TRABAJO FIN DE GRADO

GRADO EN INGENIERIA INFORMATICA

EditAR (working tittle)

Editor de escenas de Realidad Aumentada

Autor
Pablo Millán Cubero

Director
Francisco Javier Melero Rus

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Granada, Julio de 2023

EditAR
Editor 3D web para escenas de Realidad Aumentada

Pablo Millán Cubero

Palabras clave: *software libre, informática gráfica, Typescript, React, Firebase, NodeJS, Express, Android Studio, Kotlin, Sceneview, GLB*

Resumen

Las tecnologías de Realidad Aumentada nos permiten combinar elementos virtuales y reales en tiempo real pudiendo así crear experiencias aplicables a muchos campos; filtros con maquillaje para fotos en redes sociales, previsualizar cómo quedaría un mueble en tu propio salón, dibujar en un partido de fútbol la trayectoria que recorrió la pelota en una repetición, o incluso en videojuegos como Pokemon GO.

Con el avance de los teléfonos móviles de todas las gamas en cuanto a cámaras y capacidad de procesamiento en la última década estas experiencias han pasado a estar al alcance de todo el mundo que posea un dispositivo básico con resultados muy vistosos. Sin embargo en la mayoría de experiencias de Realidad Aumentada el usuario ocupa el rol de consumidor pasivo del contenido que generan los desarrolladores.

Se propone entonces desarrollar una aplicación web en el que el usuario pueda cargar múltiples modelos 3D con textura y animaciones, aplicarles a estos transformaciones como translaciones, rotaciones y escalado, reproducir animaciones y cargar pistas de audio. Todo ello desde un sencillo e intuitivo editor que puede usar cualquiera en el que no haga falta tener conocimientos previos. La escena 3D que se cree podrá posteriormente descargarse o guardarse en un servidor web asociado a una cuenta de usuario. El usuario podrá además, desde una aplicación Android cargar sus escenas creadas y reproducirlas, pudiendo visualizarlas en su entorno a través de la cámara del dispositivo móvil.

EditAR
Web editor for Augmented Reality scenes

Pablo Millán Cubero

Keywords: *software libre, informática gráfica, Typescript, React, Firebase, NodeJS, Express, Android Studio, Kotlin, Sceneview, GLB*

Abstract

Augmented Reality technologies let us merge virtual elements with camera footage in real time, letting us create experiences for many fields; makeup filters for social network photos, preview new furniture for your livingroom, draw the trayectory of the ball in a football match replay, or even in videogames like Pokemon GO.

With the improvement of smartphone's cameras and processing power in the last decade, this experiences are know within everybody's reach if you have a basic device. However, the mayority of Augmented Reality experiences are based in the pasive comsumption of content pregenerated by developers.

The proposed web application lets the user load multiple 3D models with textures and animations, apply transformations like translations, rotations and scales, and play audio tracks. All this in an easy and intuitive editor that anyone can use without previous knowledge. The resultant 3D scene can be either downloaded or uploaded to a web server asociated to an user account. Aditionaly, the user can load the created scenes within an Android app and play them with their device's camera, placing it in their environment.

D. **Tutora/e(s)**, Profesor(a) del ...

Informo:

Que el presente trabajo, titulado ***Chief***, ha sido realizado bajo mi supervisión por **Estudiante**, y autorizo la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a Junio de 2018.

El/la director(a)/es:

(nombre completo tutor/a/es)

Pablo Millán Cubero

Agradecimientos

A mis padres por permitirme estudiar esta carrera en Granada y apoyarme.

A mis abuelos, en especial a mi abuela Elena, que tanto presume de "su nieto el ingeniero".

A mis tíos "los franceses".

A mis amigos del Krustáceo Krujiente, Juan, Antonio, y mis compañeros de trinchera Pepe, Finn, Moisés y Pablo.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y fundamentos

En este capítulo se dará una primera introducción a la *Realidad Aumentada* y a la idea del proyecto, además de describir una serie de conceptos y tecnologías fundamentales para entender el trabajo y seguir con la lectura del resto del documento.

1.1. Descripción general y motivación

El término de Realidad Aumentada comprende al conjunto de técnicas y tecnologías que permiten la superposición e interacción de elementos virtuales sobre la realidad física a través de un dispositivo electrónico. Esta tecnología se ha aplicado a una gran variedad de ámbitos, tales como en la creación de filtros faciales para fotos en redes sociales como Instagram, dibujar la trayectoria que ha seguido el balón en la repetición de una jugada de fútbol, muestra de distintos gráficos y maquetas en los telediarios, previsualización de mobiliario en una habitación e incluso videojuegos como Pokémon GO.

El avance en las cámaras y procesadores de los teléfonos móviles en la última década ha facilitado que casi cualquier persona tenga en el bolsillo un dispositivo capaz de reproducir experiencias muy vistosas de Realidad Aumentada. Sin embargo, en la mayoría de casos se trata de aplicaciones en las que el usuario consume pasivamente contenido pre generado por los desarrolladores, sin la capacidad de crear ellos mismos estas experiencias.

Se propone entonces desarrollar en primer lugar una aplicación web que permita al usuario la composición de textitescenas 3D para uso en Realidad Aumentada. Cuando hablamos de escena nos referimos una composición de uno o más modelos 3D con una disposición concreta. Por ejemplo podríamos colocar el modelo de un tobogán junto al de un columpio y un balancín, formando el

conjunto la escena de un parque. La aplicación consistirá principalmente de un editor sencillo e intuitivo que permita cargar modelos 3D con texturas y aplicarles transformaciones para colocarlos en una escena. Adicionalmente, se podrá añadir a la escena una pista de audio y reproducir animaciones de los modelos. Se podrá guardar la escena en un servidor web en una colección de escenas generadas asociadas a una cuenta de usuario, desde la cual se podrán modificar o eliminar.

Adicionalmente, el usuario podrá iniciar sesión en una app móvil Android para acceder a su lista de escenas creadas y podrá reproducir cualquiera de ellas en Realidad Aumentad con su dispositivo móvil.

Las principales motivaciones para elegir este como mi trabajo de fin de grado fue mi alto grado de interés por los gráficos 3D, mis conocimientos previos en tecnologías que acabaría usando como three.js o Android Studio, y que es un proyecto relativamente multidisciplinar, requiriendo del desarrollo de una aplicación web, una aplicación Android y un servidor web que los conecte, montando así un pequeño ecosistema y con suerte dándome una intuición de lo que puede ser realizar un proyecto parecido en un entorno laboral real.

1.2. Fundamentos

A continuación se van a describir una serie de conceptos y tecnologías esenciales para la correcta comprensión del proyecto.

1.2.1. Realidad Aumentada

La *Realidad Aumentada* (*Augmented Reality* en inglés, *RA* de aquí en adelante) hace referencia al conjunto de técnicas y tecnologías que permiten crear una experiencia en la que se combina el mundo real con contenido generado por computadores. Aunque el uso más extendido es únicamente visual y auditorio, puede apelar a más sentidos como el háptico, somatosensorial y olfativo. Se puede decir que un sistema *RA* debe incorporar tres elementos básicos: una combinación del mundo real y el virtual, interacción en tiempo real y un correcto registro de la posición y movimientos de objetos tanto reales como virtuales.

Existen otros términos acuñados que son a grandes rasgos sinónimos de la *RA* como *Mixed Reality*. No debe confundirse con la *Realidad Virtual*, otro ámbito en el que el objetivo es sumergir al usuario en un mundo virtual enajenándolo de sus sentidos con el mundo real, usualmente con unas gafas envolventes diseñadas específicamente para realidad virtual. Todas estas disciplinas están recogidas bajo el término paraguas de la *Extended Reality* (*XR*).

A continuación se van a describir los tipos de escenas que podemos encontrarnos en el contexto de la Realidad Aumentada.

Figura 1.1: Ejemplo de Realidad Aumentada en el videojuego Pokémon Go

- **Posicionamiento en superficies**: Es quizás el ejemplo más sencillo tanto conceptualmente como técnicamente. El dispositivo intentará analizar sus alrededores a través de la cámara e intentará detectar superficies planas como podría ser una el suelo, una mesa o una pared. Colocará objetos en puntos concretos de estas superficies, y al mover el dispositivo esos objetos deberán visualizarse como si estuvieran colgados o colocados en dichas instancias del mundo real. Existe todavía a día de hoy una limitación importante a la hora de realizar estas escenas; si la superficie es demasiado lisa y uniforme, el dispositivo la reconocerá con gran probabilidad como un color plano, y no tendrá la capacidad de identificarla como una superficie, haciendo que no pueda llegar a colocar nada.

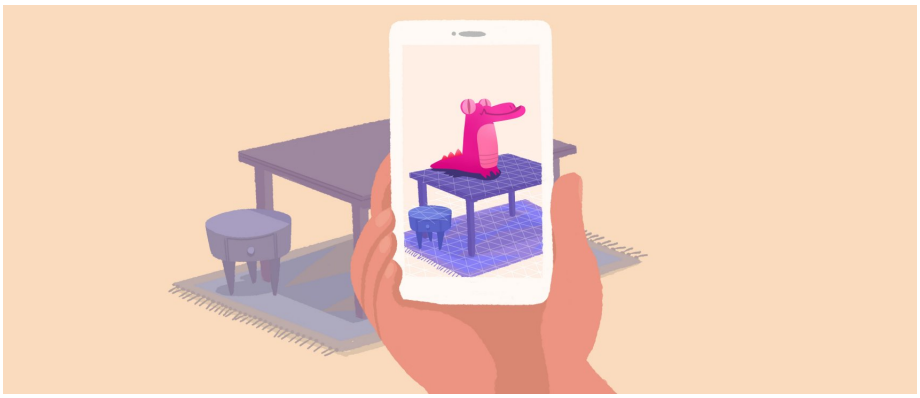

Figura 1.2: Ejemplo de posicionamiento en superficies de la documentación de *ARCore* [6]

- **Imágenes aumentadas**: En este tipo de escena, cada modelo o conjunto de modelos tiene asociada una imagen bidimensional a la que denominamos **marcador** o **activador**. Cuando se inicie, el dispositivo tratará de reconocer esta imagen en el mundo real. Una vez identificada se reproducirán sobre la imagen (tratándola como una superficie) o en una posición relativa a la imagen, los modelos pertinentes que se hayan definido. Es posible definir distintos objetos asociados a distintos marcadores y que solo se reproduzcan los elementos pertinentes.

Figura 1.3: Ejemplo de escena de imágenes aumentadas de la web de *Pictarize Studio* [1]

- **Escenas geoespaciales**: Son escenas en las que los objetos no se posicionan en una superficie, si no en unas coordenadas GPS. El usuario solo podrá ver la escena si se encuentra a una latitud, longitud y altura cercanos a aquella que se definieran para la escena. Para ello logicamente es esencial disponer de conexión a internet y que el dispositivo tenga acceso a las coordenadas actuales del usuario durante la ejecución. Las localización GPS de los dispositivos móviles no suele ser del todo precisa, así que usando solo estas es posible que si queremos, por ejemplo, colocar un modelo 3D que represente cómo era la *Alhambra de Granada* cuando se construyó justo en la puerta del edificio, es posible que algunos usuarios la vean algunos metros fuera de lugar, o incluso flotando en el aire. Para poder tener algo más de precisión, algunas APIs geoespaciales como la de *ARCore* [6] permiten extraer información de *Street View*, el visor en primera persona de *Google Maps* para reconocer a través de puntos clave,

fachadas de edificios y calles que hayan sido captadas previamente por este servicio, resultando así en una mejor aproximación a la verdadera posición del usuario y por tanto un posicionamiento más preciso de las escenas.

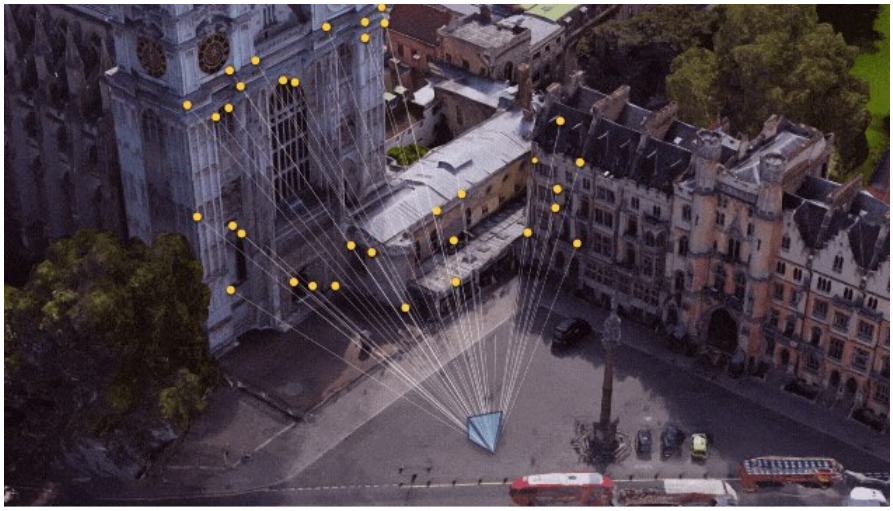

Figura 1.4: Ejemplo de la aproximación por puntos clave para escenas geoespaciales de la documentación de *ARCore* [6]

1.2.2. Visión por computador

Javier González Jiménez define en su libro *Visión por computador* [9] este campo con la siguiente cita:

"La visión por computador, también denominada visión artificial, puede definirse por el proceso de extracción de información del mundo físico a partir de imágenes, utilizando para ello un computador. Desde un punto de vista más ingenieril, un sistema de visión por computador es un sistema autónomo que realiza algunas de las tareas que el sistema de visión humano [...] realiza. La información o tareas que este sistema de visión puede llegar a extraer o realizar puede ir desde la simple detección de objetos sencillos en una imagen hasta ***la interpretación tridimensional de complicadas escenas****. "*

"[...] la información visual consiste en energía luminosa procedente del entorno. Para utilizar esa información es preciso transformarla a un formato susceptible de ser procesado [...] esta misma tarea se realiza con una cámara de vídeo (o algún otro dispositivo similar), que convierte la energía luminosa en corriente eléctrica, que puede ser entonces muestreada y digitalizada para su procesamiento en un computador."

En nuestro caso nos interesa especialmente la sentencia destacada de la cita. Esta disciplina nos permite extraer una idea de la estructura física de un escenario,

permitiéndonos detectar elementos como superficies o esquinas, indispensables si queremos por ejemplo, posicionar un objeto 3D encima de una mesa, y que el objeto parezca que siga apoyado en la mesa si movemos la cámara de lugar, cambiamos el ángulo, o nos alejamos.

Por fortuna la obtención de información a través de imágenes será un proceso completamente transparente para nosotros gracias a las distintas herramientas y tecnologías que ofrecen una API para hacer llamada de estas funciones sin que tengamos que preocuparnos por los detalles técnicos como veremos a continuación.

1.2.3. ARCore

ARCore [6] es un *SDK* (*software development kit*) de Realidad Aumentada desarrollado por Google. Ofrece una API con la que se pueden desarrollar experiencias *AR* en Android, iOS, Unity y Web renderizando modelos 3D con *OpenGL*. Para ello ARCore realiza una serie de tareas a bajo nivel relativas a la visión por computador y por las cuales el usuario no tiene que preocuparse, solo hacer las llamadas necesarias a su API. Estas funciones se detallan a continuación.

- **Seguimiento de movimiento**: Se emplea un proceso llamado *localización y asignación simultáneos*, o ANSM, para que el dispositivo obtenga un contexto sobre el mundo físico que lo rodea. Para ello ARCore detecta características visualmente distintas en las imágenes capturadas (también llamadas **puntos de atributo**) y los usa para calcular el cambio de ubicación del dispositivo. La información visual se combina con la IMU[1] del dispositivo para calcular la pose, es decir, la posición y orientación de la cámara respecto al mundo a lo largo del tiempo.
- **Comprensión ambiental**: Búsqueda de clústeres de puntos de atributos que aparentan pertenecer a superficies horizontales o verticales como pareces o mesas, y las pone a disposición del usuario como *planos* que pueden emplearse para posicionar objetos.
- **Comprensión de profundidad**: ARCore detecta y almacena *mapas de profundidad* con información sobre la distancia entre dos puntos cualquiera pertenecientes a una superficie. Esto puede emplearse para simular interacciones físicas realistas entre objetos virtuales y reales o lograr el efecto de que un objeto virtual aparezca tapado o detrás de uno real.
- **Estimación de luz**: Detecta información sobre la luz de su entorno, su intensidad y color. Con esto se pueden lograr efectos como que un objeto

[1] Un IMU (Unidad de Medida Inercial) es un dispositivo capaz de estimar y reportar información acerca de la velocidad y orientación del mismo a través de acelerómetros y giroscopios.

virtual a la sombra se represente más oscuro que uno expuesto a una fuente de luz.

- **Estimación de luz**: Detecta información sobre la luz de su entorno, su intensidad y color. Con esto se pueden lograr efectos como que un objeto virtual a la sombra se represente más oscuro que uno expuesto a una fuente de luz.

- **Interacción de usuario**: A través de raycasts[2], ARCore puede obtener un *punto de atributo* o modelo posicionado en la escena a partir de un gesto de usuario como tocar la pantalla táctil del dispositivo, lo que permite interacciones como colocar o mover objetos 3D.

- **Puntos orientados**: Los puntos orientados permiten posicionar objetos 3D en superficies que no son superficies planas. ARCore analiza los puntos de atributo vecinos al seleccionado para posicionar el objeto y realizará una estimación de la inclinación, devolviendo un ángulo que servirá para aplicar una pose al objeto para que se vea lo más natural posible.

- **Imágenes aumentadas**: ARCore puede detectar a través de la cámara imágenes 2D definidas previamente y asociar a cada una uno o varios modelos 3D que posicionar sobre estas en el caso de que se enfoquen. Los modelos mantendrán su posición de forma consistente si la cámara se mueve o rota.

1.2.4. SceneView

Sceneform es un *SDK* desarrollado por *Google* que permite importar y visualizar modelos 3D en distintos formatos para renderizar escenas 3D para aplicaciones de *ARCore* o realidad virtual sin necesidad de programar código *OpenGL* de bajo nivel. Cuenta con un *grafo de escena* a través del cual podemos definir la estructura de los modelos de la misma a través de un árbol de nodos, y un motor de físicas de *Filamente*[3].

Lamentablemente, *Sceneform* fue abandonado por *Google*, pero el proyecto ha renacido bajo el nombre de **Sceneview** [8], desligándose de *Google*, con código abierto y mantenido por su comunidad.

1.2.5. Android y Android Studio

Android es un sistema operativo desarrollado por Google, de código abierto y basado en el núcleo de *Linux*. En un principio se diseñó con dispositivos táctiles

[2] text raycast

[3] Filament: Motor de físicas para escenas 3D en tiempo real desarrollado por Google.

en mente como teléfonos inteligentes y *tablets*. Es el que actualmente tiene la mayor cuota de mercado en dispositivos de este tipo, contando en 2022 con el 72,2%[4]. También se emplea en otro tipo de sistemas como televisiones y relojes inteligentes o incluso automóviles.

Android Studio es un entorno de desarrollo integrado o *IDE*, es decir, una aplicación informática que proporciona servicios y recursos para el desarrollador de software como un editor de código, depurador, compilador e interfaz gráfica. En este caso para el desarrollo de aplicaciones en dispositivos Android. Este tipo de herramientas es de gran interés para encarar un desarrollo, ya que aglutina todas las herramientas que podemos necesitar como programadores en un solo entorno.

Kotlin es un lenguaje de programación multiplataforma, estáticamente tipado, con inferencia de tipos y de alto nivel. Está diseñado para ser totalmente inter operable con *Java*, pero usando una sintaxis más concisa. Es además, el lenguaje preferido por Google para el desarrollo de aplicaciones Android y el que se recomienda en toda la documentación.

1.2.6. Typescript

JavaScript (*JS*) es un lenguaje de programación interpretado y compilado *just-in-time*, aunque es más conocido como un lenguaje de *scripting*, es decir, un lenguaje que nos permite incrustar código dentro de páginas web.

Por otro lado, *TypeScript* (*TS*) es un lenguaje construido por encima de JavaScript que añade sintaxis adicional que lo hace **fuertemente tipado**. Es decir, siempre que declaremos una variable debemos especificar su tipo. Si más adelante en el código se intenta asignar un valor no correspondiente a ese tipo, el editor podrá detectarlo y avisar, pudiendo así identificar posibles errores antes siquiera de la ejecución del programa, cosa que con *JavaScript* no es posible.

1.2.7. Archivo JSON

Los archivos JSON (o *JavaScript Object Notation*) es un formato de texto sencillo para el intercambio de datos. Toma como referencia la notación de objetos de *JS*, pero a lo largo de los años y debido a su simplicidad, se ha convertido en la opción por defecto para intercambio de información entre servicios web, en contraposición a su principal alternativa, *XML*.

[4] https://root-nation.com/es/noticias-es/es-cuota-mercado-android-ios-estadisticas-publicadas-

1.2.8. Three.js

Three.js [?] es una librerías *JavaScript* usada para crear y renderizar gráficos 3D animados en páginas web para que puedan se reproducidas en un navegador. Para ello hace uso de *WebGL*, otra librería *JavaScript* para la renderización de gráficos pero a más bajo nivel. *Three.js* se abstrae de conceptos técnicos y de bajo nivel para que el desarrollador no tenga que preocuparse de todo eso y agilice su desarrollo. Además es una librería de código abierto mantenida por su comunidad.

1.2.9. React.js

React [?] es una librería de *JavaScript* para construir interfaces de usuario web de código abierto y mantenido por su comunidad. Opcionalmente, utiliza ficheros de extensión *.jsx* (*JavaScript Syntax Extension*) para facilitar el desarrollo. Estos son, como su nombre indica, una extensión sintáctica de *JS* que permite insertar código HTML, facilitando así la construcción de interfaces de usuario.

1.2.10. Node.js

Node [5] es un entorno de ejecución de *JavaScript* orientado a eventos asíncronos diseñado para crear aplicaciones de red rápidas capaces de manipular grandes cantidades de conexiones concurrentes con una alta escalabilidad para los proyectos.

1.2.11. Express

Express [3] es un *"Web application framework minimalista y flexible para Node.js que provee una serie de herramientas para aplicaciones web y móviles."* como lo definen ellos mismos. Cuenta con una gran variedad de utilidades para llamadas *HTTP* y *middleware*. Cuando hablamos de *middleware* nos referimos a un software a través del cual distintas aplicaciones se comunican entre sí por internet.

1.2.12. Firebase

Firebase [7] es un BAAS[5] desarrollado por *Google* que pone a disposición de los desarrolladores una serie de herramientas y servicios para facilitar la creación

[5] Backend-as-a-service: aplicaciones puestas al servicio de desarrolladores para que estos hagan usos de sus servicios y se centren en desarrollar el frontend de sus apps, siendo innecesario así que creen ni mantengan sus propios backends.

de aplicaciones. Algunas de sus servicios más importantes son:

- **Autenticación**: Soporta autenticación de usuarios usando contraseñas con correo electrónico, número de teléfono, o cuentas de *Google*, *Facebook* y *Twitter* entre otros. Ofrece también encriptación por defecto en la base de datos para contraseñas.
- **Base de datos en tiempo real**: *Firebase* tiene también una base de datos **no relacional** que se actualiza a tiempo real y se mantiene en línea incluso cuando la aplicación no se está ejecutando.
- **Hosting**: Permite *hostear* aplicaciones web de forma fácil y rápida, pudiendo almacenar contenido en una memoria caché y accesible desde cualquier parte del mundo.

1.2.13. Archivo GLB

GLB (*.glb*) es un formato de archivo binario estandarizado para representar datos de objetos 3D como modelos, escenas, iluminación, materiales, jerarquías y animaciones. Sus siglas significan *GL Transsmission Format Binary File*. Fue introducido en 2015 como una versión binaria de los ficheros **GLTF** (*.gltf*), un formato basado en la notación *JSON* muy usado en aplicaciones web y móvil debido a su ligero peso.

Capítulo 2

Descripción del problema

En este capítulo se detallará el problema que se pretende resolver con este proyecto, además de establecer los objetivos que se requieren cumplir. Se realizará una crítica al estado del arte y para finalizar se enunciará una propuesta de producto.

2.1. Problema a resolver

Como se ha comentado en apartados anteriores, existen en el mercado multitud de experiencias de Realidad Aumentada al alcance de los usuarios, aunque en la gran mayoría de ellas los usuarios juegan un rol pasivo. Esto último no hace referencia a que no puedan interactuar con el contenido, (ya hemos visto que la interacción en tiempo real es uno de los rasgos definitorios de la *RA*) sino que consumen contenidos y experiencias que han sido creadas por los desarrolladores de las distintas apps. Suelen ser sistemas cerrados donde el usuario no tendría opción de, por ejemplo, cargar un modelo 3D de su dispositivo para visualizarlo en su salón.

Por otro lado, las herramientas de modelado y animación de objetos 3D como *Blender*[1] o *3ds Max*[2] son herramientas profesionales que requieren de una formación, conocimientos previos y curva de aprendizaje para poder manejarlos con soltura. Incluso si las aplicaciones pudieran efectivamente cargar modelos en las aplicaciones ya mencionadas, estarían limitados a elementos sueltos que pudieran encontrar en internet. Esto es algo que limita mucho tus opciones creativas. Volviendo al ejemplo de la escena del parque que se puso en el capítulo 1, si se pretende recrear hacer la escena de un parque, quizás se encuentre un

[1] link blender
[2] link 3ds max

modelo 3D para ello. Pero, ¿y si se quiere tenga concretamente dos toboganes y un columpio? Es muy poco probable que alguien haya modelado y publicado una escena con exactamente los mismos requisitos. Pero contando con que se puedan encontrar los elementos por separado (un modelo de un tobogán, de un balancín, de un columpio...) lo cual es más realista, y alguna forma de componerlos en una sola escena, se podría construir un parque con una composición cualquiera.

Teniendo todo esto en cuenta se detectan dos necesidades para el usuario. Por un lado, una suerte de editor sencillo, simplificado e intuitivo en el que un usuario sin conocimiento o experiencia previa pueda cargar o crear modelos 3D con los que pueda construir una escena con total libertad creativa. Por otro lado, una manera de que el usuario pueda reproducir cómodamente sus composiciones producidas en un contexto de Realidad Aumentada en un dispositivo con cámara como podría ser un *smartphone*.

Este escenario presenta ciertos retos de diseño e implementación. El software profesional de modelado profesional es complejo por una razón: tienen muchísimas posibilidades. Si se quiere tener una herramienta sencilla inevitablemente el usuario perderá opciones. Por tanto, se debe encontrar un compromiso, un *sweetspot* en el que se intenten cubrir casi todas las necesidades que podría tener una persona sin que el software se vuelva demasiado obtuso. También se debe de tener en cuenta un aspecto técnico importante: qué formato de archivos para modelos soportará el sistema. Existe una gran variedad hoy en día, cada uno con sus peculiaridades, y no siempre compatibles entre ellos, por lo que el software deberá adoptar uno o contar con herramientas de conversión para soportar un conjunto de ellos. También hay que tener en cuenta que el tipo de dispositivo en el que principalmente se tiene interés por reproducir escenas *RA* son teléfonos móviles o tablets debido a sus cámaras integradas y su portabilidad, pero estos se controlan con gestos táctiles, los cuales no son tan precisos como un ratón de ordenador y podría ser una dificultad a la hora de implementar controles precisos para el editor. Por ello deberá buscarse una forma de adaptar este tipo de programa a un control táctil o bien implementar una conectividad ordenador-*smartphone* para poder crear la escena en el primero y reproducirla en el segundo. En este TFG se propondrá y desarrollará una solución para estas necesidades.

2.2. Objetivos

Una vez definidos los retos y problemas a resolver en este contexto, se procede a enunciar los objetivos a los que se pretende llegar con este proyecto:

- **OBJ-1**: Desarrollar un software que permita cargar y crear modelos 3D a los que aplicarles transformaciones como translaciones, rotaciones y escalado para construir una escena.

- **OBJ-2**: El software descrito descrito debe ser simple, sencillo e intuitivo para que lo pueda usar cualquier persona sin que tenga experiencia ni conocimientos previos en la materia.
- **OBJ-3**: El software debe permitir la máxima flexibilidad posible a la hora de producir las escenas teniendo en cuenta su alcance limitado y controles sencillos.
- **OBJ-4**: Desarrollar una aplicación informática que reciba como entrada uno o varios modelos 3D que compongan una escena y reproduzca con ellos una experiencia de realidad aumentada.
- **OBJ-5**: La solución aportada debe ser ligera y lo más rápida posible, para que los usuarios de dispositivos de gama media o baja puedan usarlo sin problemas.
- **OBJ-6**: El sistema debe tener memoria, es decir, un usuario deberá de alguna forma poder conservar las escenas generadas con anterioridad para que el usuario las pueda volver a modificar en un futuro.

2.3. Estado del arte

Existe en el mercado cierta variedad de opciones desarrolladas que cumplen en mayor o menor medida los objetivos que se han planteado en este TFG. Esta sección se dedicará a analizarlas, compararlas y detectar cuales son las fortalezas y carencias de cada una, con la intención de sacar en claro qué se podría aportar con una nueva solución. Después de investigar y analizar las soluciones más relevantes, se encontraron **cinco plataformas** con un propósito y características similares al proyecto que se pretende desarrollar. De estas cinco se descartaron dos, *Aryel*[3] y *Artificio*[4] ya que aunque superficialmente comparten algunos elementos como son al fin de al cabo la creación de escenas *RA*, la primera está enfocada para que se le dé uso en agencias de marketing para diseñar publicidad interactiva, y la segunda es una herramienta para diseñar interiores de viviendas.

2.3.1. Onirix Studio

Onirix Studio [?] es una plataforma gratuita en la que un usuario al darse de alta puede crear multitud de escenas *RA* a través de un editor. Las escenas producidas quedan registradas en la cuenta de usuario, y pueden ser accedidas a través de un menú que permite volver a editarlas o eliminarlas. En cuanto al editor

[3] https://aryel.io/
[4] https://arficio.com

tiene un gran abanico de opciones. El usuario puede añadir modelos a la escena, posicionarlos, rotarlos y escalarlos con total libertad a través tanto de controles gestuales de ratón como de un panel numérico. Cuenta con la posibilidad de modificar la iluminación que proyecta la escena sobre los modelos. Es posible añadir texto que se visualice junto a los modelos. Para añadir *assets* como los modelos, el usuario cuenta con un catálogo personal, con algunos elementos que añade la aplicación por defecto, pero con la opción de que el usuario añada los suyos propios. Una vez los añade los puede usar en cualquier otro proyecto.

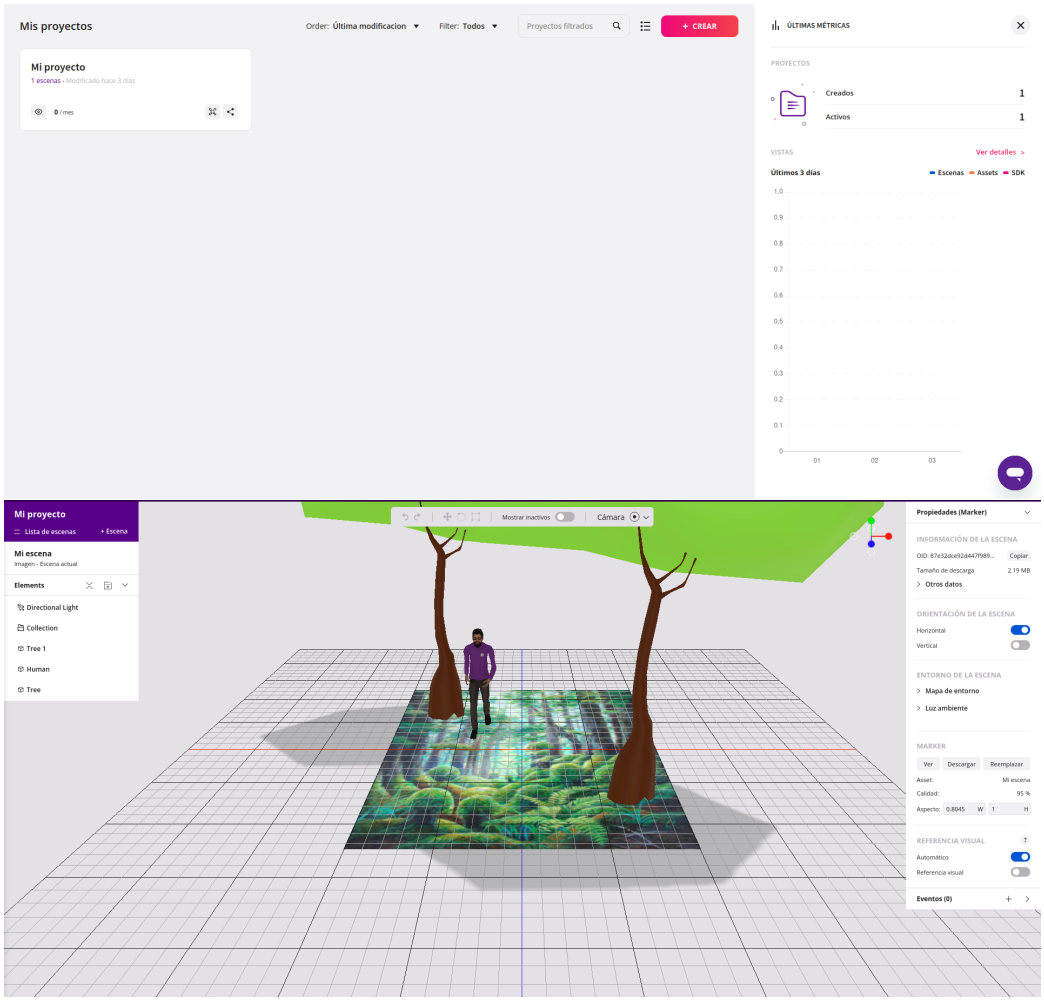

Figura 2.1: Web de creación de escenas RA Onirix Studio

Uno de los puntos más destacables que ofrece este entorno es un sistema al través del cual se pueden definir eventos dinámicos para la escena al cumplirse cierto activador. Por ejemplo, si tenemos un botón pulsable en la escena, se puede añadir una regla que ordena al pulsarse el botón la reproducción de la animación de uno de los modelos, o aparezcan modelos nuevos, desaparezcan

otros, etc.

Para la reproducción tenemos se pueden o bien reproducir en la propia web, o generar un *código QR*[5] escaneable por un *smartphone*. Este *QR* abre en el navegador un enlace a través del cual se reproduce la escena. Por un lado, esta es una solución cómoda y ágil, pero tiene puntos negativos. Para empezar el rendimiento de la cámara y renderización de elementos 3D de un dispositivo móvil se va a ver reducida si es a través de una aplicación web en lugar de una aplicación nativa del teléfono. También hay que tener en cuenta que existe un enorme catálogo de navegadores gratuitos, y aunque la mayoría de los usuarios utilizan los más populares como *Google Chrome*, *Mozilla Firefox*, *Opera* o *Safari*, no se puede asegurar la compatibilidad total con cualquier navegador, o que el rendimiento sea óptimo en todos. Otro punto negativo a destacar de *Onirix Studio* [**?**] es que solo es posible crear escenas de posicionamiento sobre superficie y de imágenes aumentadas.

2.3.2. Blippbuilder

Blippbuilder [**?**] es una web muy similar a la anteriormente comentada, y también se puede emplear sin coste alguno. A través de una cuenta el usuario puede almacenar una colección de escenas RA editables a través de un menú similar. El editor contaba con una estructura parecida, permitiendo también aplicar transformaciones con gestos de ratón. Al igual que la anterior, tiene la opción tanto de cargar recursos desde el ordenador del usuario como de acceder a una librería incluida con la web, opción de añadir texto, manipular la iluminación de la escena. También cuenta con un pre visualizador dentro del editor y con opción para probar la escena desde el navegador de un dispositivo móvil escaneando un código *QR* (es decir, tampoco es nativa).

Como aspecto diferenciador, es posible añadir a la escena geometrías básicas tales como esferas, cubos o cilindros para añadirlos. Además, cuenta con un menú a través del cual se pueden **crear animaciones** para la escena. A través de *keyframes*. Un *keyframe* o *fotograma clave* es una unidad de información que almacena atributos tales como la posición, rotación o escala de un objeto para un instante determinado. Definiendo dos *keyframes* para el mismo objeto en 2 instantes distintos, un software puede interpolar sus posiciones intermedias generando así una animación. Esta animación puede ser reproducida junto a la escena.

En *Blippbuilder* solo es posible la creación de escenas de Realidad Aumentada por superficie.

[5] explicar código qr

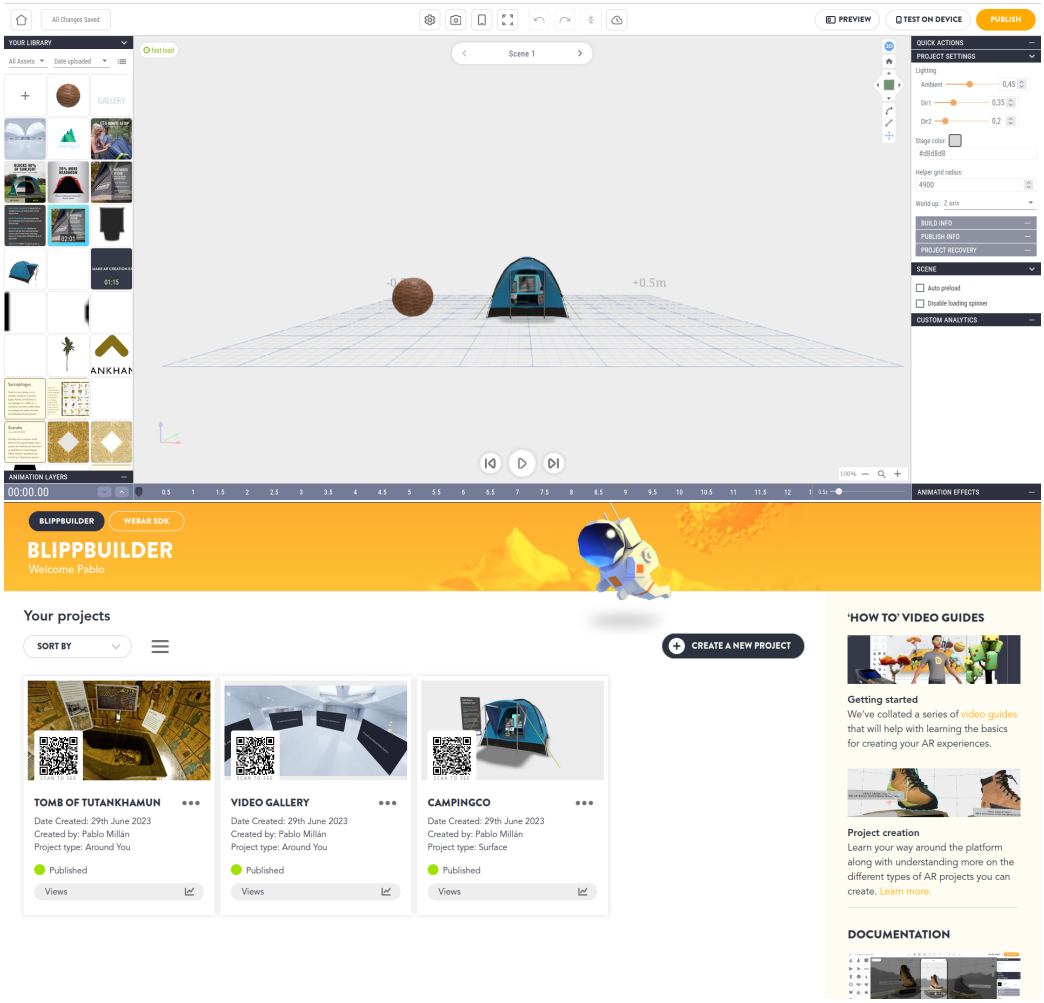

Figura 2.2: Web de creación de escenas RA Onirix Studio

2.3.3. Pictarize

Pictarize [1] de nuevo tiene un funcionamiento muy parecido a lo visto hasta ahora y sin requerir de pagos. Una galería de escenas producidas asociadas a una cuenta de usuario, y un editor para hacer las mismas. Tiene opción para cargar modelos desde el ordenador del cliente y de nuevo controles gestuales con el ratón para manipularlos. Aunque también puede añadir texto, no existe opción para alterar la iluminación. Sin embargo, es posible embeber videos de youtube en forma de textura plana, para reproducirlos al pulsarlos cuando se reproduce la escena. Es posible también asociar una pista de audio a la escena para que se reproduzca junto a las animaciones de los modelos 3D.

Para la reproducción de escenas se cuenta tanto con un pre visualizador

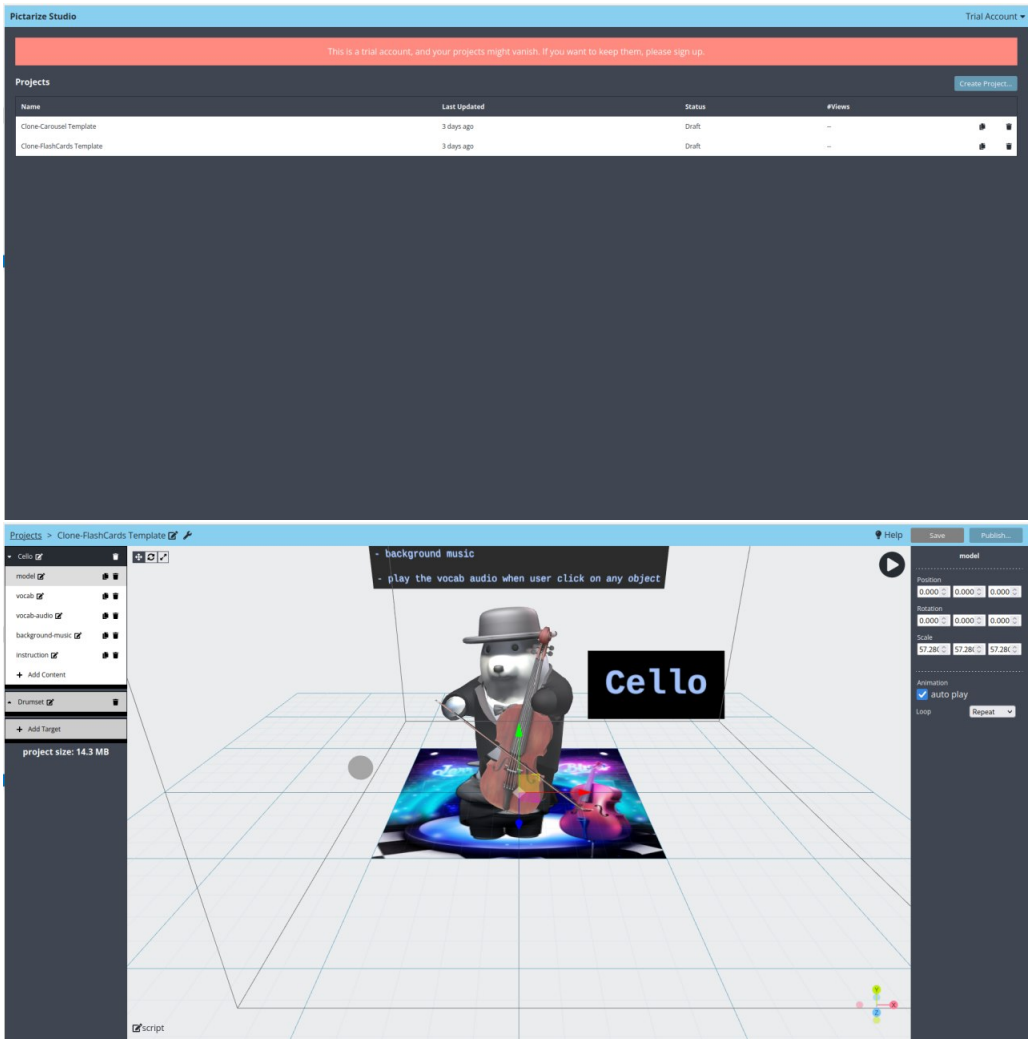

Figura 2.3: Web de creación de escenas RA Onirix Studio

dentro del editor como una opción para verlas desde un *smartphone* a través de un código *QR* y un navegador web, como se ha visto en los dos ejemplos anteriores. Sólo hay opción para escenas de imágenes aumentadas, con la opción de definir distintos activadores para distintos elementos.

2.4. Crítica

Como se ha podido dilucidar a través de estos ejemplos ya hay varias soluciones disponibles y bastante completas aun siendo gratuitas. Sus editores son potentes en cuanto a que permiten un alto grado de expresión por parte del

usuario, permitiendo un gran abanico de posibilidades de forma que el usuario podrá muy probablemente representar la escena que tenía en mente siemrpe que cuente con los modelos adecuados.

Las propuestas existentes analizadas comparten multitud de elementos en común. Todas ofrecen un menú para gestionar una lista de escenas creadas por un usuario, el cual se identifica con una cuenta para la propia web. Esta cuenta usa como identificador un nombre de usuario y correo electrónico. En todas es posible cargar objetos 3D desde el ordenador del cliente.

También permiten aplicar transformaciones de translación, rotación y escalado a los modelos de la escena con gestos del ratón a través de *gizmos*. Estos son unas estructuras visuales interactuables que ayudan a aplicar transformaciones a objetos 3D. Dependiendo de la operación que se quiera realizar, hay distintos tipos de gizmo. Cogiendo el ejemplo del *3D Move Gizmo* de la siguiente figura, si se hiciera click en una de las flechas coloreadas, se permitiría mover el objeto con un desplazamiento del ratón en el eje que señala la misma. Además de controles gestuales permiten introducir manualmente números para representar las transformaciones del objeto a través de un menú. Por ejemplo se puede especificar que un modelo está *'6.2' unidades desplazado en el eje X* o con *una rotación de 90º en el eje Y*.

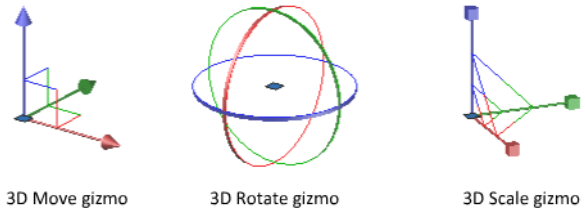

Figura 2.4: Ejemplo de Gizmo en la documentación de Autodesk

Hay otras funciones que si bien no son imprescindibles para el objetivo de crear una escena, dan posibilidades extra de personalización como la adición de texto, manipulación de luces, reproducción de audios, definición de eventos o creación de animaciones con *keyframes*. Hay que tener una cosa en cuenta con este tipo de funcionalidades, y es que si bien aportan valor y posibilidades a la aplicación, también aumentan la complejidad de la aplicación y la curva de aprendizaje que tienen los usuarios. No es descabellado pensar que alguien que no tiene nociones básicas sobre cómo funciona la animación tenga dificultades para manipular una timeline de *fotogramas clave* para crear una pequeña animación. Por tanto, se puede deducir que un porcentaje de los usuarios se sentirán confundidos por funciones como esta o la de definir eventos, o directamente no llegarán a intentar usarla.

También hay presentes algunas carencias en estas webs. Se puede ver que una de ellas permite crear tanto escenas por superficie como de imágenes aumen-

tadas, mientras que en las otras dos opciones solo es posible hacer o imágenes aumentadas o superficie. Y lo que es más, ninguna tiene soporte para generar escenas geoespaciales basadas en la localización.

Otra ausencia a destacar es la imposibilidad en ninguna propuesta de descargar guardar de forma local en el ordenador las escenas que se han creado en un formato de objeto 3D, para que el usuario pueda utilizar ese archivo en cualquier otro software que admita el tratamiento de modelos. Esto se traduce en una imposibilidad de que las escenas salgan del ecosistema de sus respectivas webs.

Por último señalar que en todas las webs analizadas la única opción para reproducir las escenas en un entorno real de Realidad Aumentada en un dispositivo móvil es a través de un navegador web al que se accede escaneando un código QR. Como se ha explicado anteriormente, se trata de una opción cómoda en tanto que es multiplataforma, no hace falta iniciar sesión de nuevo en el segundo dispositivo y en pocos segundos se puede escanear el código y tener cargando la escena, pero el rendimiento y fluidez de la cámara que se obtiene es considerablemente menor que si estuviera corriendo en una aplicación nativa, por lo que no estaría de más tener la opción de ejecutarlo de esta forma.

2.5. Propuesta

Con esta propuesta de proyecto se pretende desarrollar una alternativa completa que intentará cubrir las carencias que se encontraron analizando las webs disponibles en el mercado:

- **Intuitivo**: El editor contará con las herramientas básicas e imprescindibles para poder elaborar una escena 3D con las necesidades del usuario. Esto permitirá mantener una interfaz limpia, legible e intuitiva que permita construir escenas de una forma rápida y que no requiera conocimientos o experiencias previas con software de manipulación de entornos 3D. Así los usuarios no se sentirán perdidos o que tiene más herramientas de las que realmente necesitan.

- **Múltiples tipos de escenas**: Será posible la creación de distintos tipos de escenas de Realidad Aumentada. En concreto permitirá escenas de posicionamiento por superficies, de imágenes aumentadas y geoespaciales en unas coordenadas concretas.

- **Descarga de escenas**: Además de poder almacenar las creaciones en la nube, será posible exportar la escena con formato de objeto 3D para que el usuario pueda emplearla en cualquier otro software que permita archivos de este tipo.

- **Reproducción nativa**: Será posible reproducir escenas en una aplicación nativa para dispositivos móviles, proporcionando un mayor rendimiento y asegurando compatibilidad.

Para eso se desarrollará un servicio completo que contará de tres partes fundamentales. Para empezar, una **aplicación web** que permita la creación y gestión de escenas de realidad aumentada. Por otro lado, se tiene una **aplicación móvil** desde la que reproducir de forma nativa las escenas. Para terminar, un **servidor web** en el que se almacenarán las escenas creadas, que podrán ser recuperadas tanto desde la aplicación web como la aplicación móvil.

En el capítulo 4 se dará una hablará más en detalle del software en una descripción del mismo desde el ángulo de las metodologías ágiles, y en el capítulo 5 se explicará tanto la implementación como las decisiones de diseño y tecnologías empleadas.

Pablo Millán Cubero

Capítulo 3

Estado de la Realidad Aumentada

El software libre y sus licencias [4] ha permitido llevar a cabo una expansión del aprendizaje de la informática sin precedentes.

Capítulo 4

Planificación

4.1. Metodología utilizada

El software es un sistema complejo de crear que depende de muchas partes interconectadas. No es igual de dificultoso programar un pequeño *script* que ordene los ficheros de un directorio de un sistema operativo que construir una aplicación de cierta envergadura que responda a las necesidades de un usuario o un cliente concreto, mucho más si además se debe coordinar un equipo de personas en la concepción del mismo. Existe un gran factor de riesgo e incertidumbre, sobre todo al inicio de un proyecto. Las **metodologías de desarrollo ágil** son una forma de desarrollar software enfocada a equipos pequeños que busca paliar estos problemas a la hora de encarar un proyecto,. En otras metodologías anteriores se redactaban unos requisitos completos y fijos al inicio del proyecto y antes si quiera de comenzar la implementación. Esto era una fuente común de conflictos, ya que es muy común que los requisitos cambien a lo largo del desarrollo. Quizás el cliente cambie de idea, o el equipo encuentre mejores formas de realizar una tarea, o el contexto en el ámbito en el que se desarrolla el producto ha cambiado y hay que adaptar ciertas partes del proyecto, etc. La incertidumbre es un elemento muy presente, y las metodologías ágiles la abraza. Conviven con el cambio inevitable estableciendo unas bases menos sólidas pero más flexibles. Esta forma de trabajar se enunció oficialmente por primera vez en 2001, en el documento de **Manifiesto Ágil**[1]. Esta forma de organizar proyectos se fundamenta en cuatro puntos, extraidos del libro *Métodos Ágiles y Scrum* [2]:

- **Valorar a individuos y sus interacciones**, frente a procesos y herramientas. Aunque todas las ayudas para desarrollar un trabajo son importantes,

[1] El Manifiesto Ágil se puede consultar en esta web oficial. Hay una versión en español en la misma web. https://agilemanifesto.org/

nada sustituye a las personas, a las que hay que dar toda la importancia y poner en primer plano.

- **Valorar más el software (producto) que funciona**, que una documentación exhaustiva. Porque había llegado un punto en el que documentar el trabajo había alcanzado tanta importancia como el objeto del trabajo: el producto.
- **Valorar más la colaboración con el cliente** que la negociación de un contrato. La forma más productiva de sacar adelante un trabajo es establecer un marco de confianza y colaboración con quien nos lo encarga. Sin embargo se estaba poniendo el foco en cerrar un contrato atado que sirviera ante todo como una herramienta de protección, como si el cliente y equipo fueran dos partes enfrentadas, cuando en realidad comparten objetivos e intereses.
- **Valorar más la respuesta al cambio** que el seguimiento de un plan. Se trata de apreciar la incertidumbre como un componente básico del trabajo, por lo que la adaptación y la flexibilidad se convierten en virtudes y no en amenazas. El seguimiento ciego de un plan lleva, salvo contadas excepciones, al fracaso si no se puede corregir la dirección ante los inevitables cambios que van surgiendo.

Este típo de metodologías intentan aplicar un cambio de paradigma: mientras que tradicionalmente se fijaban unos requisitos a partir de los cuales se estimaban los recursos y plazos que se requerirían para su terminación, la forma ágil sería fijar unos recursos y fechas disponibles en las que se disponen para el trabajo, y a partir de ahí estimar los requisitos que va a tener el producto al final. Se eligió optar por esta forma de encarar proyectos ya que en primer lugar el trabajo tenía una fecha de finalización fíja, y además no se tenía experiencia en algunos de los campos requeridos para el desarrollo, como podría ser por ejemplo la implementación de interfaces web. Se iba a requerir de un periodo de formación paralela al desarrollo, lo cual con total probabilidad cambiará la visión del desarrollador sobre la estimación de los costes y tiempo del proyecto. Sería contraproducente espablecer desde un inicio unos requisitos fijos si se prevé que van a cambiar, y las metodologías ágiles se ajustaban perfectamente a este escenario.

4.2. Temporización

4.3. Seguimiento del desarrollo

Capítulo 5

Análisis del problema

Pablo Millán Cubero

Capítulo 6

Implementación

La implementación del software se ha dividido en hitos. Estos, han sido definidos en Github y cada uno de ellos contiene un grupo de *issues* que se corresponden con las distintas mejoras que se han ido incorporando al software a lo largo de su desarrollo.

Capítulo 7

Conclusiones y trabajos futuros

Bibliografía

[1] Pictarize studio. https://pictarize.com/.

[2] Carmen Lasa Gómez Alonso Álvarez García, Rafael de las Heras del Dedo. Métodos Ágiles y SCRUM, manual imprescindible. .

[3] Express.js. Fast, unopinionated, minimalist web framework for node.js. http://expressjs.com/.

[4] Free Software Foundation. GNU General Public License. http://www.gnu.org/licenses/gpl.html.

[5] Node.js Foundation. Node.js. https://nodejs.org/es/.

[6] Google. Arcore. https://developers.google.com/ar?hl=es-419.

[7] Google. Firebase. https://firebase.google.com/?hl=es-419.

[8] Thomas Gorisse. Sceneview, 3d and ar view for android, flutter and react native working with arcore and google filament. https://sceneview.github.io/.

[9] Javier González Jiménez. Visión por computador. .