Escuela Politécnica Superior de Linares

Grado en Ingeniería de Tecnologías de Telecomunicación

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

DESARROLLO DE APLICACIÓN AR PARA VISITA CULTURAL DEL YACIMIENTO DE CÁSTULO

Alumno: **Ana María Vela Cobo**

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

Septiembre, 2023

Universidad de Jaén
Escuela Politécnica Superior de Linares
Grado en Ingeniería de Tecnologías de Telecomunicación

Trabajo Fin de Grado

DESARROLLO DE APLICACIÓN AR PARA VISITA CULTURAL DEL YACIMIENTO DE CÁSTULO

Alumno: Ana María Vela Cobo Tutor: Prof. D. Raúl Mata Campos

Septiembre, 2023

Índice

1. TABLA DE ILUSTRACIONES

2. RESUMEN

La Realidad Aumentada (RA) es una tecnología en constante crecimiento gracias a su gran adaptación y aportación a, y cada vez más, casi todos los sectores. Es por ello, que en el presente proyecto se ha logrado cumplir con los objetivos de forma satisfactoria y mostrar la relevancia de esta tecnología en el ámbito cultural permitiendo mejorar las visitas y experiencias en el Monumento Nacional de Linares, el Yacimiento Arqueológico de Cástulo, de una forma dinámica mediante la aplicación gamificada que se expone.

La aplicación de este proyecto funciona mediante el uso de métodos de geolocalización y una brújula, logrando localizar y guiar al usuario hasta los distintos marcadores donde se conseguirá activar los aumentos enfocando a través de la cámara de la aplicación. Esta aplicación está disponible para dispositivos Android que contienen giroscopio. Además, se permite la edición de las fotografías tomadas con diferentes filtros y modificaciones, y guardar y compartirlas.

Palabras clave

Realidad Aumentada, aplicación gamificada, geolocalización, ámbito cultural.

Abstract

Augmented Reality (AR) is a technology in constant growth thanks to its great adaptation and contribution, and increasingly almost all sectors. It is for this reason that in the present project has been achieved to meet the objectives satisfactorily and show the relevance of this technology in the cultural field allowing to improve visits and experiences in the Linares National Monument, the Archaeological Site of Cástulo, in a dynamic way through the gamified app that is exposed.

The application of this project works by using geolocation methods and a compass, managing to locate and guide the user to the different markers where you will be able to activate the increases focusing through the camera of the application. This app is available for Android devices that contain a gyroscope. In addition, it allows the editing of photographs taken with different filters and modifications, and save and share them.

Key words

Augmented reality, gamified app, geolocation, cultural field.

3. INTRODUCCIÓN

En los últimos años, la realidad aumentada se ha presentado en nuestras vidas como una tecnología emergente muy innovadora, la cual ofrece interactuar con los proyectos, optimizar tareas y mejorar experiencias. Se trata de exponer una versión renovada del mundo físico real, lográndose a partir de información auditiva, visual o sensorial, entregada a través de la tecnología.

La incorporación de la realidad aumentada a nuestras vidas permite que el aprendizaje sea encarecido, versátil y llamativo debido a su facilidad de comprensión y a su atracción a cualquier público. Es por ello que, con el fin de incentivar las visitas al Yacimiento Arqueológico de Cástulo, se ha introducido la realidad aumentada en la Ciudad Ibero-Romana consiguiendo presentar la vida real en algunos hitos arqueológicos de la ciudad en aquella época. Así, con la aplicación 'gamificada', se consigue vivir en primera persona cómo era pasear por esas calles al encontrar las localizaciones tras ser guiados por el juego.

En este proyecto se va a implementar una app que consigue la gamificación mediante marcadores que activan unos aumentos donde se muestran las ideas de reconstrucción discurridas y aproximadas de las ruinas del Monumento Nacional. Tras conseguir los seis aumentos y al capturar todos los escenarios, se logrará llegar al final del juego, donde se podrán compartir las imágenes tomadas y editarlas al gusto del usuario.

3.1. Antecedentes

Tras las afirmaciones sobre esta gran novedosa tecnología, lo cierto es que la realidad aumentada se presentó en nuestras vidas hace más de 100 años.

La historia de la realidad aumentada es iniciada por el escritor Frank L. Baum en 1901 cuando de forma involuntaria en su novela *The Master Key*, el protagonista cuenta con unas gafas especiales que le mostrarían unas letras sobre la cabeza de las personas, que le indicarían el carácter de cada una. Es así, cuando Baum predijo la primera idea de lo que sería la realidad aumentada años después. [1]

Es Morton Heilig en 1957, cuando trae la primera realización tecnológica de la Realidad Aumentada con el Sensorama. Fue el primer dispositivo multisensorial que incluía imágenes en tres dimensiones, amplia visión, movimiento, estímulos visuales, sonido en estéreo, olores, aire y vibraciones, que recreaban una experiencia cinematográfica al conducir una moto por Nueva York. [2]

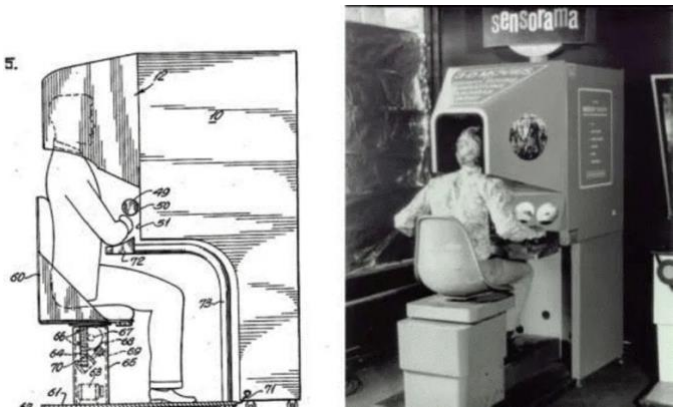

Figura 1. Sensorama

En 1966, Ivan Sutherland, un profesor de Ingeniería, y su estudiante, Bob Sproull, crearon la primera pantalla de Realidad Virtual y Realidad Aumentada que se usaría sobre la cabeza llamada Sword of Damocles. Recibe ese nombre porque al ser la pantalla tan pesada, tuvo que ser colgada del techo. Consiste en dos tubos de rayos catódicos conectados a un ordenador. El movimiento de la cabeza era completo, por lo que, al cambiar de perspectiva, el software modificaba el entorno virtual ajustándolo al real, además, era un sistema estereoscópico, que permitía seguir viendo la habitación. Al principio, tan solo permitía ver un cubo en 3D. [3]

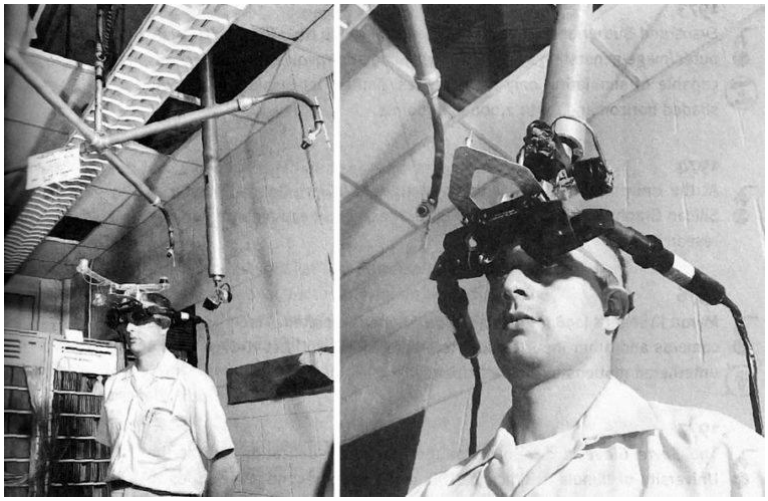

Figura 2. Sword of Damocles

Posteriormente, en 1973, Myron W. Krueger, queriendo conectar nuestro mundo real con el digital, a pesar de no usar la Realidad Aumentada, creó Videoplace, consiguiendo un sistema de reconocimiento del movimiento. Logró que ambos mundos pudiesen interactuar

a través de un lienzo por medio de cámaras de vídeo y proyectores, y las siluetas de los usuarios. [4]

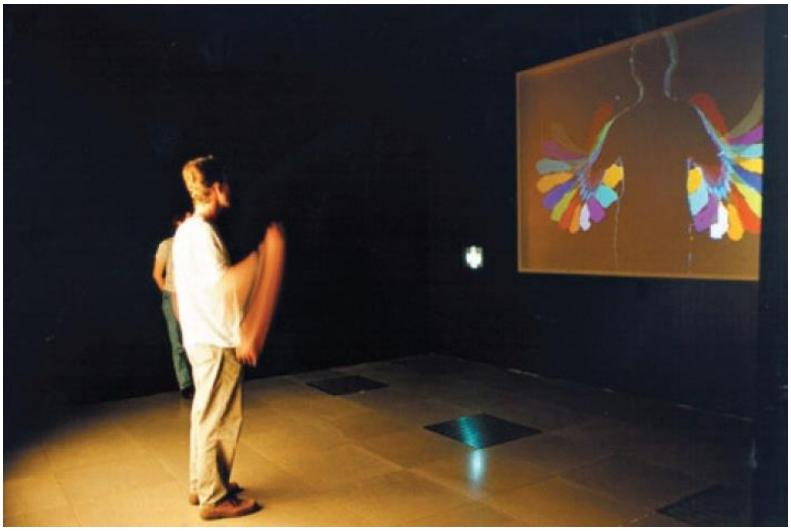

Figura 3. Videoplace

En 1979, McDonnell Douglas consiguió crear el Flight Simulator, que como su nombre indica, es un simulador de vuelo que rastrea el movimiento de los ojos del piloto y proyecta una imagen delante de ellos. [5]

Figura 4. Flight Simulator

En este mismo año, Michel Naimark, presentó su proyecto Aspen Movie Map el cual permitía la relación entre la interactividad y la virtualidad a través de un recorrido por la ciudad de Aspen. Consistía en captar las imágenes de la ciudad a partir de cuatro cámaras

situadas en el automóvil y combinarlas con los datos de posición. Los usuarios podrían así tocar los edificios y saltar sobre ellos. [6]

Figura 5. Aspen Movie Map

Es en 1990, mientras Tom Caudell crea el término de Realidad Aumentada [7], inspirándose en la maquinaria usada para reparar las configuraciones de los cables de los aviones en Boeing, en 1992, el tecnólogo Louis Rosenberg se suma a esta creación del término presentando el primer sistema de AR totalmente inmersivo llamado Virtual Fixture. Rosenberg diseñó dos robots dirigidos por un exoesqueleto sobre la cabeza del usuario, que permitiría preparar a los pilotos de la Fuerza Aérea de Estados Unidos.

Figura 6. Virtual Fixture

Además, en 1993, un grupo de científicos de la Universidad de Columbia formado por Steven Feiner, Blair MacIntyre y Dorée Seligmann, desarrollaron un prototipo denominado KARMA (Realidad Aumentada basada en el conocimiento para Asistencia de Mantenimiento), el cual, con una pantalla sobre la cabeza del usuario, consigue exponer el funcionamiento de una impresora láser a través de una proyección 3D. [6]

Figura 7. KARMA

En 1999 surgió ARToolKit, creado originalmente por el Dr. Hirokazu Kato. ARToolKit es una biblioteca software que permite crear aplicaciones de realidad aumentada. En relación con la posición de los marcadores físicos, logra calcular en tiempo real la orientación y posición de la cámara. Así podrá situar los modelos 3D sobre los marcadores reales.

Actualmente, se sigue manteniendo como código libre siendo accesible para uso no comercial. [6]

Figura 8. ARToolKit

Y, finalmente, es en el 2000 cuando se crea el primer videojuego en primera persona al aire libre de realidad aumentada por Bruce Thomas. Recibe el nombre de ARQuake y permite a los usuarios interactuar con monstruos mediante el uso de un casco con una pantalla, un GPS, un seguimiento del campo de visión conseguido con un ordenador dentro de una mochila que llevará el usuario y, será una pequeña pistola, cuya función será actuar como controlador, la que aportará la entrada de datos. [8]

Figura 9. ARQuake

Posteriormente se conseguirá presentar el primer juego de realidad aumentada para el hogar en 2007, denominado The Eye of Judgment donde en una televisión se muestra el tablero que capta una cámara y los personajes virtuales cobran vida situándose sobre las cartas reales del tablero del juego. [6]

Así, han podido crear gran cantidad de juegos de realidad aumentada hasta llegar al número 1, Pokémon Go. Este videojuego llevó la realidad aumentada a una audiencia masiva, donde, para la gran mayoría, será su primera vez en descubrir esta tecnología y, por tanto, será el nacimiento de intereses y curiosidades por la creación de nuevas aplicaciones de realidad aumentada en el futuro.

3.2. Aplicaciones de gamificación basadas en visitas culturales

- Aplicación de AR para integración de personajes históricos en visitas culturales

Esta aplicación se creó por una alumna de la Escuela Politécnica Superior de Linares para su Trabajo Fin de Grado. En esta app se hace uso de la geolocalización y consigue una gamificación para una visita cultural sobre los monumentos de algunos personajes destacados de la ciudad de Jaén, donde se muestra información de cada uno de ellos tras ser identificados por la cámara del dispositivo.

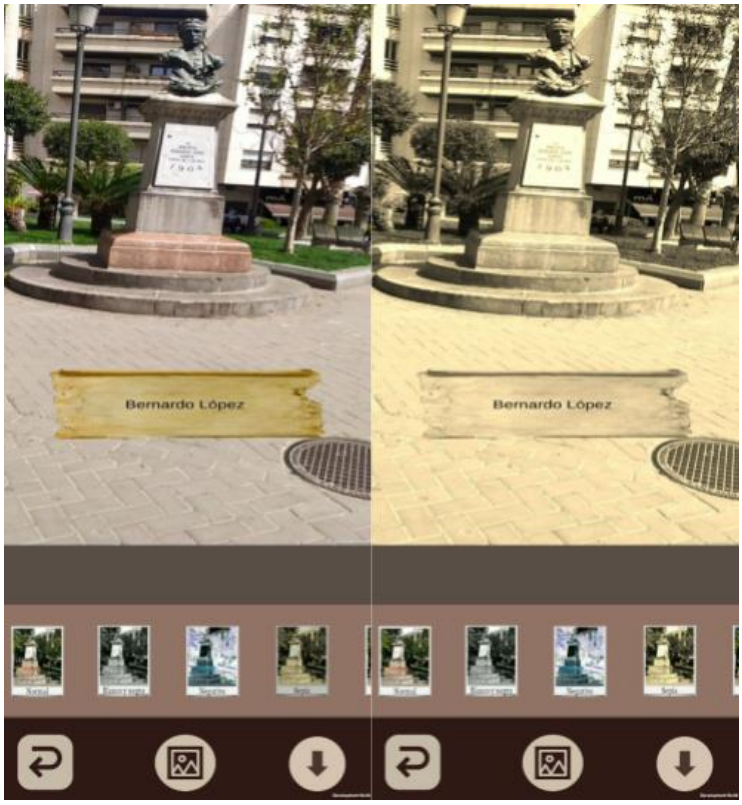

Figura 10. App de visita cultural en Jaén

Una de las diferencias que se han tratado en este proyecto con la aplicación anteriormente contada son la incorporación de personajes históricos 3D, como los soldados que aparecen luchando en el último aumento de la aplicación creada en este proyecto, donde además incluye el movimiento de estos y audio. También, en el primer aumento, se ha añadido vapor de agua con movimiento. [9]

- Guideo

Guideo se trata de una aplicación pionera en el uso de realidad aumentada en el ámbito turístico andaluz que, tras detectar la posición del usuario por la geolocalización, muestra a lo largo de la ruta monumentos y personajes históricos que existieron en ese lugar, además de añadir contenido multimedia sobre estos como textos y vídeos que exponen conversaciones y costumbres, haciendo más interesantes las historias. Los autores de la app quisieron crear un viaje a otra realidad a través de una pantalla. [10]

Figura 11. App Guideo

4. OBJETIVOS E HIPÓTESIS

El objetivo principal del proyecto es la creación de una aplicación móvil que muestre los aumentos de los diferentes hitos arqueológicos del yacimiento de Cástulo, basándose en la realidad de la época de la ciudad ibero-romana. Además, con este trabajo fin de grado, se consigue adquirir nuevos conocimientos en técnicas avanzadas de procesamiento de imagen y vídeo, y de realidad aumentada.

Se pretende desarrollar una herramienta para integrar, en visitas culturales al yacimiento de Cástulo, tanto información multimedia como presentaciones de personajes históricos relacionados con el yacimiento y recreaciones de la vida en los hitos arqueológicos descubiertos de la ciudad, mediante AR. Tras tener la imagen captada con el dispositivo, se busca crear una sensación de inmersión más divertida e interesante, donde se consiga recrear, en forma de aumento de la realidad, información al usuario de la aplicación.

A partir de la utilización de diferentes marcadores, podrás componerse diferentes aumentos con contenidos multimedia para la recreación, el suministro de información cultural, la realización de fotografías, etc.

La aplicación debe proporcionar una interfaz cómoda, intuitiva y sencilla de utilizar para facilitar su uso con finalidad de ocio.

Tras los objetivos generales del proyecto, se puntualizan algunos de los objetivos específicos:

- Creación de una aplicación gamificada en el ámbito cultural sobre el Yacimiento Arqueológico de Linares.
- Se deben definir marcadores que sean detectados por gps, con la ayuda de las pistas adecuadas que permitan encontrar la ubicación de estos.
- Se debe detectar mediante la propia cámara de la aplicación los diferentes marcadores, situando el dispositivo de frente a estos.
- La aplicación tiene que activar correctamente todos los aumentos definidos hasta completar los 6 marcadores definidos.
- Se debe actualizar de forma adecuada la información hacia el siguiente punto, así como las pistas de la dirección que se debe seguir para llegar a este.
- La aplicación tiene que permitir capturar los aumentos y la edición de estos mediante filtros, y modificaciones del brillo, contraste, saturación y tono.
- Se debe conseguir guardar en el dispositivo y compartir en las redes sociales las imágenes capturadas en el juego
- La aplicación debe ser factible y beneficiosa, logrando mejorar la experiencia de los usuarios

5. ESTADO DEL ARTE

La realidad aumentada da la posibilidad a los usuarios de ver el mundo real al cual se le incorporan objetos virtuales conviviendo ambos en el mismo espacio. Por lo tanto, complementa la realidad y la percepción de los usuarios hacia ella sin cambiarla totalmente interaccionando con el mundo real. Según Tom Caudell, "la realidad aumentada es un sistema que combina imágenes generadas por un ordenador con vistas del mundo real, lo que permite al usuario ver tanto el mundo real como los objetos virtuales en el mismo entorno" [7].

Actualmente, la realidad aumentada se encuentra en un momento culminante de su progreso, el cual se verá enriquecido en los siguientes años gracias al gran auge que está atravesando en la sociedad.

Esta tecnología se ha visto extendida a ámbitos tales como son la educación, la medicina y la ingeniería y el comercio, dentro un amplio campo, debido a aportar nuevas expectativas y aplicaciones a estos sectores en constante crecimiento a través de estudios inspirados en nuevas oportunidades y herramientas que cada rama puede ofrecer. [11]

Destacando el contenido principal de este proyecto, el patrimonio cultural ha sido uno de los ámbitos más favorecidos por esta tecnología. El uso de las nuevas tecnologías en el patrimonio cultural es una herramienta eficaz y considerable para su conocimiento y avance.

Se puede estimar la realidad aumentada como una opción que contribuye a una gran proyección al adecuarse a las necesidades, dando lugar a importantes proyectos cuyo principal objetivo es mejorar el acceso y la difusión del patrimonio a través de medios didácticos y atrayentes. [12]

5.1. Tipos de realidad aumentada

5.1.1. Realidad aumentada basada en marcadores

La realidad aumentada basada en marcadores es la más sencilla y simple de desarrollar y, tal y como indica su nombre, utiliza marcadores o imágenes que son elementos inteligibles, que no expresan nada para el ojo humano, pero que al ser reconocidos, leídos y escaneados por la cámara de un dispositivo, permite activar un aumento y da acceso al contenido virtual a través de una aplicación u página web, donde se calculará la dirección y posición del marcador para poder colocar correctamente la información que se quiere mostrar en la pantalla.

A día de hoy, estos elementos inteligibles pueden ser los códigos QR, los cuales nos acompañan en nuestro día a día y estamos acostumbrados a acceder a su contenido que puede ser desde un texto hasta una animación o vídeo.

En resumen, en la realidad aumentada basada en marcadores se necesita de un elemento visual externo para poder detectar el lugar en el cual activar el aumento. [11]

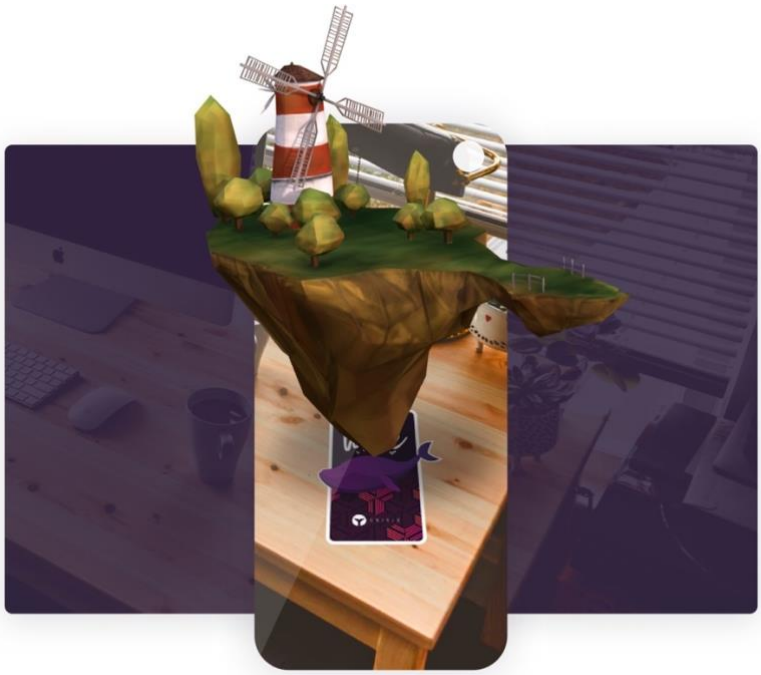

Figura 12. AR basada en marcadores

5.1.2. Realidad aumentada sin marcadores [14]:

Realidad aumentada basada en proyección

En la realidad aumentada basada en proyección, se utiliza una luz que proyecta sobre las superficie u objetos físicos, las imágenes virtuales, por lo tanto, se mantiene inmóvil, fijo en un lugar. La interacción ocurre al tocar sobre la superficie, así el usuario no necesita ningún dispositivo, tan solo el proyector.

Esto se utiliza para crear profundidad, ilusiones sobre la ubicación y la forma de los objetos.

Figura 13. Ejemplo de AR basada en proyección

Realidad aumentada basada en la ubicación

En este tipo se usan técnicas de geolocalización y se logra captar el movimiento de la persona con el fin de mostrar la proximidad del activo digital, lo cual se consigue al recibir los datos a través del GPS, brújula, acelerómetros, etc.

Así, la información y datos virtuales, se encontrarán en ciertas ubicaciones al coincidir la localización del dispositivo del usuario con la de estos.

Hay dos formas de presentar la información de AR, estas son las descritas a continuación:

- Información en tiempo real con GPS, donde mediante la cámara se consigue obtener los datos digitales al captar su posición GPS correcta.
- Información estática que se muestra al llegar a una determinada ubicación, lo que la hace más fiable y accesible, pues se obtiene la información con solo llegar a la localización específica.

Un ejemplo muy destacado es la aplicación gamificada de Pokémon Go, la cual, como bien detalla esta tecnología, introduce elementos digitales en el mundo real a través de la geolocalización.

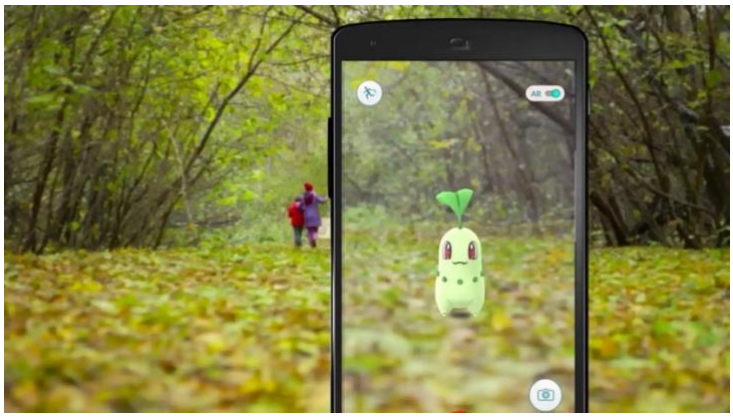

Figura 14. Pokémon Go

Realidad aumentada basada en la superposición

Este tipo de realidad aumentada se basa en reconocer objetos con el fin de sustituir con esta tecnología una parte de ellos o en su totalidad. Por tanto, su principal función es la del reconocimiento de objetos, ya que, sin identificarlos, no se podría conseguir la superposición con otro digital.

Un ejemplo del uso de esta tecnología es en medicina, donde los doctores conseguirían superponer los datos vistos en una radiografía sobre la imagen real, dando lugar a tener mayor información del posible daño.

Otro ejemplo más extendido es el de su uso en los filtros de las redes sociales como Instagram, Facebook y Snapchat.

5.2. Aplicaciones/Usos de la realidad aumentada

El avance de esta tecnología a lo largo de los años se ha visto encarecido, desde ser recibido como un término de ciencia-ficción hasta convertirse en una de las bases científicas más extendidas y con gran apogeo de la época.

Esta tecnología, gracias a las posibilidades y potencial que presenta, se ha podido introducir en una gran variedad de áreas de conocimiento. Algunas de aplicaciones de la realidad aumentada se exponen a continuación [12]:

Educación

Tras un estudio comparativo entre el aprendizaje con y sin realidad aumentada en las aulas, se determinó que el nivel de rendimiento y atención de los alumnos es más elevado en un ambiente de aprendizaje con realidad aumentada que en el de una clase basada en explicaciones mediante diapositivas. Además, se demostró que esta tecnología incrementaba

la motivación en el aula al incluir estos recursos sencillos y didácticos, convirtiéndolo en un gran medio pedagógico. E incluso, numerosas investigaciones aseguran que tiene gran eficacia en personas con Trastornos del Espectro Autista, debido a que potencia la atención prestada en la actividad, la reflexión, la memoria, etc. [15]

Enseñar haciendo uso de objetos 3D es una forma de adecuada de captar los conocimientos al determinar un vínculo con la realidad. Por ello, uno de los ejemplos de realidad aumentada en educación más actuales es la aplicación JigSpace, donde los alumnos pueden aprender cómo es el cuerpo humano o las capas de la tierra. [16]

Figura 15. JigSpace

Salud

La realidad aumentada está siendo aplicada en el ámbito de la medicina con mayor frecuencia tanto en las operaciones como en tratamientos. Los profesionales exponen que esta tecnología podría ayudar a progresar en la cirugía.

Es por ello que la medicina está teniendo mejoras notables. Se están equipando los hospitales y centros médicos con equipos actuales y avanzados que consiguen diagnósticos unívocos, lo cual permite comenzar de forma segura y eficaz nuevos tratamientos, consiguiendo abrir así horizontes con grandes ganancias a expertos como cirujanos, psiquiatras, etc. [17]

Hay multitud de ejemplos desarrollados en el ámbito de la salud. Uno de ellos es AccuVein. Esta es una aplicación, la cual, según los propios médicos "proyecta una imagen de los vasos sanguíneos sobre la propia piel del paciente. De este modo, es más fácil localizar las venas, lo que agiliza el proceso y mejora la satisfacción de los pacientes." [18].

Figura 16. AccuVein

Otro ejemplo destacado en la rama de la cirugía es el uso de gafas de realidad aumentada durante las intervenciones quirúrgicas que consigue ofrecer a los cirujanos el acceso al historial del paciente, y les permite consultar la información que necesiten durante la cirugía. Así se alcanza una mejora de los resultados y un gran ahorro del tiempo en las operaciones.

Turismo cultural

La industria del turismo es uno de los ámbitos más destacados de la economía global estando en continuo desarrollo y crecimiento. Este sector ha experimentado un gran avance debido a las nuevas tecnologías en los últimos años, consiguiendo experiencias acertadas que consiguen un mayor nivel de agrado para los usuarios.

Estas experiencias han sido favorecidas a través de la realidad aumentada, sobre todo, por el impulso gracias a la revolución digital de la industria y de la sociedad con los *smartphones*, el masivo y sencillo acceso a Internet y la geolocalización en los viajes.
Por tanto, el turismo con realidad aumentada mejora las experiencias consiguiendo que sean más auténticas, inmersivas e interactivas, accediendo a mejores formas de viajar.

Algunos ejemplos de aplicaciones en el ámbito del turismo se encuentran las guías y visitas virtuales por ciudades, información sobre monumentos característicos, traducción de textos, etc. Además de la posibilidad de introducir la realidad aumentada en los museos. [12]

Moda

Desde hace unos años, ha sido reconocida como una tendencia popular para las marcas que incluyan la realidad aumentada en el marketing de la empresa, consiguiendo así que esta tecnología modifique y favorezca la forma en la que se crea, se produce y se vende, por lo que, ninguna marca podrá quedar excluida de la realidad aumentada, ya que esta se encuentra sosteniendo un gran impacto en el sector de la moda, tanto en los desfiles como en las tiendas.

Algunas de las aplicaciones de la realidad aumentada en la moda son mostrar información al apuntar la etiqueta de la prenda, probar la ropa de forma virtual desde casa, ofrecer la visualización de desfiles 360º, etc. [19]

Un ejemplo real de aplicación se encuentra en la marca de lujo Dior que ofrecer a sus clientes una experiencia innovadora para que puedan probarse y consultar de forma virtual en su página web una amplia gama de sus productos de maquillaje, evitando que tengan que asistir de forma presencial a las tiendas.

Figura 17. Aplicación de AR en Dior

Arquitectura

En este sector de construcción, la realidad aumentada se basa en mostrar modelos virtuales 3D de infraestructuras o edificios. El objetivo principal de esta tecnología es incrementar la información que se puede ver mediante análisis de materiales, detalles para la edificación, o datos técnicos, y poder visualizarla con un teléfono móvil.

La realidad aumentada en arquitectura nos da acceso a observar cuál será el final de la construcción, agregando elementos virtualmente que se incluirán posteriormente en la vida real. Así se consigue un importante ahorro monetario y del tiempo empleado en la obra, y a su vez mejora las garantías ofrecidas a los constructores y a los clientes. [11]

Uno de los ejemplos más reconocidos en este ámbito es AR Sketchwalk. Esta aplicación logra que podamos acceder a una vista más cercana a la futura construcción gracias a las imágenes y los diseños que crea. También se permite la interacción en tiempo real de los proyectos, haciéndolo mucho más atractivo y beneficioso a la hora de presentarlos. [20]

Figura 18. AR Sketchwalk

Prensa

En estos últimos años, la prensa escrita ha introducido en sus páginas la realidad aumentada, obteniendo la información extendida de las noticias por medio de códigos QR o aplicaciones. Estos archivos obtenidos a través de esta tecnología pueden tratarse de imágenes, audios, podcasts, biografías de los escritores, noticias con elevada carga de información, vídeos, etc. Así, se ha conseguido extraer un gran rendimiento a esta tecnología para poder incrementar las visitas a la prensa escrita. [21]

Una aplicación a destacar de realidad aumentada en el sector de la prensa es la usada en la revista Esquire, donde se publicó un número con un pequeño código QR que, al capturarlo con el teléfono móvil, aparecería el hombre de la portada en 3D y el texto se movería de forma aleatoria alrededor de él.

Figura 19. Aplicación de AR en Esquire

Publicidad

Una de las aplicaciones más claras del empleo de la realidad aumentada es la publicidad. Ha creado una amplia red de usos, incluyendo folletos, campañas de televisión y carteles publicitarios. El acceso a los contenidos que constan de información privilegiada para los clientes, se facilita por un código QR o una aplicación gratuita, consiguiendo de esta forma datos muy útiles para los clientes como la venta de productos, el estudio de los ingredientes y del proceso de fabricación, etc.

Algunos de los ejemplos más conocidos de realidad aumentada en publicidad fueron la campaña que hizo Burger King donde los clientes al colocarse en frente de un anuncio por la calle, les permitía abrir la aplicación que, al detectar la imagen, estallaba en llamas y mostraban la novedosa hamburguesa del momento. [22]

Figura 20. Publicidad de Burger King

5.3. Procesos de la Realidad Aumentada [23]

A continuación, se van a exponer las diferentes acciones fundamentales para poder llevar a cabo el proceso de la realidad aumentada.

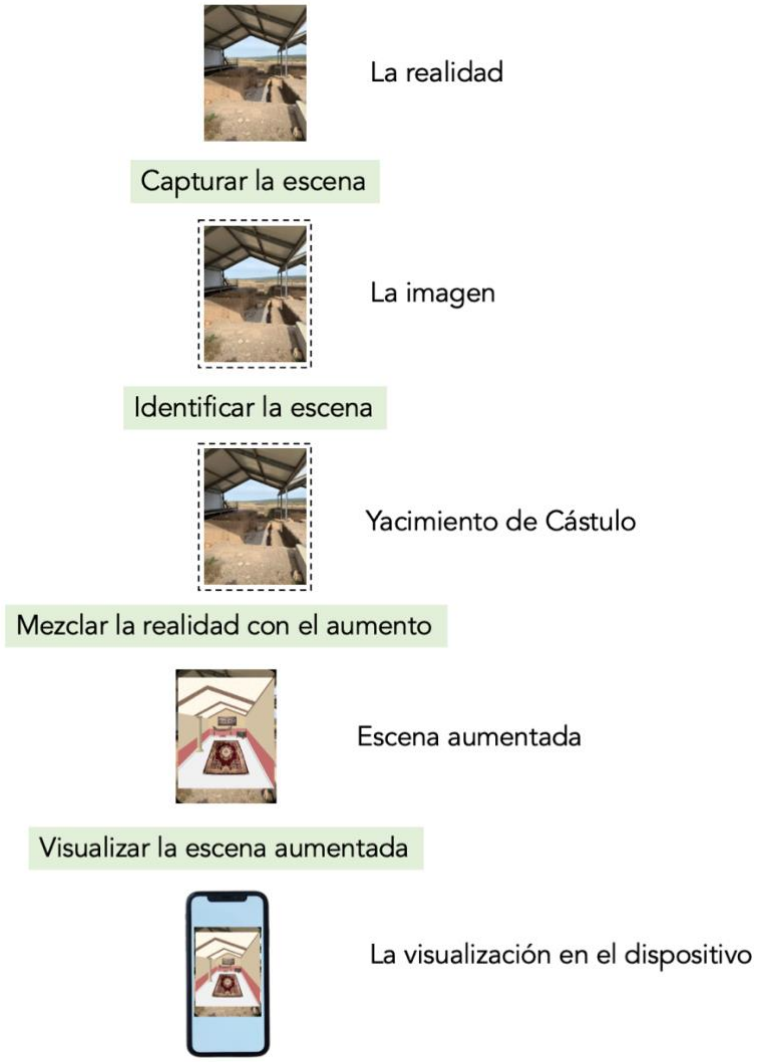

Figura 21. Procesos de la realidad aumentada

Capturar la escena

Primero, la realidad que debe aumentarse se captura a través de un dispositivo con cámara u cualquier otro mecanismo. El factor más relevante para el uso de la realidad aumentada en cualquier ámbito, es conocer la escena para poder procesarla y aumentarla.

Identificar la escena

Segundo, se tiene que escanear la realidad capturada en el paso anterior para definir la posición exacta donde se debe activar el contenido virtual. Esta posición se va a identificar mediante marcadores, los cuales son reconocidos por la cámara y activan el aumento, o mediante tecnologías de geolocalización como GPS, sensores, bluetooth, etc.

Mezclar la realidad con el aumento

Tercero, tras captar e identificar la escena tanto física como digital, se deberá sobreponer los datos e información digitales que se quieren añadir sobre la escena real. Toda esta información que se quiera añadir será de tipo texto, imagen, vídeo, audio, con la que se podrá interactuar.

Visualizar la escena aumentada

Por último, el sistema de realidad aumentada produce una imagen que es la mezcla de la escena real y del contenido virtual, y lo reproduce por pantalla.

6. DESARROLLO

En esta sección, se tratan una serie de aspectos fundamentales para la creación de la aplicación. Se exponen las distintas herramientas empleadas para su creación, su mapa de navegación y el diagrama de flujo que fijan las bases de su estructura. También, se analiza el diseño de la aplicación, detallando la disposición de sus menús y sus funcionalidades. Por último, se destacan las funciones clave de la aplicación.

6.1. Herramientas de trabajo

Durante el proceso de desarrollar una aplicación, la elección y el uso apropiado de las herramientas de trabajo desempeñan un papel fundamental. Este apartado se centra en las distintas herramientas de trabajo que han sido empleadas para el desarrollo de la aplicación, explicando tanto el software utilizado como las tecnologías implementadas y el porqué del uso de éstas.

6.1.1. Software

El proceso de crear y diseñar la aplicación ha sido mucho más sencillo gracias a distintos programas. Estos programas han ayudado a crear la interfaz y el funcionamiento de la aplicación. En esta sección se explica qué programas se han usado para transformar las ideas en componentes interactivos dentro de la aplicación.

Unity editor (2021.3.20f1)

Figura 22. Unity Editor

Aplicación informática que muestra una interfaz gráfica para desarrollar juegos y aplicaciones, como en este caso. Permite diseñar, crear y depurar en tiempo real nuestra aplicación, gracias a todas las herramientas y funcionalidades que contiene, y manejar todos los archivos necesarios para el desarrollo, tales como imágenes, aumentos, scripts y módulos.

Photoshop (14.0)

Figura 23. Photoshop

Aplicación informática que permite manipular, editar y crear imágenes. Se ha usado para crear o editar algunos de los botones, elementos gráficos y aumentos de la aplicación.

Canva

Figura 24. Canva

Plataforma en línea que permite crear diseños gráficos. Se ha usado para crear algunos elementos gráficos de la aplicación, así como para diseñar los distintos menús.

Good notes (5.9.127)

Figura 25. App Good Notes

Aplicación que permite tomar notas y crear dibujos a mano. Se ha usado para dibujar los distintos aumentos de cada marcador.

Microsoft Visual Studio (16.11.24)

Figura 26. Microsoft Visual Studio

Aplicación informática que permite desarrollar aplicaciones de software mediante lenguajes de programación. Se ha usado para crear los distintos scripts de la aplicación, habiendo sido programados en C#.

6.1.2. Tecnologías

La aplicación se ha construido usando diferentes tecnologías. Cada una de estas tecnologías han sido una parte importante para lograr los objetivos de la aplicación, haciendo que las funciones más importantes de la aplicación se ejecuten sin problemas. En esta sección se explica qué tecnologías se han usado hacer que la aplicación funcione.

C#

Figura 27. Lenguaje C#

Lenguaje de programación orientado a objetos. Se ha usado para programar los distintos scripts de la aplicación.

Vuforia Engine

Figura 28. Vuforia Engine

Kit de desarrollo de software de AR, que permite la integración con Unity. Permite crear aplicaciones con AR utilizando la cámara y los sensores de los dispositivos, en nuestro caso una tablet Android, para detectar y seguir los marcadores, y superponer el aumento deseado.

Librería UnityEngine

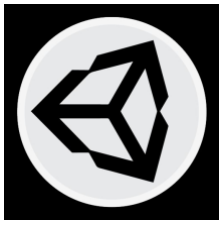

Figura 29. Librería UnityEngine

Librería de clases y funciones de Unity. Se ha usado para gestionar y programar distintas funciones de las escenas, objetos, entradas de texto, acciones del usuario y sonidos de la aplicación.

Librería DOTween [24]

Figura 30. Librería DOTween

Librería de clases y funciones de animación para Unity. Se ha usado para programar transiciones y animar objetos de la aplicación.

Librería NativeGallery [25]

Figura 31. Librería NativeGallery

Librería de clases y funciones para Unity que permite interactuar con la galería de los dispositivos. Se ha usado para acceder y guardar las distintas imágenes que tome el usuario.

Librería NativeShare [26]

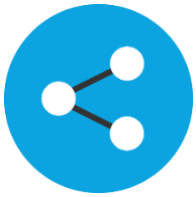

Figura 32. Librería NativeShare

Librería de clases y funciones para Unity que permite compartir contenido desde la aplicación mediante las opciones que tenga el dispositivo desde donde se ejecuta. Se ha usado para compartir las imágenes que tome y edite el usuario.

APIs de Android

Figura 33. APIs de Android

Conjunto de funciones de Unity para controlar funcionalidades del sistema Android. Se ha usado para obtener permisos, acceder a capacidades de GPS [27], giroscopio, almacenamiento y cámara del dispositivo.

6.2. Implementación de la aplicación

6.2.1. Mapa de navegación

En el presente apartado, se realiza un análisis del mapa de navegación de la aplicación, donde se expone la estructura y disposición de sus diferentes menús y submenús. De esta forma permite comprender cómo los usuarios acceden y navegan a través de las pantallas y funcionalidades que tiene la aplicación.

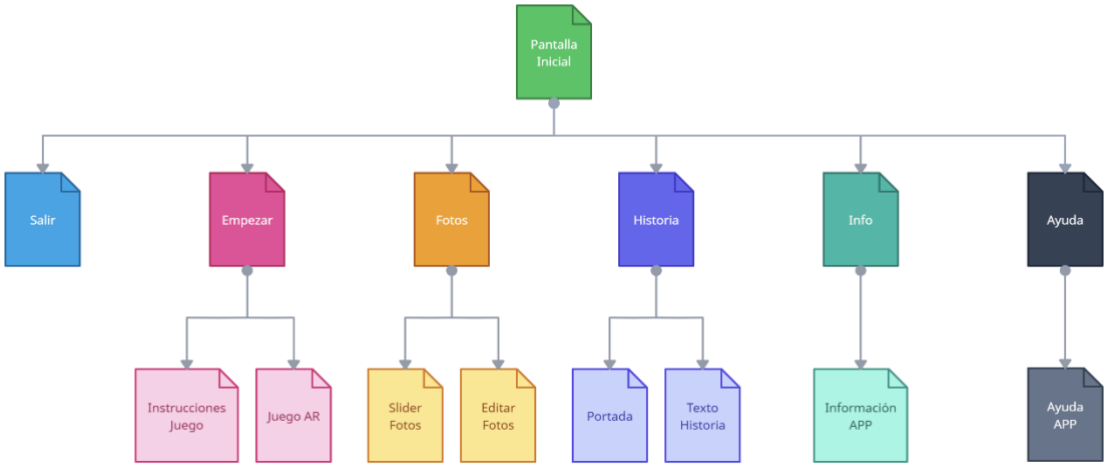

Figura 34. Mapa de navegación de la aplicación

Como se observa en el mapa, la aplicación se compone de una pantalla principal con cuatro módulos con los cuales el usuario puede interactuar con ellos.

- Salida APP: Permite al usuario cerrar la aplicación.
- Inicio Juego: Permite al usuario empezar el juego de AR de Cástulo, el cual contiene una pantalla de instrucciones y otra donde ya debe interactuar con el entorno de Cástulo.
- Fotos: Permite al usuario acceder a la galería donde podrá visualizar las imágenes tomadas en el juego y por otro lado editarlas en la pantalla de edición.
- Historia: Permite al usuario leer un breve resumen de la historia de Cástulo, creándole la sensación de que está leyendo un libro.
- Info: Permite al usuario conocer la información de la aplicación y de la autora.
- Ayuda: Permite al usuario entender el significado de cada botón que aparece en la aplicación.

6.2.2. Diagrama de flujo

A continuación, se presenta un detallado diagrama de flujo que expone de manera visual y sistemática todas las funcionalidades y acciones disponibles en la aplicación. Este diagrama no solo muestra las distintas opciones accesibles para los usuarios, sino que también enseña el comportamiento de los distintos botones y elementos de la interfaz. A través de ese diagrama se busca dar una idea de cómo los usuarios pueden interactuar con la aplicación para así sacarle el máximo partido a ésta. Más adelante se descompondrá para explicar en detalle cada una de sus acciones y funcionalidades.

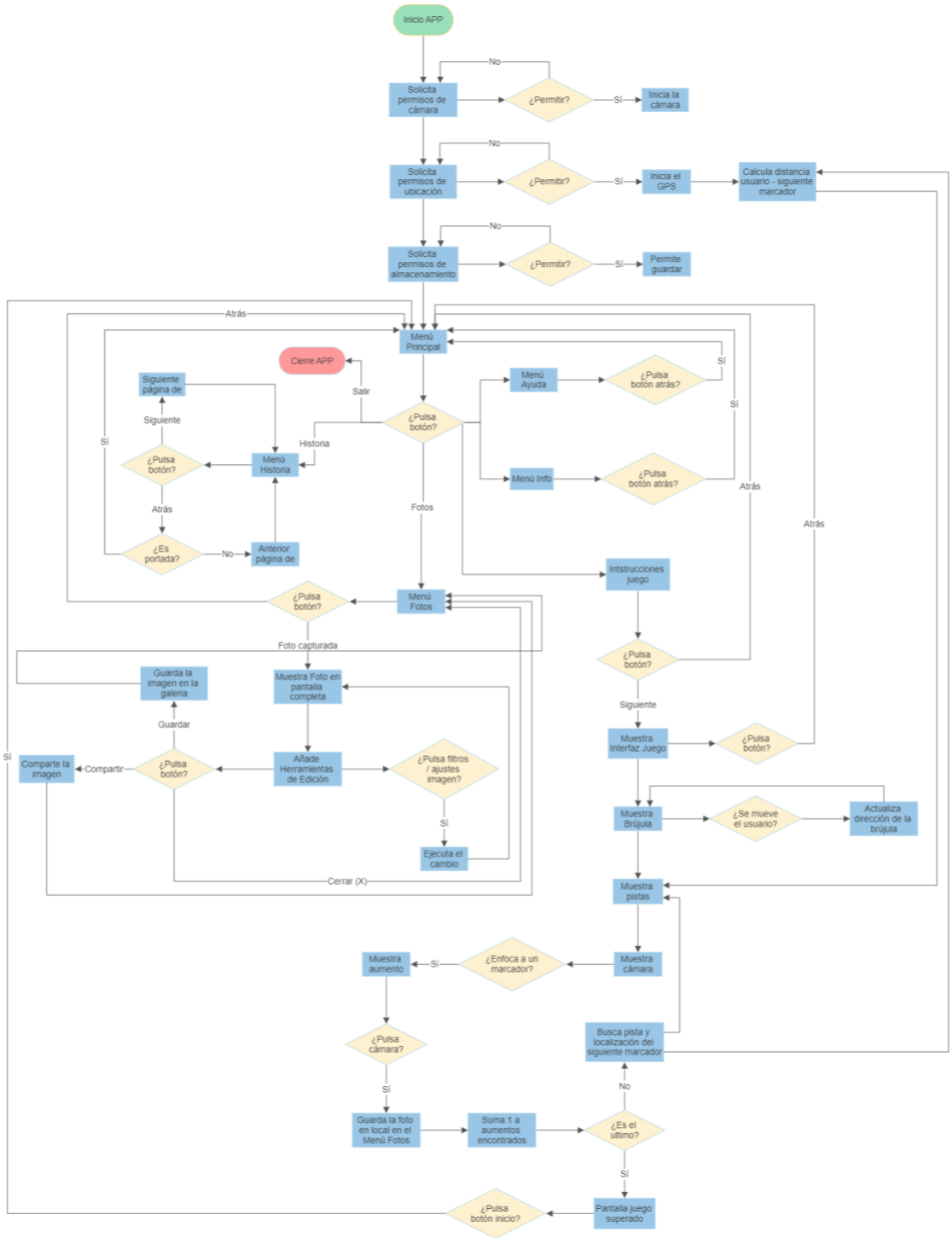

Figura 35. Diagrama de flujo de la aplicación

6.3. Diseño de la aplicación

La interfaz y los menús de la aplicación se han diseñado con Canva. A partir de este diseño, se han ido añadiendo los diferentes elementos de la UI (User Interface) y se ha programado el comportamiento de los botones, del menú, etc (acciones que realiza y transiciones), y por último las distintas programaciones específicas de cada menú, com por ejemplo la brújula, la captura de la pantalla, acciones a realizar tras el reconocimiento del Image Target, la edición de fotos, etc.

Las transiciones se han programado con la librería DOTween, usando su sintaxis (manual de DOTween en su página web). Y las programaciones restantes se han llevado a cabo en el IDE Visual Studio y en lenguaje C#.

El código de a continuación hace referencia a la inicialización de los distintos menús y botones dentro del script principal de la aplicación llamado “GameManager”.

```
public class GameManager : MonoBehaviour
{
    // Declaracion de objetos
    public event Action OnMenuPrincipal;
    public event Action OnGaleria;
    public event Action OnEmpezar;
    public event Action OnHistoria;
    public event Action OnInfo;
    public event Action OnAyuda;
    public event Action OnSiguienteTextoHistoria;
    public event Action OnAnteriorTextoHistoria;
    public event Action OnSiguienteEmpezar;
    public event Action OnAtrasEmpezar;

    public static GameManager instance;

    // Start se llama antes de la primera actualizacion de la app
    void Start()
    {
        MenuPrincipal(); // cuando la aplicacion inicie, llame al evento MenuPrincipal
    }

    public void MenuPrincipal()
    {
        OnMenuPrincipal?.Invoke();
        Debug.Log("Menu Principal Activada");
    }

    public void Galeria()
    {
        OnGaleria?.Invoke();
        Debug.Log("Galeria Activada");
    }

    public void Empezar()
    {
        OnEmpezar?.Invoke();
        Debug.Log("Camara Activada");
    }
```

Figura 36. Script de inicialización del UI

Una vez inicializados, en el script del UI de la app “UIManager”, se importan los distintos objetos y se establecen las acciones de cada uno de los menús y botones, en el código de a continuación se definen las distintas funciones que se explicarán en sus respectivos apartados para cada uno de los objetos declarados en el “GameManager”:

```
public class UIManager : MonoBehaviour
{
    // Se importan los objetos del GameManager
    [SerializeField] private GameObject MenuPrincipal;
    [SerializeField] private GameObject Galeria;
    [SerializeField] private GameObject Empezar;
    [SerializeField] private GameObject Info;
    [SerializeField] private GameObject Ayuda;
    [SerializeField] private GameObject Historia;
    [SerializeField] private GameObject SiguienteTextoHistoria;
    [SerializeField] private GameObject AnteriorTextoHistoria;
    [SerializeField] private GameObject SiguienteEmpezar;
    [SerializeField] private GameObject AtrasEmpezar;

    // Se declaran las acciones de los objetos
    public RectTransform Portada;
    public RectTransform Texto1;

    int AppIniciada=0; //cuando ejecutamos el programa
    //contador pagina texto historia
    int numTextoHistoria=0; //contador pagina historia

    float transicion = 0; //transicion si o no

    int numEmpezar=0; //contador pagina empezar

    void Start()
    {
        // Cuando se haga click en los objetos que tienen las siguientes acciones, ejecutaran las funciones
        GameManager.instance.OnMenuPrincipal += ActivarMenuPrincipal;
        GameManager.instance.OnGaleria += ActivarGaleria;
        GameManager.instance.OnEmpezar += ActivarEmpezar;
        GameManager.instance.OnInfo += ActivarInfo;
        GameManager.instance.OnAyuda += ActivarAyuda;
        GameManager.instance.OnHistoria += ActivarHistoria;
        GameManager.instance.OnSiguienteTextoHistoria += ActivarSiguienteTextoHistoria;
        GameManager.instance.OnAnteriorTextoHistoria += ActivarAnteriorTextoHistoria;
        GameManager.instance.OnSiguienteEmpezar += ActivarSiguienteEmpezar;
        GameManager.instance.OnAtrasEmpezar += ActivarAtrasEmpezar;
    }
```

Figura 37. Funciones de la aplicación

6.3.1. Menú principal

En las siguientes dos figuras se puede comparar en la primera imagen, el wireframe [28] diseñado en “Canva” y, en la segunda imagen, el resultado durante la ejecución de la aplicación.

Figura 38. Diseño del wireframe 1

Extrayendo una parte del diagrama de flujo presentado anteriormente se muestran las acciones que se pueden realizar en esta pantalla, señalando en rojo el camino a seguir:

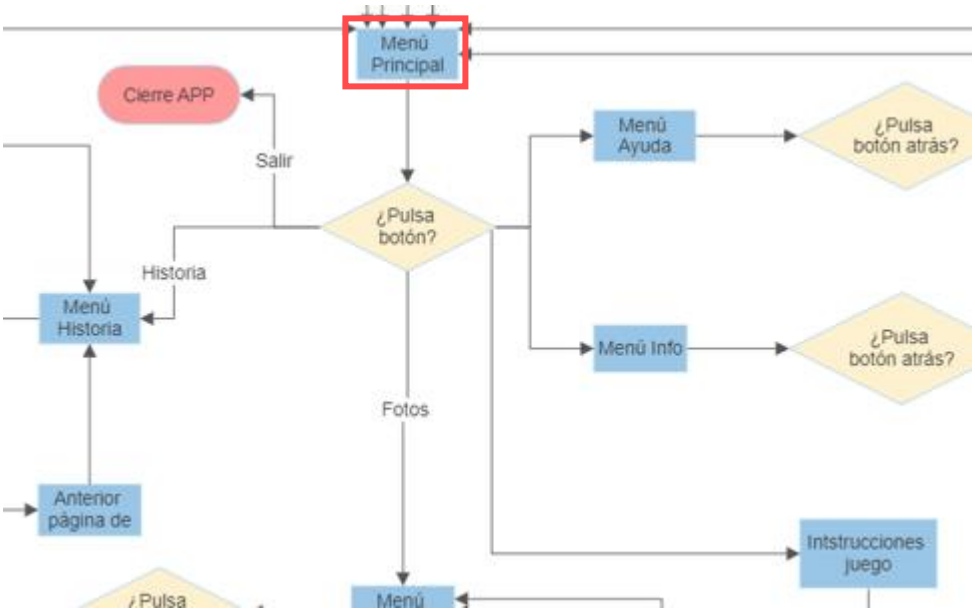

Figura 39. Zoom del diagrama de flujo 1

Primero se ha realizado un estudio de los principales apartados que debe tener la aplicación y se han agrupado en el menú principal junto a otros dos botones, uno el de cerrar la aplicación y otro el de ayuda.

Los principales apartados constan de un botón de “Inicio” para iniciar el minijuego de realidad aumentada, otro botón de “Fotos” para consultar las fotos que se han tomado en el minijuego, otro botón de “Historia” para hacer un breve resumen de la historia de Cástulo y el último botón de “Info” donde se muestra la información del proyecto.

Para crear cada uno de los botones se ha añadido el elemento “Button” de “UI” (User Inteface), y se le han añadido sus respectivos logos, convirtiéndolos primero en un “Sprite” (imagen digitalizada), y textos con el elemento “Text” de “UI”.

Al pulsar sobre cada uno de los botones excluyendo el de “Salir”, se ha programado una transición en la que se oculta el menú principal y se muestra la pantalla correspondiente al botón pulsado, en el caso de “Salir” se cierra la aplicación.

El siguiente código hace referencia a las distintas acciones y transiciones del Menú Principal, primero se controla si es la primera vez que se entra en la aplicación para mostrar todos los objetos del menú sin transición, así mismo si no es la primera vez no habrá transición. Se ocultan todos los menús dejando a la vista solo el principal.

```
// Cuando click en btn Atras/Inicio, se oculta Galeria, Empezar, Camara... (todo) y se muestra  MenuPricipal
private void ActivarMenuPrincipal()
{

    if (AppIniciada==0) { //la aplicacion es la primera vez que se abre, aparecen los botones de primeras sin transicion
        AppIniciada=1;
    } else {              // si esta iniciada aparecen con transicion
        transicion= 0.3f;
    }

    // Pongo en todos los ejes 1 para ser mostrados. Las f son los segundos de transicion para hacer la accion
    MenuPrincipal.transform.GetChild(0).transform.DOScale(new Vector3(1,1,1), 0); //Este corresponde con el Panel, nunca tiene que tener transicion, por eso es 0 siempre
    //GetChild es coger todos los hijos y cogemos del 0 a X la accion que quiero que haga
    // DOScale es una libreria para modificar las medidas del objeto
    MenuPrincipal.transform.GetChild(1).transform.DOScale(new Vector3(1,1,1), transicion);
    MenuPrincipal.transform.GetChild(2).transform.DOScale(new Vector3(1,1,1), transicion);
    MenuPrincipal.transform.GetChild(3).transform.DOScale(new Vector3(1,1,1), transicion);
    MenuPrincipal.transform.GetChild(4).transform.DOScale(new Vector3(1,1,1), transicion);
    MenuPrincipal.transform.GetChild(5).transform.DOScale(new Vector3(1,1,1), transicion);
    MenuPrincipal.transform.GetChild(6).transform.DOScale(new Vector3(1,1,1), transicion);

    //con lo de arriba me carga el menu principal, y ocultamos los demas menus (abajo)

    //Ponemos en todos los ejes 0 para ser mostrados. Las f son 0 segundos de transicion para hacer la accion
    Galeria.transform.GetChild(0).transform.DOScale(new Vector3(0,0,0), 0);
    Galeria.transform.GetChild(1).transform.DOScale(new Vector3(0,0,0), transicion);
    Galeria.transform.GetChild(2).transform.DOScale(new Vector3(0,0,0), transicion);

    Historia.transform.GetChild(0).transform.DOScale(new Vector3(0,0,0), 0);
    Historia.transform.GetChild(1).transform.DOScale(new Vector3(0,0,0), transicion);
    Historia.transform.GetChild(2).transform.DOScale(new Vector3(0,0,0), transicion);
    Historia.transform.GetChild(3).transform.DOScale(new Vector3(0,0,0), transicion);
    Historia.transform.GetChild(4).transform.DOScale(new Vector3(0,0,0), transicion);

    Info.transform.GetChild(0).transform.DOScale(new Vector3(0,0,0), 0);
    Info.transform.GetChild(1).transform.DOScale(new Vector3(0,0,0), transicion);
    Info.transform.GetChild(2).transform.DOScale(new Vector3(0,0,0), transicion);

    Ayuda.transform.GetChild(0).transform.DOScale(new Vector3(0,0,0), 0);
    Ayuda.transform.GetChild(1).transform.DOScale(new Vector3(0,0,0), transicion);
    Ayuda.transform.GetChild(2).transform.DOScale(new Vector3(0,0,0), transicion);

    Empezar.transform.GetChild(0).transform.DOScale(new Vector3(0,0,0), 0);
    Empezar.transform.GetChild(1).transform.DOScale(new Vector3(0,0,0), transicion);
    Empezar.transform.GetChild(2).transform.DOScale(new Vector3(0,0,0), transicion);
    Empezar.transform.GetChild(3).transform.DOScale(new Vector3(0,0,0), transicion);
    Empezar.transform.GetChild(4).transform.DOScale(new Vector3(0,0,0), transicion);
```

Figura 40. Función del Menú principal

6.3.2. Inicio del juego

Extrayendo una parte del diagrama de flujo presentado anteriormente se muestran las acciones que se realizan, señalando en rojo el camino a seguir:

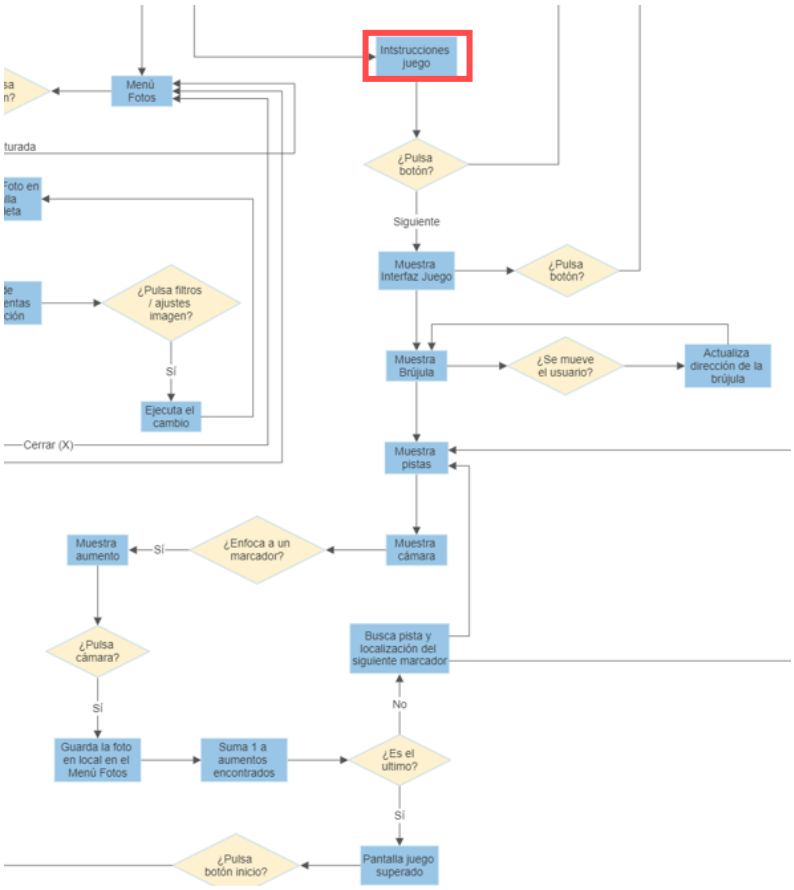

Figura 41. Zoom del diagrama de flujo 2

El siguiente código hace referencia a las distintas transiciones al pulsar el botón de Empezar del Menú Principal, donde primero se especifica que la pantalla de Empezar es la 0 para, posteriormente, controlar las acciones al pulsar el botón de Siguiente o Atrás. A continuación, se oculta el Menú Principal y se muestra la pantalla 0, en este caso el de las Instrucciones.

```
private void ActivarEmpezar()
{
    numEmpezar=0;
    //Ponemos en todos los ejes 0 para ser ocultos. Las f son los segundos de transicion para hacer la accion
    // esto oculta el menu principal
    MenuPrincipal.transform.GetChild(0).transform.DOScale(new Vector3(0,0,0), 0f);
    MenuPrincipal.transform.GetChild(1).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(2).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(3).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(4).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(5).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(6).transform.DOScale(new Vector3(0,0,0), 0.3f);

    //Ponemos en todos los ejes 1 para ser mostrados. Las f son los segundos de transicion para hacer la accion
    Empezar.transform.GetChild(0).transform.DOScale(new Vector3(1,1,1), 0f);
    Empezar.transform.GetChild(1).transform.DOScale(new Vector3(1,1,1), 0.3f);
    Empezar.transform.GetChild(2).transform.DOScale(new Vector3(1,1,1), 0.3f);
    Empezar.transform.GetChild(3).transform.DOScale(new Vector3(1,1,1), 0.3f);
}
```

Figura 42. Control de acciones de los botones del Menú principal

6.3.3. Instrucciones del juego

En las siguientes figuras se puede comparar en la primera imagen, el wireframe diseñado en "Canva" y, en la segunda imagen, el resultado durante la ejecución de la aplicación.

Figura 43. Diseño del wireframe 2

Este menú consta de un botón para ir hacia atrás y volver al menú principal, y de otro para ir hacia la siguiente pantalla que es la de juego.

Para ello se han insertado los elementos "Button" y "Textarea" de "UI". Y una función en la que según se pulse el botón de "Atrás" o "Siguiente", volverá al menú principal o iniciará el minijuego.

El siguiente código hace referencia a las distintas transiciones al pulsar el botón de Siguiente de la pantalla 0 de Empezar, donde se controla que la pantalla actual sea la 0 y a continuación se cambie a la 1, que es la del Juego, y se oculte la pantalla 0 Instrucciones y se muestre la 1 Juego.

```
private void ActivarSiguienteEmpezar()
{
    if (numEmpezar==0){
        numEmpezar++;

        //Ponemos en todos los ejes 0 para ser ocultos. Las f son los segundos de transi
        Empezar.transform.GetChild(0).transform.DOScale(new Vector3(0,0,0), 0f);
        // el boton de atras ya los estoy mostrando, por eso lo quito de aqui. ahora est
        Empezar.transform.GetChild(1).transform.DOScale(new Vector3(0,0,0), 0.3f); // bo
        Empezar.transform.GetChild(3).transform.DOScale(new Vector3(0,0,0), 0.3f);
        Empezar.transform.GetChild(4).transform.DOScale(new Vector3(1,1,1), 0.3f); // es
        Empezar.transform.GetChild(5).transform.DOScale(new Vector3(1,1,1), 0.3f); // es
        Empezar.transform.GetChild(6).transform.DOScale(new Vector3(1,1,1), 0.3f); // es
        Empezar.transform.GetChild(7).transform.DOScale(new Vector3(1,1,1), 0.3f); // es
        Empezar.transform.GetChild(8).transform.DOScale(new Vector3(1,1,1), 0.3f); // es
    }
}
```

Figura 44. Control de acciones del botón Siguiente 1

El siguiente código hace referencia a las distintas transiciones al pulsar el botón de Anterior del apartado de Empezar de la aplicación, donde se controla que, si la pantalla actual es la 0, se retorne al menú principal ejecutándose su función y sino que a la pantalla actual se le reste 1, haga las transiciones para volver a mostrar Instrucciones y ocultar el Juego.

```
private void ActivarAtrasEmpezar() //hace lo contrario de ActivarSiguienteEmpezar
{
    if (numEmpezar==0){
        ActivarMenuPrincipal();
    }

    else {
        numEmpezar--;

        //Ponemos en todos los ejes 1 para ser mostrados. Las f son los segundos de t
        Empezar.transform.GetChild(0).transform.DOScale(new Vector3(1,1,1), 0f);
        // el boton de atras ya los estoy mostrando, por eso lo quito de aqui. ahora
        Empezar.transform.GetChild(1).transform.DOScale(new Vector3(1,1,1), 0.3f); /
        Empezar.transform.GetChild(3).transform.DOScale(new Vector3(1,1,1), 0.3f);
        Empezar.transform.GetChild(4).transform.DOScale(new Vector3(0,0,0), 0.3f); /
        Empezar.transform.GetChild(5).transform.DOScale(new Vector3(0,0,0), 0.3f); /
        Empezar.transform.GetChild(6).transform.DOScale(new Vector3(0,0,0), 0.3f); /
        Empezar.transform.GetChild(7).transform.DOScale(new Vector3(0,0,0), 0.3f); /
        Empezar.transform.GetChild(8).transform.DOScale(new Vector3(0,0,0), 0.3f); /
        Empezar.transform.GetChild(9).transform.DOScale(new Vector3(0,0,0), 0.3f); /
    }
}
```

Figura 45. Control de acciones del botón Anterior

6.3.4. Juego

En las siguientes figuras se puede comparar el wireframe diseñado en “Canva” y el resultado durante la ejecución de la aplicación.

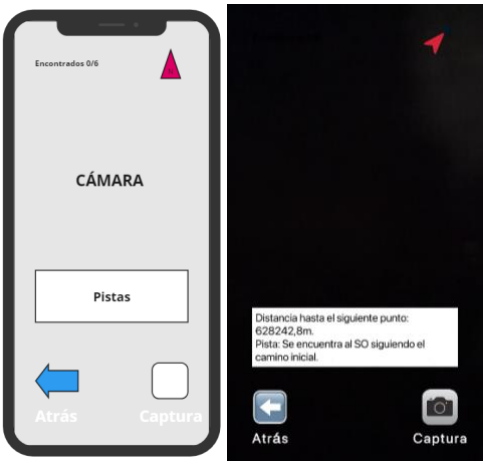

Figura 46. Diseño del wireframe 3

Este menú consta de un botón para ir hacia atrás y volver al menú principal, de otro para capturar la pantalla del juego ocultando toda la interfaz, un recuadro de pistas, una brújula, un texto informativo actualizable de los aumentos encontrados y, por último, la cámara AR.

Para ello se han insertado los elementos “Button”, “Text” e “Image” de “UI” y la “AR Camera” de “Vuforia Engine”. Y distintas funciones, una para actualizar las pistas según nuestra ubicación y el orden de los aumentos, otra para actualizar el texto informativo según los aumentos encontrados, que se explicarán a continuación, otra para actualizar la orientación de la brújula según nos vayamos moviendo, otro para capturar la pantalla y guardar la imagen temporalmente en el menú de “Fotos”, las cuales se explicarán en sus respectivos apartados y, por último, para mostrar la pantalla de instrucciones en el caso que se pulse el botón “Atrás”.

El siguiente código hace referencia a la declaración de los distintos puntos GPS de los marcadores, las pistas que se darán al usuario y el número de marcadores encontrados.

```
/*TermaPequena: 38.0367072, -3.6251989
TermaAndamio: 38.0357148, -3.6240969
TermaArcos: 38.0360245, -3.6234796
CasaMosaico: 38.0349582, -3.6247839
SalidaMosaico: 38.0348181, -3.6250022
Torre: 38.0324098, -3.6243493
*/

private List<string> imagenesDetectadas = new List<string>(); //es una lista para guardar el nombre de las imagenes que han sido detectadas
int encontrados = 0; // al iniciar la app hay cero encontrados
int indicePuntos = 0; // con 0 estoy sacando la primera latitud, primera longitud, primera pista, y corresponde con la terma pequeña
float[] puntosLatitud = new float[] { 38.0367072f, 38.0357148f, 38.0360245f, 38.0349582f, 38.0348181f, 38.0324098f }; //se declara un
array con los puntos de latitud
float[] puntosLongitud = new float[] { -3.6251989f, -3.6240969f, -3.6234796f, -3.6247839f, -3.6250022f, -3.6243493f }; //se declara un
array con los puntos de longitud
string[] pistas = new string[] {"Se encuentra al SO siguiendo el camino inicial.", "Se encuentra al SE desde el último punto.", "Se
encuentra al NE desde el último punto.", "Se encuentra al NE desde el último punto.", "Se encuentra al S desde el último punto.", "Se
encuentra al SE desde el último punto."};
```

Figura 47. Declaración de variables de ayuda para el usuario

El siguiente código hace referencia al control de los marcadores encontrados al hacer una captura. En el caso que el número de marcadores encontrados sea menor que el total, se vuelven a revisar las variables para comprobar si este número ha cambiado, y actualizar dinámicamente el mensaje de marcadores encontrados, a medida que el juego avanza.

Por otro lado, en caso de haber encontrado todos los marcadores, se realiza la transición hacia la pantalla de victoria, donde se reproduce un sonido que marca el fin del juego.

```
botonCamara.onClick.AddListener(() => // cuando hago click al boton de foto
{
    // cambiamos el texto de detectados, el de las pistas etc
    if (encontrados < 6 || indicePuntos < 6) { //si encontrados es menor que 6 (solo hay 6 marcadores)
        encontrados = imagenesDetectadas.Count; // se suma 1 al contador. Coge la lista de imagenes detectadas y con la funcion .Count
        te devuelve el valor del numero total de nombres que tiene imagenes detectadas. Y este numero se le asigna a Encontrados
        indicePuntos = imagenesDetectadas.Count;
        textoEncontrados.text = "Encontrados: " + encontrados.ToString() + "/6";
    }
    if (encontrados==6) { //cuando encontrados ya es 6, se han detectado todos los marcadores
        ARCamara.transform.DOScale(new Vector3(0,0,0), 0f); // se pone la camara oculta
        PantallaFinal.transform.DOScale(new Vector3(1,1,1), 0f); //se muestra la pantalla final
        PantallaFinal.DOAnchorPos(new Vector2 (0, 0), 0.5f, false) // pasamos de -3000 a 0 (al centro de la pantalla), se hace en 0.5
        segundos
            .OnComplete(() => { // cuando se completa la transicion anterior
                // Aqu  es donde reproducimos el sonido
                audioSource.PlayOneShot(soundEffect); // audioSource es un objeto (que se arrastra), y soundEffect es un audio que me
                descargue
                Inicio.transform.DOScale(new Vector3(1,1,1), 0.3f); // se muestra el boton de inicio de la app para volver a el
            });
    }
});
```

Figura 48. Control de marcadores encontrados

El siguiente código hace referencia al control por GPS de la distancia del usuario hasta el siguiente marcador, usando la fórmula Harvesine y actualizando dinámicamente el texto que muestra la distancia restante y las pistas.

```
// recoge la ubicacion actual
LocationInfo ubicacion = Input.location.lastData;

// calcula la distancia al punto segun las coordenadas - Esto es la formula Haversine
const float radioTierra = 6371000; // meters // const es una constante, no se puede variar como una variable.
var dLat = Mathf.Deg2Rad * (puntosLatitud[indicePuntos] - ubicacion.latitude);
var dLon = Mathf.Deg2Rad * (puntosLongitud[indicePuntos] - ubicacion.longitude);
var a = Mathf.Sin(dLat / 2) * Mathf.Sin(dLat / 2) +
        Mathf.Cos(Mathf.Deg2Rad * ubicacion.latitude) * Mathf.Cos(Mathf.Deg2Rad * puntosLatitud[indicePuntos]) *
        Mathf.Sin(dLon / 2) * Mathf.Sin(dLon / 2);
var c = 2 * Mathf.Atan2(Mathf.Sqrt(a), Mathf.Sqrt(1 - a));
var distancia = radioTierra * c;
distancia = (float)Math.Round(distancia, 1); // este es el resultado de la formula

// actualiza en texto distancia segun el resultado de la formula
textoDistancia.text = "Distancia hasta el siguiente punto: " + distancia + "m. \nPista: " + pistas[indicePuntos]; //anade la
pista correspondiente al indice puntos actual
```

Figura 49. Cálculo de las distancias a los marcadores

6.3.5. Fin del juego

En las siguientes figuras se puede comparar el wireframe diseñado en “Canva” y el resultado durante la ejecución de la aplicación.

Figura 50. Diseño del wireframe 4

Este menú consta de un botón para ir hacia el menú principal y volver al menú principal, y de otro para capturar la pantalla del juego ocultando toda la interfaz.

Para ello se han insertado los elementos “Button” e “Image” de “UI”. La pantalla de victoria aparece cuando se consiguen encontrar todos los aumentos. También se ha añadido un audio de victoria.

6.3.6. Fotos

Extrayendo una parte del diagrama de flujo presentado anteriormente se muestran las acciones que se realizan, señalando en rojo el camino a seguir:

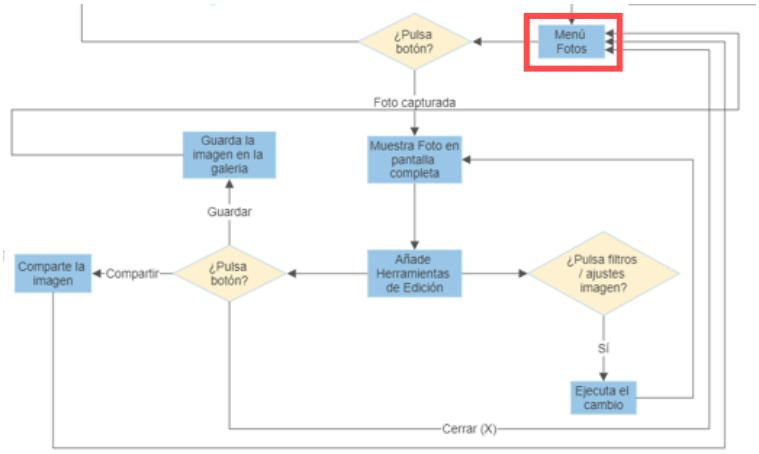

Figura 51. Zoom del diagrama de flujo 3

El siguiente código hace referencia a las distintas transiciones al pulsar el botón de Fotos del Menú Principal. Se oculta el Menú Principal para mostrar el de Fotos.

```
//Cuando click en boton Galeria, se oculta MenuPricipal y se muestra Galeria
private void ActivarGaleria()
{
    //Ponemos en todos los ejes 0 para ser ocultos. Las f son los segundos de transicion para hacer la accion
    MenuPrincipal.transform.GetChild(0).transform.DOScale(new Vector3(0,0,0), 0f);
    MenuPrincipal.transform.GetChild(1).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(2).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(3).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(4).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(5).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(6).transform.DOScale(new Vector3(0,0,0), 0.3f);

    //Ponemos en todos los ejes 1 para ser mostrados. Las f son los segundos de transicion para hacer la accion
    Galeria.transform.GetChild(0).transform.DOScale(new Vector3(1,1,1), 0f);
    Galeria.transform.GetChild(1).transform.DOScale(new Vector3(1,1,1), 0.3f);
    Galeria.transform.GetChild(2).transform.DOScale(new Vector3(1,1,1), 0.5f);
}
```

Figura 52. Control de acciones del botón Fotos

6.3.7. Galería

En las siguientes figuras se puede comparar el wireframe diseñado en “Canva” y el resultado durante la ejecución de la aplicación.

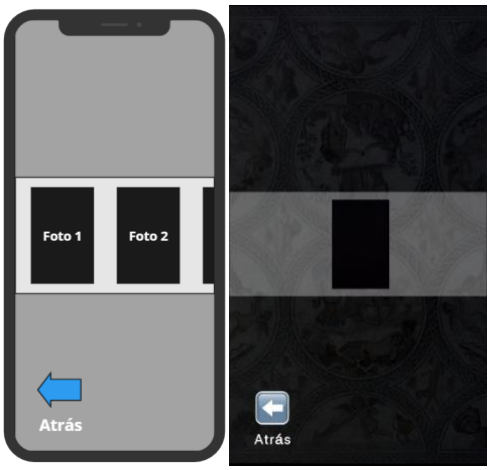

Figura 53. Diseño del wireframe 5

Este menú consta de un botón para ir hacia atrás y volver al menú principal, y de otro elemento "Slide", donde se muestran las fotos tomadas por el usuario en el minijuego y permite verlas desplazándote horizontalmente.

Para ello se han insertado los elementos "Scroll View" de "UI". Y una función en la que cada vez que se pulsa al botón de "Capturar" del minijuego, se realice la captura y se importe de forma temporal a este slider, se explicará en su respectivo apartado.

6.3.8. Edición de fotos

En las siguientes figuras se puede comparar el wireframe diseñado en "Canva" y el resultado en vivo de la APP.

Figura 54. Diseño del wireframe 6

Al pulsar sobre la imagen, permite editarla con filtros de blanco y negro, sepia, negativo. Y cambiar el brillo, contraste, saturación y tono de la misma, mediante una barra desplazable “Slider”. Después se puede guardar pulsando el botón correspondiente.

Para la programación de los filtros y el cambio de brillo/contraste/saturación/color, se obtiene la textura de la imagen y se editan los valores RGB para logran los cambios.

Para guardar las imágenes en la galería del dispositivo móvil, se ha usado una librería que nos ayuda en este proceso: UnityNativeGallery. Ya que con la fórmula que viene integrada en Unity, solo se conseguía guardar si se ejecutaba en un ordenador y no en un teléfono.

6.3.9. Historia de Cástulo

En las siguientes figuras se puede comparar el wireframe diseñado en “Canva” y el resultado durante la ejecución de la aplicación.

Figura 55. Diseño del wireframe 7

Extrayendo una parte del diagrama de flujo presentado anteriormente se muestran las acciones que se realizan, señalando en rojo el camino a seguir:

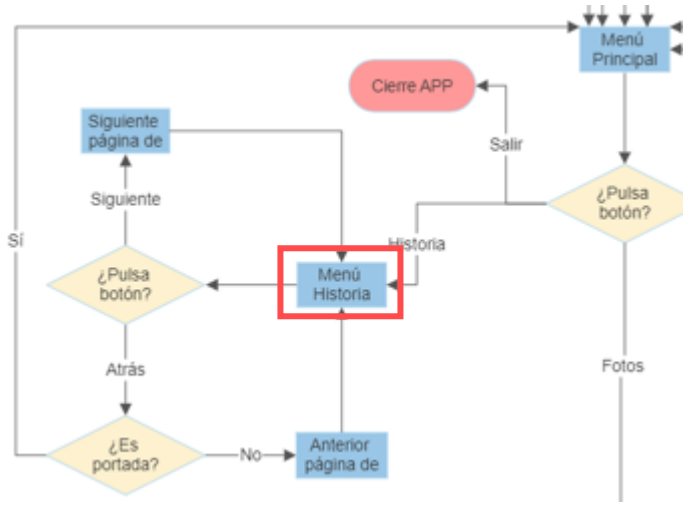

Figura 56. Zoom del diagrama de flujo 4

Este menú consta de dos botones, uno para ir hacia atrás y otro para ir hacia delante, y las acciones de estos sirven para navegar entre las distintas páginas donde se expone la historia de Cástulo. Para ello se han insertado los elementos "Image" y "Text" de "UI".

El siguiente código hace referencia a las distintas transiciones al pulsar el botón de Historia del Menú Principal, donde primero se especifica que la pantalla de Historia es la 0 para, posteriormente, controlar las acciones al pulsar el botón de Siguiente o Atrás. A continuación, se oculta el Menú Principal y se muestra la pantalla 0, en este caso el de la Portada.

```
private void ActivarHistoria()
{
    numTextoHistoria=0;
    //Ponemos en todos los ejes 0 para ser ocultos. Las f son los segundos de transicion para hacer la accion
    MenuPrincipal.transform.GetChild(0).transform.DOScale(new Vector3(0,0,0), 0f);
    MenuPrincipal.transform.GetChild(1).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(2).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(3).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(4).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(5).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(6).transform.DOScale(new Vector3(0,0,0), 0.3f);

    //Ponemos en todos los ejes 1 para ser mostrados. Las f son los segundos de transicion para hacer la accion
    Historia.transform.GetChild(0).transform.DOScale(new Vector3(1,1,1), 0f);
    Historia.transform.GetChild(1).transform.DOScale(new Vector3(1,1,1), 0.3f);
    Historia.transform.GetChild(2).transform.DOScale(new Vector3(0,0,0), 0f);
    Historia.transform.GetChild(3).transform.DOScale(new Vector3(1,1,1), 0.3f);
    Historia.transform.GetChild(4).transform.DOScale(new Vector3(1,1,1), 0.3f);

    //movemos el texto a su posicion
    Texto1.DOAnchorPos(new Vector2 (1500, 130), 0, false); //la portada y el texto estan como desplazados
}
```

Figura 57. Control de acciones del botón Historia

El siguiente código hace referencia a las distintas transiciones al pulsar el botón de Siguiente de la pantalla 0 de Historia, donde se controla que la pantalla actual sea la 0 y, a continuación, se cambie a la siguiente pantalla sucesivamente con sus respectivas transiciones.

Por otro lado, al pulsar el botón de Anterior del apartado de Historia de la aplicación, donde se controla que, si la pantalla actual es la 0, se retorne al menú principal ejecutándose su función y sino que a la pantalla actual se le reste 1, y haga las transiciones para volver a la página anterior.

```
private void ActivarSiguienteTextoHistoria() {
    //si es la portada desplazamos portada y texto1
    if (numTextoHistoria==0) { //si estamos en la portada, le sumamos 1 a la variable de pagina (++)
        numTextoHistoria++;
        Historia.transform.GetChild(2).transform.DOScale(new Vector3(1,1,1), 0f); //se activa el texto de historia
        Portada.DOAnchorPos(new Vector2 (-1500, 130), 0.3f, false); //desplazamiento en el eje x (horizontalmente) -1500,
        Texto1.DOAnchorPos(new Vector2 (0, 130), 0.3f, false);

    } else if (numTextoHistoria==1) {
        //si es texto1
    }

}

private void ActivarAnteriorTextoHistoria() {
    //si es la portada
    if (numTextoHistoria==0) { // si estamos en la portada de historia y le damos a ATRAS
        ActivarMenuPrincipal();
    } else {
        numTextoHistoria--; //sino, le restamos 1 a variable de pagina, y se pasa a la pagina anterior de historia (porta
        Portada.DOAnchorPos(new Vector2 (0, 130), 0.3f, false); // ahora se ve la portada
        Texto1.DOAnchorPos(new Vector2 (1500, 130), 0.3f, false);
    }
}
```

Figura 58. Control de acciones del botón Siguiente 2

6.3.10. Información de la aplicación

En las siguientes figuras se puede comparar el wireframe diseñado en “Canva” y el resultado durante la ejecución de la aplicación.

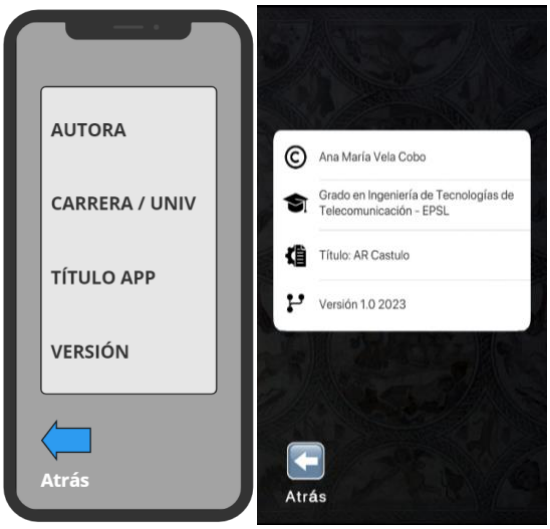

Figura 59. Diseño del wireframe 8

Extrayendo una parte del diagrama de flujo presentado anteriormente se muestran las acciones que se realizan, señalando en rojo el camino a seguir:

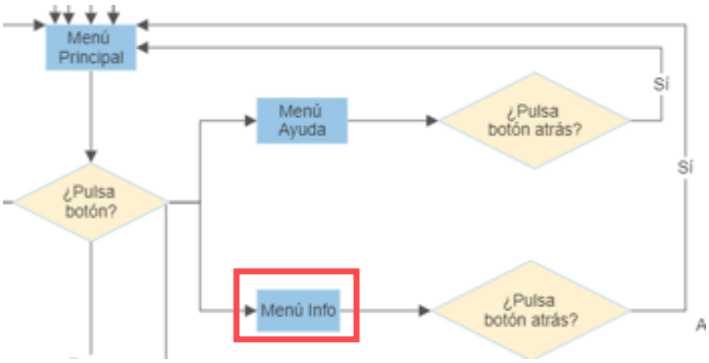

Figura 60. Zoom del diagrama de flujo 5

Este menú consta de un botón para ir hacia atrás y volver al menú principal, ocultando éste. Se muestra la información relativa al proyecto: Autor, Versión, Título, etc. Para ello se han insertado los elementos “Image” y “Text” de “UI”.

El siguiente código hace referencia a las distintas transiciones al pulsar el botón de Info del Menú Principal. Se oculta el Menú Principal para mostrar el de Info.

```
private void ActivarInfo(){
    //Ponemos en todos los ejes 0 para ser ocultos. Las f son los segundos de transicion para hacer la accion
    MenuPrincipal.transform.GetChild(0).transform.DOScale(new Vector3(0,0,0), 0f);
    MenuPrincipal.transform.GetChild(1).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(2).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(3).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(4).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(5).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(6).transform.DOScale(new Vector3(0,0,0), 0.3f);

    //Ponemos en todos los ejes 1 para ser mostrados. Las f son los segundos de transicion para hacer la accion
    Info.transform.GetChild(0).transform.DOScale(new Vector3(1,1,1), 0f);
    Info.transform.GetChild(1).transform.DOScale(new Vector3(1.2f,0.7f,1), 0.3f); //1.2 y 0.7 es la escala de la info
    Info.transform.GetChild(2).transform.DOScale(new Vector3(1,1,1), 0.3f);
}
```

Figura 61. Control de acciones del botón Info

6.3.11. Ayuda

En las siguientes figuras se puede comparar el wireframe diseñado en “Canva” y el resultado durante la ejecución de la aplicación.

Figura 62. Diseño del wireframe 9

Extrayendo una parte del diagrama de flujo presentado anteriormente se muestran las acciones que se realizan, señalando en rojo el camino a seguir:

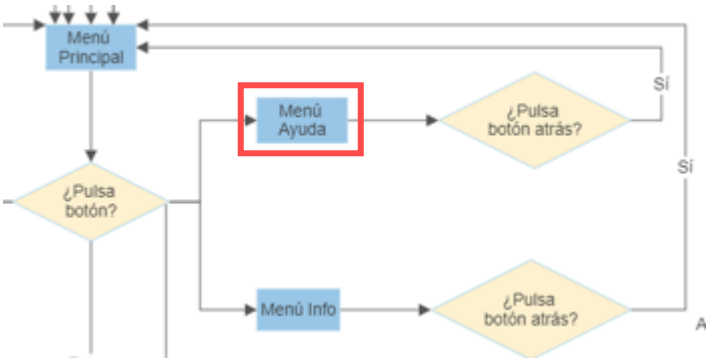

Figura 63. Zoom del diagrama de flujo 6

Este menú consta de un botón para ir hacia atrás y volver al menú principal, ocultando este. Se muestra una breve explicación de las funcionalidades de cada botón y menú que forma parte de la aplicación.

El siguiente código hace referencia a las distintas transiciones al pulsar el botón de Ayuda del Menú Principal. Se oculta el Menú Principal para mostrar el de Ayuda.

```
private void ActivarAyuda(){
    //Ponemos en todos los ejes 0 para ser ocultos. Las f son los segundos de transicion para hacer la accion
    MenuPrincipal.transform.GetChild(0).transform.DOScale(new Vector3(0,0,0), 0f);
    MenuPrincipal.transform.GetChild(1).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(2).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(3).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(4).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(5).transform.DOScale(new Vector3(0,0,0), 0.3f);
    MenuPrincipal.transform.GetChild(6).transform.DOScale(new Vector3(0,0,0), 0.3f);

    //Ponemos en todos los ejes 1 para ser mostrados. Las f son los segundos de transicion para hacer la acci
    Ayuda.transform.GetChild(0).transform.DOScale(new Vector3(1,1,1), 0f);
    Ayuda.transform.GetChild(1).transform.DOScale(new Vector3(1.2f,0.7f,1), 0.3f); //1.2 y 0.7 es la escala d
    Ayuda.transform.GetChild(2).transform.DOScale(new Vector3(1,1,1), 0.3f);
}
```

Figura 64. Control de acciones del botón Ayuda

6.3.12. Salir

Al pulsar sobre el botón Salir, se ha programado que se cierre la aplicación. A continuación, el código que lo ejecuta.

```
public void CerrarAPP()
{
    Application.Quit();
}
```

Figura 65. Control de acciones del botón Salir

6.4. Funciones

Al iniciarse la aplicación, se le solicita al usuario una serie de permisos los cuales deben ser aceptados para que ciertas funciones que se expondrán en este apartado puedan funcionar correctamente.

A continuación, extrayendo una parte del diagrama de flujo presentado anteriormente se muestran las acciones mencionadas:

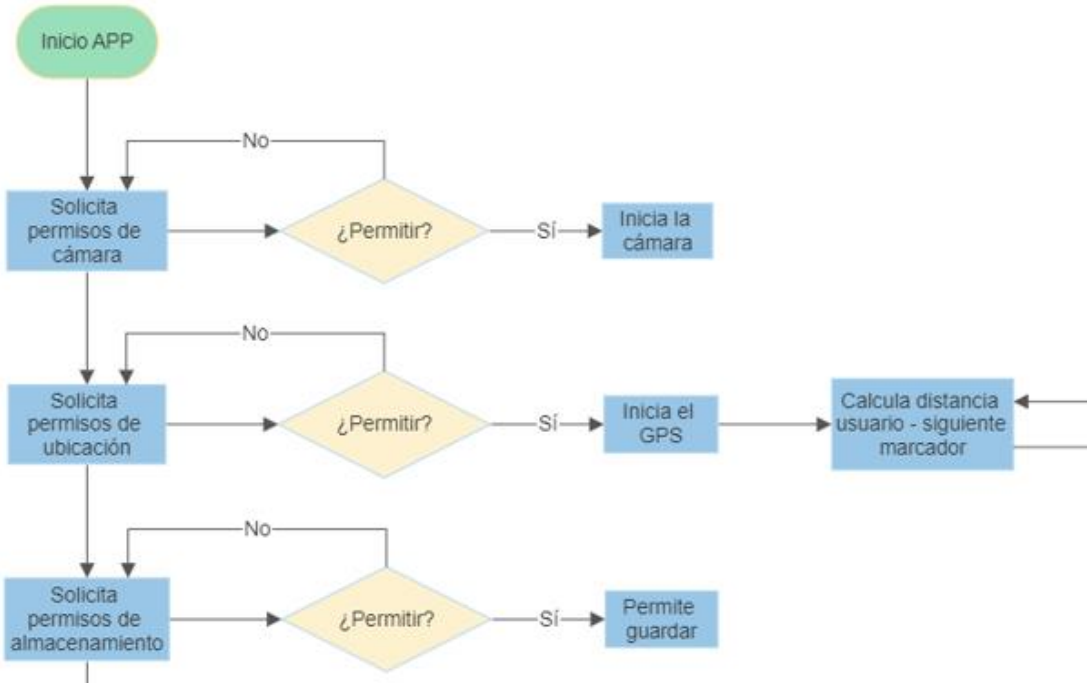

Figura 66. Zoom del diagrama de flujo 7

6.4.1. Reconocimiento

Se lleva a cabo gracias a la librería de "Vuforia Engine", usando los Image Targets, que son los marcadores tomados en Cástulo digitalizados y añadidos a la BBDD de Vuforia para su posterior reconocimiento al enfocarlo con la cámara.

Dentro del juego, llevaremos un recuento de los aumentos encontrados, y según en cual estemos que muestre la distancia al siguiente punto y la pista correspondiente ayudándonos de la brújula. Para ello dentro del script creado para la ubicación, usaremos la función de Observer de Vuforia, para controlar cuando se detectan las image targets, para poder sumarlos al recuento de los aumentos y que estos no se sumen si ya han sido detectados.

De la misma forma, actualizaremos las pistas de la distancia hacia el siguiente punto ya que añadiremos las coordenadas en un array, y este mostrará los datos de las coordenadas

siendo el índice el número de aumentos encontrados. Al finalizar y encontrar todos los aumentos se muestra una pantalla de victoria y que se puede ir al inicio, y también mientras se despliega he añadido una pista de audio de victoria.

A continuación, se muestra una parte extraída del script donde a cada image target se le añade la función Observer de Unity, para controlar la propiedad de estado en caso de ser detectada.

```
//añado funcion observer para cada image target hijo del elemento Empezar
// cojo el menu empezar y para cada elemento que tenga una propiedad 'observer' (son los image target -> los
aumentos), se añade la siguiente funcion
ObserverBehaviour[] observers = GetComponentsInChildren<ObserverBehaviour>();
foreach (ObserverBehaviour observer in observers)
{
    observer.OnTargetStatusChanged += OnTargetStatusChanged; // cada image target tiene una propiedad de estado, y
    cuando se detecta, el estado cambia. con esta funcion lo que controlas es saber cuando el estado cambia
}
if (observers.Length == 0) // si Empezar no tiene ningun hijo con la propiedad 'observer' devuelve el error (nunca va
a pasar pero se puso para las pruebas)
{
    Debug.LogError("ObserverBehaviour not found in any children.");
}
```

Figura 67. Función para el control de la propiedad de estado de cada aumento

En la siguiente parte extraída del script, se controla cuando el usuario detecta una image target. En caso de que la propiedad cambie a “TRACKED” [29], se obtendrá el nombre de la image target en cuestión para, posteriormente, añadirla al array de detectadas en caso de que sea la primera vez que se detecte.

```
private void OnTargetStatusChanged(ObserverBehaviour observerBehaviour, TargetStatus targetStatus) // lo de 'observer' de
arriba.
// esta funcion se ejecuta cuando el estado del image target cambia. Se coge el image target y su Status
{
    // comprobamos cuando el estado sea TRACKED y NORMAL (hay muchos mas estados)
    if (targetStatus.Status == Status.TRACKED && targetStatus.StatusInfo == StatusInfo.NORMAL)
    {
        // recogemos el nombre de la image target
        string nombreImagen = observerBehaviour.TargetName; // ej: TermaPequena

        // comprobamos si ha sido detectada previamente y no es una funcion llamada DeviceObserver
        if (!imagenesDetectadas.Contains(nombreImagen) && nombreImagen != "DeviceObserver") // el objeto cuando lo detecta
        tiene 2 nombres, ej TermaPEquena, y otro nombre que es 'DeviceObserver', este se pone solo en Unity
        // si en la lista no existe el nombre de la imagen (inicialmente hay 0 nombres en la lista), se anade el nombre.
        Asi si se detecta mas de 1 vez la TermaPequena, el contador solo suma 1
        {
            // a adimos el nombre a la lista de detectadas
            imagenesDetectadas.Add(nombreImagen); // aqui se anadian los 2 nombres, y sumaba 2. ahora no
        }
    }
}
```

Figura 68. Función para el control de la detección del marcador

Al reconocer cada aumento, se han añadido imágenes en 2D dibujadas por mí; efectos [30], como por ejemplo humo en las aguas termales; audios, como por ejemplo un sonido de

burbujas en las aguas termales; y personajes 3D con movimientos, para conseguir acercar al usuario al entorno de la época.

6.4.2. Captura de imágenes

Para que cuente como aumento encontrado será necesario hacer una foto o captura.

Ha habido un error al hacer fotos. Tras muchos testeos y fijarse en el log, es un error de la inicialización de los píxeles de la cámara y no se ha encontrado solución en ninguna documentación o foro, así que se ha optado por usar capturas.

Así que se ha creado un script donde hace la captura al pulsar sobre el botón de la cámara y guarda temporalmente la foto en una carpeta del caché de la aplicación haciendo que, al salir de la aplicación, en el caso de no haber sido guardada, se borre.

Las capturas que se tomen aparecerán en un Slider en el menú de Fotos, a modo de "Galería" del teléfono.

Al hacer la captura, aparecían todos los elementos del menú aparte de lo que muestra la cámara. Para solucionar esto, se ha añadido al código que antes de que se llame a la función para hacer la captura, se oculten todos los elementos del UI del menú menos la cámara AR y los aumentos.

La siguiente parte del script hace referencia a la función asignada en el momento de pulsar en el botón de capturar, donde primeramente se ocultan todos los elementos de la UI que no sean los de la cámara AR para que no aparezcan en ésta y se llama a la función que creará la captura.

```
public void TakeScreenshot() // se ejecuta cuando se hace click en la camara
{
    atras.transform.localScale = Vector3.zero; // se ponen los objetos de la pantalla en ocult
    captura solo se muestra lo que aparece en la camara
    btnCamara.transform.localScale = Vector3.zero;
    brujula.transform.localScale = Vector3.zero;
    textoEnc.transform.localScale = Vector3.zero;
    imagen.transform.localScale = Vector3.zero; // (estas a tantos m) vector 3 es que la x,y,z
    /* PC
    ScreenCapture.CaptureScreenshot(screenshotPath); // es una funcion interna que hace una ca
    anterior
    */
    StartCoroutine(WaitForScreenshot()); // llama a la funcion 'esperar a hacerse la captura'
}
```

Figura 69. Función para ocultar la UI al hacer la captura

A continuación, una parte extraída del script que ejecuta lo anteriormente mencionado, donde tras ejecutarse la función anterior, la captura que se toma de pantalla se guarda su información en una variable de textura de Unity, para seguidamente hacer un Sprite funcional para usarlo más tarde. Se creará el botón a modo de imagen dentro del slider de Fotos y, por último, se le añadirá el Sprite previamente creado y de esta forma se vea a modo de miniatura.

```
private IEnumerator WaitForScreenshot()
{
    yield return new WaitForEndOfFrame();

    Texture2D screenshotTexture = ScreenCapture.CaptureScreenshotAsTexture();

    Sprite screenshotSprite = Sprite.Create(screenshotTexture, new Rect(0, 0, Screen.width, Screen.height), Vector2.
    zero); // declara una variable Sprite que tranforma la textura y los datos de la imagen en una imagen visible

    GameObject screenshotButtonObject = new GameObject("ScreenshotButton"); // crea el objeto boton que se anadira dentr
    del Scroll View
    screenshotButtonObject.transform.SetParent(screenshotScrollView.content.transform); // lo mete dentro del Scroll Vie

    RectTransform rectTransform = screenshotButtonObject.AddComponent<RectTransform>();
    rectTransform.sizeDelta = new Vector2(240f, 375f); // altura y anchura
    rectTransform.localScale = new Vector3(1f, 1f, 1f); // tamano (x,y,z)

    Image screenshotImage = screenshotButtonObject.AddComponent<Image>(); // crea un componente que es una imagen y lo
    anade dentro del objeto boton creado antes
    screenshotImage.sprite = screenshotSprite; // le anade la imagen al objeto imagen
```

Figura 70. Función para guardar la captura en la sección de Fotos de la aplicación

6.4.3. Localización

En el script de a continuación, primero se piden los permisos al usuario tal y como sale en la documentación de Unity, y tras pedir los permisos y verificar que estos han sido autorizados, se ejecuta la siguiente función de Localizar.

```
// Comprobar si hay permisos de localizacion sino se piden al usuario
if (!Permission.HasUserAuthorizedPermission(Permission.CoarseLocation)) {
    Permission.RequestUserPermission(Permission.CoarseLocation);
}
if (Permission.HasUserAuthorizedPermission(Permission.CoarseLocation) && !Permission.HasUserAuthorizedPermission
(Permission.FineLocation)) {
    Permission.RequestUserPermission(Permission.FineLocation);
}

        // Verificar si se ha otorgado el permiso
if (!Permission.HasUserAuthorizedPermission(Permission.ExternalStorageWrite))
{
    // Solicitar permiso al usuario
    Permission.RequestUserPermission(Permission.ExternalStorageWrite);
    return;
}

// Empieza la actualizacion de la ubicacion
StartCoroutine(Localizar()); // StartCoroutine es como si fuese un if (una palabra reservada)
```

Figura 71. Permisos y función de Localizar

A continuación, como se ha mencionado anteriormente, se ejecuta la función de Localizar, donde se comprueba que los permisos estén activos si no es así los solicita cada 3 segundos. Una vez verificado, se procede a pedir la ubicación del dispositivo esperando la respuesta hasta 20 segundos. Si todo es correcto y se inicializa, seguirá ejecutando el código de a continuación. En caso contrario, esperará hasta que se agote el tiempo de espera y lanzará un mensaje de error.

```
IEnumerator Localizar() //(function, palabra reservada)
{
    // Esperar hasta que se active el servicio
    while (!Input.location.isEnabledByUser) { //mientras los permisos de localizacion no esten activados por el usuario
        yield return new WaitForSeconds(3); // espera tres segundo para volver a pedir la localizacion y comprobar si
        tiene los permisos
    }

    // Empieza el servicio de localizacion
    Input.location.Start();

    // tiempo de espera hasta que se inicialice
    int maxTiempoEspera = 20;
    while (Input.location.status == LocationServiceStatus.Initializing && maxTiempoEspera > 0) { //mientras el estado es
    que se esta Inicializando y el tiempo de espera (20) es mayor que 0, entra
        yield return new WaitForSeconds(1); //vuelve a esperar 1 seg para comprobar si aun esta Inicializandose
        maxTiempoEspera--; //al maximo tiempo de espera le quita 1 (es como una cuenta atras)
    }

    // si falla
    if (maxTiempoEspera == 0) { //si el tiempo max de espera (20) llega a 0
        Debug.Log("Tiempo de espera superado para la inicialización del servicio de ubicación"); //entonces volveria a
        arriba del todo de esta funcion
        yield break;
    }
```

Figura 72. Función para pedir la ubicación del usuario

Si las comprobaciones han sido correctas, se obtienen las coordenadas en una variable. Para calcular la distancia entre la ubicación actual y la final, que en este caso sería el aumento que se esté tratando de encontrar, primero se necesita saber las coordenadas del punto final que se ha usado Google Maps para extraer las coordenadas del aumento, estando en Cástulo. Y, por último, se ha utilizado la Fórmula de Haversine para obtener los metros restantes.

Una vez obtenidos estos datos se muestran en el recuadro de pistas y se van actualizando a medida que nos movemos.

```
// si esta funcionando
if (Input.location.status == LocationServiceStatus.Running) { //ahora si el estado esta corriendo, es decir,
funcionando
    while (true) {
        // recoge la ubicacion actual
        LocationInfo ubicacion = Input.location.lastData;

        // calcula la distancia al punto segun las coordenadas - Esto es la formula Haversine
        const float radioTierra = 6371000; // meters // const es una constante, no se puede variar como una variable.
        var dLat = Mathf.Deg2Rad * (puntosLatitud[indicePuntos] - ubicacion.latitude);
        var dLon = Mathf.Deg2Rad * (puntosLongitud[indicePuntos] - ubicacion.longitude);
        var a = Mathf.Sin(dLat / 2) * Mathf.Sin(dLat / 2) +
                Mathf.Cos(Mathf.Deg2Rad * ubicacion.latitude) * Mathf.Cos(Mathf.Deg2Rad * puntosLatitud[indicePuntos])
                *
                Mathf.Sin(dLon / 2) * Mathf.Sin(dLon / 2);
        var c = 2 * Mathf.Atan2(Mathf.Sqrt(a), Mathf.Sqrt(1 - a));
        var distancia = radioTierra * c;
        distancia = (float)Math.Round(distancia, 1); // este es el resultado de la formula

        // actualiza en texto distancia segun el resultado de la formula
        textoDistancia.text = "Distancia hasta el siguiente punto: " + distancia + "m. \nPista: " + pistas
        [indicePuntos]; //anade la pista correspondiente al indice puntos actual

        // tiempo de esperar para la siguiente actualizacion
        yield return new WaitForSeconds(1); //espera 1 seg para volver a pedir de nuevo tu ubicacion
    }
}
```

Figura 73. Función del cálculo de la distancia entre el usuario y el marcador para mostrar en las pistas

6.4.4. Orientación

Para crear una brújula funcional, primero añadimos a la UI una imagen de una brújula y luego crearemos un script usando la función interna de Unity "Compass" [31], y la activaremos. A continuación, iremos rotando la brújula según nos movamos y que ésta siempre apunte al norte, usando el sistema de ubicación y giroscopio del teléfono.

A continuación, se muestra el script mencionado, donde se recoge la imagen a utilizar de la brújula y se inicializa junto al componente de brújula de Unity. Una vez inicializado se indica cual será el punto del teléfono para la medición, en este caso la parte superior orientándolo verticalmente y, mediante otro componente de orientación de Unity, se detectarán los cambios según el usuario se gire o mueva y actualizará la posición de la imagen.

```
public class BrujulaUI : MonoBehaviour
{
    public RawImage imagenBrujula; //se declara la imagen de la brujula (flecha roja)

    private Compass brujula; // este es el componente brujula del unity

    void Start()
    {
        brujula = Input.compass; //al iniciar la app, guardas el componente brujula (compass) en la variable brujula
        brujula.enabled = true; // se activa
    }

    void Update() //una vez esta iniciado
    {
        float heading = Input.compass.trueHeading; //se crea la variable heading para indicarle a la brujula la posicion en
        vertical del telefono
        Quaternion rotation = Quaternion.Euler(0, 0, heading); // ej: si giro el telefono a la derecha, que lo detecte y gire
        a la derecha la brujula
        imagenBrujula.transform.rotation = rotation; //actualiza la imagen de la brujula para que vaya apuntando
    }
}
```

Figura 74. Función para inicializar y actualizar periódicamente la brújula

6.4.5. Edición de imágenes

Se ha programado un script que al pulsar encima de su miniatura, se muestre en pantalla completa una previsualización en modo edición usando la información guardada temporalmente.

A continuación, se muestra una parte extraída del script mencionado.

Primero se controla si la imagen está en pantalla completa. Dicha comprobación es debida a que, tanto al pulsar sobre la imagen en miniatura como para luego cerrarla, se llama a esta función en caso de que no esté en pantalla completa, es decir, se ha pulsado a la miniatura, se recoge la información del Sprite de la miniatura para añadirla a otro objeto, pero dimensionándola en pantalla completa. También se añaden las distintas herramientas de edición, filtrado, guardar y compartir. A todas estas herramientas se les añade su respectiva función al ser pulsadas.

```
public void ShowFullscreenImage(Sprite screenshotSprite) // llama al hacer click en la imagen de f
{
    if (!isFullscreenActive) // si cuando le hacemos click, la variable isFullscreenActive es fals
    {
        fullscreenImage = fullscreenImageObject.GetComponent<Image>(); // guardas en la variable f
        fullscreenImageObject
        fullscreenImage.sprite = screenshotSprite; // el valor es la imagen Sprite que es el param
        arrastrando)
        spriteOriginal = filtroImageObject.GetComponent<Image>();
        spriteOriginal.sprite = screenshotSprite;
        originalTexture = fullscreenImage.sprite.texture;

        // Establecer los valores iniciales de las barras deslizantes
        brightnessSlider.value = 0.5f;
        contrastSlider.value = 0.5f;
        saturationSlider.value = 0.5f;
        hueSlider.value = 0.5f;

        fullscreenImageObject.gameObject.SetActive(true); // estaba inactivo y ahora se muestra
        isFullscreenActive = true; // la variable ahora es true
        closeFullScreen.onClick.AddListener(() => ShowFullscreenImage(fullscreenImage.sprite)); //
        ejecutar la funcion, para cerrarlo
        saveImage.onClick.AddListener(() => saveImg("ok"));
        shareImage.onClick.AddListener(() => shareImg("ok"));
        filtroByn.gameObject.SetActive(true);
        filtroSepia.gameObject.SetActive(true);
        filtroNegativo.gameObject.SetActive(true);
        filtroOriginal.gameObject.SetActive(true);
        containerFiltro.gameObject.SetActive(true);
        brightnessSlider.gameObject.SetActive(true);
        contrastSlider.gameObject.SetActive(true);
        saturationSlider.gameObject.SetActive(true);
        hueSlider.gameObject.SetActive(true);
        closeFullScreen.gameObject.SetActive(true);
        saveImage.gameObject.SetActive(true);
        shareImage.gameObject.SetActive(true);
        filtroByn.onClick.AddListener(() => ApplySelectedFilter("byn"));
        filtroSepia.onClick.AddListener(() => ApplySelectedFilter("sepia"));
        filtroNegativo.onClick.AddListener(() => ApplySelectedFilter("negativo"));
        filtroOriginal.onClick.AddListener(() => ApplySelectedFilter("original"));

        // Suscribirse a los eventos de las barras deslizantes
        brightnessSlider.onValueChanged.AddListener(AdjustTexture);
        contrastSlider.onValueChanged.AddListener(AdjustTexture);
        saturationSlider.onValueChanged.AddListener(AdjustTexture);
        hueSlider.onValueChanged.AddListener(AdjustTexture);
    }
```

Figura 75. Función para acceder a la pantalla de edición de la foto pulsada

En caso de pulsar sobre el botón de cerrar, se vuelve a ejecutar la misma función, pero al estar en pantalla completa, oculta todos los elementos de edición y vuelve a mostrar el menú de Fotos.

La imagen se puede editar mediante unos filtros por defecto programados en el mismo script, que al pulsar encima de ellos modificarán los valores RGB y mostrarán los cambios en directo. Por otra parte, también se pueden editar los valores del contraste, saturación, brillo y tono, actualizándose también en directo.

A continuación, se muestra una parte extraída del script mencionado, en la cual se puede observar la edición del filtro de Blanco y Negro y cómo se cambian los valores RGB para más tarde aplicarlo.

```
private Texture2D ApplyFilter(Texture2D texture, string filter)
{
    Color[] pixels = texture.GetPixels();

    switch (filter)
    {
        case "byn":
            for (int i = 0; i < pixels.Length; i++)
            {
                float grayscale = (pixels[i].r + pixels[i].g + pixels[i].b) / 3f;
                pixels[i] = new Color(grayscale, grayscale, grayscale, pixels[i].a);
            }
            break;
        case "sepia":
```

Figura 76. Función para aplicar el filtro de blanco y negro

A continuación, se muestra una parte extraída del script mencionado, en la cual se puede observar la edición del brillo, contraste y saturación, y cómo se cambian los valores RGB para más tarde aplicarlo.

```
private void AdjustTexture(float valueOk)
{
    float brightness = brightnessSlider.value * 2f; // Sin multiplicar por 2f
    float contrast = contrastSlider.value * 2f;
    float saturation = saturationSlider.value * 2f;
    float hue = (hueSlider.value - 0.5f) * 2f;

        // Crear una nueva textura para contener los valores ajustados
        adjustedTexture = new Texture2D(fullscreenImage.sprite.texture.width, fullscreenImage.sprite.texture.height);

        // Obtener los p◆xeles originales de la textura
        Color[] originalPixels = fullscreenImage.sprite.texture.GetPixels();

        // Ajustar cada p◆xel en la textura
        for (int i = 0; i < originalPixels.Length; i++)
        {
            Color originalColor = originalPixels[i];

            // Ajustar el brillo
            Color adjustedColor = originalColor * brightness;

            // Ajustar el contraste
            adjustedColor.r = (adjustedColor.r - 0.5f) * contrast + 0.5f;
            adjustedColor.g = (adjustedColor.g - 0.5f) * contrast + 0.5f;
            adjustedColor.b = (adjustedColor.b - 0.5f) * contrast + 0.5f;
```

Figura 77. Función para editar brillo, contraste, saturación y valores RGB

En la siguiente parte del script, se muestra cómo se crean las miniaturas para aplicar los filtros, donde el usuario puede ver una vista previa del resultado al pulsar la miniatura de cada filtro.

```
//MINIATURAS
//original
Image scrennshootFiltroOriginal = filtroOriginal.gameObject.GetComponent<Image>(); // crea un componente que es una
imagen y lo anade dentro del objeto boton creado antes
scrennshootFiltroOriginal.sprite = screenshotSprite;
//byn
Image scrennshootFiltroByn = filtroByn.gameObject.GetComponent<Image>(); // crea un componente que es una imagen y lo
anade dentro del objeto boton creado antes
scrennshootFiltroByn.sprite = screenshotSprite;
Texture2D filteredTextureByN = ApplyFilter(scrennshootFiltroByn.sprite.texture, "byn");
scrennshootFiltroByn.sprite = Sprite.Create(filteredTextureByN, new Rect(0, 0, filteredTextureByN.width,
filteredTextureByN.height), Vector2.one * 0.5f);
//sepia
Image scrennshootFiltroSepia = filtroSepia.gameObject.GetComponent<Image>(); // crea un componente que es una imagen y
lo anade dentro del objeto boton creado antes
scrennshootFiltroSepia.sprite = screenshotSprite;
Texture2D filteredTextureSepia = ApplyFilter(scrennshootFiltroSepia.sprite.texture, "sepia");
scrennshootFiltroSepia.sprite = Sprite.Create(filteredTextureSepia, new Rect(0, 0, filteredTextureSepia.width,
filteredTextureSepia.height), Vector2.one * 0.5f);
//negativo
Image scrennshootFiltroNegativo = filtroNegativo.gameObject.GetComponent<Image>(); // crea un componente que es una
imagen y lo anade dentro del objeto boton creado antes
scrennshootFiltroNegativo.sprite = screenshotSprite;
Texture2D filteredTextureNegativo = ApplyFilter(scrennshootFiltroNegativo.sprite.texture, "negativo");
scrennshootFiltroNegativo.sprite = Sprite.Create(filteredTextureNegativo, new Rect(0, 0, filteredTextureNegativo.width,
filteredTextureNegativo.height), Vector2.one * 0.5f);
```

Al cerrar la imagen en modo edición se desharán todos los cambios, así que, el usuario en el caso de querer guardar la imagen editada o sin editar, deberá hacerlo previamente.

6.4.6. Guardar imágenes

En el mismo script que el de edición de imágenes, mediante la librería UnityNativeGallery, se ha programado el guardado en la galería del dispositivo la imagen, y con un nombre en común añadiendo fecha y hora, para no sobrescribir.

La calidad de la imagen es la misma que se muestra por pantalla, tal y como hacen aplicaciones de redes sociales como Instagram, que usan la imagen que proyecta la cámara del dispositivo para capturarla, sin hacer una foto o vídeo como tal. Se ejecutará al pulsar sobre el botón de guardar.

A continuación, el script mencionado, donde se obtiene la textura de la imagen y se le asigna un nombre junto a la fecha y hora para que sea única, y no se sobrescriba. Por último, si se tiene permisos se guardará en la galería del dispositivo mediante la función propia de esta librería.

```
private void saveImg(string ok)
{
    string fileName = "screenshot" + DateTime.Now.ToString("ddMMyyyyHHmmss") + ".png";
    NativeGallery.Permission permission = NativeGallery.SaveImageToGallery(fullscreenImage.sprite.texture,
    "MyScreenshots", fileName);
    if (permission == NativeGallery.Permission.Granted)
    {
        Debug.Log("Screenshot saved to gallery");
    }
    else
    {
        Debug.Log("Failed to save screenshot to gallery");
    }
}
```

Figura 78. Función para guardar las fotos en el dispositivo

6.4.7. Compartir imágenes

En el mismo script que el de edición de imágenes, mediante la librería UnityNativeShare, se ha programado la opción de compartir a las distintas redes sociales que tenga el dispositivo. Se ejecutará al pulsar sobre el botón de compartir.

A continuación, el script mencionado, donde se obtiene la textura de la imagen y se le asigna un nombre junto a la fecha y hora para que sea única, y no se sobrescriba en la ruta temporal. Por último, convertirla en PNG y compartirla mediante la función propia de esta librería a las distintas redes disponibles.

```
private void shareImg(string ok)
{
    Texture2D textureShare = fullscreenImage.sprite.texture;

    // Guardar la imagen en una ubicaci□n temporal
    string fileName = "screenshot" + DateTime.Now.ToString("ddMMyyyyHHmmss") + ".png";
    string path = Path.Combine(Application.temporaryCachePath, "share.png");
    File.WriteAllBytes(path, textureShare.EncodeToPNG());

    // Compartir la imagen utilizando la funcionalidad del sistema operativo
    new NativeShare().AddFile(path).Share();
}
```

Figura 79. Función para compartir las fotos

7. RESULTADOS

Finalmente, se ha desarrollado una aplicación en el ámbito cultural sobre el Yacimiento Arqueológico de Cástulo donde se han usado la geolocalización y la brújula para guiar al usuario hasta los distintos puntos destacados, en los cuales se activarán los aumentos que dan vida a estas ruinas, indicando cómo se cree que eran en la época de la ciudad ibero-romana. Con esta aplicación podremos introducirnos en la vida de antiguos habitantes y aprender más sobre Monumento nacional.

7.1. Geolocalización y brújula

La aplicación comienza pidiendo los permisos sobre la ubicación, a partir de la cual calculará a qué distancia se encuentra del primer marcador. Una vez se llega al punto que ha guiado y se captura el aumento, automáticamente indica la distancia hasta el siguiente punto.

Además, para mejorar el guiado del usuario, se ha incluido una brújula y pistas que señalarán la dirección hacia la cual se deberá dirigir para encontrar cada marcador.

7.2. Interacción

Tras guiar al usuario a las distintas zonas, se deben cumplir una serie de aspectos para lograr el objetivo de activar los aumentos y capturar las imágenes. El proceso completo consta de varias partes. Estas son: activar el aumento, capturar y editar las imágenes, guardar en el dispositivo y compartir. A continuación, se van a explicar cada una:

Activar el aumento

El primer paso es conseguir activar el aumento 2D o 3D [32]. Para ello, tras ser guiado hasta el primer punto mediante las pistas y los metros que faltan para llegar a él, se deberá enfocar este de frente, hasta que detecte el marcador que muestre el aumento por pantalla.

Figura 80. Activación del aumento

Capturar el aumento

El segundo paso, tras lograr activar el aumento y aparecer la reconstrucción de la ruina, se deberá pulsar sobre el botón de “Captura”, para capturar el aumento y añadir uno al contador de la esquina superior izquierda. De forma simultánea, se cambiará la distancia al siguiente punto a encontrar. Así, sucesivamente, hasta completar 6 de 6 encontrados.

A continuación, se verifica cómo en una prueba de la aplicación, tras detectar el marcador, mostrar el aumento y pulsar el botón de ‘Captura’, se actualiza la información de forma automática y muestra la distancia hasta el siguiente marcador.

Figura 81. Actualización de la distancia tras el primer aumento

En la primera imagen se observa que, tras activar el aumento, aún sigue marcando la distancia hasta este primer punto. En la segunda imagen, en la cual ya se ha pulsado el botón de 'Captura', aparecen encontrados 1/6 y se ha actualizado además la distancia al segundo punto.

Editar las imágenes

Tras capturar el aumento, se podrá continuar con el juego para completar todas las búsquedas o se podrá salir para acceder al botón de Fotos y ver las fotografías que se han hecho, y editarlas mediante los filtros o modificar el brillo, contraste, saturación y tono.

Si tras editar las imágenes se quiere volver al juego, solo habrá que pulsar de nuevo el botón de Inicio y el juego continuará por donde se dejó.

A continuación, se muestra en la primera imagen la galería donde se almacenan las capturas realizadas, y en la segunda la edición de una de ellas, donde se ha usado el filtro de negativo y se han modificado el contraste y el tono.

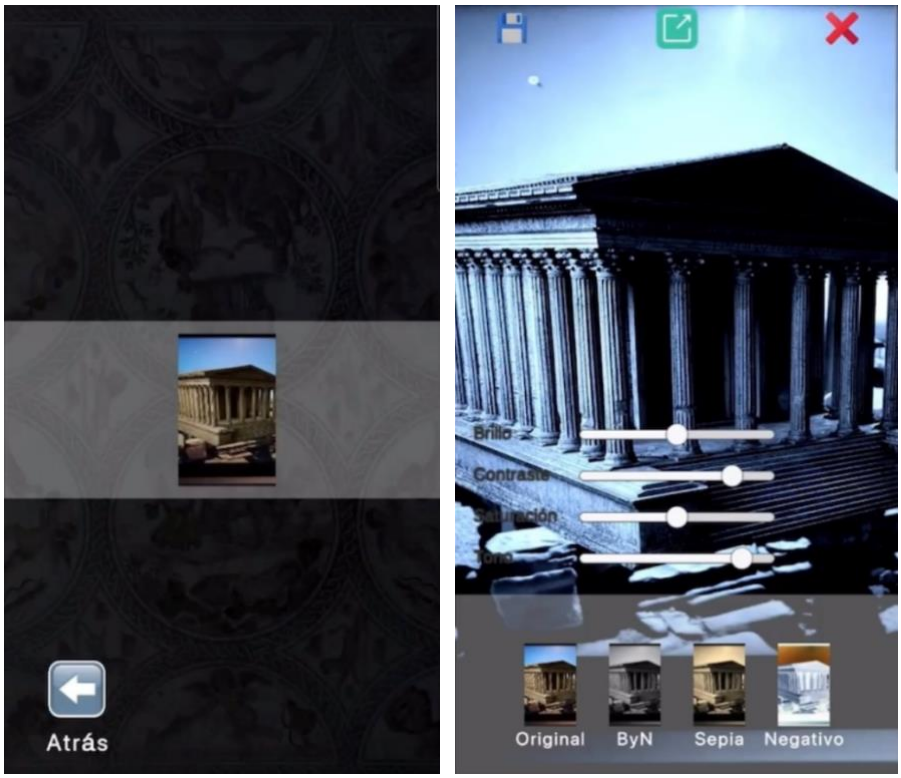

Figura 82. Edición de la imagen

Guardar en el dispositivo

Al capturar uno o todos los aumentos del juego, en la galería de fotos, se permite guardar cualquier imagen en la propia galería del dispositivo pulsando sobre el botón de la esquina izquierda. Es por ello que, al abrir la aplicación, pide los permisos correspondientes para poder guardar las imágenes en el dispositivo.

Se permite guardar tanto las fotos sin editar como las que hayan sido editadas en la aplicación.

Compartir las imágenes

Además del proceso de edición y guardado de las imágenes, en la galería de fotos, se podrán compartir en otras redes sociales o enviarlas por mensaje pulsando sobre el botón del medio. De nuevo, se permite compartir tanto las fotos sin editar como las que hayan sido editadas en la aplicación.

8. LÍNEAS FUTURAS

Al finalizar el proyecto se han llegado a unas conclusiones de mejora para líneas futuras, las cuales se van a redactar a continuación.

La primera de ellas es el incluir nuevos marcadores situados, por ejemplo, en el Museo Arqueológico de Linares, debido a que, al ser un espacio cerrado, sin variaciones por la meteorología y sol, facilitaría la detección de los marcadores y con ello la activación de los aumentos.

Esto se debe a que, durante las distintas pruebas de la aplicación, se han tenido varias dificultades al detectar correctamente los marcadores, ya que las imágenes iniciales fueron tomadas a una hora (concretamente por la tarde) diferente a la que se hizo la primera prueba (por la mañana), por lo que se tuvo que hacer nuevas fotografías de los marcadores para poder ser detectados. La mínima variación de alguna zona de luz que pase a ser zona de sombra, dificultaría la detección del marcador, ya que se trata de ruinas. Aun así, se consiguió cumplir con el objetivo de activar todos los aumentos.

Para solventar este problema, se deberían de añadir más fotografías de cada marcador a lo largo de las diferentes horas del día.

Es por ello que otra línea de futuro planteada es la de ofrecer una imagen impresa en papel al usuario al iniciar el juego, facilitando la detección de los marcadores. La finalidad de esta idea es conseguir que todos los usuarios obtengan 6/6 aumentos encontrados de forma más sencilla, debido a que, al ser un espacio exterior con demasiados cambios al día, se dificulta el uso de la realidad aumentada en este tipo de ambientes.

Además, se podrían lograr las verdaderas reconstrucciones de las ruinas ya que los contenidos multimedia presentados en la aplicación son mejorables. Pero esto quedaba fuera del alcance del proyecto debido a que se necesitarían conocimientos de modelaje 3D, lo que supone un mayor aprendizaje en esta materia y tiempo dedicado.

Y, por último, otra línea de futuro que se plantea es la de añadir otro método para la detección de los marcadores por proximidad, usando para ello la tecnología Bluetooth.

En este proyecto se ha intentado la introducción de esta tecnología, se explica a continuación.

El método consistía en que mediante el uso de un dispositivo Bluetooth (en este caso unos AirPods), cuando el teléfono detectase dicho dispositivo apareciese un mensaje de ayuda parecido a: ¡Se ha detectado la ayuda del Bluetooth! El siguiente marcador se encuentra frente a usted y debe enfocar a la torre para detectarlo.

La implementación de la detección se llevó a cabo mediante la librería disponible en el repositorio GitHub [33]. Esta librería permitiría una comunicación fluida entre la aplicación Unity y los dispositivos Android compatibles con BLE. Se han explorado otras opciones, pero siendo estas de pago.

A pesar de haber seguido la documentación proporcionada por la librería, la integración en la aplicación no resultó como se esperaba. Se procedió a la instalación de la librería siguiendo las instrucciones detalladas. Se programó un script en Unity para establecer la comunicación BLE y permitir la interacción con el dispositivo anteriormente mencionado.

Sin embargo, durante el proceso de depuración y testeo la aplicación no se iniciaba y no se generaba ningún código de error. A pesar de múltiples intentos de corrección, incluida la revisión y cambio del código, el problema no se solventó y la aplicación no pudo ser iniciada con éxito.

A continuación, se proporciona una evidencia el script desarrollado para la detección de unos cascos inalámbricos (Airpods), donde se esperaba un mensaje en la consola. Posteriormente se hubiese sustituido por la aparición del mensaje mencionado con anterioridad:

```
using BLEPlugin;

0 references
public class BLEConnectionManager : MonoBehaviour
{
    private string nombreDispositivo = "AirPods de Ana";
    private BLEDevice bleDevice;
    public Text statusText;

    0 references
    void Start()
    {
        bleDevice = new BLEDevice(nombreDispositivo);
        bleDevice.OnConnect += OnBLEConnect;
        bleDevice.OnDisconnect += OnBLEDisconnect;

        if (bleDevice.Initialize())
        {
            statusText.text = "BLE iniciado ok.";
        }
        else
        {
            statusText.text = "Error inicio BLE.";
        }
    }

    1 reference
    void OnBLEConnect()
    {
        statusText.text = "Conexión BLE ok. Ayuda de BT activada!";
    }

    1 reference
    void OnBLEDisconnect()
    {
        statusText.text = "Conexión BLE perdida.";
    }
```

Figura 83. Script para detección mediante Bluetooth

9. CONCLUSIÓN

Como conclusiones finales cabe destacar el logro de una aplicación gamificada que cumple con los objetivos previamente indicados, siendo práctica e interesante.

Esta aplicación consiste en recrear en una visita cultural, la vida de los habitantes de la ciudad ibero-romana, que introduce a los usuarios en las viviendas y zonas de ocio.

Con el desarrollo del estado del arte se ha logrado mostrar la historia completa de esta tecnología y cómo se ha introducido en el ámbito del Patrimonio Cultural, como ocurre en esta aplicación. Además, se ha realizado una comparación con otras aplicaciones ya creadas resaltando las diferencias con la app que se expone en este proyecto.

Por último, se ha realizado un estudio de cuál es el funcionamiento de la geolocalización en el dispositivo móvil con lo que se logra guiar al usuario hasta los distintos puntos, y de cómo se consigue el procesado de audio y de imagen. Tras el estudio, se logró la aplicación que cumplió con las pruebas finales dando lugar a una app interactiva basada en un juego.

10. BIBLIOGRAFÍA

[1] Joe Carmichael. 2016. "Did L. Frank Baum Predict Augmented Reality or Warn Us About Its Power?" Inverse. (https://www.inverse.com/article/18146-l-frank-baum-the-master-key-augmented-reality-futurism)

[2] Carlos Trilnick, 1962. "Sensorama." Estados Unidos. IDIS (https://proyectoidis.org/sensorama/)

[3] Jeremy M. Norman. "Ivan Sutherland and Bob Sproull Create the First Virtual Reality Head Mounted Display System" HistoryofInformation.com Exploring the History of Information and Media through Timelines (https://www.historyofinformation.com/detail.php?id=861)

[4] Juan Carlos González, 2012. Videoplace, el abuelo artista de Kinect. Xataka (https://www.xataka.com/historia-tecnologica/videoplace-el-abuelo-artista-de-kinect)

[5] Los comienzos de la inmersión simulada. (https://netartuem.hotglue.me/Realidad-Aumentada)

[6] İbrahim Sünger and Serkan Çankaya, "Augmented Reality: Historical Development and Area of Usage. Journal of Educational Technology and Online Learning, 2(3), 118-133" İzmir Democracy University, İzmir, Turkey. 2019. [Online]. Available: https://www.researchgate.net/publication/336537084_Augmented_Reality_Historical_Development_and_Area_of_Usage

[7] "Proyecto académico orientado a la aplicación de la Realidad Aumentada en periodismo." 2017. (http://realidadaumentadayperiodismo.blogspot.com/2017/12/antecedentes.html)

[8] "ARQuake: Interactive Outdoor Augmented Reality Collaboration System" 2010. (https://www.tinmith.net/arquake/)

[9] Patricia Muñoz Antón. (2022). Desarrollo de aplicación AR para integración de personajes históricos en visitas culturales. [Trabajo Fin de Grado, Universidad de Jaén]. Disponible: https://crea.ujaen.es/handle/10953.1/18056

[10] Guideo, una app turística basada en la realidad aumentada. SEOptimer https://www.seoptimer.com/es/blog/guideo-una-app-turistica-basada-en-la-realidad-aumentada/

[11] Pablo Garcia Fuente. (2021). Realidad aumentada para la visualización e interacción con objetos 3D en Unity [Tesis doctoral, Universidad de Valladolid]. https://uvadoc.uva.es/bitstream/handle/10324/47964/TFG-I-1939.pdf

[12] Ruiz Torres, D. (2013). La realidad aumentada y su aplicación en el patrimonio cultural. Gijón, Ediciones Trea. Recuperado de https://elibro.net/es/ereader/ujaen/97577

[13] Alegría Blázquez Sevilla. Realidad aumentada en Educación. Universidad Politécnica de Madrid, Gabinete de Tele-Educación. https://oa.upm.es/45985/1/Realidad_Aumentada__Educacion.pdf

[14] Types of AR. Digital Promise. https://digitalpromise.org/initiative/360-story-lab/360-production-guide/investigate/augmented-reality/getting-started-with-ar/types-of-ar/

[15] Raúl Reinoso. Introducción a la Realidad Aumentada. Tecnotic (https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjRovvLs_aAAxUdRaQEHTUGDc8QFnoECA8QAQ&url=https%3A%2F%2Fwww.educa.jcyl.es%2Fcrol%2Fes%2Frepositorio-global%2Fintroduccion-realidad-aumentada-37162.ficheros%2F511255-3711.pdf&usg=AOvVaw2LO1FwY1CYBZhyiRWxaA0n&opi=89978449)

[16] 10 aplicaciones de realidad aumentada para el aula (https://www.educaciontrespuntocero.com/recursos/aplicaciones-realidad-aumentada/)

[17] La realidad aumentada en medicina salva y mejora vida (https://iat.es/tecnologias/realidad-aumentada/medicina/)

[18] Realidad virtual y realidad aumentada en medicina. Almirallmed. 2022 (https://neurologia.almirallmed.es/blog/realidad-virtual-y-realidad-aumentada-en-medicina/)

[19] La Realidad Aumentada cambia el futuro de la moda. Yeeply (https://www.yeeply.com/blog/realidad-aumentada-cambia-futuro-moda/)

[20] Realidad aumentada en la arquitectura. (https://realidadaumentada.click/realidad-aumentada-en-la-arquitectura/#Morpholio_AR_Sketwalch)

[21] El avance del periodismo con la realidad aumentada. Sapiens. Universitas Miguel Hernández (https://umhsapiens.com/el-avance-del-periodismo-con-la-realidad-aumentada/)

[22] 10 ejemplos de realidad aumentada para aplicar a tu estrategia de marketing. 2022. Rebold by ISPD (https://letsrebold.com/es/blog/ejemplos-de-realidad-aumentada-marketing/)

[23] Application of Augmented Reality (AR) Technologies in inhouse Logistics. IAECST. 2020 (https://www.e3s-conferences.org/articles/e3sconf/pdf/2020/05/e3sconf_iaecst2020_02018.pdf)

[24] DOTween. Demigiant. A Unity Tween Engine. (http://dotween.demigiant.com/documentation.php)

[25] A native Unity plugin to interact with Gallery/Photos on Android & iOS. GitHub (https://github.com/yasirkula/UnityNativeGallery)

[26] A Unity plugin to natively share files (images, videos, documents, etc.) and/or plain text on Android & iOS. GitHub (https://github.com/yasirkula/UnityNativeShare)

[27] Runtime permissions do not work for GPS Location first two runs. Unity (https://forum.unity.com/threads/runtime-permissions-do-not-work-for-gps-location-first-two-runs.1005001/#post-6520438)

[28] Wireframes creados con Figma: https://www.figma.com

[29] TrackableBehaviour Class Reference. Unity Reference (https://library-archive.vuforia.com/sites/default/files/references/9.8/unity/classVuforia_1_1TrackableBehaviour.html)

[30] Fog tutorial. GitHub (https://github.com/etredal/FogTutorial/tree/main)

[31] Scripting API: Compass. Unity (https://docs.unity3d.com/ScriptReference/Compass.html)

[32] Aumentos y personajes 3D conseguidos en las siguientes páginas web:
https://assetstore.unity.com/
https://sketchfab.com/
https://www.cgtrader.com/
https://www.turbosquid.com/

[33] A Unity Android plugin to support basic Bluetooth Low Energy interactions. GitHub https://github.com/Velorexe/Unity-Android-Bluetooth-Low-Energy

ANEXO 1. INSTALACIÓN DE UNITY

Para instalar Unity primero se debe descargar el instalador ejecutable desde: https://unity.com/es/download

A continuación, se ejecuta y se acepta el acuerdo de licencia.

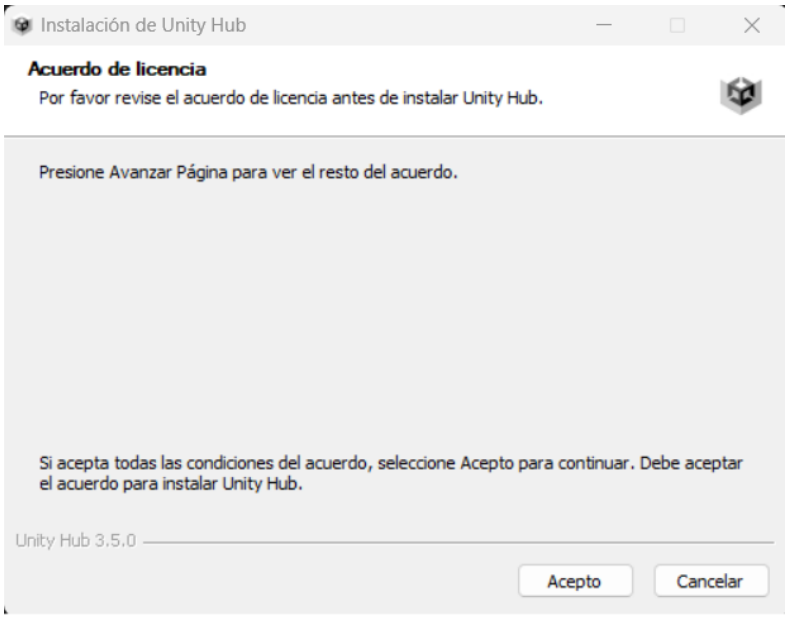

Figura 84. Imagen de instalación de Unity Hub 1

Se selecciona la ruta del destino en la que va a ser instalado.

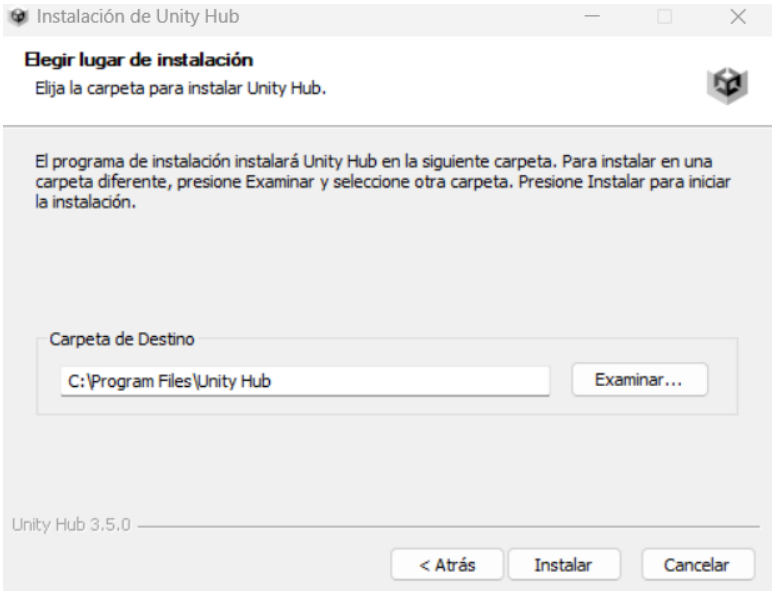

Figura 85. Imagen de instalación de Unity Hub 2

Por último, se ejecuta Unity Hub y se instala la versión más reciente.

El siguiente paso es crear una cuenta en Unity Asset Store para la posterior instalación de los distintos plug-in o librerías que se necesiten. Para ello se debe acceder a: https://assetstore.unity.com/ .

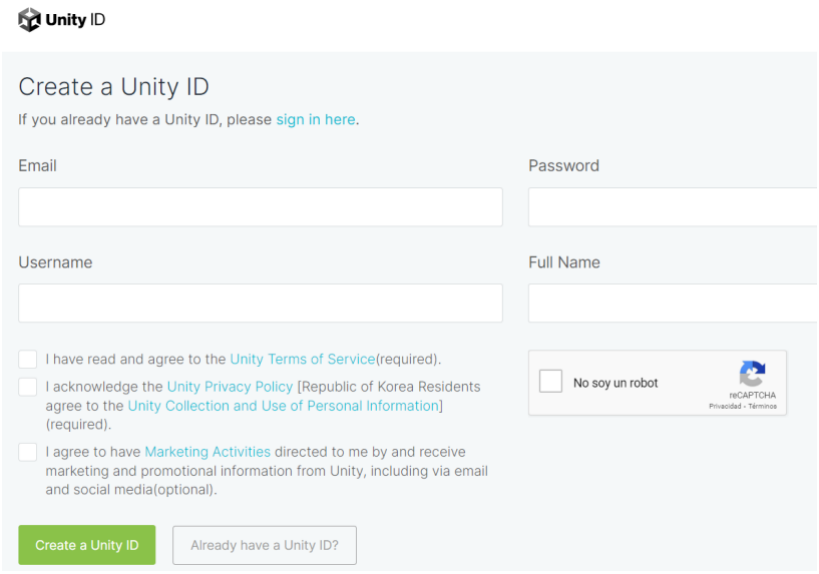

Figura 86. Imagen de creación de cuenta en Unity Asset Store

ANEXO 2. CREACIÓN DEL ENTORNO DE DESARROLLO

Una vez instalado se crea un nuevo proyecto con la plantilla de AR.

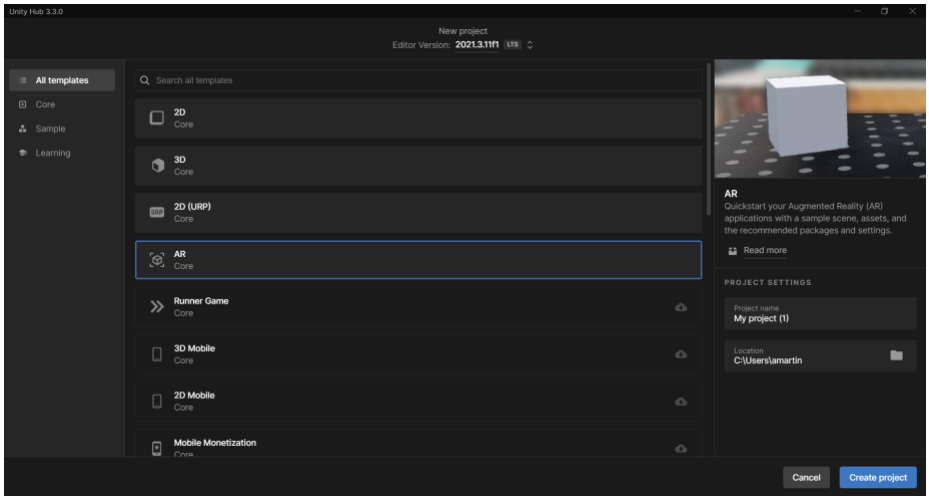

Figura 87. Creación de nuevo proyecto

En primer lugar, se necesita un dispositivo compatible con ARcore, ya sea Android o IOS. Para esta aplicación se ha usado un dispositivo móvil Android.

Se instala ARcore en Unity. Se indican los pasos a continuación: Window > Package Manager > Packages > Unity Registry y se busca ARcore.

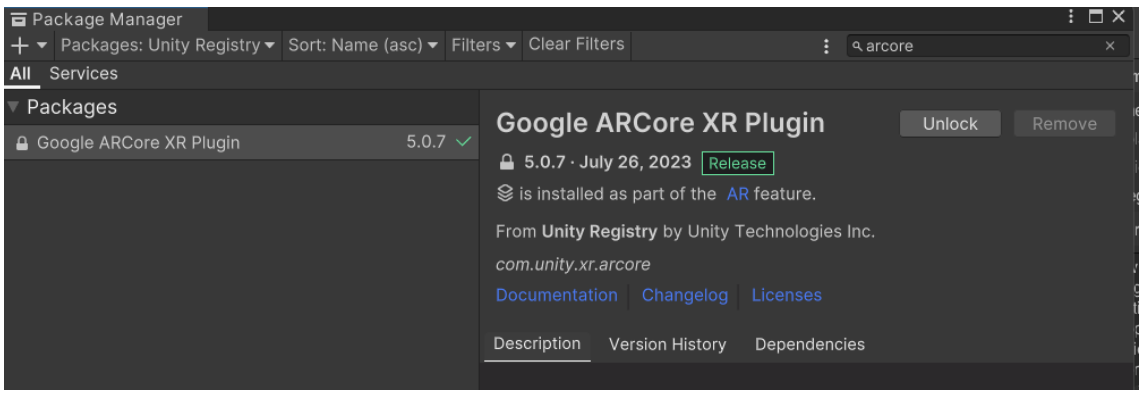

Figura 88. Importación ARCore al proyecto

Se configura el proyecto:

Primero se accederá a File > Build Settings > Android > Install > Switch Platform, para cambiar el desarrollo a Android.

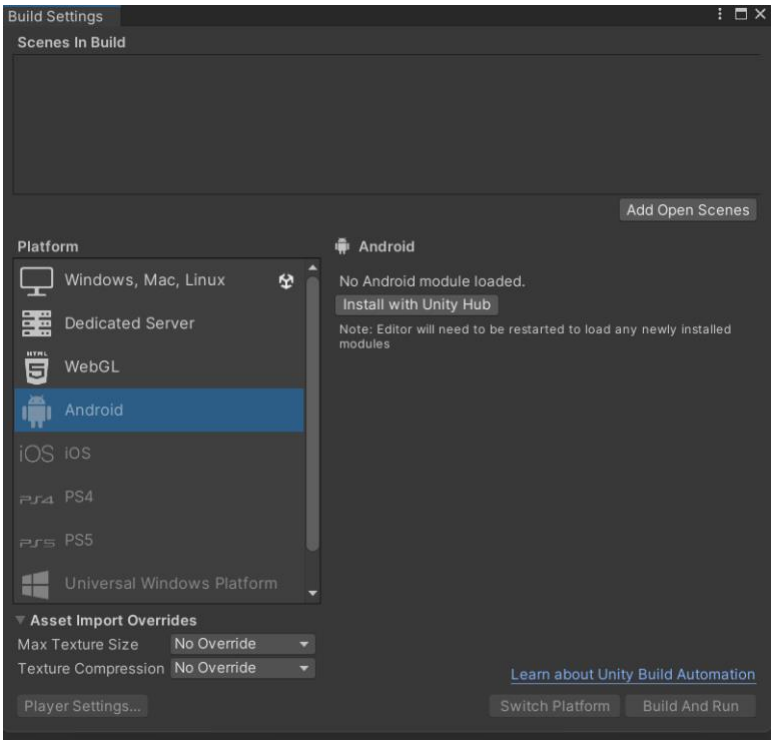

Figura 89. Cambio del desarrollo a Android

A continuación, se accede a Edit > Project Settings > Player > Other Settings y se elimina Vulkan.

En el mismo menú se cambiará el Minimum API Level a Android 8.0 y el Scripting Backend a IL2CPP y se habilita ARM64.

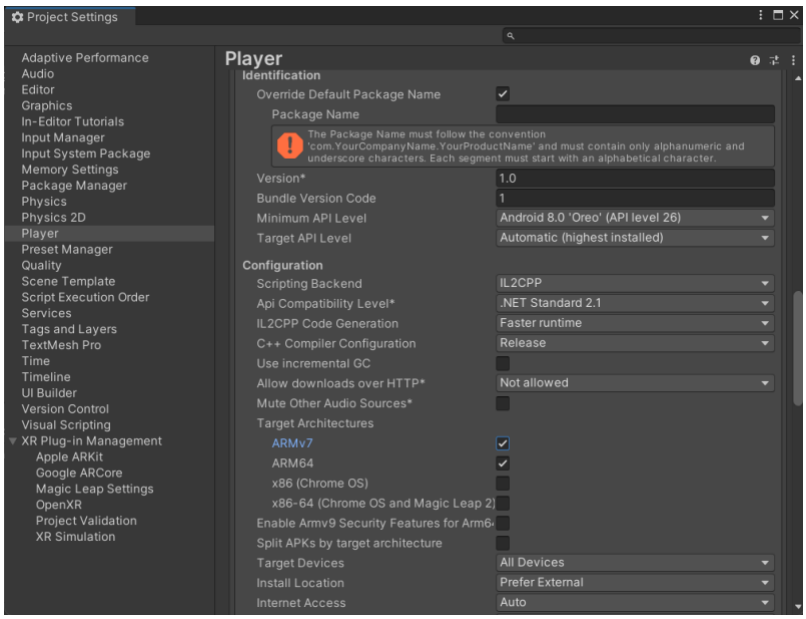

Figura 90. Configuración de parámetros para el dispositivo Android

Se debe integrar Vuforia con su SDK. Primero se procede al registro y descarga del archivo desde el siguiente link https://developer.vuforia.com/downloads/SDK .

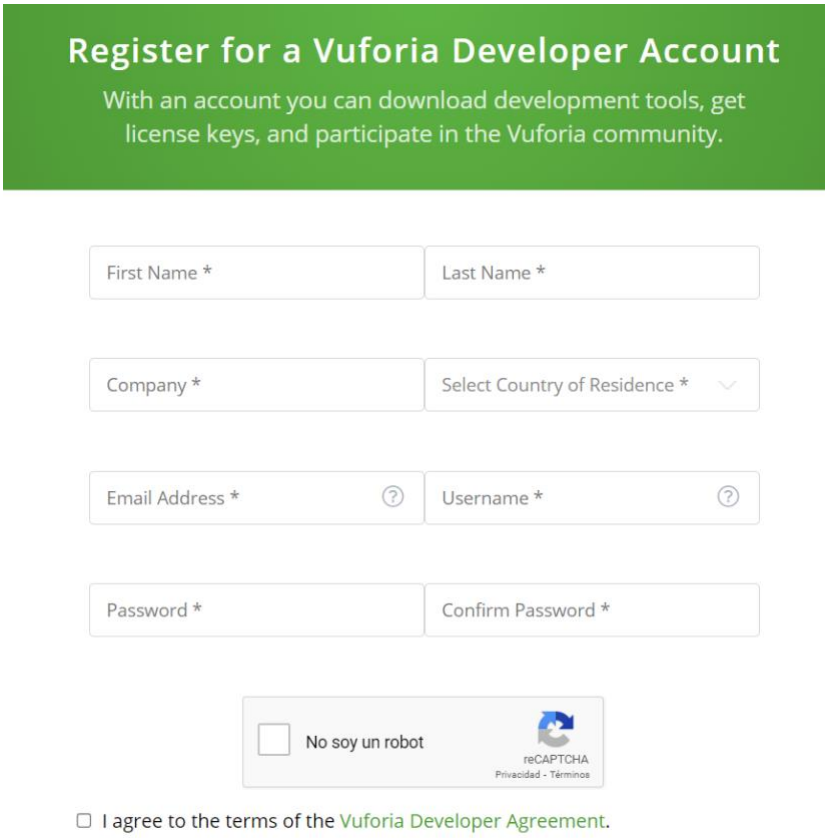

Figura 91. Registro en Vuforia

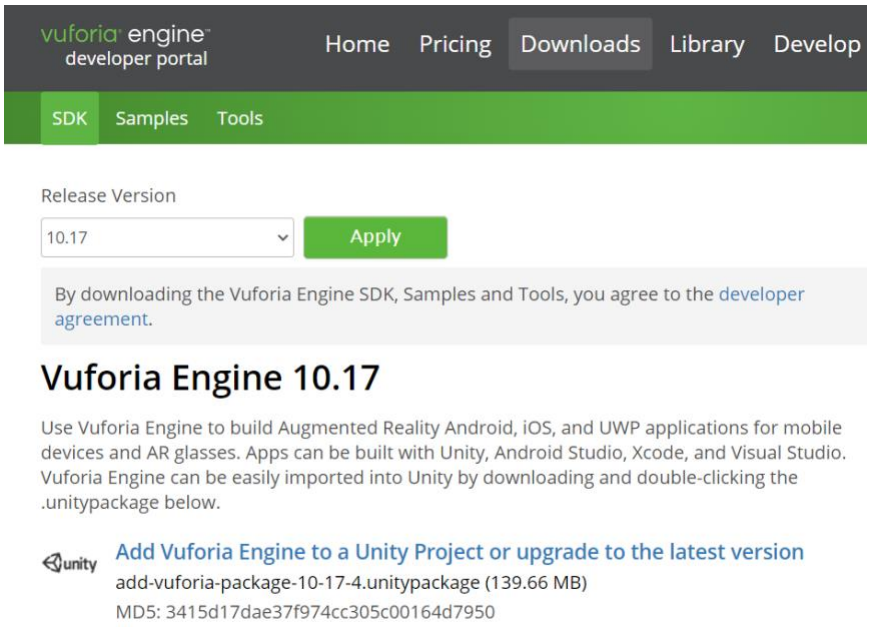

Figura 92. Página de descarga de Vuforia

Se importa el paquete en Assets > Import Package > Custom Package y se selecciona el archivo.

Figura 93. Proceso de la importación del paquete de Unity

Primero en la jerarquía del proyecto se eliminará la Main Camera y haciendo clic derecho > Vuforia Engine > AR Camera se añadirá la nueva cámara, desde la configuración de ésta se añade la licencia que se generará en el siguiente link: https://developer.vuforia.com/vui/develop/licenses

Entonces, se hará clic en Get Basic indicándole el nombre. Y, por último, se copiará en la AR Camera.

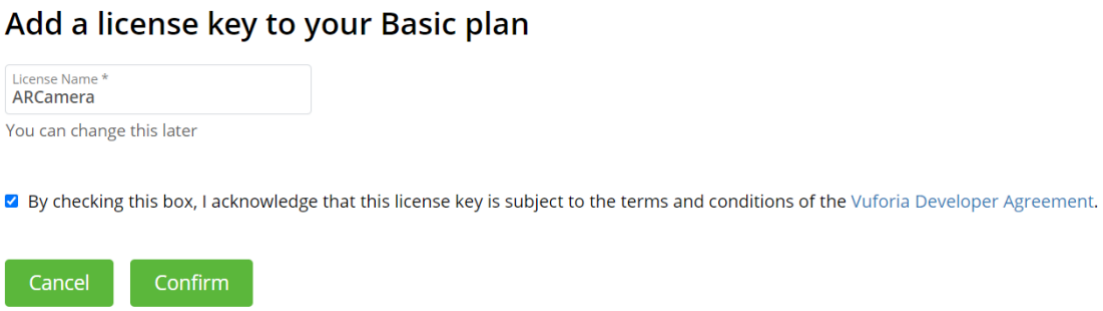

Figura 94. Generación de la clave de la ARCamera de Vuforia

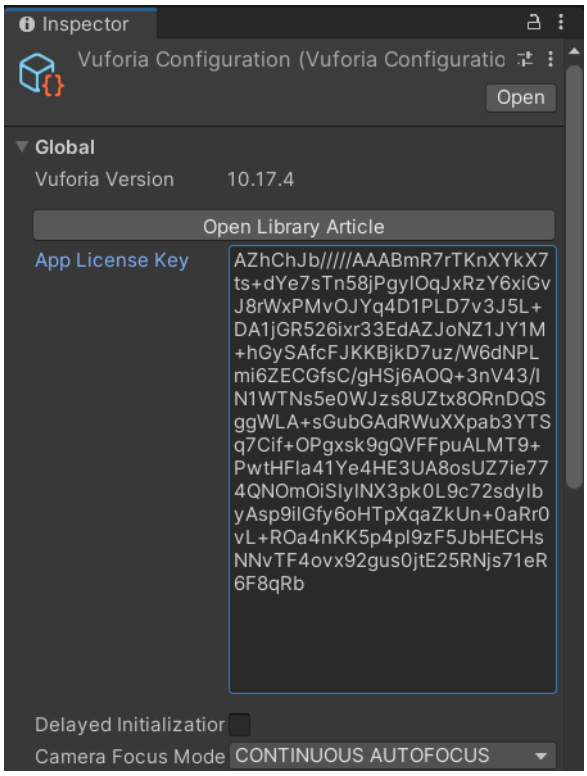

Figura 95. Licencia generada añadida a la ARCamera del proyecto

En esta misma configuración se seleccionará REQUIRED en ARCore Requirement y se desmarcará Include ARCore library.

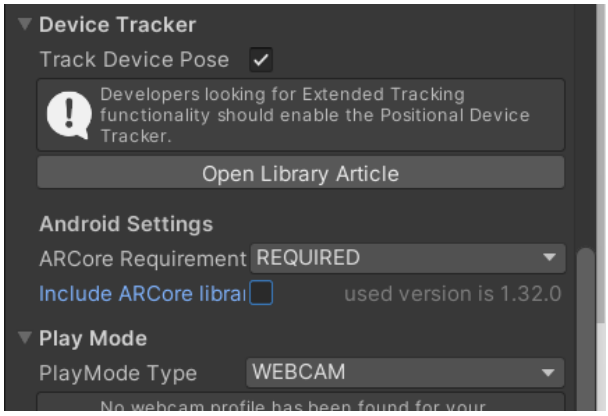

Figura 96. Selección de REQUIRED

Ya que se añadirá desde Edit > Project Settings > XR Plugin Management, y en estas opciones se seleccionará ARCore.

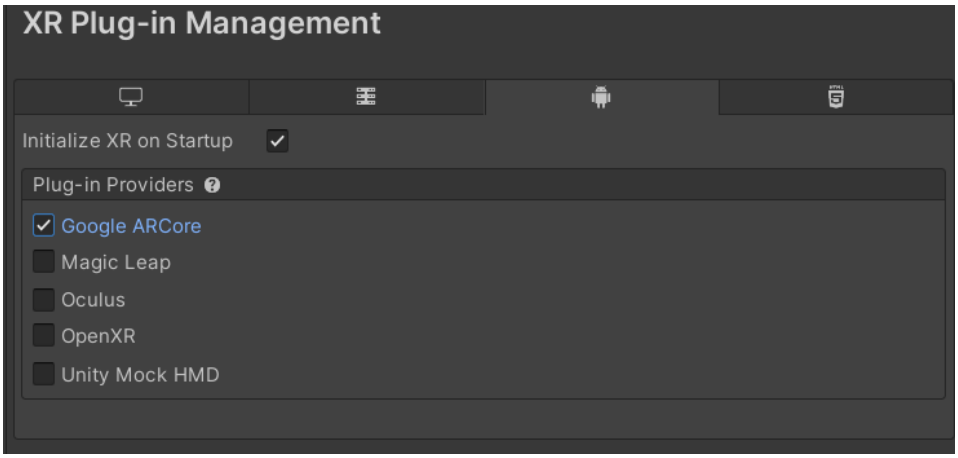

Figura 97. Selección de ARCore

El siguiente paso será instalar las demás librerías que necesitará el proyecto, como son DOTween para las transiciones, NativeGallery para guardar imágenes en el dispositivo y NativeShare para compartir en redes las imágenes.

En todas ellas se seguirá el mismo proceso. Para empezar, hay que acceder a la Asset Store de Unity, buscar dichas librerías y hacer clic en Add to My Assets, y Open in Unity.

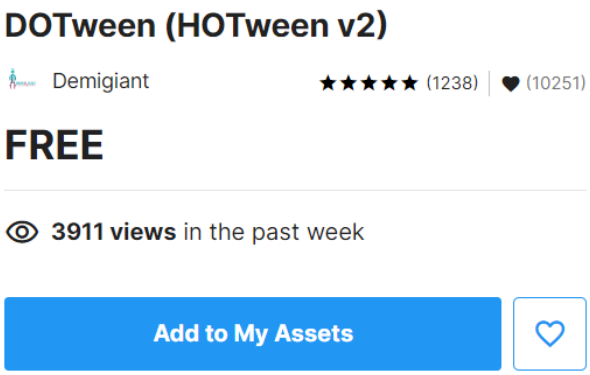

Figura 98. Instalación de librerías 1

Figura 99. Instalación de librerías 2

Por último, se hará clic en Download e Import.

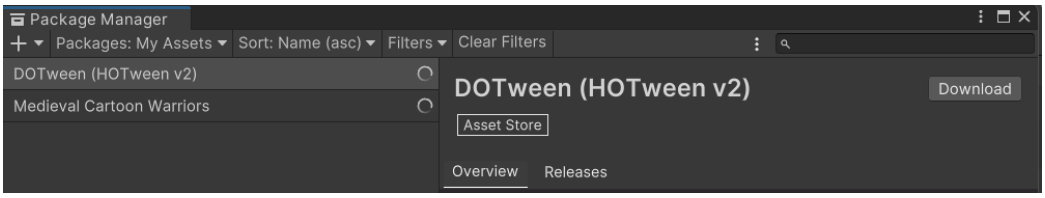

Figura 100. Descarga del paquete

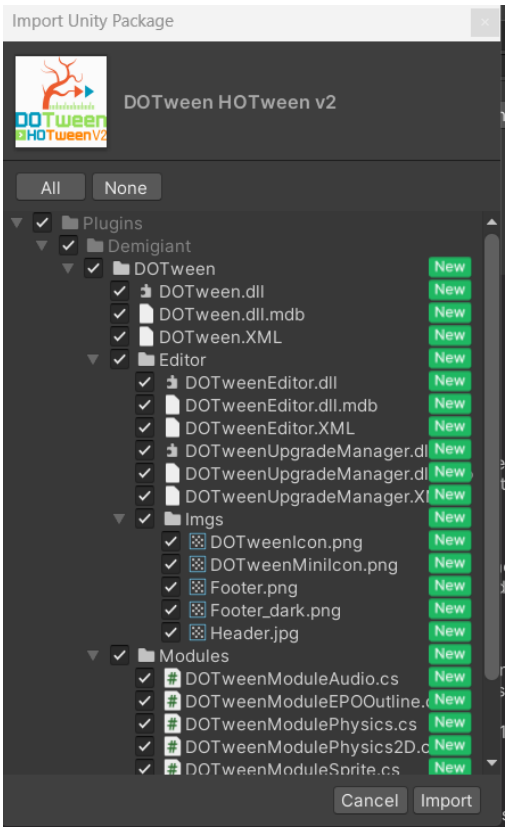

Figura 101. Importación del paquete

Una vez hecha esta configuración, se procederá a preparar nuestro dispositivo Android para llevar a cabo los testeos desde el propio Unity, teniendo acceso al log para comprobar los distintos mensajes de error que puedan aparecer.

Primero se debe activar el modo desarrollador. El método varía según el modelo, en este caso usaremos un Galaxy A51 así que se acceder a Ajustes > Acerca del teléfono > Información de software y en Versión de Kernel se pulsará tantas veces como sea necesario hasta que se active el modo desarrollador. Una vez activado, se deberá habilitar la opción de depuración por USB.

A continuación, se conectará el teléfono al ordenador, y se hará clic en File > Build and Run, seleccionando el teléfono. En cuanto finalice la compilación se ejecutará, y de esta forma se podrá llevar a cabo el testeo.

ANEXO 3. CREACIÓN BASE DE DATOS DE MARCADORES

Primero se debe obtener una fotografía del objeto con el que se desee interactuar con Realidad Aumentada, el cual actuará como marcador.

Figura 102. Imagen del marcador

A continuación, se accederá a la página de Vuforia en el apartado Developer y se hará clic en Target Manager. Una vez dentro se pulsará en Add Database y se establecerá el nombre de la base de datos.

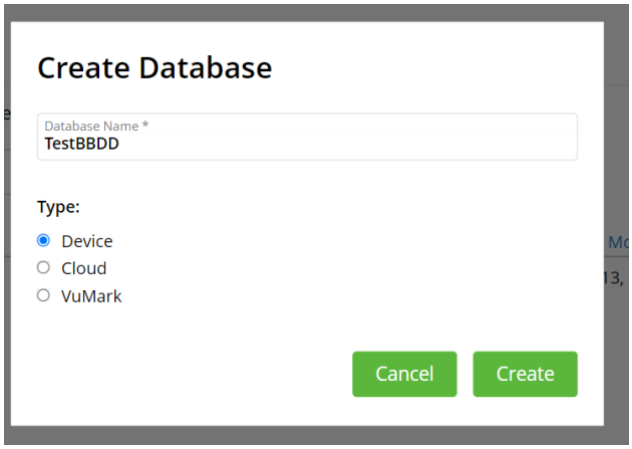

Figura 103. Creación de la base de datos

Una vez creada la base de datos, se hará clic en ella, y se añadirá un Target (objetivo). En este caso será la imagen anteriormente capturada, se identificará con un nombre y se establecerá el tamaño del objeto.

Figura 104. Adición del Target

Cuando la imagen se haya procesado, se hará clic en Download Database y se seleccionará Unity Editor.

Ahora que ya se tiene la base de datos descargada, se debe volver al proyecto de Unity y haciendo click derecho en Assets, se seleccionará la opción Import Package > Custom Package, y se abrirá la base de datos anteriormente descargada.

El siguiente paso es añadir el Target al proyecto. Para ello se deberá dirigir a GameObject > Vuforia Engine > Image Target, y se arrastrará la imagen importada en las opciones de Image Target.

A continuación, se moverá la Image Target dentro del componente ARCamera para poderlo detectar con la cámara del dispositivo móvil.

El último paso será descargar o crear un objeto de Realidad Aumentada y añadirlo dentro de la Image Target, para que cuando detecte esta imagen aparezca el objeto importado.

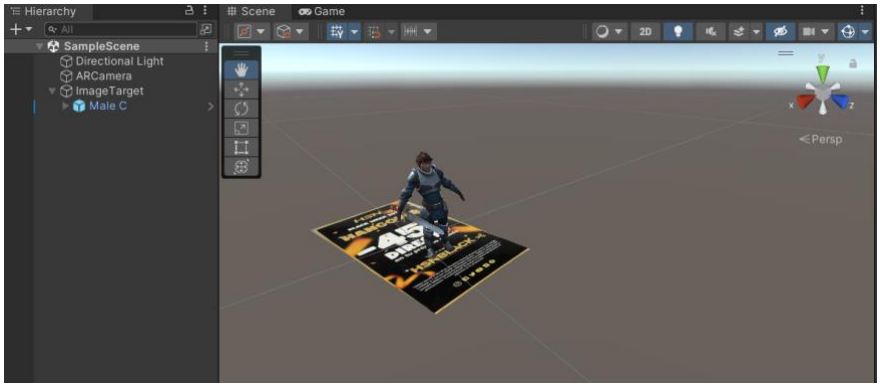

Figura 105. Inclusión de Image Target en ARCamera

ANEXO 4. MANUAL DE USO

Antes de comenzar a disfrutar de la aplicación, asegúrate de que tu dispositivo móvil cumple con los siguientes requisitos mínimos y que has otorgado los permisos necesarios:

Requisitos Mínimos

- Dispositivo Android con versión 7.0 (Nougat) o superior.
- Funcionalidad de GPS y brújula (giroscopio) para la navegación y la experiencia de realidad aumentada.
- Cámara trasera para capturar fotos y utilizar la realidad aumentada.

Permisos Requeridos

La aplicación solicitará los siguientes permisos para brindarte una experiencia óptima:

1. **Ubicación:** La aplicación necesita acceso a tu ubicación para proporcionar una experiencia de realidad aumentada basada en la geolocalización. Esto te permitirá interactuar con los marcadores y los aumentos de manera efectiva.
2. **Cámara:** Para capturar fotos y permitir la experiencia de realidad aumentada, la aplicación requerirá acceso a la cámara de tu dispositivo.
3. **Archivos:** La aplicación podría solicitar acceso a los archivos de tu dispositivo para guardar y gestionar las fotos que captures y edites durante el juego.

Cómo Otorgar Permisos: Al abrir la aplicación por primera vez, es posible que se te solicite otorgar los permisos necesarios. Asegúrate de seguir estos pasos:

1. Cuando se te pida, concede el acceso a la ubicación de tu dispositivo. Esto permitirá que la aplicación utilice la información de geolocalización para ofrecer la experiencia de realidad aumentada.
2. Asegúrate de otorgar permiso para acceder a la cámara. Esto es fundamental para capturar fotos y usar la realidad aumentada durante el juego.
3. Si se solicita, proporciona acceso a los archivos de tu dispositivo. Esto permitirá que las fotos capturadas se guarden y gestionen adecuadamente en la galería de la aplicación.

Nota Importante: La aplicación "Aventura en Cástulo" está diseñada para funcionar de manera óptima en dispositivos Android que cumplen con los requisitos mencionados y

que tienen la capacidad de soportar tecnologías de realidad aumentada. Si tu dispositivo no cumple con estos requisitos, es posible que la experiencia no sea completa o que ciertas características no estén disponibles.

Contenido

1. **Pantalla Principal:** En la pantalla principal, encontrarás seis botones que te llevarán a diferentes secciones de la aplicación:
    - **Empezar:** Inicia el juego en realidad aumentada. Sigue las instrucciones del guía Césaro para encontrar los aumentos.
    - **Fotos:** Explora y edita las fotos que has capturado durante el juego.
    - **Historia:** Lee sobre la historia de Cástulo y su contexto.
    - **Info:** Accede a los detalles y créditos de la aplicación.
    - **Ayuda:** Obtén información sobre cómo funcionan los botones de la aplicación.
    - **Salir:** Cierra la aplicación.

2. **Instrucciones del Juego:** Al pulsar "Empezar," accederás a las instrucciones del juego. Puedes navegar hacia la Pantalla Principal usando el botón de Atrás. Una vez que estés listo/a para comenzar el juego, pulsa "Siguiente."

3. **Realidad Aumentada:** Durante el juego, sigue las indicaciones de Césaro para encontrar los marcadores. Una vez que identifiques un marcador, toma una foto pulsando el botón "Capturar" para registrar el aumento encontrado. Para volver a leer las instrucciones pulsa el botón “Atrás”.

4. **Pantalla de Victoria:** Una vez que encuentres los seis aumentos, verás la pantalla de Victoria. Aquí, tienes dos opciones:
    - **Volver al Inicio:** Regresa al menú principal para explorar otras partes de la aplicación.
    - **Capturar Pantalla:** Guarda una imagen de esta pantalla para recordar tu logro.

5. **Fotos:** En la sección de "Fotos," podrás ver todas las fotos que has capturado durante el juego, para volver a la Pantalla Principal pulsa el botón de “Atrás”. También puedes realizar ediciones en estas fotos:
    - **Editar Foto:** Selecciona una foto para editarla. Ajusta filtros como blanco y negro, sepia, y negativo. Modifica el brillo, la saturación, el contraste y el tono.
    - **Guardar:** Una vez que hayas editado la foto a tu gusto, pulsa "Guardar" para conservar los cambios.
    - **Compartir:** Comparte tus fotos editadas en redes sociales y otras plataformas.
    - **Cerrar:** Vuelve a la sección de “Fotos” descartando los cambios de la edición.

6. **Historia de Cástulo:** En esta sección, podrás explorar la historia de Cástulo como si estuvieras leyendo un libro. Utiliza los botones "Siguiente" y "Atrás" para navegar por las páginas. Si deseas regresar a la Pantalla Principal, simplemente pulsa el botón “Atrás” en la portada del libro.

7. **Información de la Aplicación:** Accede a los detalles y créditos de la aplicación. Aquí encontrarás información adicional sobre la aplicación, la autora y la versión. Si deseas regresar a la Pantalla Principal, simplemente pulsa el botón “Atrás”.

8. **Ayuda:** En esta sección, encontrarás información detallada sobre cómo utilizar los botones. Asegúrate de revisar esta sección si tienes preguntas sobre cómo hacer uso de la ellos de manera efectiva. Si deseas regresar a la Pantalla Principal, simplemente pulsa el botón “Atrás”.

ANEXO 5. ENLACE DRIVE

Se añade el siguiente enlace a una carpeta en Drive, donde se han incluido dos vídeos explicativos de la aplicación en versión extendida y reducida, el código fuente y apk.

Enlace a la carpeta.