



UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

DESARROLLO DE APLICACIÓN AR PARA INTEGRACIÓN DE PERSONAJES HISTÓRICOS EN VISITAS CULTURALES

Alumno: Patricia Muñoz Antón

Tutor: Prof. D. Raúl Mata Campos

Depto.: Ingeniería de Telecomunicación

Junio, 2022



Universidad de Jaén

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

Trabajo Fin de Grado

DESARROLLO DE APLICACIÓN AR PARA INTEGRACIÓN DE PERSONAJES HISTÓRICOS EN VISITAS CULTURALES

Alumno: Patricia Muñoz Antón

MUÑOZ

ANTON

PATRICIA -

77377932K

Firmado
digitalmente por
MUÑOZ ANTON
PATRICIA -
77377932K

Fecha: 2022.06.17
19:20:20 +02'00'

Tabla de contenidos

1. Resumen	6
2. Introducción	8
Antecedentes	8
3. Objetivos e hipótesis	14
4. Estado del arte	15
Los tipos de interacción	15
Modos de interacción	17
Tipos de realidad aumentada según su contenido	17
Aplicaciones de la realidad aumentada	18
5. Desarrollo	24
Aplicaciones en el mercado actual	24
Comparativa de las distintas tecnologías disponibles.	28
Implementación de la aplicación	34
Módulo de almacenamiento	37
Módulo de localización	40
Sistema GPS	41
Implementación de localización	44
Módulo de orientación	50
Acelerómetro	51
Giroscopio	51
Implementación de orientación	53
Módulo de captación	59
Módulo de filtrado de imágenes	61
Captura de imagen	62
Filtrado	63
6. Resultados	74
Guiado del usuario	75
Interacción con el monumento	81
7. Líneas futuras	90
8. Conclusión	91
9. Bibliografía	92
Anexo 1. Manual de uso	97
Anexo 2. Instalación de Unity	107
Anexo 3. Instalación de Vuforia	119
Anexo 4. Prueba de funcionamiento	125

Tabla de ilustraciones

- Figura 1. Sensorama
- Figura 2. La espada de Damocles
- Figura 3. Videoplace
- Figura 4. KARMA
- Figura 5. ARToolKit
- Figura 6. ARQuake
- Figura 7. Marcador QR
- Figura 8. The Mandalorian AR experience
- Figura 9. Museum from Home
- Figura 10. Wallame
- Figura 11. Aplicación de las termas de l'Albir
- Figura 12. ARki
- Figura 13. Monster AR.
- Figura 14. CultureClic
- Figura 15. Layar
- Figura 16. Aplicación del cementerio de San Miguel
- Figura 17. Guideo
- Figura 18. Logotipo de Unity
- Figura 19. Logotipo de Unreal Engine
- Figura 20. Logotipo de Godot
- Figura 21. Logotipo de Viro React
- Figura 22. Logotipo de Aumentaty
- Figura 23. Diagrama de flujo
- Figura 24. Esquema de módulos
- Figura 25. Zoom del diagrama de flujo 1
- Figura 26. Clase de tipo Scriptable Object
- Figura 27. ScriptableObject resultante de tipo ruta
- Figura 28. ScriptableObject de tipo monumento
- Figura 29. Archivo cvs de rutas

Figura 30. Zoom del diagrama de flujo 2

Figura 31. Esquema sistema GPS

Figura 32. Constelación GPS

Figura 33. Función de localización

Figura 34. Función para cálculo de distancias

Figura 35. Ilustración de la fórmula de Haversine.

Figura 36. Función para el cálculo de la distancia con Haversine

Figura 38. Representación de los ejes cartesianos en un dispositivo móvil.

Figura 37. Zoom del diagrama de flujo 3

Figura 39. Esquema del giroscopio.

Figura 40. Función para la detección del monumento

Figura 41. Función para limitar los ángulos.

Figura 42. Función para la detección del sentido de giro.

Figura 43. Zoom del diagrama de flujo 4

Figura 44. Panel de configuración de la ARCamera

Figura 45. Función para finalización de ruta.

Figura 46. Zoom del diagrama de flujo 5

Figura 47. Función para captación de imágenes.

Figura 48. Esquema de las funciones de filtrado

Figura 49. Filtro de escala de grises

Figura 50. Comparación tras aplicar el filtro de escala de grises.

Figura 51. Filtro negativo.

Figura 52. Comparación de imágenes tras aplicar el filtro negativo.

Figura 53. Filtro sepia.

Figura 54. Comparación de imágenes tras aplicar el filtro sepia.

Figura 55. Filtro modificador de contraste.

Figura 56. Comparación de imágenes tras aplicar el filtro de contraste.

Figura 57. Corrección Gamma.

Figura 58. Comparación de imágenes tras aplicar la corrección gamma.

Figura 59. Ruta hacia Almendros Aguilar.

Figura 60. Ruta hacia Almendros Aguilar 2.

Figura 61. Cambio de interfaces 1.

Figura 62. Cambio de interfaces 2.

Figura 63. Paso de interfaz de guiado a interfaz al ubicarse cerca.

Figura 64. Cambio de detección de lejos a cerca del monumento.

Figura 65. Detección del frente.

Figura 66. Paso de la vista al frente a la vista hacia la izquierda.

Figura 67. Paso de la vista al frente a la vista hacia la derecha.

Figura 68. Imagen capturada con la cámara.

Figura 69. Guardado de imagen.

Figura 70. Generación de conversación.

Figura 71. Excepción de tamaño.

Figura 72. Selección de obra.

Figura 73. Cambio de obras.

Figura 74. Menú principal de la aplicación.

Figura 75. Mapa de Jaén.

Figura 76. Información de ubicación del monumento

Figura 77. Interfaz mostrada al ubicarse lejos de los monumentos.

Figura 78. Interfaz de guiado.

Figura 79. Interfaz mostrada al ubicarse cerca del monumento

Figura 80. Interfaz de filtrado de imagen.

Figura 81. Biografía del autor.

Figura 82. Obras del personaje histórico.

Figura 83. Información de la obra “El lagarto”.

Figura 84. Imagen de inicio de sesión de Unity.

Figura 85. Página de descargas de Unity 1.

Figura 86. Página de descargas de Unity 2.

Figura 87. Instalador de Unity.

Figura 88. Panel de versiones de Unity.

Figura 89. Panel de módulos de Unity.

Figura 90. Opción de instalación de Visual Studio.

Figura 91. Licencia de usuario.

Figura 92. Ampliación de módulos de Unity.

Figura 93. Creación de nuevo proyecto.

Figura 94. Elección del tipo de proyecto.

Figura 95. Interfaz de usuario de Unity.

Figura 96. Configuración de las características del proyecto.

Figura 97. Panel Build Settings.

Figura 98. Panel Manager

Figura 99. Panel Player Zoom.

Figura 100. Apartado de identificación de la aplicación.

Figura 101. Página de registro de Vuforia.

Figura 102. Instalador de Vuforia.

Figura 103. Interfaz de Unity

Figura 104. Instalación de Vuforia 1.

Figura 105. Instalador de Vuforia 2.

Figura 106. Panel Package Manager.

Figura 107. Configuración de Vuforia dentro de Unity.

Figura 108. Vuforia Configuration.

Figura 109. Creación de licencia de Vuforia.

Figura 110. Adquisición de la licencia de Vuforia

Figura 111. License Manager.

Figura 112. Licencia de Vuforia.

Figura 113. Incorporación de la licencia de Vuforia en Unity.

Figura 114. Perfil de la webcam.

Figura 115. Archivo webcamprofiles.xml 1.

Figura 116. Archivo webcamprofiles.xml 2.

Figura 117. Archivo webcamprofiles.xml 3.

1. Resumen

La tecnología ha permitido que el sector turístico evolucione para ofrecer más información a un mayor número de personas, de una manera más atractiva. Actualmente, el uso de los dispositivos móviles y los softwares de realidad aumentada y de geolocalización, han abierto un gran abanico de oportunidades a este sector. Es por ello por lo que el principal objetivo de este trabajo de fin de grado ha sido la creación de una aplicación funcional e intuitiva que permita al usuario obtener una experiencia turística más completa y enriquecedora sobre algunos personajes históricos relacionados con la ciudad de Jaén.

El funcionamiento de esta aplicación se basa en la utilización de métodos de geolocalización para situar al usuario en el lugar destinado a la incorporación de aumentos, concretamente en estatuas. La aplicación está pensada para dispositivos Android que contengan giroscopio. Para proporcionar la mayor información posible se utiliza un marcador que actúa como brújula. De este modo, dependiendo del ángulo de giro, se muestran distintos datos relevantes en formato escrito y audiovisual.

Palabras clave

Geolocalización, realidad aumentada, turismo, aplicación.

Abstract

Nowadays' technologies have contributed to the development of the tourist sector: it is instant, interactive and easier for the consumer to access the information. Also, augmented reality softwares, as well as mobile phones and geolocation offer a wide range of opportunities to the industry.

This is the reason why the main purpose of this Final Degree Project is to create a functional and intuitive mobile application which allows the user to have an enriched and complete experience when it comes to learning about historic local characters. This way, the application uses geolocation methods in order to locate the user in the spot where the augmentation will be placed, more specifically in statues. The software has been developed for Android devices with gyroscopes. A marker acting as compass is used to provide further information and different details are given according to the angular rotation, as relevant facts or sounds.

Key words

Geolocation, augmented reality, sightseeing, application.

2. Introducción

Siendo ésta una tecnología que se encuentra en pleno apogeo, no es de extrañar que exista la creencia errónea de que se trata de un concepto “actual” fruto del progreso tecnológico. Más bien tanto dicho concepto como su tecnología surgió antes incluso de que el hombre llegara a la luna.

Geolocalización y realidad aumentada son tecnologías que actualmente experimentan un crecimiento exponencial. Sin embargo, muchos de los usos de estas tecnologías están aún por explorar. Dado que el sector turístico las está implementando para crear una experiencia más inmersiva y cercana al público, se ha decidido crear una aplicación para dispositivos Android a partir de Unity, que permita a los usuarios conocer la vida y obra de algunos personajes célebres que tienen relevancia para la ciudad de Jaén, a partir de la estatua que les representa.

Para ello, expondremos los antecedentes que hemos tenido en cuenta para realizar este trabajo.

2.1. Antecedentes

Una primera idea de lo que hoy en día conocemos como realidad aumentada comienza a surgir a finales de la década de 1950 de la mano de Morton Heilig, quien diseñó la primera experiencia multisensorial virtual, llamada Sensorama, que proyectaba una película con imágenes estereoscópicas 3D mientras producía vibración, viento e incluso aromas, con el fin de crear una ilusión de la realidad. El juego dio al jugador la experiencia de montar una motocicleta en las calles de Brooklyn [1].



Figura 1. Sensorama

Posteriormente, en 1968, un equipo del Instituto de Tecnología de Massachusetts, liderado por Ivan Sutherland, llevó a cabo el primer casco de realidad virtual. Este estaba compuesto por un casco con lentes semitransparentes acoplado a un ordenador proyectando información. Dicha tecnología tenía como fin lograr “la pantalla definitiva” [2]. A este casco se le conoce como “la espada de Damocles”, debido al gran tamaño y peso del ordenador y del casco empleado, que se encontraba colgado del techo [3].

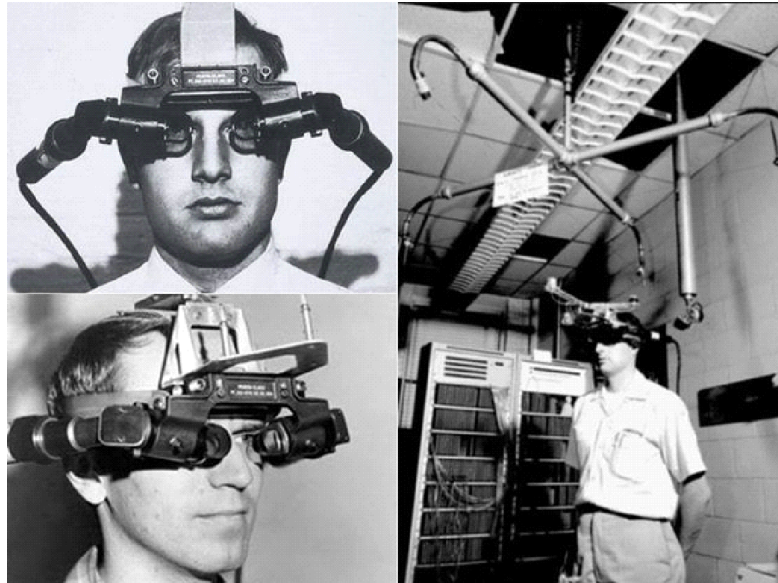


Figura 2. La espada de Damocles

En 1988, surge el proyecto Video Place, creado por el artista digital Myron Krueger, que trataba de utilizar la tecnología para rastrear las siluetas de las personas y generar efectos los cuales se proyectaban en una pantalla como si fuera un lienzo [2], permitiendo a los espectadores interactuar con los objetos virtuales en tiempo real.

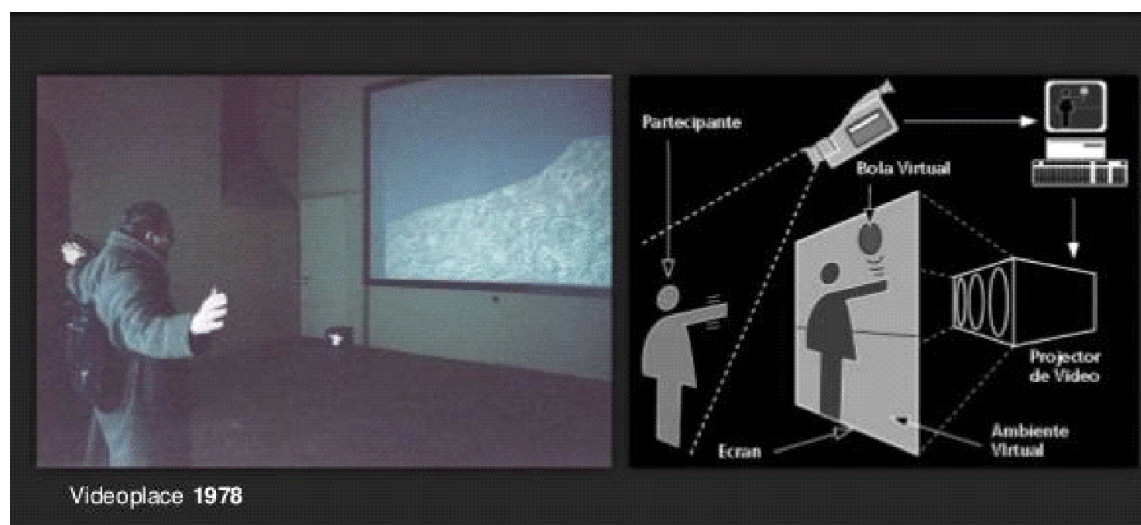


Figura 3. Videoplace

Sin embargo, no es hasta 1992 cuando Tom Caudell forja el concepto de lo que conocemos actualmente por realidad aumentada. En ese momento Caudell se encontraba trabajando en el proyecto Adaptive Neural Systems Research and *Development* de Boeing en Seattle cuando surgió la necesidad de encontrar un método más sencillo para ayudar tanto a la fabricación aviaría como a su proceso de ingeniería [4].

Ese mismo año, LB Rosenberg crea el primer sistema de funcionamiento de realidad aumentada para la Fuerza Aérea de los EE. UU. llamado Virtual Fixtures cuya finalidad era preparar a los pilotos. Tras ello se comenzó a desarrollar de forma más profunda y significativa, obteniendo dos años después el primer sistema de Realidad Aumentada en el prototipo llamado KARMA (según sus siglas en inglés, que significan Realidad aumentada basada en el conocimiento para la asistencia de mantenimiento). KARMA proyectaba una imagen en 3D para dar instrucciones al usuario de cómo tenía que recargar la impresora [4].



Figura 4. KARMA

En 1999, Hirokazu Kato (investigador del Instituto Nara de Ciencia y Tecnología) lanza en código abierto la herramienta ARToolKit, que da lugar a que esta tecnología obtenga una mayor difusión. Hasta ese momento, la realidad aumentada se basaba en un desafío con elementos voluminosos, *softwares* complejos y de gran costo que estaban al alcance de muy pocos. Estas librerías resolvían los dos principales problemas de la realidad aumentada, el seguimiento del punto de vista y la interacción con el objeto virtual, permitiendo el seguimiento de la captura de video del mundo real para combinar con la interacción de los objetos virtuales [5].



Figura 5. ARToolKit

Gracias a este avance, surge en el año 2000 el primer juego al aire libre de realidad aumentada de la mano de Bruce Thomas, ARQuake [6]. Este juego permitía a los usuarios movilizarse sin necesidad de *joystick*, requiriendo una *headset* conectada a un ordenador portátil y a giroscopios.



Figura 6. ARQuake

Tras ello han ido surgiendo en el mercado numerosas aplicaciones, siendo la más conocida Pokémon Go (2016) de Niantic. Este juego integra a los personajes de la mítica serie en el mundo real. Gracias a su éxito, se ha generado un mayor interés en este tipo de tecnologías.

3. Objetivos e hipótesis

El objetivo principal de este trabajo de final de grado es crear una aplicación móvil que permita al usuario obtener más información del entorno, en este caso, de los personajes ilustres de la provincia de Jaén. Asimismo, se pretende adquirir nuevos conocimientos en técnicas avanzadas de procesamiento de imagen/vídeo y de realidad aumentada AR.

El presente proyecto pretende desarrollar una herramienta para integrar en visitas culturales personajes históricos relacionados con el monumento, sala, etc. mediante AR con la imagen captada con el móvil para crear una sensación de inmersión más interesante y divertida.

Esta aplicación permite que, a partir de la utilización de diferentes marcadores, puedan componerse diferentes aumentos (con contenidos multimedia) para la recreación, el suministro de información cultural, realización de fotografías, etc.

La herramienta debe proporcionar una interfaz cómoda, intuitiva y sencilla de utilizar para facilitar su uso en el sector del turismo de ocio.

A partir de estos objetivos generales, se ha decidido concretarlo en los siguientes:

1. La aplicación debe detectar los distintos puntos gps implementados en ella.
2. Debe detectar de forma correcta la dirección a la que apunta el dispositivo, sabiendo diferenciar entre izquierda, derecha y el frente.
3. Si un punto no es detectado, la aplicación debe ser capaz de recuperarse al pasar por el siguiente punto.
4. Los aumentos deben mostrarse de forma correcta, cambiando la información mostrada dependiendo del monumento visitado.
5. La aplicación debe ser capaz de realizar fotografías y modificar dichas imágenes con diferentes filtros. Además, debe ser capaz de guardar la imagen mostrada.
6. La aplicación debe ser lo más inclusiva y cómoda posible, permitiendo que sea accesible para un mayor número de usuarios.
7. Por último, la herramienta debe ser lo más genérica posible con el fin de que pueda ser utilizada con distinta información realizando el menor número de cambios posibles.

4. Estado del arte

4.1. Los tipos de interacción

La siguiente clasificación está basada en las características funcionales de la herramienta, en relación con las distintas necesidades que pueda requerir.

Aumento basado en la activación

Este tipo de aumento reacciona a un estímulo o impulso, activando un objeto que se desea aumentar. Estos activadores pueden ser marcadores fijos sea papel como, por geolocalización, aumentos dinámicos de objetos además de una combinación de estos la cual conocemos como aumentado complejo [7].

Aumento basado en el reconocimiento de patrones y marcas

Utiliza marcadores de patrón de imagen física en un entorno específico en el que serán reconocidos por un *software* a través del *tracking*, (reconocimiento mediante la localización de puntos) para incorporar elementos virtuales al mundo real. Este tipo de marcador permite que el objeto ajuste su escala a partir del tamaño del disparador [8].



Figura 7. Marcador QR

Aumento basado en el movimiento corporal

Tiene como finalidad el reconocimiento de movimientos o gestos ya sea por ejemplo la detección de la posición del cuerpo o el reconocimiento del movimiento de unas manos [8].

Interacción multimodal

Estos sistemas incorporan varios modos de interacción combinados ya sea reconocimiento de gesto y voz, ofreciendo un reconocimiento más completo [8].

Aumento basado en el reconocimiento de imágenes (markerless)

Utiliza el reconocimiento de imágenes del entorno como elemento activador para incorporar el contenido virtual en el entorno “natural feature tracking” [9].

Aumento basado en la geolocalización

En este tipo de experiencias no se requiere de un marcador físico, sino que vincula una serie de datos a una ubicación específica. Estas suelen utilizarse generalmente en exteriores y con otras tecnologías como la brújula digital y el giroscopio [10]. Esta tecnología ha sido la utilizada en este trabajo final de grado.

4.2. *Modos de interacción*

En este punto tratamos las distintas maneras en las que se establece la comunicación persona-dispositivo, atendiendo al trabajo de Portalés Ricart.

Interacción con objetos virtuales/reales

Referenciado a lo que se conoce como Interfaz tangible de usuario (TUI), se trata de aquellas interfaces en las que el usuario interactúa con un objeto virtual realista que concuerda con el entorno real en el que se ubica. Un ejemplo de esto son aquellas aplicaciones cuya finalidad es la de permitir al consumidor hacerse una idea de cómo puede quedar un producto como una silla, en una habitación [8].

Interacción basada en la navegación

En este tipo de interacción, el usuario se ve guiado por el entorno, el cual no requiere de una reacción, sino que tiene como propósito únicamente la visualización del contenido. Un claro ejemplo podría ser aquellas aplicaciones que, dado un marcador, establecen un aumento inmutable. [8].

Interacción con otros usuarios

Las aplicaciones de realidad aumentada no tienen por qué estar diseñadas únicamente para ser empleado por un único consumidor, sino que puede este puede ser disfrutado por varios usuarios al mismo tiempo (entorno aumentado colaborativo) [8].

4.3. *Tipos de realidad aumentada según su contenido*

Realidad aumentada estática

En este tipo de realidad, los objetos se anclan a la escena siendo inmutables en el tiempo (no presentando ningún movimiento o acción). Textos, pistas visuales o modelos 3D cuya apariencia no varía durante la interacción del usuario con el entorno [11].

Realidad aumentada dinámica

Por el contrario, en la realidad aumentada dinámica, los contenidos si sufren una variación, independientemente de si el usuario interacciona con ello. En esta categoría podemos incluir tanto los videos como las animaciones sufren cambios al paso del tiempo debido a su movimiento [11].

4.4. *Aplicaciones de la realidad aumentada*

Hoy en día se puede encontrar bastantes aplicaciones de esta tecnología en el mundo real, desde su uso en medicina con plataformas para el aprendizaje de una cirugía o en educación con aplicaciones para el control de asistencia, o aquellas para la asimilación de conceptos teóricos tales como aquellas sobre el cuerpo humano o el sistema solar, con el fin de llegar al interés del estudiantado, haciendo del estudio de las materias algo más seductor para el alumno.

En este caso se ha decidido ver algunas de las distintas aplicaciones posibles correspondientes al sector cultural.

Cine

Con el gran interés generado por ambos temas (Realidad aumentada y cine), no es de extrañar que comenzasen a unirse. Un ejemplo de ello es una de las series del momento de la plataforma Disney Plus, The Mandalorian. Google, en colaboración con Disney y Lucasfilm ha sacado una experiencia basada en esta popular serie, en la que el usuario se encontrará mano a mano con Mando y el niño, interactuando con sus personajes favoritos, viviendo escenas de la primera temporada como un personaje más del elenco. Para lograr dicha interacción, este

programa incorpora ARCore de google empleando un sistema de oclusión mediante API ARCore Depth [12] .

Actualmente se encuentra disponible para dispositivos Google Pixel 5G, además de para unos pocos usuarios Android que tengan implementado 5G en sus dispositivos.

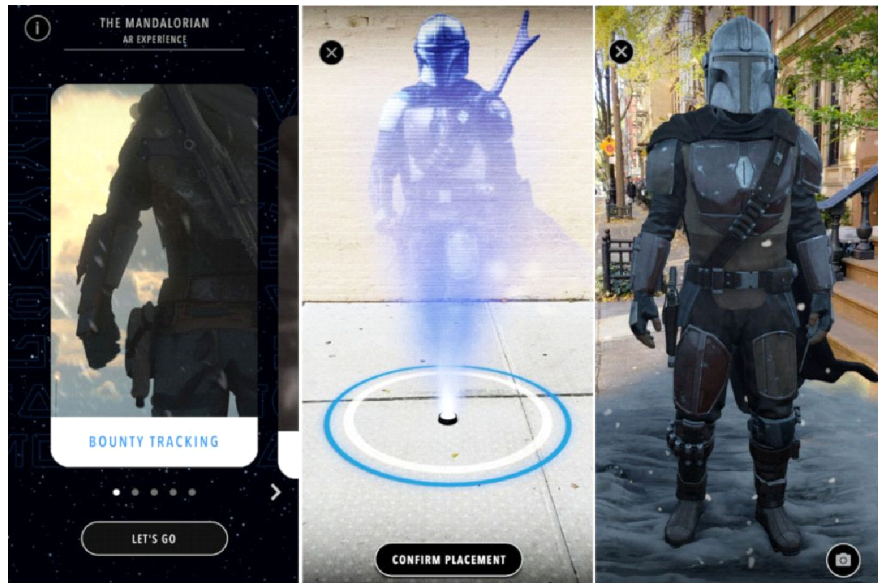


Figura 8. The Mandalorian AR experience

Museos Virtuales

Con la llegada del COVID-19 y su posterior confinamiento, muchos museos tuvieron que reinventarse para dar entretenimiento cultural a un público necesitado de pasatiempos. Museos como el Metropolitan Museum of Art o el Getty Center pusieron a disposición diversas obras, como la emblemática La noche estrellada de Vincent van Gogh. A través de la app Museum from Home el usuario puede vivir la experiencia de estar en un museo desde la comodidad del hogar, permitiendo observar dichas obras con mayor lujo de detalle [13] .

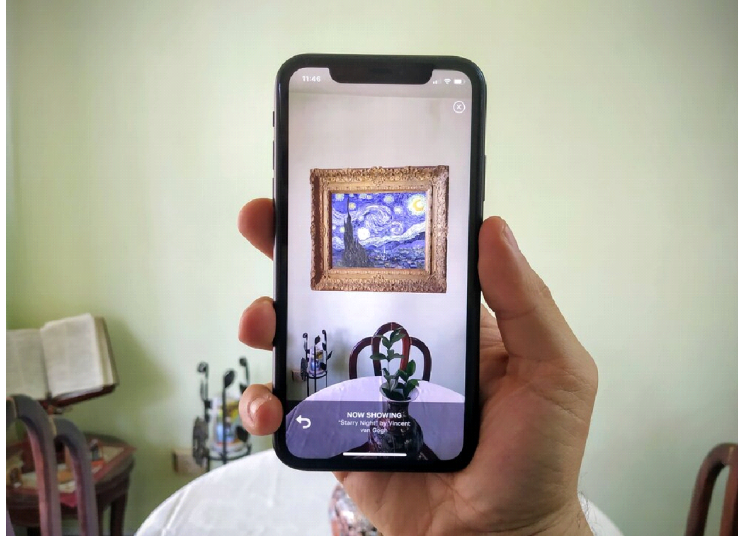


Figura 9. Museum from Home

Realidad aumentada como lienzo

No solo podemos utilizar la Realidad Aumentada para ver el arte ajeno, sino que permite que el propio usuario pueda dar rienda suelta a su imaginación. Una de las app que permiten al usuario poder compartir sus dibujos es Wallame. Se trata de una app gratuita que permite compartir dibujos en realidad aumentada con amigos. El funcionamiento es muy sencillo, el usuario realiza una foto del sitio en el que se encuentra. Después incluye dibujos, imágenes, pegatinas, mensajes, etc y por último lo comparte con sus amigos o con la comunidad. Este dibujo queda como un aumento anclado en la ubicación específica, por lo que siempre que pase por ese lugar, aparecerá ese aumento [14] .

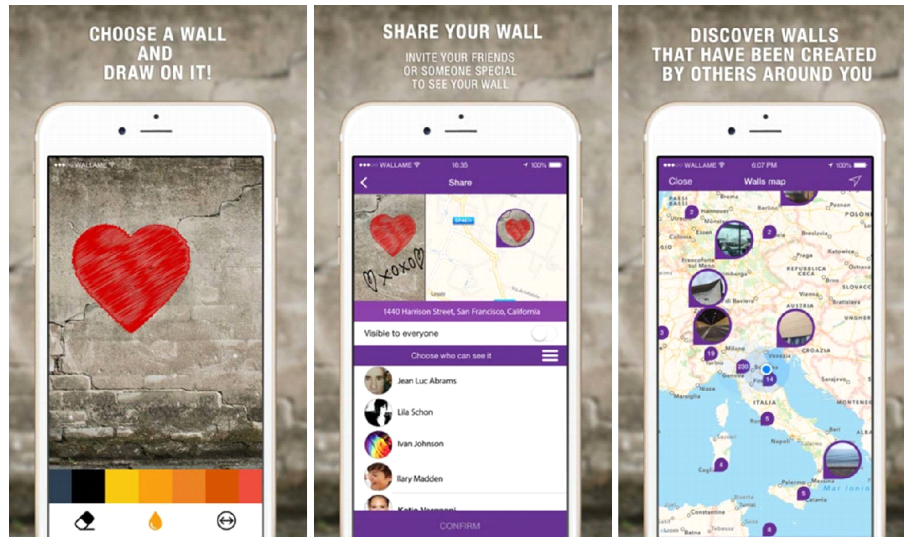


Figura 10. Wallame

Arqueología

Uno de los principales retos con el que se encuentran los divulgadores culturales es conseguir el interés e implicación del visitante, con la finalidad de que este último adquiera un mayor conocimiento sobre el tema al acabar la visita. Es por ello que el uso de herramientas como la Realidad Aumentada hace más atractiva la visita, consiguiendo que el usuario preste mayor atención [15].

Un lugar donde han combinado la Realidad Aumentada con la arqueología es en las termas de la villa romana de l'Albir, los cuales mediante una guía en realidad aumentada, como relata una trabajadora del lugar “puedan los visitantes comprender cómo eran las termas hace 2000 años.” [16].

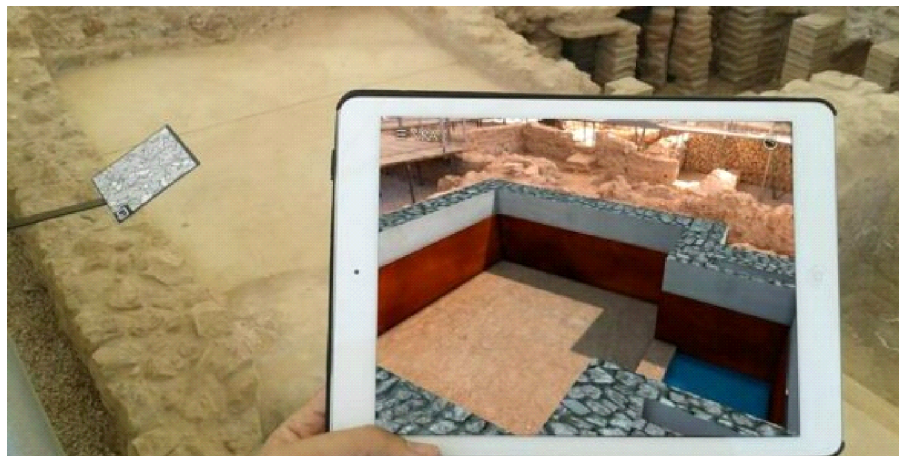


Figura 11. Aplicación de las termas de l'Albir

Arquitectura

En este ámbito es especialmente interesante ya que permite al cliente poder visualizar de forma más gráfica cualquier proyecto, obteniendo una mayor capacidad de detalle. Un ejemplo de ello es la aplicación ARki, una plataforma para la visualización en tiempo real de modelos arquitectónicos. Esta aplicación permite visualizar modelos 3D para obtener una visualización más inmersiva [17].

Actualmente se encuentra disponible tanto para usuarios de Android como de IOS que permite datos en 3D.



Figura 12. ARki

Música

Posiblemente sea uno de los sectores donde podemos encontrar un mayor desarrollo. Desde aplicaciones para realizar un concierto a través de diferentes instrumentos virtuales, como propuso un compañero de nuestra escuela, apps como Quarantour para vivir conciertos en directo del grupo Real Estate, hasta la integración de aumentos en los álbumes musicales. [18] Este es el caso del grupo MooA, que a través de la aplicación Monster AR, dan la posibilidad de escuchar música mientras la animación de un monstruo aumentado baila al son de la canción, inspirándose en el mito del laberinto de Creta. Cabe destacar que dependiendo del lugar donde se realice el aumento, cambiará la forma de percepción de la música. Como bien explican en su página web “MooA ha creado la primera portada inmersiva de la historia de la música. Una experiencia de

realidad mixta en la que, gracias a las gafas Magic Leap, se entra en el laberinto y se interactúa con los sonidos posicionados en el espacio” [19] .

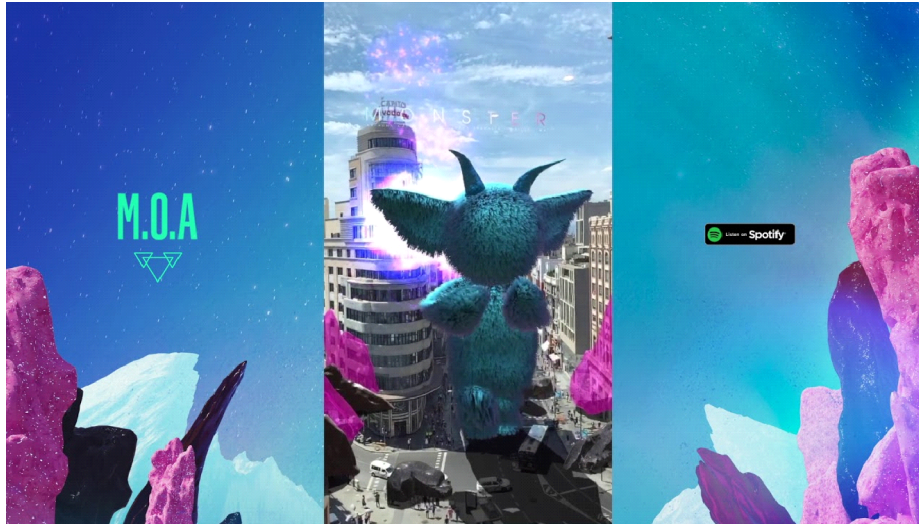


Figura 13. Monster AR

5. Desarrollo

En este siguiente apartado se realiza un estudio más exhaustivo de la aplicación desarrollada, investigando las aplicaciones existentes en el mercado (cuya finalidad es similar al propuesto en este trabajo final de grado), las herramientas disponibles actualmente para llevarla a cabo y la implementación finalmente realizada.

5.1. Aplicaciones en el mercado actual

En la actualidad se encuentra en el mercado gran variedad de aplicaciones que utilizan realidad aumentada en el ámbito cultural, como ya se ha podido ver anteriormente. En este punto se abordará las distintas aplicaciones existentes en el mercado ligadas al objeto de este trabajo final de grado.

CultureClic

Se trata de la mayor aplicación móvil de cultura existente actualmente en Francia. En este caso, se utiliza la realidad aumentada para mostrar los distintos puntos geolocalizados de mayor interés, principalmente de París. Esta app permite descubrir todas las localizaciones ubicadas en

un radio establecido por el usuario. Desde ella, permite acceder a más de 900 obras de arte y a la agenda cultural del lugar. Además, permite al usuario redescubrir los distintos lugares visitados viajando al pasado gracias al catálogo de la Biblioteca Nacional de Francia (BNF) y a la Agrupación de Museos Nacionales (RMN) [20].

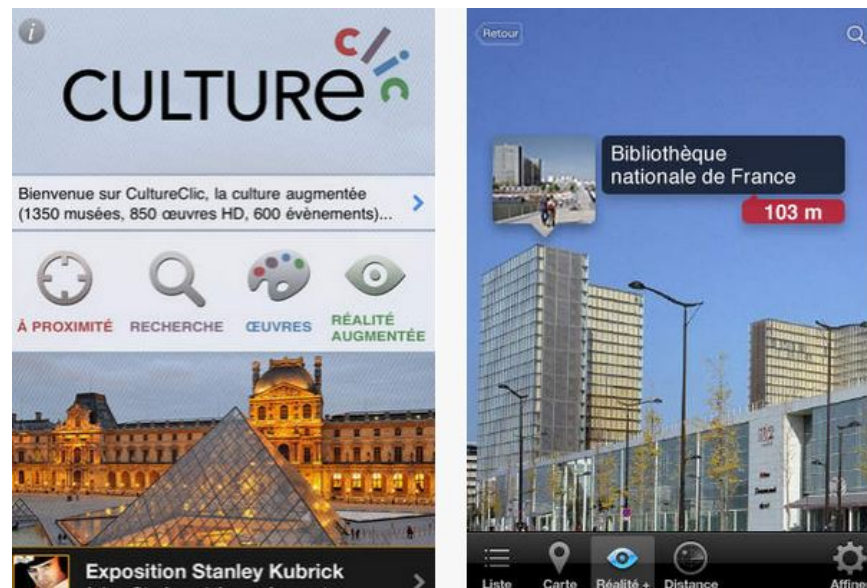


Figura 14. CultureClic

Layar

Con el fin de promover la cultura en España, el Ministerio de Educación, Cultura y Deporte, a través de la Sociedad Estatal para la Gestión de la Innovación y las Tecnologías Turísticas (SEGITTUR) han creado una aplicación con toda la oferta cultural de España mediante geolocalización [21].

Esta aplicación de características similares a la aplicación francesa CultureClic, ofrece más de 8000 referencias culturales de toda España (arquitectura, monumentos, fiestas tradicionales, museos, obras de arte...), permitiendo por ejemplo, elegir entre distintos estilos artísticos. Su radar señala los puntos ubicados en un radio de hasta 5 km. Una vez el usuario se encuentra ubicado ante un emplazamiento, se mostrará por pantalla el nombre, la dirección, el número de teléfono y el e-mail.

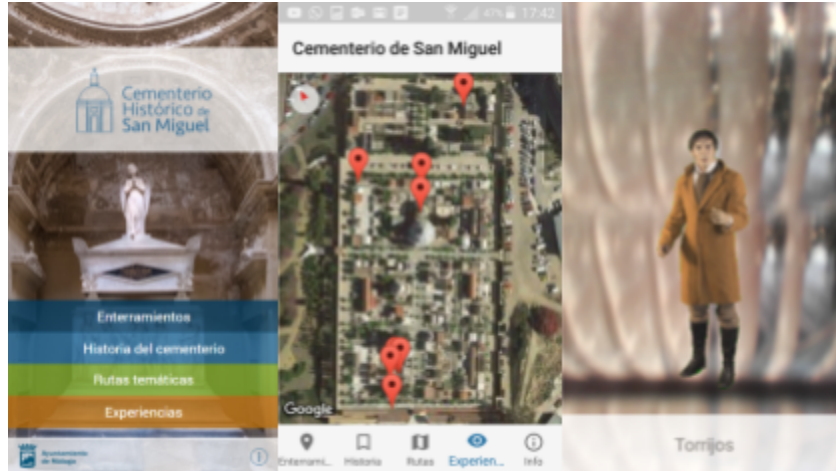


Figura 16. Aplicación del cementerio de San Miguel

Guideo

Guideo es una aplicación para la inclusión de Realidad Aumentada en el turismo que funciona por geolocalización y que guía a los usuarios mediante rutas a los distintos puntos de interés turístico. Permite contemplar en 3D monumentos ya desaparecidos, personajes históricos, leyendas... Si el usuario lo desea, Guideo proporciona narraciones con material multimedia de interés para obtener una historia más completa. Actualmente se encuentra disponible para las ciudades de Cádiz, Málaga y Sevilla [23].



Figura 17. Guideo

Tras analizar las distintas aplicaciones existentes en el mercado, se puede comprobar que esta aplicación incluye la localización mediante un radio de detección, aunque este ha sido implementado de forma distinta. Además, esta aplicación incorpora el guiado mediante una ruta prefijada la cuál va indicando el camino a seguir de forma dinámica. También se ha de remarcar que la aplicación creada incluye distintos materiales multimedia, equiparando a las actuales en el mercado (no tanto a cantidad), además de la posibilidad de realizar fotografías, con la posibilidad de tratar la imagen con los distintos filtros implementados, haciendo de esta una aplicación competitiva en el mercado actual.

5.2. Comparativa de las distintas tecnologías disponibles.

En este apartado se comentará brevemente acerca de los distintos motores de juegos existentes en el mercado, así como un análisis más detallado acerca de cada uno de ellos con el fin de dar con la herramienta que ofrezca unas mejores propiedades para el desarrollo de este proyecto.

Unity



Figura 18. Logotipo de Unity

Unity es el motor de desarrollo en tiempo real más conocido hasta el momento con “casi 1500 millones de creadores activos” según reporta la plataforma, apareciendo en más del 50% del catálogo de videojuegos para dispositivos móviles, siendo de un 90% para aplicaciones relacionadas con la realidad aumentada y virtual. Además, con esta herramienta no se requiere de un gran nivel de conocimientos para poder desarrollar aplicaciones. Es por ello que es cada vez más usual encontrarlo tanto en colegios como universidades para tomar un primer contacto con el desarrollo de aplicaciones.

En lo referente al lenguaje de programación empleado, Unity emplea el lenguaje C sharp, comúnmente conocido como C#, el cual es un lenguaje scripting orientado a objetos. Fue diseñado en 2000 por Microsoft el cual fue estandarizado posteriormente. Este lenguaje tiene algunas similitudes con C++ y Java, pero de menor complejidad.

Además, como en la propia web de Unity explican “es un lenguaje administrativo”, por lo que administra la memoria automáticamente por tí, realizando por ejemplo, asignaciones o cancelaciones de memoria.

Una de las grandes ventajas de esta plataforma es la numerosa comunidad que tiene detrás, teniendo a disposición del desarrollador una gran cantidad de documentación y foros donde poder apoyarse durante el desarrollo de su aplicación.

Otra de las grandes ventajas es el gran soporte multiplataforma que Unity presenta: para dispositivos de escritorio, de AR o VR, para consolas, teléfonos o web, conformando entre ellas más de 20 plataformas.

En cuanto al coste de esta aplicación, dependiendo del fin con el que se realice y de los recursos necesarios, podrán oscilar sus precios, siendo el menor 399\$ al año, hasta 4000\$/mes por cada 20 puestos. Cabe destacar que si la empresa tiene unos ingresos inferiores a 100.000\$ al año o si eres estudiante, podrás conseguir este motor de forma totalmente gratuita.

Apps como Pokemon Go, Call of Duty o Mario Kart presentan el distintivo logo de la marca.

Unreal Engine



Figura 19. Logotipo de Unreal Engine

Unreal es uno de los motores de desarrollo en tiempo real más longevos existentes hasta la fecha con características similares a Unity. Creado por Epic Games, este motor de desarrollo surgió tras el famoso juego “Unreal” en 1999. Uno de los usos que está recibiendo actualmente es para la creación de entornos para la formación de astronautas de la NASA.

Este permite elegir entre dos distintos lenguajes de programación: C++ y Blueprint Visual Scripting, pudiendo funcionar este último como un complemento de C++. El Blueprint es un lenguaje de programación propio de Unreal orientado a personas con un bajo nivel de programación, iniciando a estas en este mundo a través de juegos básicos.

A diferencia de Unity, Unreal Engine soporta 18 plataformas, siendo iOS y Android algunas de ellas.

La curva de aprendizaje que presenta Unreal es más pronunciada a la que puede presentar Unity, aunque esta plataforma intenta solventar dicha dificultad proporcionando una documentación más completa para hacer más accesible la plataforma.

Actualmente se encuentra de forma gratuita y a disposición de cualquier desarrollador siempre y cuando el beneficio obtenido no supere los 3000\$ el trimestre. Al superar dicho valor, el desarrollador deberá proporcionar a Unreal Engine el 5% de los ingresos obtenidos por .

Otras aplicaciones como Star Wars Jedi: Fallen Order o Borderlands 2 han sido creadas con esta plataforma.

Godot



Figura 20. Logotipo de Godot

Al contrario de los otros motores presentados, Godot es una aplicación de código abierto, bajo licencia permisiva MIT, por lo que se puede obtener de forma completa gratuitamente.

Una de las mayores desventajas presentada por Gadot es el número de plataformas soportadas por el motor, siendo este de solo 6 plataformas.

Un punto a su favor es el gran abanico de posibilidades en cuanto a la hora de emplear un lenguaje de programación, incluyendo los empleados en los otros motores C++ y C#. Cabe mencionar que dicho motor posee su propio lenguaje scripting, Script, orientado al igual que los Blueprints para personal sin experiencia en desarrollo.

ViroReact

Viro React es una plataforma para desarrolladores de código abierto para crear aplicaciones nativas de realidad virtual y aumentada empleando React Native.

Esta plataforma soporta tan solo 4 plataformas: iOS, Android, Daydream y Gear VR.

Al ser una plataforma de código abierto, se puede obtener de forma gratuita a través de su página web. Además, permite que pueda reutilizarse código ya implementado.

Algo a destacar es que el lenguaje que emplean es sencillo, basado en una mezcla entre lenguaje de marcas, como XML y Javascript a través del Framework de React Native.



Figura 21. Logotipo de Viro React

Aumentaty

Con origen en Valencia, esta plataforma está centrada en el desarrollo de aplicaciones y proyectos específicos de realidad aumentada de implementación sencilla y de carácter gratuito. Actualmente se encuentra contenida en Creator y Scope, siendo estas pertenecientes a Android e IOS. Para poder acceder a las aplicaciones creadas con esta herramienta, es necesario estar registrado previamente en su aplicación específica, aunque dichas aplicaciones pueden ser tanto de carácter público como privado. Es por ello que al presentar tan corto alcance, se ha desechado esta opción.



Figura 22. Logotipo de Aumentaty

Resumen del contenido

En la siguiente tabla se recoge de forma resumida las características de los distintos motores estudiados.

	Unity	Unreal	Godot	ViroReact
Precio	Gratuito/ Comercial	Gratuito/ Comercial	Gratuito	Gratuito
Plataformas soportadas	+20	18	6	4
Lenguaje de programación empleado	C#	C++/Blueprints	GdScript, Visual Scripting, C# y C++	JavaScript
Nivel previo	Baja/ media	Media/ Alta	Baja	Media/Baja
Recursos adicionales	Asset Store	Unreal Engine Marketplace	No	No

Se ha optado por no crear mi propio motor debido a lo que ello conlleva (creación de sus físicas desde cero, librerías, etc.) ya que no se dispone del suficiente tiempo para poder implementarlo desde cero.

Atendiendo a las distintas plataformas vistas, en este TFG se ha decantado por el uso de Unity ya que la universidad provee una licencia gratuita de la plataforma así como por la posibilidad de encontrar una documentación más completa y numerosa apoyada por diversos foros en los que poder tener una idea más clara de las distintas posibilidades que ofrece el motor. Además, al ser multiplataforma, permite llegar a más dispositivos, amplificando su difusión.

5.3. Implementación de la aplicación

Como ya se ha ido mencionando en apartados anteriores, el objeto de este proyecto es la visita guiada a través de los personajes históricos mediante realidad aumentada. Así, esta aplicación detecta la ubicación del usuario y lo guía bien mediante indicaciones establecidas por los distintos puntos que conforman la ruta o bien por la redirección a Google Maps hasta los

distintos puntos de interés. Una vez llegado, guiará al usuario a mirar hacia el monumento y dependiendo del sentido en el que esté mirando en ese momento (izquierda, derecha o de frente), mostrará distintos aumentos. Además el usuario podrá realizar fotografías y aplicarle varios filtros a ellas para almacenar un recuerdo de esa visita.

La funcionalidad de la aplicación queda resumida en el siguiente diagrama de flujo.



Se ha dividido la siguiente aplicación en diferentes módulos con funcionalidades concretas con el fin de simplificar la comprensión del funcionamiento del sistema. Estos módulos son:



Figura 24. Esquema de módulos

5.3.1. Módulo de almacenamiento

Este módulo se encarga de almacenar todos aquellos datos que no derivan de otros, que son invariantes en el tiempo tales como la localización de los distintos monumentos, la biografía de los personajes, las obras... Gracias a este módulo se obtendrán dichos datos de forma eficaz sin que ello suponga un gran gasto en memoria. Para ello los datos se almacenarán dentro de objetos de la clase Scriptable Object.

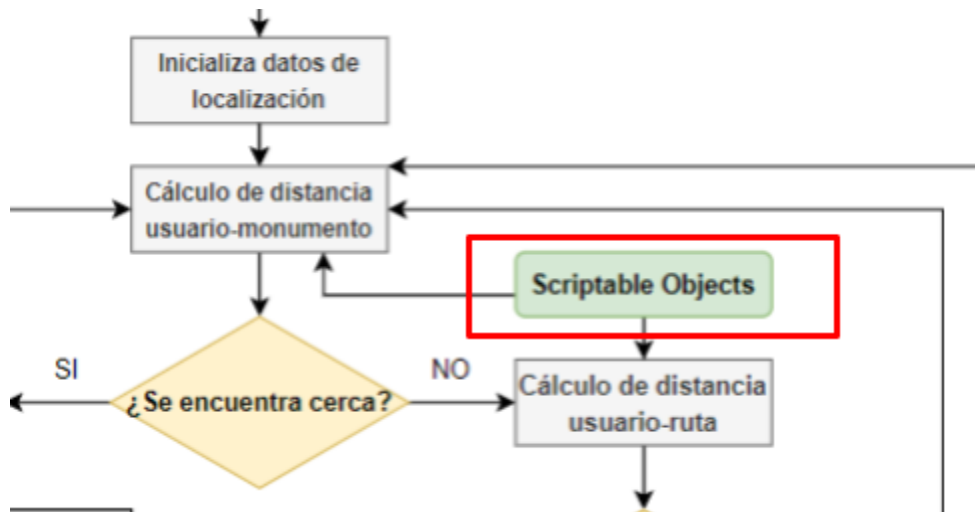


Figura 25. Zoom del diagrama de flujo 1

La clase Scriptable Object es un contenedor de datos proporcionado por Unity que sirve para almacenar grandes cantidades de datos, independientemente de las instancias de secuencia de proyectos, lo que provoca la correspondiente reducción del uso de memoria ya que evita copiar los valores [24].

Se ha creado una script en la cual se ubicará los distintos datos referente a cada personaje histórico en el siguiente caso representado:

```
public class GeneralInfoSO : ScriptableObject
{
    public int id;
    public string titulo;
    public string imagenEstatua;
    public string direccion;
    public float latitud;
    public float longitud;
    public int brujula;
    public string direccionSiguiente;
    public string nameMap;
    public string obra1;
    public string audio1;
    public string descripcion1;
    public string obra2;
    public string audio2;
    public string descripcion2;
    public bool visto;
    public bool izqCheck;
    public bool drchCheck;
}
```

Figura 26. Clase de tipo Scriptable Object

Cuando se almacenen los datos dentro de los Scriptable Objects tendrán como resultado la siguiente forma:

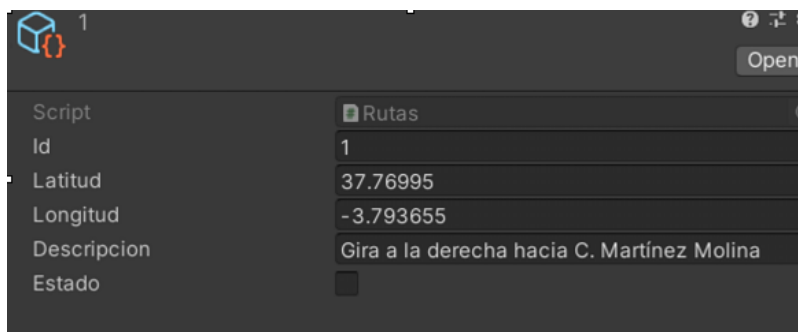


Figura 27. ScriptableObject resultante de tipo ruta

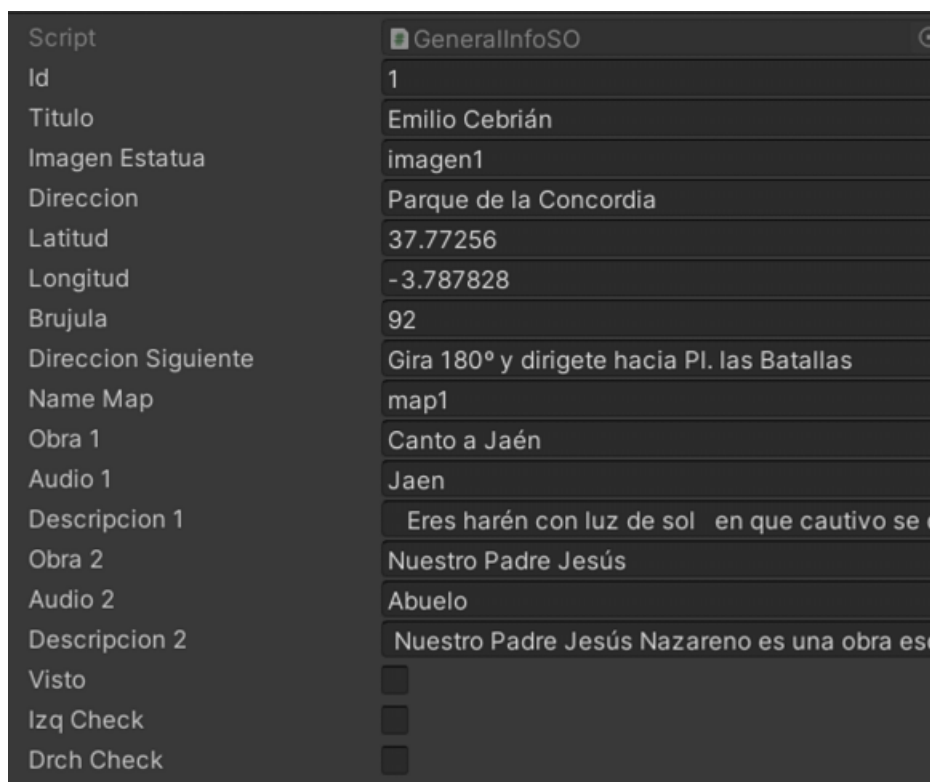


Figura 28. ScriptableObject de tipo monumento

Para almacenar dichos datos se podría hacer directamente desde Unity, creando cada objeto para cada situación o personaje y escribiendo dentro de él la información precisada para cada variable. A fin de realizar este proceso de forma automática se ha generado la script

CSVtoSO la cual parsea el archivo csv mediante el separador de comas y asocia la distinta información a los variables que componen las script de tipo Scriptable Object.

Se ha optado por este formato ya que se puede exportar de entornos conocidos como excel, proporcionando un trabajo más visual y sencillo, en el que indicamos aquellos datos que se desean almacenar, tales como las coordenadas en las que se ubica cada monumento, las distintas coordenadas de ruta, etc . Además, en el caso de la generación de rutas, se puede generar de forma sencilla desde aplicaciones como My Google Maps y exportandolo a dicho formato.

Se ha de tener en consideración que, dependiendo del entorno en el que se trabaje, se considerará el punto o la coma para indicar números decimales, por lo que se opta por separar con punto y coma y posteriormente, mediante código, reemplazarlo por punto una vez se va a almacenar los datos en los distintos contenedores.

Los documentos csv utilizan las comas para separar entre distinto contenido por lo que, si se separasen los números decimales con coma dentro del archivo, al descargarlo con csv lo detectaría como dos datos diferentes.

A	B	C	D
Ruta	lat	lon	Descripcion
1	3.777.269	-378.849	Dirigete al oeste hacia Pl. las Batallas y recorre la rotonda.
2	3.777.286	-378.888	Gira a la derecha hacia C. Cruz Roja Española
3	3.777.256	-379.056	Gira a la izquierda hacia C. del Arquitecto Berges
4	3.777.234	-379.045	Gira a la derecha hacia C. Rey Alhamar
5	3.777.162	-379.178	Gira a la derecha hacia C. Millán de Priego
6	377.717	-379.189	Gira a la izquierda hacia Plaza del Dr. Blanco Najera
7	3.777.139	-37.918	Gira a la derecha hacia Plaza del Dr. Blanco Najera
8	377.714	-379.204	Continúa por C. San Andrés
9	3.777.047	-379.372	Gira a la izquierda hacia C. Martínez Molina
10	3.777.026	-379.353	Gira a la derecha hacia C. San Antón y sigue recto hasta P. de San Juan

Figura 29. Archivo csv de rutas

5.3.2. Módulo de localización

La finalidad de este módulo consiste en la obtención de la posición del usuario en tiempo real respecto a un punto concreto, bien sea a los distintos monumentos o bien a los distintos puntos de ruta (puntos gps para guiar al usuario hacia el monumento), ya que estos actúan como marcadores que activarán los objetos pertinentes.

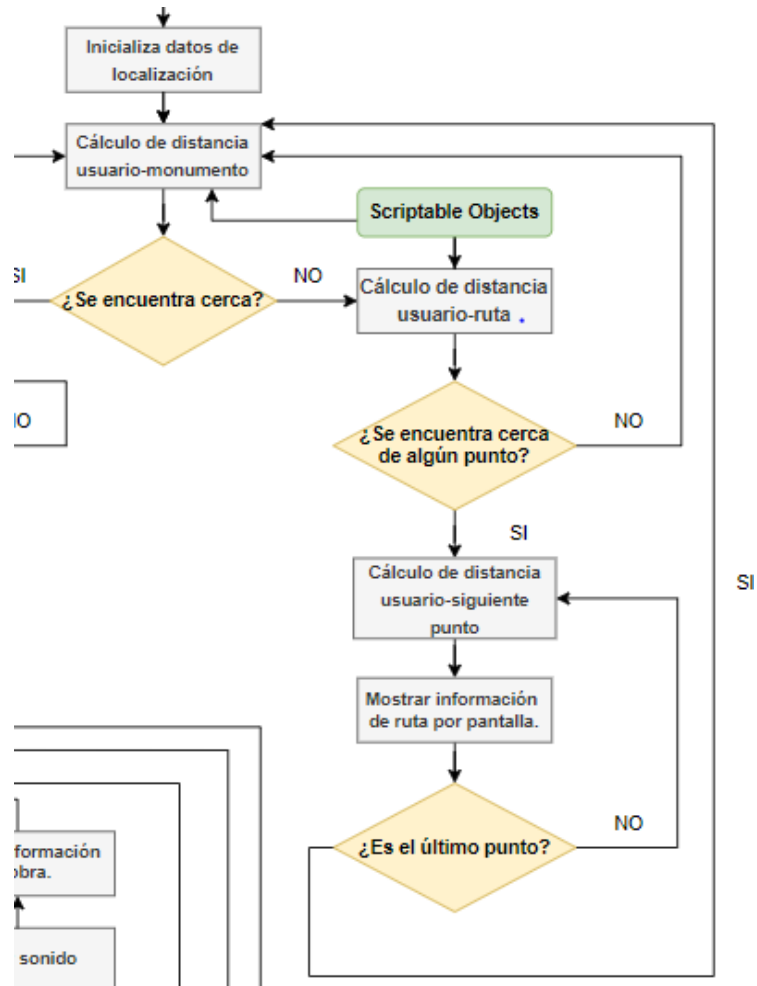


Figura 30. Zoom del diagrama de flujo 2

Para ello se vale del sensor GPS incorporado actualmente en la gran mayoría de los dispositivos móviles para captar los datos de localización.

Es interesante conocer cómo se obtiene la información recogida por el sensor GPS por ello, aunque no sea parte de la implementación de la aplicación, sí que interviene en su funcionamiento.

5.3.2.1. *Sistema GPS*

El sistema GPS (Global Positioning System) como bien indica sus siglas es un sistema de posicionamiento global que utiliza la tecnología de los satélites para ubicar al usuario en un punto concreto del planeta.

Elementos del sistema

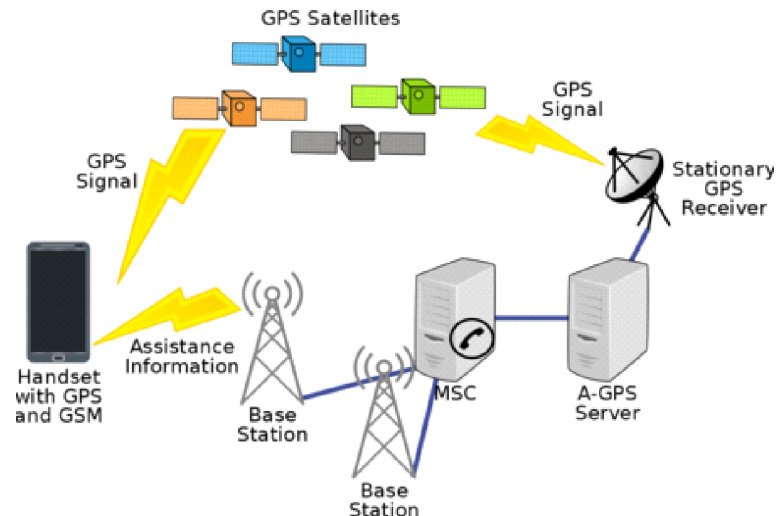


Figura 31. Esquema sistema GPS

Este sistema está compuesto por los siguientes segmentos:

Sistema satelital

Está conformada por la constelación de satélites Navstar. Consta de 24 satélites operativos (aunque en la actualidad tienen 31 satélites en órbita) y 4 de reserva colocados en 6 planos orbitales equiespaciados en longitud, con una inclinación respecto del plano ecuatorial de 55 grados y ubicados a una altura de 20.180 km [25]. Estos satélites emiten desde el espacio

señales codificadas con ruido aleatorio o señales de alcance con las que se realizan las mediciones de alcance, convirtiéndose en un sistema pasivo [26].

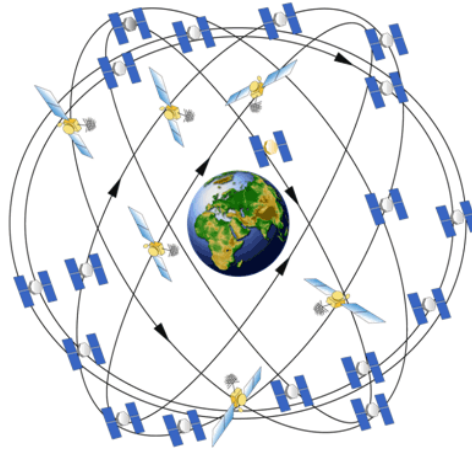


Figura 32. Constelación GPS

Sistema de control

Está formado por un conjunto de estaciones, localizadas a lo largo del mundo, encargadas del mantenimiento de los satélites y de su buen funcionamiento. Una de las tareas que realiza es el mantenimiento de los satélites en la posición orbital asignada. El sistema de control (SC) actualiza cada reloj satelital, efemérides (representación quasi-Kepleriano de la órbitas de los satélites GPS), almanaques (subconjunto de las efemérides de menor precisión) y demás datos al menos una vez al día. [26] Conforme mayor número de actualizaciones, se obtendrá un menor error de alcance.

Segmento de usuario

Formado por los receptores y por la comunidad de usuarios del sistema, estos procesan las señales transmitidas por los satélites en posición, velocidad y tiempo. Para que nuestro dispositivo móvil pueda recibir la información de los satélites que se encuentren en ese momento en su campo de visión, deberá de poder captar mensajes con una frecuencia del orden de 1GH [26].

El sistema GPS se basa en el principio matemático de la triangulación. Para que este obtenga un posicionamiento preciso, se requiere la captación de señal de al menos cuatro satélites. Esto es debido a que si recibimos la señal de un solo satélite, dicha información sólo indicaría que nos encontramos en cualquier punto de una esfera cuyo radio sea la distancia entre nuestro receptor y el satélite [26]. Con la captación de señal de dos satélites, localiza al usuario en la circunferencia resultante de la intersección entre las esferas de ambos satélites. Por ello, conforme se detecte mayor cantidad de satélites, mayor cantidad de puntos obtendrá el sistema, y por ende mayor precisión.

Precisión de los receptores

Número de satélites sintonizados

Uno de los factores que afectan en la precisión consiste en la dependencia en el número de satélites sintonizados por el receptor. Cada señal del satélite se recibe en el receptor en canales diferentes. Cuantos más canales disponga el receptor, mayor precisión podrá proporcionar al usuario. Sin embargo, es posible que el receptor no sintonice en un momento dado un satélite en cada uno de sus canales disponibles. Lo normal es sintonizar entre 7 y 9 canales para tener una buena precisión en la señal [27].

Retraso atmosférico

En sistemas GPS cuando la onda emitida por el satélite la atraviesa se produce un retardo en la señal provocada en parte por el efecto del vapor de agua sobre la señal y por los electrones libres de la ionosfera. Por lo tanto la calidad de la señal dependerá de las condiciones atmosféricas en ese momento. Esta calidad afectará a la precisión de la posición que proporciona el receptor G.P.S. Siempre será más precisa la señal de noche que de día, ya que de noche la ionosfera está menos ionizada [28].

Señal multicamino

Este error se produce debido al rebote de las ondas electromagnéticas contra obstáculos como los edificios de una ciudad, lo que da lugar a un retraso en la recepción de las señales

emitidas por el satélite. Como consecuencia el receptor GPS detecta la misma señal en diferentes rangos [27].

Errores de almanaque

Producido por el desajuste horario entre los satélites y el receptor. Si bien los satélites incorporan relojes atómicos de máxima precisión, los receptores no las incorporan por lo que se obtendrá un porcentaje de error [27].

Errores de Efemérides

Estos errores surgen a causa de la imprecisión de la situación orbital [27].

5.3.2.2. Implementación de localización

Esta parte corresponde al trabajo que se ha llevado a cabo para que la detección de la ubicación del usuario funcione correctamente en la aplicación. Esta se divide en la obtención de datos de localización y en el cálculo de las distancias.

Obtención de los datos de localización

Para poder obtener las coordenadas actuales en las que se encuentra el usuario recogidas del sensor GPS se requerirá del uso de la clase LocationService [29]. Esta clase proporcionada por Unity permite acceder a la información de ubicación del dispositivo móvil. Dicha información posteriormente será almacenada en variables locales para que puedan ser comparadas posteriormente. De esta clase se han utilizado las siguientes propiedades y métodos recogidas en este documento.

Input.location.isEnabledByUser: Esta función verifica si el servicio de ubicación del dispositivo está habilitado.

Input.location.Start(): Cuando se llama a este método se inicializan las actualizaciones del servicio de ubicación, accediendo a los últimos datos de ubicación del usuario.

Input.location.Stop(): Al contrario del método Input.location.Start, este finaliza las actualizaciones de ubicación.

Input.location.Status: Con esta funcionalidad se podrá conocer el estado del servicio de ubicación

Para la realización del siguiente código se ha tenido en consideración las ayudas proporcionadas por Unity para un empleo óptimo de la clase LocationService.

Todas las funciones mencionadas anteriormente vienen recogidas en la corrutina GetCoordinates. Se requiere del uso de corrutinas debido a que los datos de localización varían con el tiempo por lo que es necesario que esta función se llame cada x tiempo para poder actualizar los valores constantemente. Esta corrutina comprueba que el servicio de ubicación del dispositivo móvil está operativo en ese momento e inicializa las actualizaciones del servicio de ubicación. Una vez actualizados, lo almacena en las variables locales MiLatitud y MiLongitud y llamara a las funciones DistanciaMinima, que se encargará de calcular la distancia a la que se encuentra el usuario respecto a los distintos puntos, y UbicacionCercana, que se encarga de verificar si se encuentran en un radio menor a 8 metros. En caso de falla del servicio, este se detendrá gracias a la línea yield break.

El lapso de tiempo entre llamadas viene determinado por la línea `yield return new WaitForSeconds(3f)`, siendo este de 3 segundos debido en parte a los datos recogidos por los sensores de orientación que serán explicados posteriormente y por mantener un mejor rendimiento de memoria sin que ello afecte en demasía a la ejecución de la aplicación.

```
IEnumerator GetCoordinates()
{
    while (true)
    {
        //Comprueba si el servicio de ubicación está activo en el
dispositivo.
        if (!Input.location.isEnabledByUser)
            yield break;

        Input.location.Start(1f, .1f);
        Input.compass.enabled = true;

        //Le añadiremos un tiempo de espera hasta que el servicio se
inicie.
        int maxWait = 10;
```

```

                                while (Input.location.status ==
LocationServiceStatus.Initializing && maxWait > 0)
    {
        yield return new WaitForSeconds(1);
        maxWait--;
    }

    //Si el servicio no inicializado tras pasar 10 segundos,
    // mostrará un mensaje de tiempo agotado (Time out).
    if (maxWait < 1)
    {
        print("Timed out");
        yield break;
    }

    //Si ha fallado la conexión
    if (Input.location.status == LocationServiceStatus.Failed)
    {
        print("Unable to determine device location");
        yield break;
    }
    else
    {
        //Variables locales de localización
        MiLatitud = Input.location.lastData.latitude;
        MiLongitud = Input.location.lastData.longitude;

        //Variable local de orientación
        brujula = Input.compass.trueHeading;

        DistMin = DistanciaMinima();
        EstadoActual = UbicacionCercana(DistMin);

        if (primeraVezActivo) //Se ha pulsado inicio
        {
            OpcionesEstado(EstadoActual);

            // Verifico que ya estoy mirando al monumento
            if (!dentro && EstadoActual == 2)
            {
                if (DetectarMonumento())
                {
                    cmpss = Input.compass.trueHeading;
                    dentro = true;
                    panelTitulo.SetActive(true);
                }
            }
        }
    }
}

```

```
        yield return new WaitForSeconds(3f);  
    }  
}
```

Figura 33. Función de localización

Distancia Mínima

Una vez recogidas las variables de latitud y longitud en este caso, se realizará las pertinentes comprobaciones para saber si se está en el lugar. Para conocer cómo de lejos se encuentra el usuario de los distintos puntos registrados se calcula la distancia entre cada uno de los puntos y se compara para conocer cuál es el más próximo.

Si bien esta aplicación está situada en la ciudad de Jaén, se han tomado en cuenta dos tipos distintos de cálculo de distancias entre dos puntos con el fin de cumplir con el objetivo de obtener una aplicación lo más genérica y versátil posible. Así se podrá aumentar el radio de acción sin que este se vea afectado por ello.

Para distancias cortas se ha optado por calcular la distancia en línea recta entre dos puntos ya que para distancias mayores de 20 km, este método proporciona un mayor margen de error debido a la curvatura propia de la Tierra [30]. Esta fórmula es utilizada para calcular la distancia entre el usuario y el siguiente punto de la ruta ya que la diferencia nunca será mayor de 200 metros.

Por el contrario, para el cálculo de distancia entre el usuario y los distintos monumentos si que se ha tenido en cuenta la condición expuesta anteriormente.

```

public float DistanciaRecta(float lat1, float long1, float lat2, float long2)
{
    //Pasamos de polar a cartesianas
    float latm = (lat1 + lat2) / 2;

    float m_per_deg_lat = 111132.954f - 559.822f * Mathf.Cos(2 * latm)
        + 1.175f * Mathf.Cos(4 * latm);
    float m_per_deg_lon = 111132.954f * Mathf.Cos(latm);

    float deltaLat = Mathf.Abs(lat1 - lat2);
    float deltaLon = Mathf.Abs(long1 - long2);
    float distance = Mathf.Sqrt(Mathf.Pow(deltaLat * m_per_deg_lat, 2)
        + Mathf.Pow(deltaLon * m_per_deg_lon, 2));

    return distance;
}

```

Figura 34. Función para cálculo de distancias

Para largas distancias se ha optado por la fórmula de Haversine para cuerpos circulares ya que es la opción que proporciona unos buenos resultados sin requerir de un coste computacional muy elevado. Si bien en este proyecto no se va a utilizar debido a que las distancias no serán mayores de 20 Km, si que se ha implementado ya que este método es el más empleado hoy en día para la ubicación de dos puntos en el planeta y proporciona el objetivo generalista buscado en este proyecto ya que así si en un futuro se desea ampliar el radio de acción, no se verá afectado.

Dicha fórmula determina la distancia de círculo máximo entre dos puntos de la esfera a partir de sus latitudes y sus longitudes.

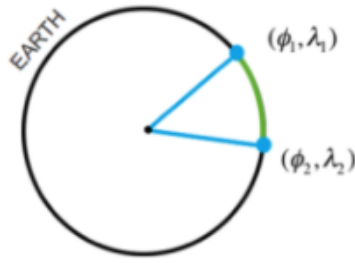


Figura 35. Ilustración de la fórmula de Haversine.

Viene dada por la siguiente fórmula:

$$\text{Haversine}(\theta) = \sin^2\left(\frac{\theta}{2}\right) = \frac{1 - \cos(\theta)}{2}$$

Siendo la distancia:

$$d = 2r \sin\left(\sqrt{\sin^2\left(\frac{\theta_2 - \theta_1}{2}\right) + \cos(\theta_1)\cos(\theta_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

```
public float FormulaHaversine(float lat1, float long1, float lat2, float
long2){
    float R = 6371000; // Radio de la Tierra en metros
    float omega1 = ((lat1 / 180) * Mathf.PI);
    float omega2 = ((lat2 / 180) * Mathf.PI);
    float variacionomega1 = (((lat2 - lat1) / 180) * Mathf.PI);
    float variacionomega2 = (((long2 - long1) / 180) * Mathf.PI);
    float a = (Mathf.Sin(variacionomega1 / 2) * Mathf.Sin(variacionomega1
/ 2)) + (Mathf.Cos(lat1) * Mathf.Cos(lat2) * Mathf.Sin(variacionomega2 / 2) *
Mathf.Sin(variacionomega2 / 2));
    float c = 2 * Mathf.Atan2(Mathf.Sqrt(a), Mathf.Sqrt(1 - a));
    return Mathf.Abs(c * R);
}
```

Figura 36. Función para el cálculo de la distancia con Haversine.

5.3.3. Módulo de orientación

La finalidad de este módulo consiste en la obtención de la dirección en la cual se encuentra el usuario en tiempo real a fin de saber hacia qué dirección está mirando el usuario en ese momento, ya que este actuará como disparador al igual que en el módulo de localización.

Para ello se vale tanto del giroscopio como del acelerómetro, por ello uno de los requisitos principales de esta aplicación es que el dispositivo los incorpore.

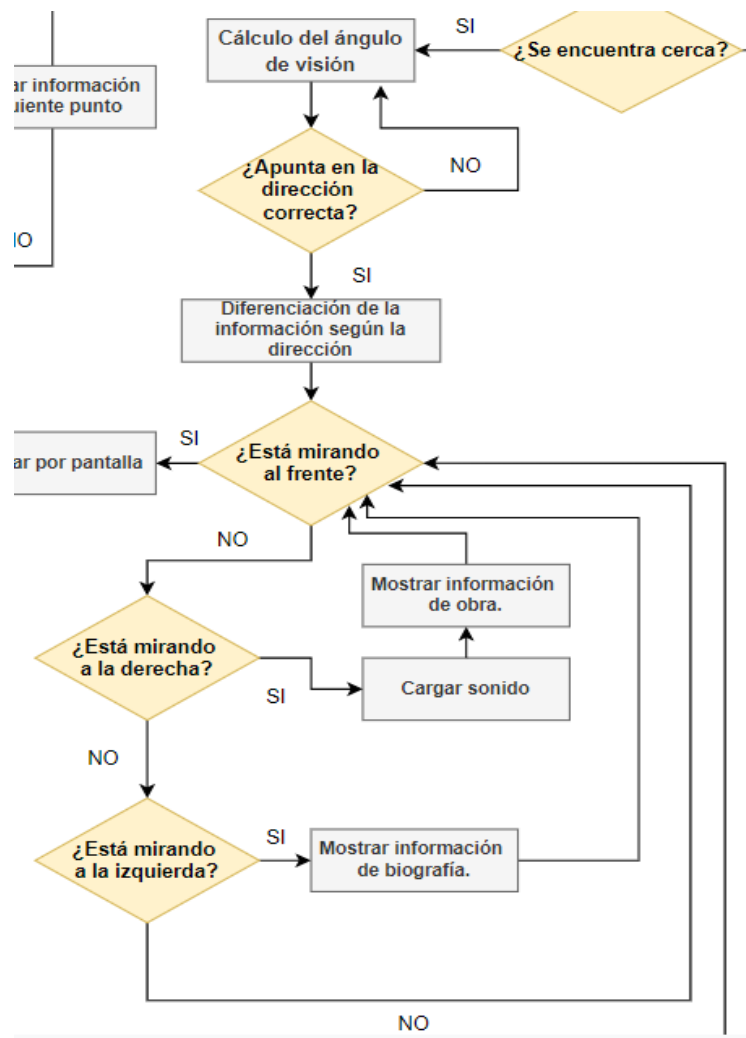


Figura 37. Zoom del diagrama de flujo 3

5.3.3.1. *Acelerómetro*

El acelerómetro es un dispositivo cuyo fin es medir y analizar la aceleración lineal y angular, más concretamente mide la fuerza que sufre en una dirección determinada y no la aceleración como tal. Los acelerómetros que se encuentran en los dispositivos móviles son de tipo MEMS (Micro Electro Mechanical System) provisto de una patilla microscópica o plaqueta del condensador. Esta se encuentra ubicada entre otras dos más separadas entre sí. Al moverse el dispositivo, la plaqueta se flexiona, disminuyendo la distancia entre las otras placas, y con ello modificando la capacidad del condensador [31].

Para el objeto de este TFG, el acelerómetro funcionará en vertical, por lo que será el eje y el que sufra las variaciones de movimiento.

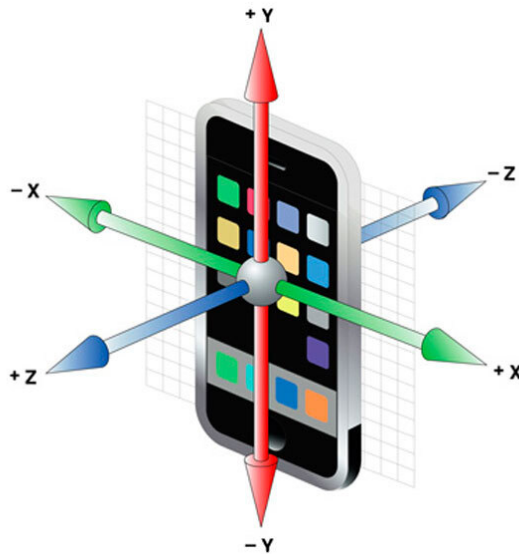


Figura 38. Representación de los ejes cartesianos en un dispositivo móvil.

5.3.3.2. *Giroscopio*

El giroscopio es un sensor que sirve para medir la velocidad y cambio de rotación que sufre el dispositivo. Esto lo consigue añadiendo una cuarta dimensión de movimiento la cual permite medir la rotación o el giro del dispositivo.

Al igual que el acelerómetro, este se trata de un MEMS (Micro Electro Mechanical System) el cual está formado por varios brazos que se encuentran en constante vibración, un brazo detector, un estator central fijo del que se encontrarán conectados los brazos y por sensores que traducen el movimiento de los brazos. [31]

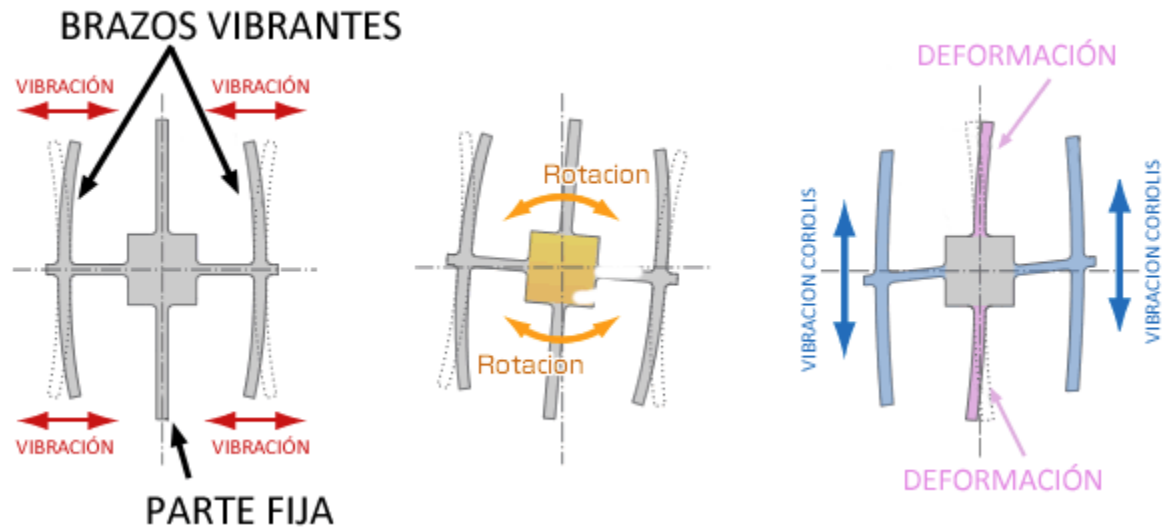


Figura 39. Esquema del giroscopio.

El funcionamiento se basa en el funcionamiento del péndulo de Foucault y el análisis de un elemento vibrante y las fuerzas de Coriolis debidas a la rotación de la Tierra [32]. Los brazos vibrantes se encuentran en constante movimiento por lo que cuando el dispositivo se mueve, provoca una alteración del sentido de la vibración. Estas variaciones son detectadas por los sensores traductores de movimiento que convierten esas variaciones en impulsos eléctricos [33].

La combinación del giroscopio y el acelerómetros es lo que nos permite conocer en qué posición del espacio, que inclinación, velocidad o aceleración se mantiene el usuario, por lo que conoce cuando este se encuentra en movimiento [33].

5.3.3.3. *Implementación de orientación*

En este módulo se divide en dos partes, en la extracción de los datos y en su procesamiento.

Extracción de datos

En esta parte del código se obtendrán los datos recogidos por los sensores de movimiento ya mencionados previamente mediante la clase Compass proporcionada por Unity [34]. Esta accede a la información de brújula del dispositivo móvil. Al igual que con el módulo de localización, estos datos quedarán guardados en variables locales. Se han utilizado para este objetivo las siguientes funciones o métodos:

Input.Compass.Enabled: Esta función sirve para activar la brújula del dispositivo. Para que las funcionalidades de esta clase funcionen es necesario que se active una vez que Input.location.Start() haya sido activado.

Input.compass.trueHeading: Esta función adquiere los datos recogidos de los sensores de movimiento (giroscopio y acelerómetro) en grados relativos al Polo Norte geográfico.

Las funciones ya mencionadas vienen recogidas en la corrutina GetCoordinates() al igual que el módulo de localización debido a la dependencia que tiene por este último. Esta funcionalidad de Unity Input.compass.trueHeading presenta un gran problema y es que para una misma posición presenta variaciones altas del resultado, de entre más o menos 20 grados, por ello se ha establecido un tiempo de repetición de 3 segundos y se ha creado una franja de acción. De esta forma, se mitigan estos cambios tan bruscos sin afectar al funcionamiento de la aplicación. Si se bajara el tiempo de repetición, los cambios serían más bruscos y requeriría de un mayor gasto de memoria. Si por el contrario se aumentará el dicho tiempo, la aplicación daría una sensación de ralentización o de congelación de la pantalla.

Procesado de datos

En este apartado se establecen las condiciones que se deben de cumplir para conocer hacia dónde se está mirando.

Primero se ha de establecer cuando se está mirando hacia el monumento cuya finalidad corresponde a la función *DetectarMonumento()*. En ella se establecen unos umbrales de detección a partir del dato ya recogido del contenedor de datos. Si el resultado es verdadero, se guardará en la variable *cmpss* el valor que esté marcando la brújula en ese momento dentro de la función *getCoordinates()*, ya que dicha variable sirve para establecer que es mirar al frente del usuario, por lo que solo se ejecuta una vez por monumento.

```
public bool DetectarMonumento()
{
    int umbralSuperior = gestion.GeneralinfoArray[position].brujula
                        + 20;
    int umbralInferior = gestion.GeneralinfoArray[position].brujula
                        - 20;

    if (brujula < umbralSuperior && brujula > umbralInferior)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Figura 40. Función para la detección del monumento

En esta parte del código, se detectará hacia donde se está mirando, es decir, el sentido del movimiento. Antes de verificarlo, se ha de establecer los valores umbrales que identificarán cuando se está hablando de derecha o izquierda. La función *ExcepcionCompass* se encarga de ello. En ella, se establecen los límites inferiores y superiores a partir de la variable *cmpss* que

fue recogida en *getCoordinates()*. Debido a que se va a trabajar en los valores comprendidos entre 0 y 360°, si alguno de estos supera esos 360°, se calculará su equivalente menor.

```
public void ExcepcionCompass()
{
    if (dentro) //Si me encuentro en el sitio
    {
        LIDerecho = (int)Mathf.Abs(cmpss) + 55;
        LSIZquierdo = (int)Mathf.Abs(cmpss) + 305;
        LSDerecho = (int)Mathf.Abs(cmpss) + 145;
        LIIzquierdo = (int)Mathf.Abs(cmpss) + 215;

        if (LIIzquierdo >= 360)
        {
            LIIzquierdo = LIIzquierdo - 360;
        }
        if (LSDerecho >= 360)
        {
            LSDerecho = LSDerecho - 360;
        }
        if (LSIZquierdo >= 360)
        {
            LSIZquierdo = LSIZquierdo - 360;
        }
        if (LIDerecho >= 360)
        {
            LIDerecho = LIDerecho - 360;
        }
    }
}
```

Figura 41. Función para limitar los ángulos.

Tras establecer los umbrales es hora de identificar hacia dónde se está mirando. Esta función la realiza *VerificarRotacion()*. En ella se compara el valor de la orientación actual con los valores umbrales establecidos. Se ha detectado que para ciertos valores, las condiciones principales no se cumplen por lo que se han realizado excepciones para dichos casos. Por ejemplo, cuando se va a detectar el giro hacia la derecha, para valores del límite inferior superiores a 270°, la comparación se tendrá que hacer desde el límite inferior hasta 360° y desde 0 hasta el límite superior. Esta función devolverá el string derecha, izquierda o frente, identificando dicho movimiento. En caso de que no se cumpla ninguna de las condiciones (izquierda o derecha) se detectará como si mirara al frente.

```
public string VerificarRotacion()
{
    ExcepcionCompass(); //se establece los umbrales de
    comparación

    if (LIDerecho > 270) //El ángulo comparador es mayor que el
    ángulo de la derecha
    {
        //Giro hacia la derecha
        if (((int)Mathf.Abs(brujula) > LIDerecho
            && (int)Mathf.Abs(brujula) < 360)
            || (int)Mathf.Abs(brujula) < LSDerecho
            && (int)Mathf.Abs(brujula) > 0)
        {
            mizq = false;

            if (!mdrch)
            {
                mdrch = true;
            }
        }
    }
}
```

```

        return "right";
    }
}
else
{

    if ((int)Mathf.Abs(brujula) > LIDerecho
        && (int)Mathf.Abs(brujula) < LSDerecho)
    {
        mizq = false;

        if (!mdrch)
        {
            mdrch = true;
        }

        return "right";
    }

}

//El ángulo comparador es menor que el ángulo de la
izquierda
if (LSIzquierdo < 45)
{
    //Giro hacia la izquierda
    if (((int)Mathf.Abs(brujula) > LIIzquierdo
        && (int)Mathf.Abs(brujula) < 360)
        || (int)Mathf.Abs(brujula) < LSIzquierdo
        && (int)Mathf.Abs(brujula) > 0)
    {

```

```

        mdrch = false;

        if (!mizq)
        {
            mizq = true;
        }
        return "left";
    }
}
else
{
    if ((int)Mathf.Abs(brujula) < LSizquierdo
        && (int)Mathf.Abs(brujula) > LIIzquierdo)
    {
        mdrch = false;

        if (!mizq)
        {
            mizq = true;
        }

        return "left";
    }
}
return "front";
}

```

Figura 42. Función para la detección del sentido de giro.

5.3.4. Módulo de captación

Este módulo se encarga de captar el entorno que le rodea y de anclar la información necesaria al cumplir las condiciones de localización y orientación. Por tanto, este módulo se encarga de mostrar por pantalla la información que se debe de mostrar en cada momento.

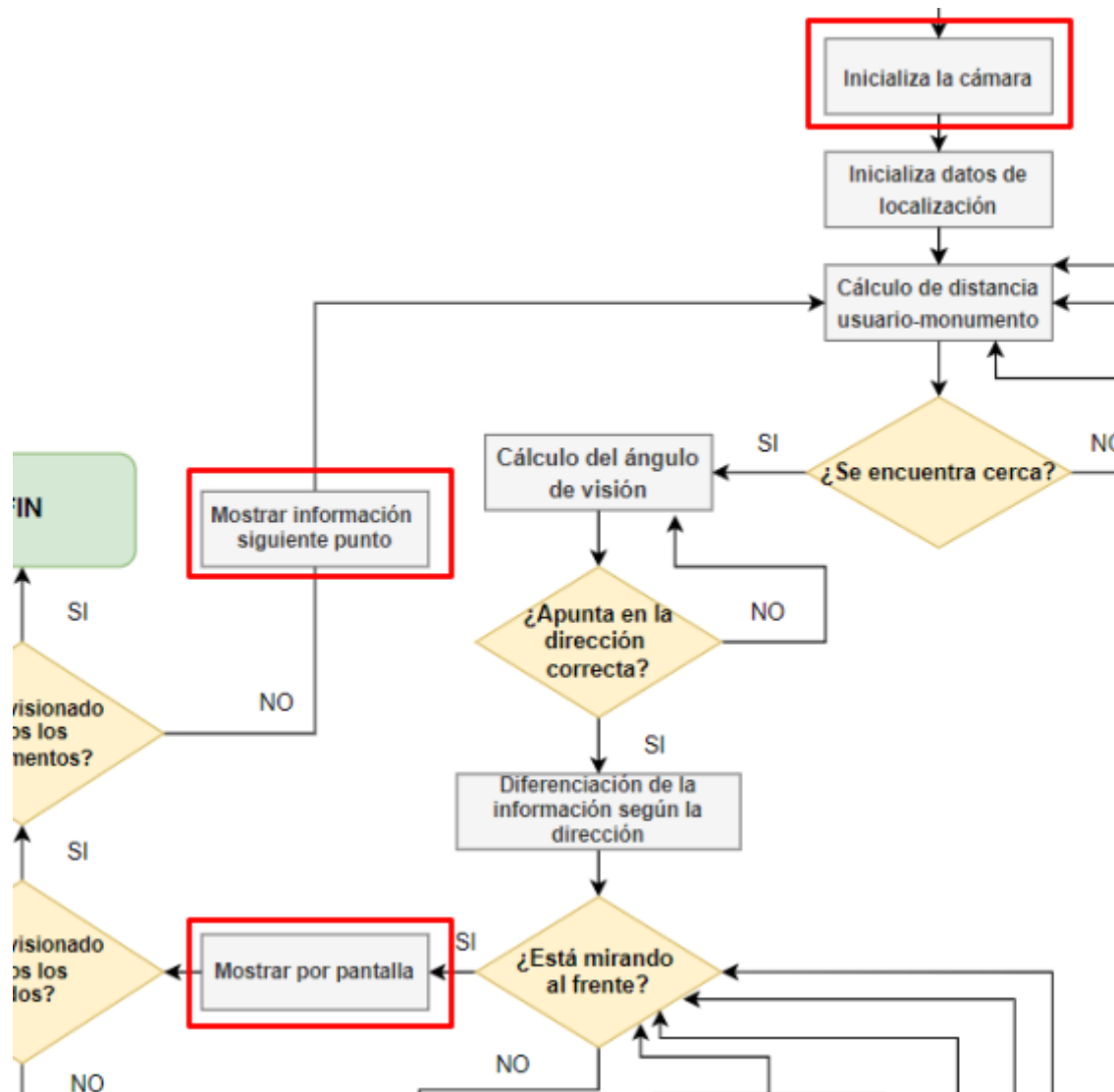


Figura 43. Zoom del diagrama de flujo 4

Cámara

La cámara es uno de los elementos más importantes ya que sin ella no podríamos hablar como tal de realidad aumentada. A día de hoy es raro no encontrar un dispositivo sin cámara por lo que no supone un problema. Este componente recoge los colores de la parte visible del espectro electromagnético.

ARCamera

Para poder captar el entorno que nos rodea con la aplicación será necesario incorporar una cámara en el proyecto de Unity que pueda obtener los datos recogidos por la cámara del dispositivo. Por defecto Unity dispone de una cámara tipo la cual es únicamente efectiva cuando se trata de proyectos de realidad virtual, pues no está pensada para realidad aumentada. Para solventar este problema, Unity ofrece dos opciones: bien utilizar la cámara de realidad aumentada proporcionada por Vuforia o bien mediante la cámara de ARTool o ARKit (Android e IOS). En este proyecto se ha decidido trabajar con la cámara de Vuforia ya que ofrece un entorno más intuitivo y cómodo con el que poder trabajar, además de la disponibilidad de operar para ambas plataformas.

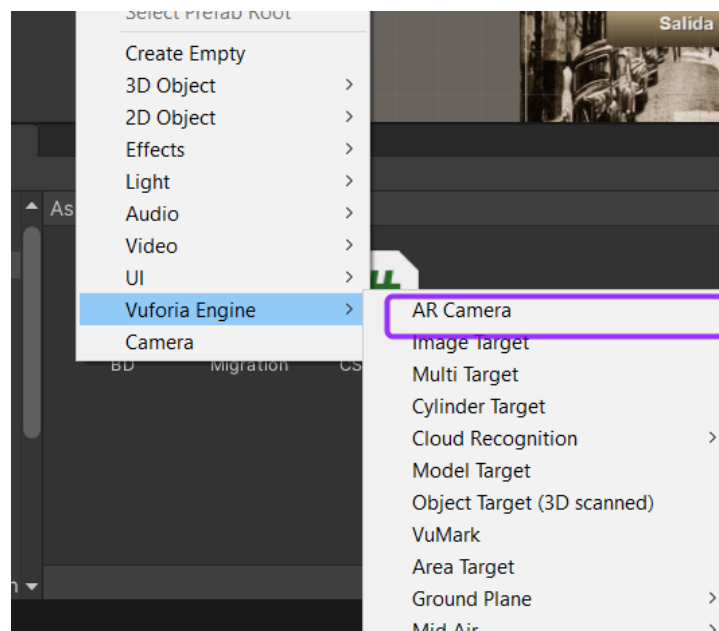


Figura 44. Panel de configuración de la ARCamera

La instalación de Vuforia se encuentra registrado en el Anexo de esta memoria.

Disparador

Esta parte del módulo se encarga de mostrar la información en pantalla en el momento oportuno. Para ello gestiona los resultados obtenidos tanto en el módulo de localización como en el módulo de orientación y llama a los distintos elementos que conforman la interfaz gráfica de la aplicación. Por ejemplo en la función descrita debajo, verifica que se hayan activado las variables *izqCheck* y *drchCheck* y se hayan mostrado los 4 monumentos para mostrar el mensaje de final de ruta. En caso contrario, mostrará por pantalla la información del siguiente camino.

```
private void InfoMostrada()
{
    if (GeneralinfoArray[geo.GetPosition()].izqCheck
        && GeneralinfoArray[geo.GetPosition()].drchCheck)
    {
        GeneralinfoArray[geo.GetPosition()].visto = true;
        MonumentoVisto++;
        if (MonumentoVisto == 4) //Se han visto todos
        {
            PanelNombre.SetActive(false);
            Finish.SetActive(true);
        }
        else
        {
            SiguienteDireccion.SetActive(true);
            MstrInf.DescrDirec.text =
GeneralinfoArray[geo.GetPosition()].direccionSiguiente;
            MstrInf.DescrDirec.fontSize = 36;
            MstrInf.DescrDirec.color = Color.black;
            MostrarImage();
        }
    }
}
```

```

    }
}

```

Figura 45. Función para finalización de ruta.

5.3.5. Módulo de filtrado de imágenes

Este módulo se encarga de capturar la imagen del entorno al ser presionado el botón de cámara y de modificar dicha imagen a partir de los filtros que integran este módulo.

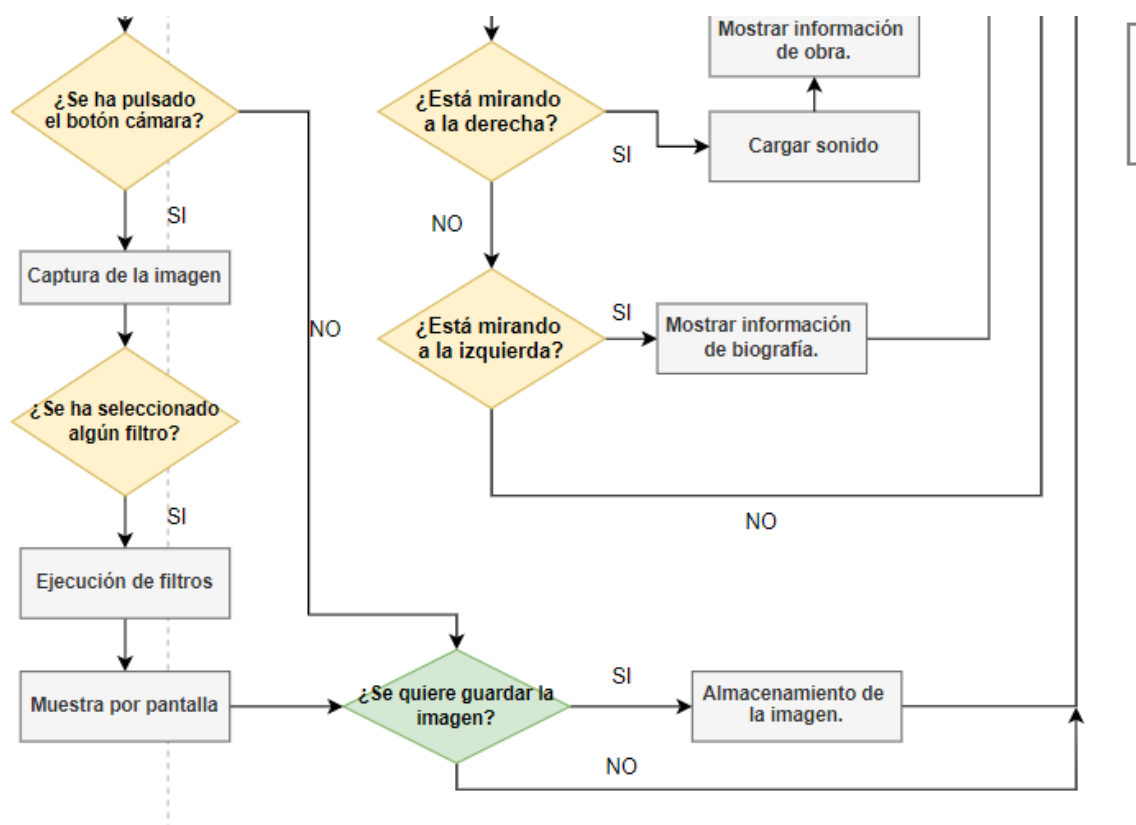


Figura 46. Zoom del diagrama de flujo 5

5.3.5.1. Captura de imagen

Para poder hacer la captura de imagen del entorno se utilizarán las siguiente función de la clase Texture2D de Unity [35].

ReadPixels: Esta función lee los píxeles de representación actual (pantalla) y los almacena dentro de una textura en la posición definida por destX y destY, en el caso de este proyecto valdrán 0.

Esta función descrita debajo se ejecuta cuando es pulsado el botón de cámara. En ese momento se deja en pantalla únicamente el cartel con el nombre del personaje histórico. A partir de ese momento se obtienen las medidas del contenedor de imagen donde estará ubicada dicha fotografía al final y se redondeará. Después se captura la imagen que se muestra por pantalla con la función *ReadPixels()* proporcionada por Unity y se almacena en una textura del tamaño de la pantalla. Debido a que el tamaño del contenedor de la imagen difiere del de la pantalla completa, será necesario redimensionar la imagen para que sea mostrada correctamente.

Esto se realiza con la función *Resize()* que se ha creado para dicha finalidad. Una vez hecho esto, ya se puede mostrar por pantalla, anclando la textura al contenedor de imagen.

```
IEnumerator TakeAndSaveScreenshot()
{

    yield return new WaitForEndOfFrame();
    principalFrente.SetActive(false);
    RectTransform rt = Container.GetComponent<RectTransform>();

    ImageOriginWidth = Mathf.RoundToInt(rt.rect.width);
    ImageOriginHeight = Mathf.RoundToInt(rt.rect.height);

    x = Mathf.RoundToInt(rt.rect.x);
    y = Mathf.FloorToInt(rt.rect.y);

    screenImage = new Texture2D(Screen.width, Screen.height,
```

```

        TextureFormat.RGB24, false);
        //Obtiene la imagen de la pantalla y la almacena en
screenImage
        screenImage.ReadPixels(new Rect(0, 0, Screen.width,
            Screen.height), 0, 0);
        screenImage.Apply();

        //Se activa la interfaz correspondiente a la visualización
de la imagen tomada
        camara.gameObject.SetActive(true);

        //Se redimensiona la imagen para adaptarla al tamaño del
marco.
        Container.texture = Resize(screenImage, ImageOriginWidth,
            ImageOriginHeight);

        //Mostramos por pantalla
        GameObject image = GameObject.Find("RawImage");
        image.GetComponent<RawImage>().texture = Container.texture;
    }

```

Figura 47. Función para captación de imágenes.

5.3.5.2. *Filtrado*

Para la realización de este apartado se ha de tener en consideración de que la plataforma Unity no está pensada para el procesado de imagen en realidad aumentada, ya que la cámara con la que trabajan no pertenece a la propia plataforma de Unity, ya que como tal no tienen. Es por ello que todos los filtros que se van a utilizar se realizan a nivel de píxel, reforzando todo lo aprendido durante la carrera.

Todas las funciones que se mostraran a continuación presentarán la siguiente estructura.

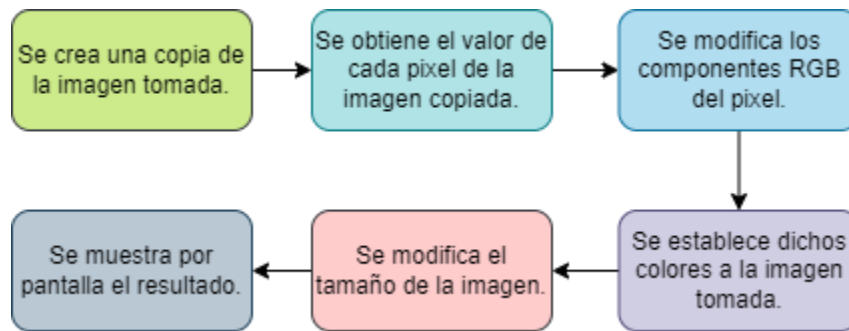


Figura 48. Esquema de las funciones de filtrado

Se utilizará algunas funciones de Unity para poder obtener información de la imagen tomada y son las siguientes [35]:

GetPixels: La función de este método consiste en almacenar en una matriz la copia de los colores de los píxeles solicitados. La matriz devuelta consiste en una matriz 2D aplanada, donde los datos aparecen fila por fila, distribuyendose los píxeles de izquierda a derecha y de abajo hacia arriba.

GetPixel: La finalidad de esta función consiste en devolver el color del píxel en las coordenadas (x,y).

SetPixels: Esta función establece un bloque de colores de píxeles a una textura, tomando una matriz de colores y cambiando los colores de la textura por el bloque de colores indicado. La matriz de colores es una matriz 2D aplanada, donde los datos aparecen fila por fila.

SetPixel: Esta función establece el color del píxel en las coordenadas (x,y).

GreyScale

Esta función realiza una copia de la imagen que se está mostrando por pantalla y modifica píxel por píxel el color de la imagen realizando la media de los valores RGB. Una vez modificada, lo muestra por pantalla, cambiando la textura del contenedor de la imagen.

```

public void GreyScale()
{
    // Copia la imagen mostrada por pantalla y la almacena en
    una nueva textura
    grisTex = new Texture2D(Screen.width, Screen.height,
        TextureFormat.RGB24, false);
    Color[] pix = screenImage.GetPixels();
    grisTex.SetPixels(pix);
    grisTex.Apply();

    for (int y = 0; y < grisTex.height; y++)
    {
        for (int x = 0; x < grisTex.width; x++)
        {
            Color pixel_color = grisTex.GetPixel(x, y);
            float grey = (pixel_color.r + pixel_color.g +
                pixel_color.b) / 3;
            Color newColor = new Color(grey, grey, grey, grey);
            grisTex.SetPixel(x, y, newColor);
        }
    }
    grisTex.Apply();
    Container.texture = Resize(grisTex, ImageOriginWidth,
        ImageOriginHeight);

    // Mostramos por pantalla
    GameObject image = GameObject.Find("RawImage");
    image.GetComponent<RawImage>().texture = Container.texture;
}

```

Figura 49. Filtro de escala de grises

Se obtiene como resultado:



Figura 50. Comparación tras aplicar el filtro de escala de grises.

Negativo

Esta función modifica el color de cada píxel de la fotografía tomada obteniendo el color inverso de los mostrados por pantalla. Para ello almacena los píxeles de la imagen y acceden uno por uno para que sean modificados.

```

public void Negativo()
{
    Texture2D negText = new Texture2D(Screen.width,
Screen.height,
TextureFormat.RGB24, false);
    Color[] pix = screenImage.GetPixels();
    negText.SetPixels(pix);
    negText.Apply();
    for (int y = 0; y < negText.height; y++)
    {
        for (int x = 0; x < negText.width; x++)
        {
            Color pixel_color = negText.GetPixel(x, y);
            Color newColor = new Color(1 - pixel_color.r, 1 -
pixel_color.g, 1 - pixel_color.b);
            negText.SetPixel(x, y, newColor);
        }
    }
    negText.Apply();
    Container.texture = Resize(negText, ImageOriginWidth,
ImageOriginHeight);

    // Mostramos por pantalla
    GameObject image = GameObject.Find("RawImage");
    image.GetComponent<RawImage>().texture = Container.texture;
}

```

Figura 51. Filtro negativo.

Se obtiene como resultado:



Figura 52. Comparación de imágenes tras aplicar el filtro negativo.

Sepia

Para realizar este filtro se modifica el color de imagen multiplicando por las constantes preestablecidas para este filtro y almacenando en forma de textura.

```
public void Sepia()
{
    sepText = new Texture2D(Screen.width, Screen.height,
        TextureFormat.RGB24, false);
    Color[] pix = screenImage.GetPixels();
    sepText.SetPixels(pix);
    sepText.Apply();
}
```

```

        for (int y = 0; y < sepText.height; y++)
        {
            for (int x = 0; x < sepText.width; x++)
            {
                Color pixel_color = sepText.GetPixel(x, y);
                float outputRed = (pixel_color.r * 0.393f) +
                    (pixel_color.g * 0.769f) + (pixel_color.b *
0.189f);

                float outputGreen = (pixel_color.r * 0.349f) +
                    (pixel_color.g * 0.686f) + (pixel_color.b *
0.168f);

                float outputBlue = (pixel_color.r * 0.272f) +
                    (pixel_color.g * 0.534f) + (pixel_color.b *
0.131f);

                Color newColor = new Color(outputRed, outputGreen,
outputBlue);

                sepText.SetPixel(x, y, newColor);
            }
        }
        sepText.Apply();
        Container.texture = Resize(sepText, ImageOriginWidth,
                                ImageOriginHeight);

        // Mostramos por pantalla
        GameObject image = GameObject.Find("RawImage");
        image.GetComponent<RawImage>().texture = Container.texture;
    }

```

Figura 53. Filtro sepia.

Se obtiene como resultado:



Figura 54. Comparación de imágenes tras aplicar el filtro sepia.

ModifyContrast

Tanto este filtro como el siguiente requerirá de un valor de contraste preestablecido. Para ello se dispone de una barra que devolverá un valor dependiendo de la posición en la que se encuentre. Este valor se empleará para modificar el valor del color de cada pixel, siendo este negro si el valor de contraste es 0 y blanco si es 1.

```

public void ModifyContrast(float contrast)
{
    contrastText = new Texture2D(Screen.width, Screen.height,
                                TextureFormat.RGB24, false);
    Color[] pix = screenImage.GetPixels();
    contrastText.SetPixels(pix);
    contrastText.Apply();
    for (int y = 0; y < contrastText.height; y++)
    {
        for (int x = 0; x < contrastText.width; x++)
        {
            Color pixel_color = contrastText.GetPixel(x, y);
            float red = ((pixel_color.r - 0.5f) * contrast) +
0.5f;

            float green = ((pixel_color.g - 0.5f) * contrast) +
0.5f;
            float blue = ((pixel_color.b - 0.5f) * contrast) +
0.5f;
            Color newColor = new Color(red, green, blue);
            contrastText.SetPixel(x, y, newColor);
        }
    }
    contrastText.Apply();
    Container.texture = Resize(contrastText, ImageOriginWidth,
                                ImageOriginHeight);

    //Mostramos por pantalla
    GameObject image = GameObject.Find("RawImage");
    image.GetComponent<RawImage>().texture = Container.texture;
}

```

Figura 55. Filtro modificador de contraste.



Figura 56. Comparación de imágenes tras aplicar el filtro de contraste.

CorrectionGamma

Este filtro modifica el color de cada píxel obteniendo unos valores de RGB resultado de la obtención del color elevado al valor de gamma. Este filtro aumenta o disminuye el grado de iluminación de la imagen dependiendo del valor de esta variable gamma.

```

public void CorrectionGamma(float gamma)
{
    float Correct_gamma = 1 / gamma;
    gammaText = new Texture2D(Screen.width, Screen.height,
        TextureFormat.RGB24, false);
    Color[] pix = screenImage.GetPixels();
    gammaText.SetPixels(pix);
    gammaText.Apply();
    for (int y = 0; y < gammaText.height; y++)
    {
        for (int x = 0; x < gammaText.width; x++)
        {
            Color pixel_color = gammaText.GetPixel(x, y);
            float red = Mathf.Pow(pixel_color.r,
Correct_gamma);
            float green = Mathf.Pow(pixel_color.g,
Correct_gamma);
            float blue = Mathf.Pow(pixel_color.b,
Correct_gamma);

            Color newColor = new Color(red, green, blue);
            gammaText.SetPixel(x, y, newColor);
        }
    }
    gammaText.Apply();
    Container.texture = Resize(gammaText, ImageOriginWidth,
        ImageOriginHeight);
    //Mostramos por pantalla
    GameObject image = GameObject.Find("RawImage");
    image.GetComponent<RawImage>().texture = Container.texture;
}

```

Figura 57. Corrección Gamma.



Figura 58. Comparación de imágenes tras aplicar la corrección gamma.

6. Resultados

La solución que se ha decidido adoptar consiste en una aplicación cultural que, mediante el uso de geolocalización y brújula, guiará al usuario hacia las distintas estatuas pertenecientes a los diferentes personajes históricos. Mediante el uso de diferentes aumentos, el usuario obtendrá tanto información relativa a la ubicación de este como información relevante de dichos personajes ilustres.

Para probar el funcionamiento de la aplicación se ha decidido dividir en dos apartados: guiado del usuario e interacción con el monumento en cuestión.

6.1. *Guiado del usuario*

Para comprobar que guía correctamente al usuario hasta el siguiente monumento, se ha posicionado a una distancia cercana al primer punto de ruta del monumento. Una vez detectado el primer punto, la aplicación irá indicando la distancia hasta el siguiente punto de ruta. Para evitar posibles errores respecto a la medición de distancias a la hora de atravesar ciertas calles, y proporcionar una ayuda extra al usuario, se ha incorporado la ayuda visual del mapa, indicando inicio y final del recorrido de los puntos de activación. En el caso de que no se detecte el punto, la aplicación será capaz de recuperar la posición al atravesar el siguiente punto de activación de la ruta.

En la siguiente secuencia de imágenes se muestra como la aplicación detecta los distintos puntos de guiado pertenecientes a las diferentes rutas establecidas en esta aplicación, la distancia a la que se hallan en ese momento y la imagen correspondiente a ese trazo de trayecto.



Figura 59. Ruta hacia Almendros Aguilar.



Figura 60. Ruta hacia Almendros Aguilar 2.

Se ha comprobado que se pasa correctamente entre distintas opciones de posicionamiento, priorizando la interfaz de guiado frente a la de indicación de larga distancia, por lo que, si el usuario abre la aplicación y se encuentra lejos del monumento más cercano, aparecerá la interfaz correspondiente a larga distancia. Una vez se detecta el punto de guiado, la interfaz de guiado tiene mayor prioridad, por lo que, aunque se encuentre lejos del monumento, se mostrará la interfaz de guiado.



Figura 61. Cambio de interfaces 1.

También se ha comprobado que pasa correctamente de mostrar la dirección hacia el siguiente punto, tras haber mostrado la información pertinente, a la interfaz de guiado.



Figura 62. Cambio de interfaces 2.

Una vez el usuario llega al monumento, se verifica que cambia de interfaz, independientemente de si se ha activado la interfaz de guiado, por lo que si el usuario se ha guiado por Google Maps (mediante la url proporcionada en la interfaz de larga distancia), este pueda seguir disfrutando de la experiencia.



Figura 63. Paso de interfaz de guiado a interfaz al ubicarse cerca.



Figura 64. Cambio de detección de lejos a cerca del monumento.

6.2. *Interacción con el monumento*

Una vez el usuario se encuentra cerca del monumento, se ha de verificar diferentes aspectos que conforman la experiencia completa. Para ello se dividirán las pruebas en los siguientes puntos: detección de giro, captación y filtrado de imágenes e interacción con la información del personaje histórico.

Detección de giro

Una de las pruebas a realizar consiste en verificar que el usuario está frente al monumento, comprobando que el mensaje en el que se indica dicha condición desaparece al cumplirse ésta, dejando paso la posibilidad de interactuar con el monumento.



Figura 65. Detección del frente.

Otras de las pruebas que se han realizado es la comprobación de la detección del sentido de giro (izquierda y derecha del usuario), puesto que mostrará distintas interfaces dependiendo del sentido. Para poder pasar de izquierda a derecha es necesario mirar al frente previamente.

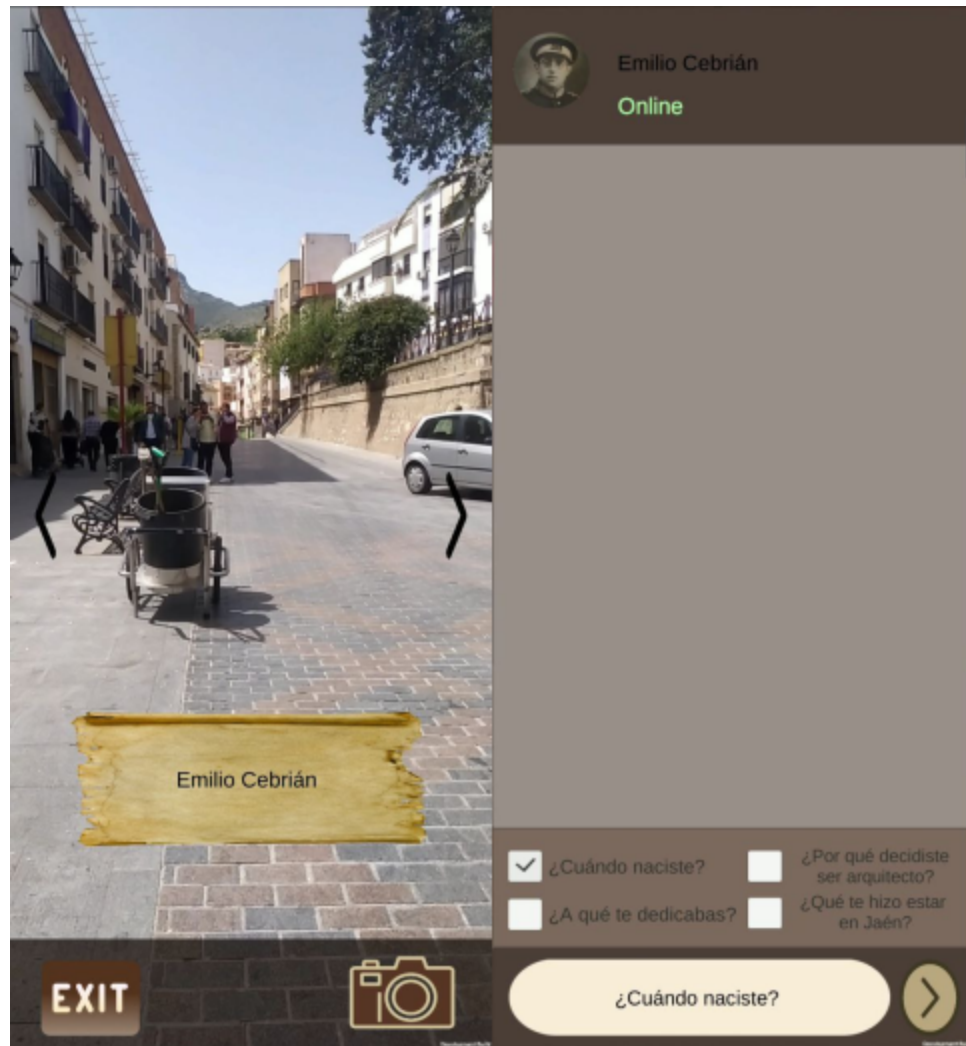


Figura 66. Paso de la vista al frente a la vista hacia la izquierda.



Figura 67. Paso de la vista al frente a la vista hacia la derecha.

Captación y filtrado de imágenes

En este apartado se comprueba que al pulsar el botón de la cámara mostrado cuando el usuario se encuentra frente al monumento, se obtiene la imagen que está captando la cámara del móvil en ese instante. La imagen resultante se mostrará redimensionada dentro de la aplicación, con el fin de poder ser mostrada (ya que se realiza una captura de pantalla).



Figura 68. Imagen capturada con la cámara.

Una vez capturada, se confirma que se ejecutan los diferentes filtros disponibles en la aplicación (ver apartado 5.3.5.2) y que la imagen es guardada correctamente en la galería del dispositivo móvil tras pulsar el botón de guardado.



Figura 69. Guardado de imagen.

Interacción con la información del personaje histórico

En este apartado se va a verificar que tanto la biografía como la información de las obras (interfaz de mirada hacia la izquierda y hacia la derecha) se ejecuta de forma correcta.

Para la interfaz correspondiente a la biografía se comprobará que al seleccionar y enviar las distintas opciones de mensaje, estas aparecen por pantalla junto a la respuesta correspondiente. Cuando los mensajes sobrepasan el panel en el que se están mostrando, se

tendrá que mover manualmente hacia el último mensaje, ya que no está contemplado que se suba automáticamente. Esto queda reflejado en la figura 67.



Figura 70. Generación de conversación.

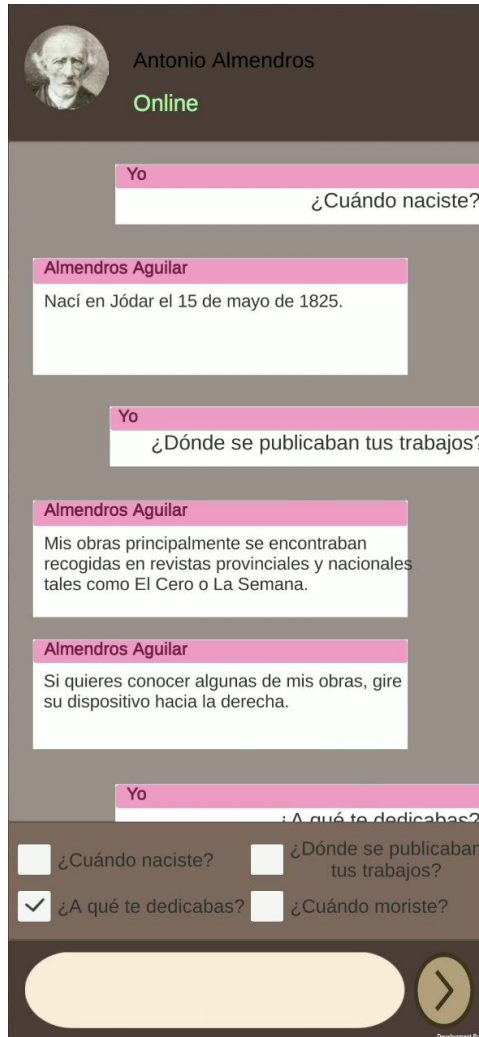


Figura 71. Excepción de tamaño.

Para la interfaz de información de las obras se verificará que se muestra la imagen correspondiente a cada obra. así como el texto y el audio de ésta una vez el usuario elige una de las opciones.



Figura 72. Selección de obra.

También se ha verificado que al cambiar entre obras, la información de estas cambian a la última seleccionada, ya que ambas opciones comparten la misma interfaz de muestra.



Figura 73. Cambio de obras.

Una de las últimas pruebas realizadas consiste en comprobar que la información tanto de la biografía como de las obras se resetean al cambiar de personaje histórico. Todas estas pruebas realizadas se pueden comprobar en el video demostrativo, obteniéndose de forma satisfactoria.

7. Líneas futuras

Tras analizar todo el trabajo realizado se han planteado las siguientes opciones de mejora que permitirán dar un paso más allá a la aplicación. Uno de los principales objetivos que me gustaría llevar a cabo en un futuro próximo, es la incorporación de una base de datos para poder acceder a la información de los distintos monumentos de forma más eficiente. Esto se podría llevar a cabo creando una base de datos con MySQL, incorporando la librería y accediendo a ella mediante script.

Otras de las opciones que me suscitaban mayor interés en incorporar a este proyecto era la inclusión de los personajes históricos o elementos 3D en la experiencia de juego, puesto que haría de la aplicación algo más atractiva visualmente. Sin embargo, para poder conseguir esto, se tendría que crear desde cero cada personaje y cada elemento, requiriendo tener previa experiencia en modelaje 3D, lo que conlleva que temporalmente sea inviable puesto que supone un tiempo extra la adquisición de conocimientos de modelaje. Sin embargo, la inclusión de estos elementos en la aplicación no supondría una gran dificultad, ya que al incorporar Vuforia, se podría anclar los elementos con la herramienta Ground Plane, la cual detecta el suelo y permite anclar los elementos en él.

Por último, sería interesante incorporar un servidor remoto en el que poder almacenar todos aquellos datos multimedia tales como las audiodescripciones de las obras. Así, se podría incluir también audioguías que permitan a la aplicación ser lo más accesible posible, sin que ello conlleve un exceso en el uso de memoria. Mediante la documentación de Unity se pueden obtener los primeros pasos para la implementación de dicho servidor UNet, sin embargo está a punto de ser eliminado de Unity debido a su desuso. Por tanto, hasta que Unity definitivamente integre un nuevo sistema, no merece la pena llevar esta propuesta a cabo.

8. Conclusión

La principal conclusión a la que se ha llegado tras finalizar este proyecto es que se ha conseguido obtener una aplicación versátil y funcional que cumple con los objetivos expuestos.

A breve resumen se ha proporcionado una aplicación de realidad aumentada que ofrece una experiencia turística innovadora e interesante de aquellas personas que marcaron un antes y un después en la historia.

A través del estado del arte se han sembrado las bases con las cuales se ha creado este proyecto. Con el estudio de las diferentes aplicaciones existentes en el mercado, se ha podido

conocer de primera mano la revolución que esta tecnología está produciendo en un sector tan importante como es el turismo, estudiando tanto sus fortalezas como sus debilidades.

Además, se ha realizado el estudio del mayor motor de juegos existente en este momento, Unity. Por último se ha estudiado cómo obtiene la información nuestro dispositivo móvil y cómo realizar procesamiento de imagen con una cámara de realidad aumentada, ampliando los conocimientos adquiridos durante esta carrera. Una vez recogida toda la información, se ha implementado la aplicación y se ha testeado con el fin de verificar que la aplicación es funcional.

9. Bibliografía

- [1] Mauricio Olguín Carbajal, Israel Rivera Zárate, and Erika Hernández Montañez, “Introducción a la Realidad Virtual,” Instituto Politécnico Nacional, Distrito Federal, México, 2006. [Online]. Available: <https://www.redalyc.org/pdf/4026/402640446002.pdf>
- [2] Badilla, M. y Sandoval A.M. (2015). Realidad aumentada como tecnología aplicada a la educación superior: una experiencia en desarrollo. *Innovaciones educativas*, 23, pp. 41-49.
- [3] Redacción, “Ivan Sutherland recibe un Premio BBVA por revolucionar la interacción humano-máquina a través de la RV,” *realovirtual.com*, Feb. 19, 2019. [Online]. Available: <https://www.realovirtual.com/noticias/6443/ivan-sutherland-recibe-premio-bbva-revolucionar-interaccion-humano-maquina-traves-rv>
- [4] Carlos Arce, “Realidad Aumentada”, Universidad Católica “Nuestra Señora de la Asunción,” Asunción. [Online]. Available: <http://jeuazarru.com/wp-content/uploads/2014/10/RA2013.pdf>
- [5] Arce, C. (2013). Realidad aumentada. Obtenido el 08 de septiembre de 2021 desde <http://jeuazarru.com/wp-content/uploads/2014/10/RA2013.pdf>
- [6] Proaño, E.P. (2018). El estatuto de la Realidad Aumentada desde el Pragmatismo norteamericano: la línea teórico-genética de Richard Rorty [Tesis doctoral, Universidad Complutense de Madrid]. <https://eprints.ucm.es/id/eprint/55727/1/T41156.pdf>

- [7] Edwards, A., Hoyt, T. y Reger, G. (2016). Classifying different types of augmented reality technology. *Annual Review of CyberTherapy and Telemedicine*, 14, 199-202.
- [8] 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?page=38>.
- [9] Colaboraeducación30. 2015. Introducción a La Realidad Aumentada. (<https://colaboraeducacion30.juntadeandalucia.es/educacion/colabora/documents/131195/980947/Realidad+aumentada+con+Aumentaty/dac0d367-cada-4627-9289-3a51ed0aba36?version=1.0&previewFileIndex=>)
- [10] Onirix (s.f.). Diferentes tipos de Realidad Aumentada. Obtenido el 08 de septiembre desde [Diferentes tipos de Realidad Aumentada - Onirix](#)
- [11] Montoya, D.M. y Jiménez-Builes, J.A. (2018). Caracterización de elementos para la creación de una herramienta computacional para la gestión del conocimiento en las organizaciones. En J. Sepúlveda (Ed.), *Retos de la investigación en ingeniería de sistemas: aplicaciones, herramientas y desarrollos* (pp. 47- 69) Coruniamericana.
- [12] Fco. José Hidalgo. 2020. "THE MANDALORIAN EN REALIDAD AUMENTADA, LO NUEVO DE GOOGLE PARA ANDROID." *Wwwhatsnew.Com*. Retrieved (<https://www.whatsnew.com/2020/11/23/mandalorian-realidad-aumentada/>).
- [13] Marcelo Belluci. 2020. "Museos: Con Realidad Aumentada, Logran Acercar Las Obras al Público." *Clarín*. (https://www.clarin.com/tecnologia/museos-realidad-aumentada-logran-acercar-obras-publico_0_PExcj_oRU.html)
- [14] Esther Prada Perona. 2019. Wallame: Jugar al Escondite En Realidad Aumentada. España. (https://intef.es/observatorio_tecno/wallame-jugar-al-escondite-en-realidad-aumentada/)
- [15] Patrimonio Virtual. 2011. "Aplicación de Realidad Aumentada a Yacimientos Arqueológicos." *Patrimonio Virtual*. Retrieved (<https://www.patrimoniovirtual.com/investigacion-realidad-aumentada-en-yacimientos-arqueologicos/>).
- [16] PatrimonioAlfas. 2014. Guía Realidad Aumentada Villa Romana de l'Albir. Youtube. (https://www.youtube.com/watch?v=f5ylu75XVvw&t=8s&ab_channel=patrimoniolalfas)
- [17] arq.com.mx. n.d. "Realidad Aumentada Para Arquitectos." *Arq.Com.Mx*. (<https://noticias.arq.com.mx/Detalles/22937.html#.YpIWYihBxPa>)

- [18] Velv Magazine. 2020. “‘Quarantour’ La Experiencia de Realidad Aumentada de Real Estate.” Velv Magazine, April 14. (<https://www.velvmagazine.com/blog/2020/4/14/quarantour-la-experiencia-de-realidad-aumentada-de-real-estate>)
- [19] Laura Coca. 2020. “Mooa, Cuando La Música y La Realidad (Aumentada) Superan La Ficción.” Los40.Com, June 10. (https://los40.com/los40/2020/06/09/musica/1591697700_055558.html)
- [20] CultureClic. n.d. “Culture Clic. Augmented Culture.” (<http://www.cultureclic.fr/>)
- [21] Sociedad Estatal para la Gestión de la Innovación y las Tecnologías Turísticas. 2012. “El MECD Lanza Una Aplicación Para Móviles Inteligentes Que Permite Conocer La Oferta Cultural de España Mediante Geolocalización.” Ministerio de Educación, Cultura y Deporte. (<https://www.segittur.es/sala-de-prensa/notas-de-prensa/el-mecd-lanza-una-aplicacion-para-moviles-inteligentes-que-permite-conocer-la-oferta-cultural-de-espana-mediante-geolocalizacion/>)
- [22] Eventos con Historia. 2017. “Eventos Con Historia Da Vida a Personajes Históricos En La App de San Miguel.” Eventos Con Historia. Retrieved (<http://eventosconhistoria.com/tag/aplicacion-movil-cementerio-historico-san-miguel/>).
- [23] Websa100. n.d. “Guideo, Una App Turística Basada En La Realidad Aumentada.” SEOptimizer. (<https://www.seoptimizer.com/es/blog/guideo-una-app-turistica-basada-en-la-realidad-aumentada/>)
- [24] Unity User Manual. 2021. “ScriptableObjects.” (<https://docs.unity3d.com/Manual/class-ScriptableObject.html>)
- [25] Eduardo Huerta, Carlos Galles, Andrés Greco, and Aldo Mangiaterra, “El GPS y la teoría de la relatividad,” Rosario. [Online]. Available: <https://www.fceia.unr.edu.ar/gps/publicaciones/GPSyREL240407.pdf>
- [26] <https://ebookcentral.proquest.com/lib/ujaen/reader.action?docID=5430709&ppg=112>
Understanding GPS/GNSS: Principles and Applications, Third Edition, edited by Elliott D. Kaplan, and Christopher Hegarty, Artech House, 2017. ProQuest Ebook Central, <https://ebookcentral.proquest.com/lib/ujaen/detail.action?docID=5430709>.
- [27] AristaSur, “Cómo funciona el sistema de posicionamiento GPS,” AristaSur, Nov. 28, 2014. (<https://www.aristasur.com/contenido/como-funciona-el-sistema-de-posicionamiento-gps#>:

[~:text=La%20constelaci%C3%B3n%20de%20sat%C3%A9lites%20que,terrestre%2C%20y%20se%20denomina%20Navstar\)](#)

- [28] E. Sardon, A. Rius, and N. Zarraoa, “Estimación del contenido total de electrones en la ionosfera usando datos del Sistema de Posicionamiento Global,” Madrid. [Online]. Available: <https://digital.csic.es/bitstream/10261/6637/1/ionosfera.pdf>
- [29] Unity, “LocationService.” UnityDocumentation. [Online]. Available: <https://docs.unity3d.com/ScriptReference/LocationService.html>
- [30] “GIS FAQ Q5.1: Great circle distance between 2 points.” [Online]. Available: <http://www.movable-type.co.uk/scripts/gis-faq-5.1.html>
- [31] J. Lorenzo Ramírez, “Experimentación en Física con dispositivos móviles,” experimentaciolliure.com. 2019. [Online]. Available: <https://experimentaciolliure.files.wordpress.com/2020/05/exfidismo-2019-3.pdf>
- [32] Vázquez Fernández-Pacheco, A. S. Ramos de la Flor, F. & Fernández Rodríguez, R. (2016). Robótica educativa: prácticas y actividades.. RA-MA Editorial. <https://elibro.net/es/lc/ujaen/titulos/106409>
- [33] Iván Linares, “El giroscopio, ¿qué es y qué ocurre si mi móvil no lo tiene?,” El español, Jul. 04, 2016. [Online]. Available: https://www.elespanol.com/elandroidelibre/tutoriales/trucos/20160704/giroscopio-ocurre-movil-no/137486934_0.html
- [34] Unity, “Compass.” UnityDocumentation. [Online]. Available: <https://docs.unity3d.com/ScriptReference/Compass.html>
- [35] Unity, “Texture2D.” UnityDocumentation. [Online]. Available: <https://docs.unity3d.com/ScriptReference/Texture2D.html>

Anexo 1. Manual de uso

Funcionamiento de la aplicación.

Inicialmente esta aplicación está pensada para dispositivos Android con una versión superior a la 4.1 y que incorporan acelerómetro y giroscopio. Para poder utilizar esta aplicación será necesario que el dispositivo tenga activado la opción de ubicación para poder acceder a los datos GPS.

Al comienzo de la experiencia nos mostrará por pantalla el menú principal de la aplicación. Dentro de ella tenemos las opciones de inicio, monumentos y salida.



Figura 74. Menú principal de la aplicación.

Monumentos

Al seleccionar la opción “Monumentos” aparecerá el mapa de la ciudad, en este caso Jaén, con los distintos puntos que la aplicación oferta como visita turística.

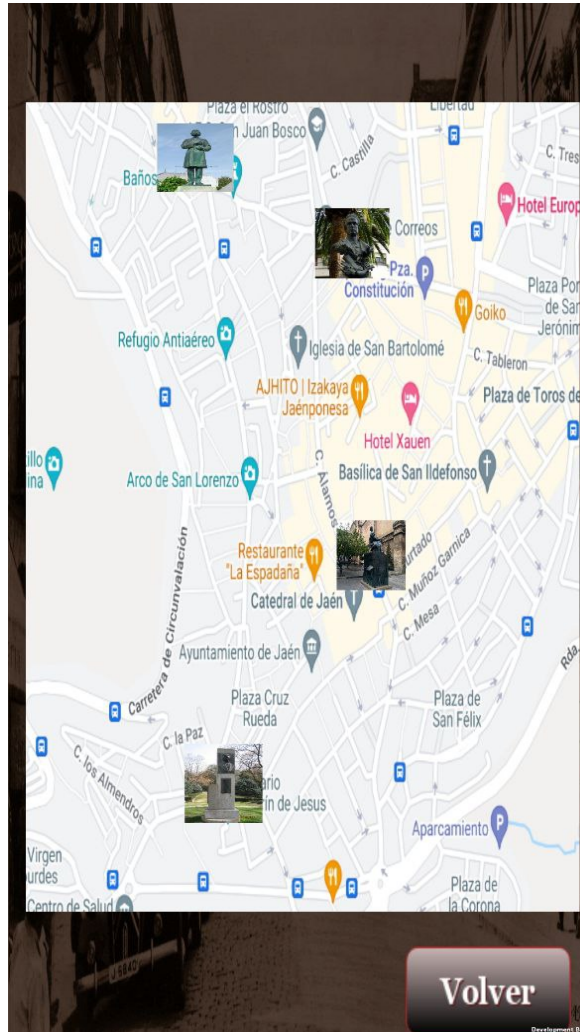


Figura 75. Mapa de Jaén.

Si el usuario toca encima de una de las estatuas, aparecerá información relevante a la estatua, sea esta la imagen real y la dirección de ésta así como el enlace a Google Maps.



Figura 76. Información de ubicación del monumento

El usuario puede volver en cualquier momento al menú principal gracias al botón “Volver” ubicado abajo a la derecha de la pantalla.

Inicio

Al pulsar esta opción, la aplicación actuará dependiendo del lugar en el que se encuentre el usuario.

Si el usuario se encuentra lejos de la estatua más cercana, aparecerá la siguiente interfaz. En ella, al igual que ocurría en Monumentos, se mostrará información sobre el usuario, incluyendo esta vez la distancia a la que se encuentra.



Figura 77. Interfaz mostrada al ubicarse lejos de los monumentos.

Si por el contrario se encuentra cerca de alguno de los puntos de ruta (estos puntos se encuentran en zonas estratégicas del mapa que sirven para guiar al usuario), aparecerá la siguiente interfaz. Aquí se mostrará un extracto del mapa para que el usuario pueda guiarse, además de indicaciones para llegar al monumento más cercano.



Figura 78. Interfaz de guiado.

Una vez el usuario ha llegado a su destino, la aplicación requerirá que éste se ubique frente a la estatua, de forma que puedas ver la cara del personaje histórico (en algunos recorridos se puede aparecer detrás del monumento).



Figura 79. Interfaz mostrada al ubicarse cerca del monumento.

Cámara

Al detectar que está en el sitio, aparecerá por pantalla el nombre del personaje histórico que se está visitando. Una de las opciones disponibles consiste en la realización de fotografías a la obra, pulsando el botón “cámara” ubicado debajo de la pantalla. Al realizarse la imagen, aparecerá la siguiente interfaz.

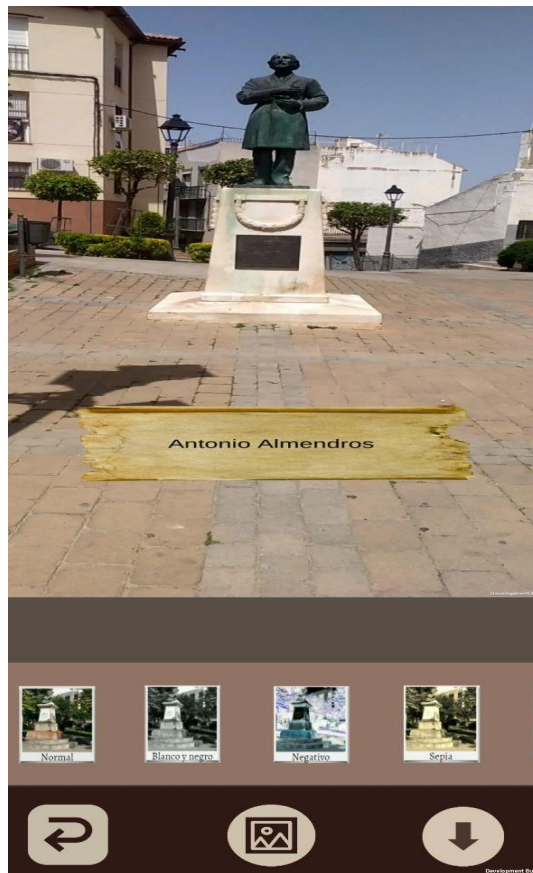


Figura 80. Interfaz de filtrado de imagen.

Una vez tomada la imagen, podrá incorporar alguno de los filtros propuestos simplemente pulsando en el botón de “filtro” ubicado abajo en el centro de la aplicación. Si está contento con el resultado, puede guardar la imagen pulsando abajo a la derecha en el botón de guardar. En caso contrario, pulse el botón de volver. Una vez haya pulsado este botón, desaparecerá la imagen tomada por lo que guarde la imagen en caso de duda.

Izquierda

Si lo que desea saber relativo a dicha celebridad es su biografía, gire su móvil hacia la izquierda. Aparecerá por pantalla un bot de mensajería, por lo que, si quieres saber algo sobre él, solamente has de pulsar el rectángulo redondeado de escritura.

Tras ello aparecerán las distintas preguntas disponibles por lo que, una vez elegida la pregunta, selecciona y pulsa el botón de envío. El personaje histórico te responderá en cuestión de segundos.

The screenshot shows a mobile application interface for a historical figure's biography. At the top, there is a profile card for Antonio Almendros, marked as 'Online'. Below this, a text box states: 'Mis obras principalmente se encontraban recogidas en revistas provinciales y nacionales tales como El Cero o La Semana.' This is followed by a pink header 'Almendros Aguilar' and a text box: 'Si quieres conocer algunas de mis obras, gire su dispositivo hacia la derecha.' Below this is a pink header 'Yo' and a text box asking '¿A qué te dedicabas?'. This is followed by another pink header 'Almendros Aguilar' and a text box: 'Pues además de político, siendo durante un breve período de tiempo diputado provincial de Jaén y secretario de su Gobierno Civil, fui poeta y dramaturgo.' Below this is another pink header 'Almendros Aguilar' and a text box: 'En 1862 consigue ser nombrado académico de la Real Academia Sevilla de Buenas Letras.' At the bottom, there is a grid of four questions with checkboxes: '¿Cuándo naciste?', '¿Dónde se publicaban tus trabajos?', '¿A qué te dedicabas?', and '¿Cuándo moriste?'. The '¿Cuándo moriste?' checkbox is checked. Below the grid is a large yellow button with the text '¿Cuándo moriste?' and a right arrow icon. A small 'Development Build' watermark is visible in the bottom right corner.

Figura 81. Biografía del autor.

Derecha

Si lo que desea es conocer alguna de sus obras, gire su dispositivo hacia la derecha. Le deberá aparecer por pantalla la siguiente interfaz.



Figura 82. Obras del personaje histórico.

En ella podrá ver las dos obras más significativas del célebre personaje. Si desea conocer más acerca de alguna de las obras, no tiene más que pulsar sobre la imagen.

En esta parte aparecerá tanto el nombre de la obra e imagen de la obra, como un extracto de la obra (en algunos casos) de forma escrita y sonora. Si desea reproducir el extracto sonoro, sólo ha de pulsar en el botón de play. Si lo desea pausar, se debe seleccionar nuevamente el mismo botón.

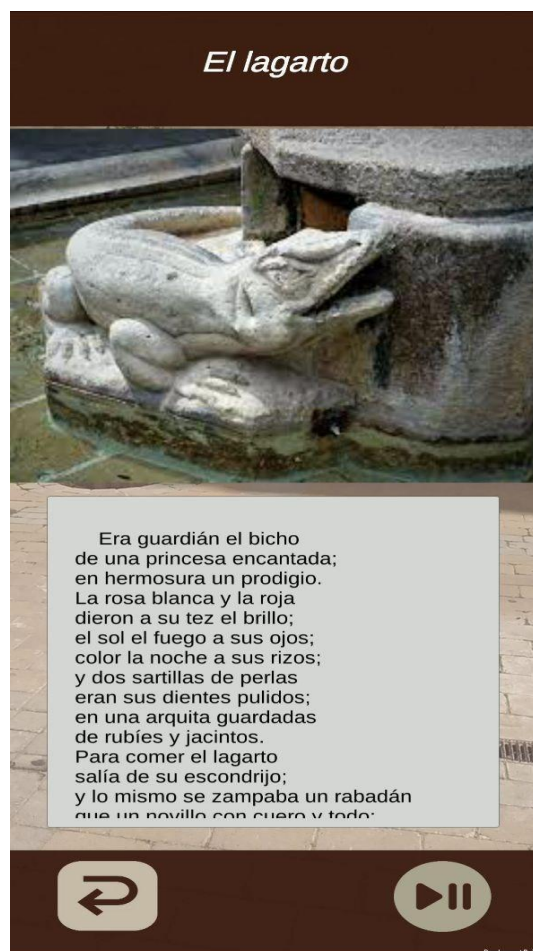


Figura 83. Información de la obra “El lagarto”.

Una vez hayas consumido toda la información propuesta, aparecerá por pantalla la siguiente indicación para ir hacia la siguiente persona ilustre. Si ha finalizado la ruta o no desea hacer la ruta al completo, puede salirse en cualquier momento pulsando en “Salir”.

Anexo 2. Instalación de Unity

Este Anexo tiene como objetivo indicar al lector los pasos a seguir para obtener la aplicación de Unity, así como los primeros pasos dentro de la aplicación.

Se comenzará primero instalando la plataforma Unity Hub desde la siguiente página web:
<https://unity.com/es/download>

Una vez dentro de la página, se recomendaría que comenzaras creandote una cuenta en la plataforma Unity.



The image shows the Unity ID login interface. At the top left is the Unity logo followed by 'Unity ID'. The main heading is 'Iniciar sesión en tu Unity ID'. Below this is a link: 'Si no tienes una Unity ID, [crea una](#).' There are two input fields: 'Correo electrónico' and 'Contraseña'. Below the email field is a green 'Iniciar sesión' button. To the right of the button is a checkbox labeled 'Recuérdame' and two links: '¿Olvidaste tu contraseña?' and 'Ayuda'.

Figura 84. Imagen de inicio de sesión de Unity.

Tras ello, deberás elegir la aplicación para el sistema operativo que porte el dispositivo desde el cual se vaya a instalar.

1. Descargar Unity Hub

Follow the instructions onscreen for guidance through the installation process and setup.

[Download Hub V3 for Windows](#)^{BETA}

[Download Hub V3 for Mac](#)^{BETA}

[Download for Linux](#)^{BETA}

[Download for Windows](#)

[Download for Mac](#)

Figura 85. Página de descargas de Unity 1.

Una vez instalado, elige la versión de Unity que más te interese, sea la última versión o una versión estable accediendo desde este link: [Unity - Download Archive \(unity3d.com\)](https://unity3d.com/download/archive)

Unity 2021.x	Unity 2020.x	Unity 2019.x	Unity 2018.x	Unity 2017.x	Unity 5.x	Unity 4.x	Unity 3.x
Unity 2021.2.9 27 Jan, 2022							
Unity Hub	Downloads (Win)	Downloads (Mac)	Downloads (Linux)	Release notes			
Unity 2021.2.8 13 Jan, 2022							
Unity Hub	Downloads (Win)	Downloads (Mac)	Downloads (Linux)	Release notes			

Figura 86. Página de descargas de Unity 2.

Otra forma de poder instalar la aplicación de Unity es a través de la plataforma Unity Hub, abriendo la aplicación y posicionándose en la opción instalar.

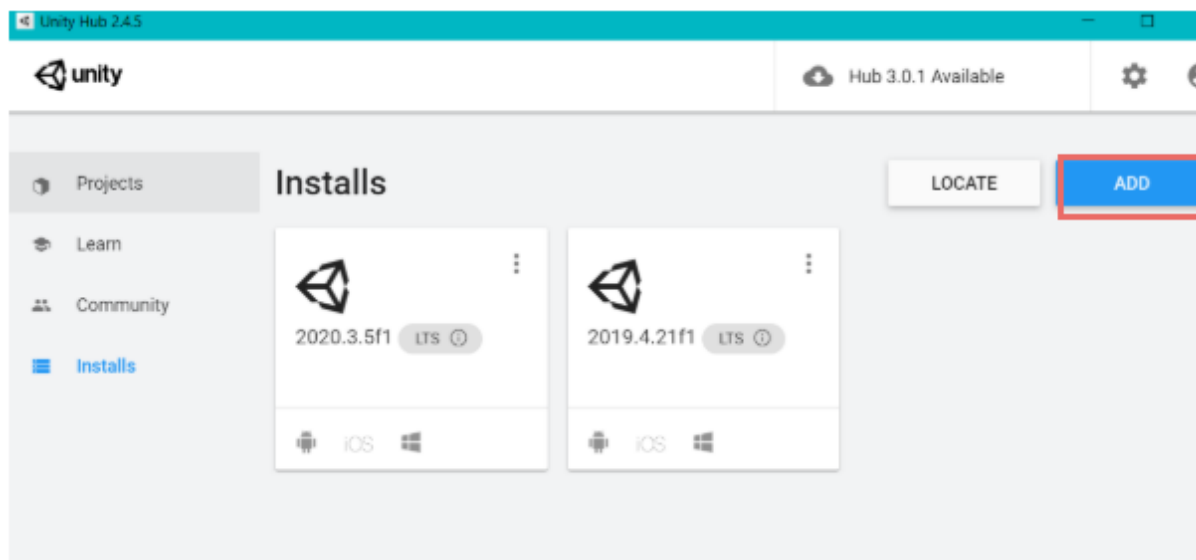


Figura 87. Instalador de Unity.

Al pulsar el botón añadir, se mostrará por pantalla todas las versiones de Unity disponibles actualmente. Mi recomendación es coger siempre la versión más estable y no la última versión ya que puede contener errores.

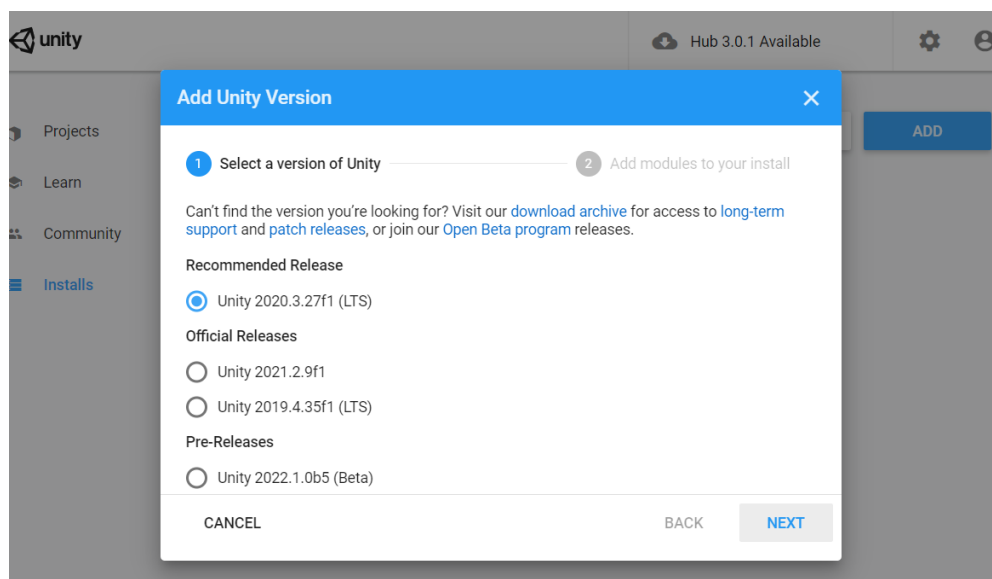


Figura 88. Panel de versiones de Unity.

Tras pulsar siguiente, tendrás que elegir entre los distintos módulos y plataformas disponibles con las que desees trabajar. Cuando hayas seleccionado todo lo que necesites, deberás pulsar siguiente.

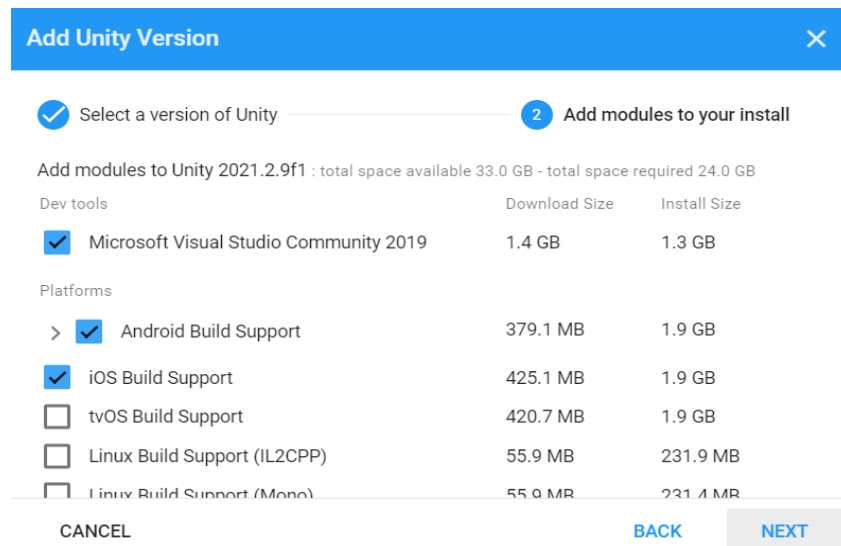


Figura 89. Panel de módulos de Unity.

Una de las opciones que ofrece Unity es la posibilidad de descargar Visual Studio para poder programar scripts de la plataforma. Si no estás interesado, puedes pulsar directamente siguiente.

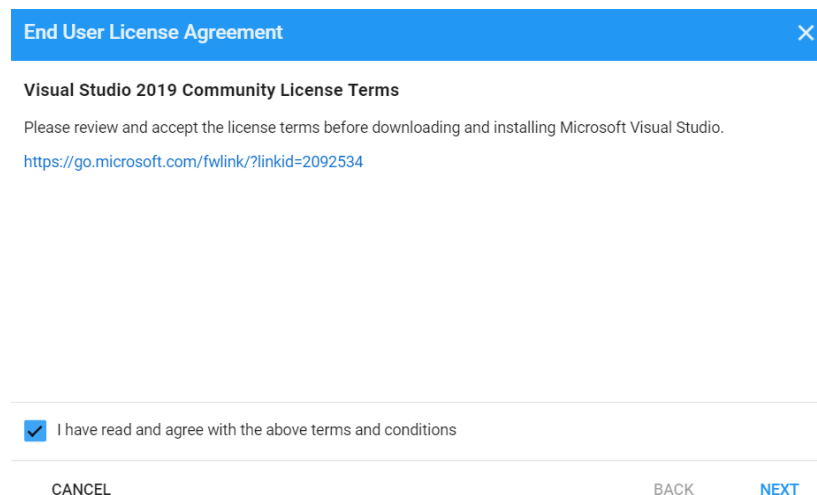


Figura 90. Opción de instalación de Visual Studio.

Por último, necesitarás aceptar el término de condiciones de la plataforma.

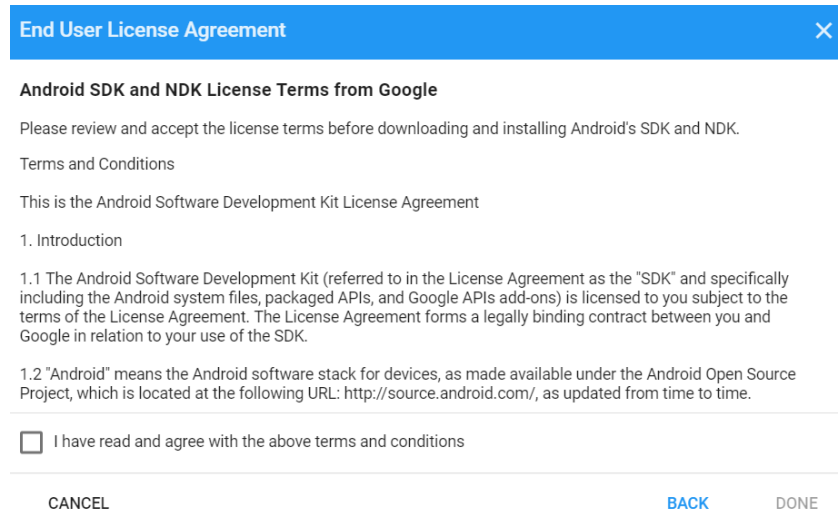


Figura 91. Licencia de usuario.

Si en algún momento necesitas añadir algún módulo extra, solamente tienes que pulsar en los tres botones ubicado en la parte superior de la versión que desees ampliar.

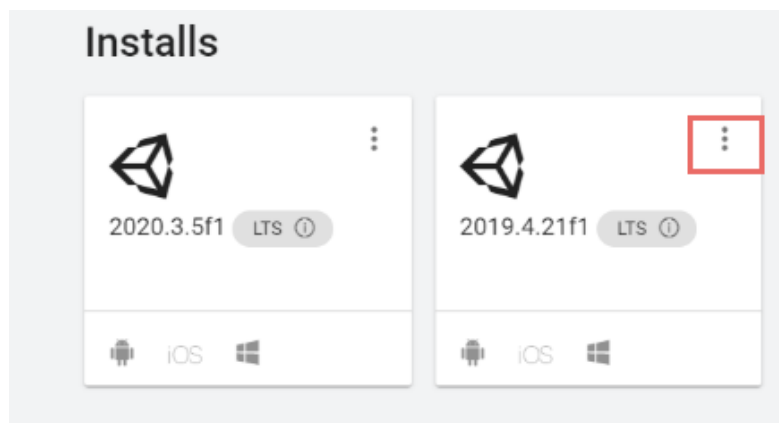


Figura 92. Ampliación de módulos de Unity.

Una vez hecho esto ya puedes comenzar a crear un proyecto con Unity. Para ello accede desde Unity Hub al panel de Proyectos. Si es la primera vez que accedes a Unity, puedes acceder a crear un nuevo proyecto pulsando en Nuevo.

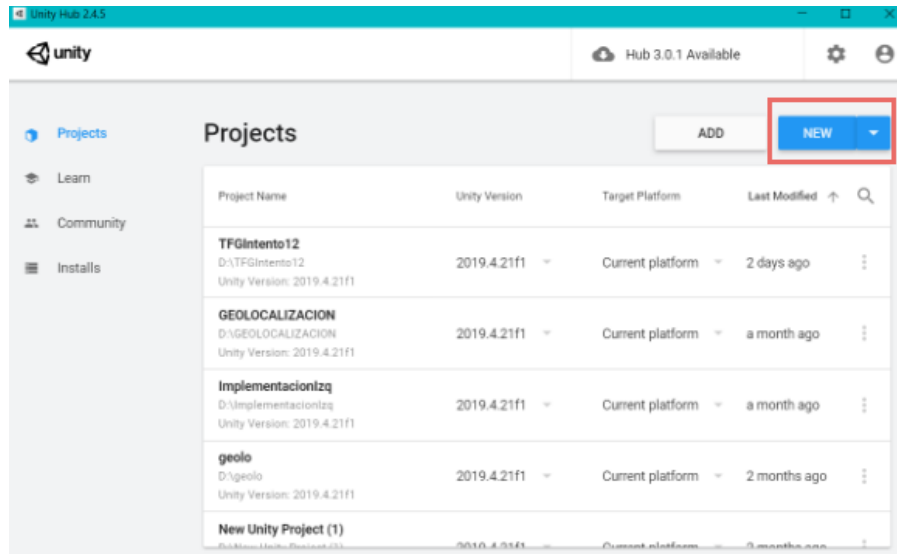


Figura 93. Creación de nuevo proyecto.

Selecciona la opción que más desees (en este caso se ha optado por la opción de 3D), elige el título y la localización en la que se ubicará el proyecto y pulsa en el botón de crear.

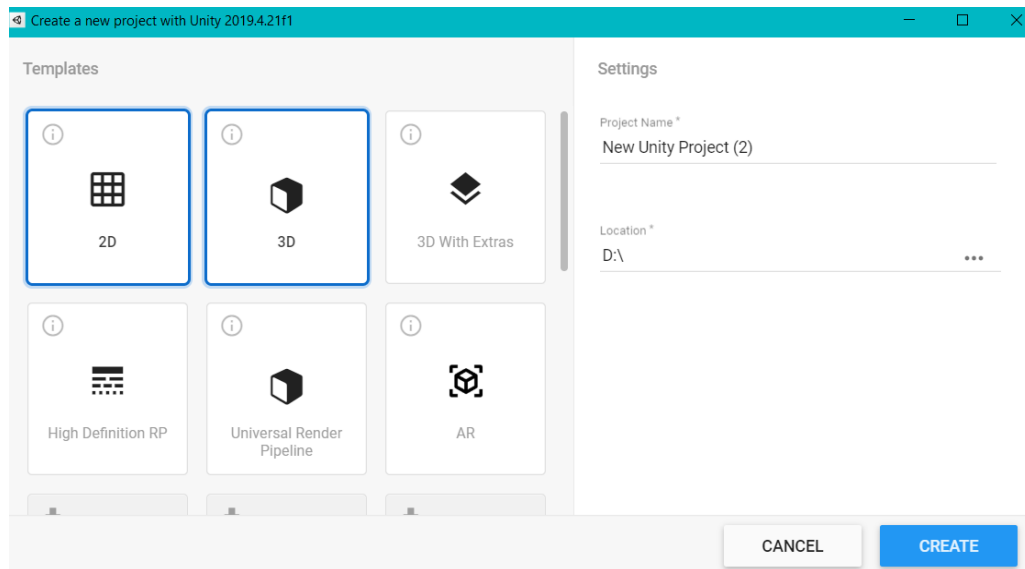


Figura 94. Elección del tipo de proyecto.

Después de ello, aparecerá por pantalla la aplicación de Unity.

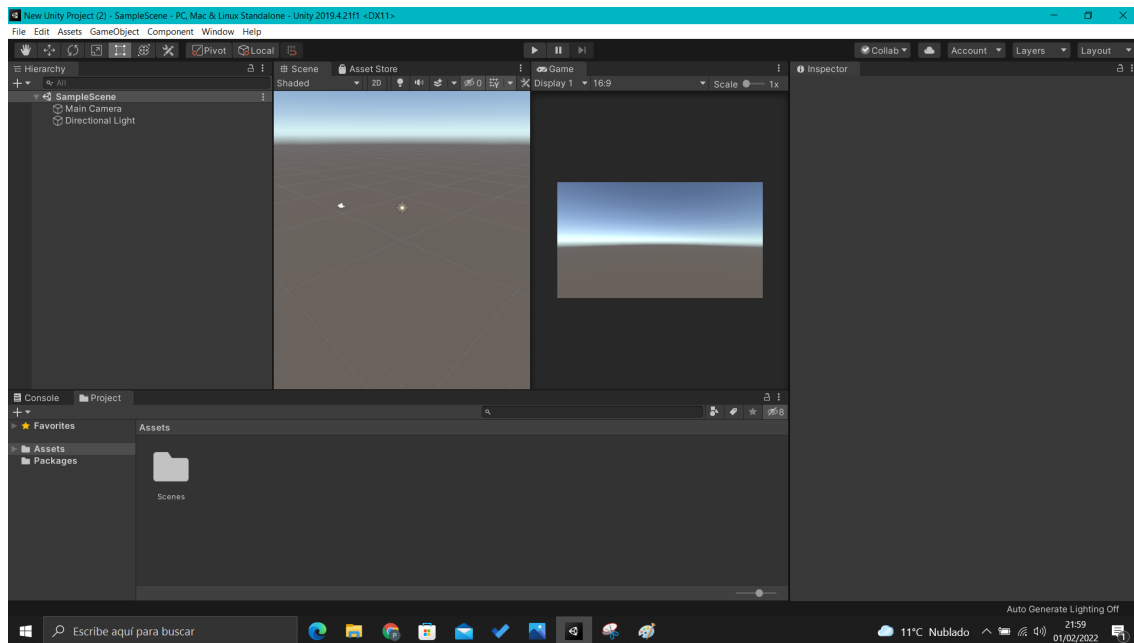


Figura 95. Interfaz de usuario de Unity.

Antes de comenzar se recomienda configurar varios datos referentes a la aplicación. Para ello accede a File >> Build Settings.

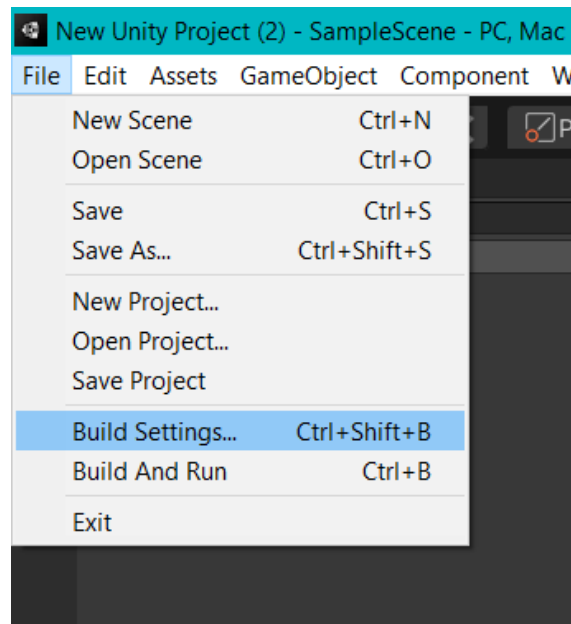


Figura 96. Configuración de las características del proyecto.

Se mostrará la siguiente pantalla. Se encuentra dividida en dos partes: Scenes in Build y en Platform. En la sección Scenes in Build puedes añadir todas las escenas que hayas creado hasta ahora pulsando en Add Open Scenes.

En la sección Platform, selecciona la plataforma en la que desees utilizar la aplicación, en este caso se ha seleccionado Android. Para que estos cambios sean guardados, pulsa en el botón Switch Platform ubicado en la esquina inferior derecha de la pantalla.

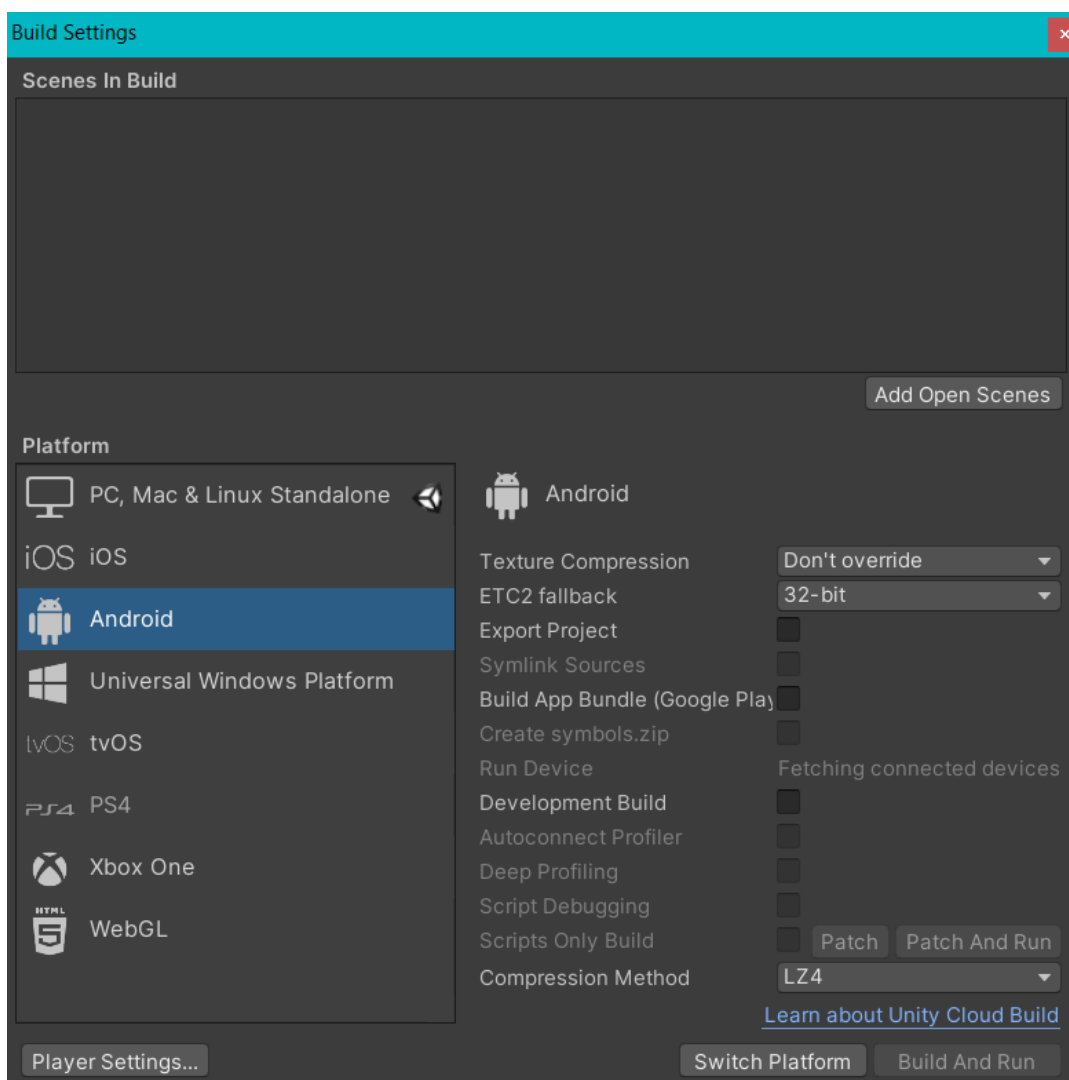


Figura 97. Panel Build Settings.

Para configurar más rasgos de la aplicación, pulsa en Player Settings, apareciendo el siguiente panel por pantalla.

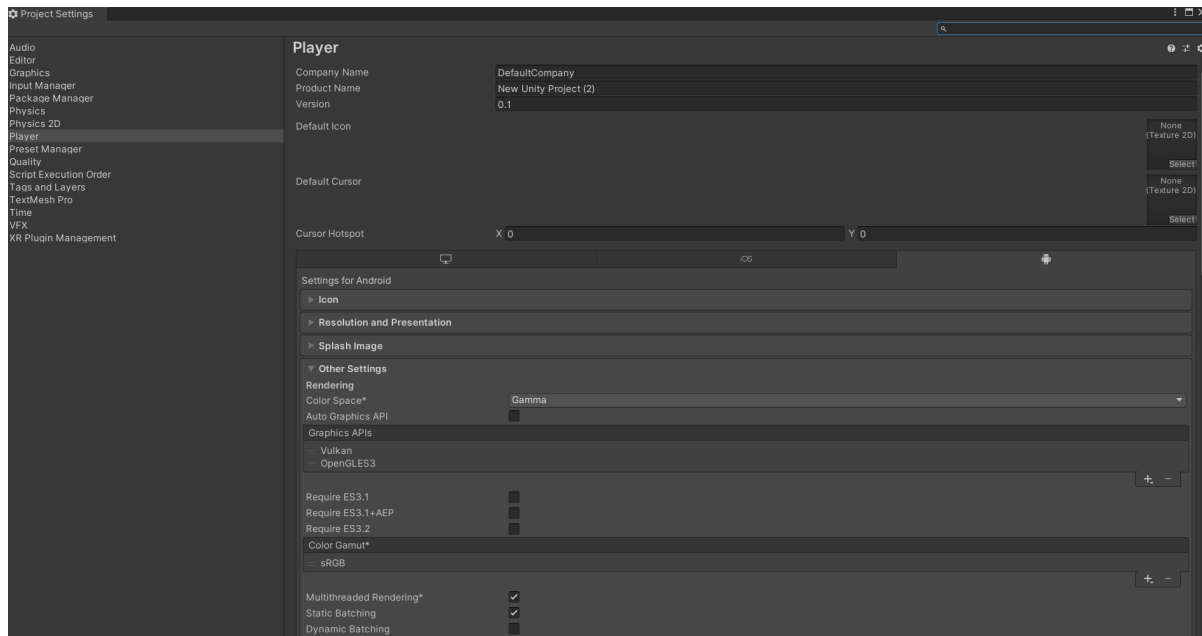


Figura 98. Panel Manager.

Configure los datos de la aplicación tales como el nombre de la app y la compañía a la que pertenece en los parámetros indicados en la siguiente imagen.

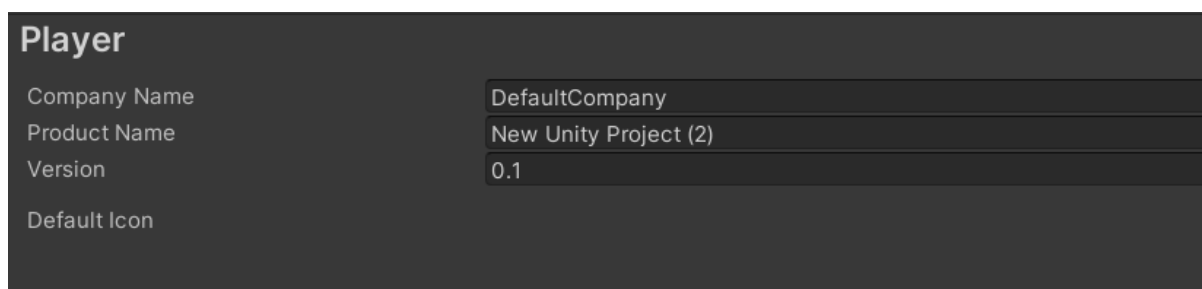


Figura 99. Panel Player Zoom.

Conforme rellenes dichos parámetros, aparecerán modificados en el apartado Identificación, ubicado dentro de Panel. Aquí puedes modificar la versión de tu aplicación, así

como el nivel mínimo de API que soportará la aplicación en Android. Se recomienda poner mínimo nivel 7.

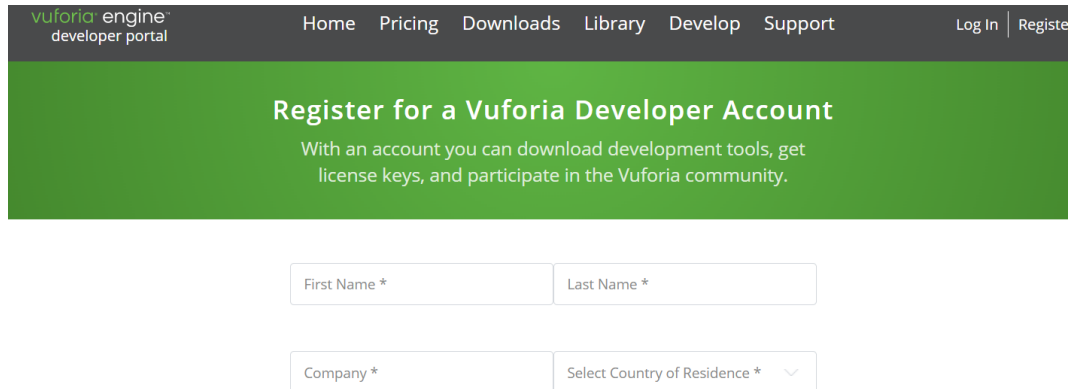
Identification	
Package Name	com.DefaultCompany.NewUnityProject2
Version*	0.1
Bundle Version Code	1
Minimum API Level	Android 4.4 'KitKat' (API level 19)
Target API Level	Automatic (highest installed)

Figura 100. Apartado de identificación de la aplicación.

Una vez hecho esto ya puedes utilizar Unity.

Anexo 3. Instalación de Vuforia

Antes de descargar el paquete de Vuforia es necesario registrarse dentro de la plataforma a través de este link: [Vuforia Developer Portal](#)



The screenshot shows the Vuforia Developer Portal registration page. At the top, there is a navigation bar with links: Home, Pricing, Downloads, Library, Develop, Support, Log In, and Register. Below the navigation bar, there is a green banner with the text "Register for a Vuforia Developer Account" and a subtext: "With an account you can download development tools, get license keys, and participate in the Vuforia community." Below the banner, there are four input fields: "First Name *" and "Last Name *" are side-by-side, and "Company *" and "Select Country of Residence *" are side-by-side. The "Select Country of Residence *" field has a dropdown arrow.

Figura 101. Página de registro de Vuforia.

Tras crear la cuenta ya podrás descargar el paquete de Vuforia. Para mayor facilidad puedes acceder desde aquí [SDK Download | Vuforia Developer Portal](#). Selecciona la siguiente opción ya que está prevista para la incorporación en Unity. Se descargará el archivo de tipo Unity Project.

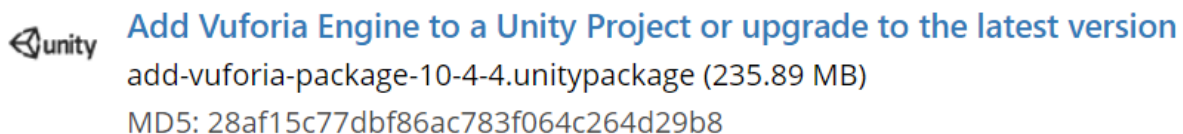


Figura 102. Instalador de Vuforia.

Una vez que ya se ha descargado la aplicación de Vuforia accederemos a la aplicación de Unity y abriremos el proyecto en el que deseemos integrar realidad aumentada.

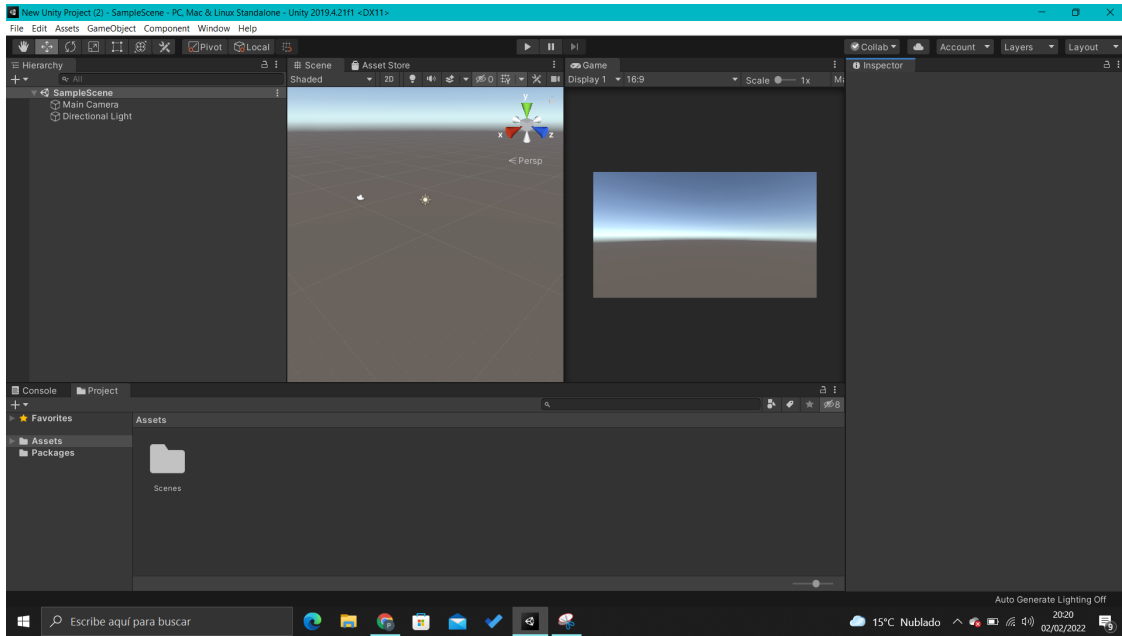


Figura 103. Interfaz de Unity.

Dentro de la aplicación tendremos que importar Vuforia, por lo que se deberá seleccionar el botón de Assets >> Import Package >> Custom Package.

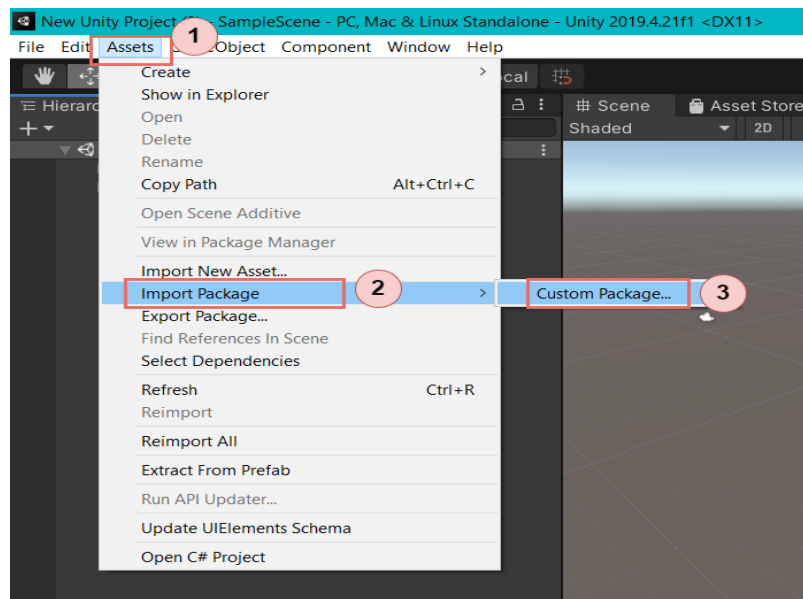


Figura 104. Instalación de Vuforia 1.

Aparecerá por pantalla las nuevas carpetas en Assets incluyendo los Plugins, Resources y una carpeta de Vuforia en el menú que contendrá distintas subcarpetas. Seleccionamos Import.

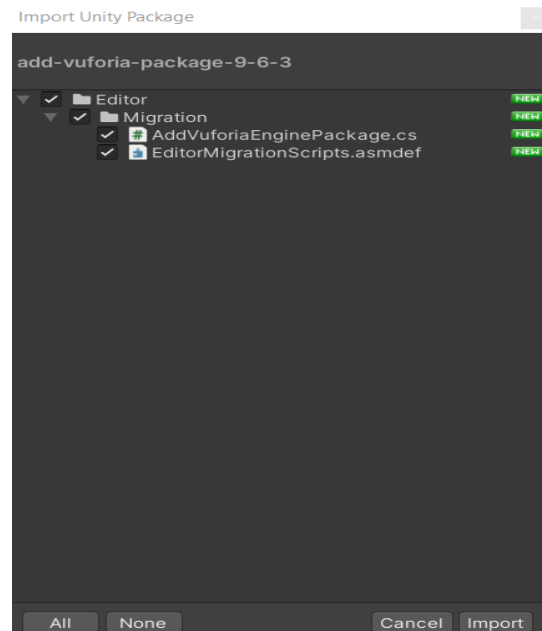


Figura 105. Instalador de Vuforia 2.

Una vez instalado aparecerá el siguiente panel:

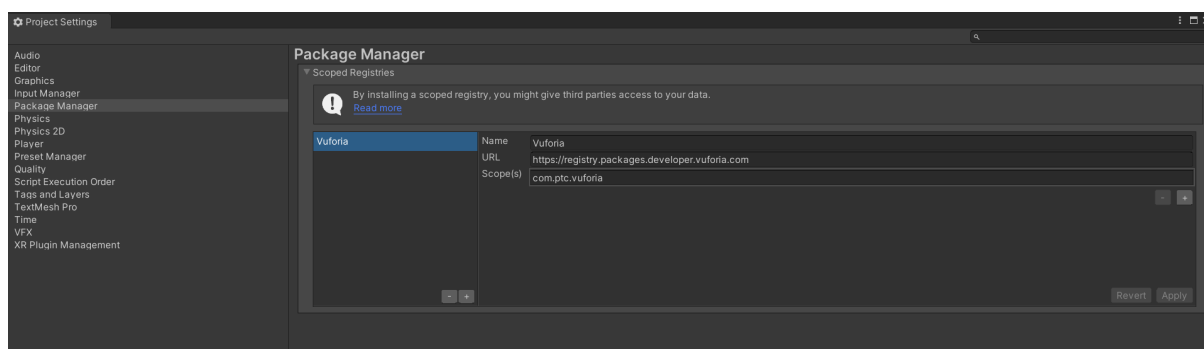


Figura 106. Panel Package Manager.

Si la versión que has instalado no es la última versión, podrás actualizarlo desde este panel. Una vez importado se tendrá que configurar Vuforia. Para ello pulsamos en Window >> Vuforia Configuration.

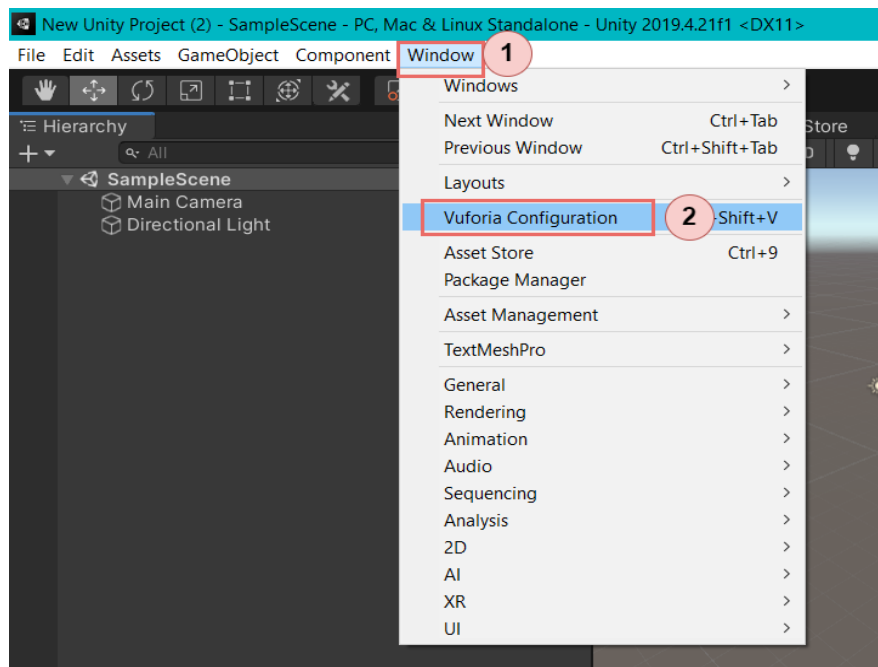


Figura 107. Configuración de Vuforia dentro de Unity.

Al pulsar aparecerá en el panel de inspector de Unity la información sobre la funcionalidad de Vuforia como se muestra en la imagen siguiente.

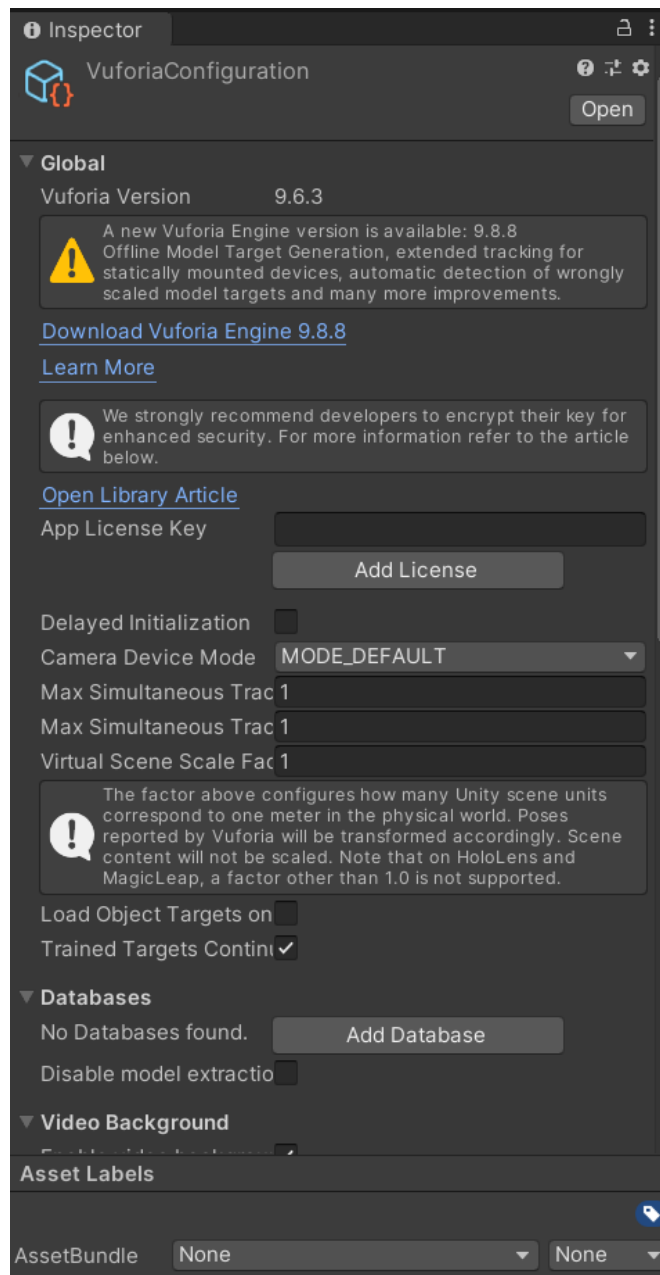


Figura 108. Vuforia Configuration.

Una vez aquí se incluirá la llave de Vuforia por lo que se necesitará entrar en la página de Vuforia. Si aún no tienes creada una, podrás crearla haciendo clic en Add License, redirigiendo este hasta la sección concreta de Vuforia.

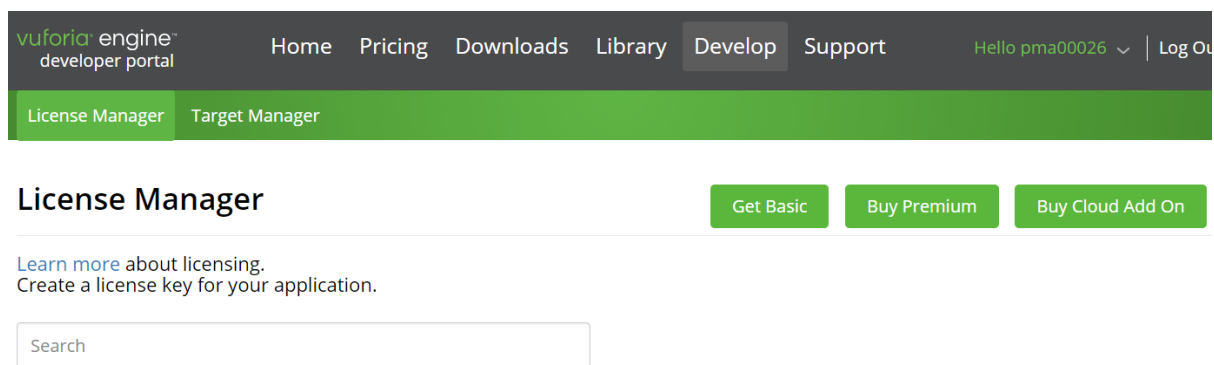


Figura 109. Creación de licencia de Vuforia.

Podrás elegir entre las distintas opciones de Vuforia para obtener dicha licencia. En este caso se ha elegido el paquete básico. Una vez hayas seleccionado la opción que más te interese, establece el nombre de tu licencia, lee las condiciones de Vuforia y por último acepta.

The image shows a form titled 'Add a license key to your Basic plan'. It includes a text input field for 'License Name *' with a note below it stating 'You can change this later'. Below the input field, the 'License Key' section lists the following details: 'Basic', 'Price: No Charge', 'Reco Usage: 1,000 per month', 'Cloud Targets: 1,000', 'VuMark Templates: 1,000 Active', and 'VuMarks: Unlimited'. At the bottom of the form, there is a checkbox with the text 'By checking this box, I acknowledge that this license key is subject to the conditions of the Vuforia Developer Agreement.' and two buttons: 'Cancel' and 'Confirm'.

Figura 110. Adquisición de la licencia de Vuforia.

Obtendrás el siguiente resultado:

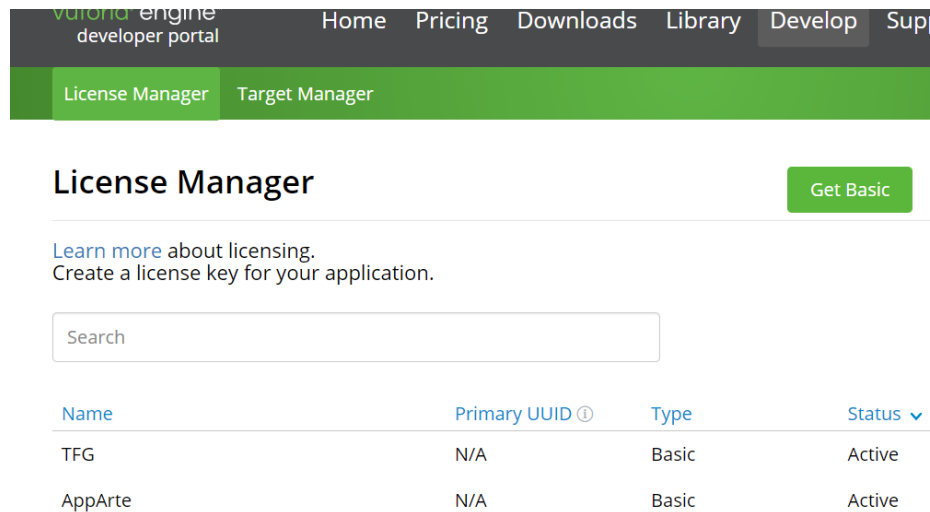


Figura 111. License Manager.

Clica sobre el nombre de la licencia creada y copiala.

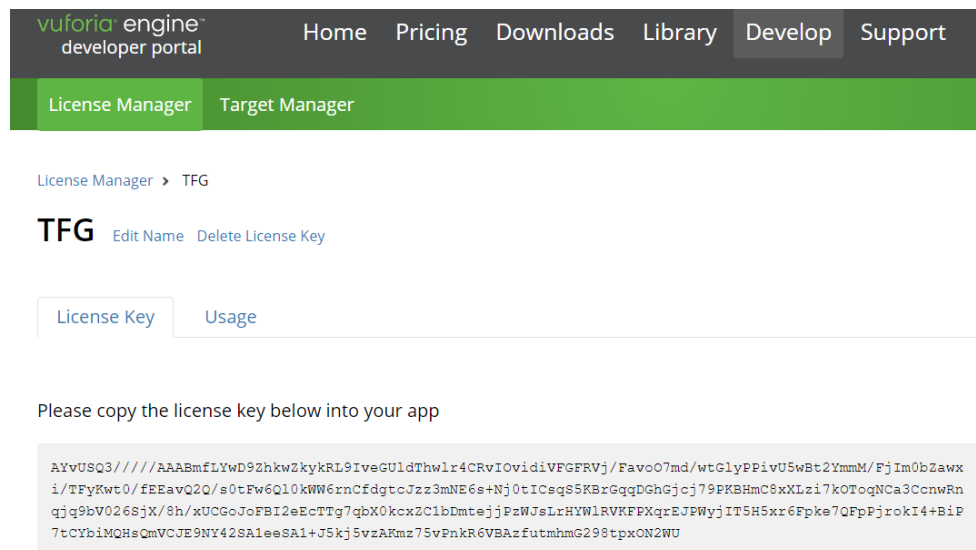


Figura 112. Licencia de Vuforia.

Pégala en el apartado App License Key.

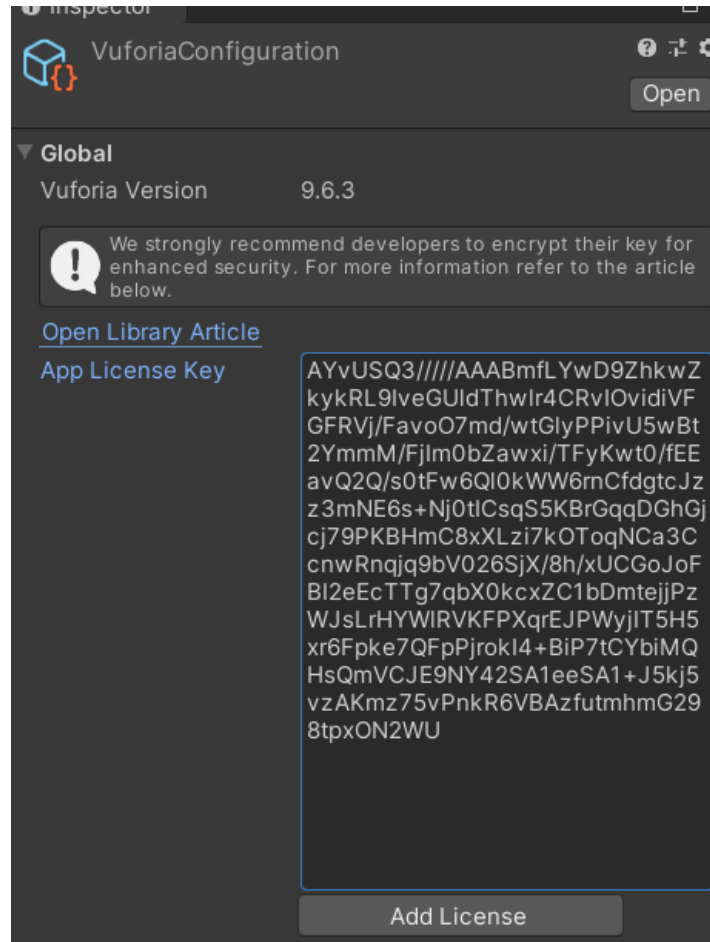


Figura 113. Incorporación de la licencia de Vuforia en Unity.

Si deseas incluir marcadores de imágenes, elementos 3D, podrás crearla dentro de la aplicación de Vuforia clicando en Add Database. Una vez creada, deberás pulsar el botón de descarga, con el formato de UnityPackage e instalar de la misma forma que se ha instalado el paquete de Vuforia.

Uno de los problemas más frecuentes es que la aplicación no detecta la webcam del ordenador por lo que será necesario modificar el archivo. Verifica si Vuforia detecta tu webcam dentro de Vuforia Configuration.

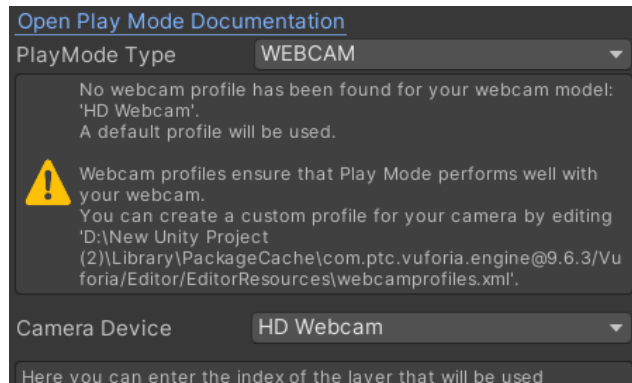


Figura 114. Perfil de la webcam.

Si vuforia no detecta tu web cam acceda a la carpeta de tu proyecto y siga el siguiente path:

\<<Nombreproyecto>>\Library\PackageCache\com.ptc.vuforia.engine@9.6.3\Vuforia\Editor\EditorResources y abra el archivo webcamprofiles.xml

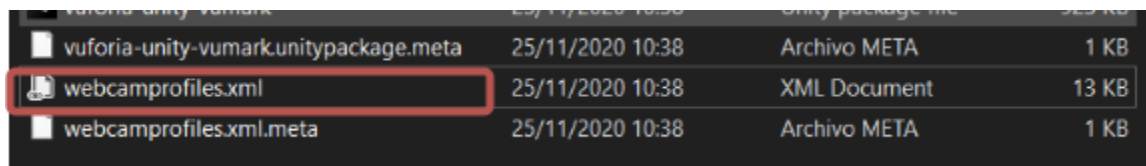


Figura 115. Archivo webcamprofiles.xml 1.

Una vez abierto aparecerá lo siguiente:

```
<?xml version="1.0"?>
profiles>
  <!-- this profile will be used if no specific profile could be found -->
  <default>
    <windows>
      <!-- size of the web cam texture requested from Unity -->
      <requestedTextureWidth>640</requestedTextureWidth>
      <requestedTextureHeight>480</requestedTextureHeight>
      <!-- size of the texture Unity's web cam texture will be rescaled to -->
      <resampledTextureWidth>640</resampledTextureWidth>
      <!-- <resampledTextureHeight>480</resampledTextureHeight> -->
    </windows>
    <osx>
      <!-- size of the web cam texture requested from Unity -->
      <requestedTextureWidth>640</requestedTextureWidth>
      <requestedTextureHeight>480</requestedTextureHeight>
      <!-- size of the texture Unity's web cam texture will be rescaled to -->
      <resampledTextureWidth>640</resampledTextureWidth>
      <!-- <resampledTextureHeight>480</resampledTextureHeight> -->
    </osx>
  </default>
```

Figura 116. Archivo webcamprofiles.xml 2.

Desplázate hasta el final de archivo y copie lo siguiente:

```
<webcam deviceName="FaceTime HD Camera (Built-in)">
  <windows>
    <!-- size of the web cam texture requested from Unity -->
    <requestedTextureWidth>640</requestedTextureWidth>
    <requestedTextureHeight>480</requestedTextureHeight>
    <!-- size of the texture Unity's web cam texture will be rescaled to -->
    <resampledTextureWidth>640</resampledTextureWidth>
  </windows>
  <osx>
    <!-- size of the web cam texture requested from Unity -->
    <requestedTextureWidth>1280</requestedTextureWidth>
    <requestedTextureHeight>720</requestedTextureHeight>
    <!-- size of the texture Unity's web cam texture will be rescaled to -->
    <resampledTextureWidth>640</resampledTextureWidth>
  </osx>
</webcam>
```

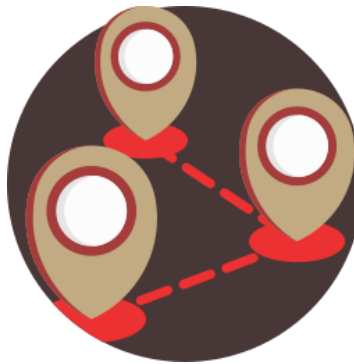
Figura 117. Archivo webcamprofiles.xml 3.

Pégalo tras la etiqueta </webcam> y modifica el deviceName por el nombre de su webcam. Una vez hecho esto, guarde el archivo y acceda a la configuración de Vuforia para verificar que los cambios se han efectuado correctamente. En caso contrario, repita el mismo proceso.

Anexo 4. Prueba de funcionamiento

En este anexo se dará las credenciales para poder acceder a las pruebas de funcionamiento de la aplicación. Se han dividido en dos vídeos: Representación de rutas e interacción con el monumento. Solamente debes pulsar la imagen para dirigirte al video en cuestión.

Representación de rutas



En este video se mostrará el funcionamiento de guiado hacia los distintos monumentos propuestos.

Interacción con el monumento



En este video se mostrará las formas que tiene el usuario de interactuar con los distintos aumentos propuestos en esta aplicación.