Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Grado en Ingeniería en Tecnologías de Telecomunicación, especialidades en Sistemas Electrónicos

Director del TFG:

Dr. Rafael Cabeza

Mobile application based on augmented reality for events location.

Carlos Manero

Junio 2014

Dedicado a mis padres y hermano, por el esfuerzo que han realizado y que están realizando a lo largo de sus vidas para ofrecerme todas las posibilidades de las que he disfrutado y sigo disfrutando, así como la educación y valores que me han inculcado siempre, los cuales son responsables del esfuerzo y dedicación constante que se requieren para conseguir objetivos como este.

Agradecido a todos los profesores por la paciencia e implicación a lo largo de estos años, en especial a Dr. Rafael Cabeza, director del TFG.

"He fallado una y otra vez en mi vida, por eso he conseguido el éxito". Michael Jordan.

"El segundo es el primer perdedor" Ayrton Senna.

Resumen

Aplicación móvil que permite al usuario obtener información relativa al lugar, entrada, asiento, forma de llegar a un evento… a través de la información contenida en su entrada.

Se puede dividir el proyecto en 4 fases:

1. El usuario debe "escanear" la entrada por medio de la aplicación. Esta, realiza un reconocimiento de texto obteniendo los datos importantes que aparecen en la entrada.
   Para su implementación se necesita el uso de Unity3d, así como el SDK[1] de realidad aumentada de Vuforia Qualcomm y como lenguaje de programación se ha utilizado C Sharp (C#).

Por medio de una conexión con una base de datos, la aplicación compara y analiza si los datos son correctos, en cuyo caso permite al usuario continuar con las siguientes fases.

2. El lugar del evento pudiéndose visualizar como acercarse hasta él.
   Su implementación se realiza por medio de Eclipse IDE (Integrated Development Environment) utilizando como herramienta Google Maps v.3 API .
   Para el desarrollo de esta fase del proyecto se ha utilizado como lenguaje de programación JavaScript
3. La entrada del evento (implementando Realidad Aumentada con geolocalización).
   La implementación de esta tercera fase al igual que la fase anterior, se ha llevado a cabo por medio de Eclipse IDE, además de ello se ha utilizado Wikitude SDK, un SDK que permite la geolocalización.
   Como lenguaje de programación se ha utilizado JavaScript.

4. *Su lugar de asiento (fase condicional en función de los resultados obtenidos en las fases anteriores).*

La unión de las distintas fases se ha realizado por medio de Eclipse, utilizando como lenguaje de programación Java/Android y siendo necesario en todas las fases el SDK[1] de Android para conseguir la aplicación final.

[1] Para más información sobre los SDKs ver apartado 2.2 del documento

Palabras clave

Augmented Reality, realidad aumentada, geolocalización, geolocation, reconocimiento de texto, text recognition, aplicación, Vuforia, Wikitude, Android.

Índice

Introducción a la Realidad Aumentada, propósitos y objetivos del TFG ____ *10*

1.1 Realidad Aumentada ____ **10**

1.2 Propósitos y objetivos del TFG ____ **12**

1.3 Estado del arte ____ **13**

Tecnología utilizada ____ *15*

2.1 Programas ____ **15**

2.1.1 Unity3d ____ 15

2.1.2 Eclipse IDE ____ 16

2.2 Software development kit (SDK) ____ **17**

2.2.1 Android SDK ____ 17

2.2.2 Vuforia Qualcomm SDK ____ 17

2.2.3 Wikitude ____ 18

2.3 Application Programming Interface ____ **18**

2.3.1 Google Maps API ____ 18

2.4 Lenguajes de programación ____ **19**

2.4.1 C Sharp ____ 19

2.4.2 Java/Android ____ 19

2.4.3 JavaScript ____ 20

2.4.4 JSON ____ 20

2.4.5 HTML/CSS ____ 20

2.4.6 MySQL ____ 21

Descripción del proyecto ____ *22*

3.1 Fase 1 ____ **23**

3.1.1 Descripción ____ 23

3.1.2 Análisis ____ 24

3.2 Fase 2 ____ **24**

3.2.1 Descripción ____ 24

3.2.2 Análisis ____ 24

3.3 Fase 3 ____ **25**

3.3.1 Descripción ____ 25

3.3.2 Análisis ____ 25

Diseño del proyecto ____ *26*

4.1 Descripción de módulos y clases ____ **27**

4.1.1 Clases Fase 1 ____ 27

4.1.1.1 QCAR Behaviour ____ 28

4.1.1.2 Default Initialization Error Handle 28
4.1.1.3 Data Set Load Behaviour 28
4.1.1.4 Web Cam Behaviour 29
4.1.1.5 Keep Alive Behaviour 29
4.1.1.6 Scene View Manager 29
4.1.1.7 Text Reco App Manager 29
4.1.1.8 TextRecognitionUIEventHandler 29
4.1.1.9 Text Reco Behaviour 29
4.1.1.10 Text Event Handler 31
4.1.1.11 MySQLCS 32
4.1.1.12 Variables. 34
4.1.1.13 CSharpToJava 34
4.1.2 Clases Fase 2 35
4.1.2.1 ActivityFase2 36
4.1.2.2 DownloadTask 36
4.1.2.3 ParserTask 37
4.1.3 Clases Fase 3 37
4.2 Exportado de la Fase 1 a Eclipse IDE 39
4.3 Unión de las distintas fases 39
4.3.1 Fase 2 – Fase 3 39
4.3.2 Fase 1 – 2 39
4.4 Guía del flujo del programa 42
4.5 Metodología de desarrollo 43
4.6 Estudio de la interfaz del usuario 44
Resultados 48
5.1 Resultados Fase 1 49
5.2 Resultados Fase 2 49
5.3 Resultados Fase 3 50
5.4 Resultado Final 51
Conclusiones 52
Bibliografía 55
ANEXOS 57

Lista de Figuras

Fig. 1 Diferencia entre realidad aumentada y virtual 11
Fig. 2 Esquema general de fases de la app 12
Fig. 3 Logotipo Unity 16
Fig. 4 Qualcomm Vuforia Logotipo 17
Fig. 5 Google Maps API 19
Fig. 6 Diagrama de flujo de la Fase 1 23
Fig. 7 Lista de clases del paquete Vuforia 27
Fig. 8 Vista de los elementos envueltos en la creación y configuración del reconocimiento de palabras 30
Fig. 9 Adición de los archivos necesario para el reconocimiento de texto 30
Fig. 10 Diagrama de la implementación por niveles de la clase Draw() 32
Fig. 11 Estructura de la base de datos 33
Fig. 12 Interconexion entre clases TextEventHandler y MySQLCS 34
Fig. 13 Diagrama UML de las clases utilizadas en la fase 2 35
Fig. 14 Diagrama UML Fase 3 38
Fig. 15 Vista del nuevo contenido de la fase 1 tras su exportación 40
Fig. 16 Línea del tiempo de exportación de C# a Java 41
Fig. 17 Guía de las fases, clases, procesos y actividades del programa 42
Fig. 18 El "modelo cascada" sin modificar 43
Fig. 19 Ventana de información relativa al proyecto 44
Fig. 20 Ventana instrucción al usuario 45
Fig. 21 Lupa de rastreo 45
Fig. 22 Menu equipo detectado 46
Fig. 23 Menu de insertar puerta 46
Fig. 24 Mapa de guiado del usuario 47
Fig. 25 Interface de la fase 3 47
Fig. 26 Detección satisfactoria 49
Fig. 27 Mapa guiado al usuario desde su posición hasta el destino 50
Fig. 28 Entorno de realidad aumentada 50

1

Introducción a la Realidad Aumentada, propósitos y objetivos del TFG

Este TFG nace fruto del interés de la realización de una aplicación para plataforma móvil de realidad aumentada.

1.1 Realidad Aumentada

La realidad aumentada (AR en adelante) es un término utilizado para describir una visión del mundo real que queda aumentado por medio de la inclusión de elementos virtuales por medio de un dispositivo tecnológico.

Una definición concluyente acerca de la AR, es la que propone Furht en su libro, "*Augmented Reality (AR) refers to a live view of physical real world environment whose elements are merged with augmented computer-generated images, creating a mixed reality*". [1]

Conviene definir bien este concepto y diferenciarlo de la realidad virtual, cuyo fin es producir una apariencia de realidad en la que el usuario tenga la sensación de estar presente en ella.

Como se puede observar la diferencia es notoria, puesto que si en este último caso, la visión del mundo es de tipo virtual, en el caso de la realidad aumentada partimos de un mundo real en el que añadimos elementos virtuales.

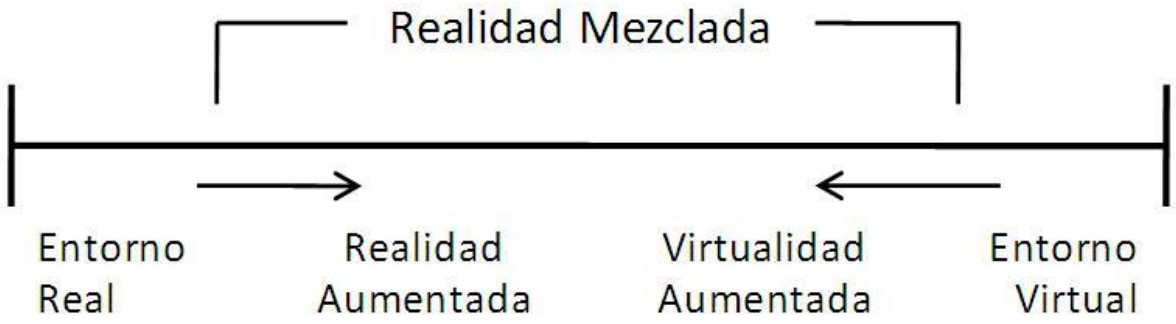


Fig. 1 Diferencia entre realidad aumentada y virtual

Pese a que ya en 1962 y 1973 empiezan las "semillas" de lo que serán más adelante los primeros proyectos de realidad virtual, no sería hasta el año 1990 cuando se acuña el termino Realidad Virtual, para que tan solo dos años más tarde, Tom Caudell creara el término de Realidad Aumentada.

Es decir, la Realidad Aumentada es una visión del mundo, relativamente joven ya que no es hasta mediados de la primera década del año 2000 cuando gracias a los dispositivos móviles, concretamente a los Smartphone, dispositivos con una gran cuota de mercado, cuando se empiezan a generar aplicaciones para los usuarios de estos dispositivos (sin obviar dispositivos móviles de otro tipo).

Pese a su relativa juventud, existen multitud de tecnologías, características, técnicas de visualización, dispositivos y herramientas para su implementación, desarrollo y uso.

Una vez realizado un estudio de las diferentes metodologías de la AR, se procedió a la definición de los detalles técnicos y el propósito de la aplicación a realizar.

1.2 Propósitos y objetivos del TFG

Fruto del interés en la AR, se barajan varias posibilidades, decidiendo finalmente realizarse una aplicación que implemente diferentes características y métodos, así como diferentes herramientas para su implementación, desarrollo y uso.

El proyecto de la aplicación tiene como objetivo en el guiado del usuario hasta el lugar en el que se produce un evento, por medio de la entrada del evento.

Así el usuario debe "escanear" la entrada del evento siguiendo las instrucciones de la aplicación. Ésta, una vez tenga los datos necesarios, guiará al usuario hasta el lugar del evento, utilizando la AR en la última parte del guiado para conseguir llevar al usuario hasta el lugar exacto.

Se puede diferenciar la aplicación en tres fases:

-Requerimiento de datos por parte de la aplicación

-Guiado del usuario por medio de mapa

-Guiado del usuario por medio de AR

La aplicación basa su funcionamiento en dos conceptos de realidad aumentada muy diferentes. Por un lado la primera parte de la aplicación centra su propósito en la detección de texto (no se trata de realidad aumentada como tal), mientras que la tercera fase del proyecto se centra en la incrustación de un objeto virtual en el mundo real (típico uso de la AR).

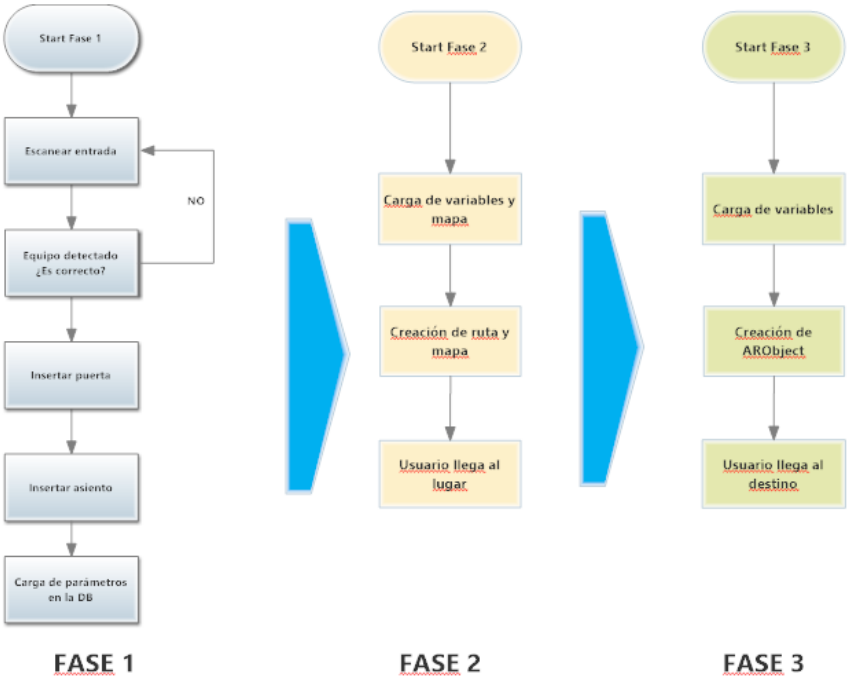


Fig. 2 Esquema general de fases de la app

1.3 Estado del arte

Como se cita en el apartado 1.1, pese a que la realidad aumentada es un concepto relativamente joven, en los últimos años su evolución y desarrollo permiten al desarrollador tener multitud de herramientas para la construcción y desarrollo de diferentes proyectos.

Hoy en día existen ya distintas tecnologías y características en cuanto a entornos de realidad aumentada se refiere.

En cuanto a tecnologías se refiere, existen complejos métodos de visión por computador tremendamente avanzados en su desarrollo.

Siguiendo con las tecnologías pero con una visión diferente, la realidad aumentada dispone de diversos tipos de display para su visualización que se pueden clasificar de diversas formas (según pantalla, según el rastreo que realizan…).

Aunque lo primero que debe definir un diseñador de AR es el tipo de sistema que va a ser implementado, es decir, interior o exterior, y fijo o móvil.

Como toda tecnología hoy en día, la realidad aumentada se somete a problemas de tipo social, donde como cualquier tecnología, se juzga si es algo útil o una mera distracción, aunque por el momento, parece que tiene una buena aceptación social.

Fruto de este uso social cada vez mayor, el futuro de la AR se presenta cuanto menos apasionante, pudiendo ser en el futuro una tecnología que ayude al ser humano en su día a día.

La AR como tal, puede depender o no del lugar, tiempo o espacio, pero lo que podemos catalogar en solo 2, es su ámbito de uso:

- ayudar a descubrir y entender nuestro entorno y aumentar la percepción de realidad.
- crear un entorno artificial distinto del que percibimos.

Este hecho de catalogar la AR según su ámbito de uso, nos puede llevar también a catalogar el tipo de entendimiento que se crea gracias a ella, que puede ser:

- aumentar con aumento de la percepción o entendimiento
- percepción de la asociación entre lo real y lo virtual
- asociación del comportamiento entre lo real y lo virtual
- sustitución de objetos reales por virtuales

La realidad aumentada no tendría sentido sin una correcta intrusión de los elementos virtuales en el mundo real, dicho esto, la AR tiene en cuenta cosas como la profundidad de percepción, la oclusión de objetos… técnicas básicas para que la escena creada parezca totalmente real.

Por último, reseñar que hoy en día el rastreo de objetos necesario para realizar cualquier entorno, posee de diversos métodos como son el rastreo de imágenes, la geolocalización, reconocimiento de texto…

2

Tecnología utilizada

En este capítulo se procede al estudio de las diferentes herramientas, programas, SDK y en definitiva, toda la tecnología que debe utilizar el proyecto.

2.1 Programas

A continuación se enuncian los programas utilizados para el desarrollo total del proyecto.

2.1.1 Unity3d

Unity es un motor de videojuegos multiplataforma creado por Unity Technologies.

Unity permite crear juegos para diferentes plataformas, de modo que utilizaremos esta característica de este programa para exportar la parte de la aplicación creada a la plataforma o

programa que nos interese, en nuestro caso, la primera fase del proyecto se desarrollará con este programa, junto con el SDK necesario para dar lugar a la primera fase del proyecto, el reconocimiento de texto en la entrada.

Los lenguajes de programación soportados por Unity3d son C#, UnityScript y BOO.

Además, la decisión principal para el uso de Unity3d, radica en la compatibilidad con el SDK de Qualcomm Vuforia, el cual nos permite una herramienta clave en la primera fase del proyecto, el reconocimiento de texto.

Fig. 3 Logotipo Unity

2.1.2 Eclipse IDE

Eclipse es un programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido". Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que está dentro del programa Eclipse.

Eclipse actualmente se encuentra desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios. [2]

La principal causa del uso de Eclipse en este proyecto, es que como se ha comentado se usa para realizar entornos de desarrollo, en particular para este proyecto se ha decidido utilizar Eclipse + ADT Plugin. Se trata de un pack recomendado para desarrolladores de Android, que incluye las herramientas necesarias para la creación de aplicaciones en esta plataforma.

El pack incluye además de Eclipse y el plugin ADT, el SDK[2] de Android, así como herramientas para Android. [3]

[2] Para más información sobre los SDKs ver apartado 2.2 del documento

2.2 Software development kit (SDK)

Un SDK (siglas en inglés de software development kit) es un conjunto de herramientas y programas de desarrollo de software que permite al desarrollador crear aplicaciones para una determinada plataforma, estructura de software, plataforma de hardware, sistema de computadora, consulta de videojuego, sistema operativo...

Las herramientas más comunes incluyen soporte para la detección de errores de programación, entornos de desarrollo integrado y otras utilidades.

Además, los SDK frecuentemente incluyen, también, códigos de ejemplo y notas técnicas de soporte u otra documentación de soporte para ayudar a clarificar ciertos puntos del material. [4]

2.2.1 Android SDK

El SDK de Android, incluye un conjunto de herramientas de desarrollo que comprende un depurador de código, una biblioteca, simuladores de teléfono, documentación, así como ejemplos de código y tutoriales.

Las plataformas de desarrollo soportadas incluyen Linux, Mac OS, y Windows.

La plataforma integral de desarrollo (IDE) soportada oficialmente es Eclipse junto con el complemento ADT (Android Development Tools plugin), aunque también puede utilizarse un editor de texto para escribir ficheros Java y Xml y utilizar comandos en un terminal (se necesitan los paquetes JDK, Java Development Kit) para crear y depurar aplicaciones. Además, pueden controlarse dispositivos Android que estén conectados directamente al ordenador de desarrollo, con los convenientes drivers instalados. [5]

2.2.2 Vuforia Qualcomm SDK

El SDK de Qualcomm Vuforia permite construir aplicaciones de realidad aumentada, y actualmente está disponible para Android, iOS, así como extensión de Unity3d.
Qualcomm Vuforia implementa entre sus herramientas el reconocimiento de texto desde mediados de 2013 en su versión 2.5 y posteriores. [6]
Esta herramienta es de vital importancia en nuestro caso, dado que la primera fase de la aplicación basa la recogida de datos en dicho reconocimiento.

Fig. 4 Qualcomm Vuforia Logotipo

2.2.3 Wikitude

El SDK Wikitude es una biblioteca de software para el desarrollo de aplicaciones móviles, utilizada para crear experiencias de realidad aumentada.

Wikitude es compatible con cualquier tipo de uso basado en geolocalización, así como los casos de uso que requieren el reconocimiento de imágenes y tecnología de seguimiento.

Al igual que otros SDKs nombrados anteriormente, Wikitude permite la creación de aplicaciones de realidad aumentada en diferentes plataformas móviles.

Lo que diferencia realmente a Wikitude como SDK, es que permite utilizar tecnologías web (HTML, JavaScript, CSS) para desarrollar aplicaciones de realidad aumentada. Esto es debido a que Wikitude no es un SDK nativo para la plataforma final en la que se desea ejecutarla, sino que la idea básica es llamar a la clase architectView en el proyecto y este controlará el ciclo de vida de todos los eventos que puedan ocurrir.[3]

2.3 Application Programming Interface

Una API (Interfaz de Programación de Aplicaciones) es un conjunto de funciones que permite al programador acceder a servicios de terceros e implementar dichos servicios en su propio desarrollo.

Gracias a las API, un desarrollador no necesita preocuparse de cómo funciona una aplicación remota ni de la forma en que las funciones fueron implementadas, para poder utilizarla en un programa. Una API puede estar disponible para un lenguaje específico o para diversos lenguajes de programación.

Versión 3 del API de JavaScript de Google Map.

2.3.1 Google Maps API

El API de JavaScript de Google Maps permite insertar Google Maps en diferentes aplicaciones. La versión 2 de esta API está especialmente diseñada aplicar tanto a móviles como a las aplicaciones de navegador de escritorio tradicionales.

El API proporciona diversas utilidades para manipular mapas y para añadir contenido al mapa mediante diversos servicios, permitiendo crear aplicaciones de mapas en aplicaciones.

La versión 2 del API de Google Maps se programa mediante Java/Android. [7]

[3] Para más información sobre clases y eventos de Wikitude, consultar la API: http://www.wikitude.com/developer/documentation

Fig. 5 Google Maps API

2.4 Lenguajes de programación

Para la realización del proyecto se han utilizado diversos lenguajes de programación en función de las herramientas y el programa utilizado.

2.4.1 C Sharp

C# es un lenguaje de programación cuya propiedad diferenciadora respecto a otros lenguajes de programación es su orientación a objetos.

Este lenguaje fue desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, el cual fue más tarde aprobado como un estándar.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque según sus desarrolladores, incluye mejoras derivadas de otros lenguajes.

Aunque C# forma parte de la plataforma .NET, ésta es una API, mientras que C# es un lenguaje de programación diseñado para generar programas independientes de dicha plataforma.

La principal ventaja de trabajar con este lenguaje es que existen compiladores que permiten generar programas para distintas plataformas como Windows, Unix, Android, iOS, Windows Phone, Mac OS y GNU/Linux. [8]

Utilizado en la primera fase del proyecto.

2.4.2 Java/Android

El lenguaje Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos y basado en clases. Su sintaxis deriva en gran medida al igual que C# de C y C++.

El objetivo del lenguaje de programación Java es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para ser utilizado en otra.

Es realmente en este proyecto ya que el desarrollo de programas para Android se hace habitualmente con el lenguaje de programación Java y el SDK que provee Google. [9] [10]

Utilizado en la segunda y tercera fase del proyecto.

2.4.3 JavaScript

JavaScript es un lenguaje de programación orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web.

Su uso en aplicaciones externas a la web, por ejemplo en aplicaciones de escritorio es también significativo, como es el caso de este proyecto.

JavaScript se diseñó con una sintaxis similar al C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo Java y JavaScript no están relacionados y tienen semánticas y propósitos diferentes.

Utilizado en la segunda y tercera fase del proyecto.

Para el uso en este proyecto, JavaScript interactúa como si se tratase de una página haciendo uso del Document Object Model (DOM). [11]

2.4.4 JSON

JSON (JavaScript Object Notation), es un lenguaje de formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

La simplicidad de JSON ha dado lugar a la generalización de su uso, especialmente como alternativa a XML en AJAX. Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos en este contexto es que es mucho más sencillo escribir un analizador sintáctico (parser) de JSON. En JavaScript, un texto JSON se puede analizar fácilmente, lo cual ha sido fundamental para que JSON haya sido aceptado por parte de la comunidad de desarrolladores AJAX, debido a la ubicuidad de JavaScript en casi cualquier navegador web.

Aunque su uso es tremendamente reducido, es de gran importancia en este proyecto, ya que permite el traspaso de datos entre el lenguaje Java/Android y el lenguaje JavaScript para la creación de la tercera fase, en el mundo de realidad aumentada que deseamos crear. [12]

2.4.5 HTML/CSS

HTML, siglas de HyperText Markup Language, hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que, en sus diferentes versiones, define una estructura básica y un código para la definición de contenido de una página web, como texto, imágenes, etc.

CSS es un lenguaje de estilo que define la presentación de los documentos HTML. Por ejemplo, CSS abarca cuestiones relativas a fuentes, colores, márgenes, líneas, altura, anchura, imágenes de fondo…

Es decir, HTML se usa para estructurar el contenido mientras que CSS se usa para formatear el contenido previamente estructurado.

2.4.6 MySQL

El lenguaje de consulta estructurado o SQL (por sus siglas en inglés Structured Query Language) es un lenguaje utilizado para interactuar con bases de datos, consiguiendo realizar diversos tipos de operaciones en ellas.

En especial para este proyecto, se ha utilizado MySQL, como sistema de gestión de bases de datos. MySQL es un software libre bajo licencia GNU GPL.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. MySQL es usado por muchos sitios web grandes y populares, como Wikipedia, Google (aunque no para búsquedas), Facebook, Twitter, Flickr, YouTube… [12]

3

Descripción del proyecto

A la vista de los objetivos a conseguir, se procede a la descripción del proyecto, así como el estudio de las herramientas necesarias y que facilitarían lo máximo posible el desarrollo del proyecto.

Para conseguir facilitar al lector el entendimiento de este capítulo, esto es, la descripción del proyecto (así como elección de las múltiples herramientas utilizadas en él y su finalidad), se dividirá este capítulo al igual que el proyecto en fases.

Se ha decidido dividir el proyecto en tres fases dada la relativa diferencia entre las tecnologías y objetivos que tiene cada una de ellas.

3.1 Fase 1

3.1.1 Descripción

Se define la fase 1 como la fase inicial del proyecto, dicha fase es ejecutada al inicio de la aplicación y es la encargada de la recogida de datos por parte del usuario.

Para conseguir los datos necesarios se pedirá al usuario que “escanee” la entrada del evento (equipo local en el ejemplo de aplicación creada) y además se pedirá al usuario que introduzca otros datos que las herramientas utilizadas no pudieran conseguir con el “escaneo” (número de puerta y asiento en la aplicación creada).

Los datos introducidos por el usuario se validarán en una base de datos y se obtendrán de ella los datos necesarios para continuar con las siguientes fases.

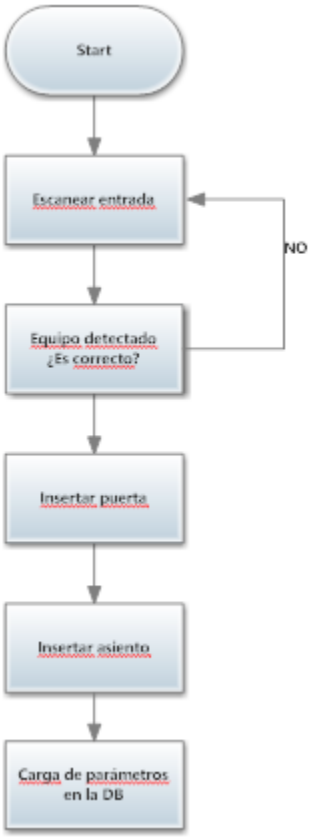


Fig. 6 Diagrama de flujo de la Fase 1

Estos datos serán las coordenadas de la información necesaria para llegar al lugar del evento.

La fase 2 será cargada tras la carga y creación de los datos que se comentan.

3.1.2 Análisis

En esta fase se debe conseguir interactuar con el usuario, pidiendo a este que inserte unos parámetros, bien a través de la cámara o bien a través del teclado.

Conseguir un parámetro por teclado puede ser hoy en día implementado por multitud de herramientas, pero para el "escaneo" de un parámetro y su reconocimiento como texto, se opta por utilizar el SDK de realidad aumentada de Qualcomm Vuforia.

El uso de este SDK unido a la necesidad de crear una interfaz de usuario, hace que optemos por Unity3d como programa en el que desarrollar esta primera fase y por lo tanto, C# como lenguaje de programación.

Una vez recogido los parámetros necesarios que deberá aportar el usuario, la aplicación deberá realizar consultas sobre estos parámetros, por lo que parece sensato pensar que será necesario el uso y conexión con una base de datos que acepte permisos de acceso remoto.

3.2 Fase 2

3.2.1 Descripción

La segunda fase del proyecto es aquella en la cual mediante los datos introducidos por el usuario en la fase anterior y los datos consultados en la base de datos, se guía al usuario mediante un mapa desde su ubicación actual hasta las proximidades de la ubicación del evento.

3.2.2 Análisis

Dado que la principal característica de esta segunda fase es el guiado por el mapa, resulta necesario buscar una herramienta de trazado de mapas y rutas.

Google implementa hoy en día numerosas APIs y servicios en función de las necesidades de los desarrolladores. Una de ellas es Google Maps[4], en concreto la versión 2 de JavaScript.

Esta versión de la API, se programa íntegramente en Java-Android.

Por tanto, dado que la idea del proyecto es particularizar la aplicación móvil para la plataforma Android y viendo los lenguajes de programación que debe utilizar esta fase, se decanta por Eclipse IDE para su desarrollo.

[4] Para más información acerca de Google Maps API:
https://developers.google.com/maps/documentation/javascript/?hl=es

3.3 Fase 3

3.3.1 Descripción

La fase 3 es aquella encargada de señalar al usuario el lugar de entrada al evento.

Esta acción se desea hacer mediante la aparición en el campo visual del usuario de una flecha,

3.3.2 Análisis

Dada la descripción de la fase 3, esta fase debe basar su acción en la cámara del móvil del usuario y AR.

Esta escena de AR debe basarse en la geolocalización del usuario por lo que se debe utilizar un SDK de AR compatible con esta característica, como es Wikitude.

Wikitude es un SDK de AR en el cual el desarrollador utiliza el lenguaje de JavaScript para su uso, aunque será necesario a su vez Java/Android y JSON.

Para la implementación de Wikitude en el proyecto, (teniendo en cuenta al igual que en la fase 2 que la idea del proyecto es particularizar la aplicación móvil para la plataforma Android), se opta por el uso de Eclipse IDE como herramienta de desarrollo.

4

Diseño del proyecto

En este apartado se describen los módulos y clases que han sido necesario para conseguir los objetivos y cumplir los requerimientos previstos anteriormente.

4.1 Descripción de módulos y clases

Se procede a la descripción de las clases y módulos utilizados en las diferentes fases de la aplicación.

> *En el Anexo I se adjunta el código que ha sido necesario editar para implementar la Fase 1 en la aplicación*

4.1.1 Clases Fase 1

En esta primera fase, una vez analizada se llego a la conclusión que se necesitaba el uso del SDK de Qualcomm Vuforia para el reconocimiento de texto. Así, es importante describir y entender la multitude de clases que nos aporta dicho SDK, en el que todas las clases son descendientes de MonoBehaviour dado que se trata de la clase base.

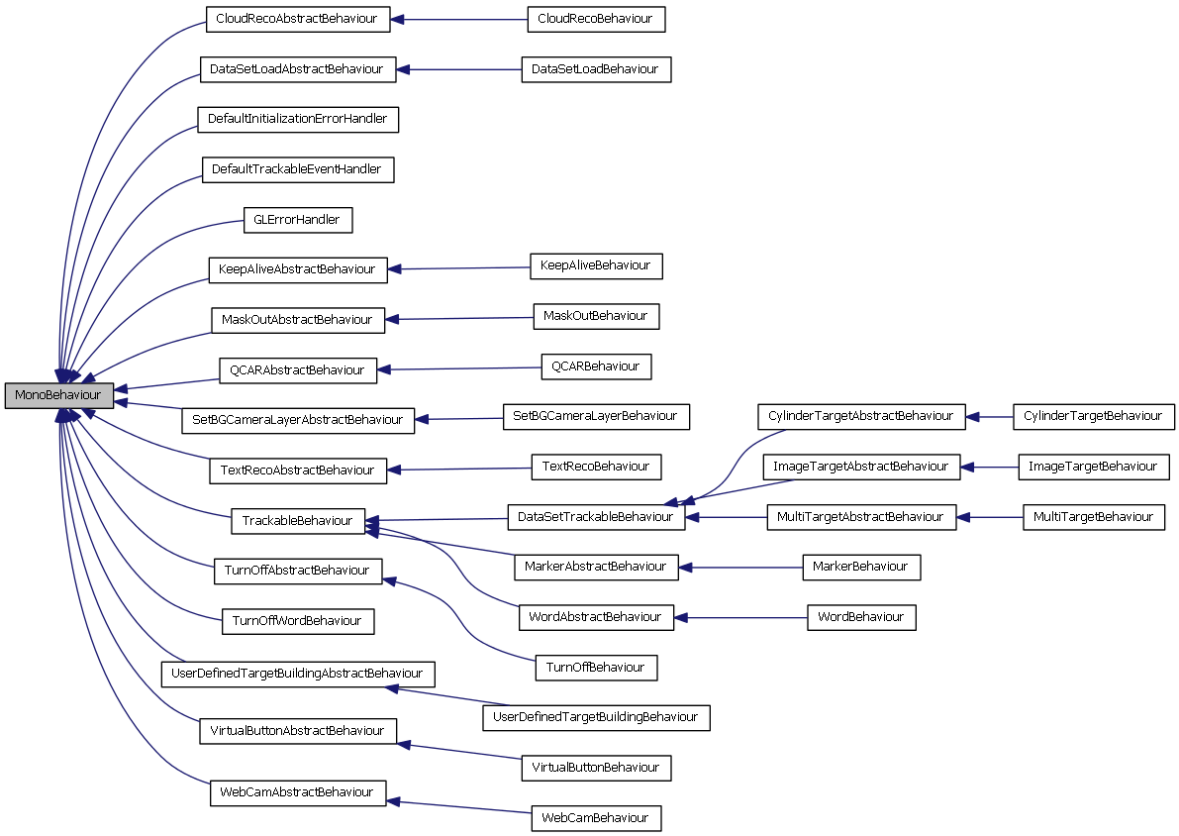


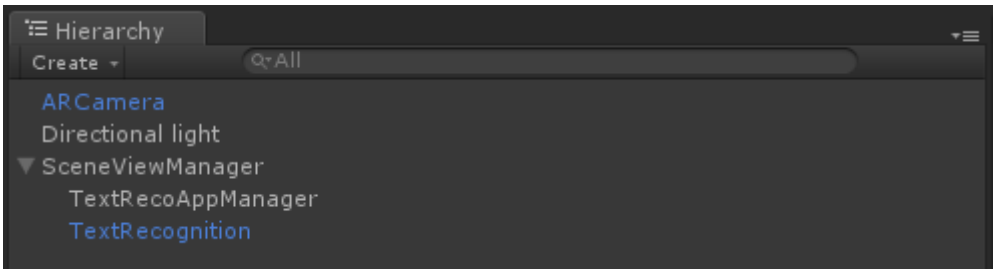
Fig. 7 Lista de clases del paquete Vuforia

Como se decide trabajar en Unity, se procede a crear 2 objetos principales de interés (en la imagen de la jerarquía de objetos se pueden observar 3, aunque el objeto “Directional light” no contiene información relevante para este estudio).

Así pues, se tiene por un lado el objeto ARCamera el cual contendrá varias clases relativas al funcionamiento de la cámara del móvil y el objeto SceneViewManager, que contendrá 2 objetos asociados a él.

Diferenciamos así pues los siguientes objetos:

- **ARCamera**
  - QCAR BEHAVIOUR
  - Default Initialization Error Handle
  - Data Set Load Behaviour
  - Web Cam Behaviour
  - Keep Alive Behaviour
- **SceneViewManager**
  - SceneViewManager
  - **TextRecoAppManager**
    - TextRecoAppManager
    - TextRecognitionUIEventHandler
  - **TextRecognition**
    - TextRecoBehaviour
    - TextEventHandler



4.1.1.1 QCAR Behaviour

Maneja el seguimiento y desencadena las acciones a realizar con el fondo de vídeo. La clase actualiza todos los trackables de la escena.

4.1.1.2 Default Initialization Error Handle

Un manejador de eventos propio que implementa la interfaz IQCARErrorHandler, para manejar los posibles errores que puedan ocurrir en el ámbito relativo a la cámara.

4.1.1.3 Data Set Load Behaviour

Esta clase permite cargar automáticamente y activar uno o más DataSet en el arranque.

4.1.1.4 *Web Cam Behaviour*

Esta clase gestiona el uso de una cámara web para el modo de reproducción en Windows o Mac.

4.1.1.5 *Keep Alive Behaviour*

La clase KeepAliveBehaviour permite que los objetos Vuforia puedan ser reutilizados a través de múltiples escenas. Esto hace que sea posible compartir conjuntos de datos y objetivos entre escenas.

4.1.1.6 *Scene View Manager*

Clase propia creada para la dirección de la aplicación, de esta forma se controla que clases relativas a los eventos de texto utilizar y que clases relativas a los eventos del usuario.

4.1.1.7 *Text Reco App Manager*

Esta clase se encarga de ejecutar la inicialización y actualización de las convocatorias de TextEventHandler como parte del mismo hilo. Extiende de AppManager.

4.1.1.8 *TextRecognitionUIEventHandler*

La clase Text Recognition UI Event Handler, implementa otra clase propia llamada ISampleAppUIEventHandler.

Por medio de esta clase, se puede controlar los eventos generados por la interfaz de usuario.

Disponemos de dos tipos de acciones:

- Con un tap, se abre el menú de opciones (flash, acerca de….)
- Con doble tap, se hace realiza la acción de focus de la imagen visualizada en pantalla por medio de la cámara.

4.1.1.9 *Text Reco Behaviour*

Esta es la clase principal de comportamiento que encapsula el reconocimiento de texto. Sólo se debe añadir a una escena y se inicializará el rastreador de texto con la lista de palabras configurada. Eventos por palabras recientemente reconocidos o perdidos serán llamados a través de la interface implementada ITextRecoEventHandlers.

Es de vital importancia tener en cuenta como se debe configurar esta clase para el reconocimiento de texto. En nuestro caso, es necesario configurar de la siguiente forma la clase[5]:

1. Adición de una “Word List”. La clase Text Reco Behaviour necesita de un fichero de tipo “.vwl”. El SDK nos provee en su API de un archivo de este tipo denominado “Vuforia-English-word”, además de proveer un archivo de texto plano para ver las palabras contenidas en dicho fichero.

[5] Para más información y clarificación de conceptos, visitar la API de Vuforia para este apartado: https://developer.vuforia.com/resources/dev-guide/word-targets

Como es de esperar, actualmente, esta lista de palabras es un diccionario totalmente en ingles.

2. Dado que solo disponemos de un diccionario en ingles, y queremos utilizar nombres de lugares o eventos (por ejemplo, “OSASUNA”), debemos utilizar lo que Vuforia llama, “Aditional Word File”. Un archivo de tipo “.lst” que debemos editar y en el que debemos introducir las palabras que deseamos añadir al diccionario.
3. Por último, Vuforia permite implementar archivos para utilizar como filtros (“Filter List File”). Dado que solo nos interesa reconocer las palabras almacenadas en la base de datos, creamos un filtro en modo “WHITE_LIST”, es decir, rechazamos todas las palabras menos las contenidas en el “Filter List File”.

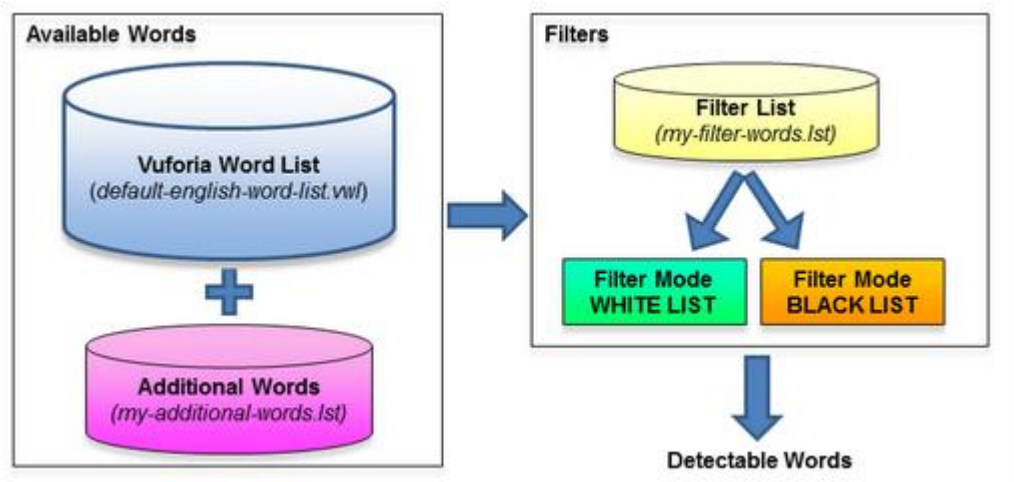


Fig. 8 Vista de los elementos envueltos en la creación y configuración del reconocimiento de palabras

Para una última aclaración de los conceptos anteriores, podemos visualizar la siguiente imagen, donde se ve la clase y como se ven añadidos los elementos comentados, además de visualizarlos en el explorador.

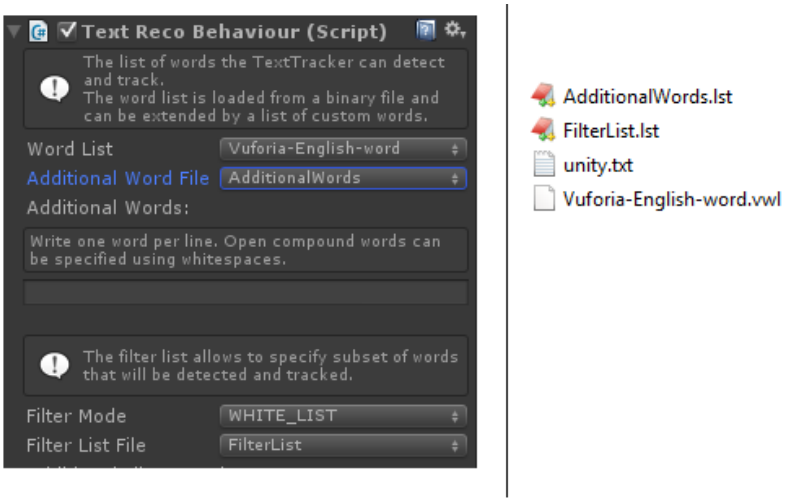


Fig. 9 Adición de los archivos necesario para el reconocimiento de texto

4.1.1.10 Text Event Handler

Text Event Handler es la clase propia más importante, dado que en ella se utiliza un manejador de evento de textos. Para ello implementa ITextRecoEventHandler, IVideoBackgroundEventHandler

ITextRecoEventHandler: es una interfaz para el manejo de resultados de palabras, tanto detectados, como perdidos.

IVideoBackgroundEventHandler: es una interfaz para la gestión de eventos en relación con el fondo de vídeo.

Metodos implementados: a continuación se describe de forma detallada los métodos más importantes a la hora de procesar los resultados del reconocimiento de texto.

- **Draw()**: este método es llamado continuamente y en el se recoge la estructura jerarquica que maneja en que situación se encuentra la Fase 1. Para estructurar de forma concisa la Fase 1.
- **Nivel1()**: es el método que engloba la primera parte de la Fase 1, aquella en la que se procesa los datos obtenidos en el reconocimiento de texto.
- **Nivel2()**:es el método utilizado en la segunda parte de la Fase 1, en ella se procesa los datos relativos a la puerta de entrada introducidos por el usuario.
- Nivel3(): este método es util en la tercera parte de la Fase 1, en la que se procesa los datos relativos al asiento del usuario, dato introducido por el propio usuario.
- **OnWordDetected()**: este método es llamado al producirse un evento del tipo reconocimiento de texto y retorna como argumento un resultado de palabra reconocida, a partir de la cual trabajaremos y almacenaremos.
- **OnWordLost()**: este método es llamado al producirse la perdida de una palabra detectada, retornando como argumento el resultado de la palabra perdida.

Como se ha comentado en las líneas anteriores, para la creación de la estructura de esta primera fase del programa se pensó en una jerarquía por niveles y subniveles, de forma que en la función Draw() se implementa un código que jerarquiza por niveles el desarrollo de la clase y por consiguiente de la interfaz de usuario.

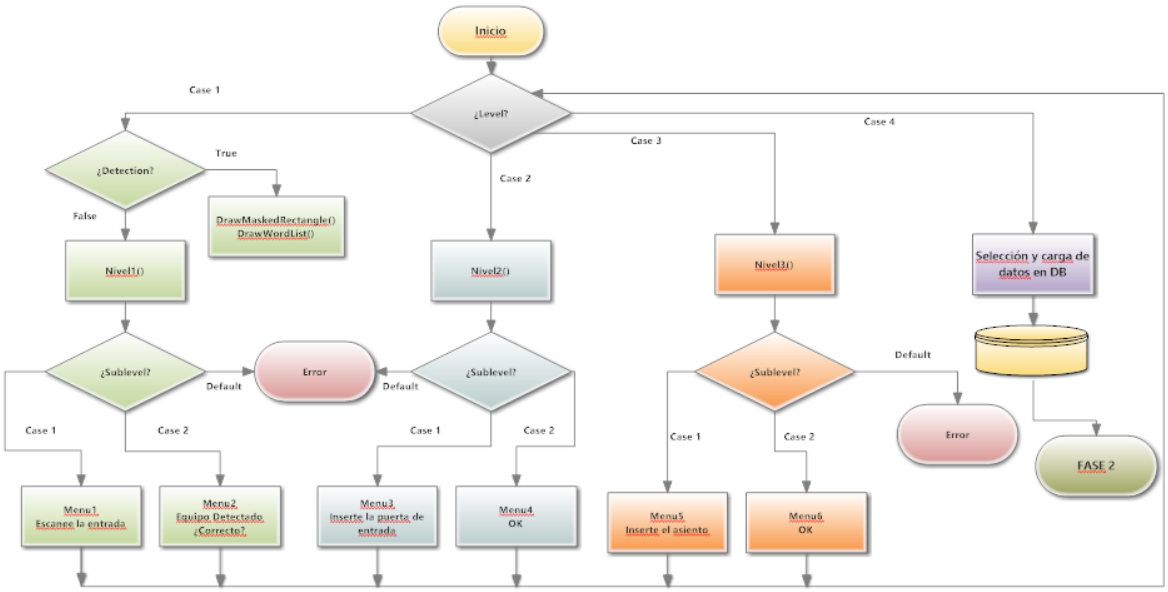


Fig. 10 Diagrama de la implementación por niveles de la clase Draw()

Nota Importante: algunos de los scripts utilizados son propiedad de Qualcomm Vuforia. Copyright (c) 2012-2014 Qualcomm Connected Experiences.

4.1.1.11 MySQLCS

Como se puede observar, el último paso que realiza el código antes de la carga de la siguiente fase es la consulta con la DB (data base).

Dicha base de datos se encuentra alojada en un servidor remoto, que permite conexiones de tipo remoto.

Para conseguir la conexión con la base de datos ha sido necesario además de crear esta clase, las librerías:

- MySql.Data
- MySql.Data.MySqlClient

Además, la estructura de la base de datos es la siguiente:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
1	id	int(11)			No	*Ninguna*	AUTO_INCREMENT
2	name	varchar(100)	latin1_spanish_ci		No	*Ninguna*	
3	direccion	varchar(300)	latin1_spanish_ci		No	*Ninguna*	
4	latitude	float			No	*Ninguna*	
5	longitude	float			No	*Ninguna*	
6	puerta0	varchar(50)	latin1_spanish_ci		No	*Ninguna*	
7	puerta1	varchar(50)	latin1_spanish_ci		No	*Ninguna*	
8	puerta2	varchar(50)	latin1_spanish_ci		No	*Ninguna*	
9	puerta3	varchar(50)	latin1_spanish_ci		No	*Ninguna*	
10	puerta4	varchar(50)	latin1_spanish_ci		No	*Ninguna*	
11	puerta5	varchar(50)	latin1_spanish_ci		No	*Ninguna*	
12	puerta6	varchar(50)	latin1_spanish_ci		No	*Ninguna*	
13	puerta7	varchar(50)	latin1_spanish_ci		No	*Ninguna*	
14	puerta8	varchar(50)	latin1_spanish_ci		No	*Ninguna*	
15	puerta9	varchar(50)	latin1_spanish_ci		No	*Ninguna*	

Fig. 11 Estructura de la base de datos

Como se puede observar, la base de datos consta de los siguientes campos:

- id: es el identificativo de cada lugar, de tipo Integer y autonumérico.
- name: nombre del estadio. Este campo es el buscado mediante la consulta SQL
- dirección: identifica el lugar por medio de un String.
- latitude: latitud del lugar
- longitud: longitud del lugar
- puertaX: es un campo de tipo varchar que contiene tanto la latitud como la longitud y que guardará strings del tipo “12.34/56.78”, donde el carácter “/” separa la latitud de la longitud.

A la vista de la estructura creada en la base de datos para alojar los datos necesarios, se procedió a la creación de la clase utilizada para la consulta y selección

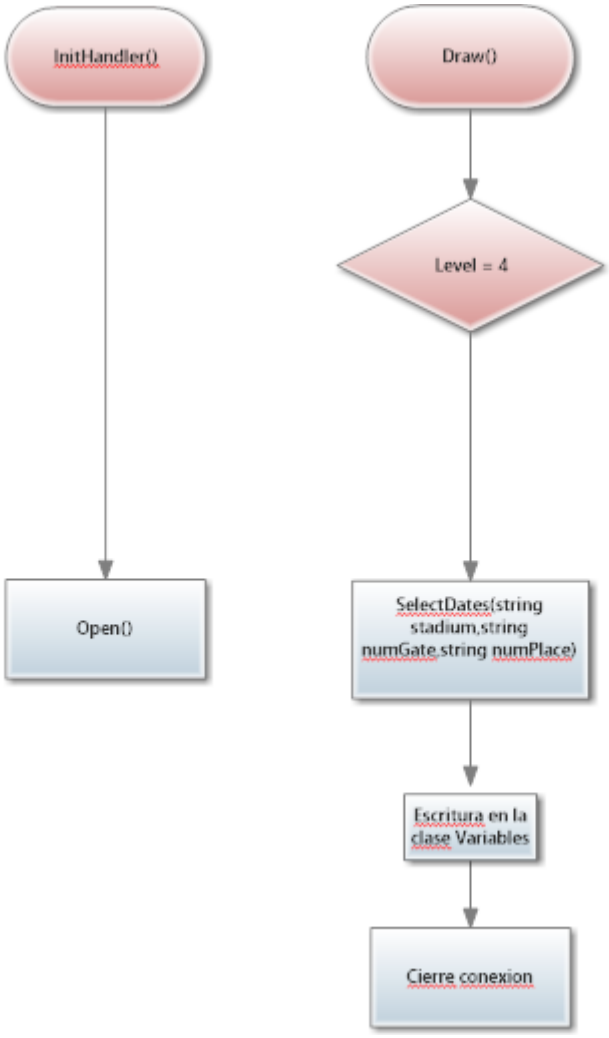


Fig. 12 Interconexion entre clases TextEventHandler y MySQLCS

4.1.1.12 Variables.

En útlima instancia, una vez están introducidos los datos por el usuario y seleccionados los datos asociados en la base de datos, se guardan todos los parámetros obtenidos en la consulta con la SQL, en esta clase.

Simplemente, la clase esta compuesta por unos atributos de tipo “static” que guardan los datos para poder tenerlos disponibles en cualquier momento que se desee.

4.1.1.13 CSharpToJava

A diferencia de las otras clases, esta clase presenta un funcionamiento distinto, puesto que como su propio nombre indica, no realiza acciones para seguir desarrollando la Fase 1 del proyecto sino que esta clase, ha sido desarrollada para pasar los datos que se encuentran en la clase “Variables”,

a la siguiente fase del proyecto, es decir, a la clase “Main Activity”, en concreto al método “estadioEncontradoJ” del paquete com.TFG.ParteIni.

Resulta ser una clase compleja para su comprensión, y dado que extiende de Unity3d y CSharp, se explica detalladamente su funcionamiento en el apartado “4.3 Unión de las distintas fases”, en este mismo capítulo.

4.1.2 Clases Fase 2

En esta segunda clase, se utiliza la API de Google Maps. Dicha API utiliza el lenguaje de Java/Android para configurar su uso.

Para ello utilizamos diversas funciones y objetos para hacer posible la geolocalización.

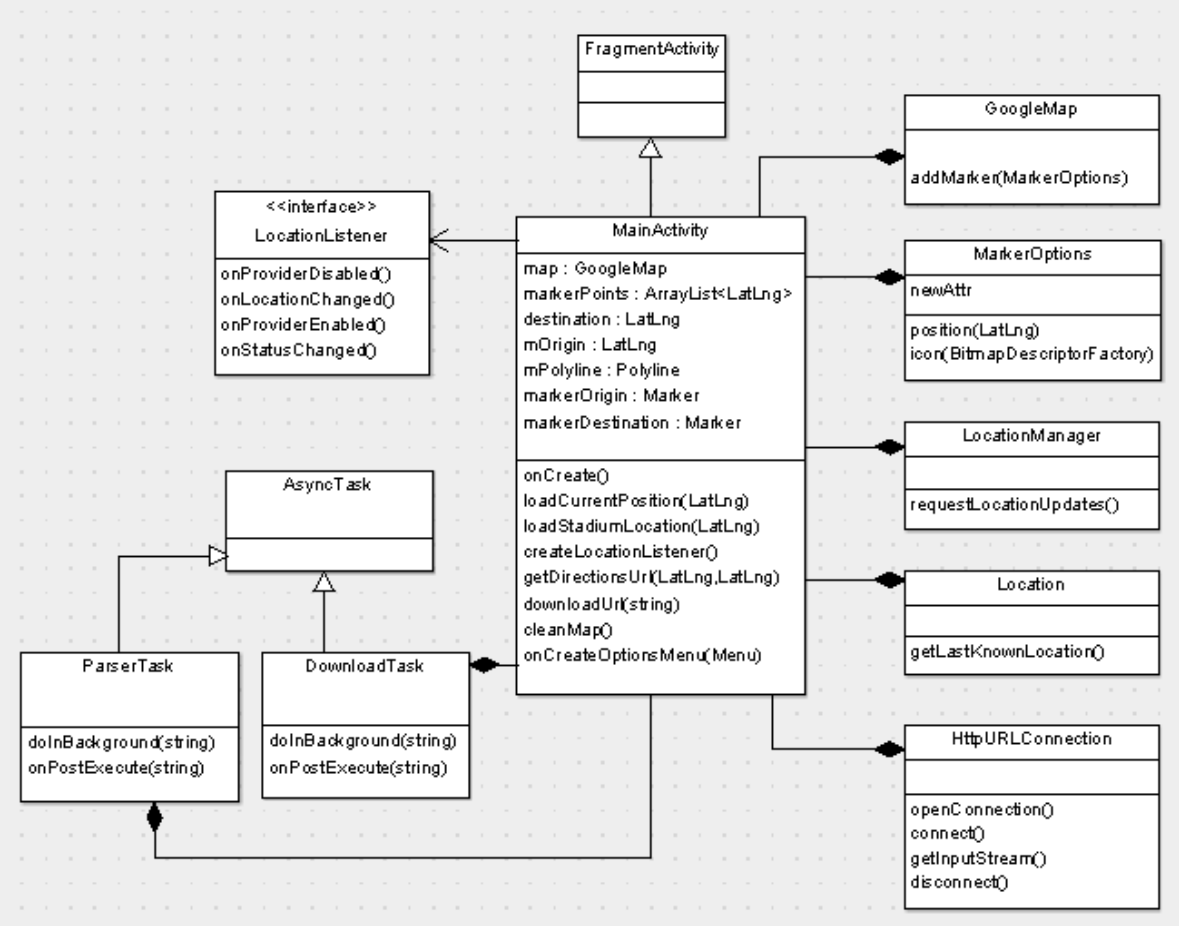


Fig. 13 Diagrama UML de las clases utilizadas en la fase 2

4.1.2.1 *ActivityFase2*

La clase ActivityFase2 es la clase principal de la fase 2. En ella se declarán las numerosas variables y métodos necesarios para llevar a cabo la geolocalización y guiado. Esta clase extiende de FragmentActivity e implementa la interface LocationListener.

Los métodos utilizados son los siguientes:

- **onCreate**: este método es ejecutado al llamar a la clase. En el creamos las vistas de contenido (capa Android), creamos los marcadores de posición (pero no inicializados) y obtenemos soporte para el mapa gracias a la clase SupportMapFragment.
  Además, añadimos manejadores de eventos de mapa.
- **loadCurrentPosition**: en este método definimos las opciones existentes para el marcador que ubica la posición actual del usuario y llamamos por último a la función loadStadiumLocation(LatLng)
- **loadStadiumLocation**: en este método, al igual que loadStadiumLocation, definimos las opciones existentes para el marcador que ubica la posición destino del usuario y se realiza una acción vital para llevar a cabo la finalidad de esta fase, gracias a la clase DownloadTask, concretamente al método "execute(string)" descargamos los datos de tipo json desde la API de Google Directions.
- **createLocationListener**: este método maneja los cambios de posición del usuario, llamando a onLocationChanged() en caso de que se haya producido dicho cambio de posición.
- **getDirectionsUrl**: por medio de esta función, creamos el string que contiene a modo de url toda la información necesaria para la consulta a la API de Google Directions.
- **downloadUrl**: obtiene y descarga los datos de tipo json de la url introducida como parámetro. En esta función se lleva a cabo una conexión de regida por el protocolo HTTP.

4.1.2.2 *DownloadTask*

DownloadTask es una clase que hereda de AsyncTask, pudiendo utilizar así los siguientes métodos heredados:

- **doInBackground**: método utilizado para realizar acciones en un hilo asíncrono de la clase ActivityFase2, concretamente realiza tareas de descarga de datos.
- **onPostExecute**: método ejecutado tras finalizar el método doInBackground, y que llama a la clase ParserTask, por metido del método "execute".

4.1.2.3 ParserTask

Al igual que DownloadTask, la clase ParserTask hereda de AsyncTask, pudiendo utilizar así los siguientes métodos heredados:

- **doInBackground**: método utilizado para realizar acciones en un hilo asíncrono de la clase ActivityFase2, concretamente realiza tareas de análisis de los datos de tipo JSON.
- **onPostExecute**: envía los datos JSON procesados en el hilo principal del programa, es decir, en la clase ActivityFase2.

En el Anexo II se adjunta el código que ha sido necesario para crear esta fase

---

4.1.3 Clases Fase 3

En esta última etapa, se utiliza el SDK de Wikitude para señalar al usuario por medio de geolocalización, la entrada del evento.

El SDK de Wikitude se apoya en el lenguaje de programación de JavaScript para su implementación.

Para ello basta con crear un ARObjet a partir de unas clases que incluya las características deseadas para este objeto y así, llevar a cabo la acción de mostrar al usuario una flecha en el lugar concreto. Esta flecha será nuestro ARObject.

Las características utilizadas son las siguientes:

- **AR.Geolocation**: toda instancia de Geolocation representa una localización en el espacio en tres dimensiones. Consiste en una longitud, una latitud, y una altitud opcional. El sistema de coordenadas usado es el WGS 84.

- **AR.RelativeLocation**: la localización relativa describe una localización relativa a otra localización, que en nuestro caso será relativa a la localización introducida en la Geolocalización. Es especificada utilizando el norte y sur desde la localización relativa a utilizar.
- **AR.ImageResource**: encapsula una imagen.
- **AR.ImageDrawable**: encapsula la imagen que va a ser utilizada para la representación del GeoObjecto. Se puede dimensionar el objeto para mejorar la visualización.

Además, conviene recalcar las siguientes propiedades:

- **ARObject.drawables.cam**: esta propiedad es utilizada para añadir el drawable que representará al ARObject en la vista de la cámara.
- **ARObject.enabled**: esta propiedad activa el ARObject.

- **ARObject.drawables.addIndicatorDrawable**: añade un drawable que ayuda en la búsqueda del ARObject cuando este no esta se está visualizando.

Todas las clases y métodos utilizados se pueden visualizar en el siguiente diagrama UML.

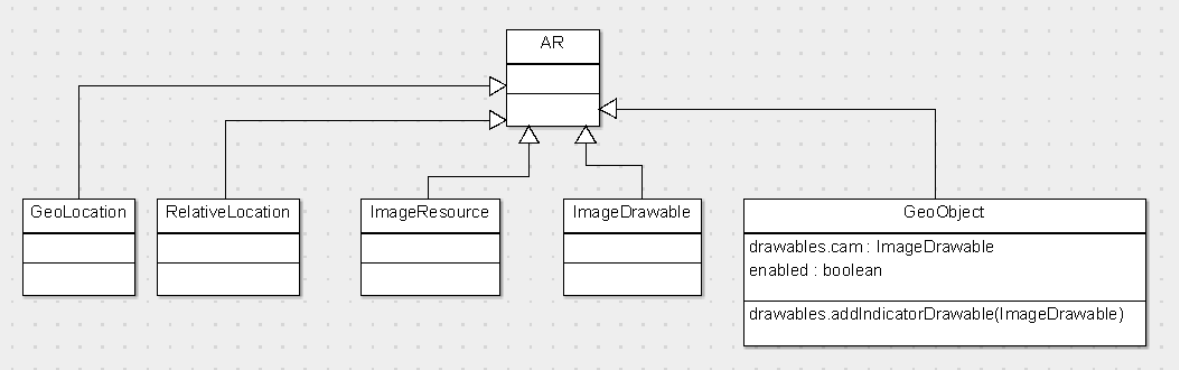


Fig. 14 Diagrama UML Fase 3

En el Anexo III se adjunta el código que ha sido necesario para crear el Objeto de AR

Los parámetros necesarios para crear este mundo de realidad aumentada en la Fase 3, son recibidos en Java/Android desde la Fase 2. Como acabamos de comentar, esta última fase esta realiza el mundo de realidad aumentada por medio de JavaScript, así, aparece el problema de intercambio de parámetros entre un lenguaje y otro.

Para ello se utiliza el lenguaje de programación JSON, de forma que se hace posible pasar datos entre la clase java "SampleCamActivity" y el código JavaScript "geo.js" utilizado.

Toda esta transferencia de parámetros se realiza mediante el método "onPostCreate" de la clase "SampleCamActivity" que a su vez, resulta ser un método heredado de la clase abstracta "`AbstractArchitectCamActivity`".

En el anexo III se puede encontrar tanto el código utilizado para el envio de estos parámetros, como el utilizado para la obtención de estos parámetros en el código JavaScript "geo.js".

4.2 Exportado de la Fase 1 a Eclipse IDE

Como hemos comentado en diversos capítulos y apartados, la Fase 1 del proyecto diverge del resto de fases en lo que respecta al programa utilizado para su desarrollo.

Así, una vez realizado la Fase 1, se decidió su exportación a Eclipse IDE para su posterior uso en el resto de fases.

A pesar de comentarse en este apartado, se ha decido no concretar los pasos seguidos, ya que basta con seguir los pasos descritos por Vuforia en su guía para desarroladores [13].

> *En el Anexo IV se detalla alguno de los pasos para una información a futuros desarrolladores más concisa y clara.*

4.3 Unión de las distintas fases

Para la unión de las distintas fases se ha utilizado diversas técnicas, en función del programa y lenguaje utilizado en cada fase.

4.3.1 Fase 2 – Fase 3

En concreto, para la unión de la fase 2 y la fase 3, partíamos de lenguajes de programación iguales y mismo entorno de desarrollo, por tanto, como era de esperar, no se tuvo problemas para la unión.

Simplemente, se utilizaron objetos de la librería Android de tipo “Intent”. Estos objetos son muy útiles a la hora de lanzar y unir lo que Android llama “Activity”.

Además, este tipo de objetos, posee atributos de tipo “extras”, con los que transferir datos de una “Activity” a otra con tal solo utilizar los métodos “putExtra” y “getExtra”.

4.3.2 Fase 1 – 2

Una de las mayores dificultades del proyecto radica en esta unión. A pesar de parecer tremendamente sencilla, ya que al igual que en la unión de la fase 2 con 3 basta con utilizar objetos de tipo Intent, se debe recordar, que la fase 1 estaba completamente programada en el lenguaje de programación CSharp por medio de Unity3d y añadiendo el SDK de Qualcomm Vuforia.

Es decir, que al realizar la exportación a Eclipse (ver apartado 4.2), toda clase, método o propiedad conocida, desaparecía, generándose un nuevo proyecto en Java, que a pesar de un funcionamiento idéntico, nada tenia que ver en cuanto a la programación realizada, pasando de una tener unas clases claras y concisas, a tener una única clase, sin contenido alguno en su interior, "MainActivity" del paquete com.TFG.ParteIni y una larga lista de "Assets" indescriptibles.

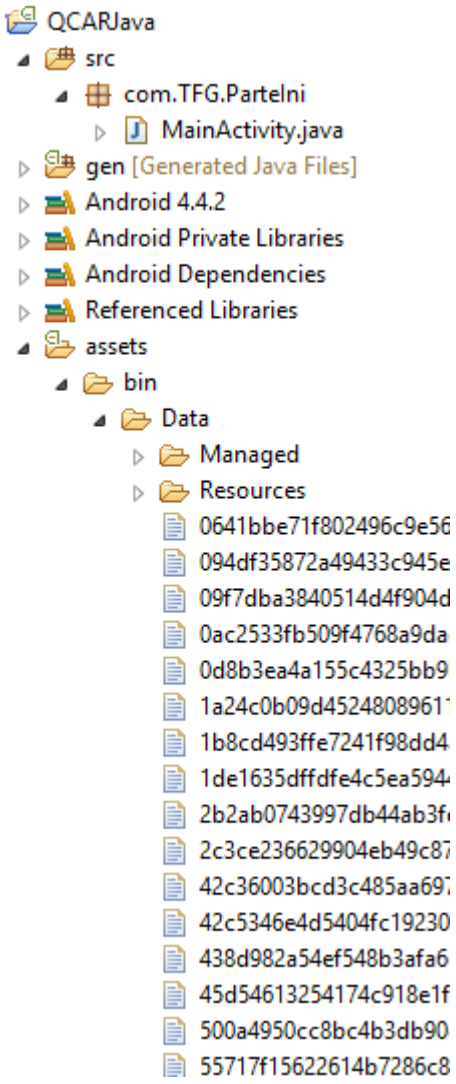


Fig. 15 Vista del nuevo contenido de la fase 1 tras su exportación

Para poder seguir trabajando con esta fase una vez exportada a Java/Android, debemos saber como poder guiar este código hacia donde nos interese.

El problema radica en que únicamente tenemos acceso una clase, "Main Activity".

Dicho esto, el "guiado" lo iniamos antes de la exportación de la fase 1 a Java/Android, concretamente en la clase "CSharpToJava".

Esta clase, por medio de los objetos de la clase “AndroidJavaClass”, interconectan clases programadas en CSharp con clases programadas en Java, es decir, por medio de las 2 siguientes líneas de código, pasamos los parámetros al meotod “estadioEncontradoJ”, método que se deberá implementar en la clase “MainActivity” al exportar.

```
AndroidJavaClass cls_CoarseLocation = new
AndroidJavaClass("com.TFG.ParteIni.MainActivity");
                        //Llamada al metodo estadioEncontradoJ de la clase
MainActivity y se pasan los parametros deseados

      cls_CoarseLocation.CallStatic("estadioEncontradoJ",
Variables.estadio,Variables.direccion,Variables.latitude,Variables.longit
ude,Variables.puerta,Variables.latPuerta,Variables.lonPuerta);
```

A continuación se puede visualizar los pasos seguidos en la exportación.

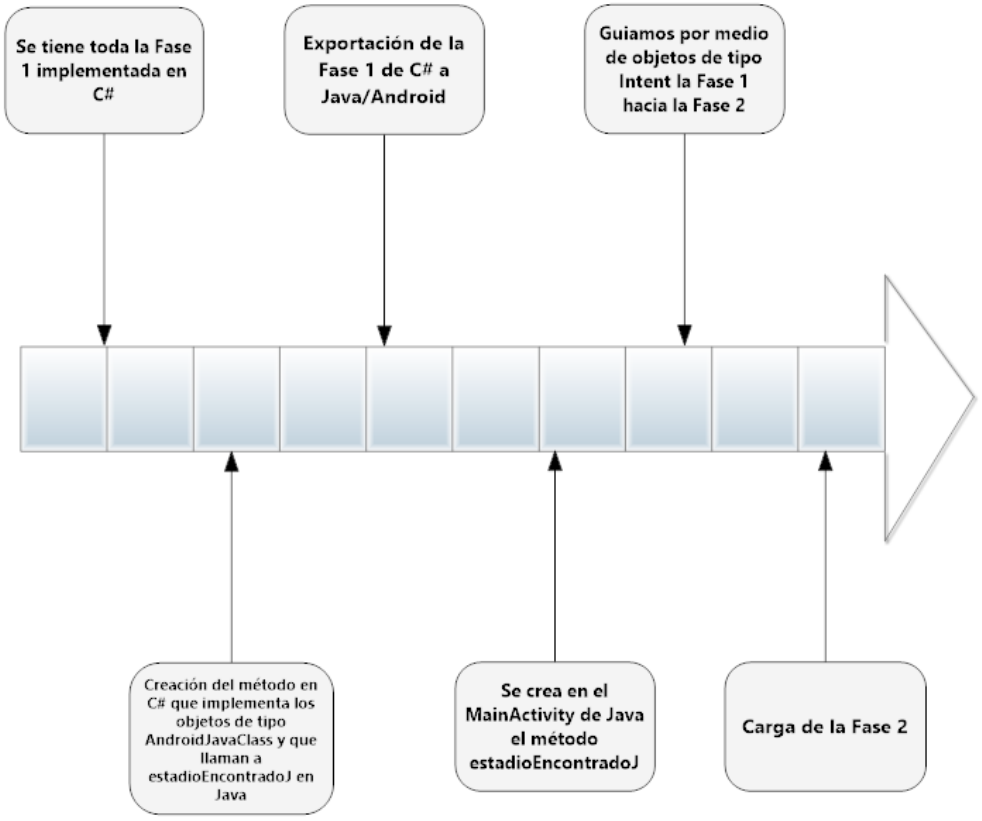


Fig. 16 Línea del tiempo de exportación de C# a Java

4.4 Guía del flujo del programa

Dado que resulta algo complejo para el lector desarrollar toda la idea de fases, exportación, clases, actividades… Se muestra a continuación una línea del tiempo de los principales procesos, clases y actividades llevadas a cabo a lo largo del programa de una forma más clara.

Se puede observar las diferentes fases según su color, las llamadas entre una clase y otra, así como algunas de las principales clases implicadas en cada escena de la primera fase.

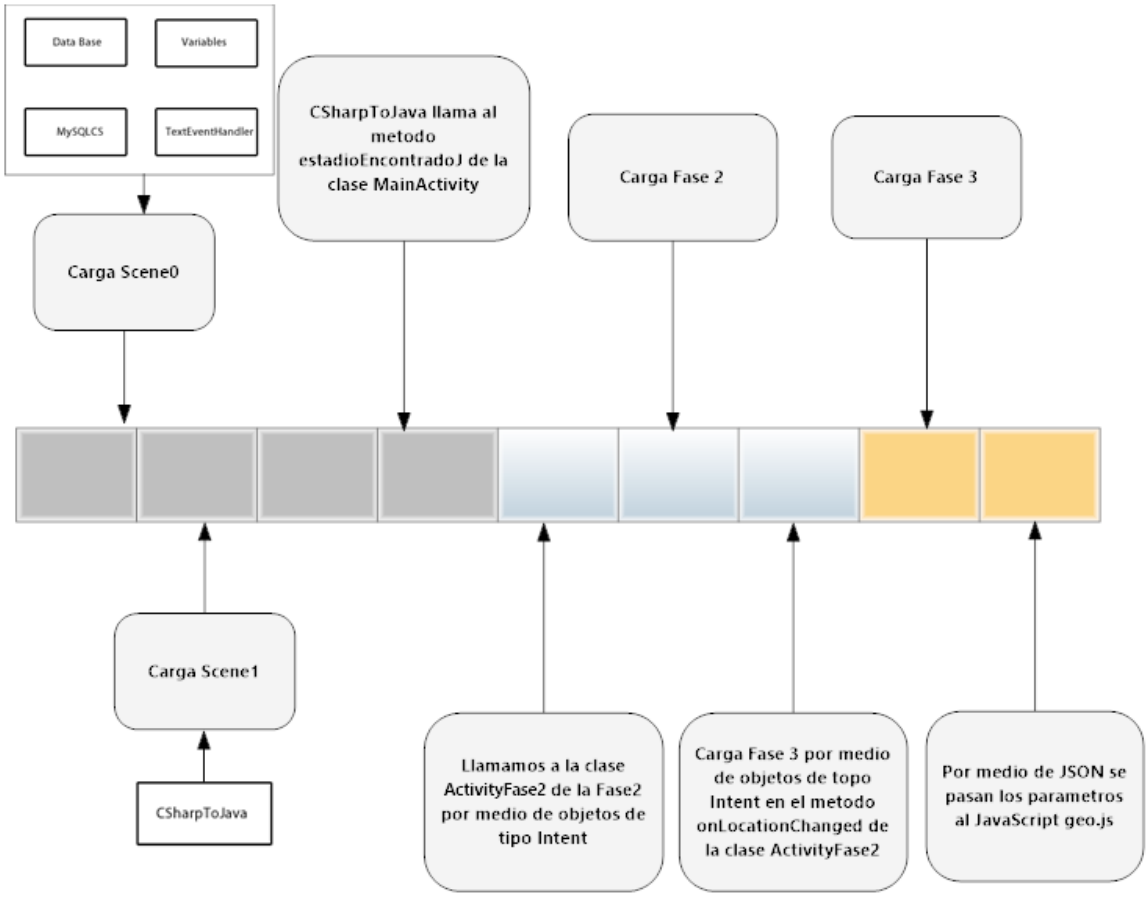


Fig. 17 Guía de las fases, clases, procesos y actividades del programa

4.5 Metodología de desarrollo

En el proceso para el desarrollo de software se ha seguido un proceso en cascada, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior.

Al final de cada etapa, el modelo está diseñado para llevar a cabo una revisión final, que se encarga de determinar si el proyecto está listo para avanzar a la siguiente fase.

Así, se ha seguido la siguiente metodología de desarrollo en cascada:

1. Análisis de requisitos.
2. Diseño del Sistema.
3. Diseño del Programa.
4. Pruebas.

De esta forma, cualquier error de diseño detectado en la etapa de prueba ha reconducido necesariamente al rediseño y nueva programación del código afectado.

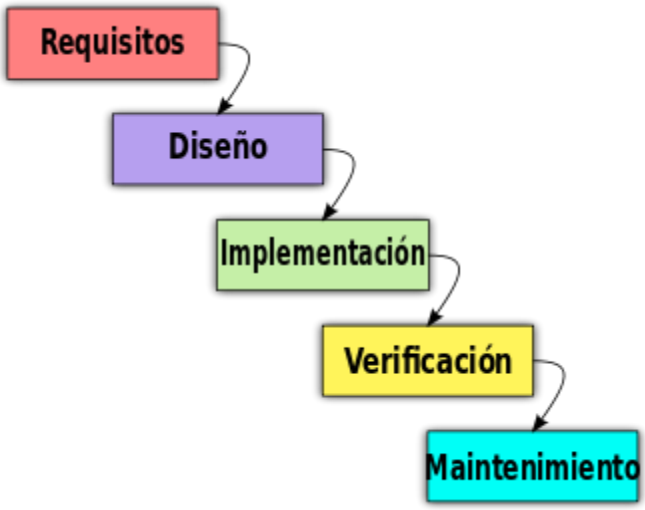


Fig. 18 El "modelo cascada" sin modificar

4.6 Estudio de la interfaz del usuario

Se ha realizado una interfaz de usuario extremadamente sencilla, que pese a limitar las opciones de uso de la aplicación, aporta como contrapartida una fácil comprensión del uso de la aplicación para todos los públicos.

Inicialmente será mostrada al usuario una ventana de información relativa al proyecto.

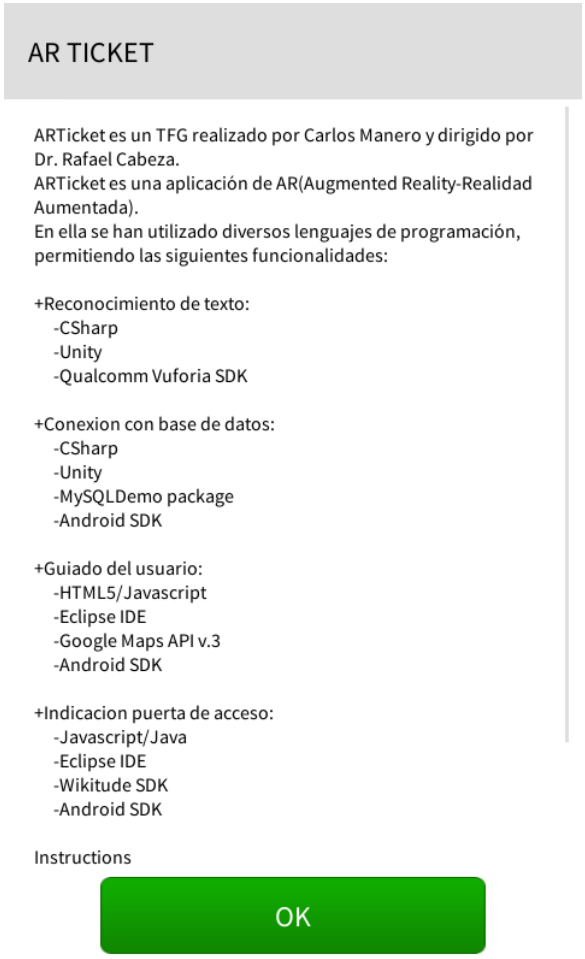


Fig. 19 Ventana de información relativa al proyecto

Una vez pulsado el botón de “OK” de la ventana inicial, el usuario recibirá la instrucción “Escanee el equipo local”.

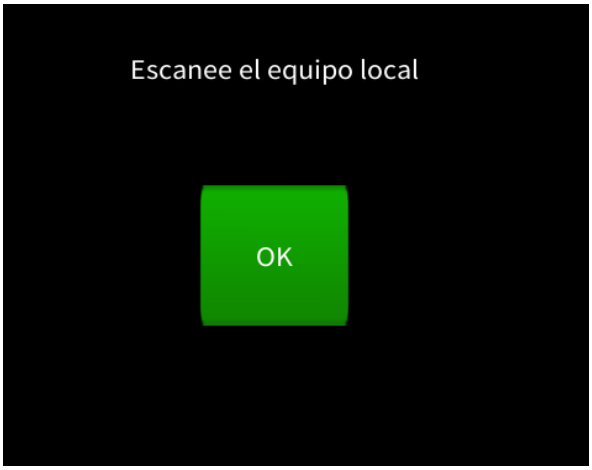


Fig. 20 Ventana instrucción al usuario

Tras pulsar el boton “OK”, se ejecutará la ventana de busqueda estadio, donde una “lupa” rastrea el lugar enfocado por el usuario.

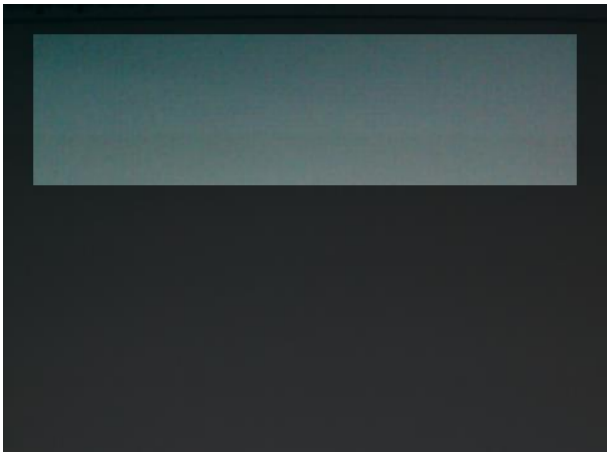


Fig. 21 Lupa de rastreo

Una vez que se reconoce un lugar, se muestra la siguiente ventana, donde el usuario debe decidir si el lugar encontrado es el deseado, o por el contrario se carga de nuevo la ventana anterior para realizar nuevamente el rastreo.

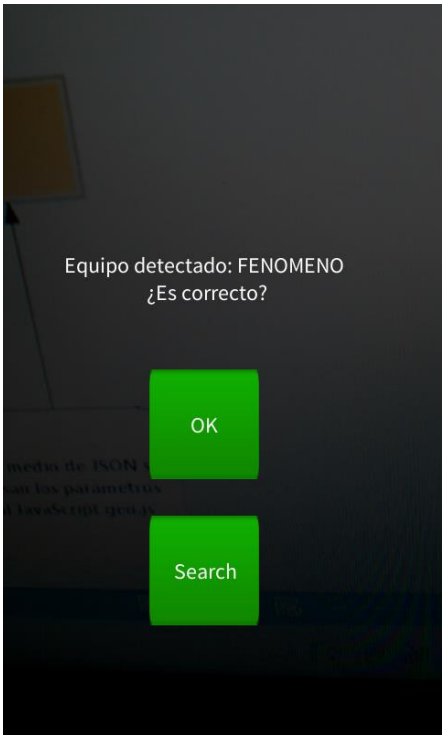


Fig. 22 Menu equipo detectado

Si el usuario a decidido que el lugar es el correcto, pasará a la siguiente ventana, donde se le indicará la instrucción “Inserte puerta de entrada”

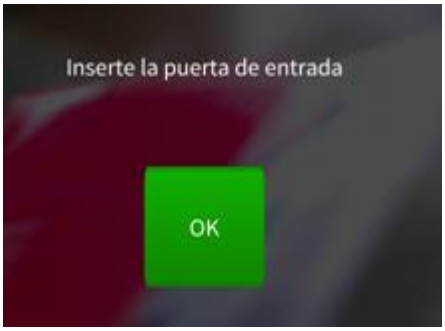


Fig. 23 Menu de insertar puerta

A continuación, se muestra un menu relativamente parecido que muestra la instrucción “Inserte asiento”

Tras introducir todos los datos necesarios, comenzará el guiado del usuario por medio del mapa.

Fig. 24 Mapa de guiado del usuario

Por último, al llegar al sitio el usuario debe pulsar el botón de llegada y podrá visualizar en la siguiente ventana el entorno de realidad aumentada donde un objeto señalará la zona de entrada al lugar.

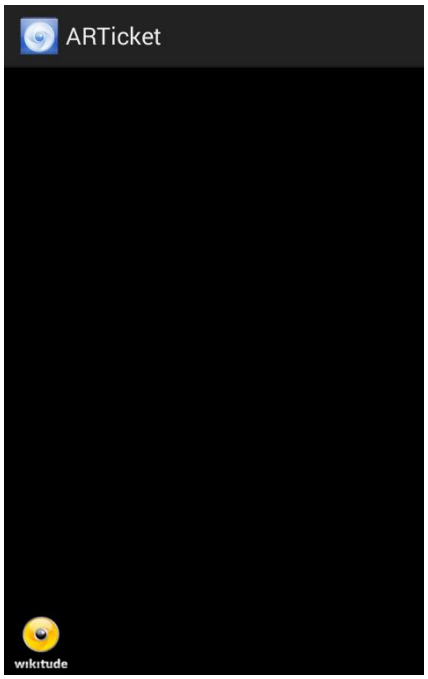


Fig. 25 Interface de la fase 3

5

Resultados

En este último capítulo se describirán los resultados obtenidos en lo que a los objetivos propuestos para cada fase y en conjunto se refiere.

5.1 Resultados Fase 1

El objetivo de la fase 1 principalmente era el reconocimiento de texto por parte de la aplicación para la posterior consulta junto con la introducción de parámetros por parte del usuario.

Este reconocimiento de texto, ha resultado totalmente satisfactorio. En condiciones de buena iluminación diurna sin iluminación adicional, la detección de texto en algunos casos no lleva más de 2-6 segundos, consiguiéndose así el primero de los objetivos de esta fase.

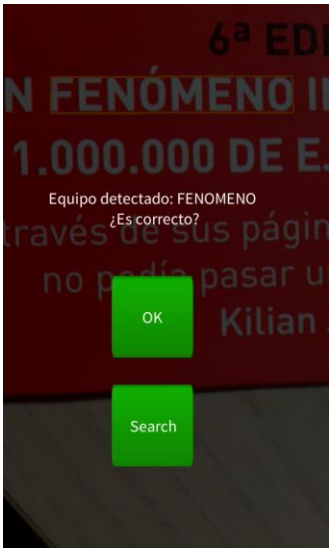


Fig. 26 Detección satisfactoria

5.2 Resultados Fase 2

El objetivo marcado para esta segunda fase, era guiar al usuario hasta el lugar asociado a la palabra reconocida.

Para ello, se hizo uso de google maps, por lo que podemos considerar que el objetivo queda cumplido ya que se puede visualizar perfectamente a través de un mapa, la posición del usuario, la posición destino, asi como la ruta que debe realizar para llegar al destino.

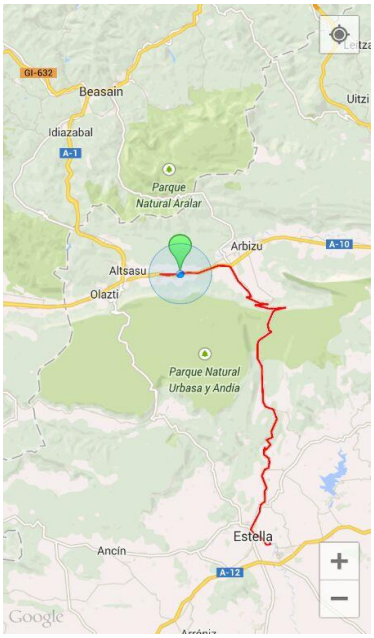


Fig. 27 Mapa guiado al usuario desde su posición hasta el destino

5.3 Resultados Fase 3

En esta última fase, el objetivo del proyecto era crear un mundo de realidad aumentada en el que una flecha indicara la puerta de entrada del evento.

Este objetivo ha quedado cumplido completamente al conseguir realizar este mundo por medio del SDK que se comenta en el análisis y descripción de esta fase.

Fig. 28 Entorno de realidad aumentada

5.4 Resultado Final

El resultado final es una aplicación que realiza todo lo estipulado consiguiendo unir cada una de las fases y tecnologías implementadas.

6

Conclusiones

En opinión del autor, este proyecto ha cumplido todo objetivo propuesto en su diseño y sobre todo, y tanto los errores como los problemas.

Los problemas a los que ha sometido este proyecto, han sido realmente tediosos, calculando que un 30-40% del tiempo invertido en el proyecto ha sido intentando subsanar problemas, fallos y errores.

Otro gran problema al que ha habido que enfrentarse, ha sido el desconocimiento de algunas tecnologías, como es el programa Unity3d, la falta de conocimiento del lenguaje de programación C#, realizar aplicaciones para Android o conceptos básicos acerca de la realidad aumentada. Este desconocimiento, ha hecho que un 20-30% del tiempo dedicado al proyecto ha debido ser dedicado a la lectura y aprendizaje de todo lo utilizado durante el proyecto.

Por último, la parte de programación y desarrollo ha dedicado el resto del tiempo, dejando margen de tiempo para las pruebas de cada fase.

Realmente, ha sido un proyecto ambicioso, en el que sin tener apenas conocimientos de nada, se ha querido hacer un proyecto, que a priori no se sabía su viabilidad, duración y desarrollo pero que finalmente se ha conseguido.

Sin duda, a pesar de las frustraciones que durante muchos momentos ha generado el proyecto, ha merecido la pena la satisfacción de conseguir llevarlo a cabo.

Bibliografía

[1] B. Furht, «Handbook of Augmented Reality,» de *Handbook of Augmented Reality*, 2011, p. Preface.

[2] Anónimo, «Wikipedia - Eclipse,» 2008. [En línea]. Available: http://es.wikipedia.org/wiki/Eclipse_(software).

[3] Inc., Google, «Android Desarrolladores,» [En línea]. Available: http://developer.android.com/sdk/index.html.

[4] Anónimo, «Wikipedia - SDK,» 2014. [En línea]. Available: http://es.wikipedia.org/wiki/SDK.

[5] Anónimo, «Wikipedia - Desarrollo de programas para Android,» [En línea]. Available: http://es.wikipedia.org/wiki/Desarrollo_de_programas_para_Android.

[6] A. D. Silva, «con-cafe,» 18 07 2013. [En línea]. Available: http://www.con-cafe.com/index.php/2013/07/qualcomm-vuforia-2-5-sdk/.

[7] Google Inc., «Google Developers,» [En línea]. Available: https://developers.google.com/maps/documentation/android/reference.

[8] Microsoft Corporation, «MSDN-the microsoft developer network,» [En línea]. Available: http://msdn.microsoft.com/es-es/library/kx37x362.aspx.

[9] Oracle, «Java™ Programming Language,» [En línea]. Available: http://docs.oracle.com/javase/7/docs/technotes/guides/language/.

[10] Anonimo, «Wikipedia - Java (programming language),» [En línea]. Available: http://en.wikipedia.org/wiki/Java_(programming_language).

[11] Anonimo, «Wikipedia - JavaScript,» [En línea]. Available: http://es.wikipedia.org/wiki/JavaScript.

[12] Anónimo, «Wikipedia - JSON,» [En línea]. Available: http://es.wikipedia.org/wiki/JSON.

[13] Anónimo, «Wikipedia - MySQL,» [En línea]. Available: http://es.wikipedia.org/wiki/MySQL.

[14] V. Qualcomm, «Vuforia Qualcomm Developer Guide,» [En línea]. Available: https://developer.vuforia.com/resources/dev-guide/extending-unity-android-activity-and-adding-custom-views-eclipse.

[15] J. Liberty, The C# Language, O'Reilly Media.

[16] W3School, «W3School,» [En línea]. Available: http://www.w3schools.com/.

[17] Oracle, «MySQL,» [En línea]. Available: http://www.mysql.com/.

ANEXOS

I. Anexo : Clases y métodos fase 1

1. *Clase TextRecognitionUIEventHandler:*

```
/// <summary>
/// UI Event Handler class that handles events generated by user-tap
actions
/// over the UI Options Menu
/// </summary>
public class TextRecognitionUIEventHandler : ISampleAppUIEventHandler {

    #region PUBLIC_MEMBER_VARIABLES
    public override event System.Action CloseView;
    public override event System.Action GoToAboutPage;
    #endregion PUBLIC_MEMBER_VARIABLES

    #region PRIVATE_MEMBER_VARIABLES
    private TextRecognitionUIView mView;
    #endregion PRIVATE_MEMBER_VARIABLES

    #region PUBLIC_MEMBER_PROPERTIES
    public TextRecognitionUIView View
    {
        get {
            if(mView == null){
                mView = new TextRecognitionUIView();
                mView.LoadView();
            }
            return mView;
        }
    }
    #endregion PUBLIC_MEMBER_PROPERTIES

    #region PUBLIC_METHODS
    public override void UpdateView (bool tf)
    {
        this.View.UpdateUI(tf);
    }

    public override  void Bind()
    {
        this.View.mCameraFlashSettings.TappedOn     +=
OnTappedToTurnOnFlash;
        this.View.mCloseButton.TappedOn             +=
OnTappedOnCloseButton;
        this.View.mAboutLabel.TappedOn              +=
OnTappedOnAboutButton;
    }

    public override  void UnBind()
    {
```

```
        this.View.mCameraFlashSettings.TappedOn     -=
OnTappedToTurnOnFlash;
        this.View.mCloseButton.TappedOn             -=
OnTappedOnCloseButton;
        this.View.UnLoadView();
    }

    //SingleTap gesture by users are captured by AppManager class that
calls this method for TapToFocus
    public override  void TriggerAutoFocus()
    {
        if
(!CameraDevice.Instance.SetFocusMode(CameraDevice.FocusMode.FOCUS_MODE_TR
IGGERAUTO)) {
            Debug.LogError("Could not trigger autofocus");
        }
    }

    public override  void SetToDefault(bool tf)
    {
        this.View.mCameraFlashSettings.Enable(tf);
    }
    #endregion PUBLIC_METHODS

    #region PRIVATE_METHODS

    private void OnTappedOnAboutButton(bool tf)
    {
        if(this.GoToAboutPage != null)
        {
            this.GoToAboutPage();
        }
    }

    private void OnTappedToTurnOnFlash(bool tf)
    {
        if(tf)
        {
            if(!CameraDevice.Instance.SetFlashTorchMode(true))
            {
                this.View.mCameraFlashSettings.Enable(false);
            }
        }
        else
        {
            CameraDevice.Instance.SetFlashTorchMode(false);
        }

        OnTappedToClose();
    }

    private void OnTappedToClose()
    {
        if(this.CloseView != null)
        {
            this.CloseView();
        }
```

```
    }

    private void OnTappedOnCloseButton()
    {
        OnTappedToClose();
    }
    #endregion //PRIVATE_METHODS
}
```

==========

2. *Clase TextEventHandler:*

```
/*============================================================================
=======
 * Copyright (c) 2012-2014 Qualcomm Connected Experiences, Inc. All
Rights Reserved.
 *
============================================================================
======*/

using System.Collections.Generic;
using UnityEngine;

/// <summary>
/// A custom event handler for TextReco-events
/// </summary>
public class TextEventHandler : MonoBehaviour, ITextRecoEventHandler,
IVideoBackgroundEventHandler
{
    #region PRIVATE_MEMBER_VARIABLES
      //VARIABLES DE DESARROLLADOR
      private int level = 1;
      private int sublevel = 1;
      private bool detection = false;
      private GUIStyle mAboutTitleBgStyle;
      private GUIStyle mOKButtonBgStyle;
      private GUIStyle mSearchButtonBgStyle;
      private GUIStyle mInputStyle;
      private float widthButton, heightButton,widthInput,heightInput;
      private string stadium;
      private string numPlace="";
      private string numGate="";
      public float a;
      private MySQLCS sql;
    // Size of text search area in percentage of screen
    private float mLoupeWidth = 0.9f;
    private float mLoupeHeight = 0.15f;
    // Alpha value for area outside of text search
    private float mBackgroundAlpha = 0.7f;
    // Size of text box for visualizing detected words in percentage of
remaining screen outside text search area
    private float mTextboxWidth = 0.9f;
    private float mTextboxHeight = 0.95f;
    // Number of words before scaling word list
    private int mFixedWordCount = 9;
    // Padding between lines in word list
    private float mWordPadding = 0.05f;
    // Minimum line height for word list
    private float mMinLineHeight = 15.0f;
    // Line width of viusalized boxes around detected words
    private float mBBoxLineWidth = 3.0f;
    // Padding between detected words and visualized boxes
    private float mBBoxPadding = 0.0f;
    // Color of visualized boxes around detected words
    private Color mBBoxColor = new Color(1.0f, 0.447f, 0.0f, 1.0f);
```

```
    private Rect mDetectionAndTrackingRect;
    private Texture2D mBackgroundTexture;
    private Texture2D mBoundingBoxTexture;
    private Material mBoundingBoxMaterial;

    private GUIStyle mWordStyle;
    private bool mIsTablet;
    private bool mIsInitialized;
    private bool mVideoBackgroundChanged;

    private readonly List<WordResult> mSortedWords = new
List<WordResult>();


    [SerializeField]
    private Material boundingBoxMaterial = null;
      #endregion

    #region UNTIY_MONOBEHAVIOUR_METHODS

    public void InitHandler()
    {
            //Conexion SQL y apertura;
            sql=new MySQLCS();
            sql.Open ();

            mAboutTitleBgStyle = new GUIStyle();
            mOKButtonBgStyle = new GUIStyle();
            mSearchButtonBgStyle=new GUIStyle();

            mAboutTitleBgStyle.normal.textColor = Color.white;
            mOKButtonBgStyle.normal.textColor = Color.white;
            mSearchButtonBgStyle.normal.textColor = Color.white;

            mAboutTitleBgStyle.alignment = TextAnchor.MiddleCenter;
            mOKButtonBgStyle.alignment = TextAnchor.MiddleCenter;
            mSearchButtonBgStyle.alignment = TextAnchor.MiddleCenter;

            mOKButtonBgStyle.normal.background = Resources.Load
("UserInterface/capture_button_normal_XHigh") as Texture2D;
            mSearchButtonBgStyle.normal.background = Resources.Load
("UserInterface/capture_button_normal_XHigh") as Texture2D;

            mAboutTitleBgStyle.font = Resources.Load("SourceSansPro-
Regular_big_xhdpi") as Font;
            mOKButtonBgStyle.font = Resources.Load("SourceSansPro-
Regular_big_xhdpi") as Font;
            mSearchButtonBgStyle.font = Resources.Load("SourceSansPro-
Regular_big_xhdpi") as Font;


            widthButton = Screen.width * 0.25f;
            heightButton = Screen.height * 0.15f;
            widthInput = Screen.width * 0.25f;
            heightInput = Screen.height * 0.10f;

            if(Screen.dpi > 300 ){
```

```
                // load and set gui style
                mAboutTitleBgStyle.font =
Resources.Load("SourceSansPro-Regular_big_xhdpi") as Font;
                mOKButtonBgStyle.font = Resources.Load("SourceSansPro-
Regular_big_xhdpi") as Font;
                mSearchButtonBgStyle.font =
Resources.Load("SourceSansPro-Regular_big_xhdpi") as Font;
            } else if(Screen.dpi > 260 ){
                mAboutTitleBgStyle.font =
Resources.Load("SourceSansPro-Regular_big_xhdpi") as Font;
                mOKButtonBgStyle.font = Resources.Load("SourceSansPro-
Regular_big_xhdpi") as Font;
                mSearchButtonBgStyle.font =
Resources.Load("SourceSansPro-Regular_big_xhdpi") as Font;
            }
            else if (Screen.height == 1848 && Screen.width == 1200)
            {
                mAboutTitleBgStyle.font =
Resources.Load("SourceSansPro-Regular_big_xhdpi") as Font;
                mOKButtonBgStyle.font = Resources.Load("SourceSansPro-
Regular_big_xhdpi") as Font;
                mSearchButtonBgStyle.font =
Resources.Load("SourceSansPro-Regular_big_xhdpi") as Font;

            }
            else
            {
                mAboutTitleBgStyle.font =
Resources.Load("SourceSansPro-Regular") as Font;
                mOKButtonBgStyle.font = Resources.Load("SourceSansPro-
Regular") as Font;
                mSearchButtonBgStyle.font =
Resources.Load("SourceSansPro-Regular") as Font;
            }
        // create the background texture (size 1x1, can be scaled to any
size)
        mBackgroundTexture = new Texture2D(1, 1, TextureFormat.ARGB32,
false);
        mBackgroundTexture.SetPixel(0, 0, new Color(0f, 0f, 0f,
mBackgroundAlpha));
        mBackgroundTexture.Apply(false);

        // create the texture for bounding boxes
        mBoundingBoxTexture = new Texture2D(1, 1, TextureFormat.ARGB32,
false);
        mBoundingBoxTexture.SetPixel(0, 0, mBBoxColor);
        mBoundingBoxTexture.Apply(false);

        mBoundingBoxMaterial = new Material(boundingBoxMaterial);
        mBoundingBoxMaterial.SetTexture("_MainTex", mBoundingBoxTexture);

        mWordStyle = new GUIStyle();
        mWordStyle.normal.textColor = Color.white;
        mWordStyle.alignment = TextAnchor.UpperCenter;
        mWordStyle.font = Resources.Load("SourceSansPro-Regular_big") as
Font;
```

```
        mIsTablet = IsTablet();
        if (QCARRuntimeUtilities.IsPlayMode())
            mIsTablet = false;
        if (mIsTablet)
        {
            mLoupeWidth = 0.6f;
            mLoupeHeight = 0.1f;
            mTextboxWidth = 0.6f;
            mFixedWordCount = 14;
        }

        // register to TextReco events
        var trBehaviour = GetComponent<TextRecoBehaviour>();
        if (trBehaviour)
        {
            trBehaviour.RegisterTextRecoEventHandler(this);
        }

        // register for the OnVideoBackgroundConfigChanged event at the
QCARBehaviour
        QCARBehaviour qcarBehaviour =
(QCARBehaviour)FindObjectOfType(typeof(QCARBehaviour));
        if (qcarBehaviour)
        {
            qcarBehaviour.RegisterVideoBgEventHandler(this);
        }
    }

    public void Draw()
    {
            //Jerarquia por niveles que permite realizar una sencilla UI
            //Se determina por medio de la variable level en que nivel se
encuentra el usuario
            switch (level)
            {
            case 1:
                  //Si el usuario se encuentra en el nivel 1 y se desea
detectar palabra
                  if(detection){
                        //draw background - tracking:
                        //Dibujo el area de busqueda y las palabras
encontradas
                        DrawMaskedRectangle(mDetectionAndTrackingRect);
                        DrawWordList();
                  }else{
                        //En caso de que en este momento no se desee
detectar palabra
                        nivel1();
                  }
                  break;
            case 2:
                  nivel2 ();
                  break;
            case 3:
                  nivel3();
                  break;
            case 4:
```

```
                  //En este ultimo nivel se obtiene en la DB los
parametros de busqueda
                  sql.SelectDates(stadium,numGate,numPlace);
                  //Una vez realizado todo, se carga la siguiente escena,
                  //que unicamente contiene un script que llama a la
clase JAVA deseada
                  Application.LoadLevel(1);
                  break;
            default:
                  print ("Incorrect intelligence level.");
                  break;
            }
    }
      private void nivel1(){
            switch (sublevel)
            {
            case 1:
                  //Menu Escanee el estadio local
                  GUI.DrawTexture(new Rect(0f, 0f, Screen.width,
Screen.height), mBackgroundTexture);
                  GUI.Label(new Rect (Screen.width/3, Screen.height/4,
Screen.width/4, Screen.height/4), "Escanee el equipo local",
mAboutTitleBgStyle);
                  if (GUI.Button(new Rect(Screen.width/3,
Screen.height/2, widthButton, heightButton), "OK" ,mOKButtonBgStyle))
                  {
                        detection=true;
                  }
                  break;
            case 2:
                  //Menu ¿Estadio Correcto?
                  GUI.DrawTexture(new Rect(0f, 0f, Screen.width,
Screen.height), mBackgroundTexture);
                  string frase="Equipo detectado: "+stadium+"\n ¿Es
correcto?";
                  GUI.Label(new Rect (Screen.width/3, Screen.height/4,
Screen.width/4, Screen.height/4), frase, mAboutTitleBgStyle);
                  //En caso de que el usuario acepte
                  if (GUI.Button(new Rect(Screen.width/3,
(float)(Screen.height*0.5), widthButton, heightButton), "OK"
,mOKButtonBgStyle))
                  {
                        detection=false;
                        level=2;//pasamos al siguiente nivel
                        sublevel=1;
                  }
                  //En caso de que el usuario desee realizar una nueva
busqueda
                  if (GUI.Button(new Rect(Screen.width/3,
(float)(Screen.height*0.7), widthButton, heightButton), "Search"
,mSearchButtonBgStyle))
                  {
                        detection=true;
                  }
                  break;
            default:
                  print ("Incorrect intelligence level.");
```

```
                  break;
            }
      }
      private void nivel2(){
            switch (sublevel)
            {
            case 1:
                  //Menu "Inserte puerta de entrada"
                  GUI.DrawTexture(new Rect(0f, 0f, Screen.width,
Screen.height), mBackgroundTexture);
                  GUI.Label(new Rect (Screen.width/3, Screen.height/4,
Screen.width/4, Screen.height/4), "Inserte la puerta de entrada",
mAboutTitleBgStyle);
                  if (GUI.Button(new Rect(Screen.width/3,
Screen.height/2, widthButton, heightButton), "OK" ,mOKButtonBgStyle))
                  {
                        sublevel=2;
                  }
                  break;
            case 2:
                  //Menu de introduccion de la puerta de entrada, para
ello se utiliza un TextField
                  GUI.DrawTexture(new Rect(0f, 0f, Screen.width,
Screen.height), mBackgroundTexture);
                  numGate = GUI.TextField(new
Rect((float)(Screen.width*0.33), Screen.height/4, widthInput,
heightInput), numGate, 25);
                  if (GUI.Button(new Rect(Screen.width/3,
(float)(Screen.height*0.5), widthButton, heightButton), "OK"
,mOKButtonBgStyle))
                  {
                        level=3;//pasamos al siguiente nivel
                        sublevel=1;
                  }
                  break;
            default:
                  print ("Incorrect intelligence level.");
                  break;
            }
      }
      private void nivel3(){
            switch (sublevel)
            {
            case 1:
                  //Menu "Inserte asiento"
                  GUI.DrawTexture(new Rect(0f, 0f, Screen.width,
Screen.height), mBackgroundTexture);
                  GUI.Label(new Rect (Screen.width/3, Screen.height/4,
Screen.width/4, Screen.height/4), "Inserte el asiento",
mAboutTitleBgStyle);
                  if (GUI.Button(new Rect(Screen.width/3,
Screen.height/2, widthButton, heightButton), "OK" ,mOKButtonBgStyle))
                  {
                        sublevel=2;
                  }
                  break;
            case 2:
```

```
                  //Menu de introduccion del asiento, para ello se
utiliza un TextField
                  GUI.DrawTexture(new Rect(0f, 0f, Screen.width,
Screen.height), mBackgroundTexture);
                  numPlace = GUI.TextField(new
Rect((float)(Screen.width*0.33), Screen.height/4-heightButton/2,
widthInput, heightInput), numPlace, 25);
                  if (GUI.Button(new Rect(Screen.width/3,
(float)(Screen.height*0.5), widthButton, heightButton), "OK"
,mOKButtonBgStyle))
                  {
                        level=4;//carga del nivel 4, donde se produce la
consulta con la DB
                  }
                  break;
            default:
                  print ("Incorrect intelligence level.");
                  break;
            }
      }

    void OnRenderObject()
    {
        DrawWordBoundingBoxes();
    }

    public void UpdateHandler()
    {
        // once the text tracker has initialized and every time the video
background changed, set the region of interest
        if (mIsInitialized && mVideoBackgroundChanged)
        {
            TextTracker textTracker =
TrackerManager.Instance.GetTracker<TextTracker>();
            if (textTracker != null)
            {
                CalculateLoupeRegion();

textTracker.SetRegionOfInterest(mDetectionAndTrackingRect,
mDetectionAndTrackingRect);
            }
            mVideoBackgroundChanged = false;
        }
    }

    #endregion // UNTIY_MONOBEHAVIOUR_METHODS



    #region ITextRecoEventHandler_IMPLEMENTATION

    /// <summary>
    /// Called when the text reco system has finished initializing
    /// </summary>
    public void OnInitialized()
    {
        CalculateLoupeRegion();
```

```
        mIsInitialized = true;
    }

    /// <summary>
    /// This method will be called whenever a new word has been detected
    /// </summary>
    /// <param name="wordResult">New trackable with current pose</param>

    public void OnWordDetected(WordResult wordResult)
    {
        var word = wordResult.Word;
        if (ContainsWord(word))
            Debug.LogError("Word was already detected before!");

        Debug.Log("Text: New word: " + wordResult.Word.StringValue + "("
+ wordResult.Word.ID + ")");
        //Llamamos a la funcion AddWord pasando el parametro wordResult
            //Este hecho no afecta para nada al funcionamiento del
programa
            //Se trata simplemente de un almacenamiento de las palabras
que se van detectando
            AddWord(wordResult);
            //Se almacena la palabra detectada en la variable stadium
            //que posteriormente se utilizara en la busqueda de datos en
la DB
            stadium = wordResult.Word.StringValue;
            Debug.LogError (stadium);
            detection = false;//deteccion producida por tanto, deteccion
ya no necesaria
            sublevel = 2;
    }

    /// <summary>
    /// This method will be called whenever a tracked word has been lost
and is not tracked anymore
    /// </summary>
    public void OnWordLost(Word word)
    {
        if (!ContainsWord(word))
            Debug.LogError("Non-existing word was lost!");

        Debug.Log("Text: Lost word: " + word.StringValue + "(" + word.ID
+ ")");

        RemoveWord(word);
    }

    #endregion // PUBLIC_METHODS



    #region IVideoBackgroundEventHandler_IMPLEMENTATION

    // set a flag that the video background has changed. This means the
region of interest has to be set again.
    public void OnVideoBackgroundConfigChanged()
    {
```

```
            mVideoBackgroundChanged = true;
        }

        #endregion // IVideoBackgroundEventHandler_IMPLEMENTATION



        #region PRIVATE_METHODS

        /// <summary>
        /// Draw a 3d bounding box around each currently tracked word
        /// </summary>
        private void DrawWordBoundingBoxes()
        {
            // render a quad around each currently tracked word
            foreach (var word in mSortedWords)
            {
                var pos = word.Position;
                var orientation = word.Orientation;
                var size = word.Word.Size;
                var pose = Matrix4x4.TRS(pos, orientation, new
Vector3(size.x, 1, size.y));

                var cornersObject = new[]
                    {
                        new Vector3(-0.5f, 0.0f, -0.5f), new Vector3(0.5f,
0.0f, -0.5f),
                        new Vector3(0.5f, 0.0f, 0.5f), new Vector3(-0.5f,
0.0f, 0.5f)
                    };
                var corners = new Vector2[cornersObject.Length];
                for (int i = 0; i < cornersObject.Length; i++)
                    corners[i] =
Camera.current.WorldToScreenPoint(pose.MultiplyPoint(cornersObject[i]));
                DrawBoundingBox(corners);
            }
        }

        /// <summary>
        /// Print string values for all currently tracked words.
        /// </summary>
        private void DrawWordList()
        {
            var sortedWords = mSortedWords;

            var textBoxWidth = Screen.width * mTextboxWidth;
            var textBoxHeight = (Screen.height -
mDetectionAndTrackingRect.yMax) * mTextboxHeight;
            var textBoxOffsetLeft = (Screen.width - textBoxWidth) * 0.5f;
            var textBoxOffsetTop = mDetectionAndTrackingRect.yMax +
(Screen.height - (textBoxHeight + mDetectionAndTrackingRect.yMax)) *
0.5f;

            var textBox = new Rect(textBoxOffsetLeft, textBoxOffsetTop,
textBoxWidth, textBoxHeight);
            Rect wordBox;
```

```
        var scale = ComputeScaleForWordList(mSortedWords.Count, textBox,
out wordBox);


        var oldMatrix = GUI.matrix;

        GUIUtility.ScaleAroundPivot(new Vector2(scale, scale), new
Vector2(Screen.width * 0.5f, textBoxOffsetTop));

        wordBox.y += wordBox.height*mWordPadding;
        foreach (var word in sortedWords)
        {
            if ((wordBox.yMax - textBoxOffsetTop) * scale >
textBox.height)
                break;
            GUI.Label(wordBox, word.Word.StringValue, mWordStyle);
            wordBox.y += (wordBox.height + wordBox.height *
mWordPadding);
        }

        GUI.matrix = oldMatrix;
    }

    private void DrawMaskedRectangle(Rect rectangle)
    {
        // draw four texture quads in UI that mask out the given region
        GUI.DrawTexture(new Rect(0f, 0f, rectangle.xMin, Screen.height),
mBackgroundTexture);
        GUI.DrawTexture(new Rect(rectangle.xMin, 0f, rectangle.width,
rectangle.yMin), mBackgroundTexture);
        GUI.DrawTexture(new Rect(rectangle.xMin, rectangle.yMax,
rectangle.width, Screen.height - rectangle.yMax), mBackgroundTexture);
        GUI.DrawTexture(new Rect(rectangle.xMax, 0f, Screen.width -
rectangle.xMax, Screen.height), mBackgroundTexture);
    }

    private void DrawBoundingBox(Vector2[] corners)
    {
        var normals = new Vector2[4];
        for (var i = 0; i < 4; i++)
        {
            var p0 = corners[i];
            var p1 = corners[(i + 1)%4];
            normals[i] = (p1 - p0).normalized;
            normals[i] = new Vector2(normals[i].y, -normals[i].x);
        }

        //add padding to inner corners
        corners = ExtendCorners(corners, normals, mBBoxPadding);
        //computer outer corners
        var outerCorners = ExtendCorners(corners, normals,
mBBoxLineWidth);

        //create vertices in screen space
        var vertices = new Vector3[8];
        for (var i = 0; i < 4; i++)
        {
```

```
            vertices[i] = new Vector3(corners[i].x, corners[i].y,
Camera.current.nearClipPlane);
            vertices[i + 4] = new Vector3(outerCorners[i].x,
outerCorners[i].y, Camera.current.nearClipPlane);
        }
        //transform vertices into world space
        for (int i = 0; i < 8; i++)
            vertices[i] = Camera.current.ScreenToWorldPoint(vertices[i]);

        var mesh = new Mesh()
            {
                vertices = vertices,
                uv = new Vector2[8],
                triangles = new[]
                    {
                        0, 5, 4, 1, 5, 0,
                        1, 6, 5, 2, 6, 1,
                        2, 7, 6, 3, 7, 2,
                        3, 4, 7, 0, 4, 3
                    },
            };

        mBoundingBoxMaterial.SetPass(0);
        Graphics.DrawMeshNow(mesh, Matrix4x4.identity);
        Destroy(mesh);
    }


    private static Vector2[] ExtendCorners(Vector2[] corners, Vector2[]
normals, float extension)
    {
        //compute positions along the outer side of the boundary
        var linePoints = new Vector2[corners.Length * 2];
        for (var i = 0; i < corners.Length; i++)
        {
            var p0 = corners[i];
            var p1 = corners[(i + 1) % 4];

            var po0 = p0 + normals[i] * extension;
            var po1 = p1 + normals[i] * extension;
            linePoints[i * 2] = po0;
            linePoints[i * 2 + 1] = po1;
        }

        //compute corners of outer side of bounding box lines
        var outerCorners = new Vector2[corners.Length];
        for (var i = 0; i < corners.Length; i++)
        {
            var i2 = i * 2;
            outerCorners[(i + 1) % 4] = IntersectLines(linePoints[i2],
linePoints[i2 + 1], linePoints[(i2 + 2) % 8],
                                                       linePoints[(i2 + 3) % 8]);
        }
        return outerCorners;
    }

    /// <summary>
```

```
    /// Intersect the line p1-p2 with the line p3-p4
    /// </summary>
    private static Vector2 IntersectLines(Vector2 p1, Vector2 p2, Vector2
p3, Vector2 p4)
    {
        var denom = (p1.x - p2.x) * (p3.y - p4.y) - (p1.y - p2.y) * (p3.x
- p4.x);
        var x = ((p1.x * p2.y - p1.y * p2.x) * (p3.x - p4.x) - (p1.x -
p2.x) * (p3.x * p4.y - p3.y * p4.x)) / denom;
        var y = ((p1.x * p2.y - p1.y * p2.x) * (p3.y - p4.y) - (p1.y -
p2.y) * (p3.x * p4.y - p3.y * p4.x)) / denom;
        return new Vector2(x, y);
    }


    private static bool IsTablet()
    {

#if UNITY_IPHONE
        return iPhone.generation == iPhoneGeneration.iPad1Gen ||
               iPhone.generation == iPhoneGeneration.iPad2Gen ||
               iPhone.generation == iPhoneGeneration.iPad3Gen ||
               iPhone.generation == iPhoneGeneration.iPad4Gen ||
               iPhone.generation == iPhoneGeneration.iPadMini1Gen ||
               iPhone.generation == iPhoneGeneration.iPadUnknown;
#else

        var screenWidth = Screen.width / Screen.dpi;
        var screenHeight = Screen.height / Screen.dpi;
        var diagonal = Mathf.Sqrt(Mathf.Pow(screenWidth, 2) +
Mathf.Pow(screenHeight, 2));
        //tablets usually have a screen size greater than 6 inches
        return diagonal >= 6;
#endif

    }

    /// <summary>
    /// compute the scale for printing the string values of the currently
tracked words
    /// </summary>
    /// <param name="numWords">Number of currently tracked words</param>
    /// <param name="totalArea">Region where all words should be
printed</param>
    /// <param name="firstWord">Region for first word</param>
    /// <returns>necessary scale to put all words into the area</returns>
    private float ComputeScaleForWordList(int numWords, Rect totalArea,
out Rect firstWord)
    {
        if (numWords < mFixedWordCount)
            numWords = mFixedWordCount;


        var originalHeight = mWordStyle.lineHeight;
        var requestedHeight = totalArea.height / (numWords + mWordPadding
* (numWords + 1));
```

```
        if (requestedHeight < mMinLineHeight)
        {
            requestedHeight = mMinLineHeight;
        }
        var scale = requestedHeight / originalHeight;

        var newWidth = totalArea.width / scale;
        firstWord = new Rect(totalArea.xMin + (totalArea.width -
newWidth) * 0.5f, totalArea.yMin, newWidth, originalHeight);
        return scale;
    }


    private void AddWord(WordResult wordResult)
    {
        //add new word into sorted list
        var cmp = new ObbComparison();
        int i = 0;
        while (i < mSortedWords.Count && cmp.Compare(mSortedWords[i],
wordResult) < 0)
        {
            i++;
        }

        if (i < mSortedWords.Count)
        {
            mSortedWords.Insert(i, wordResult);
        }
        else
        {
            mSortedWords.Add(wordResult);
        }
    }

    private void RemoveWord(Word word)
    {
        for (int i = 0; i < mSortedWords.Count; i++)
        {
            if (mSortedWords[i].Word.ID == word.ID)
            {
                mSortedWords.RemoveAt(i);
                break;
            }
        }
    }

    private bool ContainsWord(Word word)
    {
        foreach (var w in mSortedWords)
            if (w.Word.ID == word.ID)
                return true;
        return false;
    }

    private void CalculateLoupeRegion()
    {
        // define area for text search
```

```
        var loupeWidth = mLoupeWidth * Screen.width;
        var loupeHeight = mLoupeHeight * Screen.height;
        var leftOffset = (Screen.width - loupeWidth) * 0.5f;
        var topOffset = leftOffset;
        mDetectionAndTrackingRect = new Rect(leftOffset, topOffset,
loupeWidth, loupeHeight);
    }


    #endregion //PRIVATE_METHODS
}
```

==========

3. *Clase MySQLCS:*

```
using UnityEngine;
using MySql.Data;
using MySql.Data.MySqlClient;
using System;
using System.Data;
using System.Collections;
using System.Collections.Generic;

public class MySQLCS
{
      string estadio;
      string direccion;
      float latitud;
      float longitud;

    // MySQL instance specific items
      string constr = "Server=qrv183.dbname.net;Database=qrv183;User
ID=qrv183;Password=Manero2014;Pooling=true";
    // connection object
    MySqlConnection con = null;
    // command object
    MySqlCommand cmd = null;
    // reader object
    MySqlDataReader rdr = null;

    // object definitions
    public void Open()
    {
        try
        {
            // setup the connection element
            con = new MySqlConnection(constr);
            // lets see if we can open the connection
            con.Open();
            Debug.LogError("Connection State: " + con.State);
        }
        catch (Exception ex)
        {
            Debug.Log(ex.ToString());
                  Open ();
        }
```

```
    }

      //selecciona de la DB los datos relativos a los parametros
introducidos por el usuario
      //en el manejador de eventos
      public void SelectDates(string stadium,string numGate,string
numPlace){
            //se crea la consulta para la DB
            string query = "SELECT * FROM `Estadios` WHERE
name='"+stadium+"'";
            try{
                  //Ejecutamos la consulta por medio de la conexion
abierta y la query
                  cmd = new MySqlCommand(query, con);
                  rdr = cmd.ExecuteReader();
                  while (rdr.Read())
                  {
                        //obtenemos los parametros devueltos por la
consulta
                        estadio = rdr["name"].ToString();
                        direccion = rdr["direccion"].ToString();
                        latitud =float.Parse(rdr["latitude"].ToString());
                        longitud
=float.Parse(rdr["longitude"].ToString());
                        int limitGate=int.Parse(numGate);
                        string indice="puerta"+numGate;
                        //Nos quedamos solo con la puerta que le interesa
al usuario
                        if((limitGate>=0)&&(limitGate<=9)){
                              string puerta=rdr[indice].ToString();
                              string[] arr = puerta.Split('/');
                              Variables.latPuerta=arr[0];
                              Variables.lonPuerta=arr[1];
                        }
                        //Guardamos los parametros obtenidos en la clase
Variables
                        Variables.estadio=this.estadio;
                        Variables.direccion=this.direccion;
                        Variables.latitude=this.latitud;
                        Variables.longitude=this.longitud;
                  }
                  //rdr.Dispose();
            }catch(Exception ex){
                  Debug.Log(ex.ToString());
            }finally
            {
                  //se cierra el reader y la conexion
                  rdr.Close();
                  con.Close();
            }
      }
}
```

==========

4. *Clase Variables:*

```
using UnityEngine;
using System.Collections;

public class Variables: MonoBehaviour {

      public static string estadio;
      public static string direccion;
      public static float latitude;
      public static float longitude;
      public static string puerta;
      public static string latPuerta;
      public static string lonPuerta;
}
```

==========

5. *Clase CSharpToJava:*

```
using UnityEngine;
using System.Collections;

public class CSharpToJava : MonoBehaviour {

      // Use this for initialization
      void Start () {
            //Clase CSharp que se llama al cargar el objeto al que esta
asociado esta clase
            estadioEncontradoC ();
      }

      public static void estadioEncontradoC(){
            //Llamada a la clase Java Unity Player del paquete
com.unity3d.player
            using (AndroidJavaClass cls_UnityPlayer = new
AndroidJavaClass("com.unity3d.player.UnityPlayer")) {
                  //Creacion del objeto
                  using (AndroidJavaObject obj_Activity =
cls_UnityPlayer.GetStatic<AndroidJavaObject>("currentActivity")) {
                        //Llamada a la clase MainActivity del paquete
com.TFG.ParteIni
                        AndroidJavaClass cls_CoarseLocation = new
AndroidJavaClass("com.TFG.ParteIni.MainActivity");
                        //Llamada al metodo estadioEncontradoJ de la clase
MainActivity y se pasan los parametros deseados

      cls_CoarseLocation.CallStatic("estadioEncontradoJ",
Variables.estadio,Variables.direccion,Variables.latitude,Variables.longit
ude,Variables.puerta,Variables.latPuerta,Variables.lonPuerta);

                  }
            }
      }
}
```

II. Anexo : Clases y métodos fase 2

1. *ActivityFase2*

```
package com.manero.fase_v2;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

import org.json.JSONObject;

import android.content.Intent;
import android.graphics.Color;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.util.Log;
import android.view.Menu;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.GoogleMap.OnMapClickListener;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.Polyline;
import com.google.android.gms.maps.model.PolylineOptions;

import com.TFG.ParteFinal.*;

public class ActivityFase2 extends FragmentActivity implements
LocationListener {
	float lat,lng,latPuerta,lngPuerta;
	GoogleMap map;
	ArrayList<LatLng> markerPoints;
	LatLng mOrigin;
	LatLng mDestination;
	LatLng puerta;
	Polyline mPolyline;
	Marker markerOrigin,markerDestination;
```

```
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);

		// Getting reference to SupportMapFragment of the
activity_main
		SupportMapFragment fm =
(SupportMapFragment)getSupportFragmentManager().findFragmentById(R.id.map
);

		// Getting Map for the SupportMapFragment
		map = fm.getMap();

		if(map!=null){

			// Enable MyLocation Button in the Map
			map.setMyLocationEnabled(true);

			// Setting onclick event listener for the map
			map.setOnMapClickListener(new OnMapClickListener() {

				@Override
				public void onMapClick(LatLng point) {


				}
			});
		}
		//una vez creado el mapa creamos el marker destino y la ruta
hasta el
		//Intent intent= getIntent();
		Bundle extras = getIntent().getExtras();
		//float lat=intent.getFloatExtra("lat",42.796813f );
		lat=extras.getFloat("lat");
		lng=extras.getFloat("lng");//por defecto cargamos la
direccion del sadar si esto falla,NO va a pasar nunca
		String stringLatPuerta=extras.getString("latPuerta");
		String stringLngPuerta=extras.getString("lngPuerta");
		latPuerta=Float.parseFloat(stringLatPuerta);
		lngPuerta=Float.parseFloat(stringLngPuerta);
		puerta=new LatLng(latPuerta,lngPuerta);
		loadStadiumLocation(new LatLng(lat,lng));
		// Initializing
		markerPoints = new ArrayList<LatLng>();

		createLocationListener();

	////////////////////////////////////////////////////////////////
///
		/*   fase3Intent = new Intent (this,ActivityFase3.class);
	    fase3Intent .putExtra("lat", latitude);
	fase3Intent .putExtra("lng", longitude);
	startActivity(fase2Intent);*/

	////////////////////////////////////////////////////////////////
///
```

```
	}


	private void updatePath(){

	}

	private void loadCurrentPosition(LatLng point){

		mOrigin=point;
		map.moveCamera(CameraUpdateFactory.newLatLng(point));
		map.animateCamera(CameraUpdateFactory.zoomTo(17));

		// Creating MarkerOptions
		MarkerOptions options = new MarkerOptions();
		// Setting the position of the marker
		options.position(point);
		//the color

	options.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptor
Factory.HUE_GREEN));
		// Add new marker to the Google Map Android API V2
		markerOrigin= map.addMarker(options);
		//CORREGIR
		String url = getDirectionsUrl(mOrigin, mDestination);
		DownloadTask downloadTask = new DownloadTask();
		// Start downloading json data from Google Directions API
		downloadTask.execute(url);
	}

	private void loadStadiumLocation(LatLng stadium_point){
		mDestination=stadium_point;
		// Getting URL to the Google Directions API
		MarkerOptions options = new MarkerOptions();
		// Setting the position of the marker
		options.position(stadium_point);

	options.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptor
Factory.HUE_RED));
		markerDestination=map.addMarker(options);
	}

	private void createLocationListener(){
		LocationManager locationManager = (LocationManager)
getSystemService(LOCATION_SERVICE);
	  Location location =
locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
	  if(location!=null){
		onLocationChanged(location);
	  }

locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER,
5000, 3, this);
	}

	private String getDirectionsUrl(LatLng origin,LatLng dest){
```

```
            // Origin of route
            String str_origin =
"origin="+origin.latitude+","+origin.longitude;

            // Destination of route
            String str_dest =
"destination="+dest.latitude+","+dest.longitude;


            // Sensor enabled
            String sensor = "sensor=false";

            // Building the parameters to the web service
            String parameters = str_origin+"&"+str_dest+"&"+sensor;

            // Output format
            String output = "json";

            // Building the url to the web service
            String url =
"https://maps.googleapis.com/maps/api/directions/"+output+"?"+parameters;


            return url;
      }

      /** A method to download json data from url */
    private String downloadUrl(String strUrl) throws IOException{
        String data = "";
        InputStream iStream = null;
        HttpURLConnection urlConnection = null;
        try{
                URL url = new URL(strUrl);

                // Creating an http connection to communicate with url
                urlConnection = (HttpURLConnection) url.openConnection();

                // Connecting to url
                urlConnection.connect();

                // Reading data from url
                iStream = urlConnection.getInputStream();

                BufferedReader br = new BufferedReader(new
InputStreamReader(iStream));

                StringBuffer sb  = new StringBuffer();

                String line = "";
                while( ( line = br.readLine())  != null){
                        sb.append(line);
                }

                data = sb.toString();

                br.close();
```

```
            }catch(Exception e){
                    Log.d("Exception while downloading url", e.toString());
            }finally{
                    iStream.close();
                    urlConnection.disconnect();
            }
            return data;
      }


      // Fetches data from url passed
      private class DownloadTask extends AsyncTask<String, Void, String>{


            // Downloading data in non-ui thread
            @Override
            protected String doInBackground(String... url) {

                  // For storing data from web service
                  String data = "";

                  try{
                        // Fetching the data from web service
                        data = downloadUrl(url[0]);
                  }catch(Exception e){
                        Log.d("Background Task",e.toString());
                  }
                  return data;
            }

            // Executes in UI thread, after the execution of
            // doInBackground()
            @Override
            protected void onPostExecute(String result) {
                  super.onPostExecute(result);

                  ParserTask parserTask = new ParserTask();

                  // Invokes the thread for parsing the JSON data
                  parserTask.execute(result);

            }
      }

      /** A class to parse the Google Places in JSON format */
    private class ParserTask extends AsyncTask<String, Integer,
List<List<HashMap<String,String>>> >{

      // Parsing the data in non-ui thread
            @Override
            protected List<List<HashMap<String, String>>>
doInBackground(String... jsonData) {

                  JSONObject jObject;
```

```
                List<List<HashMap<String, String>>> routes = null;


            try{
                  jObject = new JSONObject(jsonData[0]);
                  DirectionsJSONParser parser = new
DirectionsJSONParser();

                  // Starts parsing data
                  routes = parser.parse(jObject);
            }catch(Exception e){
                  e.printStackTrace();
            }
            return routes;
            }

            // Executes in UI thread, after the parsing process
            @Override
            protected void onPostExecute(List<List<HashMap<String,
String>>> result) {

                  ArrayList<LatLng> points = null;
                  PolylineOptions lineOptions = null;
                  MarkerOptions markerOptions = new MarkerOptions();

                  // Traversing through all the routes
                  for(int i=0;i<result.size();i++){
                        points = new ArrayList<LatLng>();
                        lineOptions = new PolylineOptions();

                        // Fetching i-th route
                        List<HashMap<String, String>> path =
result.get(i);

                        // Fetching all the points in i-th route
                        for(int j=0;j<path.size();j++){
                              HashMap<String,String> point = path.get(j);


                              double lat =
Double.parseDouble(point.get("lat"));
                              double lng =
Double.parseDouble(point.get("lng"));
                              LatLng position = new LatLng(lat, lng);

                              points.add(position);

                        }

                        // Adding all the points in the route to
LineOptions
                        lineOptions.addAll(points);
                        lineOptions.width(4);
                        lineOptions.color(Color.RED);

                  }
```

```
                // Drawing polyline in the Google Map for the i-th
route
                mPolyline=map.addPolyline(lineOptions);

            }
        }

        private void cleanMap(){
          //quito la ruta
          if(mPolyline!=null){
                mPolyline.remove();
          }
          //y los marquers
          if(markerOrigin!=null){
                markerOrigin.remove();
          }
          if(markerDestination!=null){
                markerDestination.remove();
          }
        }

          @Override
          public boolean onCreateOptionsMenu(Menu menu) {
                // Inflate the menu; this adds items to the action bar if it
is present.
                getMenuInflater().inflate(R.menu.main, menu);
                return true;
          }

          @Override
          public void onLocationChanged(Location location) {
                // TODO Auto-generated method stub
                cleanMap();
                double latitude = location.getLatitude();
               double longitude = location.getLongitude();
               LatLng latLng = new LatLng(latitude, longitude);


               loadCurrentPosition(latLng);

               Location destino= new Location("Mock");
               float desLat=(float) mDestination.latitude;
               float desLon=(float) mDestination.longitude;
               destino.setLatitude(desLat);
               destino.setLongitude(desLon);
               destino.setAccuracy(3.0f);
               float distancia=location.distanceTo(destino);
                 // Si la distancia entre donde estamos y el punto de destino
          es menor que 50m, cargamos la siguiente fase, fase 3.
               if(distancia<50){
                 Intent fase3Intent = new Intent(  this,
SampleCamActivity.class);
                 fase3Intent .putExtra("latPuerta", latPuerta);
                 fase3Intent .putExtra("lngPuerta", lngPuerta);
                 startActivity(fase3Intent);
               }
          }
```

```
	@Override
	public void onProviderDisabled(String provider) {
		// TODO Auto-generated method stub

	}

	@Override
	public void onProviderEnabled(String provider) {
		// TODO Auto-generated method stub

	}

	@Override
	public void onStatusChanged(String provider, int status, Bundle
extras) {
		// TODO Auto-generated method stub

	}
	}
```

III. Anexo : Clases y métodos fase 3

1. *Clase SampleCamActivity*

```
package com.TFG.ParteFinal;

import java.io.File;
import java.io.IOException;

import org.json.JSONException;
import org.json.JSONObject;

import android.hardware.SensorManager;
import android.location.LocationListener;
import android.os.Bundle;
import android.util.Log;
import android.widget.Toast;

import com.wikitude.architect.ArchitectView.ArchitectUrlListener;
import com.wikitude.architect.ArchitectView.SensorAccuracyChangeListener;
import com.wikitude.sdksamples.R;

public class SampleCamActivity extends AbstractArchitectCamActivity {
	public float lat, lng;
	/**
	 * extras key for activity title, usually static and set in
Manifest.xml
	 */
	protected static final String titulo = "ARTicket";

	/**
	 * extras key for architect-url to load, usually already known
upfront, can be relative folder to assets (myWorld.html -->
assets/myWorld.html is loaded) or web-url
("http://myserver.com/myWorld.html"). Note that argument passing is only
possible via web-url
	 */
	protected static final String url = "samples"
				+ File.separator + "System(Geo)"
				+ File.separator + "index.html";


	/**
	 * last time the calibration toast was shown, this avoids too many
toast shown when compass needs calibration
	 */
	private long lastCalibrationToastShownTimeMillis =
System.currentTimeMillis();

	@Override
	public String getARchitectWorldPath() {
		Log.i("asdasd",url);
```

```
            return url;
      }

      @Override
      public String getActivityTitle() {
            return titulo;
      }

      @Override
      public int getContentViewId() {
            return R.layout.sample_cam;
      }

      @Override
      public int getArchitectViewId() {
            return R.id.architectView;
      }

      @Override
      public String getWikitudeSDKLicenseKey() {
            return WikitudeSDKConstants.WIKITUDE_SDK_KEY;
      }

      @Override
      public SensorAccuracyChangeListener getSensorAccuracyListener() {
            return new SensorAccuracyChangeListener() {
                  @Override
                  public void onCompassAccuracyChanged( int accuracy ) {
                        /* UNRELIABLE = 0, LOW = 1, MEDIUM = 2, HIGH = 3
*/
                        if ( accuracy <
SensorManager.SENSOR_STATUS_ACCURACY_MEDIUM && SampleCamActivity.this !=
null && !SampleCamActivity.this.isFinishing() &&
System.currentTimeMillis() -
SampleCamActivity.this.lastCalibrationToastShownTimeMillis > 5 * 1000) {
                              Toast.makeText( SampleCamActivity.this,
R.string.compass_accuracy_low, Toast.LENGTH_LONG ).show();

      SampleCamActivity.this.lastCalibrationToastShownTimeMillis =
System.currentTimeMillis();
                        }
                  }
            };
      }

      @Override
      public ArchitectUrlListener getUrlListener() {
            return new ArchitectUrlListener() {

                  @Override
                  public boolean urlWasInvoked(String uriString) {
                        // by default: no action applied when url was
invoked
                        return false;
                  }
            };
      }
```

```
	@Override
	public ILocationProvider getLocationProvider(final LocationListener
locationListener) {
		return new LocationProvider(this, locationListener);
	}

	@Override
	public float getInitialCullingDistanceMeters() {
		// you need to adjust this in case your POIs are more than
50km away from user here while loading or in JS code (compare
'AR.context.scene.cullingDistance')
		return
ArchitectViewHolderInterface.CULLING_DISTANCE_DEFAULT_METERS;
	}

	//Metodo utilizado para la transferencia de parámetros a la
JavaScript
	@Override
	protected void onPostCreate( final Bundle savedInstanceState ) {
		super.onPostCreate( savedInstanceState );

		if ( this.architectView != null ) {

			// call mandatory live-cycle method of architectView
			this.architectView.onPostCreate();

			try {
				// load content via url in architectView, ensure
'<script src="architect://architect.js"></script>' is part of this HTML
file, have a look at wikitude.com's developer section for API references
				this.architectView.load(
this.getARchitectWorldPath() );
				//Se crea un nuevo objeto de tipo JSON
				JSONObject json= new JSONObject();
				//Se introduce en el objeto datos de tipo
llave/valor
				json.put("lat",lat);
				json.put("lng", lng);
				//Se pasan los parámetros JSON al JavaScript
utilizado en el ArchitectView
				//es decir, al JavaScript que llama el asset
index.html
				this.architectView.callJavascript("newData('"
+json+"')");
				if (this.getInitialCullingDistanceMeters() !=
ArchitectViewHolderInterface.CULLING_DISTANCE_DEFAULT_METERS) {
					// set the culling distance - meaning: the
maximum distance to render geo-content
					this.architectView.setCullingDistance(
this.getInitialCullingDistanceMeters() );
				}

			} catch (IOException e1) {
				e1.printStackTrace();
			} catch (JSONException e) {
				// TODO Auto-generated catch block
```

```
                        e.printStackTrace();
                    }
                }
            }
            @Override
            public void onCreate(Bundle savedInstanceState){
                super.onCreate(savedInstanceState);
                Bundle extras = getIntent().getExtras();
                lat=extras.getFloat("latPuerta");
                lng=extras.getFloat("lngPuerta");
            }

}
================================================================================
```

2. *Geo.js*

```
var lat,lng;
//Funcion para la recogida de datos JSON
function newData(json){
      //Recogemos los datos de tipo JSON
      var response=JSON.parse(json);
      //Almacenamiento de datos para su posterior uso
      lat=response.lat;
      lng=response.lng;
}
var Estadio = {
    planetsInfo: null,

    init: function() {

            //Creamos un objeto de geolocalizacion
        var geoLocation = new AR.GeoLocation(lat, lng);

            //Creamos un objeto que contiene una geolocalizacion relativa
al objeto geoLocation
            //10m north, 10m east, 10m altitude
        var locationSadar = new AR.RelativeLocation(geoLocation, 10, 10,
10);

        //Obtenemos la imagen
            var arrowImg = new AR.ImageResource("assets/arrow.png");
            //Creamos un drawable con la imagen
        var arrowDrawable= new AR.ImageDrawable(arrowImg, 10);


            //Creamos el geoObjeto y retocamos algunas de sus opciones
            geoObject = new AR.GeoObject(locationSadar);
        geoObject.drawables.cam=arrowDrawable;
        geoObject.enabled=true;

        var indicatorImg = new AR.ImageResource("assets/indi.png");

        // Add indicator to geoObject
```

```
        var indicatorDrawable = new AR.ImageDrawable(indicatorImg, 0.3, {
verticalAnchor: AR.CONST.VERTICAL_ANCHOR.TOP });
        geoObject.drawables.addIndicatorDrawable(indicatorDrawable);
    }
};

Estadio.init();
```

IV. Anexo : Pasos exportación Unity - Eclipse

1. Create a Vuforia Unity project using the Unity Editor.
2. Build the project for the Android platform; when doing so, open the "Player settings" and take note of the package name that you have chosen as your "Bundle Identifier" (e.g., "com.my.org") and the Android API level (e.g., 2.3 "Gingerbread"), as you will reuse these settings later.

   Se debe cambiar el nombre del paquete (por defecto, com.Company.ProductName), en caso contrario, obtendremos un error de compilación más tarde en Eclipse IDE sobre los ficheros Dex.

3. Save the APK file when asked by Unity, giving it the name you want (name is not relevant).
4. Use your file explorer to browse to your Unity project directory, where there is a folder called "/Temp" containing a subfolder called "/StagingArea." Open the StagingArea folder to see a set of files and directories (such as /res, /assets, /bin, AndroidManifest.xml,...) as in a typical Android project.
5. Copy the content of the StagingArea folder to another location on your file system (for instance, a location like "C:\Development\Android\Unity\MyVuforiaUnityProject" or similar). Any location is fine, as long as it is not under the original Temp folder of your Unity project.
6. Open Eclipse, and create a new Android project "from existing source"; click **Browse** and point to the folder where you just copied the StagingArea content (see last step).
7. Give the project a name, for instance, "QCARUnity."

   El nombre es irrelevante, pero es útil a la hora de seguir los pasos sin equivocación.

8. Right-click the **QCARUnity** project, select **Properties**, select **Android** and tick the "IsLibrary" checkbox (i.e., make the project a library); press **Apply/OK**.
9. Create another new project and name it, for instance, "QCARJava"; be sure to select the same package name and the same API level that you previously selected when building your Unity project with the Unity Editor.
10. Right-click the "QCARJava" project, select **Properties > Java build Path > Libraries**, click **Add External Jars...** and point to Editor\Data\PlaybackEngines\androidplayer\bin\classes.jar under your Unity installation directory, then press **OK**.

    Puede pasar que el directorio que se comenta no sea exactamente el mismo, en cuyo caso, se debe realizar una búsqueda en "Editor\Data\PlaybackEngines\androidplayer\" del archivo "clasess.jar".

    En el caso particular de este proyecto, el directorio elegido fue: "Editor\Data\PlaybackEngines\androidplayer\development\bin\classes.jar"

11. Right-click **QCARJava** again and select **Properties > Android**, scroll the page down until you see a Reference Projects table, and add the "QCARUnity" project to the list of referenced projects.
12. Move the "/assets" and "/libs" folders from the "QCARUnity" project directory to the "QCARJava" project folder.

Este paso se debe hacer en el explorador del sistema operativo, de no ser así, Eclipse suele poner problemas al comando mover (distinto de copiar)

13. Copy Vuforia.jar and QCARUnityPlayer.jar from the "/plugins" directory of the "QCARUnity" project to the "/libs" folder of the "QCARJava" project.
14. Copy the "/QCAR" folder (this should contain your Dataset .XML/.DAT files) from the "/raw" directory of the "QCARUnity" project to the "/assets" folder of the "QCARJava" project.
15. Copy the AndroidManifest.xml file from "QCARUnity" to the "QCARJava" project (overwrite the existingAndroidManifest.xml file).

Se recomienda realizar el paso 13, 14 y 15 en el explorador.