UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

Grado en Ing. Sist. de Telecom., Sonido e Imagen

UNIVERSIDAD
POLITECNICA
DE VALENCIA

"Desarrollo de un aplicación de Realidad Aumentada, orientada a la gestión de servicios e información en un Smart Place"

TRABAJO FINAL DE GRADO

Autor/a:
Pedro Pérez Palazón

Tutor/a:
Fernando Boronat Seguí
Mario Alberto Montagut Climent

GANDIA, 2016

Resum:

L'objectiu d'este projecte és el d'investigar en el camp de la Realitat Augmentada i desenrotllar una aplicació, basada en este tipus de tecnologia, orientada a la gestió de servicis i informació en un "Smart Place" (Smart City, Smart Port, Smart Campus) .

En concret, este TFG s'ha s'ha centrat en el cas d'un Smart Campus, desenrotllant una aplicació de RA per a este tipus d'entorns, en particular, per al Campus de Gandia de la UPV, amb dos modes d'ús: mode exploració (per a visualitzar continguts virtuals) i mode navegació (per a guiar l'usuari pel Campus) .

Un Smart Campus o "Campus Inteligente", es pot definir com aquell que és capaç d'aprofitar les dades que es produïxen en el seu funcionament diari, per a generar nova informació que li permeta millorar la seua gestió i ser més sostenible, més competitiu i oferir millor qualitat de servici i benestar, gràcies a la participació i col·laboració de tota la comunitat universitària.

Però, Quin paper juga l'aplicació de RA? Este tipus d'aplicació és un ferramenta idònia per a contribuir amb la idea de Smart Places: per mitjà de l'ús d'este tipus d'aplicacions, l'usuari podrà orientar-se amb major facilitat en l'entorn del campus i tindre a la seua disposició multitud de continguts virtuals complementaris als mitjans tradicionals (imatges, vídeos, models 3D) , relacionats amb la informació, els servicis i les activitats que el campus posa a disposició de l'usuari, inclús amb la pròpia activitat acadèmica dels estudiants. Gràcies a l'anàlisi de l'ús d'esta aplicació, es poden extraure conclusions per a millorar la gestió i qualitat dels servicis oferits pel Smart Campus.

Paraules Clau

Smart Campus, Aplicació mòbil, RA (Realitat Augmentada) , Contingut virtual.

Resumen:

El objetivo de este Trabajo Fin de Grado (TFG) es el de investigar en el campo de la Realidad Aumentada (RA) y desarrollar una aplicación, basada en este tipo de tecnología, orientada a la gestión de servicios e información en un "Smart Place" (Smart City, Smart Port, Smart Campus).

En concreto, este TFG se ha se ha centrado en el caso de un Smart Campus, desarrollando una aplicación de RA para este tipo de entornos, en particular, para el Campus de Gandía de la UPV, con dos modos de uso: modo exploración (para visualizar contenidos virtuales) y modo navegación (para guiar al usuario por el Campus).

Un Smart Campus o "Campus Inteligente", se puede definir como aquél que es capaz de aprovechar los datos que se producen en su funcionamiento diario, para generar nueva información que le permita mejorar su gestión y ser más sostenible, más competitivo y ofrecer mejor calidad de servicio y bienestar, gracias a la participación y colaboración de toda la comunidad universitaria.

En este contexto, una aplicación de RA es un herramienta idónea para contribuir con la idea de Smart Places: mediante el uso de este tipo de aplicaciones, el usuario podrá orientarse con mayor facilidad en el entorno del campus y tener a su disposición multitud de contenidos virtuales complementarios a los medios tradicionales (imágenes, videos, modelos 3D), relacionados con la información, los servicios y las actividades que el campus pone a disposición del usuario, incluso con la propia actividad académica de los estudiantes. Gracias al análisis del uso de esta aplicación, se pueden extraer conclusiones para mejorar la gestión y calidad de los servicios ofrecidos por el Smart Campus.

Palabras Clave

Smart Campus, Aplicación móvil, RA (Realidad Aumentada), Contenido virtual.

Abstract:

The objective of this project is to investigate the field of Augmented Reality and to develop an application based on this technology, oriented to services and information management in a "Smart Place" (Smart City, Smart Port, Smart Campus,). This project focuses on the case of Smart Campus, so the application developed will be a propose for cover the EPSG Campus, thanks for two modes of app: exploration mode (for visualize virtual content) and navigation mode (to guide the user).

A Smart Campus or “Intelligent Campus”, can be defined as one that is able to leverage the data that are produced in their daily operations to generate new information that could improve their management and be more sustainable, competitive and to offer better quality of service and welfare thanks to the participation and collaboration of the entire university community.

But, what kind of role does the implementation of AR? This type of application is a suitable tool to contribute to the idea of Smart Campus: by using this application, users can orient themselves more easily in the campus environment and they could visualize a multitude of virtual content (audio, images, video, 3D models) that it related to information, services and activities that the university offers to users, even academic student activity itself. By analyzing the use of this application, will be possible to draw conclusions to improve the management and quality of services offered by the Smart Campus.

Key Words

Smart Campus, Mobile application, AR (Augmented Reality), Virtual content

Índice

1. Introducción

Según Borko Furth (Furht 2011), la **Realidad Aumentada (RA)** (Wikipedia.org 2001) se puede definir como la vista directa o indirecta de un entorno real, que ha sido realzada o aumentada añadiendo información virtual generada por ordenador. La tecnología de RA permite tanto la interactividad con el entorno como la presentación de contenidos 3D, ya que se incluyen tanto objetos reales como virtuales. El concepto de "realidad virtual continua" (RV), es definido por Paul Milgram y Fumio Kishino (P. Milgram 1994) como un continuo que comprende desde el entorno real hasta el entorno virtual (EV), compuesto de RA y Virtualidad Aumentada (VA): donde la RA se encuentra limitada por el mundo real y la VA por el entorno virtual, como se puede observar en la Ilustración 1:

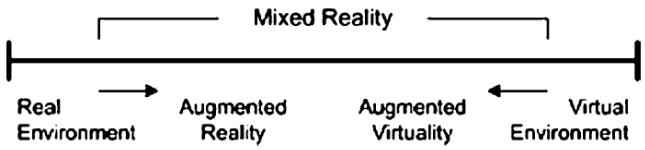


Ilustración 1. Realidad virtual continua (P. Milgram, "Taxonomy of Mixed Reality Visual Displays").

La RA tiende a simplificar la vida del usuario proporcionándole información, no sólo en las inmediaciones circundantes a él, sino también en cualquier vista indirecta del entorno real. La RA mejora la percepción del usuario y la interacción con el mundo real. Mientras que la tecnología de **Realidad Virtual** (RV[1]), o **Entorno Virtual** (EV) definidos por Milgram (P. Milgram 1994), sumerge completamente al usuario en un mundo sintético sin ver el mundo real (Ilustración 2), la tecnología de RA aumenta la sensación de realidad gracias a la superposición de objetos virtuales y señales sobre la visión del mundo, en tiempo real.

Ilustración 2. Escena de un Entorno Virtual (http://www.unciencia.unc.edu.ar).

Se debe tener en cuenta, como Azuma y otros afirman (Ronald Azuma 2001), que no se puede considerar la RA restringida al uso concreto de un tipo de tecnología para mostrar el contenido, como los *displays* adaptados a la cabeza a modo de "gafas" (*HMD*[2]*, Head-Mounted Display*), que limitan la percepción del usuario al sentido de la vista.

[1] https://en.wikipedia.org/wiki/Virtual_reality

[2] https://en.wikipedia.org/wiki/Head-mounted_display

La RA puede aplicarse a todos los sentidos: olores aumentados, audio, incluso tacto. Este tipo de tecnología puede también ser usada para aumentar o substituir sentidos perdidos por los usuarios, gracias a la sustitución sensorial, como por ejemplo, mejorar la percepción de personas ciegas[3] o con poca calidad de visión, por medio de señales auditivas, o mejorar la percepción auditiva de usuarios sordos por medio del uso de señales visuales.

Azuma también considera (Ronald Azuma 2001) las aplicaciones de RA que requieren de la eliminación de objetos reales del entorno, comúnmente llamadas mediadas o reducidas (*mediated or disminished*), además de la inclusión de objetos virtuales.

La eliminación objetos del mundo real corresponde, por ejemplo, a cubrir los objetos con información virtual que hace de fondo, para proporcionar al usuario la impresión de que el objeto no está ahí. Los **objetos virtuales** añadidos al entorno real muestran información al usuario, que éste no puede detectar directamente con sus sentidos. La información mostrada por el objeto virtual puede ayudar al usuario en la realización de sus tareas cotidianas, por ejemplo, guiar a los trabajadores a través de la líneas eléctricas en una aeronave, por medio de la representación digital de información virtual, con el dispositivo adecuado. Asimismo, la aplicación puede simplemente tener un propósito de entretenimiento (p.ej., videojuegos), médico, publicitario, para mantenimiento y reparación, etc[4].

Éste Trabajo Final de Grado se centra en el uso de la RA, para cubrir de contenidos virtuales auxiliares un *SmartCampus* (caso particular de *Smart Place*) donde el usuario pueda obtener ayuda, información y servicios por medio del uso de una aplicación de RA desarrollada, denominada **Pocket Campus – Augmented EPSG**.

1.1. Smart Campus

En esta sección, se profundizará en el concepto de Smart Place y, en especial, en el caso concreto de un SmartCampus, ya que una de las principales motivaciones de este TFG es la de contribuir al proyecto de Smart Campus, con el uso de la aplicación de RA desarrollada. Un Smart Place[5] o entorno inteligente, es aquel entorno capaz de aprovechar los datos que produce en su funcionamiento diario para generar información nueva, que le permita mejorar su gestión y ser más sostenible, más competitivo y ofrecer mejor calidad de servicio y bienestar, gracias a la participación y colaboración de todos los usuarios o participantes del entorno (UCAM 2016).

En un Smart Campus[6] (Ilustración 3), el entorno a cubrir es el campus universitario, y los usuarios o participantes son aquellos estudiantes, profesores, personal del centro y visitantes que disfruten de la actividad del campus y de los servicios del campus. Para contribuir con la propuesta de Smart Campus, es importante establecer una estrategia centrada en el uso intensivo de las "Nuevas Tecnologías" (NFC, RA, RV), en un contexto de absoluto respeto por el medio ambiente. Para conseguir esto, es necesario perseguir varias premisas: la implantación de redes y nuevos sistemas tecnológicos avanzados, garantizar la conectividad e interconexión y la puesta en marcha de acciones sostenibles.

---

[3] http://www.elconfidencial.com/tecnologia/2013-12-06/unas-gafas-de-realidad-aumentada-para-ciegos-sustituiran-al-baston-y-al-perro-guia_63052/

[4] https://moverio.epson.com/jsp/pc/pc_application_list.jsp

[5] http://www.upv.es/contenidos/SMARTALC/

[6] http://smart.uji.es

Ilustración 3. Ejemplo ilustrativo de un Smart Campus (http://www.bulldogsolutions.com/Portfolio/smart-campus).

La adquisición de datos y su procesado son las claves para dotar de inteligencia al sistema de gestión: mediante la sensorización (cubrir de sensores) del entorno, el análisis de datos sobre el uso de los servicios e información del entorno y la participación de los mismos usuarios, ya que su "*feedback"* es determinante.

1.2. Objetivos

El objetivo principal de este TFG, es el de **diseñar y desarrollar una aplicación de RA** que sirva para mostrar contenidos virtuales sobre servicios e información de interés y que, por tanto, sirva de ayuda para los usuarios o visitantes del Smart Campus – EPSG (entorno Smart del Campus de la Escuela Politécnica Superior de Gandia – UPV).

Además, se persiguen varios sub-objetivos u objetivos secundarios:

- Investigación sobre los **recursos y herramientas** actuales para desarrollar aplicaciones basadas en la tecnología de RA; aprendizaje tanto de conceptos generales de RA como del uso de algunos SDKs[7] (*Software Development Kit*) de RA disponibles en el mercado (p.ej., Wikitude, Vuforia, Kudan, etc).

- Planificación y diseño de la estrategia a seguir apropiada para desarrollar la app y dotarla de contenidos: definición de los casos de utilidad, tipos de información a proporcionar, modo de proporcionar dicha información, interacción con el usuario (servicios), *lookandfeel*[8] de la aplicación, etc.

[7] https://es.wikipedia.org/wiki/Kit_de_desarrollo_de_software

[8] https://es.wikipedia.org/wiki/Look_and_feel

- Implementación de dos modos de uso o funcionalidades para la app:

  a) **Modo exploración**: Para que el usuario pueda visualizar contenido virtual (modelos 3D, información 2D, videos, botones virtuales,…) en base a referencias o elementos físicos (*Targets*) repartidos por el *Smart Campus.*

  b) **Modo navegación**: Para guiar al usuario hacia los POIs (*Point of Interest*[9]), situados en el *Smart Campus.* Gracias al uso de POIs, el usuario podrá visualizar contenidos virtuales que se reproduzcan en base a una posición GPS (*Global Positioning System*), a diferencia del modo exploración que hace uso de *targets* para reproducir el contenido.

- Creación y/u obtención de **contenidos virtuales**, que sean prácticos, útiles y adaptados al caso real de *Smart Campus:* EPSG.

- Propuesta de evaluación subjetiva para la aplicación *"**Pocket Campus**".*

1.3. Motivaciones

Una de las motivaciones principales de este TFG, es la de ayudar a los visitantes y/o usuarios de un ***Smart Place*** (*Smart City*, *Smart Campus, Smart Port, Smart Hospital*[10], etc), a orientarse en el entorno, de una manera clara, intuitiva, interactiva e innovadora, así como proporcionarles información de interés de manera dinámica.

No hace mucho tiempo, cuando se realizaba un viaje hacia un destino desconocido, se utilizaba un mapa en papel, que en la mayoría de ocasiones resultaba tan tedioso de abrir como de comprender. Este arcaico sistema de navegación quedó obsoleto al evolucionar los sistemas GPS. De hecho, actualmente, la mayoría de personas están acostumbradas a usar dispositivos GPS o aplicaciones móviles, como *Google Maps*, para orientarse y guiarse, ya que estas herramientas facilitan en gran medida la tarea de navegación. De igual manera, la tecnología de RA puede contribuir a un cambio y mejora en el modo de orientarse y guiarse dentro de *Smart Places* (p.ej., aeropuertos, universidades, hospitales, parques de atracciones) ofreciendo información sobre servicios y puntos de interés propios.

Por ejemplo, se puede cambiar el tradicional mapa físico en 2D (ilustración 4) por un mapa 3D (ilustración 5) interactivo del entorno, y sustituir las señales o indicadores físicos por contenidos virtuales específicos, según el fin que persiga el usuario: ya sea para navegación y orientación o para obtener información útil del entorno.

[9] https://es.wikipedia.org/wiki/Punto_de_interés

[10] http://www.lifesmarthospital.eu

Ilustración 4. Mapa 2D del Campus de la EPSG (Captura web. Planos www.upv.es).

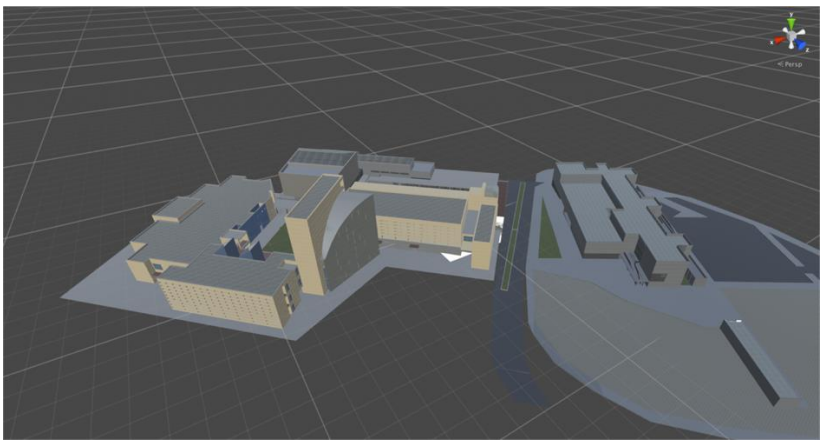


Ilustración 5. Modelo 3D para mapa del campus EPSG (Captura del modelo 3D de la EPSG desde Unity).

Por otro lado, con el uso de una aplicación de RA para un *Smart Place*, se posibilita que usuarios o visitantes tengan a su disposición multitud de contenidos virtuales, que conformen un **entorno virtual**. Estos contenidos virtuales puede ser de diferente índole: modelos 3D, animaciones, videos, contenidos 2D interactivo, etc Asimismo estos contenidos podrían complementar en gran medida los medios tradicionales para facilitar información a los usuarios, y revolucionar por completo el modo en el que se disfruta de algunos servicios, que el mismo *Smart Place* pone a disposición del usuario.

A continuación se citan dos ejemplos relevantes en un *Smart Campus*:

- La cartelería que publicita eventos a realizar en el campus puede ser complementada con elementos virtuales que añadan valor a la propuesta del evento y contribuyan a "enganchar" (*engagement*[11]) al usuario. Estos elementos podrían ser: un botón virtual que reproduzca un video de presentación del evento sobre la captura del cartel o un enlace a la web donde se realiza la inscripción, un modelo 3D animado de la mascota del evento que invita a asistir al usuario, etc.

- El servicio de Biblioteca del Campus podría beneficiarse enormemente del uso de este tipo de tecnología, por ejemplo, incluyendo la tecnología AR en el sistema de búsqueda de ejemplares tradicional y proporcionar información útil al usuario, con solo capturar con la cámara del dispositivo el ejemplar buscado.

[11] http://engagement.softwarecriollo.com

En definitiva, lo que motiva para la realización de este TFG es la propuesta de cubrir un entorno real con **contenidos virtuales**, creando así un entorno virtual, que complemente o mejore el modo de facilitar servicios al usuario y el modo de presentarle la información útil de interés. Para conseguir que esta propuesta llegue a buen puerto, este entorno virtual, paralelo al entorno real, debe de ser útil, sencillo de entender , fácil de usar y accesible para todos los visitantes del campus.

1.4. Estructura de la memoria

La memorias ha estructurado en cuatro capítulos, cuyo contenido se resume a continuación:

1.- **Introducción**: Este capítulo contiene el contexto, las motivaciones, requisitos y objetivos de este TFG.

2.-**Estado del arte:** En este capítulo se introducen los conceptos y métodos sobre los cuales se basa la tecnología de RA (Etapas de RA), y se presentan diferentes ejemplos de aplicaciones de RA, similares a la app que se va a desarrollar en este TFG.

3.- **Desarrollo de la aplicación**: Este capítulo contiene la parte más importante del TFG. En esta sección se describirán los pasos que se han seguido para desarrollar la app de RA: Planificación, Creación de contenidos virtuales, Diseño e implementación de la app.

4.- **Propuesta de Evaluación de la aplicación**: Por último, se propondrá un método de evaluación subjetivo para la app, basado en diferentes investigaciones de expertos en este ámbito del desarrollo de apps.

2. Estado del Arte

En este apartado se ofrece una visión y revisión general sobre la tecnología de **RA**. Se describen las diferentes etapas en el desarrollo de este tipo de tecnologías, los conceptos en los que se basan y sus aplicaciones. Como se ha comentado anteriormente, la RA se puede entender como un tipo de realidad, que el usuario percibe cuando se combina el entorno físico real y elementos virtuales superpuestos.

Esa RA es percibida gracias al uso de dispositivos capaces de mezclar la realidad física (por ejemplo, capturada con la cámara de un dispositivo) con contenidos virtuales (generados por ordenador), y mostrar el resultado al usuario, por medio de una pantalla o visor en el dispositivo.

Estos **dispositivos** pueden ser *smartphones*[12], *tablets*[13], gafas de realidad aumentada o *smartglasses*[14] (Ilustración 6), y, en general, cualquier dispositivo que cumpla con las características anteriormente descritas.

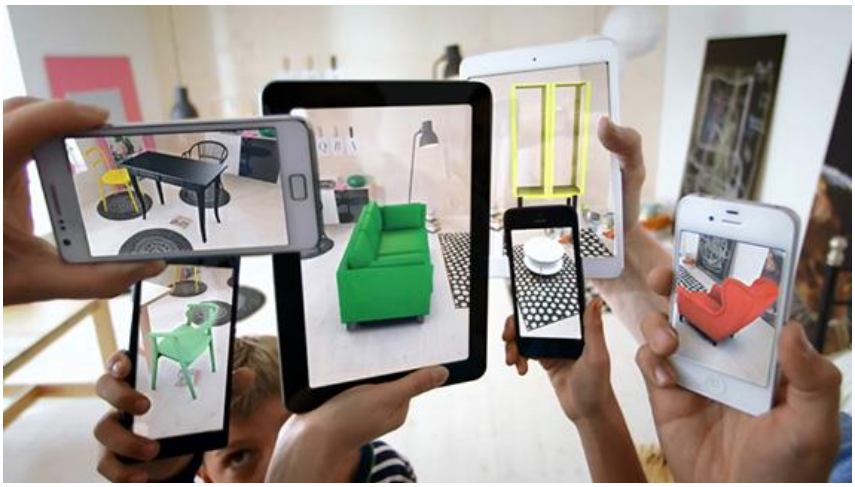


Ilustración 6. Dispositivos móviles para RA[15].

En la Ilustración 7, se puede un ejemplo de aplicación de RA. En este caso, la app muestra información relevante del entorno que rodea al usuario, mediante indicadores y cuadros de texto virtuales superpuestos a la visión del mundo real. En dicha ilustración, se pueden diferenciar dos tipos de elementos:

- **Elementos reales**: Edificios, vehículos, personas, etc (elementos del entorno real).

- **Elementos virtuales o no reales**: Iconos de redes sociales y notificaciones (parte izquierda superior), marcadores de información útil (sobre vehículos, personas y puntos de interés), sección de información contextual del entorno y sección de información meteorológica (parte inferior derecha).

[12] https://es.wikipedia.org/wiki/Teléfono_inteligente

[13] https://es.wikipedia.org/wiki/Tableta_(computadora)

[14] https://en.wikipedia.org/wiki/Smartglasses

[15] Fuente - Ilustración 6: http://ciencialaultima.blogspot.com.es/2015/01/realidad-aumentada-y-realidad-virtual.html

Estos dos tipos de elementos (reales y virtuales) se mezclan para enriquecer la realidad que percibe el usuario, como la que se muestra en la Ilustración 7. Por ello, una característica fundamental de toda aplicación de RA, es que se muestren contenidos reales y virtuales, con predominancia de los reales sobre los virtuales, ya que de lo contrario, la experiencia del usuario puede resultar caótica, estresante y/o inapropiada.

Ilustración 7. Elementos reales y virtuales en la visión de Realidad Aumentada[16].

Además de esta características, el concepto de RA también puede conllevar algunas otras propiedades concretas, como define Ronald Azuma (Ronald Azuma 2001). Según Azuma, toda aplicación de RA cumple tres características:

- Combina elementos reales o virtuales (fundamental).

- Es **interactiva** en tiempo real: la aplicación es capaz responder a la interacción del usuario, generando contenidos de manera simultánea a la visión del mundo real. Por ello, la introducción de elementos aumentados por ordenador en el cine no se puede considerar RA[17].

- La experiencia es en **3D**: Los contenidos virtuales superpuestos son en 3D, para mayor inmersión del usuario en la experiencia, por ello se llama Realidad Aumentada. Según esta definición, las animaciones en 2D o reproducción de videos aumentados no podrían considerarse ejemplos de RA.

[16] Fuente – Ilustración 7: https://carlosmatallana.files.wordpress.com/2011/03/realidad-aumentada.jpg

[17] http://www.pangeareality.com/la-realidad-aumentada-es-el-futuro-del-cine/

2.1. Etapas de la Realidad Aumentada (RA)

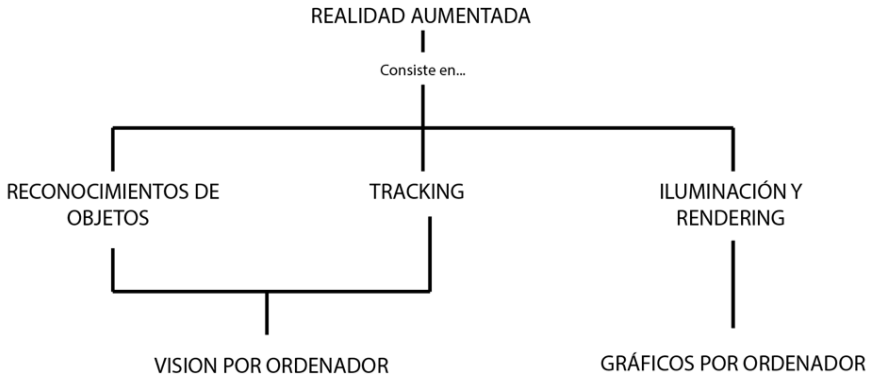


Ilustración 8. Esquema conceptual sobre etapas de la Realidad Aumentada (Creado con Adobe Illustrator).

La Realidad Aumentada, es el resultado de aplicar 3 etapas esenciales[18] (Ilustración 8):

- **Reconocimiento de objetos**.
- **Tracking** o seguimiento.
- **Iluminación y Rendering** (renderizado o representación de contenidos virtuales).

Por una parte, la visión por ordenador, también conocida como Visión Artificial, engloba tanto el reconocimiento de imágenes y/o objetos, como el tracking de estos elementos. Concretamente, consiste en desarrollar un software para que reconozca y "entienda" las características de una escena o imagen. Los objetivos principales de la visión artificial son:

- Detección, segmentación, localización y reconocimiento de ciertos elementos en la imagen (por ejemplo, reconocimiento facial o de escritura).
- Evaluación o interpretación de los resultados obtenidos (por ejemplo, segmentación y registro en un software capaz de contar monedas)
- Mapeo de escena reales (Tecnología SLAM[19] - *Simultaneous Location And Mapping*).

Por otra parte, la RA también engloba se trata el campo de los Gráficos por ordenador (CGI – Computer Generated Imagery), incluyendo la iluminación y renderizado (*rendering*[20]) que consiste en crear contenidos virtuales por ordenador y su integración en escenas reales, de manera eficaz.

[18] https://en.wikipedia.org/wiki/Augmented_reality

[19] https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping

[20] https://en.wikipedia.org/wiki/Rendering_(computer_graphics)

3. Creación de la app de RA: "Pocket Campus – Augmented EPSG"

En este capítulo se detalla, paso a paso, todo el proceso de creación de la app (diseño e implementación): desde la elección del software utilizado para el desarrollo de la aplicación, la creación de contenidos virtuales, el diseño y la implementación de la app en código, hasta el resultado final de la aplicación.

3.1. Requisitos

A la hora de establecer los requisitos necesarios para llevar a cabo este proyecto, se debe diferenciar entre requisitos técnicos y requisitos de aplicación.

- **Requisitos técnicos**: son aquellos que surgen cuando se hace referencia al tipo tecnología empleada. Estos, a a su vez, podrían clasificarse en:

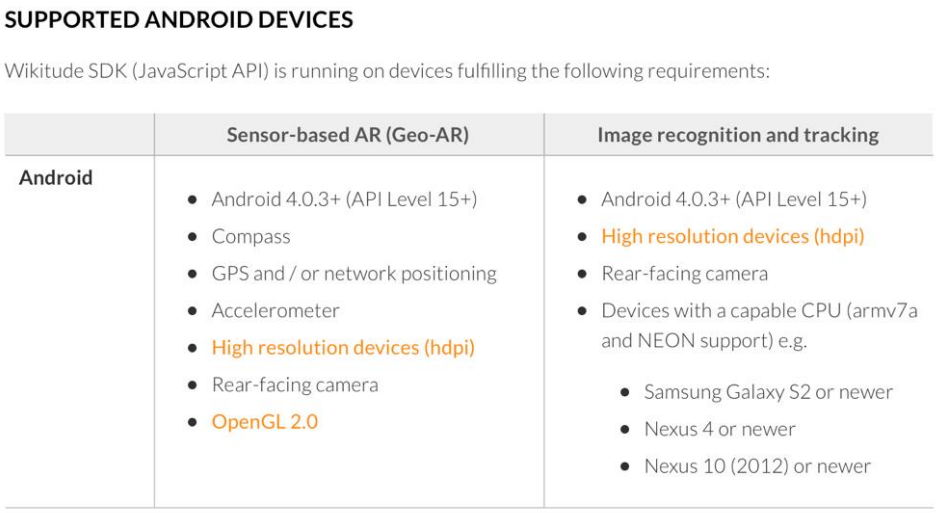
  a) De desarrollo: Características del *software* y del *hardware* necesarias para el desarrollo. Este tipo de requisitos dependen, en gran medida, del SDK de RA utilizado (en el caso de este TFG, Wikitude SDK – JavaScript API) para desarrollar la aplicación y del entorno de desarrollo de software que se use (en el caso de este TFG, *Android Studio*[21]). El SDK de Wikitude es una librería de *software* y un *framework* para aplicaciones de dispositivos móviles usadas para crear experiencias de RA.

  Para este TFG, tan sólo se necesita un ordenador en el que instalar estas herramientas de desarrollo y un dispositivo móvil (Smartphone, Tablet o Smart Glasses) con el cual poder testear las diferentes etapas de desarrollo, ya que, aunque exista la posibilidad de probarlo con la cámara integrada en el ordenador, es mucho más cómodo para el desarrollador y se obtiene una idea mucho más fidedigna de lo que sería la experiencia final del usuario.

  b) De ejecución: características especificas de los dispositivos móviles y del software, necesarias para que el usuario pueda ejecutar la aplicación en el dispositivo.

  Pocket Campus se debe ejecutar en un dispositivo móvil que tenga cámara integrada y que funcione con un sistema operativo Android, ya que por el momento se desarrollará únicamente para este sector de usuarios. No obstante, surgen otros requisitos de ejecución que dependen del SDK utilizado.

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



SUPPORTED ANDROID DEVICES

Wikitude SDK (JavaScript API) is running on devices fulfilling the following requirements:

	Sensor-based AR (Geo-AR)	Image recognition and tracking

Android	• Android 4.0.3+ (API Level 15+)
	• Compass
	• GPS and / or network positioning
	• Accelerometer
	• High resolution devices (hdpi)
	• Rear-facing camera
	• OpenGL 2.0
		• Android 4.0.3+ (API Level 15+)
		• High resolution devices (hdpi)
		• Rear-facing camera
		• Devices with a capable CPU (armv7a and NEON support) e.g.
		  • Samsung Galaxy S2 or newer
		  • Nexus 4 or newer
		  • Nexus 10 (2012) or newer

Ilustración 9. Requisitos del dispositivo móvil para ejecutar la app (Documentación en web de Wikitude).[22]

Al haber elegido la opción de **Wikitude**[23] como SDK utilizado para el desarrollo, será éste quien establezca los requisitos de ejecución. En la anterior ilustración, se pueden ver los requisitos que debe tener el dispositivo móvil para poder ejecutar la aplicación (en amarillo, los requisitos básico del dispositivo, y en gris, los recomendados para un correcto funcionamiento de la app desarrollada).

Es importante mencionar que la aplicación se ha diseñado pensando en su futura expansión hacia otros sistemas operativos (sobre todo a IOS), por lo que se ha elegido la versión en JavaScript de Wikitude, para el desarrollo de *Pocket Campus*. De esta manera, se ahorrará trabajo y tiempo, al poder reutilizar gran parte del trabajo desarrollado, correspondiente a la implementación de la escena de RA en si misma (*modoExploration.js* y *modoNavigation.js*).

- **Requisitos de la aplicación**: son las características generales que debe presentar la aplicación para garantizar un buen producto final: Pocket Campus – Augmented EPSG. Por ello, la aplicación debe ser:

  a) Intuitiva y accesible: Esto es esencial para que el producto tenga éxito, ya que va dirigida a todo tipo de público.

  b) Atractiva: De por sí, la tecnología de RA atrae al público que no está acostumbrado a este tipo de tecnologías, pero, aun así, se debe intentar que la experiencia sea lo más atractiva posible, cuidando al detalle el *lookandfee*l de la aplicación. Si se cumple este requisito, se garantiza que el usuario tenga intención de volver a probar la aplicación.

  c) Precisa: Los contenidos mostrados tienen que estar adecuadamente relacionados con el elemento real que complementan o, por ejemplo, bien situados si dependen de posicionamiento

[22] http://www.wikitude.com/documentation/

[23] http://www.wikitude.com

GPS. Cumplir con este parámetro conllevará que el usuario entienda de forma más clara y precisa la información o funciones que la aplicación presenta.

d) Rápida y fluida: La aplicación debe proporcionar una experiencia de usuario fluida, sin grandes tiempos de espera y lapsus en el *rendering* de los contenidos virtuales, ya que de esta forma, la experiencia aumentada presentará mayor grado de inmersión y, por lo tanto, resultará más placentera al usuario. Por ello, es muy importante escoger bien el SDK de RA utilizado para el desarrollo.

c) Interactiva: El usuario debe poder interactuar con la aplicación y no limitarse a observar el contenido. Se trata de un requisito muy importante, ya que influirá tanto en el grado de inmersión que el usuario experimenta, como en la atracción que sienta al vivir la experiencia (*engagement*).

3.2. Elección de la Plataforma de desarrollo

La aplicación está orientada al uso de todo tipo de público. Por ello, su desarrollo debe ir enfocado para su uso en dispositivos móviles comunes: tablets y smartphones. Es necesario, por tanto, conocer las diferentes plataformas que hay en el mercado. Las principales plataformas, en cuanto a volumen de mercado, son: Android, IOS, Windows Phone y BlackBerry 10 (Ilustración 10).

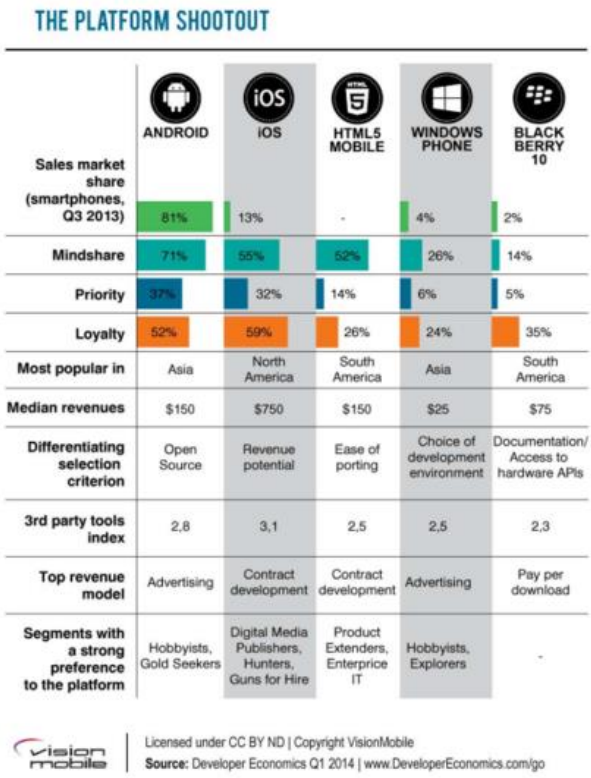


Ilustración 10. Comparativa de las principales plataformas de desarrollo[24].

[24] Fuente – Ilustración 10: http://www.xatakamovil.com/mercado/desarrollo-de-aplicaciones-moviles-i-asi-esta-el-mercado

a) ANDROID

Para el desarrollo de *Pocket Campus* se ha elegido la plataforma Android, ya que:

- Ofrece facilidades para el desarrollador: Código abierto y con un único pago para publicar la app en el *Play Store* de 25$.

- Es el sistema operativo más extendido del mundo y, por tanto, el alcance de mercado, resulta mayor. No se busca el beneficio económico, si no que el mayor número de usuarios pueda disfrutar de la app.

3.3. Elección del entorno de desarrollo

Actualmente, existen multitud de SDKs disponibles para desarrollar aplicaciones de RA. Un SDK es un conjunto de herramientas y/o programas que permiten al desarrollador crear aplicaciones para una determinada plataforma o sistema operativo. Otra definición de SDK podría ser la de conjunto de librerías que contienen el código utilizado más comúnmente y de forma repetitiva en el desarrollo de apps.

Las herramientas que proporciona un SDK para RA son, principalmente, recursos que simplifican la implementación de las siguientes funciones:

- *Recognition* o reconocimiento: función que proporciona la habilidad de comprender que es lo que esta capturando la cámara del dispositivo.

- *Tracking* o seguimiento: son los ojos de la aplicación, es decir, la función que permite no perder al target de vista.

- *Rendering* o renderizado de contenidos: donde se crean y representan los contenidos virtuales que el usuario final podrá visualizar.

La elección del SDK a utilizar en este TFG se ha basado en varios factores claves: las funcionalidades que se buscan con la utilización del mismo y a las plataformas a las que se pretende llegar, la facilidad de distribución de la app desarrollada, y ,el coste de la licencia que siempre resulta una variable a tener en cuenta. Todas estas características, y algunas más, se recogen en una tabla comparativa sobre los SDK disponibles en el mercado, en una entrada de la pagina web Social Compare[25].

A continuación, se detallan las características y especificaciones del SDK elegido para el desarrollo de ***PocketCampus – Augmented EPSG.***

a) WIKITUDE

Al igual que las otras opciones de SDK para RA, Wikitude presenta versiones tanto para IOS como para Android nativo y para IOS/Android en JavaScript. Además, cuenta con plugins para utilizar en otros entornos de desarrollo, como: Unity, Xamarin, Cordova y Ttanium; así como una versión del SDK para desarrollar aplicaciones para SmartGlasses, como pueden ser EPSON, Google Glass o Vuzix. Las características o funcionalidades que ofrece este SDK dependen de la plataforma de desarrollo que se va a utilizar:

[25] http://socialcompare.com/es/comparison/augmented-reality-sdks

- Para IOS/Android Nativo: *Client/Cloud Recognition* o reconocimiento desde el cliente (local) o desde el Cloud (servidor), y *tracking* tanto 2D como 3D (no se pueden combinar ambos).

- Para IOS/Android JavaScript: *Client/Cloud Recognition*, modelos 3D aumentados, capacidad de manejar POIs (*Point of Interest* o Punto de interés) en los que el *target* coincide con una posición GPS, *Video Drawables* (videos aumentados) y la posibilidad de combinar *Client Recognition* y POIs.

El precio de la licencia varía según las funcionalidades y los servicios de gestión de targets que ofrece Wikitude, pero existe la posibilidad de usar una versión gratuita con marca de agua en la visión de la captura del dispositivo (Ilustración 11).

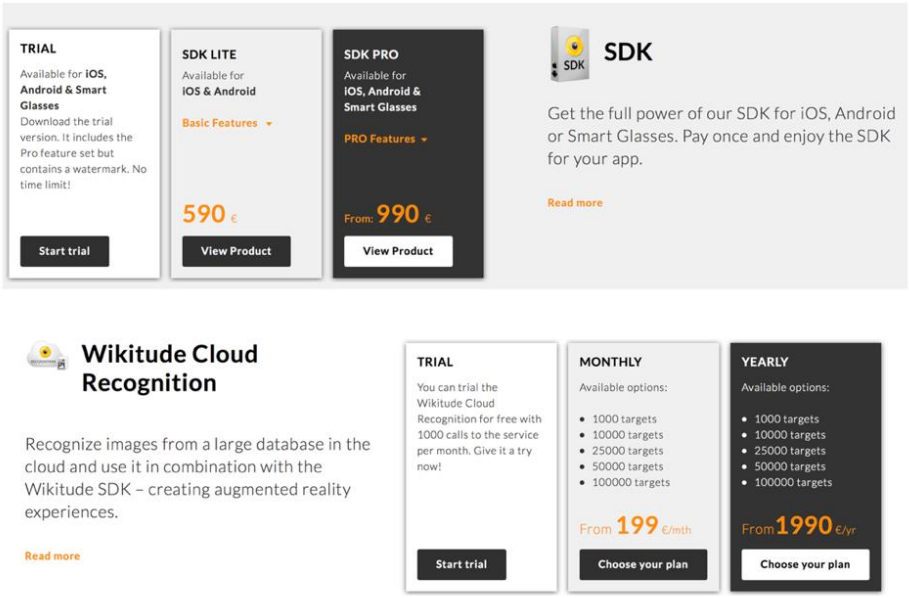


Ilustración 11. Tipos de licencia y prestaciones de Wikitude SDK (Documentación en web de Wikitude).

En la página de documentación para desarrollo de Wikitude[26] (© Wikitude GmbH 2012-2016) se puede encontrar una guía y/o documentación de ayuda en los primeros pasos con el SDK de Wikitude. Además, se explican los conceptos más avanzados del SDK, y hay disponibles ejemplos para desarrollar un proyecto propio de RA.

En este TFG se ha elegido la versión gratuita para Android de Wikitude en JavaScript, como ya se ha comentado anteriormente, ya que:

- El desarrollador ya había trabajado con Wikitude, aunque también con Vuforia (Otro SDK existente para RA).
- El desarrollador esta familiarizado e interesado en el lenguaje JavaScript.
- Permite la gestión de POIs, muy útiles para la función de navegación en la aplicación a desarrollar. De hecho, ésta es la razón principal razón por la cual se ha elegido este SDK.

[26] http://www.wikitude.com/developer/documentation/android

Por tanto, la fórmula elegida para el entorno desarrollo en este TFG es:

[SDK de Wikitude para Android (en JavaScript) + Android Studio]

3.4. Diseño de Pocket Campus – Augmented EPSG

La aplicación consta de dos modos de uso: "**Modo Exploración**" y "**Modo Navegación**". A estos dos modos se podrá acceder a través de una pantalla de inicio (Home), la cual se mostrará nada más abrir la aplicación. Esta pantalla presenta los dos modos de uso de la aplicación, anteriormente mencionados, proporcionando información útil sobre ellos.

El diagrama de flujo definido para el diseño de la app, se puede ver en la Ilustración 12.

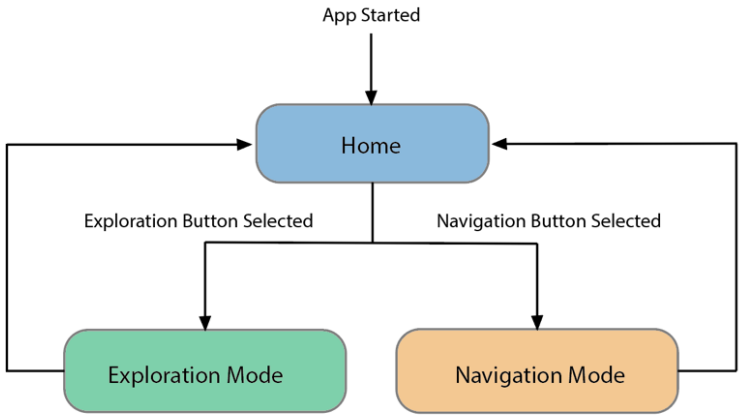


Ilustración 12. Flujograma de la app Pocket Campus (Creada con Adobe Illustrator).

3.4.1. Diseño del modo exploración

Esta funcionalidad, o modo de la aplicación, estará destinada a tareas de reconocimientos de ***targets***, de tipo imagen; es decir, el usuario podrá ir libremente por el Smart Campus y utilizar la aplicación para visualizar contenidos virtuales, que aporten información o servicios complementarios sobre ciertos elementos distribuidos por el *Smart Campus*. Dichos elementos se reconocerán porque incluirán el logo de *Pocket Campus* (app desarrollada) y pueden ser de diferente índole, como un cartel de un evento, una obra en una exposición, una indicación de advertencia, una tablón de anuncios, etc.
Cuando se detecten dichos elementos, apuntando con la cámara hacia ellos se mostrarán contenidos virtuales relacionados. Dichos contenidos, como se ha comentado anteriormente, deben ser de utilidad y deben presentarse bien estructurados y organizados, para no saturar al usuario.

Para este modo de uso, se han implementado algunos de los ejemplos anteriormente enunciados, como ejemplo de aquellas situaciones o casos donde se podrían incluir el uso de esta funcionalidad de Pocket Campus. En concreto, se van a implementar tres de los casos anteriormente planteados:

- Aumentando un ***Info-Marker***:

Dentro de este caso, se han diferenciado dos tipos de Info-Markers: Info-Map e Info-Warning.

1. ***Info-Map***: Se trata de un target con el logo de Pocket Campus y la palabra Map en mayúsculas, repartido por diferentes puntos del *Smart Campus*. Al capturarlo con la app, se muestra un mapa 3D del campus interactivo aumentado (Ilustración 13).

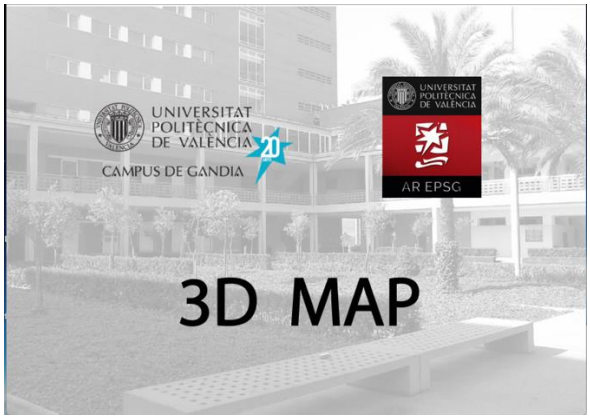


Ilustración 13. Info-map (Diseñado con Adobe Illustrator y Photoshop)

2. ***Info-warning***: Se trata de un target (de forma circular, para imitar la apariencia de una señal), con el logo de Pocket Campus y la palabra Warning en mayúsculas, situado en algunos puntos de peligro del Campus. Al capturarlo con la aplicación, se muestra información útil sobre el peligro localizado en las proximidades. En la app ha desarrollar, este *info-warning* avisará sobre el peligro de incendio, como peligro localizado (Ilustración 14).

Ilustración 14. Info-warning (Diseñado con Adobe Illustrator).

- Aumentando un **cartel de un evento**:

En este caso, el usuario tiene la posibilidad de utilizar la aplicación para visualizar contenidos virtuales mientras captura con la cámara de su dispositivo, un cartel propagandístico o publicitario. Como ejemplo, se ha utilizado un cartel del evento **Gandia Game Jam**[27] (Ilustración 15), al que se le asociará contenido

[27] http://gandiagamejam.webs.upv.es

aumentado: información contextual del evento, video de presentación y, cierta interacción con el usuario, gracias a la implementación de un botón virtual, para acceder directamente al sitio web donde, se realiza la inscripción al evento.

Ilustración 15. Cartel_Target (Diseñado con Adobe Illustrator)

Los dos casos, o situaciones, definidas anteriormente, se ha implementado en el modo exploración de la aplicación, cuya interfaz no es más que una visión a pantalla completa de la captura de la cámara, como si el usuario estuviese grabando un video.

Esto permite al usuario visualizar contenidos virtuales aumentados en una mayor superficie de pantalla, por lo que disfrutará en mayor medida de la experiencia de AR. Si por el contrario, se añaden demasiados elementos en la interfaz de usuario, la experiencia de usuario podría resultar confusa y/o estresante.

De hecho, hay que recalcar que es el usuario quien debe identificar los elementos del entorno real que pueden ser capturados para visualizar contenido virtual aumentado. Esto será tan fácil como, identificar el logotipo de la aplicación (Ilustración 16) en algún elemento o lugar del campus (carteles, paneles informativos, indicaciones, mostradores,…) y capturarlo con la cámara del dispositivo.

Ilustración 16. Logotipo de Pocket Campus - Augmented EPSG (Diseñado a partir del logo de la app UPV)

3.4.2. Diseño del modo navegación

El modo de navegación permite que los usuarios puedan encontrar el acceso a cualquier espacio del Smart Campus de la EPSG, sin la necesidad de interpretar mapas 2D, ni de utilizar sistemas de navegación alternativos.

Esta funcionalidad podría ser de gran utilidad, por ejemplo, cuando se llevan a cabo las jornadas de puertas abiertas para nuevos estudiantes ya que, estos estudiantes no conocen aún como está distribuido los espacios en el campus, y muchos de los POIs importantes para visitar pueden pasar desapercibidos. Incluso puede resultar útil cuando el estudiante deja de ser "novato", ya que algunos campus universitarios ocupan una extensión equiparable a la de un barrio, o en una población, y hasta encontrar un simple despacho de un profesor, se puede convertir en una tarea muy complicada.

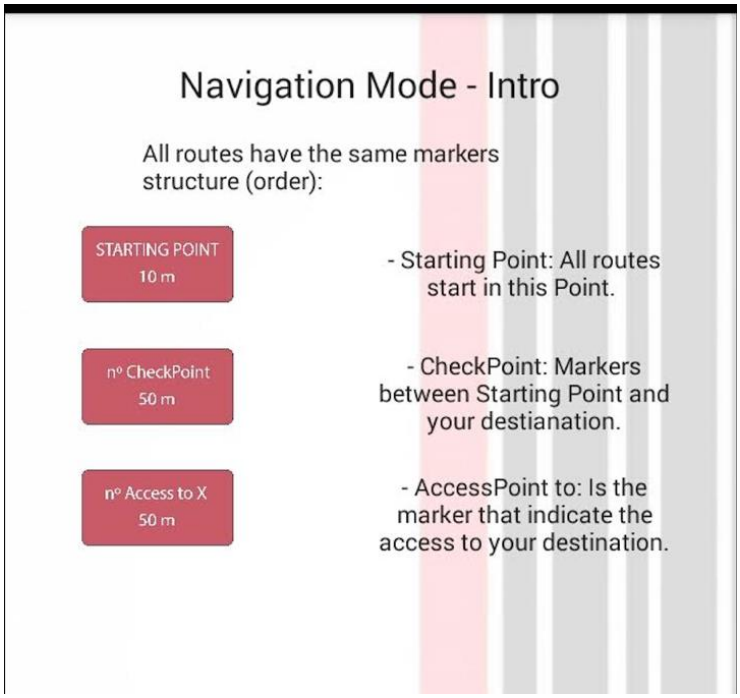


Ilustración 17. Pantalla intro del modo navegación (Captura de una pantalla de la app Pocket Campus).

Para que la aplicación pueda guiar al usuario por el Campus de la EPSG, sólo hace falta que este seleccione el modo de navegación en la pantalla de inicio de *Pocket Campus*, y visualice a su alrededor con la cámara del dispositivo. El modo navegación mostrará al arrancarse información útil sobre el uso de esta funcionalidad. Una vez que el modo navegación ha sido iniciado, el usuario será dirigido a un **Punto de Inicio** (***Start Point***) desde donde comienzan todas la rutas disponibles. A continuación, el usuario podrá seleccionar cualquiera de los espacios o **Puntos de Interés (POIs)** disponibles en el menú de selección de destino. Cuando se haya seleccionado un destino, la aplicación mostrará al usuario indicadores virtuales sobre los diferentes puntos que forman la ruta hasta llegar al dicho destino. Como se ha comentado anteriormente, todas las rutas comienzan en el Punto de Inicio (para situar al usuario en una ubicación dentro del campus, lo más centrada posible), en su trayectoria los usuarios se encontrarán con indicadores intermedios (***CheckPoints***) que lo guiarán hacia el destino, en orden creciente (*CheckPoint 1, CheckPoint 2,…*). Por último, los usuarios llegarán a uno o varios indicadores de acceso al espacio o destino del campus seleccionado (***Access to***). En todos estos marcadores, se puede leer, el tipo de indicador y la distancia del usuario hasta dicho indicador (Ilustración 17).

De esta manera, el usuario podrá visualizar los marcadores e interpretar la ruta con solo echar un vistazo a la captura de su dispositivo, mientras pasa por cada uno de los indicadores presentados hasta el destino seleccionado.

La interfaz para este modo de la aplicación, será una interfaz basada en la captura de la cámara, a pantalla completa, al igual que en el caso de exploración, para que el usuario pueda visualizar los diferentes indicadores sin obstáculos innecesarios, ya que los propios elementos virtuales, formarán parte de la interfaz de usuario.

3.5. Creación y obtención de recursos multimedia para Pocket Campus – Augmented EPSG

En este apartado, se explicará cómo se han obtenido y/o creado los diferentes recursos (conocidos como ***assets***), utilizados como contenidos virtuales a mostrar en la aplicación de RA.

3.5.1. Recursos multimedia utilizados en el modo exploración

En base a los casos definidos en el diseño de este modo de la app (Apartado 3.4.1), se han elegido los siguientes recursos multimedia:

- Recursos para aumentar en un *info-warning*

En primer lugar, se ha de diseñar el **target** para el caso del *info-warning* (ilustración 14), tal y como se ha descrito en el apartado (3.4.1). Se ha de tener en cuenta que el *target* no es más que un cartel o indicación física situada en alguno de los espacios del campus (mostradores, paneles informativos, accesos a los laboratorios,…). Por tanto, el diseño del target debe ser totalmente reconocible por el usuario. Si no, pasará desapercibido.

Para informar al usuario, sobre cómo actuar en caso de que se produzca un incendio, este *info-marker* aumenta una señal o indicador virtual de *warning* (contenido 2D) y un video explicativo, sobre cómo actuar en caso de que se produzca dicho peligro. Para diseñar el contenido 2D, se ha utilizado

la herramienta *Adobe Illustrator*. Por otra lado, como video explicativo, se ha utilizado un video[28], creado por la empresa Ruva Seguridad, como ejemplo de uso de un extintor para apagar el fuego.

Se han de tener en cuenta los requisitos que impone el uso del SDK de Wikitude, sobre el tipo de codificación de video y resolución requerida, por ejemplo.

La clase ***AR.VideoDrawable***[29] del SDK de Wikitude (la cual se utiliza para reproducir videos en modo RA), sólo permite cargar videos con una resolución de 720p (1280x720 píxel) y que usen el códec de video H.264. Por tanto, se deben adaptar los videos utilizados, si no se cumplen estas características mediante el uso de herramientas profesionales como *Adobe Premier Pro*. Si por otra parte, el video es de creación propia u original, se deben tener en cuenta estas características a la hora de exportar el archivo de video.

- Recursos para aumentar en un *info-map*:

  En primer lugar, al igual que para el caso del *info-warning*, y para todos los casos definidos en el modo exploración, se debe diseñar un target unívoco para el *info-map* (ilustración 13), ya que estará almacenado en la ***Target Collection***, junto a los demás *target*s (como se explicará más adelante), y cada uno debe mostrar un contenido virtual específico.

  Por otro lado, el recurso principal para implementar el caso del *info-map*, es el **modelo 3D** del Campus de la EPSG. Implementar un modelo 3D de estas características es un trabajo largo y costoso, para el cual se necesitan conocimientos avanzados de modelado 3D y diseño. Por suerte, la **UPV** dispone de los modelos 3D de los tres campus universitarios, gracias al trabajo llevado a cabo por Carlos Sánchez, Eduardo Vendrell y José Luis Diez (Carlos, Eduardo y José Luis, 2011). Gracias a su colaboración, se ha podido obtener el modelo 3D del Campus de la EPSG.

  Una vez que se ha obtenido el modelo 3D, gracias a la mediación del tutor, se debe comprobar que sus especificaciones cumplen con los requisitos que impone el uso del SDK de Wikitude. Para conocer estos **requisitos**, se ha de visitar la sección que trata sobre el flujo de trabajo (*Workflow*) con contenidos 3D[30], en la página oficial de Wikitude. Allí, se especifica que los archivos 3D soportados por Wikitude (para convertir a formato .wt3, explicado más adelante), sólo pueden ser de tipo .fbx (máx. Versión 2013) o .dae (Collada[31]). Además de este requisito básico, se enuncian algunas características de modelos 3D que no son soportadas por el SDK empleado (como multi-texturas), y algunas recomendaciones a tener en cuenta, en beneficio del proceso de renderizado de los modelos 3D.

  De hecho, el modelo de la EPSG facilitado por los compañeros de la UPV, es de tipo .max (utilizado por la herramienta de modelado 3D max). Este tipo de modelos 3D no son soportados por el SDK de Wikitude, por lo que es necesario convertir el modelo 3D a .fbx o .dae, para su uso en el desarrollo de la app. Para ello, se ha utilizado la herramienta de modelado **Blender**. El primer paso para conseguirlo es abrir el archivo de extensión .max con Blender. Una vez que el software ha cargado el modelo 3D, se ha de exportar el archivo con formato .fbx desde el menú File/Export/FBX(.fbx), como se puede ver en la Ilustración 18. Esto generará un archivo de extensión .fbx, adecuado para utilizar con el conversor Wikitude (*3DEncoder*). Para exportar el archivo como .dae, se ha de seguir el mismo proceso.

---

28 https://www.youtube.com/watch?v=2rEphDkxS6g

29 http://www.wikitude.com/external/doc/documentation/4.1/htmlcss/video.html#video-drawables

30 http://www.wikitude.com/external/doc/documentation/4.1/htmlcss/assetsworkflow.html#3d-assets-workflow

31 https://www.khronos.org/collada/wiki/Main_page

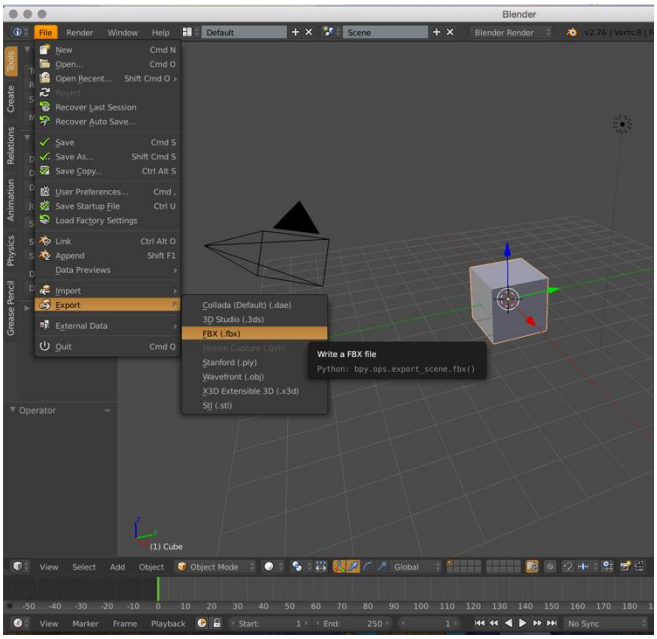


Ilustración 18. Exportación del modelo 3D como .fbx, (Captura de la interfaz de Blender).

- Recursos para aumentar en un cartel de un evento:

Como ejemplo de evento, se ha elegido la *Gandia Game Jam*, un evento dedicado al desarrollo de videojuegos que se realiza cada año en el Campus de la EPSG.

Al igual que los casos anteriores, se ha diseñado un *target* específico (Ilustración 15); para este caso, se ha utilizado el propio cartel publicitario del evento, incluyendo el logo de *Pocket Campus*, para que el usuario pueda detectar que el cartel puede ser aumentado usando la app.

Para promocionar el evento, se ha decidido usar un video de presentación del mismo, otro video resumen de la edición pasada del evento, un panel 2D con información descriptiva y un botón 2D virtual para dirigir al usuario hacia el sitio web de inscripción al evento.

Por otro lado, para el diseño del panel 2D informativo, se ha usado ***Adobe Illustrator***, al igual que para definir la apariencia y forma del botón virtual. Estos contenidos creados con Adobe Illustrator deben exportarse como .jpg, para que puedan ser utilizados en el desarrollo de la aplicación y crear los elementos virtuales necesarios.

3.5.2. Recursos multimedia utilizados en el modo navegación

Para el modo navegación, por ahora, sólo se necesita diseñar la apariencia y forma de los indicadores o marcadores virtuales, utilizados para indicar el recorrido que debe seguir el usuario para llegar al destino seleccionado. Para ello, se ha diseñado un panel 2D (con Illustrator), sobre el cual presentar la información del indicador de ruta: tipo de indicador y distancia a la que se encuentra el usuario (indicadores de la Ilustración 17).

Más adelante, tal vez se necesite crear u obtener algún contenido gráfico o multimedia concreto (para crear algún botón específico de la app, por ejemplo), pero, por ahora, ya se han obtenido todos los recursos necesarios para el desarrollo de la aplicación, en base al diseño planteado en el apartado 3.4.

3.6. Desarrollo de Pocket Campus – Augmented EPSG

En primer lugar, se debe recordar la fórmula elegida como entorno de desarrollo:

[SDK de Wikitude para Android (en JavaScript) + Android Studio]

- USO RECOMENDADO DEL SDK:

La documentación del SDK[32] está dispuesta de manera que pueda guiar al desarrollador a través de las diferentes etapas, en el proceso de desarrollo. Para comenzar a desarrollar la aplicación se ha seguido cada una de las etapas indicadas a continuación:

1. **Configuración** del proyecto: En esta sección se describen las tareas necesarias, en detalle, para configurar correctamente un proyecto de RA .

2. Ver los **ejemplos**: Todos los ejemplos incluidos en el SDK, son experiencias de RA completas, guardados en *SDKExamples app*. Lo ideal sería navegar a través de esta sección para dar forma a la nueva idea de app, e investigar de lo que es capaz el SDK a descargar. Estos ejemplos están diseñados para ayudar al desarrollador a conseguir un buen comienzo en el uso de Wikitude SDK. Para ver los ejemplos, basados en visión de RA, se requieren las correspondientes imágenes de referencia. Todas estas imágenes están disponibles directamente en la descripción del ejemplo o recogidas en una colección, en la que se puede visualizar o imprimir cualquiera de ellas.

3. Escribir tu propio ***Architect World****:* Esta sección se adentra en la explicación de buenas prácticas para el flujo de trabajo de desarrollo (*development workflow*) una vez que se ha escrito el código para una experiencia de RA propia.

- ARQUITECTURA DE WIKITUDE SDK:

El SDK permite crear tanto los casos de uso basados en ubicación (modo navegación) como los casos en los que se requiere de reconocimiento de imagen y tecnología de seguimiento *tracking* (modo exploración); lo que se conoce como visión basada en RA.

[32] http://www.wikitude.com/external/doc/documentation/4.1/htmlcss/assetsworkflow.html#3d-assets-workflow

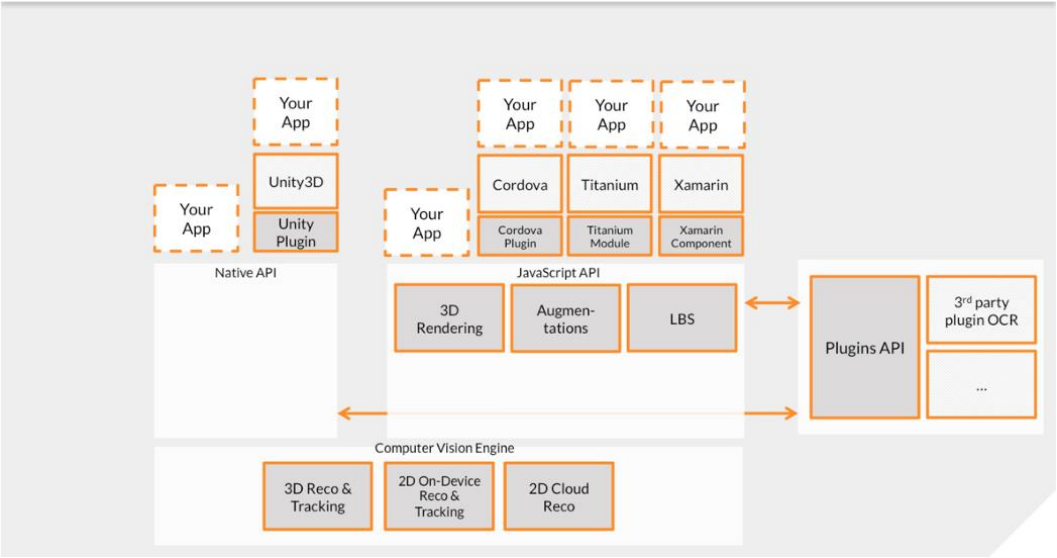


Ilustración 19. Arquitectura del SDK de Wikitude (Documentación en Web de Wikitude).

En la ilustración anterior se muestran los diferentes componentes del SDK de *Wikitude* y las posibles estrategias para crear aplicaciones de RA mediante el mismo. Cada una de estas estrategias está basadas en **cierto entorno de desarrollo** (IDE – *Integrated Development Environment*) y plataformas. A continuación, se explican en detalle para explicar los elementos que conforman la estrategia elegida, para el desarrollo de *Pocket Campus*:

- ***Computer Vision Engine***: El *engine* (motor) de visión por ordenador, es un componente básico del SDK de *Wikitude*, el cual es utilizado por todas las plataformas. Este no es directamente accesible, sino que se accede mediante el uso de *Native API* o *JavaScript API,* como se puede observar en la Ilustración 19.

- ***Wikitude SDK – JavaScript API***: Permite construir entornos de RA en base a los lenguajes HTML y JavaScript. Esta API se encuentra disponible tanto para Android como para IOS.

- CONFIGURACIÓN DEL PROYECTO EN ANDROID STUDIO:

Tras descargar el SDK para *Android* en *JavaScript*, en el siguiente enlace: Wikitude SDK – Android JavaScript, se deben llevar a cabo algunos pasos de configuración previos a la implementación de la experiencia de RA en sí misma.

PRIMEROS PASOS:

- Crear un Proyecto de Aplicación Android en blanco o vacío.

- Probar que el proyecto se ejecuta sin ningún tipo de error (aunque aún este vacío) y poner el proyecto bajo la gestión de la **herramienta *Git*** (sistema de control de versiones). De esta forma, se podrá avanzar en el desarrollo de la aplicación, paso a paso, realizando una "copia de versión" cuando se logre un pequeño avance significativo.

- Así, siempre habrá una opción de deshacer lo cambios realizados en el código, ya que, en ocasiones, resulta muy complicado encontrar el fallo si se ha avanzado demasiado en el desarrollo, sin salvar la versión. Para ello, hay que descargar la herramienta *Git* de la pagina oficial[33] e instalarla utilizando la consola del ordenador. A continuación, se debe acceder al directorio en el que este contenido el proyecto que se ha creado, para poder gestionar las versiones del proyecto.

- Copiar el archivo *libs/wikitudesdk.aar* en la carpeta del módulo de la app (Project/root/app/libs).

- Abrir build.gradle desde dicho módulo, y añadirle el *wikitudesdk.aar* como una dependencia, notificando al *gradle* para buscar en la carpeta de libs, con el siguiente código.

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile (name: 'wikitudesdk', ext:'aar')
    compile 'com.android.support:appcompat-v7:21.0.3'
}

repositories {
    flatDir{
        dirs 'libs'
    }
      }
```

- Si ya se ha adquirido una licencia, se debe ajustar el *applicationID* (“xxx”).

```
defaultConfig {
        applicationId "xxxx"
          }
```

- A continuación, se añaden los siguientes permisos en el *AndroidManifest.xml*:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_GPS" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-feature android:name="android.hardware.camera" android:required="true"
/>
<uses-feature android:name="android.hardware.location"
android:required="true" />
<uses-feature android:name="android.hardware.sensor.accelerometer"
android:required="true" />
<uses-feature android:name="android.hardware.sensor.compass"
android:required="true" />
```

- La actividad que contiene el *AR-View,* explicado más adelante (llamado ***architectView***), debe tener incluido o instalado *Android:configChanges = “screenSize | orientation”* en el *AndroidManifest.xml*, por lo que quedaría algo como…

```
<activity android:name="com.yourcompany.yourapp.YourArActivity"
   android:configChanges="screenSize|orientation"/>
```

- Introducir una **clave de licencia** válida. El SDK *Wikitude* requiere de una clave de licencia válida, para que todas las funcionalidades del SDK se activen. Una clave de licencia vacía o perdida bloquea la vista de RA para mostrar cualquier contenido significativo, y se verá una marca de agua

[33] https://git-scm.com

en la pantalla con las palabras: *License Key Missing* (clave de licencia ausente) y, por tanto, todas las llamadas a la *API JavaScript* serán ignoradas y no interpretadas. Cuando se descarga el SDK de *Wikitude*, se direcciona al desarrollador a la página de generación de licencia, donde se genera automáticamente una clave de licencia para su uso.

A) LICENCIAS PROPIAS

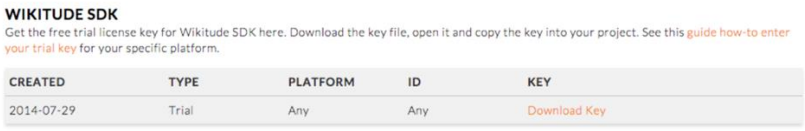


Ilustración 20. Plataforma Mis Licencias de Wikitude (Documentación en web de Wikitude).

Se debe copiar la clave (Ilustración 20) en la aplicación a desarrollar, la cual desbloqueará el modo de prueba (Trial) del SDK de *Wikitude*. El modo de prueba contiene todas las características de la versión de pago del SDK , pero se muestra una marca de agua (marca de *trial*) en la pantalla cuando se abre la vista de RA. Cada clave de licencia de prueba, es válida para todo ID de aplicación en todos los sistemas operativos, es decir, se puede usar la misma clave de licencia para múltiples aplicaciones.

B) INTRODUCCIÓN DE LA CLAVE DE LICENCIA (ANDROID)

Para el uso del SDK de *Wikitude* en *Android*, se necesita proporcionar una clave de licencia válida al *lifecycle-method* (método de ciclo de vida) en *onCreate()* del *Architect View*. Esto se puede hacer directamente proporcionando la clave como un *string* en la llamada del método *onCreate()*: *onCreate(final String Key)* o creando un objeto *StartupConfiguration*, pasándole la licencia como un *string*, y entonces llamar al método *onCreate (final StartupConfiguration)*. Para ver un ejemplo práctico sobre cómo introducir la licencia, se puede consultar el *AbstractArchitectCamActivity* contenido en *SDK Examples* del paquete *Wikitude* SDK descargado.

- AR VIEW EN ACTIVIDAD DE ANDROID:

Hay que tener en cuenta que el SDK de *Wikitude* no es un SDK nativo de *Android* como otros SDKs. El concepto básico es el de añadir un *architectView* en el proyecto y notificar sobre los eventos de ciclo de vida de la aplicación. El *architecView* crea una superficie de cámara y toma eventos de los sensores.
La experiencia en sí misma, a lo que se ha hecho referencia a veces como *ARchitect World*, está implementada en *JavaScript* y contenida en un *package*, en la carpeta *assets* de la aplicación. Las experiencias están implementadas en *HTML* y *JavaScript*, y los métodos de llamada son de tipo AR.”EspacioDeNombre”, por ejemplo, *AR.GeoObject*.

Por lo tanto, se ha de incluir la siguiente línea,

```
<script src="architect://architect.js"></script>
```

, en los archivos HTML, para usar el AR.”EspacioDeNombre” y manejar adecuadamente el *architectView*. Para probar un *ARchitect World* en un navegador de escritorio se debe incluir la herramienta ade.js para evitar errores de lenguaje *JavaScript* y visualizarlos en la consola de desarrollo.

Es recomendable manejar la experiencia de RA separada de la Actividad (*Activity*-Android). Se debe declarar el *architectView* dentro del *layout.XML*. Por ejemplo, añadir el siguiente código en las *parent tags* (etiquetas padre) del *FrameLayout:*

```
<com.wikitude.architect.ArchitectView android:id="@+id/architectView"
   android:layout_width="fill_parent" android:layout_height="fill_parent"/>
```

El *ArchitectView* está creando una superficie de cámara, así que hay que asegurarse de liberar la cámara en caso de que se se vaya a usar en algún otro sitio de la aplicación. Además de la cámara (*front or back-facing*), el *ArchitectView* también hace uso de los valores del compás y del acelerómetro del dispositivo, por lo que requiere de *OpenGL 2.0* y , como mínimo, *Android 4.0*.

ArchitectView.isDeviceSupported(Context context) comprueba si el dispositivo actual tiene presentes todos los requisitos de hardware y software.

Es muy importante notificar al *ArchitectView* sobre los eventos de ciclo de vida (*life-cycle events*) de la Actividad. De hecho, se llama a *onCreate(), onPostCreate(), onPause(), onResume() y onDestroy(*) de *architectView*, dentro de los métodos *life-cycle* de la Actividad. Una buena práctica es la de definir una variable miembro para el *architectView* en la Actividad.

Posteriormente, configurarlo justo después de *setContentView* en el método *onCreate(*) de *Activity*, y entonces acceder a *architectView* por medio de la variable miembro.

```
this.architectView = (ArchitectView)this.findViewById( R.id.architectView );
final StartupConfiguration config = new StartupConfiguration( * license key */
);
this.architectView.onCreate( config );
```

El método *onPostCreate ()* es el mejor lugar para cargar la experiencia RA.

```
this.architectView.onPostCreate();
this.architectView.load( "YOUR-AR-URL" );
```

El argumento *architectView.load()* es la parte del archivo HTML, que define la experiencia de RA. Este puede tener relación con la carpeta de *assets* o con una URL web (que comience por http://). Por ejemplo, *architectView.load("arexperience.html")* abre el archivo HTML en la carpeta de assets del proyecto y, obviamente, architectView.load("http://yout-server.com/arexperience.html") carga un archivo de un servidor.

*Nota: Sólo se pueden pasar argumentos de tipo archivo HTML cuando se carga vía URL. Por ejemplo, architectView.load ("arexperience.html?myarg=1") no funcionará.

- LOCALIZACIÓN:

La gestión de la localización es importante en aplicaciones de RA basadas en localización. Dependiendo del tipo de uso, se puede utilizar servicios de localización según la señal de GPS o la señal captada de cobertura 3G, dependiendo del tipo de servicio de posicionamiento definido.

En el proyecto *SDKExamples* se proporciona una implementación básica de un *LocationProvider*, la cual es sin duda la mejor estrategia de localización (LocationStrategies) para Android. Aun así, para el desarrollo de *Pocket Campus*, se ha usado la *API de Google Maps*, tal y como se explicará más adelante.

Finalmente, tras todos estos pasos de configuración e inicialización del proyecto, se puede empezar a implementar la parte de la aplicación que corresponde a la experiencia de RA en sí misma. Esta experiencia estará implementada en *Javascript*, y se cargará desde su respectivo código HTML, almacenado en la carpeta de assets de la app.

3.6.1. Desarrollo del modo exploración

En primer lugar, se abre el proyecto en ***Android Studio***, anteriormente configurado. Para comenzar con el desarrollo o implementación, se ha de crear la actividad correspondiente a este modo de la app, haciendo clic con el botón derecho del ratón en el directorio app del proyecto y seleccionando New/Activity/EmptyActivity (Ilustracion 21).

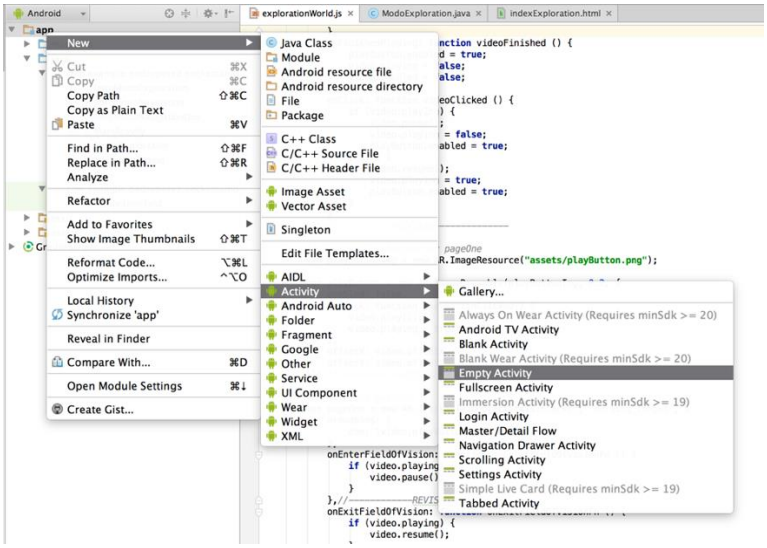


Ilustración 21. Creando la actividad de android, llamada ModoExploration.java (Pantalla de Android Studio).

A esta actividad se le ha llamado *ModoExploration.java,* e implementa los métodos propios de cualquier actividad de Android (*onCreate*(), *onStart*(), *onPause*(), etc) para gestionar el *ArchitectView* (la experiencia AR en sí misma), dependiendo del estado del ciclo de vida en el que se encuentre la app.

Esta actividad se ha almacenado en la carpeta java, generada por defecto cuando se crea la aplicación, y tendrá asociado un *layout* propio para definir el aspecto de la actividad gracias al uso de código *XML* (*activity_modo_exploration.xml*).

```
@Override
protected void onPostCreate (Bundle savedInstanceState){
    super.onPostCreate(savedInstanceState);

    architectView.onPostCreate();

    try {
        this.architectView.load("indexExploration.html");
    } catch (Exception e){
    }
}
```

Ilustración 22. Fragmento de código donde se carga el ArchitectView (Código fuente Pocket Campus).

Como se puede apreciar en el fragmento de código anterior, la **experiencia de RA** implementada en HTML y *Javascript* se carga desde el método ***onPostCreate()*** (Ilustración 22), del ciclo de vida de la actividad, mediante el método *architectView.load().*

A este método, se le pasa como argumento un archivo de tipo HTML, el cual estará contenido en la carpeta de ***assets*** de la app. Para crear esta carpeta, simplemente se debe hacer clic con el botón derecho del ratón sobre la carpeta ***app***, que contiene todos los recursos del proyecto, y seleccionar la opción de carpeta de assets.

El archivo HTML, al que se ha llamado ***indexExploration.html***, define cuales son los *scripts.js* que deben ser cargados para crear la experiencia de RA, como: el *architect.js, el ade.js y el explorationWorld.js*, donde se implementarán todos los casos definidos en los apartados anteriores, es decir, donde se crean los contenidos de RA. De hecho, para implementar el modo exploración, solo se necesita ir creando los elementos del entorno de RA (***explorationWorld.js***).

En primer lugar, se debe crear un script con la extensión .js y almacenarlo en una carpeta, o directorio, dedicado para almacenar este tipo de *scripts* (directorio js). Una vez que se ha ubicado correctamente el *script*, se ha de implementar el código.

Básicamente, se ha de crear una variable llamada ***World***, de tipo var (ya que en *Javascript* no hace falta declarar el tipo de la variable) y, tras haberla creado, se ha de llamar al método *init()* de la propia clase (*World.init()*), para que la aplicación inicie la experiencia AR. Dentro de la variable *World*, se han de crear cada uno de los contenidos virtuales que la aplicación va a ser capaz de mostrar, y se asocian cada uno de estos contenidos a un target específico, según los casos definidos para el modo exploración (apartado 3.4.1).

Todos estos contenidos virtuales serán implementados como *Overlays* o contenidos superpuestos en la función *createOverlays()*, dentro de la variable *World*. Pero antes de empezar a definir contenidos, se ha de crear el ***tracker*** para todos los elementos que pueden ser reconocidos por el *ClientTracker*, es decir, por el proceso de reconocimiento y seguimiento llevado a cabo en el dispositivo del usuario, ya que el *tracker* se aloja en la aplicación misma, dentro de la carpeta *assets*.

```
createOverlays: function createOverlaysFn() {

    //Create the tracker for all elements that could be recognize for ClientTracker
    this.tracker = new AR.ClientTracker("assets/tracker.wtc", {
        onLoaded: this.worldLoaded
    });
```

Ilustración 23. Fragmento de código para crear el tracker (Código fuente Pocket Campus).

- OBTENCIÓN DEL TRACKET.WTC :

  Para obtener el *tracker* se han de tener preparados todos los *targets* o elementos físicos que vayan a poder ser reconocidos por el *ClientTracker*, ya que gracias a este archivo de tipo .wtc, se almacenan todos los targets en un solo archivo.wtc, adecuado para permitir al software de *Wikitude* extraer la información necesaria de cada *target* y, así, reconocerlos cuando se ha inicie la variable *World* del *explorationWorld.js*, es decir, la experiencia de RA.

Una vez que se tienen los *targets* preparados, hay que dirigirse a la sección de *Tools/Target Manager*[34] de la página web oficial de Wikitude, donde es necesario iniciar sesión. Si aún no se ha registrado, hay que registrarse facilitando algunos datos sobre el desarrollador, igual que para obtener la licencia de uso del SDK. Tras iniciar sesión, aparecerá una pantalla de la plataforma Wikitude Studio, en la que aparecen los proyectos Wikitude que se estén desarrollando. Para crear uno nuevo, simplemente se debe pulsar en *add Project* (Ilustración 24).

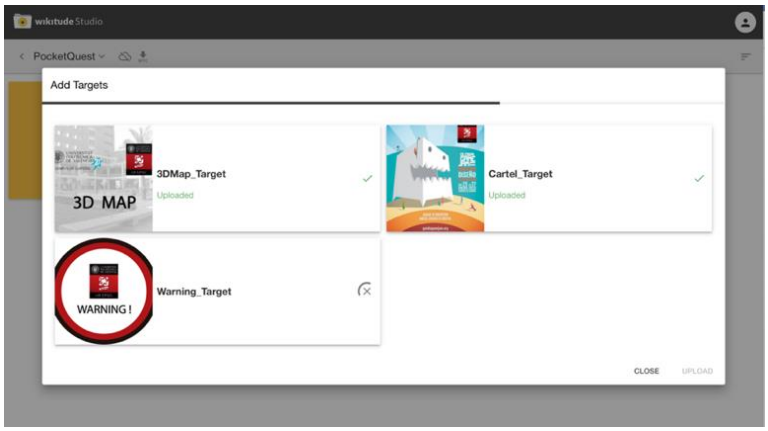


Ilustración 24. Pantalla de la plataforma Wikitude Studio, donde se cargan todos los targets (Pantalla del Target Manager)

Tras haber creado el proyecto, se han de crear los *targets* necesarios. Para ello, se ha de pulsar en la opción *add targets* y cargar el archivo (.jpg o .png sin transparencia), que corresponde a dicho *target*, como se puede ver en la ilustración anterior, según el diseño planteado para el desarrollo de la app. Tras haber seleccionado el archivo, se ha de pulsar el botón *upload* y dicho *target* será añadido a la colección, necesaria para crear el *tracker.wtc*. Tras haber añadido todo ellos, definidos en el apartado de diseño, se ha de descargar el archivo .wtc pulsando en el botón de la parte superior de la pantalla (El logo de la flecha *download*, con las letras WTC).

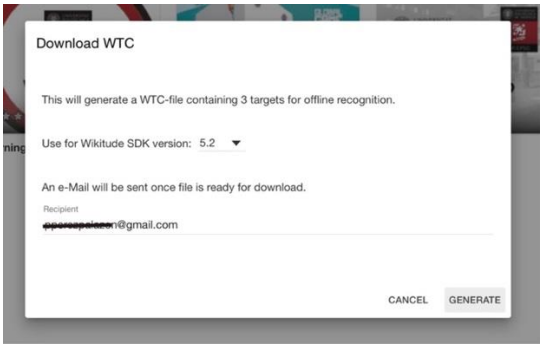


Ilustración 25. Pantalla para generar el tracker.wtc (Pantalla del Target Manager).

Como se puede observar en la ilustración anterior, el *archivo.wtc* se envía al correo con el que se ha dado de alta el desarrollador. Por tanto, tras haber pulsado el botón de *Generate*, hay que dirigirse al correo electrónico facilitado en el registro, y descargar el archivo enlazado en el correo enviado por el *Target Manager de Wikitude*. Solo en algunas ocasiones la plataforma no requiere del envío de este correo para proporcionar el *archivo.wtc*, y te permite descargarlo directamente desde el navegador.

[34] https://targetmanager.wikitude.com/projects

Una vez que el *tracker.wtc* (nombre por defecto, cuando se descarga) ha sido descargado, se incluye en el directorio assets del proyecto Android.

El siguiente paso es la creación de los diferentes contenidos aumentados, conocidos como *drawables*. El SDK de *Wikitude* permite utilizar diferentes tipos de *drawables*. De hecho, para crear los contenidos del modo exploración, se han usado los siguientes tipos:

- **De tipo *AR.ImageDrawable*:** Para mostrar contenido gráfico 2D (imágenes) en la experiencia de RA. La clase *ImageDrawable* recibe, en su declaración, un objeto de tipo *ImageResouce*, en formato .jpg o .png, que se desea mostrar en la experiencia de RA, como se puede observar en la ilustración 26. Además, el objeto recibe algunos parámetros, como el valor de escala y la posición que ocupa en referencia al centro del *target* reconocido.

```
// Create overlay for pageOne
var imgOne = new AR.ImageResource("assets/assets_page_one/flamableSignal.png");
var overlayOne = new AR.ImageDrawable(imgOne, 1, {
    offsetX: -0.15,
    offsetY: 0
});
```

Ilustración 26. Creación de un objeto drawable, de tipo ImageDrawable (Código fuente Pocket Campus).

- **De tipo *AR.VideoDrawable*:** Para mostrar contenidos audiovisuales (videos) en la experiencia de RA. A diferencia de la clase *ImageDrawable*, a esta clase se le pasa, en su declaración, directamente el archivo .mp4 que se desea mostrar en la experiencia de RA. Además, recibe algunos parámetros como el valor de escala y la ubicación del video (ilustración 27), en base al centro del *target* reconocido.

```
// Create video2, the first videdrawable for pageTwo
 var video2 = new AR.VideoDrawable("assets/assets_page_two/spotGGJ.mp4", 0.7, {
            offsetX: 0.5,
            offsetY: 0.7
});
```

Ilustración 27. Creación de un objeto (drawable) de tipo VideoDrawable (Código fuente Pocket Campus).

- **De tipo *AR.Model***: Permite mostrar contenido 3D (modelos 3D) en la experiencia de RA. Para crear un objeto de tipo *AR.Model* se le ha de pasar, en su declaración, el archivo correspondiente al modelo 3D que se desea mostrar, pero con la extensión .wt3.

  Para convertir el archivo al tipo .wt3, se ha utilizado el *3Dencoder*, disponible en la web de Wikitude[35]. Simplemente, se descarga la herramienta y se instala. A continuación, se abre el programa, se carga el archivo correspondiente al modelo 3D (.fbx o .dae) y se pulsa el botón de convertir.

  Por último, se ha guardado el modelo.wt3 exportado en la carpeta de *assets* del proyecto en *Android*, para su posterior uso.

  En el caso del mapa 3D de la EPSG, el proceso de conversión no se podía llevar a cabo, ya que el número de mallas y vértices, usados por los creadores para modelar el mapa, es muy alto y, por tanto, el *modelo.wt3* resultante era muy pesado. Por este motivo, se ha utilizado la herramienta de modelado 3D *Blender*, para reducir el número de mallas y vértices, suavizando los bordes del modelo

[35] http://www.wikitude.com/products/wikitude-sdk-features/wikitude-3d-encoder/

3D y eliminando bordes innecesarios, que no aportan mucho en la interpretación del modelo por parte del usuario.

Por otro lado, cuando este problema se había solucionado, el conversor volvía a responder con un error, al intentar convertir el modelo.fbx. En este caso, el problema era causado por el tipo de iluminación utilizada para visualizar el modelo. Tras investigar un poco, se llegó a la solución de no añadir ningún tipo de iluminación desde la herramienta *Blender* y proseguir con la conversión del modelo. De esta forma, el conversor añade al modelo una iluminación por defecto, que si es permitida en el tipo de archivos .wt3.

```
// Create Model 3D for pageThree
this.modelEPSG = new AR.Model("assets/mapa3D.wt3", {

    onLoaded: this.loadingStep,
    scale: {
        x: 0.025,
        y: 0.025,
        z: 0.025
    },
    translate: {
        x: 0.0,
        y: 0.0,
        z: 0.0
    },
    rotate: {
        roll: 180.0,
        tilt: 30.0,
        heading: 0.0
    }
});
```

Ilustración 28. Creación de un objeto drawable de tipo AR.Model (Código fuente Pocket Campus).

A la hora de declarar un objeto de tipo *AR.Model*, se pueden establecer algunos parámetros como el parámetro *onLoaded*, para ejecutar cualquier tipo de acción cuando el modelo 3D se esta cargando. De hecho, en el modo exploración, al cargar el modelo 3D, se ejecuta la función *loadingStep()*, implementada dentro de la misma variable *World*. Otros parámetros son, la escala del modelo (en las tres coordenadas: X,Y y Z), la posición que ocupa, en referencia al centro del *target* (o imagen) reconocido y la rotación del modelo 3D. Este último es muy importante en el caso a tratar a implementar, ya que se trata de un mapa 3D que debe estar bien posicionado para permitir la correcta visualización por parte del usuario. Por ello, se ha de definir correctamente los valores de escala, rotación y translación del modelo 3D (Ilustración 28).

Haciendo uso de estos tres tipos de *drawables*, se han creado dentro de la función *CreateOverlays*(), todos los contenidos virtuales que se mostrarán en el modo exploración, organizados según los casos planteados en el diseño del mismo. Así, el caso del *info-warning*, corresponde al objeto *pageOne* declarado en el código, el caso del cartel de un evento, al objeto *pageTwo* y para el caso del *info-map* al objeto *pageThree*. Estos tres objetos creados son de tipo *AR.Trackable2DObject*, gracias a los cuales, el modo exploración será capaz de asignar el contenido correspondiente para cada target y podrá representarlo usando el *architectView*.

Para crear un objeto de tipo *AR.Trackable2DObject*, hay que pasarle, en su declaración, el archivo *tracker.wtc* (generado anteriormente) y especificar el nombre del *target* para el cual se pretende asociar los *drawables* (tal y como se han nombrado en el *Target Manager*), posteriormente facilitados. Los *drawables* se le pasan al objeto *Trackable2DObject*, como tercer parámetro en su declaración, almacenados en un *array* de objetos *drawables* (ilustración 29).

```
// Create the page Two
var pageTwo = new AR.Trackable2DObject(this.tracker, "Cartel_Target", {
    drawables: {
        cam: [video2,video3,overlayTwo,pageTwoButton]
    },
    onEnterFieldOfVision: function onEnterFieldOfVisionFn () {
        video2.play(3);

        if (video3.playing) {
            video3.pause();
        }
    },//------------REVISAR--------------
    onExitFieldOfVision: function onExitFieldOfVisionFn () {
        if (video3.playing) {
            video3.resume();
        }
    }//------------REVISAR--------------

});
```

Ilustración 29. Creación de pageTwo, un objeto de tipo Trackable2DObject (Código fuente Pocket Campus).

Para terminar de implementar el modo exploración, se han añadido algunas funcionalidades en algunos objetos *drawables* del *explorationWorld.js*:

- **Se ha incluido la funcionalidad de reproductor de video**: para el AR.*VideoDrawable* incluido en la *pageOne* (caso del *info-warning*) y para uno de los dos AR.*VideoDrawables* incluidos en la *pageTwo* (caso del cartel de un evento).

```
onClick: function videoClicked () {
    if (video.playing) {
        video.pause();
        video.playing = false;
        playButton.enabled = true;
    } else {
        video.resume();
        video.playing = true;
        playButton.enabled = true;
    }
}
```

Ilustración 30. Uso de la función onClick() de la clase AR.VideoDrawable para pausar y reproducir el video aumentado[36].

Para ello, se ha hecho uso de la función *onClick*() del objeto AR.*VideoDrawable*, gracias a la cual se puede manejar la reproducción del video con solo hacer clic en el video virtual mostrado en la vista de RA (Ilustración 30).

- **Se ha añadido una animación al mapa 3D de la EPSG (modelo 3D)**, gracias al uso de la clase *AR.PropertyAnimation()*. Para ello, en primer lugar un botón que active dicha animación, llamando a la función *toggleAnimationModel*(), en la propiedad *onClick* de dicho botón (objeto de tipo AR.*ImageDrawable*), la cual lleva acabo la activación y desactivación de dicha animación.

Para el caso del mapa 3D, se ha elegido una animación de rotación. Dado que puede ser interesante permitir dicha funcionalidad, para tener una vista general de todas las perspectivas del campus y,a de esta forma, tener un idea más clara de su organización en el espacio. Esta animación ha sido creada como ***rotationAnimation*** y se le ha asignado un objeto de tipo *AR.PropertyAnimation* al cual se le han pasado como parámetros: el objeto *AR.Model*, el tipo de animación que se quiere

[36] Fuente – Ilustración 30: (Código fuente Pocket Campus)

implementar (en este caso, *rotate.roll*), el arco de rotación que se espera que realice la animación y el tiempo que tarda el modelo en girar dicho arco de rotación, como se apreciar en la siguiente ilustración.

```
    /*
        Additionally to the 3D model an image that will act as a button is added to the image target. This can be accomp
    */
    var imgRotate = new AR.ImageResource("assets/assets_page_three/rotateButton.png");
    var buttonRotate = new AR.ImageDrawable(imgRotate, 0.2, {
        offsetX: 0.35,
        offsetY: 0.60,
        onClick: this.toggleAnimateModel
    });

    /*
        Similar to 2D content the 3D model is added to the drawables.cam property of an AR.Trackable2DObject.
    */

    //Create pageThree
    var pageThree = new AR.Trackable2DObject(this.tracker, "3DMap_Target", {
        drawables: {
            cam: [this.modelEPSG,buttonRotate]
        }
    });

    //Create a animation for modelEPSG in pageThree
    this.rotationAnimation = new AR.PropertyAnimation(this.modelEPSG, "rotate.roll", 180, -180, 10000);

},//createOverlaysFn()

toggleAnimateModel: function toggleAnimateModelFn() {
    if (!World.rotationAnimation.isRunning()) {
        if (!World.rotating) {
            // Starting an animation with .start(-1) will loop it indefinitely.
            World.rotationAnimation.start(-1);
            World.rotating = true;
        } else {
            // Resumes the rotation animation
            World.rotationAnimation.resume();
        }
    } else {
        // Pauses the rotation animation
        World.rotationAnimation.pause();
    }

    return false;
},
```

Ilustración 31. Implementación de una rotationAnimation, haciendo uso de la clase AR.PropertyAnimation[37].

- **Por último, se ha incluido una función que avise al usuario del estado en la carga del modo exploración,** es decir, de los contenidos que componen la experiencia de RA, haciendo uso de la propiedad *onLoaded()* de la variable *World*. Cuando esta es inicializada, se llama a la función *WorldLoaded()*, la cual hace aparecer un panel con texto donde se avisa al usuario que el modo exploración está preparándose para poder ejecutarse.

  Para implementar dicho panel se ha hecho uso de la librería para interfaces ***JQuery Mobile***[38], ideal para el caso que nos ocupa, ya que utiliza los lenguajes HTML, CSS y Javascript para crear multitud de elementos UI (User Interface o Interfaz de Usuario). Tan sólo se ha de descargar la librería y añadirla al directorio *assets* del proyecto e incluirla en el respectivo index.html, para poder utilizarla.

3.6.2. Desarrollo del modo navegación

Al igual que en el caso del modo exploración, el primer paso es crear una actividad en blanco, dentro de la carpeta *java* (creada por defecto al inicializar el proyecto en *Android*). A esta nueva actividad, se le ha llamado *ModoNavigation.java*. Esta clase *java*, además de heredar de *Activity* los métodos básicos del ciclo de vida de una actividad, deberá implementar algunas clases, como: *GoogleApiClient.ConnectionCallbacks,*

[37] Fuente – Ilustración 30: (Código fuente Pocket Campus)

[38] https://jquerymobile.com

GoogleApiCliente.OnConectionFailedListener y *LocationListener*, para dotar a la app de un servicio de geolocalización, basado en el uso de la ***API de Google Maps***[39].
La API de Google Maps para geolocalización, es usada solamente en el *ModoNavigation.java* (Actividad de Android), ya que es la parte de la app, donde se debe configurar y establecer el servicio de geolocalización utilizado, y donde se definen los *callbacks* necesarios para avisar de los cambios en la localización obtenida por el servicio.

Antes de usar esta *API de Google*, se ha intentado implementar el servicio de geolocalización usando tan sólo las librerías disponibles para ello en *Android*. Sin embargo, la única manera de conocer la localización GPS del usuario, usando estas librerías de *Android*, es obteniendo la última posición conocida (*getLastKnowLocation()*), que no siempre coincidirá con la posición GPS actual. Para evitar este inconveniente, se ha hecho uso de la *API de Google*, que permite obtener la posición actual aproximada mientras el servicio de mapas de *Google* esté activado. Así, estará recibiendo actualizaciones de la posición GPS según el intervalo de tiempo que se le haya facilitado en la implementación.

```
@Override
protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_modo_navigation);

    textMyLocation = (TextView) findViewById(R.id.textMyLocation);

    this.architectView = (ArchitectView) this.findViewById(R.id.architectView);
    final StartupConfiguration config = new StartupConfiguration("EAxUMzCetyJMu/rDt8/+JOcrdnF7GT
    this.architectView.onCreate(config);

    googleApiClient = new GoogleApiClient.Builder(this)
            .addApi(LocationServices.API)
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .build();

    locationRequest = new LocationRequest();
    locationRequest.setInterval(50 * 1000);
    locationRequest.setFastestInterval(10 * 1000);
    locationRequest.setPriority(LocationRequest.PRIORITY_BALANCED_POWER_ACCURACY);


}
```

Ilustración 32. Método onCreate() de la clase ModoNavigation.java (Código fuente Pocket Campus).

Para continuar con el desarrollo de la actividad ModoNavigation.java, se han declarado algunas variables de las que hará uso el servicio de geolocalización, como: el *locationProvider (LocationServices.FusedLocationAPI), el GoogleApiClient o el locationRequest*. A continuación, se implementa el método *onCreate()*, donde se introduce la clave de la licencia y se crea el *architectView*. Por otro lado, también se crea, dentro de *onCreate()* (Ilustración 32), el ***googleApiClient*** que facilitará todas las herramientas necesarias para usar el servicio de geolocalización. Además, se inicializa el *locationRequest*, pasándole algunos parámetros, como: el intervalo de tiempo para realizar peticiones de posición, un intervalo más rápido en caso de cambiar las condiciones del *locationProvider* y el tipo prioridad usada a la hora de obtener una posición (Precisión, carga de la batería, nivel de señal 3G, nivel de señal *WIFI*, etc).
En el caso del modo navegación, se ha elegido una prioridad de tipo balanceada, es decir, ha sido obtenida a partir de la media, entre precisión y potencia de carga de la batería, del dispositivo donde se esté ejecutando la app.

Además del método *onCreate()*, en la clase *ModoNavigation.java* se han implementado otros métodos propios del ciclo de vida de la aplicación: *onStart(), onPostCreate(), onResume(), onPause(), onStop() y onDestroy()*, todos ellos implementados en la clase *AppCompactActivity*. Cabe mencionar el uso de

[39] https://developers.google.com/maps/?hl=es

onResume(), utilizado para pedir actualizaciones de la localización si el *googleApiClient* está conectado, y el uso del *onPause()*, usado para detener las actualizaciones de posición.

```
@Override
protected void onResume(){
    super.onResume();

    architectView.onResume();

    if (googleApiClient.isConnected()){
        requestLocationUpdates();
    }
}

@Override
protected void onPause (){
    super.onPause();

    architectView.onPause();

    LocationServices.FusedLocationApi.removeLocationUpdates(googleApiClient, this);

}
```

Ilustración 33. Implementación de los métodos onResume() y onPause(), en el ModoNavigation.java (Código fuente Pocket Campus).

Por otro lado, se deben crear otros métodos usados para la gestión del servicio de geolocalización, ya que el *ModoNavigation.java* implementa las clases *GoogleApiClient.ConnectionCallbacks* y *GoogleApiCLient.OnConnectionFailesListener*. Dichos métodos son:

- ***onConnected():*** utilizado para lanzar el método *requestLocationUpdates()* y de esta forma, empezar a recibir actualizaciones.

- ***onLocationChanged(location)****:* utilizado para detectar cambios en la posición del usuario y guardar las coordenadas de la nueva posición; es decir, este método es el encargado de "obtener" las coordenadas de la posición actual del usuari, para poder pasárselas a *architectView.setLocation()*. Este método ubica al usuario en la experiencia de navegación y en relación a la ubicación de los diferentes *POIs* definidos.

  En el caso de *Pocket Campus* estos POIs coinciden con puntos del campus EPSG, por donde pasan las rutas habilitadas. Dichas rutas, parten de un punto de inicio en la zona central del campus, y llevan hasta cada uno de los accesos a los edificios o espacios de la EPSG.

```java
@Override
public void onConnected(Bundle bundle) {
    requestLocationUpdates();

}

private void requestLocationUpdates() {
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PE
        // TODO: Consider calling
        //    ActivityCompat#requestPermissions
        // here to request the missing permissions, and then overriding
        //   public void onRequestPermissionsResult(int requestCode, String[] permissions,
        //                                          int[] grantResults)
        // to handle the case where the user grants the permission. See the documentation
        // for ActivityCompat#requestPermissions for more details.


        return;
    }

    LocationServices.FusedLocationApi.requestLocationUpdates(googleApiClient, locationRequest, this);
}

@Override
public void onLocationChanged(Location location) {

    myLatitude = location.getLatitude();
    myLongitude = location.getLongitude();
    //myAltitude = location.getAltitude();
    myAccuracy = location.getAccuracy();

    try {
        architectView.setLocation(myLatitude, myLongitude /*, myAltitude*/, 20f);

    }catch (Exception e){

    }

    textMyLocation.setText("Your position is: " + "\n Latitude: " + myLatitude + "\n Longitud: " + myLongitude

}
```

Ilustración 34. Implemenación del método onLocationChanged(location), para llamar al método architectView.setLocation [40].

Una vez que el *ModoNavigation.java* está correctamente implementado, se crea el archivo HTML que estructura la experiencia de RA y carga los recursos necesarios. Para ello, se ha creado el archivo *indexNavigation.html*, donde se incluyen todos los *link*s y *scripts* (recursos *javascript y css*) que van a ser utilizados por el modo navegación (Ilustración 35).

```html
<link rel="stylesheet" href="css/default.css">

<!-- positioning of poi-radar -->
<link rel="stylesheet" href="css/poi-radar.css" />

<!-- jquery mobile CSS -->
<link rel="stylesheet" href="jquery/jquery.mobile-1.3.2.min.css" />

<!-- required to set background transparent & enable "click through" -->
<link rel="stylesheet" href="jquery/jquery-mobile-transparent-ui-overlay.css" />

<!-- jquery JS files -->
<script type="text/javascript" src="jquery/jquery-1.9.1.min.js"></script>
<script type="text/javascript" src="jquery/jquery.mobile-1.3.2.min.js"></script>

<!-- marker representation-->
<script src="js/marker.js"></script>

<!-- World logic-->
<script type="text/javascript" src="js/navigationWorld.js"></script>

<!-- radar component -->
<script type="text/javascript" src="js/radar.js"></script>
```

Ilustración 35. Recursos declarados en el indexNavigation.html (Código fuente Pocket Campus).

Los recursos más importantes, incluidos en el *indexNavigation.html* (Ilustración 35) son:

- ***navigationWorld.js***: este recurso *javascript* implementa el modo navegación en sí. Para ello, se han creado algunos métodos que sean capaces de gestionar la información de los POIs definidos y para manejar la interacción del usuario con la interfaz del modo navegación.

  En primer lugar, se debe declarar la variable *World* que implementará todos los contenidos de la experiencia de navegación.

[40] Fuente – Ilustración 30: (Código fuente Pocket Campus)

Dentro de ésta, como primer paso, se han declarado algunas variables útiles, como los *drawables* que representan los indicadores (diseñados en el apartado 3.5.2) o el *arrayList* llamado *markerList[]*, que contiene objetos JSON[41], lo cuales especifican la información necesaria para crear POIs (tratados como *markers* o marcadores, en el código).

A continuación, se ha creado una función clave para el modo navegación: *onLocationChangedFn().* Esta función sirve para pedir los datos iniciales del modo navegación, actualizar los valores de las distancias al usuario cuando el *flag* (o bandera) *initialLoadedData* (inicializado como *false*, al comienzo de la variable *World*) es igual a *false* (Ilustración 36).

```
if (!World.initiallyLoadedData) {

    World.requestInitialData();
    World.updateDistanceToUserValues();

    World.initiallyLoadedData = true;

} else if (World.initiallvLoadedData) {
```

Ilustración 36. initialLoadedData es igual a false, aún no se ha iniciado el modo navegación (Código fuente Pocket Campus).

Si por lo contrario, el valor de dicho *flag* es igual a true (es decir, la experiencia de navegación ya ha sido iniciada), entonces se actualiza el contenido según el valor de la opción seleccionada en el menú de destino y se actualiza la cuenta de POIs o indicadores activos (Ilustración 37).

```
} else if (World.initiallyLoadedData) {

    World.updateOptionSelected();
    // helper used to update placemark information every now and then (e.g. every 10 location upadtes fired)
    World.locationUpdateCounter = (++World.locationUpdateCounter % World.updatePlacemarkDistancesEveryXLocationUpdates);

}
```

Ilustración 37. initialLoadedDta es igual a true, ya se ha iniciado el modo navegación (Código fuente Pocket Campus).

Como se puede observar en las dos últimas ilustraciones, para realizar todas estas acciones se han utilizado algunos métodos implementados dentro de la variable *World*: *World.requestInitialData(), World.updateDistanceToUserValues() y World.updateOptionSelected()*.

```
requestInitialData: function requestInitialDataFn(){

        var poiData = [];

        World.showModalPanel('startModal');

        poiData.push({"id":("X"),"longitude":(-0.16568072140216827),"latitude":(38.99594383726395),"description":("This is the point where

        World.loadPoisFromJsonData(poiData);

},
```

Ilustración 38. Función requestInitialData (Código fuente Pocket Campus).

[41] http://www.json.org

```
// request POI data
updateOptionSelected: function updateOptionSelected() {

    $('#myselect').on('change',function(){

        var poiData = [];

        var valor = $(this).val();

        for (var i =0; i < poiData.length-1; i++){
              poiData.splice(i,6);      // max number of POI in a Route (Route of H Destination)
        }

        if (poiData.length != 0 ){
            alert("The POI List Data was not update correctly");
        }

        AR.context.destroyAll();

        if (valor === "A"){
            poiData.push({"id":("A"),"longitude":(-0.16568072140216827),"latitude":(38.99594383726395),"descripti
            poiData.push({"id":("A"),"longitude":(-0.16586646437644958),"latitude":(38.996303948820255),"descript
            poiData.push({"id":("A"),"longitude":(-0.16588658094406128),"latitude":(38.996670833383725),"descript
            poiData.push({"id":("A"),"longitude":(-0.16642838716506958),"latitude":(38.99667812936426),"descripti


            World.loadPoisFromJsonData(poiData);

        } else if (valor === "B"){
            poiData.push({"id":("B"),"longitude":(-0.16568072140216827),"latitude":(38.99594383726395),"descripti
            poiData.push({"id":("B"),"longitude":(-0.16587316989898682),"latitude":(38.99628206076058),"descripti
            poiData.push({"id":("B"),"longitude":(-0.16583427786827087),"latitude":(38.995739026255954),"descript

            World.loadPoisFromJsonData(poiData);

        } else if (valor === "C"){
```

Ilustración 39. Función UpdateOptionSelected()(Código fuente Pocket Campus).

Es importante mencionar que cuando se llama a la función *requestInitialData()* (Ilustración 38), ésta tiene implementada la llamada a la función *loadPoisFromJsonData()*, que es la encargada de cargar los datos asociados a los indicadores que se desea mostrar. En cambio, cuando el usuario selecciona un destino en el menú desplegable, se ejecuta la llama a la función *updateOptionSelected()*, que elimina todos los objetos de tipo *marker* (*marker.js*) de la lista de *markers* (indicadores) visibles o activos, y actualiza la lista con los nuevos *markers*, según la opción seleccionada.

Algo que se ha intentado, pero no se ha conseguido, es modificar la función *loadPoisFromJsonData()* para que cargue los datos referentes a los *markers* desde un archivo llamado *myJsonData.js*, que contiene la variable *jsonData* (de tipo *arrayJSON*); incluso preguntando a los asistentes para desarrolladores de Wikitude, que no sabían que podría estar ocurriendo, ya que todo parecía estar correcto. Por ello, los datos referentes a los indicadores, se han cargan directamente desde el código de la función *updateOptionSelected()* (Ilustración 39), qué es donde se llama a *loadPoisFromJsonData()*, para hacer visibles los nuevos *markers* o indicadores. Esto debería solucionarse más adelante.

Además de estas funciones, se han utilizado otras para manejar la interacción del usuario con los indicadores virtuales y elementos de la interfaz del usuario, manejar *popups* (paneles animados), etc., de esta manera, se completa la interacción con el usuario.

Para terminar de implementar *navigationWorld.js*, se ha de llamar a la función *onLocationChanged()* para inicializar modo navegación y que se carguen todos los contenidos iniciales.

- **marker.js**: esta clase ha sido rescatada de uno de los ejemplos facilitados junto al SDK de *Wikitude*, con alguna modificación. Básicamente, implementa un objeto de tipo *java* y recibe datos referentes a los POIs (*poiData*) para crear los indicadores que se harán visibles en la escena de navegación, en RA. Para ello, se define su apariencia, tamaño y forma, y, sobretodo, la información que mostrará. Para añadir cierta interacción, el objeto *marker* incluye una propiedad de animación, para emular el efecto de "*click*".

En el modo de navegación, cuando el usuario pulse el indicador, aparecerá un indicador (*ImageDrawable*), con forma de flecha, que apuntará en dirección al indicador seleccionado, siempre y cuando no esté presente en la escena visible (*architectView*) para no perder el rumbo al indicador seleccionado.

- **radar.js**: al igual que la anterior, se ha recatado de un ejemplo en la carpeta del SDK *Wikitude*, ya que es perfecto para el propósito del modo navegación. El objeto radar, representa sobre dos *drawables* (a modo de panel radar), unos puntos blancos que corresponden a la ubicación de los indicadores activos, en referencia al usuario (centro del elemento gráfico que representa el radar). Contiene una función que actualiza la posición cuando cambia la ubicación actual del usuario. Además, cuando se selecciona un indicador, éste se volverá de color rojo en el radar (antes blanco).

Estos son los principales recursos utilizados, ya que son los que forman la experiencia del modo navegación, en RA.

Para introducir los dos modos de funcionamiento de la app: exploración y navegación, se ha creado una pantalla de inicio, en la que aparece cierta información sobre el uso de estos y presenta los dos botones correspondientes a cada modo de la app. Además, se han implementado algunas actividades complementarias, a modo de *popups* (con información útil), para dar la sensación al usuario de navegar por la apps cuando interactúa con ella (*lookandfeel*).

Para terminar este apartado de desarrollo de la app, se ha de mencionar que una de las tareas más complicadas de lidiar, durante el desarrollo la app de RA, es la de testear o probar cada pequeño avance significativo logrado, en algunas partes de la app, ya que todas aquellas funciones que hagan uso de la tecnología de RA no pueden ser simuladas por el emulador de Android Studio, y se debe hacer instalando la app en un dispositivo físico. Se trata de un proceso un poco tedioso, por lo que se ha aprovechado la interfaz del usuario para recoger mensajes de alerta y depuración de errores, lanzados desde partes del código que se desean controlar. De esta manera, al menos, se pueden encontrar de manera más fácil los problemas, en caso de que surjan. Para instalar la app en el dispositivo, se han utilizado la app *easyInstaller* (descargada del *Play Store*) que esta instalada en el dispositivo, y el asistente de Android para instalar aplicaciones desde un Mac (disponible para descargar en la página de *Android*).

Además, cuando se desarrolla una aplicación de esta magnitud resulta indispensable utilizar una herramienta de control de versiones como Git, si no, el desarrollo podría acabar en un gran caos.

Por último, para visualizar una muestra del resultado obtenido, tras el desarrollo de la aplicación, se han realizado dos videos para mostrar cómo se vería el resultado final, para ambos modos de la app. Éstos se encuentra almacenados en una carpeta de Dropbox sobre el proyecto. El enlace para poder acceder a la carpeta es: https://www.dropbox.com/sh/xjr6tnyigwouybm/AACt13Fsy25Et_aoDp4suoTha?dl=0.

En la carpeta, se pueden encontrar dos videos, cuyos nombre para cada modo de la app, son:

- Modo Exploración: **demostraciónModoExploracion.**
- Modo Navegación: **demostraciónModoNavegación.**

4. Propuesta de Evaluación subjetiva para la app desarrollada

En este capítulo, se va a proponer una forma para **evaluar** la app desarrollada: Pocket Campus – Augmented EPSG. Esta propuesta, se basa en la realización de un cuestionario a varios usuarios, tras haber probado la aplicación, con sus diferentes modos y funcionalidades. Además, se ha grabado un **video-tutorial** explicando en qué consiste y cómo se debe usar esta app, para que los usuarios puedan visualizarlo antes de probar Pocket Campus. Si se desea visualizar el video-tutorial, este se puede encontrar en la carpeta de Dropbox: https://www.dropbox.com/sh/xjr6tnyigwouybm/AACt13Fsy25Et_aoDp4suoTha?dl=0, con el nombre de **pocketCampusVideoTutorial**. Además, los usuarios de *Pocket Campus* podrán acceder al video desde un botón con la palabra tutorial en la pantalla de inicio de la aplicación.

Tras investigar bastante sobre el tema de evaluación en experiencias con apps móviles, y, un poco, en apps de RA (Anexos), se ha definido un **cuestionario**, que puede servir para evaluar la experiencia de cada usuario, de una forma subjetiva, es decir, se evalúa la calidad e experiencia (Quality of Experience, QoE) en el uso de la app. Para que los resultados del cuestionario sean significativos, éste se debe pasar al menos a 10 usuarios, aunque cuanto mayor sea el número, más significativas serán las conclusiones obtenidas. Con el fin de realizar el test de la app, se pueden escoger personas de diferentes edades, sexo y nivel de estudios que estén dispuestos a probar una app de RA, ya que ésta destinada al uso de todos los públicos interesados.

El test, presenta varias cuestiones que pueden ser respondidas con valores del 1 al 5, lo cuáles corresponden a diferentes valoraciones subjetivas, como se puede comprobar en el siguiente enlace:

https://docs.google.com/forms/u/0/d/1sc729WH2vO6U7YfPCIfFS3RatNlgER1kfmlzlGOWtfk/edit

En dicho enlace puede realizarse la encuesta creada con la herramienta de *Google* para formularios. Este es enlace que visitarán los usuarios tras probar la app.

- ¿**Facilidad** de uso de la app?
- ¿**Comodidad** de uso de la app?
- ¿"***LookandFeel"*** de la aplicación o apariencia?
- ¿**Motivación** de uso de la app, dentro del contexto del Smart Campus EPSG?
- ¿**Fluidez** en el uso de la app? (Salvando las prestaciones del dispositivo utilizado)
- ¿**Utilidad** de la app, dentro del contexto del *Smart Campus* EPSG?
- ¿**Uso justificado** de la tecnología **de RA**, para el propósito de la app?
- ¿Sensación de **inmersidad**?
- ¿**Uso** de la app si estuviese disponible?
- Y por último, ¿**Valoración global** de la app?

Asimismo, el cuestionario incluye apartados en los que los usuarios pueden aportar sugerencias para la futura mejora de la app (p.ej., cambios u otras funcionalidades), así como opiniones sobre otros escenarios en lo que una app de este tipo pueda ser útil o ventajosa.

5. Futuros caminos de desarrollo

En este capítulo se comentan algunas opciones o propuestas, para mejorar la app *Pocket Campus – Augmented EPSG*, ya que la app desarrollada durante la realización de este proyecto se puede considerar una demo demostrativa de las posibilidades en este ámbito. De esta manera, se muestra en mayor grado el papel que la RA puede jugar en el contexto de *Smart Places* de la misma, y en particular de *Smart Campus*.

Algunos caminos inmediatos para mejorar la app desarrollada, pueden ser:

- Dentro del modo exploración, **seguir avanzando con** la implementación del ***info-map***: Durante la implementación de este *info-marker*, se ha intentado incluir un menú virtual en la vista del mapa 3D que permita al usuario seleccionar diferentes puntos del mapa. Cuando el usuario seleccione uno de ellos, el edificio que acoja dicho espacio se alzará levemente sobre la base del modelo 3D para dar *feedback* al usuario sobre donde se encuentra dicho punto o espacio en el mapa.

  El problema al hora de testearlo es que no carga el modelo 3D cuando se seleccionaba un punto para animar. No se ha encontrado ninguna solución.

  Por tanto, éste podría ser un aspecto a mejorar en la app desarrollada, además de la inclusión de otros elementos de interfaz y animaciones, que den al usuario la sensación de interacción directa con el *info-map* aumentado. El uso del plugin de *Wikitude* para *Unity*, podría resultar muy útil, ya que en *Unity* es más sencillo implementar este tipo de escenas de RA, si se consigue integrar una escena *Unity* en el proyecto *Android*.

  Otro aspecto mejorar, es el de crear un servidor de recursos media, que de soporte a la app. De este modo, la app no tendría que contener todos los recursos de los que hace uso, y por tanto, resultaría menos pesada de instalar. Para que no resulte perjudicial para la fluidez del funcionamiento de la app, se ha de elegir un servidor que: proporcione un alta velocidad de descargas de contenidos y que, sobre todo, ofrezca una alta capacidad de almacenamiento; ya que este tipo en este tipo de app el número de recursos puede convertirse en un problema, aunque por otro lado, el consumo de datos por parte del usuario, cuando utilice la app, también puede convertirse en un problema. Se ha de buscar la mejor opción, dependiendo el tipo y volumen de contenidos que vaya a manejar la app.

- **Mejora del modo Navegación**: hay varios aspectos en los que se podría mejorar este modo de la app.

  En primer lugar, el uso de un sistema de geolocalización, como el proporcionado por la *API de Google Maps*, a veces resulta un poco impreciso, aunque se preste atención al parámetro de precisión en la configuración del servicio, para el espacio que ocupa, en este caso, el campus de la EPSG. En ocasiones, se localizan varios POIs en 20 m^2 y el usuario se mueve por el entorno más rápido que la velocidad con la que se actualiza la posición del usuario, lo que puede provocar momentos de confusión. Para solucionarlo, se puede intentar combinar el uso de la señal GPS con la tecnología *SLAM*[42]. Esta modificación asegurará un mejor

[42] http://www.wikitude.com/wikitude-slam/

funcionamiento del modo navegación y cambiaría por completo el modo en el que la app guía al usuario, según el destino que seleccione. El único inconveniente es que aún no se incluye en las funcionalidades del SDK de *Wikitude* para *JavaScript*, tan sólo para la versión nativa de Android. Podría replantearse el entorno elegido para el desarrollo de la app.

Para mejorar este modo de la app, también se podría crear una base de datos que dé soporte a la aplicación, para almacenar la información de los POIs, por ejemplo. Ésta se encontraría alojada en un servidor y la app, se comunicaría con dicho servidor, para obtener los datos que requiera.

Estos, han sido los principales aspectos detectados para mejorar durante el desarrollo de esta app. Algunos de ellos, incluso, se han intentado incluir, pero la inexperiencia en el desarrollo de app, y más aún de RA, así como la limitación del tiempo, han imposibilitado el avance en dichas funcionalidades más avanzadas.

6. Conclusión

El margen de mejora de la app es muy alto: la RA es una tecnología que está desarrollándose aún, cada día con mayor alcance. Lo complicado es aplicar el uso de esta tecnología al caso de un ***Smart Place***, y desarrollar una app tan completa, que permita al usuario disfrutar de las más actuales funcionalidades en RA, como se ha intentado conseguir en la demo desarrollada.

Aun así, con la realización de este proyecto, se ha conseguido presentar una **clara propuesta demostrativa**, basada en el uso de esta tecnología, para contribuir con el concepto **de *Smart Campus***.

Para ello, se ha desarrollado una demo completa de la app, que da una idea sobre lo que podría llegar a ser la app ***Pocket Campus*** en un posible futuro. El objetivo principal para el desarrollador se ha cumplido satisfactoriamente: **investigar** y **aprender** sobre el uso de la tecnología de **Realidad Aumentada (RA)**, para crear un app basada en ella, que tenga cabida en la actividad del Campus de la EPSG. Siguiendo esta premisa, se conseguido implementar una app con dos posibilidades de uso: modo exploración (para ver contenidos multimedia aumentados, repartidos por el Campus) y modo de navegación (para ayudar al usuario a orientarse en el campus).

Se puede afirmar, con absoluta certeza, que la RA cambiará por completo la forma en la que se percibe el mundo, por lo que surge la ocasión idónea para contribuir con una buena aplicación basada en ésta, que **resulte útil, entretenida y accesible**.

Referencias

- © Wikitude GmbH. *Development - Documentation.* 2012-2016. http://www.wikitude.com/developer/documentation/android.

- A. D¨unser, R. Grasset, and M. Billinghurst. «A survey of evaluation techniques used in augmented reality studies.» Singapore, 2008.

- Corpuz, John. *www.tomsguide.com.* 12 de Julio de 2016. http://www.tomsguide.com/us/pictures-story/657-best-augmented-reality-apps.html.

- D. A. Bowman, J. L. Gabbard, and D. Hix. *Presence – Teleoperators and Virtual Environments.* Vol. 11, de *A Survey of Usability Evaluation in Virtual Environments: Classification and Comparison of Methods*, de D. A. Bowman, 404 - 424. 2012.

- Furht, Borko. *"Handbook of Augmented Reality".* New York: Springer-Verlag. Springer Science + Business, 2011. ISBN 978-1-4614-0063-9 e-ISBN 978-1-4614-0064-6.

- J. E. Swan and J. L. Gabbard. «Survey of User-Based Experimentation in Augmented Reality.» Las Vegas , 2005.

- Mark Billinghurst, Andreas D¨unser. «Evaluating Augmented Reality Systems (cap V - User Based Evaluation).» En *Handbook Of Augmented Reality*, de Borko Furht, 296 - 298. New York: Springer, 2011 2011. ISBN 978-1-4614-0063-9 e-ISBN 978-1-4614-0064-6.

- P. Milgram, A.F. Kishino. *"Taxonomy of Mixed Reality Visual Displays".* Vols. E77-D(12), de *IEICE Transactions on Information Systems*, de P. Milgram A.F. Kishino, pp. 1321–1329. 1994.

- Ronald Azuma, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier, Blair MacIntyre. «"Recent Advances in Augmented Reality".» *IEEE*, November/December 2001.

- UCAM. *www.murcia.es.* 2016. https://www.murcia.es/medio-ambiente/medio-ambiente/educacion1/material%5CPresentaciones%5CRAFAEL%20MELENDRERAS%20RUIZ%20UCAM.pdf. 15/06/2016.

- Wikipedia.org. *www.wikipedia.org.* 15 de Enero de 2001. https://www.wikipedia.org (último acceso: 2016).

- Carlos Sánchez, Eduardo Vendrell y José Luis Diez, 2011, UPV3D: Visualización interactiva online de un campus universitario, Revista Iberoamericana de Automática e Informática industrial.

Índices

I. Glosario de abreviaciones y términos

RA: Realidad aumentada. *En inglés, AR (Augmented Reality).*
VA: Virtualidad aumentada. *En inglés, AV (Augmented Virtuality).*
RV: Realidad Virtual. *En inglés, VR (Virtual Reality).*
EV: Entorno Virtual. *En inglés, VE (Virtual Enviroment).*
SDK: *En inglés, Software Development Ki, es decir,* Kit de software para el desarrollo.
POI: *Point of Interest.* En castellano, Puntos de Interés.
GPS: *Global Positioning System*. En castellano, Sistemas de Posicionamiento Global.
Target: En castellano meta, objetivo o diana. Define el elemento físico (imagen) que se debe capturar con la cámara para visualizar el contenido virtual aumentado.
Marker: indicador. Se trata de un Panel 2D que contiene la información del POI..

II. Tabla de ilustraciones