey(KeyCode.Escape))
        {
            Application.Quit();
        }
    }
}
```

Script SalirInicio

En este script creamos una variable pública llamada *Salir*, que al pulsar el botón asociado nos saldremos de la aplicación, gracias a la línea de código *Application.Quit()* y dentro de la función de **Update** le añadimos una condición, gracias al código *Input.GetKey(KeyCode.Escape)* le indicamos que cuando se pulse el botón de volver con el que cuentan todos los dispositivos móviles Android, también se salga de la aplicación.

Esta condición se añadirá a los scripts de diferentes escenas, pero con otra función, la de cargar la escena anterior.

El siguiente script es el que se encarga de la geolocalización de nuestro dispositivo, para el desarrollo de este script hemos sacado obtenido la función que realiza el enlace del dispositivo móvil con Unity de la siguiente página web, http://wirebeings.com/markerless-gps-ar.html, no todo lo que aparece en esta página ha sido utilizado, tan solo las bases que nos aparecen en el manual de Unity, en la página https://docs.unity3d.com/ScriptReference/LocationService.Start.html, pero nos hizo de guía para poder desarrollar este script con las funciones que nos interesaban.

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

public class Geolocalizacion : MonoBehaviour {

    private float MiLatitud, MiLongitud;
    private float InicioLat = 38.095414f;
    private float FinalLat = 38.094279f;


    void Start () {

        StartCoroutine("GetCoordinates");
    }

    IEnumerator GetCoordinates()
    {
        //Mientras esta condición continúe en true, la función sigue funcionando
        //una vez iniciada la aplicación.
        while (true)
        {
            //Esta condición sirve para comprobar si el usuario tiene
            //el servicio de ubicación activado en su dispositivo.
            if (!Input.location.isEnabledByUser)
                yield break;
            //En esta línea le indiciamos iniciar el servicio antes de consultar la
ubicación.
            Input.location.Start(1f, .1f);

            //Le añadiremos un tiempo de espera hasta que el servicio se inicialice.
            int maxWait = 20;
            while (Input.location.status == LocationServiceStatus.Initializing &&
maxWait > 0)
            {
                yield return new WaitForSeconds(1);
                maxWait--;
            }

            //En caso de que el servicio no se inicialice en 20 segundos,
            //nos mostrará un mensaje de tiempo agotado (Time out).
            if (maxWait < 1)
            {
                print("Timed out");
```

```
                yield break;
            }




            //Esta condición nos indica cuando ha fallado la conexión,
            //mostrándonos un nuevo mensaje.
            if (Input.location.status == LocationServiceStatus.Failed)
            {
                print("Unable to determine device location");
                yield break;
            }
            else
            {
                //En caso de que no falle, los datos de longitud y latitud
                //se sobrescriben cada vez.
                MiLatitud = Input.location.lastData.latitude;
                MiLongitud = Input.location.lastData.longitude;
            }
            Input.location.Stop();
        }
       }
    private void Update()
    {
        if(MiLatitud < InicioLat && FinalLat < MiLatitud)
        {
            SceneManager.LoadScene("OtrasTiendas");
        }
    }
}
```

Script de Geolocalización.

Como podemos observar le hemos añadido el espacio de nombre *UnityEngine.UI*, con el cual podremos manipular los objetos creado en el menú de UI. Para añadir valores debemos declarar una variable del tipo *float*, esta variable almacena valores de punto flotante. Para inicializar esta variable debemos usar el sufijo 'f' detrás de los números, ya que si no lo ponemos obtendremos un error de compilación porque Unity intenta almacenar un valor *double.* Estos valores serán de carácter privado, como vemos tenemos creado *MiLatitud, MiLongitud,* que será donde se almacenen los valores de las coordenadas de nuestro dispositivo, *InicioLat y FinalLat,* que son las coordenadas donde nos saltará la escena 'OtrasTiendas' al llegar por cualquiera de las dos entradas del pasaje del comercio de Linares. En la función de **Start** tenemos una corrutina que se iniciará al arrancar la escena, en nuestro caso se llama *GetCoordinates* y para establecer una corrutina en ejecución se necesita usar la función *StartCoroutine*. Seguidamente tenemos la corrutina, fuera de la función **Start**, que es una función que tiene la habilidad de pausar su ejecución y devolver el control a Unity para luego continuar donde lo dejó en el siguiente frame, en C# una corrutina se declara con *IEnumerator.* Es esencialmente una función declarada con un tipo de retorno *IEnumerator* y con una instrucción de retorno *yield* incluida en algún lugar

de su cuerpo. La instrucción *yield* es el punto en el cual la ejecución se pausará y reanudará en el siguiente frame. Esta corrutina se ha extraído del manual de Unity, tan solo hemos de modificar los nombres, que en nuestro caso son *MiLatitud y MiLongitud.* Una vez tengamos modificado el script deberemos crear una condición en la cual le decimos que, si las coordenadas de mi dispositivo se encuentran entre los valores que hemos implementado, *InicioLat y FinalLat,* se cargue la escena llamada 'OtrasTiendas'.

El siguiente script que nos encontramos en el prefab de 'AR Camera' es el llamado 'QRcode', este se encarga de la detección de los códigos QR, ya que en la Assets Store no nos proporcionaba el código gratuitamente, este script se ha descargado de la siguiente página web, https://stackoverflow.com/questions/30546548/unity-zxing-qr-code-scanner-integration, en el cual las funciones principales se han dejado igual, como la corrutina y el contenido de la función **Update**, las modificaciones y aportaciones las subrayaremos de color amarillo. Tenemos que tener en cuenta que para poder utilizar los espacios de nombre hay que tener importado la librería de ZXing-unity mencionada en el apartado del desarrollo de la aplicación.

```
using UnityEngine;
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine.UI;
using Vuforia;
using System.Threading;
using ZXing;
using ZXing.QrCode;
using ZXing.Common;

[AddComponentMenu("System/VuforiaScanner")]
public class QRcode : MonoBehaviour
{
    private bool cameraInitialized;
    private GameObject DestinoURL;
    public GameObject PanelQR;
    public String textQR = null;

    private BarcodeReader barCodeReader;

    void Start()
    {
        barCodeReader = new BarcodeReader();
        StartCoroutine(InitializeCamera());
        DestinoURL = GameObject.FindGameObjectWithTag("URL");
        PanelQR.gameObject.SetActive(false);
    }

    private IEnumerator InitializeCamera()
    {
        // Esperamos un poco para evitar que Vuforia se colapse.
        yield return new WaitForSeconds(1.25f);
```

```
        var                                 isFrameFormatSet                                =
CameraDevice.Instance.SetFrameFormat(Vuforia.Image.PIXEL_FORMAT.GRAYSCALE, true);
        Debug.Log(String.Format("FormatSet : {0}", isFrameFormatSet));



        // Esta parte es para forzar el autoenfoque de la cámara.
        var                                   isAutoFocus                                   =
CameraDevice.Instance.SetFocusMode(CameraDevice.FocusMode.FOCUS_MODE_CONTINUOUSAUTO)
;
        if (!isAutoFocus)
        {

CameraDevice.Instance.SetFocusMode(CameraDevice.FocusMode.FOCUS_MODE_NORMAL);
        }
        Debug.Log(String.Format("AutoFocus : {0}", isAutoFocus));
        cameraInitialized = true;
    }


    private void Update()
    {
        if (cameraInitialized)
        {
            try
            {
                var                                 cameraFeed                                =
CameraDevice.Instance.GetCameraImage(Vuforia.Image.PIXEL_FORMAT.GRAYSCALE);
                if (cameraFeed == null)
                {
                    return;
                }
                var       data       =       barCodeReader.Decode(cameraFeed.Pixels,
cameraFeed.BufferWidth,                                       cameraFeed.BufferHeight,
RGBLuminanceSource.BitmapFormat.RGB24);
                if (data != null)
                {
                    // Cuando se detecta un código QR se realizan estas acciones.
                    Debug.Log(data.Text);
                    PanelQR.gameObject.SetActive(true);
                    DestinoURL.GetComponent<Text>().text = data.Text;
                    textQR = data.Text;

                }
                else  //En caso de no detectarse nos muestra este mensaje.
                {
                    Debug.Log("Codigo QR no detectado !");
                }
            }
            catch (Exception e)
            {
                Debug.LogError(e.Message);
            }
        }
    }
    public void BotonQR()
    {
        Application.OpenURL(textQR);
        PanelQR.gameObject.SetActive(false);
    }
    public void BotonSalir()
    {
```

```
        PanelQR.gameObject.SetActive(false);
    }
}
```

Script QRCode.

Para este script hemos tenido que añadir los espacios de nombre *UnityEngine.UI y System.Collection.Generic,* ya que al compilar el script se producían una serie de errores. Al añadirlos nos seguían produciendo errores debido a que los espacios de nombre requerían ser mencionados dentro de la corrutina y de la función **Update**. Por tanto, se modificó añadiéndole la referencia del espacio del nombre, en este caso se trataba del nombre de espacio *Vuforia*, en esta misma línea tuvimos que modificar el formato de los códigos QR, ya que este script nos lo proporcionaba en formato RGB y nuestros códigos son en blanco y negro, por lo que se modificó a *GRAYSCALE*. Una vez modificado, es hora de añadirle las funciones que vamos a usar para nuestra aplicación, en nuestro caso creamos dos GameObject, uno público y otro privado, el público será utilizado para activar y desactivar el panel, mientras que el privado se utilizará para buscar en la interfaz de Unity el objeto de 'Texto', con el Tag llamado 'URL', donde aparecerá la URL asociada a cada código QR. Para que este texto cambie cada vez que detecta un código QR diferente, debemos crear también una cadena de texto con la función *String*, en nuestro caso le hemos puesto el nombre de textQR y un valor nulo para estar vacío mientras no se detecte un código QR. En la función **Start** buscamos el objeto a través del Tag mediante *GameObject.FindGameObjectWithTag("nombre del Tag")* y pondremos el panel desactivado para que no aparezca todo el tiempo en la pantalla. En la función **Update** le añadimos que se active el panel cuando detecta un código y mediante *DestinoURL.GetComponent<Text>().text = data.Text* le indicamos que nos muestre el texto asociado al código QR, que es la dirección URL de las páginas web de las tiendas, también le indicamos que el *String* creado tome ese texto como valor, puesto que al pulsar sobre el texto, es decir, sobre el botón transparente creado, se no abrirá el navegador del dispositivo móvil con esa URL mediante la línea de código *Application.OpenURL()*, de esta manera no hace falta crear la función de abrir la página web para cada tienda, se hará automáticamente al obtener la información del código QR. Pulsando este mismo botón desactivaremos el panel de QR. En este panel contamos con un botón para salir por si no queremos acceder a la página web, por tanto, se ha creado otra función que desactiva este panel.

Siguiendo el orden de jerarquía, el siguiente script que nos encontramos está dentro de 'ScriptToggle', este script se llama 'EnableDisableGPS' y es el encargado de activar y desactivar el script de geolocalización, mediante el botón de toggle y además se encarga de los paneles del botón de información.

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

public class EnableDisableGPS : MonoBehaviour {

    private Geolocalizacion hCamera;
    public int counter = 0;
    public Text mytext = null;
    public GameObject PanelInfo;
    public GameObject PanelPrincipal;
    public GameObject PanelAcercade;
    public GameObject PanelComoUsar;

    void Start () {
        hCamera = GameObject.Find("ARCamera").GetComponent<Geolocalizacion>();
        PanelPrincipal.gameObject.SetActive(true);
        PanelInfo.gameObject.SetActive(false);
        PanelAcercade.gameObject.SetActive(false);
        PanelComoUsar.gameObject.SetActive(false);
    }

    public void DesactivarGPS()
    {
        counter++;
        if (counter % 2 == 1)
        {
            hCamera.enabled = false;
            mytext.text = "(Desactivado)";
        }
        else
        {
            hCamera.enabled = true;
            mytext.text = "(Activado)";
        }
    }
    public void BotonInfo()
    {
        PanelInfo.gameObject.SetActive(true);
        PanelPrincipal.gameObject.SetActive(false);

    }
    public void VolverInicio()
    {
        PanelInfo.gameObject.SetActive(false);
        PanelPrincipal.gameObject.SetActive(true);
    }
    public void TextAcercade()
    {
        PanelAcercade.gameObject.SetActive(true);
        PanelInfo.gameObject.SetActive(false);
    }
    public void TextComoUsar()
    {
        PanelComoUsar.gameObject.SetActive(true);
        PanelInfo.gameObject.SetActive(false);
```

```
    }



    public void VolverPanelInfo()
    {
        PanelAcercade.gameObject.SetActive(false);
        PanelComoUsar.gameObject.SetActive(false);
        PanelInfo.gameObject.SetActive(true);
    }

}
```

Script EnableDisableGPS.

Para la función de activar y desactivar el GPS de nuestro dispositivo, creamos la clase privada de *Geolocalización* y la llamamos 'hCamera', de esta manera le indicamos que haga referencia al script 'Geolocalización', también creamos un contador con la función *int* con valor inicial igual a cero y un texto con valor nulo con la función *Text*. Para hacer que 'hCamera' encuentre el script de 'Geolocalización' le indicaremos en la función **Start** la siguiente línea de código *GameObject.Find("donde se encuentre el script").GetComponent<nombre del script>()* de esta manera podremos controlar el script de 'Geolocalización'. Cuando creas el botón de Toggle te aparecerá en la ventana del Inspector la función que quieres que realice este botón, por este motivo, hemos creado una variable como las que creamos para los botones normales llamada 'DesactivarGPS', dentro encontramos *counter++* que se utiliza para sumarle el valor de uno cada vez que pulsemos el botón, a continuación, tenemos una condición, en la cual dice que cuando lleguemos al valor dos en el contador vuelva a valer uno para el botón, para que aunque el contador siga sumando cada vez que pulsas, vuelva a realizar la acción. Dentro de esta condición hemos implementado que se desactive el script con la línea de código *hCamera.enable = false* y también que nos aparezca el texto '(Desactivado)' debajo del botón, de lo contrario se activará el script y en el texto aparecerá '(Activado)'. Lo siguiente que nos encontramos son variables para los botones del panel de información, estos se encargan de activar y desactivar cada panel según los botones que presionemos. Por ejemplo, al pulsar el botón de información, activamos su panel y desactivamos el panel principal (en el que sale el botón de activar y desactivar el GPS), si dentro del panel de información presionamos en el botón de 'Acerca de…' se desactivará el panel de información y se activará el panel 'TextAcercade' y así sucesivamente con el resto de paneles de los que cuenta el botón de información. En este script no se ha podido poner la acción de volver con el botón del dispositivo móvil, ya que no se trata de diferentes escenas y si lo pulsamos saldremos de la aplicación porque así lo tenemos configurado en el script 'SalirInicio'.

Los siguientes scripts utilizados son los de reconocimiento de texto, estos se encuentran dentro de los prefabs llamados 'Word', por lo que tendremos un script para cada palabra de reconocimiento. La función que tienen estos scripts es que cuando se reconozca la palabra cargue la escena del menú de la tienda, por lo que sólo veremos uno de ellos ya que el resto se ha realizado de la misma forma. Para realizar estos scripts, vamos a utilizar uno de los que vienen por defecto en los scripts de Vuforia, llamado 'DefaultTrackableEventHandler' y que también viene incorporado en este prefab, pero para poder realizar las diferentes acciones (para cada una de las tiendas) debemos crear un nuevo script y copiar el contenido de este, ya que si no lo copiamos en el nuevo script y lo modificamos directamente, se guardarán las modificaciones para todas las tiendas, siendo imposible cargar las diferentes escenas en cada palabra. Al nuevo script con el contenido copiado, le hemos llamado 'DefaultRecoSpring' (en este caso se trata de la tienda de Springfield). Al copiarlo le tenemos que cambiar el nombre de la clase y ponerle el mismo nombre del script. Como se trata de un script que viene por defecto tan solo se expondrá la parte modificada en la cual se carga la escena.

```
private void OnTrackingFound()
        {
            //Cuando ha detectado la palabra cargará la escena.
            SceneManager.LoadScene("MenuRecoTextSpring");

            Renderer[] rendererComponents = GetComponentsInChildren<Renderer>(true);
            Collider[] colliderComponents = GetComponentsInChildren<Collider>(true);

            // Enable rendering:
            foreach (Renderer component in rendererComponents)
            {
                component.enabled = true;
            }

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

            }

            Debug.Log("Trackable " + mTrackableBehaviour.TrackableName + " found");
               }
```

Parte del script DefaultRecoSpring

Esta es la variable que se encarga de realizar una acción cuando la palabra es detectada, el script que viene por defecto de por sí solo no hace ninguna acción. Para cargar una escena tan solo debemos escribir el código que viene subrayado. Hay que añadirle también el espacio de nombre *UnityEngine.SceneManagement* para poder utilizarlo.

Los últimos scripts que nos encontramos en esta escena son los utilizados por los marcadores, estos cumplen la función de abrir el navegador con la página oficial de la tienda y el otro te abre la aplicación de Maps que vienen incorporados en los dispositivos móviles.

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

public class BotonMangoURL : MonoBehaviour
{
    private void OnMouseDown()
    {
        Application.OpenURL("http://shop.mango.com/es/mujer");
    }
}
```

Script BotonMangoURL.

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

public class BotonMangoMaps : MonoBehaviour
{
    private void OnMouseDown()
    {
        float startTime;
        startTime = Time.timeSinceLevelLoad;

        //open app maps

Application.OpenURL("externalhttp://maps.google.com/maps?saddr=&daddr=38.0947087,-
3.6353285&dirflg=w");

        if (Time.timeSinceLevelLoad - startTime <= 1f)
        {
            //Fail. Open Chrome
            Application.OpenURL("https://www.google.es/maps/dir//38.0947087,-
3.6353285/@38.0947087,-3.6353285,21z/data=!4m2!4m1!3e2");
        }
    }
}
```

Script BotonMangoMaps

En estos scripts tenemos la función de *OnMouseDown* la cual utilizamos para darle la función de botón. En el primero utilizamos la misma función de *Application.OpenURL()* y en el segundo script le añadimos una variable de tiempo, que se utilizará para el caso en que tarde mucho tiempo en abrirse la aplicación de Maps, se abra el navegador con la página de Google Maps. Para acceder a la aplicación de Maps tenemos que usar la referencia *externalhttp://maps.google.com/maps?,* dentro de la función *Application.OpenURL,* para indicar las coordenadas que queremos que nos salgan en la ruta le añadimos *saddr=&daddr=* y las coordenadas donde se encuentre la tienda y por

último le ponemos seguidamente como se va a realizar la ruta, añadiendo *&dirflg=w*, al ponerle la 'w' le estamos indicando que la ruta se hará andando, si le ponemos una 'c' la ruta se realizaría en coche. En caso de que la aplicación no cargue, se abrirá el navegar con la URL en la cual ya aparecerá la ruta indicada.

En la escena del menú de la tienda tenemos tan solo un script que se encarga de las funciones de los botones y del banner o publicidad que se ha realizado manualmente, en este caso se trata de la tienda InSide y la escena se llama 'MenuRecoTextInside'.

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

public class MenuRecoInside : MonoBehaviour
{
    public GameObject fondo1;
    public GameObject fondo2;
    public GameObject fondo3;
    public GameObject fondo4;
    public GameObject fondo5;
    public GameObject fondo6;
    public float tiempo = 0.0f;

    private void Start()
    {
        fondo1.gameObject.SetActive(false);
        fondo2.gameObject.SetActive(false);
        fondo3.gameObject.SetActive(false);
        fondo4.gameObject.SetActive(false);
        fondo5.gameObject.SetActive(false);
        fondo6.gameObject.SetActive(false);
    }
    public void Salir()
    {
        SceneManager.LoadScene("Inicio");
    }
    public void VideoPantComp()
    {
        SceneManager.LoadScene("VideoPantCompInside");
    }
    public void PaginaWeb()
    {
        Application.OpenURL("http://inside-shops.com/");
    }
    public void Info()
    {
        SceneManager.LoadScene("InfoInside");
    }
    public void Tiendas()
    {
        SceneManager.LoadScene("OtrasTiendas");
    }

    void Publi()
    {
        if (tiempo < 5)
        {
```

```
            fondo1.gameObject.SetActive(true);
        }
        else
                if (tiempo > 5 && tiempo < 10)
        {
            fondo2.gameObject.SetActive(true);
            fondo1.gameObject.SetActive(false);
        }
        else
                if (tiempo > 10 && tiempo < 15)
        {
            fondo3.gameObject.SetActive(true);
            fondo2.gameObject.SetActive(false);
        }
        else
                if (tiempo > 15 && tiempo < 20)
        {
            fondo4.gameObject.SetActive(true);
            fondo3.gameObject.SetActive(false);
        }
        else
                if (tiempo > 20 && tiempo < 25)
        {
            fondo5.gameObject.SetActive(true);
            fondo4.gameObject.SetActive(false);
        }
        else
                if (tiempo > 25 && tiempo < 30)
        {
            fondo6.gameObject.SetActive(true);
            fondo5.gameObject.SetActive(false);
        }
        else if (tiempo > 30.5)
        {
            tiempo = 0;

            fondo6.gameObject.SetActive(false);

        }
    }

    public void AbrirFacebook()
    {
        float StartTime;
        StartTime = Time.timeSinceLevelLoad;

        //Abrir App de Facebook
        Application.OpenURL("fb://page/147895946325");
        if (Time.timeSinceLevelLoad - StartTime <= 2f)
        {
            Application.OpenURL("https://www.facebook.com/InsideShops/");
        }
    }
    public void Llamada()
    {
        Application.OpenURL("tel: 953651623");
    }
    public void Email()
    {
        Application.OpenURL("mailto:contact@inside-shops.com");
    }
    public void AtCliente()
```

```
    {
        Application.OpenURL("tel: 968301944");
    }

    void Update()
    {
        if (Input.GetKey(KeyCode.Escape))
        {
            SceneManager.LoadScene("Inicio");
        }

        tiempo = tiempo + 1 * Time.deltaTime;
    }
}
```

Script MenuRecoInside

Prácticamente todo el contenido de este script ha sido diseñado para la creación del banner. Comenzamos creando GameObject públicos para poder arrastrarles en la ventana del Inspector las imágenes con el contenido de la publicidad. En este caso se ha diseñado para seis contenidos de información, pudiendo añadir todos los que nos interesen. Lo siguiente que creamos una variable de tiempo del tipo *float*, con un valor inicial de 0 segundos. Nada más arrancar la escena estas imágenes están desactivadas como podemos ver en la función **Start**, aunque al tener el bucle insertado en la función **Update**, llamado *Publi()*, este inicia su función prácticamente al iniciar la escena, otro dato importante en la función Update está relacionado con la variable de tiempo, ya que debemos ponerle la función *Time.deltaTime* para tener un contador de tiempo real, se ha añadido con la siguiente línea, *tiempo = tiempo + 1 * Time.deltaTime*, sumando un segundo a cada frame a nuestra variable de tiempo. La variable diseñada como bucle tiene las condiciones de que cada imagen dure cinco segundos activado, al pasar ese tiempo se desactiva y se activa la siguiente imagen, gracias a la función *SetActivate(),* en conjunto todas las imágenes duran treinta segundos y cinco décimas de segundo después, el tiempo vuelve a valer cero, que es la última condición que nos encontramos en el bucle, empezar a activar y desactivar imágenes. Una vez explicado cómo se ha realizado este bucle para hacer que esté en constante movimiento, el resto de contenido del script son para las acciones de los botones, estas acciones se van a realizar con dos funciones que ya hemos visto anteriormente, *SceneManager.LoadScene()* y *Application.OpenURL().* Dentro del banner tenemos que cuatro imágenes nos abren otras aplicaciones de nuestro dispositivo móvil cuando pulsamos sobre ellas en la aplicación. La función de llamada se abre mediante el parámetro *tel:*, introduciéndolo de la siguiente manera, *Application.OpenURL("tel: Nº de telf.")*, en nuestra aplicación esta función se utilizara para llamar a la tienda perteneciente al pasaje del comercio de linares y para llamar al teléfono de atención al cliente. Otra de las funciones que abrimos con este banner es la del Mail, esto se consigue con el parámetro *mailto:*, de la misma manera que en caso anterior,

Application.OpenURL("mailto:Correo electrónico"), algunas de estas tiendas no cuenta con correo electrónico para comunicarte con ellos y lo tienes que hacer a través de su página web, en estos caso se ha obtenido la dirección URL de esa sección de la página y al pulsar el botón en vez de abrirse la aplicación de Mail se nos abre el navegador con su página de contacto. La última aplicación que nos abre este banner es la de Facebook, dentro de esta variable se ha introducido un tiempo, el cual se inicia cuando se presiona la imagen, para el caso de que si nuestra aplicación de Facebook del dispositivo móvil tarda en arrancar un cierto tiempo entonces se abriría el navegador con la página oficial de Facebook de la tienda. El parámetro que abre la aplicación de Facebook es *fb:*, cada página de Facebook está asociada a unos números, ya que si le ponemos la URL del navegador, la aplicación no se abrirá. Estos números se obtiene de la siguiente manera, desde nuestro navegador entramos a Facebook y buscamos la página oficial de alguna de las tiendas que nos interesen, en este caso InSide, una vez que estemos dentro pulsamos botón derecho del ratón para abrir su código fuente, en él encontraremos este número de referencia buscando en el código *property="al:android:url"*, justo después nos encontramos con su contenido (*content*) y de ahí copiaremos el código hasta donde acaben los números. Como muestra la siguiente figura.

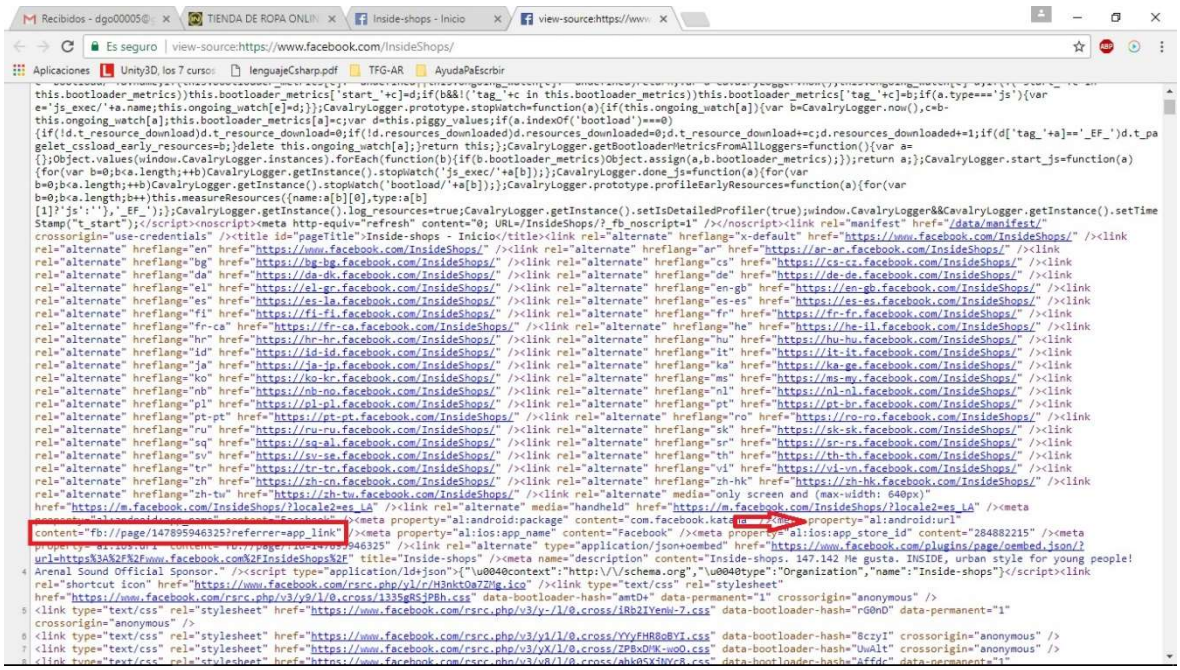


Figura 3.74. *Obteniendo la URL para móvil de Facebook.*

Con el resto de variables restantes del script le daremos funciones a los botones del menú, teniendo uno para volver a la escena anterior pulsando el botón o dándole al botón de volver del dispositivo móvil, otro para abrir el navegador con la página oficial de la tienda y

otros tres que nos abrirán las escenas de información, video en pantalla completa y la escena de otras tiendas.

Dentro de la escena de información de cada tienda, en este caso 'InfoInside', tenemos un script con el contenido de funcionamiento del banner, con los mismo parámetros y variables que los explicados en el script anterior, además tenemos dos variables que hacen la función del botón de audio, ya que el contenido es prácticamente el mismo que en el anterior, tan solo mostraremos las variables del botón de audio. Donde se encuentra la creación de los fondos, al inicio del script, debemos añadirle otro para la imagen del botón de audio, mediante la línea *public GameObject DesactivaAudio;* y arrastramos la imagen en la ventana del Inspector. En la función **Start** le indicamos que esta imagen esté desactiva cuando se inicie la escena, mediante la siguiente línea *DesactivaAudio.gameObject.SetActivate(false)*, por último tenemos las acciones que realizaran los botones añadidas mediante variables.

```
public void PonerBotonTachado()
    {
        DesactivaAudio.gameObject.SetActive(true);
    }
    public void QuitarBotonTachado()
    {
        DesactivaAudio.gameObject.SetActive(false);
    }
```

Variables del botón de audio dentro del script InfoInside.

Como podemos ver, al pulsar el botón activaremos o desactivaremos la imagen asociada a tachar la corchea o quitar el tachado. También cuenta con las dos opciones de volver a la escena anterior, pulsando el botón de la interfaz o pulsando el botón de volver del dispositivo móvil.

Otro de los scripts asociados a la aplicación se encuentra en la escena de 'VideoPantComZara', que para esta explicación se ha escogido la tienda de Zara. Este script tan solo cuenta con las opciones de volver al menú anterior, ya que en esta escena se muestra un video en pantalla completa relacionado con la tienda.

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

public class VolverVideoZara : MonoBehaviour
{

    public void Volver()
    {
        SceneManager.LoadScene("MenuRecoTextZara");
    }
```

```
    void Update()
    {
        if (Input.GetKey(KeyCode.Escape))
        {
            SceneManager.LoadScene("MenuRecoTextZara");
        }
    }
}
```

Script VolverVideoZara

Como podemos observar hemos utilizado la función de cargar escenas, en este caso para volver al menú anterior ya sea presionando el botón de la interfaz o pulsando en el botón de volver del móvil.

La última escena que nos encontramos es la llamada 'OtrasTiendas', en ella tan solo tenemos el script llamado 'BotonesOtrasTiendas' que se encarga de cargar las escenas correspondientes a los menús de cada tienda, por lo que solo utilizaremos la función *SceneManager.LoadScene()* y le asignaremos a cada botón su tienda correspondiente. También cuenta con la función de volver a la escena 'Inicio', como en los anteriores.

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

public class BotonesOtrasTiendas : MonoBehaviour {
    public void Zara()
    {
        SceneManager.LoadScene("MenuRecoTextZara");
    }
    public void Inside()
    {
        SceneManager.LoadScene("MenuRecoTextInside");
    }
    public void Springfield()
    {
        SceneManager.LoadScene("MenuRecoTextSpring");
    }
    public void JackJones()
    {
        SceneManager.LoadScene("MenuRecoTextJack");
    }
    public void PullBear()
    {
        SceneManager.LoadScene("MenuRecoTextPull");
    }
    public void Stradivarius()
    {
        SceneManager.LoadScene("MenuRecoTextStradivarius");
    }
    public void Mango()
    {
        SceneManager.LoadScene("MenuRecoTextMango");
    }
```

```
    public void Pimkie()
    {
        SceneManager.LoadScene("MenuRecoTextPimkie");
    }
    public void Salir()
    {
        SceneManager.LoadScene("Inicio");
    }
    void Update () {
        if (Input.GetKey(KeyCode.Escape))
        {
            SceneManager.LoadScene("Inicio");
        }
    }
}
```

Script OtrasTiendas

CAPITULO 4

DISCUSIÓN

En este apartado vamos a explicar los problemas que nos surgieron durante el desarrollo de nuestro proyecto y las soluciones implementadas a dichos problemas.

Dentro de la interfaz de Unity nos encontramos con el problema de que no podíamos hacer funcionar objetos 3D con los objetos del menú de UI, en un primer momento se quería combinar estos dos objetos para dar una sensación aún mayor de realidad aumentada, como textos en 3D combinados con paneles y objetos del menú UI, pero el motor de Unity sólo nos mostraba los objetos UI haciendo caso omiso a los objetos 3D, por este motivo se descartó combinarlos y se implementó los objetos del menú UI en los menús de las tiendas, mientras que para los marcadores se implementó objetos 3D.

En la creación de los marcadores nos encontramos con que los primeros que creamos no era lo suficientemente buenos como para que Vuforia tuviese las referencias necesarias para una mayor valoración, es por ello que se crearon nuevos marcadores con un mayor número de referencias, implementado fondos de muchos objetos como piedras, pelotas, etc.

Los videos relacionados con la publicidad de las tiendas se obtuvieron en un primer momento con mucha resolución, esto hacia que la aplicación, a la hora de reproducir el video, tardara demasiado en cargarlo, además de que nuestra aplicación ocupaba un tamaño mayor de almacenamiento que con los nuevos videos implementados de menor resolución, también depende del procesador del dispositivo móvil, ya que la computación se realiza de forma más rápida en dispositivos de última generación. Se decidió poner estos videos de carácter estático por si el dispositivo no contaba con acceso a internet y para no tener que esperar a la descarga de dicho video, haciendo que se pueda disfrutar al instante.

Dentro del reconocimiento de palabras nos encontramos con varios problemas que se fueron descubriendo a medida que probábamos esta función. El paquete descargado de palabras en inglés no reconocía algunos de los nombres de las tiendas y no se podían implementarlas dentro del paquete, como por ejemplo Pimkie, esto se descubrió gracias a una escena que viene implementada en la librería de Vuforia, llamada ‘Vuforia-3-TextReco’, la cual se encuentra dentro de la carpeta ‘SamplesScenes’ que se crea en el directorio Assets al importar dicha librería. Esta escena de ejemplo realiza un reconocimiento de palabra obteniendo las palabras o parte de ellas que se encuentran dentro del paquete de palabras, dándonos así las referencias sobre que se va a reconocer para cada nombre de

las tiendas. En el caso de Springfield no había ningún problema, ya que reconocía toda la palabra, pero con Pimkie o Stradivarius no sucedía lo mismo, ya que de Pimkie solo reconocía la palabra 'Pi' y en Stradivarius tan solo se reconocía las palabras 'strad' y 'arius', es por ello que en nuestra aplicación tuvimos que implementar estas partes del nombre de la tienda, en vez del nombre completo. Una vez realizado y comprobado dicho reconocimiento con textos en papel, era hora de probarlos con los nombres reales que aparecen en las tiendas, donde encontramos otros problemas. En las tiendas que no se reconocían los nombres completos y pusimos las partes que detectaba, a la hora de reconocimiento tenemos que enfocar solo la parte que introducimos, es decir, en el caso de Stradivarius, para obtener un buen reconocimiento debemos intentar enfocar 'strad', de lo contrario puede que la aplicación no reconozca esta parte del nombre completo. Otro de los problemas fue el diseño de fuente con el que estuviese escrito el nombre de la tienda, ya que en el caso de Mango su logotipo tipográfico está formado por letras gruesas con amplio contraste entre sus formas, es por ello que a la hora de reconocer el nombre de la tienda en el pasaje del comercio no funcionase bien y costase mucho trabajo realizar este reconocimiento. Otro de los inconvenientes que nos encontramos fue en la tienda de Jack & Jones, debido a que esta tienda tiene su nombre escrito en un escaparate de cristal, esto hacia que se produjesen reflejos de la luz solar y no captase bien el reconocimiento de la palabra, por lo que la luz y los reflejos en algunas superficies influyen a la hora de reconocer la palabra. Todo esto teniendo en cuenta que Unity no enfocaba las palabras si el dispositivo se encontraba en vertical, se probó a rotar 90 grados cada palabra en la interfaz de Unity para poder reconocerlo en vertical, pero tampoco funcionaba, es por ello que se decidió explicar cómo debía estar posicionado el dispositivo móvil dentro del panel de 'como usar', accediendo a través del panel de información de la escena principal.

En la función GPS que desarrollamos en nuestra aplicación tuvimos que modificar el script y ponerlo para una nueva escena, 'OtrasTiendas', ya que en un primer momento se quería hacer saltar cada menú de tienda de forma independiente, es decir, que cuando llegásemos a la puerta de la tienda Springfield automáticamente saltase el menú de esta tienda, el problema encontrado fue que cuando dos tiendas se encontraban una pegada a la otra, o una enfrente de la otra, el dispositivo no era capaz de distinguir cuando se encontraba en una ubicación exacta, ya que los valores de latitud y de longitud que nos proporciona nuestro dispositivo móvil tienen un margen de error de entre 8 y 11 metros, dependiendo del dispositivo, es decir, los valores de latitud y longitud varían constantemente aunque nos quedemos quietos en un mismo lugar, esto hacía imposible que el dispositivo detectase diferencias entre valores cercanos de latitud y longitud. Esto se comprobó viendo estos valores en una aplicación que se hizo de prueba para el correcto funcionamiento de la

función de GPS, en esta prueba los valores de longitud fluctuaban bastante más que los de latitud, y en un principio nuestro script tenía una condición dentro de otra condición, en la cual la primera condición se trataban de valores de latitud y la segunda condición de valores de longitud, una vez que el dispositivo se encontrase dentro de estos valores debía saltar el menú de la tienda, pero el resultado no fue el esperado. Es por ello que se creó solo unos valores de latitud, al empezar y al terminar el pasaje del comercio, ya que los valores de longitud hacían más complicado que saltase la escena debido a su variación continua. Nuestra solución fue poner una escena que saltase una vez que te encontrase en una de las entradas del pasaje del comercio, con una diferencia mayor entre los valores de latitud, para su correcto funcionamiento.

En la parte de información de las tiendas, se intentó en un primer momento que el texto introducido fuese leído desde el mismo Unity, con ayuda de algún paquete o script, pero este no fue el caso, ya que no se encontró ni en el manual de Unity ni en ninguna página de internet, paquetes relacionados con esta función. Por lo que se optó en añadir estos audios con carácter estático realizados desde el programa TTSReader.

El último inconveniente que nos encontramos a la hora de realizar nuestra aplicación fue implementar un banner, en un primer momento se descargó un paquete que proporciona google, llamado 'GoogleMobileAds' el cual se puede enlazar con Unity, pero estos anuncios los proporcionaría google, siendo los más populares en ese momento, y no nos dejaba introducir anuncios relacionados con nuestras tiendas, sin meternos en términos legales. Este paquete está orientado a ganar dinero con la publicidad en tu juego o aplicación, dando por ejemplo monedas o algún tipo de beneficio para su aplicación a los usuarios que viesen dichos anuncios. Ya que en nuestro caso no teníamos interés en sacar beneficios de esa forma y que lo que queríamos era poner publicidad o anuncios relacionados con cada tienda, se optó por desarrollar un banner con la información que deseábamos, en este caso con información de contacto para cada tienda.

CAPITULO 5

CONCLUSIONES

Antes de realizar una valoración de los resultados obtenidos, es importante recordar cuáles eran los objetivos principales de este proyecto. En primer lugar, se debía crear el diseño de una aplicación de realidad aumentada que suministre información de carácter turístico/comercial sobre una calle comercial de una ciudad. Para su uso deberemos utilizar marcadores y posicionamiento GPS para definir el elemento sobre el que se suministrará información aumentada, debiendo contar con información tanto de carácter estático como de carácter dinámico.

Una vez estudiado el diseño y en que plataforma de desarrollo se iba a utilizar, llegamos al objetivo principal de este proyecto, el desarrollo de la aplicación que permita suministrar información añadida dentro de una aplicación para dispositivo móvil.

Habiendo mencionado los objetivos de este proyecto considero que los resultados obtenidos de la aplicación creada, han sido correctos. En esta aplicación se pude comprobar cómo se hace el uso de posicionamiento GPS, así como de marcadores, además, también se le añadió reconocimiento de texto y lector de códigos QR, todo ello generado con ejemplos de diferentes páginas, con ayuda de los manuales de Vuforia y Unity, y con diferentes tutoriales vistos en YouTube. Para un desarrollador que trabaje para una gran empresa, tendría la opción de realizar una aplicación con ayuda de la tienda de distribución de Unity, que nos ofrece paquetes de diferentes precios con los códigos y contenidos para poder realizar un proyecto con estas características, no obstante, ninguno es gratuito.

Ahora vamos a centrarnos en los resultados obtenidos de los diferentes programas y librerías utilizadas en el desarrollo de este proyecto.

5.1 UNITY Y VUFORIA

Para la realización de este proyecto se probaron dos plataformas, Android Studio y Unity. Gracias a los diferentes tutoriales, manuales, ejemplos y a su intuitiva interfaz, se llegó a la fácil conclusión de que nuestra plataforma seria Unity. Desde mi punto de vista es una de las mejores plataformas en su competencia, ya que se trata de una plataforma para hacer aplicaciones hibridas. Este proyecto estaba orientado a realizar la aplicación en los sistemas operativos iOS y Android, pero debido a las dificultades que se presentaban a la hora de realizarla para el sistema operativo iOS, tales como pagar para hacerte

desarrollador o disponer de un ordenador Apple para poder realizar el desarrollo de la aplicación, se tomó la decisión de realizarla solo en Android.

Es una herramienta fantástica para aprender y entretener a un gran colectivo de personas, pero Unity en sí no nos hubiese servido de nada, ya que está pensado para realizar aplicaciones en realidad virtual orientadas a juegos, gracias a la versatilidad de esta plataforma, se puede incorporar las librerías de Vuforia, la herramienta clave para poder realizar realidad aumentada, siendo de gran ayuda todos los contenidos proporcionados por las propias librerías, desde ejemplos de escenas a Prefabs que cumplen funciones específicas. Gracias a ellos y a un gran número de documentos y tutoriales, se pudo realizar los objetivos de nuestro proyecto, haciendo qué, los elementos virtuales que se le implementan a la realidad física se hagan de manera intuitiva y simple, eligiendo su tamaño, posición, textura, color, etc. en una interfaz gráfica, sin los quebraderos de cabeza que conlleva incorporarlos desde los scripts.

5.2 LÍNEAS FUTURAS

Cabe mencionar las posibles mejoras de implementación que podrían introducirse a la aplicación, como crear una base de datos con las palabras específicas, además de las distintas fuentes de texto para el correcto funcionamiento del reconocimiento de texto. Por otra parte, sería interesante poder realizar este tipo de aplicación en colaboración con una empresa, como Zara, por ejemplo, creando así una aplicación con el contenido de la base de datos de la tienda en cuestión y añadir el contenido de sus productos para poner las distintas ofertas que tengan, o los productos que se encuentran en la tienda, haciendo también posible una adaptación del posicionamiento GPS para esa tienda en concreto. En caso de que se realice esta colaboración, también sería necesario adaptar el contenido grafico para que se ajuste a los gustos de la empresa.

Otro trabajo a realizar, podría ser la incorporación de esta aplicación a un gran centro comercial, como el que se encuentra en Granada, llamado “Nevada”. En caso de realizar este trabajo debería hacerse un mapa interno, con la ayuda de Unity, y combinarlo con el posicionamiento GPS para realizar un mapa del centro comercial y que el usuario sepa donde se encuentra cada tienda o saber que ofertas o tiendas tenemos a nuestro alrededor.

Por último, otro proyecto futuro seria la incorporación de uno o varios videojuegos, ya que siempre que vamos a comprar, ya sea centros comerciales o pasajes comerciales, nos encontramos con personas que solo están esperando a que su acompañante realice sus comprar. De esta manera, combinando información con entretenimiento, se puede tener a ambos usuarios entretenidos con la aplicación.

Además, en un futuro próximo, podremos ver esta tecnología a pleno rendimiento, olvidándonos así de los marcadores, gracias a la tecnología SLAM. Con esta tecnología se podrían crear lunas delanteras de automóviles que cuenten con realidad aumentada, de esta manera podremos tener información de todo nuestro entorno, además de la posible utilización de GPS, para que cuando nos encontremos en la carretera podamos seguir las indicaciones que nos proporciona, como un Google Maps o un TomTom.

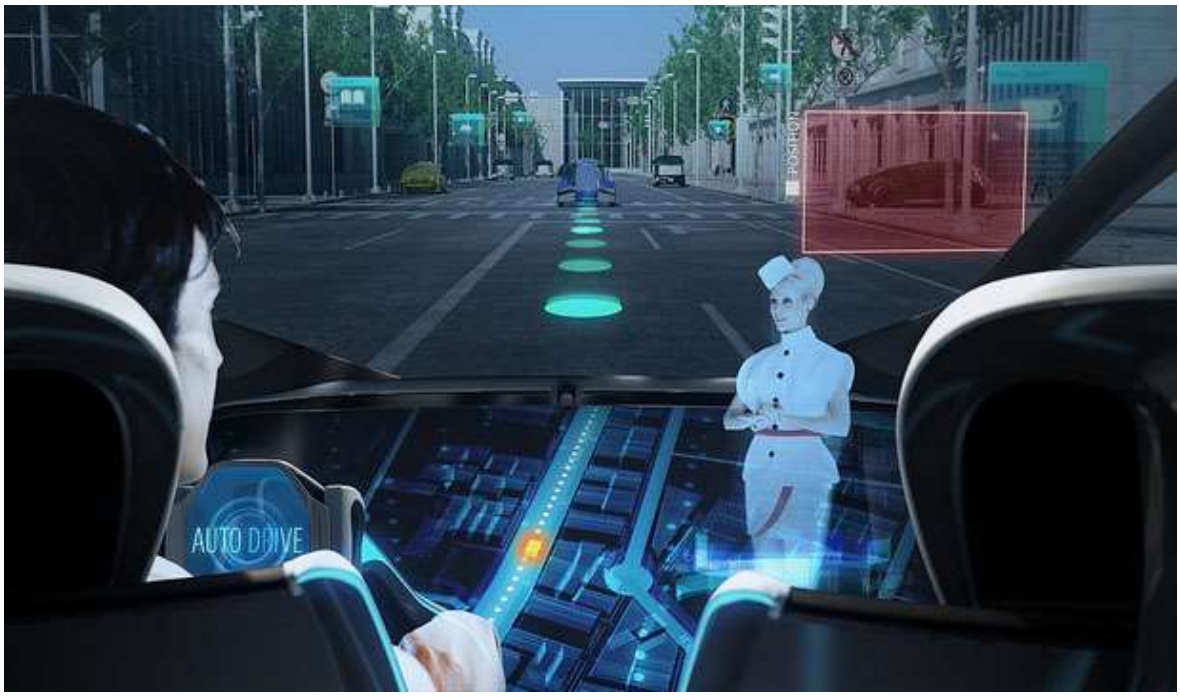


Figura 5.1. Coches futuristas.

Otro posible uso de esta tecnología es la de implementar publicidad en las fachadas de los edificios de una ciudad, como podemos observar en la película futurista “Ghost in the Shell”. Pero para ello deberemos esperar unos años más.

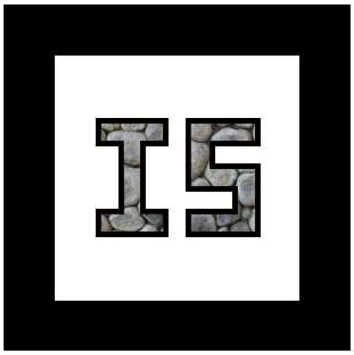
Figura 5.2. Ciudad de la película "*Ghost in the Shell".*

En resumen, el desarrollo que se ha implementado en este trabajo de fin de grado ha sido muy satisfactorio, el trabajo obtenido cumple con los objetivos planteados inicialmente, cuanto más se profundizaba en estos programas, más funciones se descubrían. Además, gracias a la realización de este proyecto se han obtenido muchos conocimientos nuevos, de los cuales seguro me serán útiles en el futuro.

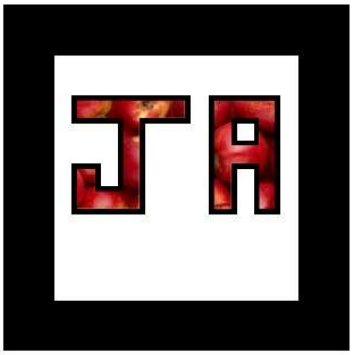
ANEXOS

ANEXO 1

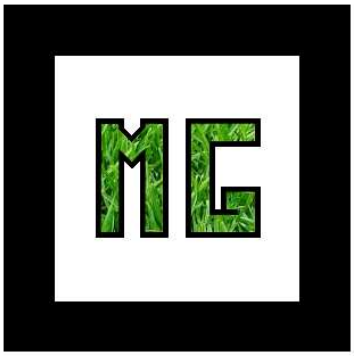
Marcadores realizados.



Marcador tienda InSide.

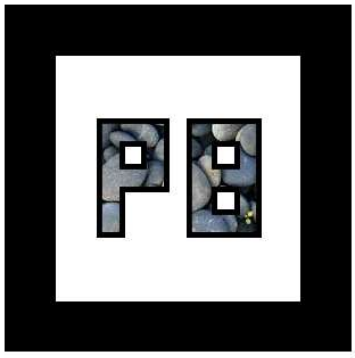


Marcador tienda Jack & Jones.

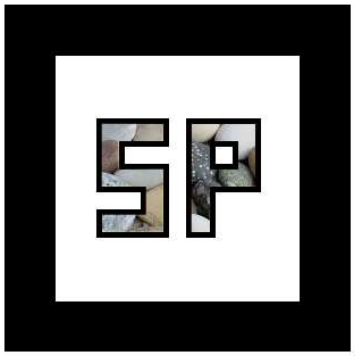


Marcador tienda Mango.

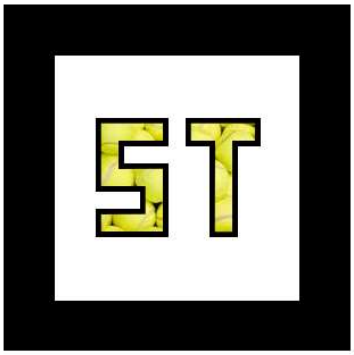
Marcador tienda Pimkie.



Marcador tienda Pull & Bear.



Marcador tienda Springfield.



Marcador tienda Stradivarius.

Marcador tienda Zara.

ANEXO 2

Video demostrativo.

En este anexo explicaremos lo que podemos observar en el video demostrativo que podemos descargar en el anexo 3.

Las primeras imágenes que nos aparecen están dedicadas al reconocimiento de marcadores, en ellas podemos apreciar como la aplicación detecta el marcador y aparecen objetos virtuales en 3D. Se comprueba las funciones de sus botones, un botón abre la página web oficial y el otro abre la aplicación de GoogleMaps con la localización de la tienda. También se muestra las funciones con las que cuenta el menú principal de cada tienda, así como los paneles de información que encontramos en la aplicación y las funciones del banner.

En las siguientes imágenes se comprueba el funcionamiento del GPS integrado en el dispositivo, entrando desde los dos extremos del pasaje del comercio. Seguidamente por una escena de la variación que sufre el dispositivo con respecto a latitud y longitud. A continuación, se muestra una incorporación a la aplicación referente a la geolocalización con suficiente margen para comprobar que se pueden realizar puntos de interés en coordenadas concretas siempre que tenga la distancia adecuada entre ellos.

Las últimas escenas pertenecen a la función de reconocimiento de texto, donde observamos el problema mencionado en el capítulo 4 sobre no tener el nombre de la tienda en el listado (por ejemplo, Pimkie) y el correcto funcionamiento de otras, en este caso la palabra de Springfield.

ANEXO 3

Enlace externo.

El video demostrativo y la aplicación se pueden obtener del siguiente enlace ya que ocupan más tamaño que el permitido para subirlo a ilias.

Enlace: https://drive.google.com/open?id=0B_qJPKgjxxXzWVFvU3U3b0NBd1E

BIBLIOGRAFÍA

[1] http://alejandraplicacionesmoviles.blogspot.com.es/2015/08/

[2] http://www.ppmmobile.com/single-post/2015/11/03/Historia-y-evoluci%C3%B3n-de-las-APPs-m%C3%B3viles

[3]http://noticias.universia.com.ar/ciencia-nn-tt/noticia/2014/10/03/1112557/caracteristicas-debe-tener-aplicacion-exitosa.html

[4] http://blog.aulaformativa.com/caracteristicas-gran-aplicacion-movil-posee/

[5]https://www.xatakandroid.com/aplicaciones-android/es-necesaria-la-tienda-de-apps-de-amazon-para-android

[6] https://hipertextual.com/archivo/2013/08/f-droid-la-alternativa-libre-a-google-play/

[7] https://es.slideshare.net/cobiruto/historia-de-las-aplicaciones-moviles

[8]https://www.lancetalent.com/blog/tipos-de-aplicaciones-moviles-ventajas-inconvenientes/

[9] https://wiboomedia.com/que-son-las-aplicaciones-web-ventajas-y-tipos-de-desarrollo-web/#tab-con-1

[10] http://www.raona.com/es/Solutions/Template/163/App-nativa-web-o-h%C3%ADbrida-

[11]http://searchdatacenter.techtarget.com/es/definicion/Desarrollo-de-aplicaciones-moviles-hibridas

[12] https://eduarea.wordpress.com/2017/04/29/que-es-la-realidad-mixta-mr-o-hibrida/

[13]http://www.tiumag.com/features/interviews/reactable-systems-rotor-sergi-jorda-entrevista/

[14] https://digger.mx/2017/virtualidad-aumentada-y-el-virtuality-continuum/

[15] https://www.openfuture.org/es/new/conoces-los-tipos-de-realidad-virtual-que-exi

[16] http://www.librorealidadaumentada.com/

[17]http://blogs.larepublica.pe/realidad-aumentada/2014/02/21/que-es-la-realidad-aumentada/

[18] http://blogthinkbig.com/realidad-aumentada-origen/

[19] https://elartedigital.wordpress.com/artistas/myron-krueger/

[20] http://aumentada.blogspot.com.es/p/wikitude.html

[21] https://sites.google.com/site/aumentadatema/niveles-de-la-realidad-aumentada

[22] http://larealidadaumentada.weebly.com/hardware-y-software.html

[23] http://www.vicomtech.org/t1/e6/realidad-aumentada

[24] https://www.noticias3d.com/noticia.asp?idnoticia=70310

[25] https://es.slideshare.net/marjorievherrera/qu-es-la-realidad-aumentada

[26]http://eleconomista.com.mx/tecnociencia/2016/07/21/10-aplicaciones-realidad-aumentada-que-no-son-pokemon-go

[27] http://www.20minutos.es/noticia/2644374/0/nueva-aplicacion-movil-realidad-aumentada-turismo-galicia-guia-para-visitantes/

[28]https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=1&qpcustomb=&qptimeframe=Y

[29] http://pacmac.es/que-es-y-para-que-sirve-ios/

[30] http://www.malavida.com/es/analisis/la-historia-de-android

[31]http://electronica.practicopedia.lainformacion.com/android/como-funciona-android-1576

[32] https://play.google.com/apps/publish/signup/

[33]https://winphonemetro.com/2013/11/windows-phone-el-sistema-operativo-que-mas-crece

[34] http://www.disca.upv.es/magustim/val/pfcs_anteriors/arToolkit/ARToolkit.html

[35] https://es.wikipedia.org/wiki/ARToolKit

[36]https://telos.fundaciontelefonica.com/url-direct/pdf-generator?tipoContenido=articulo&idContenido=2010090114230001

[37]http://www.pcactual.com/noticias/actualidad/vuforia-realidad-aumentada-segun-qualcomm-2_11555

[38] https://code.tutsplus.com/es/tutorials/introducing-augmented-reality-with-vuforia--cms-27160

[39] http://www.inglobetechnologies.com/en/new_products/arplayer/info.php

[40] http://www.t-immersion.com/about-us

[41]https://elandroidelibre.elespanol.com/2017/04/project-tango-analisis-lenovo-phab2-pro-realidad-aumentada.html

[42] https://en.wikipedia.org/wiki/Metaio

[43] https://www.layar.com/features/developers/

[44] https://www.layar.com/about/

[45] http://arcrowd.com/about/

[46] https://es.wikipedia.org/wiki/Unity_(motor_de_juego)

[47] https://blogs.unity3d.com/es/2016/06/16/evolution-of-our-products-and-pricing/

[48] http://docs.unity3d.com/es/current/Manual/index.html

[49]. https://developer.android.com/studio/intro/index.html

[50]. https://es.wikipedia.org/wiki/Android_Studio