Realidad aumentada sin marcadores: posibilidades, librerías y prueba de concepto.

Por
Colin Ulrich Cop, Patricia Cabrero Villar, David González Jiménez

UNIVERSIDAD COMPLUTENSE
MADRID

Grado en Grado en Desarrollo de Videojuegos
Facultad de Informática

Pedro Pablo Gómez Martín
Realidad aumentada sin marcadores: posibilidades, librerías y prueba de concepto.

Madrid, 2018–2019

Autorización de difusión y uso

Sobre TEFLON

Teflon(cc0 1.0(documentación) MIT(código))es una plantilla de LaTeX creada por David Pacios Izquierdo con fecha de Enero de 2018. Con atribuciones de uso CC0.

Esta plantilla fue desarrollada para facilitar la creación de documentación profesional para Trabajos de Fin de Grado o Trabajos de Fin de Máster. La versión usada es la 1.3.

V:1.3 Overleaf V2 with pdfLaTeX, margin 1in, NO-bib

Contacto
Autor: David Pacios Izquiero
Correo: dpacios@ucm.es
ASCII: asciifdi@gmail.com
Despacho 110 - Facultad de Informática

Índice general

Página

Resumen

Abstract

Capítulo 1

Introducción

1.1. Motivación

En los últimos años la realidad aumentada se ha convertido en una tecnología madura y presente en la mayoría de los usuarios. Apoyada por el auge de dispositivos móviles inteligentes y la mejora de los componentes de estos la realidad aumentada es adoptada cada vez más por un mayor público. Se trata de un tema de actualidad recurrente y de referencia en múltiples productos de innovación. Se estima que el tamaño del mercado

de la RA crecerá de 3.5 mil millones en el 2017 a más de 198 mil millones de dólares en el 2025.[30] En los años próximos se espera que revolucione mercados como son el arte, la educación, la publicidad, procesos de fabricación y montaje, turismo y especialmente el mundo de los videojuegos entre otros. Debido a este gran crecimiento del sector es un excelente tema para tratar de cara a conocer las limitaciones y puntos a destacar de cada una de las tecnologías existentes en la actualidad.

1.2. Objetivos

El objetivo principal del desarrollo de este proyecto será conocer las posibilidades y limitaciones de las librerías de realidad aumentada sin marcadores existentes en la actualidad. En base a este objetivo se fijaron los siguientes objetivos específicos:

a). Investigación de las principales librerías de realidad aumentada sin marcadores.

b). Implementación de pruebas de concepto de cada una de ellas para analizar los pros y contras.

c). Análisis de los resultados obtenidos de las pruebas de concepto.

d). Planteamiento e implementación de diferentes aplicaciones de realidad aumentada sin marcadores en función de los resultados obtenidos en el análisis.

e). Desarrollar una aplicación/concepto de realidad aumentada en multijugador con los cloud anchors.

1.3. Metodología

Para llevar a cabo estos objetivos se investigará a través de fuentes en internet, artículos científicos, estudios previos y libros, todos ellos reflejados en la bibliografía y webgrafía. Estos recursos serán la base de la fundamentación del proyecto y por ello se abrirán dos vías de investigación principales: se analizará por una parte acerca de las diferentes librerías de desarrollo en realidad aumentada y por otra las diferentes aplicaciones en el mercado de esta tecnología.

La revisión bibliográfica que se llevará a cabo vendrá definida por las dos áreas del conocimiento que se deben investigar de cara a desarrollar el objetivo principal planteado. En el campo del conocimiento técnico de la realidad aumentada se investigará a través de bibliografía recomendada por profesores de la Facultad de Informática de la Complutense. Los libros más significativos por tratar serán "*Handbook of Augmented Reality*" de Borko Furht, "*Augmented reality games I, Understanding the Pokémon GO phenomenon*" de Vladimir Geroimenko y "*Augmented reality games II, The gamification of education, medicine and art*" de Vladimir Geroimenko.

Debido al continuo cambio que experimentan las tecnologías de realidad aumentada gran parte de la investigación se verá supeditada a artículos científicos, así como a la documentación de las distintas librerías que lideran el mercado.

Una vez completada la investigación teórica se testearán diferentes aplicaciones ya existentes con el objetivo de encontrar sus fortalezas y debilidades. A través de estas conclusiones se podrá llevar a cabo una prueba de concepto más solida y veraz evitando cometer errores anteriormente observados. Se buscará conocer las características

específicas de las librerías escogidas delimitando los pros y contras de cada una de ellas. Para identificarlas se realizará un test definido y cerrado, poniendo a prueba las diferentes librerías en el mismo dispositivo de cara a establecer una comparativa entre todas. Estas aplicaciones test se desarrollarán cuando sea posible en el entorno Unity. Por último, se

desarrollarán tres aplicaciones de mayor nivel de complejidad que nos permitan explotar las virtudes de tres librerías diferentes de cara a mostrar y fijar las conclusiones extraídas del anterior estudio.

A continuación, se detallan las tecnologías que se utilizarán a lo largo del desarrollo del proyecto.

Para la creación de las aplicaciones, se utilizará como entorno de desarrollo principalmente **Unity 2019.3**, uno de los motores de videojuegos punteros y referentes en la industria. Gracias a su versatilidad e interfaz intuitiva nos permitirá iterar rápidamente a lo largo de los test y pruebas de concepto. Este motor es uno de los escogidos por la facultad para estudiar a lo largo del grado de desarrollo de videojuegos con lo que nos resultará más cómodo y familiar, favoreciendo de nuevo la agilidad en el desarrollo.

Como IDE [1] se utilizará **Visual Studio 2019** acompañado por Visual Studio Tools

[1] IDE: Integrated Development Environment (Entorno de desarrollo integrado)

para Unity una extensión gratuita de Visual Studio que lo convierte en una completa herramienta con la que desarrollar aplicaciones y juegos multiplataforma con Unity. Esta herramienta permite la integración de Visual Studio con el editor de Unity haciendo más eficaz el desarrollo. El sistema de control de versiones escogido será **Github** ya

que estamos habituados a la plataforma y nos permite integrarlo con herramientas como Visual Studio y Microsoft Teams.

La herramienta de seguimiento de tareas y comunicación entre el equipo escogida será **Microsoft Teams** dada su versatilidad y posibilidad de añadir herramientas.

Gracias a todas estas metodologías tanto prácticas, teóricas y técnicas y a las se lograrán cumplir los objetivos propuestos.

1.4. Plan de trabajo

La primera parte del trabajo consistirá en informarnos e investigar sobre las librerías de RA sin marcadores que lideran el mercado esta parte se realizará conjuntamente por los tres. Se estudiarán las diferentes tecnologías que componen la experiencia de la RA en los dispositivos móviles para entender cómo funciona a bajo nivel y estar actualizados con la demanda del mercado de cara a poder hacer una prueba de concepto verosímil con las aplicaciones de RA actuales.

En una segunda parte, después de identificar las librerías que existen en el mercado, se probarán (las que sean posibles) teniendo en cuenta los dispositivos y plataformas soportadas por cada una de ellas, a su vez se tendrá en cuenta la existencia de una licencia gratuita o de prueba. Se desarrollarán diversas aplicaciones de carácter básico a modo de test, éstas nos permitirán poner a prueba cada una de las principales librerías de realidad aumentada sin marcadores, comprobando su eficiencia. Los test serán ejecutados en las mismas condiciones lumínicas y con el mismo dispositivo de cara a obtener una mayor precisión en la comparación. En este caso nos dividiremos el desarrollo de los test de manera equitativa.

Una vez que se hayan encontrado las librerías que mejor se ajusten a nuestras necesidades, se desarrollarán las pruebas de conceptos las siguientes pruebas de concepto:

- Montar muebles de Ikea: se mostrará el proceso de montaje de un mueble en realidad aumentada. Ayudando al usuario a montar cada una de las piezas del proceso.
- Juego multijugador usando *cloud anchors*: se desarrollará un juego en el que poder poner a prueba las tecnologías de realidad aumentada sin marcadores combinada con puntos de localización en la nube permitiendo crear un juego multijugador.
- Visualizador de modelos 3D con las gafas Aryzon: se hará uso de unas gafas de realidad aumentada tipo *cardboard* en las que podremos ver superpuesto un modelo 3D, se podrá interactuar con el modelo 3D a través del mando de Xbox permitiendo rotarlo, escalarlo y moverlo.

Capítulo 2

Antecedentes y estado del arte

2.1. Definición

Se llama realidad aumentada al conjunto de tecnologías que permite a un usuario ver contenido virtual superpuesto al mundo real mediante un dispositivo tecnológico. Las tecnologías de realidad virtual se diferencian en que sumerge al usuario dentro de un entorno completamente sintético, sin tener consciencia del mundo real que lo rodea. La Realidad Aumentada no sustituye la realidad, sino que la complementa y puede mejorar la experiencia en ciertos ámbitos.

2.2. Historia

En la década de 1950, surgió por primera vez el término realidad aumentada cuando Morton Heilig, un cinematógrafo, pensó en un prototipo de un cine que estimulara todos los sentidos del ser humano de manera efectiva. Años más tarde, concretamente en el 1962, Heilig construyó dicho prototipo, llamado Sensorama, se trataba de un cine inmersivo y novedoso que incluía funcionalidades como 3D, visión angular (actualmente conocido como IMAX), vídeo en color, sonido en estéreo, además de estimular otros sentidos con aromas, viento, y vibraciones como podemos observar en la figura 2.1.

Figura 2.1: Cartel publicitario de la máquina Sensorama

En el 1968, Ivan Sutherland, inventó el HMD (*Human Mounted Display*), siendo así, el primer sistema que permitía ver las aristas de sencillos objetos 3D (*Wireframe*) en tiempo real. Empleaba dos sistemas de *tracking* para calcular el registro de la cámara; uno mecánico y otro basado en ultrasonidos. Debido a su gran peso, se decidió colgar el artefacto en el techo, como se puede ver en la figura 2.2.

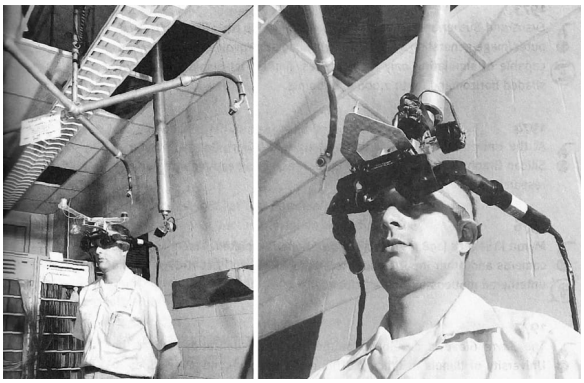


Figura 2.2: Ivan Sutherland "Espada de Damocles" 1968.

Sin embargo, no fue hasta 1992 cuando se acuñó el término de Realidad Aumentada por Tom Caudell y David Mizell, dos ingenieros de Boeing que proponían el uso de esta novedosa tecnología para mejorar la eficiencia y experiencia de las tareas realizadas por operarios humanos asociadas a la fabricación de aviones. La aparición del primer videojuego en realidad aumentada ocurrió en el año 2000. Bruce Thomas demostró en el ISWC (The International Symposium on Wearable Computers) su videojuego ARQuacke. El sistema empleaba una brújula digital, un receptor de GPS y métodos de visión basados en marcas [5]. Los jugadores tenían que llevar una especie de ordenador portátil a la espalda, un casco de visión estereoscópica y un mando de dos botones [2] como podemos ver en la figura 2.5 . El funcionamiento del HMD, como se puede ver en la figura 2.3 consistía en un divisor de haz que recibía la imagen virtual desde una pantalla que permitía ver el mundo real y el virtual en los ojos del usuario. La imagen resultante, como se puede observar en la figura 2.4 permitía una inmersión muy lograda para la época.

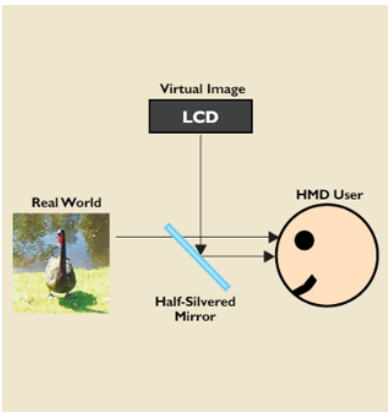


Figura 2.3: Funcionamiento del Head Mounted Display

Figura 2.4: Ejemplo de ARQuake. Imagen sacada de(http://www.tinmith.net/arquake/)

Figura 2.5: Dispositivo utilizado para ARQuake

Imágenes de las figuras 2.3 y 2.5 sacadas del libro de ARQuake [2]

2.2.1. Immersive computing

En 2007, en el ISMAR(Simposio internacional de realidad aumentada y mixta), Klein y Murray presentan el algoritmo PTAM (Parallel Tracking and Modeling) una variante al SLAM, que separa la localización y el mapeado en hilos diferentes. El SLAM es un algoritmo que sirve para que localizar la posición dentro de un entorno y modelarlo, más tarde lo explicaremos mas a fondo. Separar estos dos procesos en hilos diferentes permitía conseguir unos resultados en tiempo real muy sólidos.

En 2008 se creó Wikitude, una aplicación que utilizaba el GPS para mostrarte información de la Wikipedia según el lugar en el que estuvieras, pudiendo aprender datos sobre monumentos, esculturas o construcciones que te fueras encontrando. Como se puede ver en la figura 2.6, el cámara está enfocando un castillo y el móvil le dice cuál es, además le aporta información adicional como la distancia a la que está y el origen del castillo.

Figura 2.6: Wikitude App 2008

Un año más tarde, en 2009, desarrolló el videojuego ARhrrrr! del género shooter, el primer videojuego en realidad aumentada para smartphone con contenido 3D de alta calidad. Se generaba un mapa 3D sobre un marcador, y el objetivo era rescatar a los humanos de la ciudad y matar a los zombies (ver figura 2.7) [5].

Figura 2.7: Ejemplo del videojuego ARhrrrr!. Imagen sacada de (https://github.blairmacintyre.me/site-archive/ael-2015/research/games/arhrrrr/)

En el mismo año, el estudio español Novorama crea el videojuego de PSP (PlayStation Portable) Invizimals, un éxito mundial, vendiendo más de 8 millones de copias en todo el mundo en el primer trimestre de 2010. Este juego necesita usar la cámara adicional que se conecta a la PSP, y gracias al uso de los marcadores, se puede registrar la posición del jugador.

2.2.2. Project Tango

En 2014, Project Tango nació como uno de los primeros desarrollos de realidad aumentada pensado para ser distribuido mundialmente en los smartphones. En 2015, Tango pasa a formar parte de Google. El objetivo era crear un dispositivo portátil que permitiese mapear espacios 3D. Esta tecnología solo se desarrolló para el dispositivo Phab2 Pro de Lenovo, el cual incluía un mayor número de cámaras, concretamente 3. Las tres funcionalidades principales para el desarrollo de esta tecnología han sido:

- Seguimiento del movimiento: Se trata del uso de las características visuales del entorno combinadas con los datos proporcionados por los sensores de movimiento incorporados en el teléfono, el acelerómetro y por el giroscopio, teniendo como objetivo realizar un seguimiento de los movimientos hechos del dispositivo.
- Reconocimiento del ambiente: Tango almacena la información del entorno que le rodea, buscando los puntos característicos en cada fotograma que recibe de las cámaras (más adelante explicaremos con más detalle cómo funciona el reconocimiento del ambiente).
- Percepción de profundidad: Gracias a las cámaras especiales que incorpora el dispositivo, Tango puede calcular tamaños y distancias en el entorno que se encuentra.

Después de 3 años, en 2017, Google decide cerrar el desarrollo de Tango y se centra en su tecnología actual, ARCore, que es el competidor directo de ARKit, la librería de Apple.

2.3. Aspectos técnicos

2.3.1. Descripción

La realidad aumentada es una tecnología que permite entremezclar el mundo virtual con el mundo, añadiendo la información virtual a la información física ya existente en tiempo real, permitiendo al usuario comprender mejor el entorno que le rodea. Hace unos pocos años, con la aparición de los dispositivos móviles más potentes y las librerías gratuitas de realidad aumentada, el desarrollo de esta tecnología se ha vuelto muy accesible para cualquier desarrollador. Esto implica en el número de ideas y aplicaciones que se generan cada día, que va aumentando drásticamente con el paso de los años.
En la actualidad existen tres tipos de realidad aumentada:

- Realidad Aumentada con marcadores (2D y 3D): Los marcadores pueden ser imágenes impresas o dibujos en los que la aplicación reconoce el marcador y activa la experiencia sobre dicho marcador.

- Realidad Aumentada sin marcadores: Esta es la tecnología más novedosa, ya que combina diferentes tecnologías como (SLAM, seguimiento del movimiento, reconocimiento del ambiente, detección de planos ...) para proyectar el objeto y mantenerlo en el mismo punto de anclaje sin ayuda de ningún marcador.
- Realidad Aumentada por Geolocalización: Este tipo de experiencias vinculan a la RA con una ubicación geolocalizada específica. Normalmente se utilizan en exteriores y proporcionan información contextual sobre el ambiente que nos rodea.

Las posibilidades de la realidad aumentada sin marcadores están en pleno auge y cada vez aparecen más aplicaciones que mejoran la experiencia de usuario y facilitan el trabajo en algunos sectores como puede ser la fábrica, la arquitectura, la medicina, la educación y muchos más.

2.3.2. Métodos de tracking

El Tracking es como conocemos al proceso de localización espacial del usuario en un entorno. Es uno de los aspectos clave en el desarrollo de aplicaciones de realidad aumentada ya que cuanto mejor sea la estimación de la posición y orientación del dispositivo sensor, mejores y más acertados serán los resultados y la inmersión por parte del usuario.[6]
El cálculo del *tracking* se encarga de posicionar la cámara relativamente a los objetos de la escena. Existen multitud de tecnologías y métodos para llevarlos a cabo, siendo los más comunes sensores mecánicos, magnéticos, sónicos, dinámicos y basados en visión. Éstos últimos son los más extendidos, ya que la mayoría de los dispositivos desde los que se despliegan las aplicaciones de realidad aumentada, como móviles o tablets, disponen de una o varias cámaras. [5]
El *tracking* basado en cámaras de visión es un subcampo del *tracking* 3D, en el que se utilizan algoritmos de visión por ordenador para obtener de la manera más precisa posible el posicionamiento de seis grados de libertad del dispositivo (tres grados de posición y otros tres de orientación.
En este tipo de posicionamiento es necesario disponer de un conjunto de marcadores o referencias tridimensionales para situar la cámara con respecto a ellas. Aunque recientemente se ha tendido a utilizar en menor medida los marcadores físicos para dar una experiencia más rápida y cómoda al usuario, han sido una herramienta imprescindible en los primeros pasos de la realidad aumentada para la obtención de la localización relativa de la cámara. Según David Marimón[3], fundador y director general de Catchoom, se pueden distinguir dos aproximaciones distintas a la hora del tracking: los métodos Bottom-Up y los Top-Down.

Bottom-Up

Las aproximaciones del tipo Bottom-Up pretenden obtener la posición del dispositivo basándose en la información que recibe a través de la cámara. Para este método de tracking la posición y orientación se calculan en base a la obtención de características geométricas de objetos y sus relaciones. Dependiendo de los datos procesados, el seguimiento puede ser con marcas o sin ellas.
El tracking basado en marcas era el método más extendido en los inicios de la realidad aumentada. Hace sus cálculos con la ayuda de marcadores físicos que en su mayoría

presentan un gran contraste entre blanco y negro para que los sensores puedan percibirlos con mayor facilidad. Existen también marcadores que usan códigos de colores y diferentes formas geométricas, aunque después de ser sometidos a prueba se comprobó que los más sólidos eran los marcadores cuadrados. Por otra parte, este método es especialmente sensible a la oclusión, ya que cuando se pierde el marcador, es imposible calcular la posición del dispositivo. Por este motivo, se han diseñado marcadores que puedan hacer frente a este problema con imágenes en escala de grises que completan marcas que no son visibles.
Paralelamente, el *tracking* sin marcas se basa únicamente en las características intrínsecas de la escena, estructuras físicas de fácil percepción como las esquinas de una mesa.
Existen técnicas en este campo que utilizan información sobre superficies planas detectadas en el campo de visión, siendo su principal inconveniente su alto coste computacional (actualmente este tipo de localización no lo pueden llevar a cabo todos los dispositivos del mercado).
Por otra parte, hay técnicas basadas en modelos. No están considerados marcadores porque son parte del medio natural, pero al igual que con éstos los cálculos se basan en el reconocimiento de los objetos que existen y que el programa está preparado para procesar.
Finalmente, existen métodos que actúan en escenarios donde no se es capaz de obtener planos o modelos. Se suelen emplear restricciones epipolares de las cámaras en movimiento. Sin embargo, esta técnica no es utilizada habitualmente por sus altos requisitos de cómputo.

Top-Down

Las aproximaciones del tipo Top-Down intentan estimar desde la posición actual del dispositivo si se está percibiendo lo que se esperaba. Es decir, primero se estima la posición y después se confirma esa estimación con los datos del medio. En este caso, se emplean modelos del movimiento basados en filtros bayesianos para hacer una predicción de la localización del dispositivo. Partiendo de esta estimación, se busca mediante la cámara una serie de referencias parciales que corrijan la predicción y mejoren el posicionamiento del observador. Por ello, todos los modelos Top-Down se ven obligados a trabajar con filtros y modelos de asociación de datos. El uso de estos filtros permite combinar varios métodos de tracking y mantener un registro constante de los objetos y la cámara, aunque los marcadores, modelos o planos sean parcialmente visibles por oclusión o se hayan escapado del campo de visión. Además del seguimiento óptico, se han desarrollado numerosas alternativas con las que proporcionar otros métodos de localización (como los beacons o la ubicación del GPS) y así complementar y facilitar una localización más precisa y correcta. A las aproximaciones que se valen de varias de estas técnicas se las denomina métodos de fusión.

2.3.3. Tecnologías implicadas en la RA sin marcadores

El objetivo de la realidad aumentada es integrar contenido virtual en el mundo real. Idealmente, dicho contenido se tendría que comportar exactamente como uno real, esto requiere una información muy precisa sobre la posición del dispositivo que usa el usuario con respecto al objeto virtual. Para ello, se han desarrollado diferentes tecnologías que

junto a los sensores de los teléfonos actuales (giroscopio, acelerómetro, sensor de luz) y a la cámara, permiten disfrutar de una experiencia casi ideal. [1]

SLAM(Simultaneuos localization and mapping)

Mapeo y localización simultáneos se le llama a la tecnología que se basa en una serie de algoritmos complejos que utiliza los datos de los sensores para construir un mapa de un entorno desconocido y a su vez para saber dónde está localizado el dispositivo. Esta técnica es usada por robots y por vehículos autónomos. ¿Cuál es el objetivo? Descubrir donde estoy. La tecnología SLAM, en el momento que empieza el algoritmo, no tiene ningún tipo de información del entorno. Normalmente, sólo tarda unos pocos segundos en crear un mapa aproximado del entorno con lo que calcula una posición inicial. Más adelante, el mapa creado va creciendo y mejorando en base a la información que obtiene desde el fotograma de la cámara. Aunque este término empezó a aparecer en la década de los 90, las primeras implementaciones carecían de cámaras o sensores que proporcionaban información visual. En 2005 comenzó a abaratarse el coste de los ordenadores y de las cámaras, y los investigadores empezaron a combinar el SLAM con sensores visuales. Hasta este punto, esta tecnología estaba pensada para la navegación con robots en entornos desconocidos, hasta que en 2007 Georg Klein y David Murray vieron el potencial de usar esta tecnología en la realidad aumentada. [8]

Reconocimiento del ambiente

Cuanto mejor entienda la aplicación como es el entorno que le rodea, mejor será la experiencia de usuario. En este apartado, también entra el reconocimiento de superficies, tanto horizontales como verticales. El funcionamiento de esta tecnología consiste en procesar cada fotograma obtenido por la cámara y encontrar puntos característicos, estos puntos pueden ser cualquier cosa que ayuden a identificar objetos (esquinas, líneas, bordes de objetos, colores, gradientes, etc…), por lo que si se intenta detectar un plano en una superficie donde el color sea uniforme, y carezca de textura o patrones, como puede ser una pared totalmente blanca, seguramente no funcione con normalidad [22]. Con estos puntos, luego se construye una maya que va a servir como superficie en la escena de nuestra aplicación y con la cual podremos interactuar activándole las físicas y colisión.

Estimación de la luz

Esta tecnología es muy importante ya que aporta un nivel de detalle excelente, los modelos 3D virtuales se comportan como si fueran reales, se iluminan con la iluminación del mundo físico y emiten sombras. La estimación de la luz es posible gracias a la combinación de la información del seguimiento del movimiento y usando un algoritmo de análisis de imagen que determina la intensidad de la luz en la imagen del dispositivo. Cuanto más se mueva la cámara y más información recoja sobre el entorno, más precisos serán los datos de dónde viene la mayor fuente de luz, analizando el nivel de brillo de los píxeles de los fotogramas, por lo que se puede estimar la dirección en la que viene. A esta información se puede acceder desde Unity o el motor que se use, donde se le aplican los valores obtenidos a una la luz posicional.

Oclusión

El término oclusión se refiere a cuando un objeto nos impide ver otro objeto o imagen que hay detrás. Para disfrutar de una experiencia de realidad aumentada realista, esta tecnología es esencial, los objetos virtuales tienen que seguir esta regla, porque en el momento en el que cruza una persona delante del objeto, o cruzas la esquina, y sigues viendo el objeto virtual, se arruina la inmersión que podemos llegar a tener. Por lo que no sirve únicamente saber dónde está situado nuestro dispositivo con respecto al objeto, si no también hace falta saber si hay otro objeto o superficie en medio.[4]

Detección de rostros

Uno de los puntos más importantes de la realidad aumentada en la actualidad es la detección de caras y su reconocimiento.

Cada cara está compuesta por al menos 80 rasgos distinguibles, como la distancia que existe entre los extremos de la mandíbula, la profundidad de las cuencas oculares o la separación que hay entre los agujeros de la nariz. [7]Los humanos somos especialmente buenos reconociendo estos rasgos porque tenemos una zona del cerebro dedicada específicamente a interiorizar patrones. Basándonos en el funcionamiento del cerebro de una persona hemos desarrollado algoritmos que imitan estas asociaciones, dividiendo las caras en un conjunto de puntos de referencia a los que llamamos nodal points y buscando correspondencias con otras fotos tomadas anteriormente. El algoritmo que sigue un sistema para tratar de identificar si lo que está viendo es una cara pasa por comprobar si existe en la imagen un patrón similar al que formarían normalmente los rasgos más característicos de una cara, para después preguntarse de quién es esa cara. Sin embargo, no existen dos fotos de una misma persona que sean iguales, de manera que los algoritmos tienen que lidiar con 4 problemas fundamentales a la hora de reconocer la cara de una persona: el envejecimiento, la pose, la iluminación y las emociones.

En los últimos años se ha desarrollado un sistema de reconocimiento en 3D llamado Deepface, que es capaz de tomar una foto en 2D de un individuo y crear un modelo tridimensional. De esta forma, el sistema tendrá muestras de los rasgos faciales desde todos los ángulos disponibles, solucionando así el problema de la pose.

Por otra parte, el problema del envejecimiento también ha sido debidamente prevenido, ya que al crear la estructura 3D de la cara se tienen en cuenta los nodal points más importantes y que menos varían con el transcurso de los años, como son las curvas de los ojos, de la nariz o de la barbilla. Pero lo que realmente ha supuesto un avance en este campo es el Deep Learning, que es un sistema de algoritmos que guía al programa redirigiéndole si va por mal camino. Cada vez que asocia una cara correcta o incorrectamente, registra el proceso por el que ha pasado para realizar la comprobación y queda guardado en un mapa que va ampliándose sucesivamente con cada acierto o error del sistema. De esta manera, cuantas más conexiones se creen mayor será la fiabilidad a la hora de reconocer una cara.

Facebook por ejemplo se vale de este método para el reconocimiento facial y posee una red neuronal de más de 20 millones de nodos, con una fiabilidad del 97.35 %(datos de 2015)[37] y que aun así es inferior a la capacidad de detección de una persona.

Se espera que con la mejora de la tecnología el reconocimiento facial sea una forma de identificación tan válida como las huellas dactilares y que puedan identificarse las caras de las personas incluso en grabaciones de seguridad en calidad baja y sin color, así como un método de reconocer el género, edad y otras características del individuo para ofrecerle un servicio o producto más acorde con él en ámbitos como la publicidad.

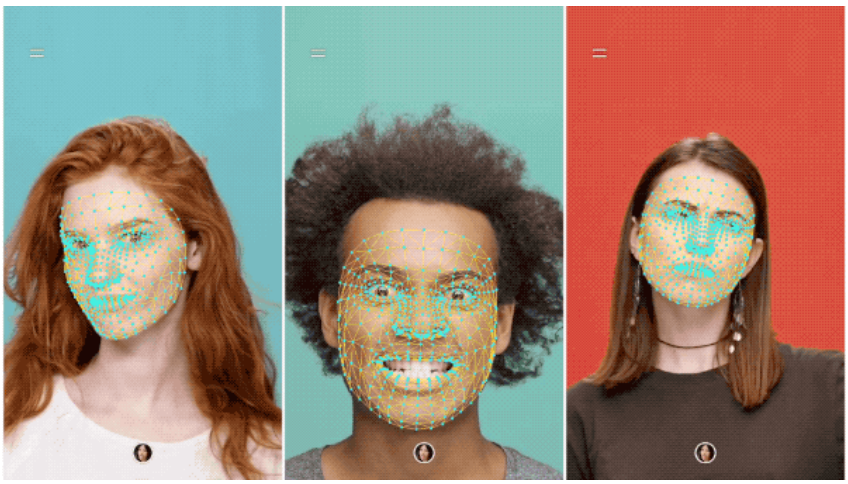


Figura 2.8: Detección de rostros

Puntos de ancla en la nube (Cloud Anchor)

El Cloud Anchor es un mecanismo que permite a los usuarios de una aplicación de realidad aumentada añadir objetos virtuales a una escena. De esta manera múltiples usuarios pueden interactuar y ver los mismos objetos desde dispositivos distintos, pero compartiendo un mismo espacio físico. Su funcionamiento es muy similar al de los anchors comunes, que se utilizan para fijar un objeto en una posición, con la diferencia de que los Cloud Anchors se hospedan en los servidores de Google. De esta manera varios dispositivos pueden consultarlos para situar los objetos en la aplicación.

Para ser utilizados, la aplicación en cuestión tiene que tener conexión a internet. Los Cloud Anchors son actualmente propios del SDK de ARCore, están soportados tanto en Android como en iOS (siempre que el dispositivo lo permita) y funcionan de la siguiente manera: ARCore tiene que generar primero un mapa de las proximidades del punto de ancla que será el centro de interés. Para ello, la cámara recopila información y características del entorno cercano desde diferentes ángulos y posiciones durante 10 segundos. Cuanto más precisa sea la información recopilada, mejor será la experiencia del usuario. Una vez transcurrido el tiempo, los parámetros del punto se hospedan en la nube y se establece el anchor, devolviendo el servidor un número de identificación único (el Cloud Anchor ID). Cuando otro usuario de la aplicación dirige su cámara hacia el mismo punto de interés, el Cloud Anchor procesa las características visuales del entorno físico desde el nuevo punto de vista. Estas características son comparadas con el mapa 3D que se ha generado anteriormente por el otro dispositivo y se establece la posición y orientación del nuevo usuario con respecto a ello para que pueda ver los objetos virtuales con la mayor precisión posible.

Para identificar un punto de ancla en la nube desde otro dispositivo se debe apuntar al lugar en que está situado sin importar la posición del dispositivo, siempre y cuando haya una línea recta entre ambos y no estén separados por una distancia superior a 10 metros.

En el caso de ARKit la tecnología para el usuario es igual, pero por dentro no funciona exactamente igual. ARKit no manda los datos a un servidor, si no que utiliza el framework MultipeerConnectivity de Apple para mandar la información del mapa (ARWorldMap) por una conexión cliente a cliente. [36]

Cabe mencionar también que los Cloud Anchors tienen una serie de limitaciones en el almacenamiento y el acceso a los datos. Sólo pueden accederse hasta 24 horas después de haber sido colocados y 7 días después cualquier dato en la nube será borrado. El mapa hospedado en la nube no puede ser descargado por ningún usuario y no se puede determinar un lugar geográfico o reconstruir imágenes basándose en el mismo. Además, los datos que envía un dispositivo para que sean comparados con el mapa guardado no se almacenan nunca.

Para hacer un buen uso de ellos se debe evitar colocar puntos de anclado en superficies brillantes e intentar que la zona tenga una iluminación buena y consistente.

2.4. Aplicaciones

En este apartado se recogen las principales aplicaciones por sectores que existen de la realidad aumentada. Estas son la medicina, educación, arte, seguridad, publicidad, turismo, ecommerce y videojuegos.

2.4.1. Medicina

[18]

2.4.2. Educación

2.4.3. Arte

2.4.4. Fabricación

2.4.5. Publicidad

2.4.6. Turismo

2.4.7. Videojuegos

2.4.8. Comercios electrónicos(ecommerce)

Los probadores virtuales están marcando tendencia en las nuevas generaciones de aplicaciones y estrategias de venta. Estas aplicaciones hacen uso de la realidad aumentada

con tecnologías como el reconocimiento de rostros, reconocimiento de superficies, estimación de luces… Siendo punto de referencia en el mercado los siguientes ejemplos.

a). **Ikea Place**
Esta aplicación permite al usuario ver el catálogo de Ikea y una vez seleccionado el elemento verlo en la habitación real con las dimensiones reales del objeto.

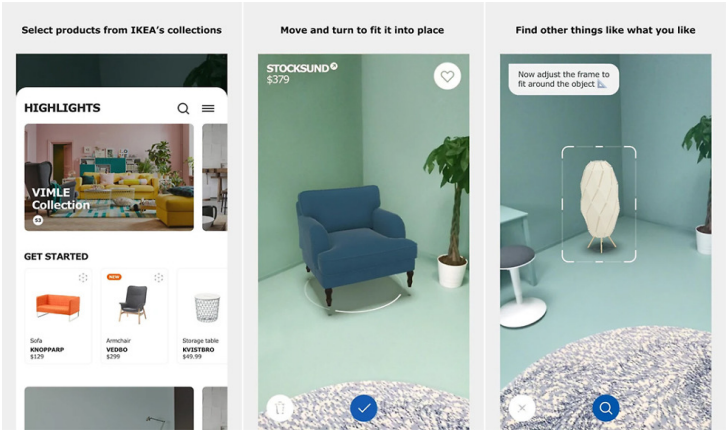


Figura 2.9: IKEA Place AR probador virtual

b). **YouCam Makeup**
Esta aplicación es un excelente y conseguido ejemplo de ecommerce en el mundo de la belleza donde se aplica esta tecnología. La calidad del tracking del pelo es bastante razonable generando una experiencia agradable.

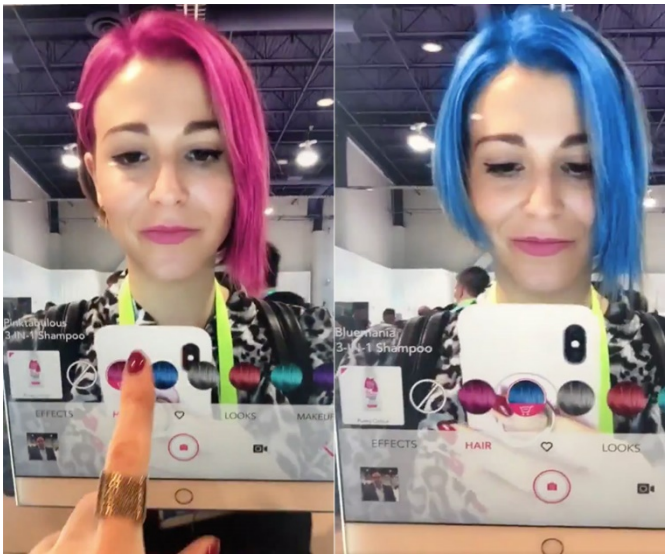


Figura 2.10: Imagen representativa de YouCam Makeup

c). **L'Oreal (Modiface)**
Aplicación que permite al usuario maquillarse con los productos de L'Oreal en realidad aumentada, así como escoger el color que más pegue con sus prendas. Mostrando los productos relacionados a esa tonalidad y ofreciendo la posibilidad de comprarlos dentro de la aplicación.

Figura 2.11: Pantallas de ejemplo de la aplicación L'Oreal Modiface

2.5. Experiencia de usuario en aplicaciones de RA sin marcadores

2.6. Librerías de realidad aumentada sin marcadores (SDK)

En este apartado se describirán las principales tecnologías de realidad aumentada sin marcadores para más tarde estudiar las capacidades y posibilidades particulares de cada una de ellas en el apartado desarrollo. Por cada librería se recogerán los siguientes datos:

- Breve descripción
- Última versión
- Funciones
- Plataformas disponibles
- Tipos de licencia

Luego se compararán todas juntas para ver las funcionalidades que tienen, las plataformas con las que son compatibles y los lenguajes que soportan.

2.6.1. Wikitude

Desarrollada por Wikitude GmbH, es una de las librerías pioneras en el mundo de la realidad aumentada. Lanzaron su primera aplicación en el 2008, desde entonces, son líderes del mercado. La versión que hemos usado ha sido Wikitude SDK 8.7.0 (2019-08-13).[19] Las principales funcionalidades son: Geo AR (Puntos de anclaje vía GPS) Reconocimiento de imágenes 2D (marcadores) Reconocimiento de objetos 3D Las plataformas móviles soportadas son:

- Android
- iOS
- Windows
- Unity
- Cordova
- Xamarin
- Flutter
- Titanium

Soporte para Smart Glasses:

- Epson Moverio
- Hololens
- Vuzix

Otras plataformas:

- React Native
- Ionic
- Adobe Air
- Qt by Felgo
- LBAR

Licencias:

- Wikitude Demo. Licencia de 30 días con marca de agua 499€
- Wikitude SDK PRO (Sólo con marcadores y Geo AR). 1 año de licencia 1990€
- Wikitude SDK PRO 3D (Paquete completo). 1 año de licencia 2490€

2.6.2. ARKit

Desarrollado por Apple, presentado en la Apple Worldwide Developers Conference de 2017. La versión con la que trabajamos es la ARKit SDK 3.0.[12] A diferencia del resto, para usar esta librería en Unity, no hace falta descargar ningún plugin, viene incluido en el paquete de Unity ARFoundation 2.2. Funcionalidades:

- Reconocimiento de imágenes 2D (marcadores)
- Reconocimiento de objetos 3D
- Reconocimiento de rostro (hasta 3 simultáneamente)
- Oclusión
- SLAM
- Estimación de luces
- Puntos de anclaje en la nube

Las plataformas soportadas son:

- iOS
- Unity (via ARFoundation)
- Unreal Engine 4.[17]

La licencia es gratuita.

2.6.3. ARCore

Desarrollado por Google, fue lanzado en febrero de 2018 como respuesta para competir contra ARKit de iOS. La versión con la que trabajamos es ARCore SDK for Unity v1.11.0 (2019-05-05).[21] Funcionalidades:

- Reconocimiento de imágenes 2D (Marcadores)
- Reconocimiento de objetos 3D
- Reconocimiento de rostro.
- SLAM
- Mapeado de áreas grandes
- Estimación de luces
- Puntos de anclaje en la nube

Las plataformas soportadas son:

- Android
- Android NDK
- Unity (Android, iOS)
- Unreal Engine 4
- iOS

La licencia para usar ARCore es completamente gratuita.

2.6.4. Vuforia

Desarrollado por la empresa PTC, un proveedor tecnológico mundial de la plataforma líder de IoT (Internet of Things) y realidad aumentada. La versión que hemos utilizado ha sido Vuforia SDK Android 8.3.8 (2019-06-13).[34] Funcionalidades:

- Reconocimiento de imágenes 2D (Marcadores)
- Reconocimiento de objetos 3D
- Escáner de objetos 3D

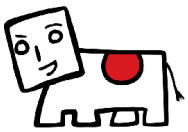
Plataformas:

- Android
- iOS
- Windows
- Smart Glasses

Licencias:

- Básica, 42$ al mes.
- Básica con base de datos en la nube para los marcadores 99$ al mes.
- Para la versión pro, la cual incluye todas las funcionalidades, hay que contactar y hacen presupuesto a medida para la empresa.

2.6.5. Kudan



Kudan es una empresa que se dedica al desarrollo de la realidad aumentada, virtual y mixta, además de la conducción autónoma, drones y robots. La versión que hemos utilizado ha sido la Kudan SDK Unity 1.6.0 (2019-07-16).[35] Funcionalidades:

- Reconocimiento de imágenes 2D (Marcadores)
- SLAM

Plataformas:

- Unity (Android, iOS)
- iOS
- Android

Licencias:

- AR Indie: Gratis. Pensado para la fase de desarrollo, protegido con marca de agua.
- AR Business: 1500$. Para las empresas con menos de un millón de dólares en ingresos.
- AR Enterprise: Para las empresas con más de un millón de dólares en ingresos, hay que contactar con Kudan y proporcionan un presupuesto personalizado.

2.6.6. MaxST

Maxst se fundó en 2010 y se dedica a la investigación y desarrollo de la tecnología de realidad aumentada, han lanzado Maxst AR SDK, el cual hemos probado en la versión MaxstARSDK_Unity 4.1.3. [29] Funcionalidades:

- Reconocimiento de imágenes 2D (Marcadores)
- Reconocimiento de objetos 3D
- Reconocimiento de códigos de barras y QR.

Plataformas:

- Unity (Android,iOS)
- Android
- iOS
- Windows
- macOS
- Epson MOVERIO BT-300,350 y ODG R-7

Licencias:

- Free. Gratis, para uso no comercial, incluye marca de agua.
- Pro-one. Para aplicaciones con menos de 100k descargas (no incluye actualizaciones). Pago único de 499$
- Pro-Subscription. Subscripción anual, incluye actualizaciones 599$ por año
- Enterprise. Para aplicaciones con más de 100k de descargas. Hay que contactar con Maxst para recibir un presupuesto.

2.6.7. 8th Wall

8th Wall desarrolla dos productos diferentes, 8th Wall Web y 8th Wall XR for Unity. El producto que vamos a analizar en este caso es el 8th Wall XR for Unity 11.2.6.519, para que la comparación entre las librerías sea mas precisa, ya que la potencia que tiene en navegador es menor a la que puede llegar a tener una aplicación de Unity. [9] Funcionalidades:

- Reconocimiento de imágenes 2D (Marcadores)
- 6 grados de libertad.
- SLAM
- Estimación de la luz

Plataformas:

- Unity (Android, iOS)
- Web (A-Frame, BabylonJS, Sumerian, three.js)

El uso de 8th Wall XR de Unity es gratuito. En el caso de 8th Wall Web, la licencia se cobra según las visitas en la web. Aparte, se necesita una licencia de desarrollador que cuesta 250$/mes.

Pago por visita (PPV)	Paquete estándar	PPV de alto tráfico	Paquete alto tráfico
1000$/mes	3000$/mes	6000$/mes	6000$/mes
0 visitas incluidas	500k visitas incluidas	0 visitas incluidas	5M visitas incluidas
0.01$/visita	0.01$/visita extra	0.0025$/visita	0.0025$/visita extra

Licencias 8th Wall

2.6.8. Easy AR

EasyAR es una compañía china que lleva en el mercado desde 2016. Hemos probado la versión EasyARSense Unity SDK v3.0.1(2019-07-07)[16] Funcionalidades:

- Reconocimiento de imágenes 2D (Marcadores)
- Reconocimiento de objetos 3D
- SLAM
- Grabación de pantalla

Plataformas soportadas:

- Unity (Android, iOS)
- Android
- iOS
- Windows

Licencias:

- EasyAR SDK Basic. Gratis
- EasyAR SDK Pro. Añade el reconocimiento de objetos 3D, la grabación de pantalla y reconocimiento de más de un marcador simultáneo
- EasyAR SDK Pro trial. Lo mismo que el Pro, pero limitado a 100 usos por día.

2.6.9. ARFoundation

Este último no se trata exactamente de una librería, es un paquete de Unity (aún en fase experimental) que integra una API de alto nivel (wrapper) que permite tener el mismo código funcional para ARCore y ARKit, según si exportamos el proyecto en Android o en iOS. La versión más reciente es ARFoundation 2.2 (Unity 2019.1), la cual incluye ARKit 3.[31]
Soporta las mismas funcionalidades que ARCore y ARKit:

- Reconocimiento de imágenes 2D (marcadores)
- Reconocimiento de objetos 3D.
- Reconocimiento de rostro
- Oclusión (iOS con ARKit)
- SLAM
- Mapeado de áreas grandes (ARCore)
- Estimación de luces
- Puntos de anclaje en la nube

Plataformas soportadas:

- Unity (Android, iOS)

Su licencia, al igual que ARCore y ARKit, es gratuita.

Capítulo 3

Comparación y análisis de las librerías

El análisis de las librerías se estructurará en los puntos que se describen a continuación:

- Calidad de la documentación y primeros pasos: en este punto evaluaremos la dificultad para realizar una aplicación básica con cada librería, desde el momento en el que se descarga el SDK, hasta que se construye la APK.
- Evaluación de las capacidades de la librería: en este apartado se tendrán en cuenta las funcionalidades, las tecnologías que soporta y el nivel personalización dentro de la App, es decir, hasta que nivel podemos usar la API que nos proporcionan.
- Conclusiones: gracias al estudio realizado estableceremos unas conclusiones sobre el uso de cada librería y decidiremos si nos facilita el desarrollo de alguna prueba de concepto.

Para realizar la evaluación de las capacidades y los límites de cada librería se realizará un test que consistirá en:

- Instanciar un objeto.
- Movernos alrededor de dicho objeto, para comprobar la estabilidad del punto de anclaje.
- Realizar movimientos bruscos y veloces para ver si pierde la referencia en algún momento.
- Hacer que pierda la referencia y comprobar el tiempo en el que vuelve a aparecer el objeto.
- Alejarnos del objeto y ver hasta que distancia sigue funcionando.
- Comprobar cómo se comportan las librerías con diferentes intensidades de luz.

Los vídeos se encuentran aquí: https://www.youtube.com/playlist?list=PLqQgTAUiabc8AQrcc48Jdglnytus9IoXe

3.1. Wikitude

3.1.1. Calidad de la documentación y primeros pasos

La experiencia al crear una aplicación usando Wikitude es buena. Te guían desde el momento en el que entras a la web, a descargar el SDK que necesites, eso sí, hace falta registrarse y para que funcione la aplicación hay que descargar una clave de licencia, la cual hay que introducir en el componente Wikitude Camera que trae el paquete. Todos estos pasos vienen documentados, y para facilitar el proceso de prueba, traen varias escenas montadas en las que se pueden probar diferentes tipos de tecnologías. Para hacer funcionar todas las escenas, es necesario que el *Package Name* de la aplicación sea "com.wikitude.unityexample", ya que el SDK está protegido de esta manera. La documentación de la API es muy completa y está muy bien estructurada.[20]

3.1.2. Evaluación de las capacidades de la librería

Condiciones de luz mínimas:
Esta prueba está disponible en el link: https://youtu.be/_4fLQas1NcM

La sala está únicamente iluminada por un haz de luz perteneciente a una habitación situada al otro lado del pasillo.

Consigue instanciar el objeto dentro de un plano y posee buena iluminación. El plano falla cuando empezamos a movernos haciendo un giro de 45º no esperado, si estamos encima del modelo se pierde, si damos la vuelta completa sigue perdido. Necesitamos volver a referenciarlo por que no se recupera. La calidad del modelo y su textura es óptima para las condiciones de luz que posee. Se pierde la referencia fácilmente ante los giros. Cuando se realiza un movimiento de cámara en el que el punto de anclaje sale del *frustrum*[1] y más tarde se vuelve a enfocar a él, tarda un segundo en volver a posicionar el plano y su objeto. En esta primera prueba no hemos conseguido que pierda la referencia por distancia, se repetirá con un escenario más amplio. Hace estimaciones con oclusión desapareciendo con la pared.

Condiciones de luz ambiente:
Esta prueba está disponible en el link: https://youtu.be/wlqITlPTz3o

La sala está únicamente iluminada por dos ventanas a la luz de la tarde.

Consigue instanciar el objeto dentro de un plano y posee buena iluminación. El plano falla cuando empezamos a movernos haciendo de nuevo un giro de 45º no esperado, si estamos encima del modelo se pierde pero en este caso se recupera con cierta facilidad. Podemos acercarnos bastante manteniendo el nivel de detalle. La calidad del modelo y su textura es óptima para las condiciones de luz que posee. Se pierde la referencia fácilmente ante los giros. Cuando se realiza un movimiento de cámara en el que el punto de anclaje sale del *frustrum* y más tarde se vuelve a enfocar a él, tarda menos de un segundo en volver a posicionar el plano y su objeto. En esta primera prueba no hemos conseguido que

[1]Región cerrada del espacio que delimita los objetos que aparecen representados en la pantalla.

pierda la referencia por distancia. Hace estimaciones con oclusión desapareciendo con la pared.

	Luz Ambiente	Luz Mínimas
Estabilidad del punto de anclaje	4	6
Estimación y calidad de iluminación	7	8
Resistencia a movimientos	1	2
Recuperación del ancla	8	8

Cuadro 3.1: Análisis Wikitude

3.1.3. Conclusiones

3.2. ARKit

3.2.1. Calidad de la documentación y primeros pasos

3.2.2. Evaluación de las capacidades de la librería

Condiciones de luz mínimas:

Esta prueba está disponible en el link: https://youtu.be/uwjq-bF_J4M

La sala está únicamente iluminada por un haz de luz perteneciente a una habitación situada al otro lado del pasillo y una pequeña iluminación frontal.
Necesita significativamente más luz(2) para reconocer el plano, en parte debido al dispositivo. La estimación de la iluminación es excelente como podemos ver a lo largo de la prueba. Al realizar un movimiento alrededor del modelo no se pierde ni se desestabiliza en ningún momento. Al posicionar el teléfono sobre el modelo sigue estable. Las texturas del modelo se ven de manera nítida y realista. En la prueba de resistencia a movimientos bruscos se desplaza el plano en ocasiones recuperándose en un breve periodo de tiempo (menor a un segundo). Cuando se realiza un movimiento en el que el modelo desaparece del campo de visión este no llega a desaparecer nunca del entorno virtual por lo que al volver a enfocar al punto de anclaje la transición es limpia.

Condiciones de luz ambiente:

Esta prueba está disponible en el link: https://youtu.be/zcm59hRtJeQ

La sala está únicamente iluminada por dos ventanas a la luz de la tarde.
Reconoce los planos de casi instantáneamente, por lo que nos permite posicionar el objeto sin esperas. La calidad de las texturas son muy buenas y la estimación de luz es impresionante, la luz cambia mientras rodeamos al modelo, estando mas oscuro cuando nos encontramos a contraluz, e iluminado nos situamos entre la luz entrante y el modelo. La estabilidad del objeto es sobresaliente, nos permite rodear el objeto y acercarnos todo lo posible sin que se mueva. La resistencia a movimientos bruscos funciona muy bien, situando al modelo siempre en su punto origen. Lo mismo ocurre en la prueba de sacar el punto de anclaje del campo de visión de la cámara,el dragón nunca desaparece, por lo que al volver a enfocar el punto de partida, vemos una transición muy natural.

	Luz Ambiente	Luz Mínimas
Estabilidad del punto de anclaje	4	6
Estimación y calidad de iluminación	7	8
Resistencia a movimientos	1	2
Recuperación del ancla	8	8

Cuadro 3.2: Análisis ARKit

3.2.3. Conclusiones

Como hemos comprobado anteriormente, en unas condiciones de luz mínimas el trabajo de esta librería ha sido casi excelente. Su único fallo era la resistencia a movimientos bruscos,

la cual se arregla cuando las condiciones lumínicas son óptimas, pudiendo disfrutar de una experiencia perfecta. Si ya con la luz mínima era casi perfecta, con unas condiciones buenas de luz, estamos ante una de las mejores opciones a la hora de crear una experiencia de realidad aumentada. Cabe destacar que ARKit sólo está soportado en dispositivos iOS, por lo que se puede controlar mucho más el abanico de dispositivos en el que se va a usar la aplicación. Lo cual es útil a la hora de asegurar el correcto funcionamiento de las aplicaciones.

3.3. ARCore

3.3.1. Calidad de la documentación y primeros pasos

Hacer funcionar una aplicación con ARCore es tan fácil como descargar el SDK desde su Github oficial [25], añadir una de las escenas de ejemplo a la build, seleccionar dentro de Unity3D la opción de "Player Settings ->XR Settings ->ARCore Supported".Para probar la realidad aumentada con ARCore no hace falta registrarse ni obtener ninguna licencia para que funcione, exceptuando el uso de los *Cloud Anchors*, en el siguiente capítulo explicaremos cómo activarlos. La documentación de los pasos a seguir y de la API es bastante completa, además es de que al ser una de las librerías más usadas en la comunidad, hay muchos tutoriales en el que puedes aprender sobre ella. Para facilitar el desarrollo de aplicaciones con ARCore, Google ha desarrollado una aplicación *ARCore Instant Preview*, la cual se engancha vía USB o Wifi con el editor de Unity3D, y permite probar las aplicaciones en el teléfono sin necesidad de generar una APK.

3.3.2. Evaluación de las capacidades de la librería

Condiciones de luz mínimas:

Esta prueba está disponible en el link: https://youtu.be/uSRztU8z18U
La sala está únicamente iluminada por un haz de luz perteneciente a una habitación situada al otro lado del pasillo y una pequeña iluminación frontal.

Necesita más luz(2) para reconocer el plano. La estimación de la iluminación es excelente como podemos ver a lo largo de la prueba. Al realizar un movimiento alrededor del modelo no se pierde ni se desestabiliza en ningún momento. Al posicionar el teléfono sobre el modelo sigue estable. Las texturas del modelo se ven de manera nítida y realista. En la prueba de resistencia a movimientos bruscos no desaparece nunca ni vibra la imagen dando unos resultados óptimos. Cuando se realiza un movimiento en el que el modelo desaparece del campo de visión este no llega a desaparecer nunca del entorno virtual por lo que al volver a enfocar al punto de anclaje la transición es limpia.

Condiciones de luz ambiente:

Esta prueba está disponible en el link: https://youtu.be/PLXAEFJn4rQ

En este caso reconoce los planos al instante gracias a las condiciones lumínicas. La estimación de la iluminación es sorprendente, ya que mientras giramos alrededor del modelo la luz reacciona a nuestros movimientos. Podemos observar que al posicionarnos en contraluz el objeto está menos iluminado que al comienzo de la prueba donde la luz era directa. En el segundo caso creemos que la intensidad de la luz estimada es un poco excesiva. Los movimientos de cámara se mantienen como en la primera prueba teniendo un resultado perfecto.

	Luz Ambiente	Luz Mínimas
Estabilidad del punto de anclaje	10	10
Estimación y calidad de iluminación	8	9
Resistencia a movimientos	10	10
Recuperación del ancla	10	10

Cuadro 3.3: Análisis ARCore

3.3.3. Conclusiones

Los resultados obtenidos con ARCore han sido magníficos, el único punto flojo es cuando las condiciones de luz son muy bajas, porque no es capaz de detectar la superficie. Quitando esa situación, la experiencia obtenida es muy buena, porque es muy estable y además la estimación de luz funciona casi a la perfección. Como hemos comentado antes, hay situaciones en las que es exagerado el brillo que obtiene. A pesar de sus resultados, también tiene una parte mala, su lista de dispositivos soportados [23]. Aunque con el paso del tiempo esa lista va creciendo y los móviles nuevos cada vez son más potentes, aun queda mucho porcentaje de dispositivos que no están soportados, por lo que el público al que puede llegar una aplicación de realidad aumentada usando ARCore es limitado.

3.4. Vuforia

3.4.1. Calidad de la documentación y primeros pasos

3.4.2. Evaluación de las capacidades de la librería

Condiciones de luz mínimas:

Esta prueba está disponible en el link: https://youtu.be/4Y_Enzrx17w
La sala está únicamente iluminada por un haz de luz perteneciente a una habitación situada al otro lado del pasillo y una iluminación frontal.

El nivel de luz de la sala no supone ningún problema para posicionar el modelo. La calidad de las texturas del objeto son malas, además no existe ninguna estimación de iluminación sobre el modelo. El anclaje alrededor del objeto con movimientos suaves es malo ya que se mueve con nosotros. La estabilidad del objeto cuando nos movemos en sus proximidades es mala ya que el objeto cambia de posición a medida que nos acercamos. No conseguimos perder el objeto con la distancia. La capacidad de soportar movimientos bruscos es buena, ya que no pierde la posición del ancla. Al sacarlo del campo de visión lo mantiene en su posición en todo momento.

Condiciones de luz ambiente:

Esta prueba está disponible en el link: https://youtu.be/zlUj-jh7Uts
Después de mejorar las condiciones de luz se mantienen los problemas de estimación de luz. La visualización del modelo renderiza las texturas de manera muy pobre siendo muy poco realista. La estabilidad del objeto mejora ya que podemos girar alrededor del objeto sin perder la referencia, permitiéndonos acercarnos en esta ocasión. Se mantiene la capacidad de soportar movimientos bruscos y la robustos en cuanto al campo de visión.

	Luz Ambiente	Luz Mínimas
Estabilidad del punto de anclaje	4	6
Estimación y calidad de iluminación	7	8
Resistencia a movimientos	1	2
Recuperación del ancla	8	8

Cuadro 3.4: Análisis Vuforia

3.4.3. Conclusiones

3.5. Kudan

3.5.1. Calidad de la documentación y primeros pasos

Los primeros pasos con Kudan no son tan gratificantes en comparación a sus competidores. Empezando por la descarga del SDK, ésta ni si quiera se encuentra en la página oficial [28], si no que se encuentra en XLSoft [35]. Una vez metido el SDK en Unity, en las carpetas vienen escenas de ejemplo, en las que podemos ver como se monta una aplicación con Kudan. La librería está protegida por *Package Name*, contactando con los desarrolladores de Kudan puedes obtener una clave de licencia para cualquier *Package Name*, sin embargo, proporcionan una general con la que podemos desarrollar "com.xlsoft.kudanar". [27]La documentación para los primeros pasos es aceptable, pero sobre el SDK y la API no hay documentación, únicamente los comentarios en los scripts. Una vez instalada la aplicación en el teléfono, seguramente no funcione ya que la aplicación no pide los permisos para usar la cámara, y sin ellos Kudan no puede inicializarse. Para arreglar este error, hace falta añadir en algún script esta línea de código "Permision.RequestUserPermission(Permission.Camera)", que se encuentra en el paquete ÜnityEngine.Android".

3.5.2. Evaluación de las capacidades de la librería

Condiciones de luz mínimas:

Esta prueba está disponible en el link:https://youtu.be/cLJMyV9bV6s
La sala está únicamente iluminada por un haz de luz perteneciente a una habitación situada al otro lado del pasillo.

El nivel de luz de la sala no supone ningún problema para posicionar el modelo. La calidad de las texturas del objeto son malas, además no existe ninguna estimación de iluminación sobre el modelo. El anclaje alrededor del objeto con movimientos suaves es aceptable pero en el momento que nos acercamos se pierde y hay que volver a referenciarlo. La estabilidad del objeto cuando nos movemos en sus proximidades es muy mala, cambiando de tamaño sin sentido aparente. La distancia máxima de captura es de siete metros aproximadamente. La capacidad de soportar movimientos bruscos es mala, pierde totalmente la posición del ancla con resultados incorrectos e incluso a veces pierde la referencia del todo. Al sacarlo del campo de visión no lo posiciona en el mismo punto donde estaba, llegando a perder en ocasiones el punto de referencia.

Condiciones de luz ambiente:

Esta prueba está disponible en el link:https://youtu.be/JyjBmQZZ5E4
Gracias a las condiciones de luz la estabilidad del objeto en la escena mejora sustancialmente, pudiendo dar la vuelta casi perfecta al objeto sin problemas, exceptuando la posición cenital que genera desestabilización breve en el objeto. Se mantiene la distancia máxima de siete metros. Además en esta ocasión la resistencia ante movimientos bruscos mejora considerablemente, sin llegar a ser correcta ya que pierde la referencia en una ocasión. En el caso del campo de visión recupera el objeto de manera más óptima sin llegar a volver a posicionar con precisión el objeto.

	Luz Ambiente	Luz Mínimas
Estabilidad del punto de anclaje	3	6
Estimación y calidad de iluminación	0	0
Resistencia a movimientos	2	7
Recuperación del ancla	3	6

Cuadro 3.5: Análisis Kudan

3.5.3. Conclusiones

Los resultados obtenidos con Kudan no son muy buenos, además de que no renderiza la cámara en la aplicación, por lo que se ve con un fondo negro, impidiendo que la experiencia sea tan inmersiva frente a las demás librerías. Los resultados con condiciones de luz mínimas han sido muy malos, y aunque cuando hemos aumentado la cantidad de luz ha mejorado bastante, no llega al nivel de los competidores.

3.6. Maxst

3.6.1. Calidad de la documentación y primeros pasos

Para usar Maxst, hace falta registrarse en su página web [29], y acceder a la descarga del SDK. Además, hay que generar una clave de licencia específica para nuestro *Package Name*.La documentación es buena en ambos casos, en la guía de integración y de la API. Una vez importado la librería en Unity, se puede ver que hay escenas de ejemplo ya hechas, por lo que simplemente hay que añadirlas a la *Build* de la aplicación. Se pueden probar las aplicaciones también desde el editor, usando una cámara que esté conectada al ordenador, obviamente funciona peor que en un móvil, ya que la cámara no tiene sensores, pero sirve para darnos una idea del tamaño de los objetos.

3.6.2. Evaluación de las capacidades de la librería

Condiciones de luz mínimas:

Esta prueba está disponible en el link:https://youtu.be/QFX_B8HX1yU
La sala está únicamente iluminada por un haz de luz perteneciente a una habitación situada al otro lado del pasillo.

El nivel de luz de la sala no supone ningún problema para posicionar el modelo. La calidad de las texturas del objeto son notables, no existe ninguna estimación de iluminación sobre el modelo. El anclaje alrededor del objeto con movimientos suaves es aceptable pero en el momento que nos acercamos se pierde y no se llega a recuperar el punto teniendo que volver a referenciarlo. No somos capaces de dar la vuelta al modelo completo sin perderlo. No conseguimos perderlo con la distancia. La capacidad de soportar movimientos bruscos es mejorable ya que no desaparece el modelo, pero si pierde su referencia en el espacio moviéndolo a una posición diferente. Al sacarlo del campo de visión no lo posiciona de nuevo en la mayoría de las ocasiones. Sin embargo, cuando consigue mantenerlo, en la mayoría de ocasiones se desplaza del punto correcto y muy rara vez muestra la opción correcta.

Condiciones de luz ambiente:

Esta prueba está disponible en el link:https://youtu.be/cspBAaQCfew
La estabilidad a mejorado notablemente, permitiendo dar la vuelta completa. En ocasiones pierde la referencia y se desplaza el modelo un poco si nos acercamos demasiado. Si pierdes el punto de visión por ejemplo cruzando una pared desaparece el modelo, pero si volvemos al punto de partida es capaz de recuperar la referencia en aproximadamente 7 segundos. La resistencia a los giros bruscos ha mejorado muy considerablemente llegando a ser casi perfecta, a veces se desplaza en algún fotograma pero se reposiciona rápidamente. Al sacar el modelo del campo de visión de la cámara y luego volver a meterlo rinde mejor con el entorno iluminado, consiguiendo que esté bien posicionado, eso sí, a veces sigue desapareciendo y volviendo a aparecer, impidiendo así una transición limpia.

	Luz Ambiente	Luz Mínimas
Estabilidad del punto de anclaje	4	6
Estimación y calidad de iluminación	1	1
Resistencia a movimientos	4	9
Recuperación del ancla	3	7

Cuadro 3.6: Análisis Maxst

3.6.3. Conclusiones

Apenas necesita tiempo para posicionar el ancla, no hay que esperar a que reconozca una superficie plana, pero la estabilidad debe mejorar un poco. A pesar de que al estar en unas condiciones lumínicas ideales, el modelo a veces se mueve cuando estamos muy cerca e incluso puede llegar a perderse la referencia. Esto es crítico si se quiere desarrollar una aplicación en la que se pueda mover alrededor de un punto, o acercarse mucho a él para poder verlo con detalle.

3.7. 8th Wall XR

3.7.1. Calidad de la documentación y primeros pasos

Para desarrollar aplicaciones de realidad aumentada con 8thWall, es necesario registrarse en su página web [9], descargar el SDK, y generar una clave de licencia, que tendremos que especificar en el objeto "XRAppSettings" que se encuentra dentro del paquete de Unity3D. Estos pasos vienen bien documentados en la guía que proporcionan, por lo que la calidad de la documentación es buena, también en la parte de la API. Tienen tutoriales subidos en los que enseñan y explican cómo desarollar aplicaciones con su SDK. En los dispositivos que se encuentran en la lista de ARCore [23], 8thWall usa ARCore para la tecnología de realidad aumentada. Para los que no se encuentran en esa lista, tienen su propia tecnología, que no funciona tan bien,pero por lo menos permite cubrir un rango muy amplio de dispositivos soportados. Para facilitar el desarrollo, han creado una aplicación para probar en el móvil desde el editor de Unity3D, llamada *8th Wall XR Remote*, que se puede encontrar en *Google Play* [11].

3.7.2. Evaluación de las capacidades de la librería

Condiciones de luz mínimas:

Esta prueba está disponible en el link:https://youtu.be/6edM5PhhXj0
La sala está únicamente iluminada por un haz de luz perteneciente a una habitación situada al otro lado del pasillo.

El nivel de luz de la sala supone un problema para posicionar el modelo con lo que aumentamos el nivel de luz. La calidad de las texturas del objeto son buenas, además posee estimación de iluminación sobre el modelo correcta y muy eficaz. El anclaje alrededor del objeto con movimientos suaves es muy buena pudiendo dar una vuelta sin problema y acercarnos para ver el detalle del dragón sin que desaparezca. No conseguimos perderlo con la distancia en ningún momento. La capacidad de soportar movimientos bruscos es muy resistente, nunca se pierde y su anclaje sigue en la posición correcta. Al sacarlo del campo de visión lo mantiene creando una transición suave.

Condiciones de luz ambiente:

Esta prueba está disponible en el link:https://youtu.be/5TBd3ml35RY

La detección del plano es muy rápida. Mantiene y mejora la calidad de las texturas del objeto, así como la estimación de iluminación. El anclaje alrededor del objeto con movimientos suaves es muy buena pudiendo dar una vuelta sin problema y acercarnos para ver el detalle del dragón sin que desaparezca. No conseguimos perderlo con la distancia en ningún momento. La capacidad de soportar movimientos bruscos y movimientos fuera del campo de visión continua siendo muy resistente.

	Luz Ambiente	Luz Mínimas
Estabilidad del punto de anclaje	10	10
Estimación y calidad de iluminación	8	8
Resistencia a movimientos	10	10
Recuperación del ancla	10	10

Cuadro 3.7: Análisis 8th Wall

3.7.3. Conclusiones

Aunque por debajo utilice la misma tecnología que ARCore, la capa que han desarrollado por encima simplifica bastante el uso de la librería y el desarrollo de una aplicación con realidad aumentada. Por esta razón, y por los resultados obtenidos, que han sido excelentes, esta librería una muy buena opción para desarrollar una aplicación. Además, el otro producto que desarrolla la empresa, 8th Wall for Web, permite llevar la realidad aumentada a un navegador de manera muy sencilla, sin necesidad de instalar una aplicación completa. Aunque los resultados no son tan buenos, se defiende muy bien [10].

3.8. Easy AR

3.8.1. Calidad de la documentación y primeros pasos

Es necesario registrarse para acceder a la descarga del SDK y generar una clave de licencia. Una vez importado el paquete dentro de Unity, hay que buscar el objeto "EasyARKey" y especificar la clave. Las escenas que vienen de ejemplo no son del todo intuitivas ni modificables, ya que para cambiar el objeto que se instancia hay que entrar y hacer unos cambios en el código, lo ideal sería poder cambiarlo desde el editor. La documentación es mejorable, solo tienen una guía para hacer la *Build* en cada plataforma, pero no hay ninguna guía que explique como crear una aplicación desde cero, ni explican cuales son las clases y componentes importantes e inprescindibles.

3.8.2. Evaluación de las capacidades de la librería

Condiciones de luz mínimas:

Esta prueba está disponible en el link:https://youtu.be/G_iY6gdoMOU

La sala está únicamente iluminada por un haz de luz perteneciente a una habitación situada al otro lado del pasillo.

Coloca el modelo enfrente del usuario automáticamente sin buscar ningún plano de referencia. La estabilidad del objeto es muy mala ya que no nos podemos acercar porque el objeto también se mueve con el usuario, lo mismo ocurre al intentar dar la vuelta sobre él. La calidad de texturas es aceptable y no posee estimación de luz. La resistencia a movimientos bruscos es buena, nunca se deja de ver el objeto y se mantiene en su posición; lo mismo ocurre al sacarlo del campo de visión.

Condiciones de luz ambiente:

Esta prueba está disponible en el link:https://youtu.be/rA2uLYET5ck

Continúan apareciendo los problemas sin podernos acercar al modelo ni dar la vuelta correctamente. Además aparece un nuevo problema dónde la aplicación hace uso de muchos recursos generando un *framerate*[2] muy bajo, por debajo de los 30 fotogramas por segundo. Mejora su capacidad de resistir movimientos bruscos. EL resultado de la prueba de sacar el objeto fuera del campo de visión también es bueno, es capaz de mantener la referencia de la posición.

	Luz Ambiente	Luz Mínimas
Estabilidad del punto de anclaje	2	3
Estimación y calidad de iluminación	1	1
Resistencia a movimientos	10	10
Recuperación del ancla	10	10

Cuadro 3.8: Análisis EasyAR

[2]Frecuencia a la cual un dispositivo muestra los fotogramas

3.8.3. Conclusiones

Aunque la estabilidad del punto de anclaje no sea bueno, la librería instancia el objeto sin buscar un plano, de hecho no utiliza la cámara, únicamente hace uso del giroscopio y del compás digital. El uso que se le podría dar a esta librería es para una aplicación que esté pensado para usarse sin moverse del sitio, da igual las condiciones de luz, la resistencia a los movimientos bruscos es muy buena, al igual que la reposición del objeto cuando se sale del campo de visión de la cámara y luego vuelve.

3.9. ARFoundation

3.9.1. Calidad de la documentación y primeros pasos

Usar ARFoundation es muy cómodo, ya viene integrado en Unity, simplemente hay que añadirlo desde el *Package Manager*. La documentación es buena y además hay varios tutoriales subidos por la comunidad, por lo que es fácil aprender a usar ARFoundation. Unity proporciona una serie de escenas de ejemplo que son muy útiles a la hora de entender el funcionamiento de la API y para realizar pruebas de concepto rápidamente[33].

3.9.2. Evaluación de las capacidades de la librería

Condiciones de luz mínimas:

Esta prueba está disponible en el link:https://youtu.be/MrYmYtdJKrY

La sala está únicamente iluminada por un haz de luz perteneciente a una habitación situada al otro lado del pasillo.

El nivel de luz de la sala no supone ningún problema para posicionar el modelo y reconocer el plano. La calidad de las texturas del objeto son notables, además existe estimación de iluminación sobre el modelo pero no es del todo correcta. El anclaje alrededor del objeto con movimientos suaves es buena nos podemos acercar y alejar manteniendo el nivel de detalle. No conseguimos perderlo con la distancia. La capacidad de soportar movimientos bruscos es casi perfecta ya que no desaparece el modelo, pero a veces se mueve un poco recolocándose en un breve período de tiempo. Al sacarlo del campo de visión lo mantiene en su posición.

Condiciones de luz ambiente:

Esta prueba está disponible en el link:https://youtu.be/y3jS70BuPck

La calidad de la iluminación y de la textura es bastante buena, por lo que da una impresión de realismo. La estabilidad del punto de anclaje es perfecta, podemos dar la vuelta completamente, acercarnos para ver el más mínimo detalle sin que el modelo se desplace ni se deje de ver. La resistencia a los movimientos bruscos es perfecta, no desaparece nunca el modelo ni se mueve de su posición inicial, ocurre igual con la prueba de sacarlo del *frustum*, el dragón permanece en su posición y permite una transición limpia y natural cuando se vuelve a enfocar a su punto de anclaje.

	Luz Ambiente	Luz Mínimas
Estabilidad del punto de anclaje	10	10
Estimación y calidad de iluminación	8	9
Resistencia a movimientos	9	10
Recuperación del ancla	10	10

Cuadro 3.9: Análisis ARFoundation

3.9.3. Conclusiones

Sin duda es una de las opciones más óptimas para desarrollar una aplicación de realidad aumentada de calidad. ARFoundation engloba ARCore y ARKit en una misma API, lo que permite tener una aplicación en ambas plataformas usando el mismo código. El único punto flojo es el rango de dispositivos que soportan la tecnología, que son prácticamente los mismos que ARCore [23]. Quizás hoy en día el público al que puede llegar la aplicación es demasiado breve, pero con el paso del tiempo ira creciendo drásticamente, gracias a la mejora del hardware de los dispositivos.

3.10. ARToolKit

Información de la librería, desarrollo de HolaMundo y conclusiones En nuestros primeros pasos en el mundo de la realidad aumentada exploramos algunas librerías como ARToolKit, con el fin de familiarizarnos con el desarrollo de este tipo de aplicaciones.

ARToolKit es una de las librerías de desarrollo pioneras en el ámbito que investigamos, disponible desde el año 2004 para descargar de manera gratuita y que cuenta con más de 160.000 descargas desde entonces. Se distribuyó para diversas plataformas como SGI IRIX (que dejó de utilizarse en 2006), Linux, MacOS y Windows y fue desarrollada originalmente por el Dr. Hirokazu Kato para posteriormente pasar a manos del Human Interface Technology Laboratory en la Universidad de Washington, la de Nueva Zelanda y ARToolworks.Inc en Seattle.

Muchas librerías posteriores se han basado en el código de ésta para ampliar sus funcionalidades, dando lugar a algunas como ARTag (que promete mayor fiabilidad a la hora de procesar imágenes por su mejor manejo de la luz), FLARToolKit (consistente en un port en ActionScript 3), ARDesktop (que facilita la creación de interfaces) o Studierstube Tracker (que mejora sus características, pero deja de ser de código abierto). Además de todas las derivaciones de ARToolKit, también podemos encontrar software no orientado a programadores como ATOMIC Authoring Tool, que permitía a cualquier usuario el desarrollo de una aplicación de realidad aumentada de manera sencilla y con una interfaz intuitiva. Esta herramienta acabó cayendo en desuso a principios de la década de 2010 debido a que ya existían librerías mejores que ARToolKit y mejores alternativas en lo que a SDK se refiere.

Al ser ARToolKit una de las primeras herramientas para el desarrollo de realidad aumentada, no contemplaba un uso de esta sin marcadores. Una de las mayores dificultades a las que se enfrentó fue el seguimiento del "ojo" del usuario, es decir, el foco de la cámara del dispositivo. Para saber desde qué perspectiva debía dibujar los elementos virtuales la aplicación necesitaba saber a dónde está mirando el usuario en el mundo real. La librería solventa este problema utilizando algoritmos de visión que calculan la localización y orientación de la cámara basándose en marcadores físicos en tiempo real. Los marcadores que es capaz de identificar consisten en la mayoría de los casos en un cuadrado negro bien contrastado con un fondo e interior blancos. Además, cada marcador, para diferenciarse del resto incluye pequeñas variaciones como otras figuras geométricas dentro del cuadrado.

Para nuestros experimentos con ARToolKit, en lugar de utilizar la librería original, utilizamos un port de la misma para ser utilizada en Unity, que puede encontrarse actualmente en https://github.com/artoolkit/arunity5. Esta extensión nos permite el acceso a componentes como ARController y ARMarker dentro del editor.

Para el desarrollo de este "HolaMundo" con ARToolKit en Unity hemos seguido los siguientes pasos: creamos un gameObject que servirá como "raíz" de la escena y otro que actuará como mánager del sistema de realidad aumentada. Al mánager le incluimos el componente ARController, que está encargado de las opciones de video y del seguimiento de los marcadores. Dentro de éste modificamos la Layer a la que debe prestar atención.

Por otra parte, el objeto raíz de la escena incluye la luz direccional y la cámara, y además le añadimos el script AROrigin, que permite situar espacialmente la escena. La cámara, además de su script de cámara recibe un ARCamera para poder detectar los marcadores.

Ahora creamos un objeto que llevará la información del marcador y le añadimos el componente ARMarker, que lleva el tag del marcador que hace las veces de identificador único. Este componente tiene dos tipos de patrones para identificar por defecto: *hiro* y *kanji*. En este caso utilizaremos el patrón *hiro*, que es el que consiste en un cuadrado negro simple.

Añadimos a la raíz de la escena un objeto que será contenedor del objeto 3D que queremos que aparezca cuando enfocamos al marcador y que lleva el script *ARTrackedObject* y dentro del campo "Marker Tag" introducimos el identificador del marcador asociado al objeto.

Conclusiones: si bien este sistema fue útil en su día para sentar las bases del desarrollo de programas en realidad aumentada, hoy en día no tiene mucho sentido su uso. No se encuentra documentación actualizada para su uso y la página web que le daba soporte ha desaparecido [13]. Además, sus funcionalidades son muy limitadas y su rendimiento es muy inferior al que presentan otras alternativas más actuales como Vuforia, que permite también el uso de marcadores.

3.11. Evaluación

3.11.1. Tabla de funcionalidades

A continuación, hay una tabla en el que se puede comparar rápidamente las funcionalidades que tiene cada una de las librerías que vamos a analizar. Más adelante, en la parte de comparación y análisis de las librerías, evaluaremos la calidad y eficiencia de estas funcionalidades para cada librería.

SDK	Reconocimiento 2D	Reconocimiento 3D	Detección de planos	SLAM	Reconocimiento de rostro	Estimación de luces	Otras
Wikitude	✓	✓	✓	✓	-	✓	Geo AR
ARKit	✓	✓	✓	✓	✓	✓	Oclusión, Cloud Anchor
ARCore	✓	✓	✓	✓	✓	✓	Cloud Anchor
Vuforia	✓	✓	✓	✓	-	✓	
Kudan	✓	-	✓	✓	-	✓	
MaxST	✓	✓	✓	✓	–	✓	
8th Wall XR	✓	–	✓	✓	–	✓	
EasyAR	✓	✓	✓	✓	–	✓	Grabación de pantalla
AR Foundation	✓	✓	✓	✓	✓	✓	

Cuadro 3.10: Comparación de funcionalidades

Como vemos en la tabla, menos en el reconocimiento de rostro, casi todas las librerías tienen las mismas funcionalidades, por lo que a priori nos sirve cualquiera para realizar nuestras pruebas de concepto, pero antes toca probarlas y ver cuál es la librería que mejor implementadas y pulidas tiene estas funcionalidades.

3.11.2. Usabilidad

Este apartado consistirá en dos apartados, en la primera compararemos en que plataformas se pueden ejecutar las librerías, y luego comparemos en que plataforma y lenguajes se pueden programar las aplicaciones.

SDK	Unity3D (Android, iOS)	Unreal Engine 4	Java	Objective-C	C++	JavaScript
Wikitude	✓	–	✓	✓	–	✓
ARKit	✓	✓	–	✓	–	–
ARCore	✓	✓	✓	✓	–	–
Vuforia	✓	–	✓	✓	✓	–
Kudan	✓	–	✓	✓	–	–
MaxST	✓	–	✓	–	✓	–
8th Wall	✓	–	–	–	–	✓
EasyAR	✓	–	✓	✓	✓	–
AR Foundation	✓	–	–	–	–	–

Cuadro 3.11: Comparación de plataformas y lenguajes soportados

Podemos ver que prácticamente todas las librerías soportan Android e iOS, con lo que podemos deducir que es donde más se está invirtiendo en el mercado. Quizás las Smart Glasses puedan brindar una experiencia de realidad aumentada más agradable, pero

todavía está muy lejos de ser accesible para la mayoría de la población, mientras que un dispositivo móvil es mucho más asequible.
Con esta tabla, es muy fácil ver que Unity3D sale ganador, podemos trabajar con absolutamente todas las librerías, además, gracias a su editor y entorno visual, resulta muchísimo más cómodo y ahorra mucho tiempo a la hora de hacer una aplicación de realidad aumentada.

3.12. Conclusiones prueba de concepto

	Wikitude	ARKit	ARcore	Vuforia	MaxST	EasyAR	Kudan	8th Wall XR	ARFoundation
Calidad de la documentación									
Estabilidad del punto de anclaje									
Comportamiento con luz intensa									
Comportamiento con luz ambiente									
Comportamiento con luz tenue									
Estimación de luces y calidad de									
Total (puntuación)									

Cuadro 3.12: Análisis de las características de las librerías de RA sin marcadores

Capítulo 4

Pruebas de concepto

4.1. Propuestas

Una vez se han testeado y analizado las diferentes tecnologías surgen diferentes ideas como prueba de concepto en función de las características particulares de cada tecnología. Las cuales son enumeradas a continuación:

- Desarrollo de un *plugin* capaz de gestionar los *cloud anchor* en interiores y almacenar la información durante un período de tiempo ajustable a las necesidades del producto. Será capaz de identificar estas posiciones *online* y *offline* mediante una base de datos almacenada tanto en servidor como en local si fuese necesario. Plataformas deseables: Android (prioritario) y iOS.
- API de alto nivel que permite desarrollar de manera más versátil y sencilla aplicaciones en realidad aumentada teniendo compatibilidad plena entre ARKit y ARCore. El usuario será capaz de desarrollar una aplicación en realidad aumentada completa sin apenas líneas de código ya sea mediante *blueprints* o módulos.
- Juego multijugador con *Cloud Anchors*. Juego en el que dos jugadores o más compiten por conseguir más puntos por destruir edificios. (Inspirado en Bombardero - Amstrad CPC)
- Plataforma de realidad aumentada: permitirá al usuario acceder en el momento a diferentes contenidos ya sean vídeos, juegos o experiencias sin necesidad de salir de la aplicación. Esta idea estaba pensada para el uso con unas gafas de AR y mando inalámbrico para el control del dispositivo. Inspirado en Google DayDream.
- Reconocimiento de objetos.
- Proceso de montaje de muebles paso a paso del en realidad aumentada.

4.2. Juego multijugador con cloud anchor (ARCore) - BombARdero+

La tecnlogía que mas nos ha impresionado ha sido los Cloud Anchors, nunca habíamos oído hablar de ella. Cuando nos enteramos de su existencia, estábamos todos de acuerdos en desarrollar un videojuego multijugador online.

Los Cloud Anchors, como hemos explicado anteriormente, se hostean en la nube de Google. Para usarlos, hace falta conseguir una clave de licencia de *ARCore Cloud Anchor API* en Google Cloud Platform [24]. Una vez conseguida la credencial, hay que introducirla en Unity3D, "Edit ->Project Settings ->Google ARCore ->Cloud Anchor API Keys ". En los ejemplos que ofrece Google, la implementación de los Cloud Anchor utiliza la *Multiplayer High Level API(Multiplayer HLAPI)* de Unity, por lo que también hay que activar el servicio de multijugador en nuestro proyecto desde el dashboard de Unity [32]. Una vez realizado estos dos pasos, al compilar la escena de ejemplo, debería de funcionar. En esa escena, un usuario tiene que crear una sala e instanciar un objeto, el punto de anclaje principal, que se convertirá en las coordenadas (0,0,0) del entorno. Este punto de anclaje tarda unos diez segundos en subirse a la nube y estar disponible para más usuarios, que al entrar a la aplicación le aparece un listado con las salas disponibles.
Para desarrollar este juego, primero desarrollamos dos aplicaciones anteriormente. Como al princpio no teniamos dos móviles que soportasen ARCore, programamos el juego en monojugador, para acostumbrarnos al entorno y para ahorrarnos trabajo en un futuro.
En paralelo, como teníamos que implementar la parte de multijugador online y no teníamos ningún tipo de experiencia, realizamos unas pruebas para estudiar el funcionamiento de la Multiplayer HLAPI. Esas pruebas nos permitieron entender y aprender a desarrollar el multijugador de la prueba de concepto que nos hemos puesto como objetivo.
Una vez que conseguimos tener dos dispositvos que soportasen ARCore, empezamos a desarrollador la aplicación multijugador. Gracias a haber desarrollado el *BombARdero*, nos permitió avanzar muy rápido ya que estaban los modelos, *prefabs*[1], y scripts con las mecánicas básicas ya creadas. Primero tuvimos que mirar como estaba montada la implementación de Google, para entender el proceso de cómo se subía el marcador a la nube y la cantidad de información de control que teníamos. Este proceso fue bastante sencillo, ya que el código de ejemplo es muy entendible e intuitivo. Una vez conseguido implementar el BombARdero con los Cloud Anchor, hubo que hacer unos pequeños retoques a los scripts e añadir componentes a los objetos, para implementar la parte multijugador.
Continuará...

4.3. Instrucciones de montaje de muebles en AR - AmueblAR

En nuestra investigación hemos visto que una de las utilidades más populares actualmente en el campo de la realidad aumentada son las aplicaciones de decoración de interiores. Ikea, por ejemplo, ha lanzado Ikea Place, una aplicación de realidad aumentada para dispositivos móviles que permite visualizar cualquier habitación de tu casa con muebles virtuales. De esta manera podemos decidir si encajan con nuestro salón y si las medidas son las adecuadas para después comprar el mueble en la tienda física.
Este programa despertó nuestra curiosidad y pensamos que podríamos desarrollar una aplicación con una utilidad parecida, pero en este caso aplicada al montaje de los muebles. Sabemos que comprar sofás o estanterías por piezas es en ocasiones más barato y facilita enormemente el transporte, sin embargo la parte en la que muchos compradores experimentan problemas es a la hora de montarlos. Muchos son incapaces de ver en el

[1] Objetos prefabricados de Unity3d

plano de las instrucciones en qué lugar va cada tornillo o de qué forma se unen las patas. Por eso, la aplicación que nosotros hemos propuesto consiste en un pequeño manual de instrucciones en realidad aumentada sin marcadores. De esta manera se puede ver en todo momento el modelo 3D del mueble, rotarlo y además desplazarte alrededor de él para apreciar en detalle cada uno de los pasos en el proceso de montaje.

El funcionamiento es muy sencillo: para empezar escaneamos el plano para que el programa cree una superficie donde colocar el mueble. A continuación tocamos la pantalla en el lugar en el que queremos colocarlo. Ahora podemos rotar el mueble hacia izquierda y derecha para dejarlo en una posición en la que nos resulte cómodo acercarnos y alejarnos mientras vemos los pasos de montaje. Finalmente, al pulsar el botón de aceptar entramos en la fase de construcción. En ella tenemos un botón a cada lado de la pantalla: el de la izquierda retrocede al paso anterior y el de la derecha avanza al paso siguiente. Mientras se suceden las animaciones que van explicando cómo armar progresivamente el mueble el usuario puede moverse libremente con su dispositivo para apreciar los detalles e incluso retroceder si no ha entendido bien el paso a seguir.

Para el desarrollo hemos utilizado la librería ARCore, que permite el despliegue de realidad aumentada sin marcadores. Utilizando la tecnología que nos vienen dada por la librería buscamos una serie de puntos en el suelo que nos permitirán crear un plano virtual. Una vez generada esta superficie el plano se representa gráficamente en pantalla con la unión de los puntos. A continuación dividimos la aplicación en tres estados: despliegue, rotación y montaje.

Empezamos en la fase de despliegue, en la que el programa está continuamente esperando la interacción del usuario. Cuando se toca la pantalla, se comprueba si ha habido colisión con el plano virtual, y de ser así se instancia el mueble en el lugar en el que hemos tocado. Si este proceso se ha llevado a cabo con éxito, pasamos a la fase de rotación. Aquí aparecen dos botones azules a los lados de la pantalla y uno verde en la parte inferior. Los botones laterales llevan una referencia al mueble para que al ser pulsados lo roten en una u otra dirección. El botón inferior por otra parte nos sirve para comunicarle a la aplicación que estamos conformes con la posición actual del objeto y que puede pasar a la siguiente fase.

Finalmente nos encontramos con dos nuevos botones a los laterales, que como hemos explicado anteriormente se encargan de adelantar y retroceder los pasos de montaje. El sofá tiene un componente Animator Controller que lleva el árbol de estados. Este árbol consta de 10 estados fijos, en los que se queda el mueble una vez ha terminado su animación correspondiente y 18 estados móviles o animaciones, que se reproducen cuando pulsas el botón de avanzar o retroceder. Para ello, existe una variable que controla el avance y otra que hace lo propio con el retroceso, que se anulan una vez se entra en un estado estático y se activan al pulsar el botón que corresponde, pasando así a la animación del paso siguiente o anterior.

Cabe destacar el proceso de creación del mueble en sí mismo, que en el caso del ejemplo se trata del sofá Kivik (FOTO). Como no era posible encontrar un modelo dividido en piezas y fácilmente animable tuvimos que modelarlo desde cero con la ayuda de la documentación que encontramos en la web de Ikea. El mueble consta de las siguientes piezas: el asiento del sofá, el respaldo del sofá, el asiento de la *cheslong* [2] y su respaldo, los dos brazos, dos escuadras, una pletina, ocho patas, dieciocho tornillos con sus correspondientes tuercas y arandelas, cinco cojines cuadrados y un cojín alargado para la cheslong. Salvando los

[2] Tipo de sofá que posee una prolongación lo suficientemente larga en forma de L como para soportar las piernas humanas.

tornillos, de los que encontramos en una página de recursos gratuitos [15], todos los demás fueron creados por nosotros en Blender, herramienta de la que ya hemos hablado anteriormente. Una vez terminadas las piezas del sofá, las texturizamos con colores azules y aspecto de sofá, teniendo las piezas como los tornillos o las escuadras una textura metálica.
Pensamos que podríamos empezar a animar cada parte por separado dentro de un objeto conjunto del que fueran hijas todas las piezas, pero para que Unity pueda interpretar cada acción como propia de un objeto mayor a las partes es necesario hacer un *rigging*[3] que abarque todas piezas del sofá. Una vez construido el esqueleto hay que asignarle a cada hueso el objeto que se va a encargar de mover, lo cual dio también algunos problemas, ya que al tratarse de un objeto conjunto ciertos huesos acababan emparentándose automáticamente con partes del sofá que nos les correspondían, de modo que la única solución fue establecer manualmente los pesos (es decir, la influencia que tiene cada hueso sobre una parte concreta) sobre los diferentes conjuntos de vértices hasta que conseguimos el resultado que esperábamos. En lo referente a las animaciones nos fue muy útil el vídeo que tiene publicado la cuenta de Ikea España en la plataforma Youtube [26] en el que se explica el procedimiento que se sigue para construir el sofá. Habiéndolo analizado dividimos el montaje en 9 pasos diferenciados y los animamos también en Blender, haciendo desaparecer las partes que no se están utilizando en ese momento para que sea más claro y fácil verlo. Al contrario que en los ejemplos con marcadores en los que pudimos exportar directamente el archivo de Blender para su uso en Unity, aquí nos dio problemas ya que el editor dejó de reconocer los clips de vídeo de algunas animaciones. Solventamos este problema exportando el objeto desde el programa de modelado en formato fbx, pero nos surgió otro nuevo inconveniente: todas las partes del sofá habían perdido sus texturas. Ya desde Unity creamos los materiales por separado y se los asignamos a cada pieza. Fue un proceso bastante agotador, sobre todo en la parte artística, pero consideramos que el resultado es satisfactorio y útil, además ofrece nuevas posibilidades en un campo que no deja de crecer.

4.4. Visualizador de objetos en AR con gafas Aryzon (ARCore)

En la fase de investigación, pensamos que una de las limitaciones de la realidad aumentada en dispositivos móviles es el hecho de tener que que estar sujetando el móvil con las manos, por lo que el control y la experiencia de usuario se ve condicionada negativamente. Decidimos buscar un headset de realidad aumentada encontramos las gafas de Aryzon [14], compramos un ejemplar y probamos qué tal funcionaban.
Descargamos las aplicaciones y juegos que promocionan para ver como era la experiencia usando las gafas. Nos resultó curioso y decidimos desarrollar una aplicación en la que se pudiera ver objetos y/o modelos 3D y poder caminar alrededor de ellas. Para añadirle más funcionalidad, usamos un mando de *Xbox One* que nos permitirá mover, rotar y escalar el modelo a nuestro gusto, con el objetivo de poder manipular la aplicación sin necesidad de tocar el móvil, ya que éste estará colocado en las gafas.
Para el desarrollo de la aplicación usamos el SDK de Aryzon, que nos facilita la vista

[3] Proceso de crear un sistema de controles digitales y agregárselos a un modelo 3D para que así pueda ser animado fácilmente y eficientemente

estereoscópica[4] para simular nuestros ojos, ya que las gafas están compuestas por dos espejos que reflejan la imagen al cristal por donde nosotros vemos el mundo real, permitiendo tener esa visión de realidad aumentada como se puede observar en la figura 4.1.

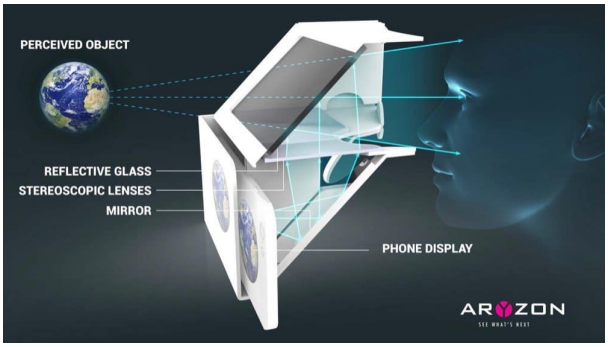


Figura 4.1: Como funciona las gafas de Aryzon. Imagen sacada de (https://www.igadgetsworld.com/aryzon-diy-augmented-reality-headset/

Todos los ejemplos que proporciona Aryzon son con marcadores o no usan de verdad realidad aumentada, si no que que es una escena de realidad virtual con fondo negro y vista esteoroscópica, entonces da el efecto de ser realidad aumentada, pero la aplicación en sí es realidad virtual. Nosotros hemos decidido usar ARCore, ya que queremos usar la realidad aumentada sin marcadores y movernos alrededor del modelo, y como hemos visto en las pruebas, la estabilidad de los puntos de anclaje de ARCore funcionan muy bien.
Continuará...

[4] Dos puntos de vista poco separados entre sí

Capítulo 5

Conclusiones

Capítulo 6

Contribuciones

6.1. Colin Ulrich Cop

6.2. Patricia Cabrero Villar

6.3. David González Jiménez

Índice de figuras

Índice de cuadros

Bibliografía

[1] Julie Carmigniani y Borko Furht. "Augmented Reality: An Overview". En: 3.ª ed. Vol. 4. 5. Handbook of Augmented Reality. Nueva York: Springer, jul. de 1993. Cap. 1, págs. 03-46.

[2] Wayne Piekarski y Bruce Thomas. *ARQuake: The Outdoor Augmented Reality Gaming System*. 1.ª ed. Bubok Publishing S.L, 2002.

[3] D. Marimon. "Advances in Top-Down and Bottom-Up Approaches to Video-Based Camera Tracking". Tesis doct. France: École Poly- technique Fédérale de Lausanne., jul. de 2007.

[4] Guan Tao y Wang Cheng Wang Tian Yuan. "Real-Time Occlusion Handling in Augmented Reality Based on an Object Tracking Approach." En: *MDPI* 10.10 (jul. de 2010).

[5] J.A Albusac Jiménez y J.J Castro Sánchez C. González Morcillo. *Realidad Aumentada. Un enfoque práctico con ARToolkit y Blender*. 1.ª ed. Bubok Publishing S.L, 2012.

[6] Shoaib Ehsan y Adrian F. Clark Erkan Bostanci Nadia Kanwal Nadia. "User Tracking Methods for Augmented Reality". En: *International Journal of Computer Theory and Engineering* 5 (2013), pág. 98.

[7] BBC Earth Lab. *How Does Facial Recognition Work? | Brit Lab*. 2015. URL: https://www.youtube.com/watch?v=1aHub80AHFk (visitado 02-09-2019).

[8] Maxst. *SLAM, Core technology of AR, What is it?* 2018. URL: https://medium.com/maxst/slam-core-technology-of-ar-what-is-it-e6c9ae4839b4 (visitado 02-09-2019).

[9] 8thWall. *8thWall developer documentation*. 2019. URL: https://www.8thwall.com/index.html (visitado 01-09-2019).

[10] 8thWall. *8thWall Jini Web Demo*. 2019. URL: 8th.io/jini (visitado 08-09-2019).

[11] 8thWall. *8thWall XR Remote*. 2019. URL: https://play.google.com/store/apps/details?id=com.the8thwall.XRRemote&gl=ES (visitado 08-09-2019).

[12] Apple. *Apple Developer Documentation*. 2019. URL: https://developer.apple.com/augmented-reality/arkit/ (visitado 01-09-2019).

[13] ARToolKit. *ARToolKit Webpage*. 2019. URL: artoolkit.org (visitado 09-09-2019).

[14] Aryzon. *Aryzon official Website*. 2019. URL: https://www.aryzon.com (visitado 09-09-2019).

[15] cbspicer. *Modelo 3D de tornillos*. 2019. URL: https://www.turbosquid.com/3d-models/free-bolt-assembly-nut-3d-model/317816 (visitado 09-09-2019).

[16] EasyAR. *EasyAR*. 2019. URL: https://www.easyar.com/view/sdk.html (visitado 01-09-2019).

[17] E. Games. *Getting started with Unreal Engine 4 and Arkit*. 2019. URL: https://www.unrealengine.com/en-US/blog/getting-started-with-ue4-and-arkit (visitado 01-09-2019).

[18] V. Geroimenko. *Augmented reality games II, The gamification of education, medicine and art*. 1.ª ed. Springer, 2019.

[19] Wikitude GmbH. *Wikitude Documentation*. 2019. URL: https://www.wikitude.com/download/ (visitado 01-09-2019).

[20] Wikitude GmbH. *Wikitude Unity Documentation*. 2019. URL: https://www.wikitude.com/external/doc/documentation/latest/unity/gettingstartedunity.html#getting-started (visitado 08-09-2019).

[21] Google. *ARCore developer documentation*. 2019. URL: https://developers.google.com/ar/ (visitado 01-09-2019).

[22] Google. *ARCore Fundamental Concepts*. 2019. URL: https://developers.google.com/ar/discover/concepts (visitado 02-09-2019).

[23] Google. *ARCore supported devices*. 2019. URL: https://developers.google.com/ar/discover/supported-devices (visitado 08-09-2019).

[24] Google. *Google Cloud Platform*. 2019. URL: https://console.cloud.google.com/home/dashboard (visitado 09-09-2019).

[25] Google. *Google's Github*. 2019. URL: https://github.com/google-ar/arcore-unity-sdk/releases (visitado 08-09-2019).

[26] Ikea. *Instrucciones de montaje del sofá KIVIK - IKEA*. 2019. URL: https://www.youtube.com/watch?v=F_2Mh3aa0Zs (visitado 09-09-2019).

[27] Kudan. *Kudan License Keys*. 2019. URL: https://www.xlsoft.com/doc/kudan/development-license-keys/ (visitado 08-09-2019).

[28] Kudan. *Kudan official webpage*. 2019. URL: https://www.kudan.io/ (visitado 08-09-2019).

[29] Maxst. *Maxst developer documentation*. 2019. URL: https://developer.maxst.com/ (visitado 01-09-2019).

[30] Statista. *Topic: Augmented Reality (AR)*. 2019. URL: https://www.statista.com/topics/3286/augmented-reality-ar/ (visitado 07-08-2019).

[31] Unity. *About AR Foundation*. 2019. URL: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@2.1/manual/index.html (visitado 01-09-2019).

[32] Unity. *Unity Dashboard Services*. 2019. URL: https://developer.cloud.unity3d.com/projects/ (visitado 09-09-2019).

[33] Unity. *Unity's Github*. 2019. URL: https://github.com/Unity-Technologies/arfoundation-samples (visitado 09-09-2019).

[34] Vuforia. *Vuforia developer documentation*. 2019. URL: https://developer.vuforia.com/pricing (visitado 01-09-2019).

[35] Xlsoft. *Kudan developer documentation*. 2019. URL: https://www.xlsoft.com/en/products/kudan/price.html (visitado 01-09-2019).

[36] Lior Wolf Yaniv Taigman Ming Yang y Marc'Aurelio Ranzato. *Creating a Multiuser AR Experience*. 2019. URL: https://developer.apple.com/documentation/arkit/creating_a_multiuser_ar_experience?language=objc (visitado 02-09-2019).

[37] Marc'Aurelio Ranzato Yaniv Taigman Ming Yang y Lior Wolf. *DeepFace: Closing the Gap to Human-Level Performance in Face Verification*. 2019. URL: https://research.fb.com/publications/deepface-closing-the-gap-to-human-level-performance-in-face-verification/ (visitado 02-09-2019).

PASCAL
ENERO 2018
Ult. actualización 9 de septiembre de 2019
LaTeX lic. LPPL & powered by TEFLON CC-ZERO