UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Aplicación de realidad aumentada para la divulgación histórica y arquitectónica

Trabajo Fin de Grado

Grado en Tecnología Digital y Multimedia

AUTOR/A: Pérez Manzaneda, Pau

Tutor/a: Cerdá Boluda, Joaquín

CURSO ACADÉMICO: 2022/2023

Resumen

Este trabajo trata sobre la adaptación de la historia de una ciudad, en específico ’s-Hertogenbosch, en Países Bajos, a tecnologías actuales e innovadoras.

Se plantea el desarrollo de una aplicación que hace uso de diversas APIs de Google mediante el motor de Unity para situar un modelo en AR en unas coordenadas específicas, siendo estas las de la catedral de la ciudad, la Sint-Janskathedraal, la cual sufrió un incendio y se reconstruyó.

Para este trabajo se superpone un modelo en 3D de la torre en su antiguo estilo sobre la actual. Esto hace posible que los ciudadanos puedan apreciar el cambio estilístico entre ambas. Además, cuenta con aspectos interactivos y educativos, y es que, sobre la torre 3D, se muestran diversos focos de fuego, mediante los cuales, si se presiona sobre ellos, aparece una pregunta sobre la ciudad y sus habitantes a lo largo de su historia.

El principal objetivo es conectar datos digitales sobre la historia de la ciudad, sus habitantes y visitantes y vincularlos con la actualidad. De este modo, las personas, los objetos y los acontecimientos que han sido decisivos para la historia de ’s-Hertogenbosch pueden ser visualizados y conectados con el presente de manera más interactiva y cercana al ciudadano actual.

Resum

Aquest treball tracta sobre l’adaptació de la història d’una ciutat, en específic ’s-Hertogenbosch, en Països Baixos, a tecnologies actuals i innovadores.

Es planteja el desenvolupament d’una aplicació que utilitza diverses APIs de Google mitjançant el motor de Unity per a situar un model en AR en unes coordenades específiques, seguent aquestes les de la catedral de la ciutat, la Sint-Janskathedraal, la qual va sofrir un incendi i va ser reconstruïda.

Per a aquest treball se superposa un model en 3D de la torre en el seu antic estil sobre l’actual. Això fa possible que els ciutadans puguen apreciar el canvi estilístic entre les dos. A més, compta amb aspectes interactius i educatius, i és que, sobre la torre 3D, es mostren diversos focus de foc, mitjançant els quals, si es pressiona sobre ells, apareix una pregunta sobre la ciutat i els seus habitants al llarg de la seua història.

El principal objectiu és connectar dades digitals sobre la història de la ciutat, els seus habitants i visitants i vincular-los amb l’actualitat. D’aquesta manera, les persones, els objectes i els esdeveniments que han sigut decisius per a la història de ’s-Hertogenbosch poden ser visualitzats i connectats amb el present de manera més interactiva i pròxima al ciutadà actual.

Abstract

This work deals with the adaptation of the history of a city, specifically ’s-Hertogenbosch in the Netherlands, to current and innovative technologies.

It proposes the development of an application that makes use of some Google APIs through the Unity engine to place an AR model at specific coordinates, these being those of the city’s cathedral, the Sint-Janskathedraal, which suffered a fire and was rebuilt.

For this work, a 3D model of the tower in its old style is overlaid on the current one. This makes possible for citizens to appreciate the stylistic change between the two. In addition, there are interactive and educational aspects to the project. On the 3D tower, there are several fire lights which, if you click on them, is displayed a question about the city and its inhabitants throughout its history.

The main objective is to connect digital data about the history of the city, its inhabitants and visitors and to link them to the present day. In this way, people, objects and events that have been decisive for the history of ’s-Hertogenbosch can be visualized and connected to the present in a way that is more interactive and closer to today’s citizen.

Agradecimientos

A mis padres y mi hermana por estar conmigo desde el principio y motivarme a continuar dando pasos hacia delante hasta cuando yo los veía muy cuesta arriba.

A la familia que me eligió y que ya es para siempre, mis amigos, que me han ayudado a despejarme tantas veces, que han soportado mi mal humor en época de exámenes y me han seguido apoyando pese a todo. Aprovecho para agradecer especialmente a Ángel que, desde que cursábamos primaria, me enseñó lo bonito de la informática y el mundo digital (nunca te agradecí cómo me transmitiste tu ilusión y me ilusionaste con ella). Y a Aarón, que aunque parezcamos una pareja de casados de 50 años sin ser nada de eso, nos encanta y nos encantará quejarnos de todo y de todos siempre que tengamos oportunidad. Sé que si me pasa cualquier cosa estaríais en la primera línea de la batalla.

A Diego, mi pareja, que me acompaña a hacer cualquier tontería que se me cruce con la misma ilusión que dos niños comiendo caramelos.

Y a mis compañeros y compañeras, que, pese a todo, me han dado los mejores años de mi vida. A Zaira, por enseñarme que ser *bordes* no siempre es negativo. A Zoé, por enseñarme que un poco de locura y caos nunca sobra en nuestras vidas. A Elena por dejarme que te la líe y aún así me ofrezcas una compañera de buceo. A Claudia por conectarnos juntas para que los trabajos se nos hiciesen más entretenidos, incluso a casi 11.000 km de distancia. A David, a Sergio, a Pascual, a Jordi... a todos, muchas gracias.

Y a quien esté leyendo, que como mis escritores favoritos dicen, nada de esto tendría sentido sin un lector al otro lado.

Índice general

Índice de figuras

Listado de siglas empleadas

API Application Programming Interface.

APK Android Application Package.

AR Augmented Reality.

DAFO Debilidades, Amenazas, Fortalezas y Oportunidades.

FODA Fortalezas, Oportunidades, Debilidades y Amenazas.

ISO International Standarization Organization.

JSON JavaScript Object Notation.

LTS Long Term Support.

MR Mixed Reality.

QR Quick Response.

RGB Red, Green, Blue.

SDK Software Development Kit.

SIG Sistema de Información Geográfica.

SWOT Strenghts, Weaknesses, Opportunities and Threats.

UI User Interface.

url Uniform Resource Locator.

URP Universal Render Pipeline.

VPS Visual Positioning Service.

VR Virtual Reality.

Parte I

Memoria

Capítulo 1

Introducción y Objetivos

1.1. Visión general

"*Aquellos que no pueden recordar el pasado están condenados a repetirlo*". Esta oración fue originalmente formulada por el polímata[1] español Jorge Agustín Nicolás Ruiz de Santayana y Borrás [2] en *La razón en el sentido común*, el primero de sus 5 volúmenes que componen su obra *La vida de la razón* [3]. Esta frase se ha visto empleada en gran variedad de contextos: políticos, económicos, sociales y muchos otros, para hacer referencia a que si una sociedad no aprende de los errores cometidos y las lecciones aprendidas es altamente probable volver a enfrentar situaciones similares en el futuro. Es por ello que se ha de prestar atención a la historia y las experiencias pasadas para evitar caer en los mismos patrones problemáticos y avanzar a un futuro en el que se tomen decisiones más informadas y con la esperanza de obtener resultados más satisfactorios para todos.

El proyecto original fue llamado *Den Bosch Time Machine* [4] y la motivación principal para su desarrollo fue y es la conservación y transmisión correcta de la historia. Pretende proporcionar una visión más profunda sobre cómo las personas antiguamente lidiaban con su entorno y sus problemas contemporáneos. Por lo tanto, encaja bien con temas actuales como la creación y tratamiento responsable de los datos y la sostenibilidad, la adaptación del clima, la economía, la seguridad y la salud.

Además, en este proyecto también encara el desafío de adaptar la historia a las tecnologías digitales actuales, lo que facilita un aprendizaje histórico más cercano a la sociedad actual y más agradable de asimilar e incorporar en el día a día.

1.2. Contexto

Bajo el nombre de *Den Bosch Time Machine* se intenta devolver a la vida a 800 años de historia generando subproductos y contenidos a modo de paraguas para permitir, a todos aquellos interesados, navegar por la ciudad de Den Bosch de manera histórica en el espacio y en el tiempo gracias a archivos históricos, objetos, hallazgos arqueológicos y monumentos en un sistema de información digital.

[1] Del latín moderno *polymathes* 'que sabe mucho'. [1]

Es esta inquietud en la conservación y transmisión cultural a la actualidad la cual genera la producción de este trabajo, el cual se acoge a un proyecto mayor con el mismo objetivo, que, a su vez, se acoge a uno mayor a nivel europeo.

El punto de partida de la idea de una máquina del tiempo proviene de Venecia, donde científicos se han dedicado a digitalizar miles de años de mapas y manuscritos del apogeo de la ciudad con escáneres y algoritmos de última generación. A raíz del proyecto en Venecia, se creó un consorcio europeo de máquinas del tiempo *Time Machine Europe* [5], en el que se han unido más de 30 países y cientos de organizaciones.

1.3. Objetivos

El objetivo principal del consorcio recientemente mencionado es desarrollar nuevas tecnologías para escanear, analizar, acceder, preservar y comunicar el patrimonio cultural a gran escala. Los datos resultantes constituyen la base para la reconstrucción de la evolución histórica de una gran parte de las ciudades europeas y las redes económicas, culturales y migratorias entre estos centros urbanos.

Time Machine Europe reúne las diversas iniciativas locales. Con la máquina del tiempo de Den Bosch se trabaja adicionalmente con otras máquinas del tiempo que se instalan en otras ciudades. En los Países Bajos, esto sucede en Ámsterdam, Limburgo, Utrecht, Frisia y Dordrecht.

Por otro lado, *Den Bosch Time Machine* consta de tres bloques de construcción importantes. Estas actividades principales pueden iniciarse simultáneamente, pero deben cumplirse en el futuro:

- **Una red de datos vinculados**: conectar y cohesionar datos de Erfgoed ’s-Hertogenbosch y otras organizaciones y proyectos, como descripciones de fincas, planos de construcción, registros de población y datos arqueológicos y de historia de la construcción, manuscritos medievales, objetos históricos, imágenes, fragmentos de audio, pinturas e historias personales de los habitantes de Den Bosch.
- **Mapas históricos 2D**: conectar datos abiertos vinculados a mapas en un sistema de información geográfica (SIG) en georreferencia uniforme y vectorización de los mapas históricos de ’s-Hertogenbosch con todas las estructuras históricas conservadas.
- **Reconstrucciones 3D**: recrear la ciudad histórica en 3D de manera que se hagan visibles las transformaciones a través del tiempo y se enriquezca con la cuarta dimensión del tiempo.

Es en este tercer bloque en el que este trabajo ofrece su máximo potencial para historiadores y otros académicos, pero principalmente para la gente de a pie y aquellos ciudadanos con interés por aprender sobre la historia de su ciudad.

Den Bosch Time Machine tiene la finalidad de crear un sistema de información digital aplicable a numerosas situaciones como la investigación, el desarrollo urbano, el turismo o la educación. De esta manera, las personas, objetos y eventos que han determinado la historia de Den Bosch pueden ser investigados, visualizados y conectados con el presente.

1.3.1. Metas específicas: Objetivos del subproyecto

Para el proyecto que se va a tratar en profundidad en este documento las metas u objetivos a alcanzar son mucho más específicas y concisas, pues se trata de un producto dentro del marco del paraguas que abarca *Den Bosch Time Machine*.

El proyecto, de manera mucho más particular comprende sus propios objetivos y metas propias. No obstante, el objetivo principal era conseguir un **prototipo funcional e interactivo** de una aplicación **en realidad aumentada**.

Como objetivos secundarios fueron marcados los siguientes:

- Creación de una aplicación funcional y **entretenida y divertida**.
- **Atraer** a la población de (al menos) Den Bosch **hacia el aprendizaje** de su propia historia.
- Creación de un **producto valioso** para la ciudad de Den Bosch, Países Bajos, y para la máquina del tiempo *Europe Time Machine*.

1.4. Planificación

Por último, en este primer capítulo se han querido dejar marcados unos tiempos y periodos que permitieron la estructuración del desarrollo del proyecto y que a su vez pautaron los límites del trabajo. Unos límites necesarios y que mantenían anclado de manera razonable a la realidad las posibilidades de éxito de la creación de un prototipo viable y funcional.

Para conseguir esto se aplicó una metodología de diseño centrado en el usuario, también conocido como *Design Thinking*. Esta metodología pasa por 5 fases, como seguidamente se puede observar en la figura 1.1.

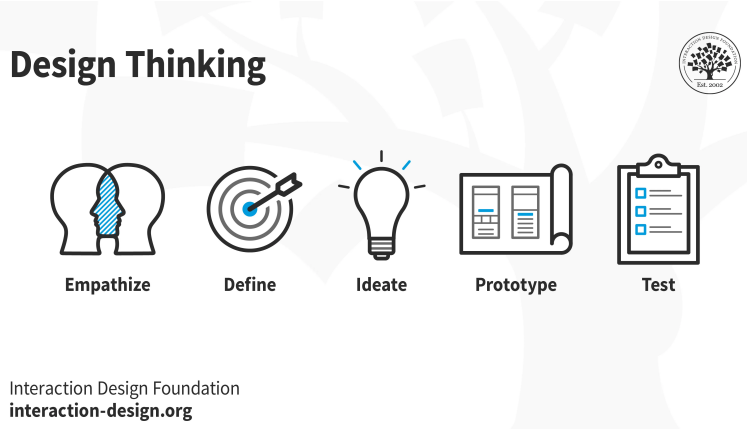


Figura 1.1: Fases de la metodología [6]

Sin embargo, esta metodología no aplica sus fases de manera lineal, sino que las emplea de manera que interactúan entre ellas conforme el desarrollo del producto va avanzando, con tal de conseguir una visión más acertada sobre el usuario final y la problemática real según se van planteando retos

en el proceso. Con ello se consigue alcanzar una creación de producto mucho más apropiada y adaptada al usuario, además proporciona una oportunidad de innovación mucho más adecuada para la situación específica para la que se quiera aplicar el diseño resultante.

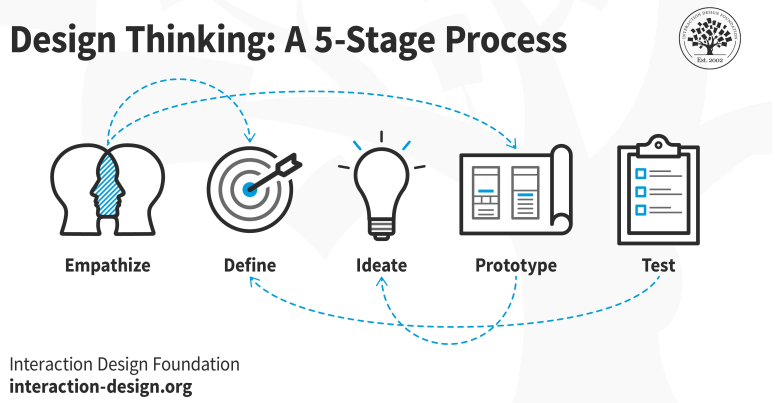


Figura 1.2: Ejemplo de interconexión entre fases [6]

Por otro lado, a lo largo de la evolución de este prototipo, se ha ido aplicando una recurrencia de revisiones y puntos en común entre todos los procesos llevados a cabo de manera simultánea dentro del proyecto. Cada dos semanas se realizaba una recapitulación sobre los avances en el desarrollo del proyecto, estos puntos de recapitulación se verán referidos en adelante como iteraciones, habiendo un total de 7. Si bien es cierto que en alguna ocasión alguna de estas denominadas iteraciones fueron de una duración superior a dos semanas, el tiempo estándar se estableció en dos semanas por cada una de ellas.

Esta estructuración se verá en mayor profundidad en el capitulo 3 de este documento, en el que se explica el desarrollo y las metodologías empleadas en el avance del prototipo. No obstante, a continuación se puede leer un resumen de los objetivos que se fueron estableciendo principalmente para cada una de ellas.

- **Iteración 1**: conocer a los miembros del equipo y delimitar las fortalezas y debilidades de los integrantes. Recopilar y analizar toda la información posible sobre el cliente y sus expectativas respecto a los resultados a obtener en este producto.
- **Iteración 2**: definir un boceto de objetivo final aproximado, lluvia de ideas para posibles creaciones e investigación sobre la viabilidad de las mismas y posibles implementaciones.
- **Iteración 3**: filtrar y seleccionar de manera final de la idea a realizar, configurar el entorno de trabajo de Unity y realizar primeras pruebas sobre el mismo.
- **Iteración 4**: programar un *quiz* y su *layout* funcional, realizar una documentación superficial para mantener un registro de las decisiones tomadas y para traspasar el proyecto para su futura posible mejora de prototipo a producto de cara al público.
- **Iteración 5**: mejorar la interfaz de usuario o UI para la aplicación, cargar y arreglar el modelo 3D de la catedral en su localización final y crear la interactividad con los objetos.

- **Iteración 6**: continuar mejorando la UI y programar correctamente la interactividad de los objetos virtuales.
- **Iteración 7**: recopilar y repasar por última vez los documentos de traspaso, organizar los archivos del repositorio de *github* creado tras problemas de traspaso entre los programadores y organizar una presentación final para todos los *stakeholders* involucrados.

Debido a que la tecnología AR no se había utilizado antes en este ámbito, no había un objetivo final claro al principio. A medida que avanzaba el proyecto, se determinaba un objetivo final en constante contacto y acuerdo con el cliente y este fue la entrega de un prototipo en realidad aumentada que permitiese a los usuarios interactuar a la par que aprender sobre su historia.

De manera simultánea a todos estos objetivos grupales, como diseñadora y desarrolladora paralela, yo, iba realizando una serie de tests y pruebas de desarrollo de implementaciones innovadoras e interesantes para el prototipo final. Además de los objetivos grupales, algunos de mis objetivos fueron los siguientes:

- **Iteración 1**: conocer mi equipo y sus capacidades como desarrolladores y como investigadores, ya que puesto que cada uno cargaba con un trasfondo o *background* diferente era importante conocer cómo se podía trabajar de manera óptima conjuntamente.
- **Iteración 2**: hacia el final de esta iteración se comenzó con la creación de prototipos, en mi caso con la geolocalización de objetos en coordenadas específicas y su representación en AR. Diseñé logotipos para el equipo.
- **Iteración 3**: tras filtrar la idea deseada continué con la programación de objetos geolocalizados y tras comprobar su viabilidad y ponerlo en conjunto con el resto de integrantes del equipo, realicé un traspaso de archivos para su desarrollo en profundidad por parte de mis compañeros mientras yo continuaba investigando y testeando soluciones innovadoras para el producto final. Además, comencé a esbozar un prototipo inicial de interfaz para su implementación correcta en la aplicación.
- **Iteración 4**: realicé documentación de traspaso junto con mis compañeros y comencé a investigar una nueva tecnología AR con planos sobre los que se reproducían vídeos relacionados.
- **Iteración 5**: arreglar los modelos 3D que estaban corruptos, adaptación y corrección de transformaciones espaciales, escalas y texturas. Primeras implementaciones de una nueva API llamada ManoMotion. Diseñé logotipos para el proyecto en específico y no para el equipo esta vez.
- **Iteración 6**: diseñar y definir mejor la UI y acabar las implementaciones de la API anteriormente mencionada. Debido a diversos problemas con la interactividad de los objetos 3D me fue asignado contactar con un experto, pues este era de habla hispana y mis compañeros no lo eran, para solucionar nuestra situación. Otro integrante también contactó con otro programador senior, este sí de habla holandesa. Ambas puestas en contacto fueron claves para el avance del proyecto.
- **Iteración 7**: realizar los documentos visuales finales, recopilar y repasar por última vez los documentos de traspaso y organizar los pósteres a mostrar y los vídeos demostrativos para la presentación final.

A medida que avanzaban las iteraciones, se necesitaba ir recopilando *feedback* de los *stakeholders* y para ello también desarrollé una aplicación en realidad aumentada que era empleada en las ferias de exposición que se realizaban tras finalizar cada una de las iteraciones y que permitían a los clientes integrarse más con el tipo de contenido que se ofrecería al finalizar este proyecto.

En resumen, en el escrito que se va a desarrollar a continuación se va a poder extraer una visión detallada del proceso de creación, diseño e implementación de un producto digital para un cliente real. Simultáneamente este producto ofrecerá diversas aplicaciones de sí mismo en diversos campos como puede ser el campo educativo, entre otros.

Capítulo 2

Herramientas de trabajo

En este capítulo se van a exponer y explicar las herramientas empleadas en el desarrollo de este proyecto.

Debido a que el planteamiento inicial de este proyecto era la creación de una solución que, empleando tecnologías innovadoras, atrajese a los potenciales usuarios de las mismas entre los habitantes de la ciudad, hubieron que investigarse diversas soluciones e implementaciones y alguna se hubo que descartar del producto final.

2.1. Unity

Unity [7] es en un motor de juego en tiempo real para el desarrollo de videojuegos multiplataforma.

2.1.1. Orígenes y evolución de la herramienta

Unity surge en 2005, como respuesta al fracaso del juego de *GooBall* [8], primer juego desarrollado por la propia empresa, *Unity Technologies*, fundada en 2004. Sin embargo, lejos de desmotivarse, los responsables de la empresa se dieron cuenta de las capacidades que el motor que habían diseñado y programado les estaba proporcionando y decidieron desarrollarlo en mayor profundidad.

Si bien es cierto que en sus primeras versiones el motor solo estaba disponible para desarrolladores de Apple, pues fue creado únicamente para Mac, esta herramienta obtuvo el suficiente impulso como para continuar su desarrollo y adaptación a otras plataformas (Windows, Linux, Android…).

Precisamente, consiguió el suficiente impulso como para continuar su desarrollo debido a que los creadores se centraron principalmente en la adaptabilidad multidisciplinar de su motor, esto es, hacer que su entorno fuese fácil e intuitivo de usar y, a su vez, asequible para aquellos desarrolladores independientes que, generalmente, no pueden permitirse la creación o adquisición de las herramientas necesarias para crear este tipo de productos.

2.1.2. Entorno de trabajo

La interfaz de usuario de Unity es el medio que el motor ofrece para la interactividad con él mismo y para la creación de contenido. Nada más arrancar muestra varios elementos importantes:

- **Barra de herramientas** [9]: es la usual barra de menú de todos los programas. Esta se encuentra en la parte superior de la ventana y da acceso a las funciones principales del motor. Desde guardar el proyecto, cargar o exportar a los ajustes de preferencias del proyecto.
- **Jerarquía** [10]: panel a la izquierda del entorno de trabajo que ofrece una vista rápida y general de todos los objetos que se encuentran presentes en la escena actual, entendiéndose escena como el entorno en el que se construye y edita el mundo del juego o aplicación. En este elemento de la interfaz se puede modificar el orden y la jerarquía de los objetos, además de editar las propiedades de los mismos.
- **Área de contenido**: es el espacio principal de trabajo. Permite ver y editar todos los contenidos del proyecto y puede mostrar dos ventanas:
  - **Vista de escena** [11]: es una representación en 3D de la escena actual. Pueden existir varias escenas en un mismo proyecto y dentro de ellas se pueden manipular elementos como las luces, las cámaras y los objetos, entre otros.
  - **Vista de juego** [12]: es una representación en tiempo real de la escena en la que se esté trabajando tal y como se vería en el dispositivo de salida. Esto permite probar el software mientras está en proceso de desarrollo para realizar test y pruebas a su funcionamiento.
- **Inspector** [13]: panel a la derecha de la ventana que refleja información en detalle sobre el objeto seleccionado. También permite editar transformaciones aplicadas al mismo como la escala, la rotación... y otros atributos.
- **Ventana de proyecto** [14]: es un panel situado en la parte inferior de la ventana que muestra todos los archivos y carpetas que comprende el proyecto. Esto permite organizar e importar nuevos elementos de manera rápida.
- **Ventana de consola** [15]: es un panel que se intercambia con la ventana de proyecto, este muestra los errores de compilación, las advertencias de posibles mal funcionamientos y el resto de información mientras se está ejecutando el software. Este panel permite que los desarrolladores puedan detectar los problemas y solucionarlos si fuese necesario.

Además de estas ventanas, Unity ofrece otras que se pueden ir abriendo desde la barra de menú como, por ejemplo, ventanas para la creación de animaciones o para la creación de *shaders*. Y no sólo eso, sino que también ofrece la posibilidad de la creación de aquellas que el desarrollador considere necesarias a través de paquetes ofrecidos en la *Unity Asset Store* (tienda oficial de la empresa de Unity).

Todo ello hace que la interfaz de este motor sea intuitiva y sencilla de usar. Cada elemento realiza una función específica pero además se pueden adaptar a las necesidades de cada desarrollador.

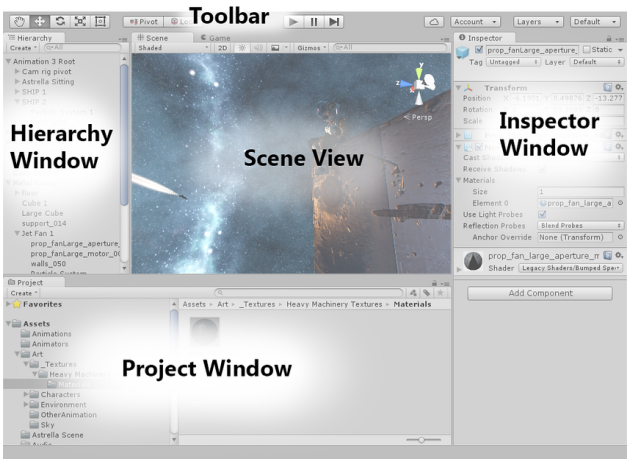


Figura 2.1: Entorno de trabajo de Unity [16]

2.1.3. Características principales

Entre sus características principales destacan:

- **Creación de contenidos para diferentes plataformas**: Unity ofrece la posibilidad de creación de software para diferentes plataformas como:
  - **PC**: Windows, Mac y Linux.
  - **Web**: Unity Web Player y WebGL.
  - **Smartphones**: Android y iOS.
  - **Consolas**: PlayStation 4, PlayStation 5, Xbox One, Xbox Series X/S, Nintendo Switch.
    - **Realidad Extendida (Aumentada y Virtual)**: Oculus, HTC Vive, Google Daydream, Samsung Gear VR, Microsoft HoloLens y Magic Leap.
  - **Smart TV y dispositivos de streaming**: Apple TV, Android TV, Amazon Fire TV y Roku.
  - **Automóviles**: aplicaciones de entretenimiento para sistemas de infoentretenimiento en automóviles.
  - Entre otros [17].

Figura 2.2: Tipos de plataformas admitidas en Unity

Esto permite desde este motor crear una única aplicación y exportarla posteriormente a las diferentes plataformas necesarias.

- **Objetos y entornos**: esta herramienta también permite la creación de *assets* nativos, es decir, aporta la posibilidad de crear objetos/entornos desde su propia área de trabajo.

  Permite la creación de dichos ítems y su perfeccionamiento con *shaders* y nodos. Un *shader* se emplea para especificar y definir cómo se van a visualizar los objetos en 3D aportando un conjunto de instrucciones para controlar como se representan en la pantalla. Existen diferentes tipos de *shaders* según el objetivo que se quiera conseguir. Unity permite editarlos a través de un apartado de su interfaz llamado *Shader Graph*, lo que da la libertad a los desarrolladores de personalizar sus creaciones sin necesidad de escribir código, haciendo el proceso mucho más intuitivo para el groso de los usuarios. Además, dichos *shaders* se pueden compartir y reutilizar, lo que, de nuevo, facilita la creación de las composiciones.

  Además, Unity, gracias a su gran comunidad de usuarios, ha conseguido realizar la integración de otros programas como: *Blender, 3ds Max, Maya, ZBrush, Cinema 4D*, algunos de *Adobe*… para la creación y desarrollo de *assets* (ítems u objetos) en 3D.

  Esto proporciona gran adaptabilidad y libertad a los desarrolladores a la hora de trabajar con sus proyectos ya que pueden emplear diversos ficheros importados desde diferentes programas más adecuados para su concepción y realizar proyectos más completos u obtener resultados mucho más detallados.

Figura 2.3: Ejemplos de programas admitidos en Unity

- **Contenido 2D y 3D**: Unity también posibilita el desarrollo de contenido interactivo no solo en 3D sino también en 2D [18].

  Es mayormente conocido por su vasto potencial en la creación de contenido en 3D, no obstante, es capaz de desarrollar contenido también en 2D. Esto es debido a la amplia variedad de herramientas que también ofrece en este campo.

  Ofrece, por ejemplo, herramientas para la animación y creación de personajes 2D, admite tanto animaciones óseas como *sprites*, que son imágenes 2D. Estos se pueden crear y editar dentro de la propia plataforma Unity, pero también es compatible con otros programas de creación de los mismos, como *Photoshop* o *GIMP*.

  Además, desde Unity se aporta un potente sistema de física 2D, que simula las leyes físicas en un entorno 2D, esto es, los objetos 2D pueden tener características como masa, resistencia y gravedad, lo cual permite que dichos objetos interactúen con su entorno de manera más realista.

  Respecto a la interacción, Unity cuenta con herramientas para la creación de botones, menús y barras de progreso, es decir, elementos correspondientes de las interfaces de usuario en 2D. Pero también permite la detección de eventos táctiles, clics y otras entradas de interacciones.

  En decir, Unity proporciona la viabilidad de desarrollo de contenido interactivo en 2D. Esto

permite a los desarrolladores crear juegos, aplicaciones y experiencias en 2D, y, si quieren, combinarlos con elementos en 3D.

- **Lenguajes de programación admitidos**: esta herramienta es que permite programar desde diversos lenguajes de programación.

  A lo largo de su desarrollo permitía diferentes fuentes de código, pero ha acabado enfocándose principalmente al lenguaje C# para crear el comportamiento y la lógica de la aplicación. Este lenguaje está orientado a objetos y es muy parecido a Java.

  Otros lenguajes de programación que admite son:

  - **C++**: para la creación de *plugins* y código de alto rendimiento.
  - **JavaScript**: empleado de manera similar a C# para la escritura de *scripts*. Este destaca en las aplicaciones web y móviles.

  Por otro lado, excluye muchos otros lenguajes de programación, como **Python** mismo. No obstante, en muchos casos mediante softwares de terceros permite integrarlos. Siguiendo el ejemplo anterior de Python seria mediante el paquete llamado IronPython. Este paquete se ejecuta en .NET, una plataforma de Microsoft, ya que Unity también utiliza esta.

2.1.4. Otras particularidades

Por último, es importante también mencionar que, el motor, ofrece gran potencia en la creación de productos interactivos en 3D además de en otras tecnologías "innovadoras" como la realidad virtual o VR, la realidad aumentada o AR y la realidad mixta o MR (siendo esta última la unión de las dos anteriores).

Mediante el empleo de extensiones y *plugins* de programación permiten al motor trabajar con estas tecnologías de manera más sencilla. Cuenta con integraciones de diversas plataformas tan conocidas como *ARCore* de Google o *ARKit* de Apple, entre otras. Dichas extensiones proporcionan funcionalidades específicas que facilitan a los desarrolladores crear productos con experiencias en realidad aumentada inmersivas. Presentas las herramientas necesarias para el seguimiento del movimiento del entorno y la cámara, la detección de superficies y la colocación de objetos en el mundo real mediante la cámara del dispositivo empleado, sea este un Tablet, un Smartphone u otro dispositivo similar.

En este aspecto, una de las principales capacidades del motor de Unity es la extensión ***ARFoundation*** nativa del mismo y desarrollada por la empresa. Dicha extensión actúa como una capa de abstracción proporcionando una interfaz de usuario unificada y simplificada para el desarrollo de aplicaciones en AR, esto es, al ser este un software nativo de Unity se integra de manera directa en el flujo de trabajo del motor y proporciona componentes, *scripts* y *API*s que sintetizan la elaboración del software, dotando así de eficiencia a las aplicaciones creadas con el mismo pues les aporta la adaptabilidad para las diversas plataformas tan característica de Unity.

Actualmente este programa es muy versátil, gracias a las características comentadas se entiende fácilmente que ofrece una amplia adaptabilidad a las necesidades de los desarrolladores. Esto hizo que la balanza a la hora de seleccionar una herramienta se inclinase hacia esta, no obstante, la decisión final se tomó en base a que, en comparación con otros motores de desarrollo de videojuegos como podría haber sido *Unreal Engine, CRM32Pro SDK* o *GameMaker: Studio 2*, ya se

contaba con experiencia previa desarrollando en esta plataforma ya que, y de nuevo gracias a su asequibilidad y entorno intuitivo, ya había sido empleada en proyectos anteriormente.

2.2. Blender

Blender [19] es un programa de código abierto gratuito que se emplea para el modelado, la animación y el contenido en 3D.

2.2.1. Orígenes y evolución de la herramienta

Esta herramienta surge en 1995, en los Países Bajos. Ton Roosendaal, un programador y diseñador que tuvo la idea de proporcionar una herramienta asequible para los artistas sin tener que invertir grandes cantidades de dinero, tal y como ocurrió con Unity.

En aquel entonces los programas de modelado y animación 3D estaban prácticamente limitados a aquellas empresas que podían costearlos por lo que el groso de los artistas y empresas de menor tamaño no tenían acceso a las herramientas necesarias para la creación de objetos digitales en 3D de alta calidad.

Roosendaal fundó la empresa *Not a Number* (NaN) en el ’98 para comercializar su herramienta y continuar desarrollándola. Sin embargo, en el año 2000, la empresa tuvo que liberar el código fuente debido a dificultades financieras en los que se encontraba la compañía. Esto permitió que los programadores de todo el mundo pudieran desarrollar software que contribuyese a su crecimiento.

Poco a poco la comunidad de Blender fue creciendo lo cual aportó una gran evolución a la herramienta. Cada vez se hacía más intuitiva y fácil de usar, se le añadían más características y todo ello repercutía en una mejoría de los resultantes productos realizados mediante el programa. Actualmente es una herramienta muy potente y, cada vez más, popular en la industria de la producción 3D. Si bien sigue sin ser el estándar de la misma, ya que por ejemplo, en el caso de la animación 3D el programa de Adobe, Maya, sigue siendo el estándar en la gran mayoría de empresas y estudios. No obstante, muchos de estos estudios, empiezan a hacer una transición a esta herramienta, pues reconocen que es lo suficientemente potente y además es de software libre, lo que no solo les ayuda a abaratar costes sino que les permite adaptar mucho mejor el código a sus necesidades.

2.2.2. Entorno de trabajo

Blender posee una interfaz muy personalizable. Dicha interfaz se divide en diversas áreas para diversas herramientas que el programa pone a disposición del usuario.

Las principales partes que componen la interfaz son las tres siguientes:

- **Topbar** *(azul)* [20]: barra situada en la parte superior. De nuevo, como en la gran mayoría de programas, Blender ofrece una barra en la que se encuentran las herramientas básicas de modelado.
- **Áreas** *(verde)* [21]: situadas en el medio. Nada más abrir la aplicación la ventana de Blender está dividida en varios rectángulos llamados Áreas. Las áreas reservan espacio de pantalla para los Editores, ventanas que sirven para mostrar y modificar diferentes aspectos de

los datos. Las áreas pueden personalizarse para las necesidades específicas y se denominan Workspaces a los cuales se les puede asignar un nombre para su posterior reutilización. Entre estas áreas destacan el 3D Viewport, espacio en el cual se pueden modelar, editar y animar los objetos, o el Outliner, espacio que permite organizar la jerarquía del proyecto.

- **Barra de estado** *(rojo)* [22]: en la parte inferior. Este apartado muestra información contextual como atajos de teclado o advertencias.

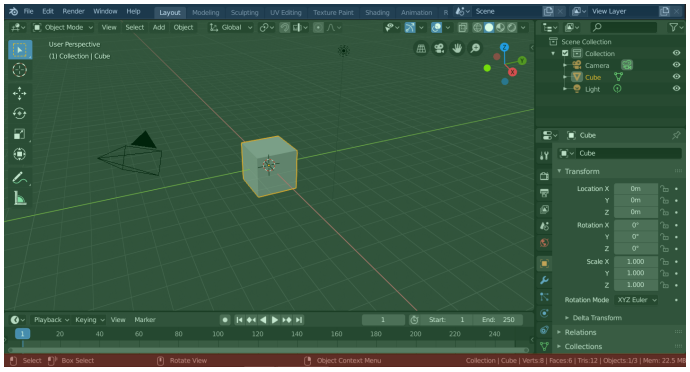


Figura 2.4: Diseño de pantalla predeterminado de Blender [23]

Aparte de las principales áreas que se muestran al ejecutar primeramente la aplicación, Blender también ofrece otras áreas de trabajo como por ejemplo:

- **Línea del tiempo**: sirve para animar objetos y crear animaciones. Muestra los fotogramas clave y permite su edición.
- **Panel de propiedades**: muestra información detallada sobre el objeto que se seleccione, incluyéndose las transformaciones aplicadas al mismo. También permite cambiar la apariencia y comportamiento de los objetos.
- **Editor de nodos**: se emplea para la creación de materiales y texturas. Los nodos son como bloques de información de un concepto específico, como una textura o un efecto de sombreado, por ejemplo. En esta ventana se pueden conectar dichos nodos entre sí para crear texturas y materiales más complejos.
- **Editor de vídeo**: esta área ofrece, tal y como su propio nombre indica, un editor de vídeo multipista. Permite también la corrección de color y audio y su posterior renderización en diversos formatos de archivo.
- **Editor de textos**: sirve para crear y modificar texto, tanto en 3D como en 2D. A estos textos se les pueden añadir también animaciones.
- **Editor de partículas**: se emplea para generar y animar partículas como humo, fuego, polvo o pelo. Permite editar y adaptar la cantidad, las físicas que le afectan y el comportamiento de las partículas. También se puede emplear para controlar la apariencia de las mismas: color, brillo, opacidad...
- Entre otros.

2.2.3. Características principales

- **Modelado 3D**: esta aplicación pone a disposición del usuario un gran abanico de instrumentos de modelado. Desde modelado de polígonos y curvas hasta escultura 3D y animaciones de la malla de los objetos.
- **Animación**: asimismo, Blender está enfocada como una aplicación de animación por lo que aporta una gran variedad de herramientas para la creación y desarrollo de animaciones con cuerpos en 3D. Además cuenta con un motor de físicas que se pueden aplicar directamente a los mismos para simular más verídicamente la realidad. Algunas de las herramientas que proporciona son para la animación de esqueletos, la animación de mallas, la de partículas y la de cámaras.
- **Efectos visuales o VFX**: cuenta también con herramientas para la simulación de humo o fuego, explosiones y otras simulaciones útiles en la generación de efectos visuales.
- **Renderizado**: quizás la parte más importante para la obtención de resultados con calidad del programa, su motor de renderizado. Permite renderizar imágenes en tiempo real y por lotes. Además de un sistema de iluminación lo suficientemente potente como para proporcionar realismo a las escenas creadas mediante su entorno.
- **Composición**: en referencia a los nodos, estas herramientas hacen posible la creación de efectos visuales más complejos. Además también permite componer escenas con varias capas de imágenes y videos de manera simultánea.
- **Edición de vídeo**: como se ha comentado anteriormente, Blender cuenta con un editor de vídeo y sus consiguientes herramientas para ello. Entre ellas destacan la corrección de color, la mezcla de sonido o la creación de animaciones para los títulos.
- **Personalización**: entre todas, esta es sin duda, la mayor ventaja con la que cuenta este programa. Hace posible la completa adaptabilidad para cada artista en cada momento atendiendo a sus necesidades específicas. Pero además, no sólo permite personalizar y adaptar la interfaz y los atajos de teclado sino que también acepta la incorporación de complementos y *scripts* personalizados debido a su código abierto.
- **Comunidad**: debido a la característica anterior, esta se hace importante remarcarla. Ya que la aplicación es de código abierto cuenta con una comunidad de usuarios cuantiosa. Esto hace que muchos de ellos aporten sus creaciones como recursos y tutoriales de como emplear el programa.

2.3. Google Developers

En el siguiente apartado se tratará el entorno de desarrolladores que Google pone a disposición de los usuarios.

Este entorno, también conocido como *Google developers* [24], provee a los programadores de herramientas como la *API* de Google o servicios como *Google Maps* o *Google Cloud* para su combinación en sus proyectos realizados con Unity.

2.3.1. Orígenes y evolución de la herramienta

Google Developers se remonta a 2005, año en el que Google publicó su primer grupo de *APIs*, llamado *Google Maps API*. Este grupo hacía posible integrar mapas en aplicaciones web y móviles, lo cual ocasionó la creación de una gran cantidad de nuevos e ingeniosos softwares en el mercado. Poco a poco Google siguió lanzando publicaciones de *APIs* como la de *Google Calendar*, la de *Google Analytics* o la de *Google Places*.

En 2008, Google realizó su primera conferencia anual para desarrolladores, evento que a día de hoy es de los más importantes en el campo tecnológico. En dicha conferencia presentó tecnologías y productos nuevos que se habían desarrollado y además, a todos los asistentes, les fue proporcionado acceso a 'talleres' y 'entrenamientos', véase como pequeños cursos intensivos, en relación al mismo campo tecnológico.

Poco a poco siguieron lanzando y publicando documentación y tecnología. En 2017 lanzaron *ARCore* ya que un año antes se dieron cuenta en la empresa de Google que la realidad aumentada estaba ganando terreno en la actualidad tecnológica. Si bien ya existía *Daydream*, plataforma de Google de VR, se percataron que no sería suficiente para abarcar la realidad aumentada decidiendo así comenzar el desarrollo de *ARCore*. Día tras día, esta empresa va agrandando y evolucionando sus servicios lo cual repercute en un gran crecimiento de la tecnología y la innovación.

2.3.2. Herramientas principales

Emplear esta plataforma de programación ofrece una amplia gama de instrumentos para ayudar a los desarrolladores a llevar a cabo sus proyectos.

Herramientas como:

- Una **consola de desarrolladores** que mediante una interfaz en la web permite la administración de los proyectos. Desde ella se pueden crear y gestionar *API keys*, esto es, claves de acceso para identificar al desarrollador/es, aplicaciones o usuarios de la *API*. Y también se puede acceder a la documentación del software que Google tiene publicado, entre otras posibilidades.
- **Bibliotecas de código pre-compilado** o SDK (Software Development Kit) que permiten interactuar con la *API* de Google de manera más inmediata y sencilla. *ARCore*, no es exactamente una SDK, ya que es una plataforma completa para el desarrollo. Bien es cierto que *ARCore* proporciona una *API* propia para la integración de servicios de Google como *Google Maps* mas no se considera una biblioteca de código sino, como se ha comentado, una plataforma para el desarrollo de aplicaciones en AR que, además, integra otros servicios de Google.
- **Herramientas de desarrollo**, que son similares a las bibliotecas comentadas en el bullet-point anterior, de hecho, en algunas fuentes son denominadas también como SDK, mas contienen algunas diferencias. Mientras que las bibliotecas de clientes o de código pre-compilado reúnen las herramientas para interactuar con las *API* de manera más rápida, las herramientas de desarrollo son aquellos programas que se emplean para crear, depurar y realizar un mantenimiento de las aplicaciones y servicios creados con las mismas.

- **Eventos y recursos de mejora de las capacidades** de los desarrolladores, esto es, herramientas, sean estas cursos, documentación o eventos como webinars o conferencias, en los que los desarrolladores pueden ampliar sus habilidades y capacidades en relación con las tecnologías que ofrece Google.

Gracias a esta gama de implementaciones el entorno de Google Developers es un gran punto de partida o de apoyo en muchos proyectos de software, ofreciendo a los desarrolladores gran libertad de creación.

2.4. ARCore

Las experiencias en realidad aumentada combinan el mundo real captado por las cámaras a disposición del dispositivo con elementos virtuales, generalmente en 3D, con la intención de crear una experiencia con cierto nivel de inmersión para el usuario. Siendo la finalidad del proyecto la implementación de una tecnología innovadora, pero al alcance del consumidor medio la realidad aumentada brindaba la oportunidad perfecta para ello.

Desde Unity se puede trabajar con diversas herramientas, desde *ARFoundation*, como ya se ha comentado anteriormente, hasta otras como *ARCore, Vuforia, Unity ARKit Plugin* o *Unity MARS*, por ejemplo.

Estas extensiones proporcionan herramientas necesarias para el desarrollo en realidad aumentada, cada una proporciona lo siguiente:

- ***Vuforia*** [25]: seguimiento de objetos, detección de marcadores y reconocimiento de imágenes, entre otros recursos.
- ***ARCore*** [26]: seguimiento de movimiento, detección de superficies y reconocimiento de objetos para dispositivos Android.
- ***Unity ARKit Plugin*** [27]: mismas capacidades que *ARCore*, pero enfocada a dispositivos iOS.
- ***Unity MARS*** [28]: herramientas de creación de escenarios en AR más complejos ya que también permite el uso de la realidad mixta. También permite definir como los objetos virtuales se comportan en el mundo real y aporta adaptabilidad e interacción a las aplicaciones en relación con su entorno.

Para hacer de la experiencia una interesante fueron investigadas diversas vertientes en cómo aplicar esta tecnología y por la que el proyecto acabó decantándose fue por la geolocalización de objetos. Esto hacía que la compatibilidad de una de las extensiones fuera evidente frente a otras, ya que *ARCore* está desarrollada por Google, lo cual permitía la coalición con su *API* y por tanto las coordenadas de *Google Maps*, en comparativa con, por ejemplo, Vuforia, que carecía de dicha funcionalidad. Sin embargo, también se tratará en este documento alguna de las opciones que se decidieron descartar cómo por ejemplo la integración de la *API* de ManoMotion y su interactividad con la realidad aumentada.

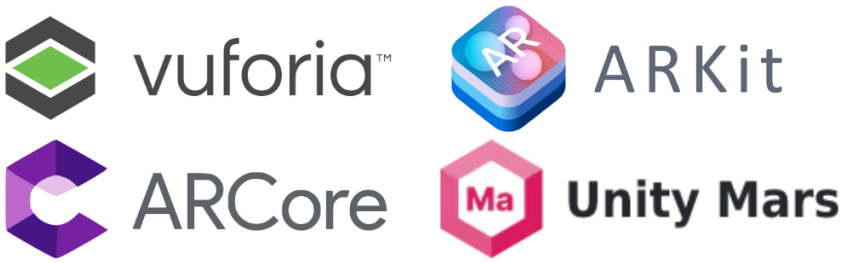


Figura 2.5: Extensiones para realidad aumentada compatibles con Unity

2.4.1. Herramientas principales

Recuperando la mención superficial sobre herramientas que aportan estas extensiones mencionadas, seguidamente serán explicadas con mayor detalle las que *ARCore* ofrece en específico.

- **Seguimiento de movimiento**: detecta la posición y la orientación del dispositivo respecto al entorno, de manera que los objetos 3D pueden interactuar de manera más realista con su alrededor.
- **Detección de superficies**: otorga la capacidad de detección de planos en el contexto que le envuelve, sean estos planos el suelo, paredes o incluso muebles.
- **Reconocimiento de objetos**: puesto que los objetos los recopila como imágenes, *ARCore* posee la destreza de identificar objetos como carteles y emplearlos como anclaje para los ítems 3D colocados en la aplicación.
- **Integración de la luz**: recopila la información del entorno para generar la iluminación digital sobre los objetos virtuales 3D e integrarlos en la experiencia de manera más verosímil.
- **Soporte de dispositivos**: *ARCore*, al estar desarrollado por Google es compatible con dispositivos Android.
- **Realidad virtual compartida**: posibilita la interacción entre varios dispositivos Android en una misma experiencia en AR simultáneamente.
- **Integración con *Google Maps***: al tener la misma empresa de desarrollo *ARCore* cuenta con la incorporación de la información en tiempo real del mundo tangible. Esto consigue que los desarrolladores puedan incluir coordenadas geográficas reales para localizar los objetos virtuales en una ubicación mucho más específica.

2.4.2. ARCore Geospatial API

Esta *API* forma parte de la plataforma de *ARCore*. Funciona como una herramienta para la creación de software con realidad aumentada que integra a su vez información geospacial. Esto lo realiza en dispositivos Android compatibles con *ARCore* empleando tecnología de reconocimiento de imágenes y de geolocalización que le otorgan la capacidad de superponer objetos 3D digitales en el mundo real. Permite también la creación de experiencias en al que los usuarios pueden colocar y visualizar dichos objetos en unas coordenadas específicas.

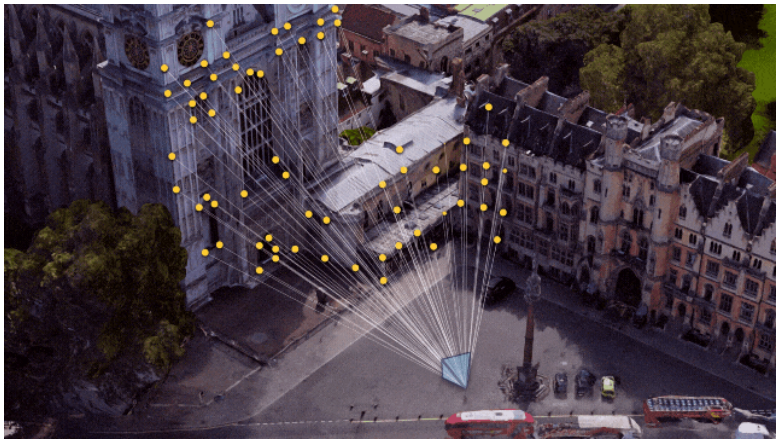


Figura 2.6: Esbozo del funcionamiento de la API [29]

Es necesario un dispositivo Android para el uso de esta API. Funciona mediante **VPS** (*Visual Positioning Service*), esto es, un sistema de posicionamiento no sólo recoge información de la cámara, sino que también la unifica y combina la que reúne con el giroscopio y los sensores de movimiento integrados en los smartphones actuales para determinar su posición exacta y, una vez tiene la suficiente información, procede a orientar de manera correcta el entorno virtual y sus objetos.

2.4.2.1. Características más importantes

Como se ha visto, esta tecnología tiene el potencial para generar softwares muy potentes. Esto es debido a, principalmente, los siguientes aspectos destacados de la API:

- **Compatibilidad con Unity**: evidentemente, y debido a que este proyecto emplea Unity, cabe destacar esta particularidad y es que está diseñada para emplearse con dicho motor de juego y sus desarrollos en AR.
- **Reconocimiento de superficies**: ya que es una herramienta de *ARCore* hace uso de otras herramientas de la plataforma para realizar el reconocimiento de planos como paredes, suelos o muebles para poder colocar los objetos 3D correctamente.
- **Anclajes geolocalizados y seguros**: puesto que emplea geolocalización es capaz de realizar anclajes en coordenadas específicas y colocar objetos 3D en coordenadas y ubicaciones precisas. Pero no solo eso, sino que también permite a los usuarios guardar los anclajes para usos posteriores.
- **Altura adaptable**: desde el punto de vista de los desarolladores, esta característica es realmente útil pues permite ajustar la altura en la que se realizará la representación de los modelos y entornos virtuales respecto al mundo real.
- **Escala del mundo real**: así pues, entendiendo el punto anterior, se comprende que esta característica sea tan destacable, ya que al usar la información obtenida por la cámara reescalará los objetos virtuales al mundo real con tal de dotar de mayor realismo a la experiencia.
- ***ARCore Cloud Anchor***: esta tecnología es realmente innovadora pues permite a los usuarios compartir la experiencia AR en tiempo real.

2.5. ManoMotion

ManoMotion [30] es una compañía que ha desarrollado una *API* de *tracking* de movimiento de manos. Trata de emplear la mínima cantidad de recursos, siendo su principal recurso una simple cámara RGB que ya viene integrada en los smartphones actuales. Esta tecnología rastrea y reconoce los movimientos de las manos de los usuarios y permite al consumidor interactuar con objetos virtuales en tiempo real.

2.5.1. Origen y evolución de la herramienta

ManoMotion nace en 2015, como una StartUp, en Suecia. Dos emprendedores, Zsombor Szabó y Daniel Carlman, se dispusieron a realizar un sistema de *tracking* de manos preciso y accesible.

Dos años más tarde presentó su primer producto en el cual se podía controlar juegos y aplicaciones con diversos gestos de la mano. Esta aplicación fue recibida en el mercado con una gran acogida y les permitió continuar el desarrollo de su sistema.

Un año más tarde, en 2018, lanzaron su primera SDK para Unity que permitia integrar el *hand tracking* o seguimiento de mano en aplicaciones y juegos desarrollados por programadores independientes e incluir su tecnología en sus productos. Esto, evidentemente, fue un enorme paso hacia delante para la empresa lo que repercutió en un crecimiento. Desde entonces, han seguido mejorando su tecnología y innovando en sus soluciones para incorporar más motores de juego y más plataformas de AR, VR y MR.

En 2020, lanzaron una actualización de su SDK con mayor precisión y mayor estabilidad en el seguimiento, además de la adición de herramientas para la personalización y ajuste de su tecnología.

Esta empresa a día de hoy se encuentra asociada con otras importantes empresas del sector de desarrollo de realidad virtual y aumentada, como lo son Oculus, HTC Vive o Microsoft. Gracias a su potencial, la compañía ha recibido varios premios y reconocimientos, por ejemplo, en 2019 ganó el premio a ”la mejor tecnología de interacción.[en] los Premios de Realidad Virtual.

2.5.2. Características y herramientas destacables

Destacan algunas características y herramientas principales:

- **Reconocimiento de gestos específicos**: contiene la capacidad de reconocer una gama de gestos que ya vienen implementados en su código. Esto permite a los desarrolladores asignar X gesto a Y acción o evento específico si se realiza.
- ***Tracking* en tiempo real**: el tiempo de respuesta del código al *tracking* de los gestos de las manos es prácticamente inmediato, por lo que los programadores pueden crear aplicaciones interactivas.
- **Compatibilidad entre plataformas**: se encuentra disponible en diversas plataformas, entre ellas: Unity, Android e iOS.
- **Integración con otras tecnologías**: se puede emplear en diversas tecnologías, siendo estas la realidad virtual, la realidad aumentada y la mixta, lo cual crea la oportunidad de realizar experiencias mucho más inmersivas.

- **Intuitiva**: debido a la gran cantidad de documentación que aporta de manera clara y detallada se hace intuitiva de usar para los programadores que la emplean.
- **Alta adaptabilidad**: su *API* es personalizable y ajustable a las necesidades que el desarrollador requiera para la configuración de seguimiento de la mano. Además, su interfaz de usuario también es personalizable, haciendo posible la adaptación de la interfaz y que esta reaccione según el movimiento de la mano que se realice.

En resumen, se trata de un mecanismo potente para la interacción entre el software y el usuario que lo emplea, además de entretenido e innovador, lo cual aporta un gran atractivo actualmente. Sin embargo, el *tracking* que realiza aún deja bastante margen de mejora.

Capítulo 3

Desarrollo y Metodologías

3.1. Estructuración

La estructuración, como se ha mencionado en el capítulo introductorio, fue planificada en iteraciones de dos semanas (entendiendo esta duración como estándar) por cada una de ellas.

Seguidamente se explicará lo que se realizó en cada una de ellas.

3.1.1. Iteración 1

En esta primera iteración se definió como objetivo principal el de conocer al resto de los integrantes del grupo. Para ello, se realizaron diversos análisis personales y grupales. Estos análisis, generalmente se emplean para la planificación estratégica y la gestión de riesgos empresariales.

3.1.1.1. Análisis SWOT

Se empezó por un análisis SWOT [31], el cual en español es conocido como análisis o matriz DAFO o FODA, que se ve regido según la normativa ISO 9001:2015 elaborada por la Organización Internacional para la Estandarización o ISO por sus siglas en inglés, establecida para la gestión de calidad y la planificación estratégica. En dicho análisis se presentan y se evalúan de manera interna y externa las debilidades, amenazas, fortalezas y oportunidades que el equipo de creación y desarrollo posee y se enfrenta con la finalidad de reforzar, mejorar y minimizar las situaciones que puedan suponer un riesgo para el desarrollo de un producto de calidad.

- **Análisis Interno:**

  Este depende directamente de los integrantes que conformen el equipo o de la empresa misma. Este es un ejercicio de autocrítica que no siempre puede resultar sencillo. En este análisis interno se definen las fortalezas que pueden suponer una ventaja en la creación y producción de valor en el mercado.

  - **Fortalezas (S - *Strengths*)**: para determinar de qué fortalezas se disponen, es importante realizar preguntas concretas sobre como se puede aportar valor a esta elaboración. En este caso las preguntas realizadas fueron algunas como las siguientes:

    - ¿Poseo experiencia previa en el campo de la tecnología y la digitalización? ¿Puede ser útil?
    - ¿En qué destacan mis conocimientos sobre el resto y cómo puedo aplicarlos?
- **Debilidades (W - *Weakness*)**: de igual manera, para detectar debilidades, se han de realizar preguntas de autocrítica y determinar los puntos débiles a los que se puede enfrentar el proceso.
    - ¿En qué áreas se posee menos experiencia?
    - ¿Qué aspectos han influido de manera negativa en producciones pasadas? (en este caso se debía analizar de manera personal, es decir, en experiencias pasadas y no comunes, ya que no se había trabajado en conjunto anteriormente).

En estas preguntas, cada uno de los miembros del equipo fue analizándose personalmente y poniendo en común sus puntos de vista. Tras ello se realiza un filtrado de los puntos comunes y se anota en la matriz los que se consideraron más relevantes.

- **Análisis Externo:**

  Por otro lado, se ha de comprender también de qué forma se puede encajar en el mercado con tal de ser atractivo al ojo del consumidor. Por ello, se han de evaluar los factores que pueden influir de manera directa o indirecta en los resultados que se quieran obtener. Estos factores no dependen de la empresa o equipo, por lo que se han de trabajar y comprender para distinguirse del resto de artículos disponibles.

  - **Oportunidades (O - *Opportunities*)**: para delimitar las oportunidades de las que se disponen, una vez más, se han de plantear ciertas preguntas, pero esta vez de manera conjunta. Algunas de estas preguntas podrían ser:
    - ¿Cómo se pueden aprovechar las nuevas tendencias tecnológicas?
    - ¿Se puede mejorar algún aspecto en nuestro entorno con una visión a largo plazo?
  - **Amenazas (T - *Threats*)**: por último, se han de dejar claras las amenazas que se sufren para poder realizar estrategias de aproximación a ellas. Preguntas que se pueden emplear para su definición pueden ser:
    - ¿Qué factores pueden afectar al óptimo desarrollo del producto? ¿Afecta de manera directa o indirecta?
    - ¿Se pueden solucionar?¿Y se pueden sobrellevar a corto plazo?

  Tras realizar todas estas preguntas el equipo anotó las más relevantes en una matriz, mostrada en la figura 3.1 mostrada seguidamente.

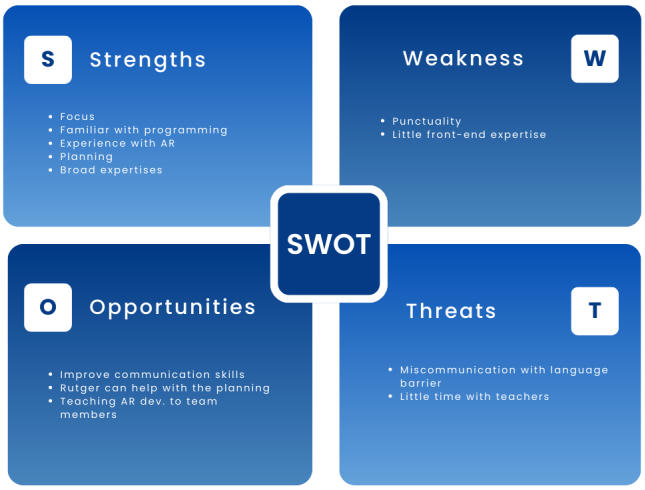


Figura 3.1: Análisis SWOT

3.1.1.2. Análisis de proyecto y cliente

Correlativamente, se realizaron más estudios sobre la situación por la que se demandaba dicha producción, sobre el cliente y sobre sus objetivos a cumplir.

Para ello se emplearon diversas técnicas para aproximarse y definir los límites del proyecto a realizar. Estas fueron la elaboración de diversas tablas o *canvas* (en inglés) llamadas: 5W+1H canvas, 5 times why? canvas, value proposition canvas, entre otras.

- **5W + 1H canvas**: esta metodología consiste en realizar las preguntas:
  - **Quién** - (*Who*): hace referencia a para quién va a ser relevante este producto creado.
  - **Qué** - (*What*): describe las características principales que se quieran emplear.
  - **Cuándo** - (*When*): se refiere al momento en que el producto vaya a verse aplicado.
  - **Dónde** - (*Where*): expone el lugar en el que se quiere emplear.
  - **Por Qué** - (*Why*): especifica por qué motivo o razón se ha precisado el proceso de creación del producto.
  - **Cómo** - (*How*): se presenta cómo puede llevarse a la práctica.

  Haciendo estas preguntas permiten encontrar oportunidades de mejora e identificar soluciones más acertadas.

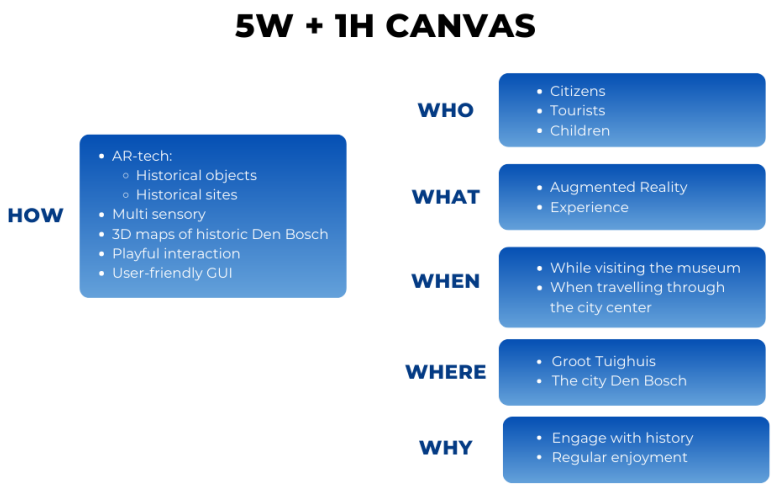


Figura 3.2: 5W + 1H Canvas

- **5 times why? canvas**: esta tabla es complementaria a la inmediata anterior y, así como el análisis anterior, ayuda a comprender mejor la situación que rodea el problema. Esta técnica fue desarrollada por el fundador de Toyota Industries Corporation, Sakichi Toyoda [32], sobre el 1930 para detectar sus fallos y mejorar la calidad de su producción. Hoy en día se puede aplicar también a infinidad de situaciones para lograr vislumbrar la raíz real del problema.

  En este trabajo se comienza con la premisa de que hoy en día hay poco interés histórico por parte de la ciudadanía y se fue extrayendo poco a poco los motivos por los que, desde nuestro punto de vista, esto ocurría. Las conclusiones finales gráficamente pueden observarse en la siguiente figura 3.3.

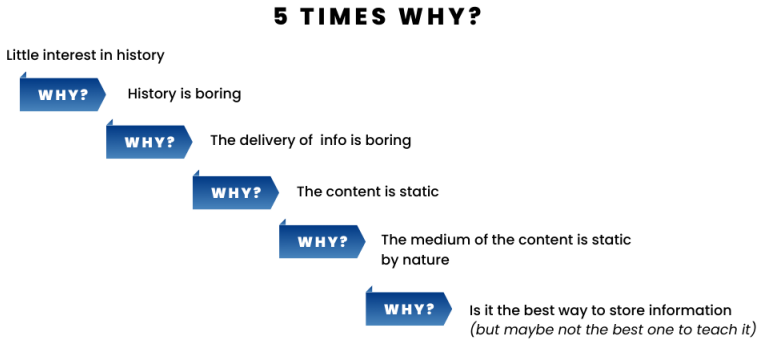


Figura 3.3: 5 Times Why? Canvas

- **Value proposition canvas**: por último, fue realizado una proposición de valor, esto es, de manera visual, presentar qué podrían ganar, perder y qué trabajos habrían de realizar empleando a este producto.

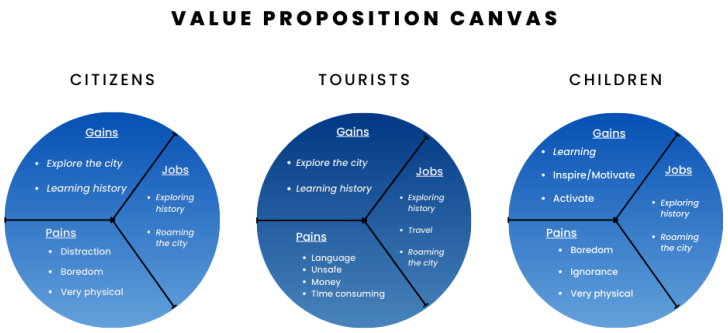


Figura 3.4: Value Proposition Canvas

Tras todos estos pasos, esta investigación se puso a disposición del cliente o *stakeholder* principal. Éste fue comunicando sus impresiones, generalmente positivas, ante nuestra rapidez de respuesta y análisis de la situación y corrigió, acorde a su enfoque, aquellas suposiciones que no debían tomarse en cuenta a la hora de continuar en el proceso de creación de su producto. Algunas de estas fueron, por ejemplo, el enfoque a los turistas que se había tenido en cuenta anteriormente. Si bien se partía de un razonamiento lógico, ya que yo misma era una extranjera y por tanto, podría ser considerada una turista, el cliente corrigió este enfoque y dejó claro que su finalidad principal eran los propios ciudadanos de Den Bosch y quizá posibles escuelas locales, pero estas últimas no eran tan relevantes.

3.1.2. Iteración 2

Para la segunda iteración el objetivo principal fue el de realizar una lluvia de ideas para generar posibles salidas.

3.1.2.1. Estudio de Mercado

En primer lugar se realizó un estudio de mercado y este consistía en una pequeña encuesta con las siguientes preguntas:

1. *¿Vive usted en Den Bosch (o sus alrededores inmediatos)?*
2. *¿Tiene interés por la historia de la ciudad?*
3. *¿Le gustaría aprender sobre la historia de Den Bosch?*
4. *¿Estaría dispuesto/a a ir de manera física a una localización inicial para una experiencia en AR?*
5. *¿Cuánto tiempo y energía estaría dispuesto a invertir en el aprendizaje de historia de la ciudad?*

Para realizar estas entrevistas se procedió a ir a la estación principal de la ciudad, lugar con la mayor afluencia de personas de la ciudad de manera inmediata, y a preguntar a transeúntes, de rangos de edad adulta, que cedían su tiempo a esta investigación.

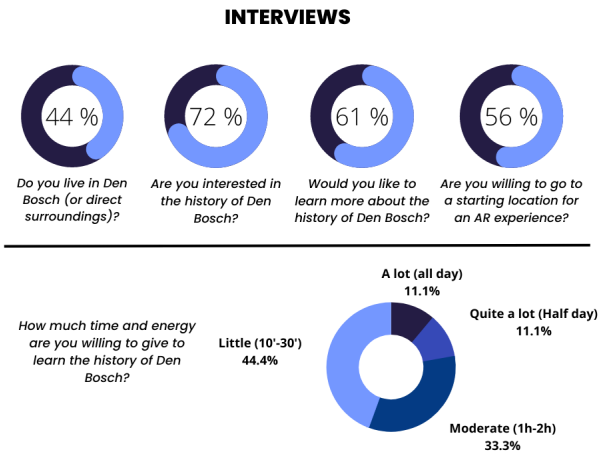


Figura 3.5: Resultados de las entrevistas

3.1.2.2. *Brainstorm* o Lluvia de Ideas

A continuación, teniendo un estudio de mercado más certero, se realizó una lluvia de ideas teniendo en cuenta las tendencias de la gente. Fueron especialmente relevantes las dos últimas preguntas realizadas ya que aportaba información sobre la libertad de creación que se podrían aplicar, esta es, creando una solución más 'estática' o no y cuán intensiva se podría diseñar.

Por una parte se definieron de manera amplia y general ideas clave o conceptos que se debían tener en mente a la hora de posibles implementaciones con **tecnologías innovadoras**, que era el concepto que había sido designado por el cliente. No obstante, el enfoque principal iba a estar en la realidad aumentada ya que estaba disponible para dispositivos como los *smartphones*, dispositivo que hoy en día convive con la gran mayoría de la población.

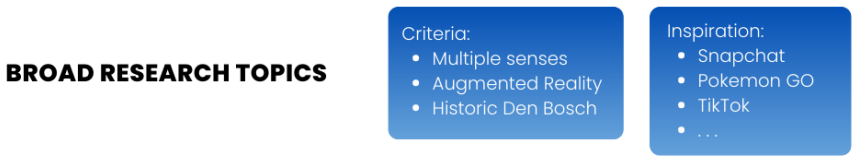


Figura 3.6: Ideas clave

Teniendo en consideración estos criterios previos se realizó la lluvia de ideas que se puede observar en la figura 3.7.

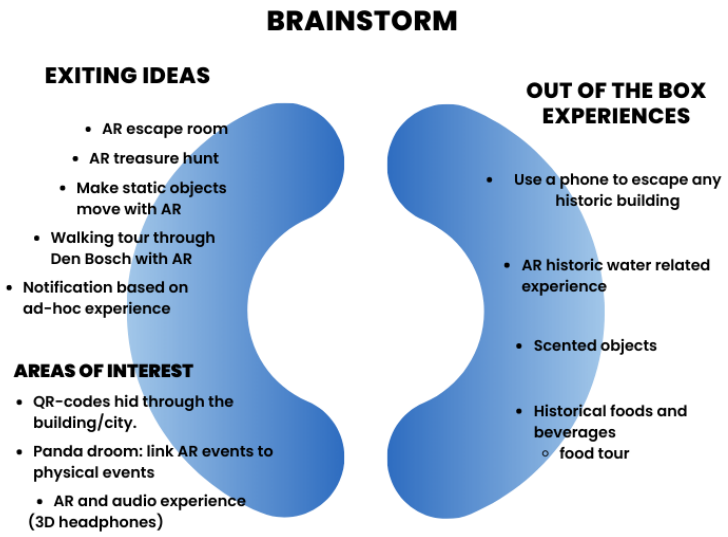


Figura 3.7: Lluvia de ideas

Y se catalogaron los conceptos extraídos en tres grupos:

- **Ideas existentes**:
  1. Escape room en AR.
  2. Caza del tesoro en AR.
  3. Hacer que objetos estáticos cobren vida con AR.
  4. Pasear por Den Bosch con un tour en AR.
  5. Notificaciones basadas en una experiencia ad hoc[1].
- **Experiencias novedosas**:
  1. Escape rooms en edificios históricos utilizando un *Smartphone*.
  2. Experiencia histórica acuática en AR.
  3. Objetos con esencias.
  4. Comidas y bebidas tradicionales: tour de comidas tradicionales.
- **Conceptos de interés**:
  1. Códigos QR escondidos entorno a un edificio o a lo largo de la ciudad.
  2. Panda droom: enlazar eventos AR con eventos físicos.
  3. AR y experiencia auditiva (auriculares 3D - sonido holofónico).

Y para filtrarlas realizamos una tabla COCD para comprobar la viabilidad de las ideas. Esta metodología se emplea para comprobar la viabilidad de las ideas y lo originales o no que son. Es especialmente útil para organizar el potencial de las ideas obtenidas.

El método COCD [34] consiste en generar una matriz 2x2 que catalogue desde lo más normal hasta lo más innovador u original, mientras que por el otro eje catalogue lo viable o no de la creación de los conceptos o ideas.

[1] Locución latina; literalmente ’para esto’. [33]

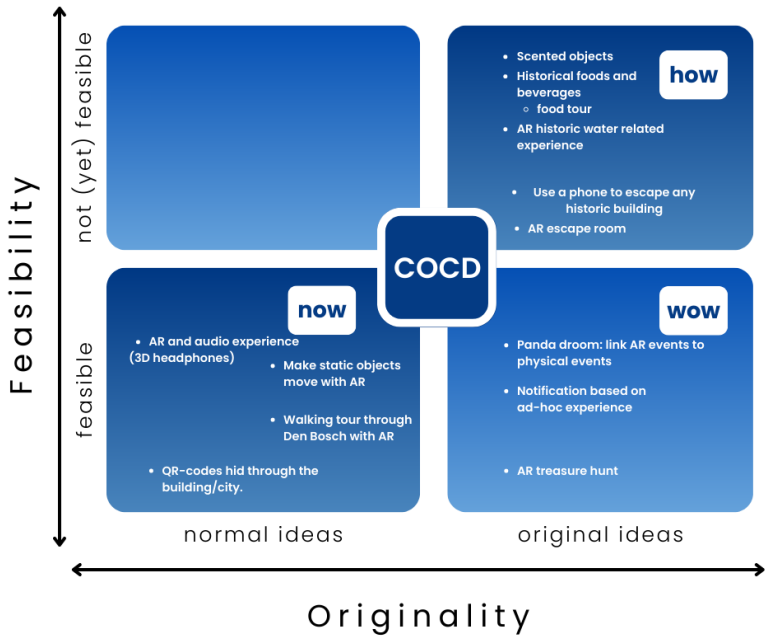


Figura 3.8: Tabla COCD

3.1.2.3. Objetivo final inicial

Pese a la contradicción del título, este corto apartado hace referencia al esbozo principal que se hizo para fijar unas bases para el objetivo final a obtener al final de todo el proceso.

Se marcó que, a rasgos generales, el objetivo del proyecto sería encontrar perspectivas novedosas en cómo la realidad aumentada o AR podía ser empleada para que los ciudadanos, niños y visitantes se interesen más en la historia de la ciudad de Den Bosch.

3.1.3. Iteración 3

En esta tercera iteración el objetivo principal se definió en 2 partes:

1. Seleccionar la idea para el prototipo final.
2. Configurar Unity como entorno de trabajo y realizar primeras pruebas básicas.

3.1.3.1. Idea Final

Como se ha podido comprobar en la COCD box, la idea más original y viable era la del *Treasure hunt*, esto es, una caza del tesoro. Esto fue comunicado al cliente y, siguiendo sus directrices, se redefinió más concretamente la idea. Finalmente, el cliente insistió en realizarlo en una misma localización por lo que esto cerró mucho el margen de originalidad inicial, una gran limitación ya que la idea original era enlazar diversos focos históricos importantes por la ciudad. No obstante, esto, de manera simultánea, facilitó el proceso de desarrollo en cuanto a tiempo se refiere, pues

algunos de los integrantes del equipo no estaban muy familiarizados con el entorno de Unity, VS Code y los objetos 3D.

Al redefinir la idea, el equipo hubo de redactar de manera concreta cómo se iba a realizar. Primeramente, se realizó una investigación de sucesos históricos relevantes que pudieran ser interesantes sobre los cuales desarrollar el producto. Se valoraron 5 situaciones:

1. **La iconoclasia de 1566**: el verano de 1566, Cornelis Walraven llegó, acompañado por transeúntes armados y seguridad, a la ciudad de 's-Hertogenbosch. Poco tiempo después de su llegada, los reformados comenzaron a demandar edificios eclesiásticos para poder realizar sus prácticas religiosas.

   Al ser negados los edificios de la iglesia, debido al intento del ayuntamiento de mantener la paz en la ciudad, la ira comenzó a apoderarse de la gente de la fe reformada.

   Fue entonces cuando comenzó la iconoclasia. El evento de tres días de duración causó la destrucción de altares, estatuas, muebles, libros y túnicas y fue seguido por otra iconoclasia, 2 meses después, que destruyó la capilla que se conservó en su mayor parte en la iconoclasia anterior.

   Solución AR: Para mostrar con precisión las ideas sobre la iconoclasia, la idea es mostrar información de ambos lados. Por ejemplo, podría ser interesante un modelo en 3D de Cornelis Walraven, o alguien de la fe reformada explicando su versión de la historia y su motivación. Por otro lado, podría estar uno de los miembros del consejo de la ciudad, dando más información sobre su versión de la historia, o quizás alguien de la fe católica. Primero se explicaría el desafío con un poco de información de fondo, es decir, la anterior comentada. Tras ello una estatua de la historia principal estaría dividida en varias piezas en la pantalla del usuario. El jugador tendría que encontrar las piezas. Cada pieza se implementaría con una interacción entretenida.

2. **El incendio de la torre de la catedral en 1584**: la Catedral de Sint Jans vivió un gran incendio en la torre, causado por un rayo, en la noche del 25 de Julio, debido a que la torre del medio tenía 85 metros de altura y estaba rematada con una estatua de cobre de Sint Jans. La torre principal se quemó en su totalidad y el techo de la catedral casi se incendió también debido a la lluvia de chispas que cayeron sobre él, la torre oeste se incendió levemente y provocó el colapso de la torre principal. Al final, el resto de ella aterrizó, aún en llamas, en el crucero sur.

   Solución AR: Para este evento, se quería utilizar la tecnología AR para mostrar la catedral tal como era antes del incendio de 1584. Luego se le presentaría al usuario parte de la información (mediante texto, vídeo o animación) e información adicional, después de lo cual serían evaluados para calificar sus conocimientos. Este 'cuestionario' podía tener la forma de un cuestionario real con logros y misiones vinculados a él, o un minijuego en el teléfono del usuario. El objetivo del juego es apagar los incendios resolviendo cuestionarios basados en la información proporcionada previamente.

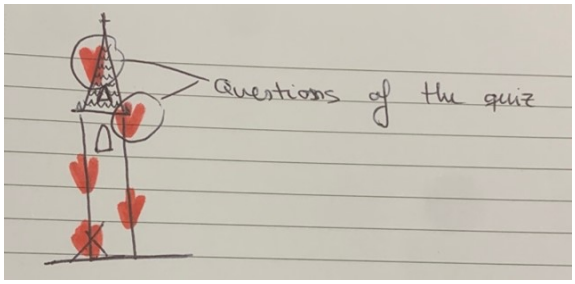


Figura 3.9: Boceto conceptual

3. **La fuente del dragón (*Drakenfontein*) de 1903**: se trata de la fuente más conocida del municipio de 's-Hertogenbosch. Se encuentra frente a la estación de tren, en el cruce '*Stationweg*' y '*Koninginnenlaan*'. Es una obra de arte con un dragón dorado en la parte superior. En 1903, 's-Hertogenbosch celebró un concurso para el diseño de una estatua. El municipio realizó esto en respuesta a una donación de 10.000 florines de Jonkheer Bosch van Drakestein. El motivo de la donación del escudero fue honrar a sus difuntas hijas gemelas que murieron en 1881 a la edad de 17 años.

   El 12 de octubre de 2000 el dragón dorado se cayó de su pedestal a causa del metal oxidado. Esto no es extraño después de 97 años porque el esqueleto del dragón está hecho de metal. Cuando el dragón dorado se estaba cayendo , dañó los 4 dragones de bronce que son la base del pedestal. Después de la restauración, el dragón se volvió a montar el 14 de diciembre de 2001 y se volvió a dorar en oro brillante. En el detalle de la cola del dragón se puso un macetero con un dibujo del dragón, una foto y un trozo de pan de oro para poder reconstruirlo en el futuro.

   Solución AR: se puede mostrar una animación de forma en que el dragón caiga de su pedestal. Después de esto, se mostraría la restauración de los dragones con imágenes detalladas o material de vídeo para que la gente pueda ver el monumento desde una distancia cercana. Debido a que a la ciudad de Den Bosch a veces se le conoce el '*dragón del pantano*', se considera la posibilidad de que la gente pudiese tener interés en esta fuente. Tras mostrar la animación del dragón cayendo del pedestal se le comunicaría al jugador que ha de encontrar la estatua y devolverla a la fuente. Los jugadores pueden encontrar la estatua al rededor de la misma localización y resolviendo algunos acertijos que involucran la historia de la estatua.

4. **El baluarte naranja en 1634**: se trata del último bastión que se añadió a la fortaleza de Den Bosch. Esto ocurrió en el año 1634, tras la expulsión de los españoles en 1629. El baluarte está protegido por gruesos muros y una gruesa capa de tierra en caso de ataque. Hay un gran cañón en el bastión que no funcionaba correctamente. Durante la descarga, las balas salieron del cañón en pedazos y este problema nunca se solucionó. El cañón todavía se puede admirar en el Bastión en estos días. En 1874, el bastión quedó inutilizable para la defensa de la ciudad contra las fuerzas enemigas, pero aún brindaba protección contra el agua.

   Solución AR: mediante AR se puede mostrar cómo se construye el baluarte, cómo funciona y cuál es la función defensiva de la ciudad. Es una defensa estratégica bien pensada. También sufrió varios ataques este edificio, de los cuales todavía se pueden ver rastros en las paredes. Esto se puede mostrar a los visitantes por medio de una simulación. Además, el funcionamiento de un cañón también se puede explicar a los visitantes por medio de AR. Este cañón todavía está en la fortaleza y, por lo tanto, encajaría muy bien con ello. Se puede mostrar

cómo se dispara una bala en un ataque y por qué este cañón no funcionaba correctamente en ese momento.

5. **La cerámica de Robbrecht de Potter de 1437**: esta cerámica se encuentra expuesta en el centro de un centro comercial, valga la redundancia.

   En el año 1437, Robbrecht de Potter fundó una panadería de cerámica. Su lugar de trabajo estaba a las afueras del centro de la ciudad, la cual estaba densamente poblada, para evitar causar olores molestos y potenciales incendios. La cerámica que producía eran cuencos, sartenes, tinajas, platos, lámparas de aceite y similares, que en ocasiones se rompían o se fundían en el horno. Estas fallas fueron arrojadas a un gran pozo, que se encontró varios cientos de años después, antes de la construcción del centro comercial que hoy conocemos como Arena.

   La cerámica fallida no fue lo único que se descartó en un pozo en la Arena. En el año 1456, en el solar contiguo a la alfarería, se construyó el monasterio de Elisabeth Bloemkamp. La basura que producía este monasterio también se desechaba en un pozo. En la excavación de 1994 y 1995 se encontraron varias estatuas y carteles con símbolos religiosos.

   Solución AR: crear una aplicación que 'escanee' la cerámica ubicada en el centro comercial Arena activando varias imágenes AR de cómo se veía la cerámica en el pasado. Además, dependiendo de los modelos 3D disponibles, se mostraría en realidad aumentada cómo era trabajar en una panadería de cerámica en el año 1437, con gente caminando por la zona haciendo sus trabajos.

Estas fueron las ideas que fueron creadas y se concluyó que la relacionada con el incendio en la torre de la catedral era la más interesante para desarrollar.

3.1.3.2. Configuración de Unity y Posibles implementaciones I

Este apartado se incrusta cronológicamente en las iteraciones ya que, a lo largo del proceso de creación, el entorno se tuvo que adaptar a múltiples proyectos y tests y paulatinamente se irá explicando su adaptación de manera más específica.

1. **Configuración de Unity**: tras la instalación de Unity se ha de realizar una serie de comprobaciones y configuraciones con tal de maximizar la compatibilidad de la aplicación a desarrollar con el objetivo deseado, en este caso, la compatibilidad con la realidad aumentada o AR.

   Para los **primeros tests** de este proyecto se decidió emplear la **geolocalización** como punto clave para implementar. Debía descartarse la tecnología si esta no funcionaba correctamente y definir nuevamente la idea pues todavía se estaba a tiempo de hacerlo. Para ello, era necesaria la instalación de la versión de Unity 2020.3.2019f1, esta era la versión LTS, de esta manera la versión de Unity sería estable y no sufriría demasiados cambios en caso de que en el futuro se quisiera trabajar con el proyecto.

   Seguidamente, se realiza la creación de un proyecto y se procede a realizar las configuraciones necesarias en el mismo. Una vez se ha decidido para qué plataforma se va a desarrollar se ha de asentar todo el proyecto sobre ello. En este caso se iba a desarrollar para Android. Esto fue decidido tras realizar una encuesta entre los estudiantes del *minor* y en su mayoría

tenían un dispositivo Android. Además los desarrolladores de este grupo poseían casi todos, a excepción de uno, ordenadores Windows. Esto toma importancia a la hora de compilar la aplicación ya que sin ordenador Apple, no se puede desarrollar proyectos correctamente para dispositivos iOS. Para esto se ha de ir al menú *File* → *Build Settings* → *Android* y pulsar el botón que especifica '*Switch Platform*'.

Siguiendo con la configuración se ha de pulsar *Player Settings*, situado en la esquina izquierda inferior de la misma ventana de plataformas. Una vez se abra la siguiente ventana hay que dirigirse a *Player* → *Other Settings* y eliminar *Vulkan* en el apartado de *Graphics APIs* seleccionándolo y pulsando al símbolo '-' de la esquina inferior derecha del recuadro en el que se encuentra.

A continuación, y en la misma ventana, si se desliza hacia abajo, se ha de seleccionar el nivel mínimo de API admitida. También es importante conocer el dispositivo en el que se va a trabajar, es decir, si se va a trabajar con un dispositivo muy antiguo es recomendable revisar la versión de Android que tiene instalada, de otra manera la aplicación desarrollada no funcionará. Por lo tanto, sabiendo esto, en este proyecto se podía emplear hasta un nivel 30 (Android 11.0), sin embargo, se empleó un nivel 24 sin problemas, correspondiente a un Android 7.0, puesto que si el dispositivo disponía de una versión de Android inferior era mejor situar el nivel en uno inferior, pues no causaba interferencias con el más moderno y permitía el correcto funcionamiento en uno de gama inferior.

Justo debajo de esta opción, se expone la configuración del *Scripting de Backend*, que también tendrá que ser adaptada y pasar de `Mono` a `IL2CPP` y el nivel de compatibilidad (*API Compatibility Level**) a un .NET 4.x. Y por último se ha de habilitar la arquitectura ARM64.

Por otra parte, para emplear la API geospacial y VPS se ha de instalar la extensión *ARCore Extensions*, paquete que, como se ha explicado, amplia las funcionalidades de *ARCore*. Se ha de visitar la página *ARCore Extensions* [35]. Una vez ahí, deslizando hacia abajo se han de seguir los pasos marcados en el segundo recuadro en el que se explica cómo instalar el paquete de extensiones de *ARCore* desde el *Package Manager* del proyecto de Unity. Se añadirá el paquete, para ello se pulsará el '+' de la esquina superior izquierda de la ventana del *Package Manager* y mediante el URL de git proporcionado en la página anterior se instalará una vez se pulse '*Add*'. En la consola puede aparecer un error, no es importante a menos que se desarrolle para iOS, y, tal y como se especifica, no es el caso.

2. **Primer test con geolocalización**: para comenzar es importante incorporar los elementos básicos para realidad aumentada. En primer lugar, se ha de eliminar la cámara que viene por *default* en la jerarquía e incluir *XR* → *AR Session*, *XR* → *AR Session Origin* y *XR* → *ARCore Extensions*. Una vez añadidos a la jerarquía, seleccionando el último elemento mencionado, este es, *ARCore Extensions*, aparecerá en la ventana del inspector un *script* en el que se habrá de referenciar todos los campos como se puede ver en la figura 3.10. Para los dos últimos elementos será necesario desarrollar manualmente unos *scripts* específicos para cada uno de ellos.

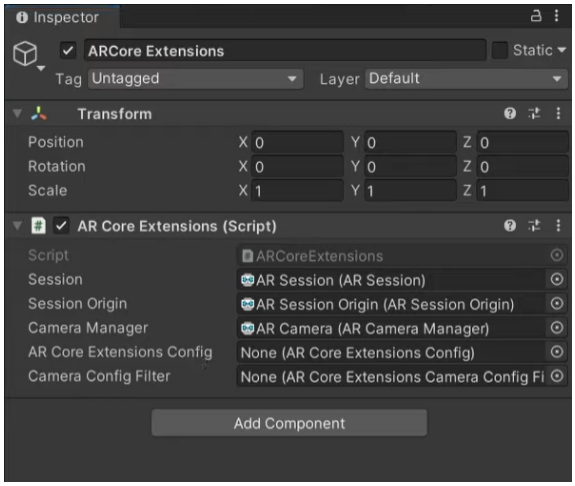


Figura 3.10: *Script* donde se han de referenciar los elementos en el Inspector

Para mantener un espacio de trabajo organizado, lo más óptimo será crear una carpeta dentro de la carpeta de *Assets* que viene por *default* en los proyectos de Unity. Dentro de esta carpeta, que en este proyecto se llamó *Resources* se crean los siguientes *scripts*:

a) *Botón derecho → Create → XR → ARCore Extensions Config*

b) *Botón derecho → Create → XR → Camera Config Filter*

Y se referencian en sus correspondientes espacios.

En el primer *script*, en la ventana de inspector aparecen dos extensiones: *Cloud Anchor* y *Geospatial*. Para activar esta segunda, que es la finalidad de este proyecto, se ha de cambiar la opción que aparece en *Disabled* a *Enabled* y en la consola de desarrolladores de Google. En *File → Build Settings → Player Settings → XR Plug-in Management* se ha de activar ARCore. Una vez activado se ha de cambiar de pestaña a *ARCore Extensions* de la misma ventana, en esta se habrá de cambiar la opción que aparece en *Android Authentication Strategy* de '*Do Not Use*' a '*API Key*', clave que será obtenida de la consola de desarrollador de Google. Y en esta misma ventana se ha de activar *Optional Features → Geospatial*.

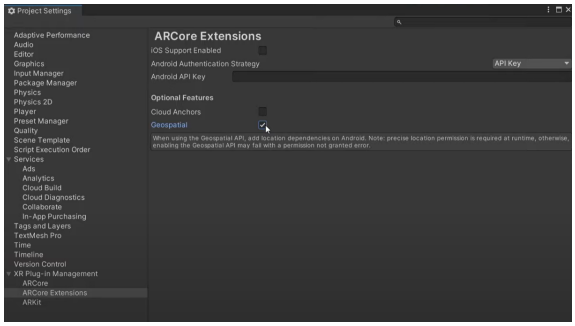


Figura 3.11: Activación de ARCore

En la consola de desarrollador de Google, tras crear un proyecto nuevo se ha de activar la API de *ARCore*. Para ello, se ha de ir a la página *ARCore API* y pulsar en '*Enable*', y esto cargará la página del *dashboard* del proyecto de manera automática. En el apartado de *Credenciales*

se ha de crear las credenciales necesarias para obtener la API Key que se ha de introducir en el proyecto de Unity en la pestaña que se puede ver en la figura 3.11.

Ahora se ha de programar el *script* necesario para la representación en AR de los objetos 3D deseados en cualquier coordenada global. Este *script*, llamado `VPSManager.cs`, se encuentra en el apartado de anexos de la memoria, capítulo 1.

De nuevo en el proyecto, en la ventana de jerarquía se ha de seleccionar *AR Session Origin* con la finalidad de añadir un componente en la ventana del Inspector, componente llamado *AR Anchor Manager*. Seguidamente, se ha de crear un objeto vacío al que se denominará *VPSManager* y al que se le incorporará el *script* recién programado. En este objeto, a su vez, se ha de añadir otro componente más, el *AR Earth Manager*, también en la ventana del Inspector. Este último servirá para referenciarse en el *script* que se ha programado e incorporado al objeto vacío *VPSManager*. Quedará así otro campo a referenciar. Este es el *AR Anchor Manager* y esto se enlaza o referencia arrastrando el *AR Session Origin* de la jerarquía sobre el campo vacío de la ventana del Inspector. Esto habilita de manera definitiva el localizar objetos en realidad aumentada en coordenadas reales del planeta.

Por último, se posicionarían dichos objetos. Para ello, en la lista que aparece en el *script* en la ventana del editor tras referenciar los campos necesarios, se añadirán os objetos deseados y las coordenadas específicas.

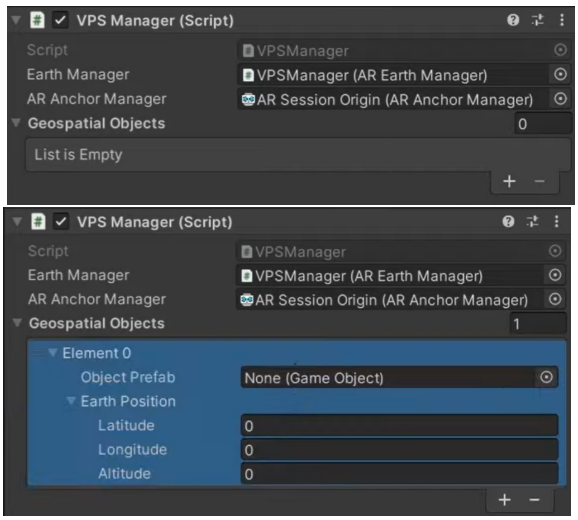


Figura 3.12: Lista de objetos geospaciales

Como puede observarse en la figura 3.12 anterior, para cada objeto digital especificado en el campo del *Prefab* se establecerá unas coordenadas en latitud, longitud y altura. Es importante saber que, para la altura, la API geospacial emplea un sistema de referencia basado en WGS84. Este sistema asume que la tierra tiene la forma de un elipsoide. Esto es relevante pues, al extraer las coordenadas desde el Google Maps para pasarlas al Google Earth para obtener la altura, estos emplean otro sistema de referencia, basado en EGM96, distinto al empleado en ARCore, y entienden la tierra como un geoide. Esto implica que, aunque en longitud y en latitud las coordenadas coincidan, la altura difiere.

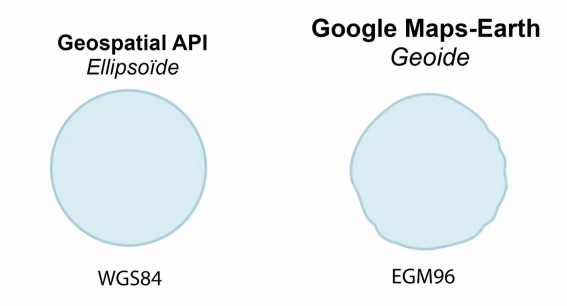


Figura 3.13: Representación visual de ambos sistemas

Para corregirla se puede emplear este conversor online GeoidEval [36] que transfiere las coordenadas de un sistema a otro. Google Earth, en la esquina inferior derecha muestra la altura, en metros, de las coordenadas especificadas.

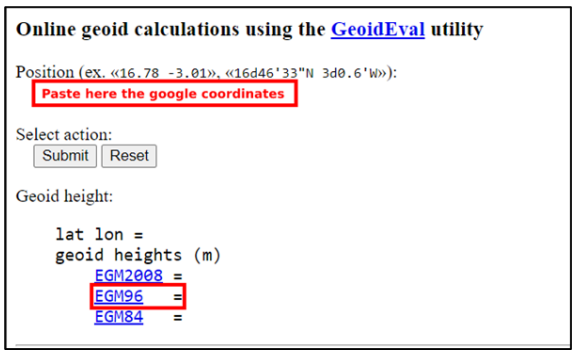


Figura 3.14: Conversor GeoidEval

Para obtener la altura que se ha de introducir en Unity se habrá de realizar la siguiente fórmula en el que se suman los valores obtenidos en Google Earth, en GeoidEval y la altura deseada entre el suelo y el objeto digital en AR (para los tests se empleó una altura de 2 metros aproximados). El valor resultante de dicha ecuación será el que se deba especificar en los metadatos de Unity.

$$AltitudUnity = AltitudGoogleEarth + GeoidEvalEGM96 + AlturaSueloObjeto \tag{3.1}$$

Tras ello sólo se ha de compilar la aplicación. Con el dispositivo que se quiera probar en modo desarrollador (accediendo a esto desde ajustes del dispositivo) y activando el modo depuración se instalará la versión de prueba de la aplicación para poder realizar pruebas. De otro modo se puede compilar el APK e instalar directamente la aplicación final en el dispositivo Android. Sin embargo, esto se trataba de un primer test de tecnología en AR por lo que el modo de depuración era suficiente.

Una vez realizado este primer prototipo, se decidió mostrar los avances en la exposición y el público respondió de manera muy positiva a ellos por lo que, tras realizar una retrospectiva sobre el proyecto, el grupo acordó partir de dicho prototipo como la base del producto final.

3.1.4. Iteración 4

Para la cuarta iteración se establecieron unos objetivos a corto-medio plazo necesarios:

- Comenzar la documentación de traspaso.
- Profundizar en el desarrollo y la programación de la aplicación.
- Buscar posibles implementaciones adicionales.

3.1.4.1. Documentación de traspaso

Esto era necesario en varios aspectos. En primer lugar, y por evidentes motivos, la documentación de traspaso era precisada por el cliente para poder continuar desarrollando y mejorando si así se decidía.

En segundo lugar, no menos relevante, el resto del equipo y el resto de *stakeholders* e interesados en el proyecto, requerían una recopiplación de de las decisiones tomadas y los motivos de las mismas. Si bien esta documentación era una diferente a la de traspaso requerida por el cliente principal, era igualmente indispensable.

Por último, de manera interna empezaba a ser necesario realizar una colección de decisiones y procesos realizados para saber los tests que se habían realizado ya y no caer en la repetición.

Esto hace un conjunto de documentos estructurados que se irían rellenando conforme fuese transcurriendo el proceso del proyecto.

3.1.4.2. Desarrollo en profundidad

El equipo, tras aceptar como base de la aplicación el prototipo geospacial realizado en la iteración anterior, comenzó el desarrollo del resto de los componentes que iban a ser necesarios en la aplicación final, estos son, el resto de escenas y *scripts* que habilitarían los eventos e interacciones que el usuario realizase. Sin embargo, en esta iteración se centró en la escena que iba a poner en funcionamiento el cuestionario o *quiz* de la aplicación.

El funcionamiento del cuestionario era el siguiente: tras pulsar uno de los objetos 3D en AR de focos de fuego situados sobre la torre debía aparecer sobre una ventana *pop-up* una pregunta aleatoria extraída de un archivo `JSON` al cual se accede mediante una url en el *script* GameRequest.cs adicionado en el capítulo 2 de los anexos.

Este archivo debe contener las preguntas a realizar y sus consiguientes respuestas. La respuesta correcta deberá ser especificada en el propio archivo `JSON` para que, tras el procesado mediante el *script* de GameRequest.cs si el usuario ha pulsado la respuesta correcta se sumen puntos al recuento final. Para especificar la respuesta correcta el archivo `JSON` tenía la siguiente estructura *(extracto del archivo final)*:

```
[
    {
        "game": {
            "gameID": "1",
            "questions": [
                {
```

```
                "question": "How many archives does the heritage centre of Den Bosch have?",
                "answers": [
                    "over 900",
                    "over 1000",
                    "over 950",
                    "less than 50"
                ],
                "correctAnswer": 0,
                "timer": 10,
                "description": "Heritage of Den Bosch",
                "link": "https://www.erfgoedshertogenbosch.nl/onderzoek/archiefonderzoek"
            },
            {
                "question": "What is the name of the general who liberated Den Bosch in the WWII?",
                "answers": [
                    "Hendrik Valk",
                    "Robert Valk",
                    "Robert Ross",
                    "Gerald Ross"
                ],
                "correctAnswer": 2,
                "timer": 10,
                "description": "A general liberates Den Bosch in the WWII",
                "link": "https://en.wikipedia.org/wiki/Robert_Knox_Ross"
            }
        ]
    }
}
]
```


Este archivo debía ser accedido desde el *script* y mostrar en la interfaz de la aplicación una ventana *pop-up* que permitiese interactuar con él. Si el usuario, de entre los 4 botones que se mostrarían en el *layout*, pulsaba el que contenía la respuesta correcta, el *script* aumentaría el contador que mostraría los puntos adquiridos, si por otra parte pulsaba cualquier otro de los botones con respuestas incorrectas, el contador no sufriría ninguna modificación y simplemente cambiaría a la siguiente pregunta.

El desarrollo y programación de la interactividad de *script* sufrieron varios problemas de funcionamiento con los objetos 3D y los botones del *layout*, y esto ocasionó que los miembros del equipo decidiesen contactar con expertos, no obstante, esto se dio más adelante en el proyecto y se tratará en la iteración 6 del documento.

3.1.4.3. Posibles implementaciones II: *Tracking de Imágenes*

Este apartado trata sobre más implementaciones que fueron investigadas de manera simultánea al desarrollo de la aplicación principal.

Como se ha ido comentando, a lo largo del desarrollo de este proyecto se investigaban de manera paralela tecnologías novedosas que pudieran ser interesantes para su incorporación en el producto. En esta iteración, además, era necesario realizar un recopilatorio de todos los avances en el proyecto de manera individual. Para ello, desarrollé una aplicación que, mediante reconocimiento de imágenes, identificase una como *target* y mostrase un plano en AR sobre la misma y sobre el cual reprodujese un vídeo.

Ya que es un desarrollo secundario se procede a explicar de manera superficial su funcionamiento, esto es, sin ahondar en el procedimiento del mismo a detalle.

En primer lugar, se ha de instalar todos los plugins necesarios para el desarrollo: *ARCore XR Plugin* y *AR Foundation* y se habilitan en los ajustes del proyecto. Seguidamente, se han de añadir en la jerarquía los elementos para un proyecto AR, tales cómo la *AR Session* o la cámara dentro de la *AR Session Origin*. Dentro de esta última se tendrá que agregar el componente que *AR Tracked Image Manager* que servirá para crear un *gameObject* cada vez que se detecte una imagen en el entorno.

A continuación, se ha de crear la librería de imágenes que van a ser reconocidas. En esa librería se añadirán todas las imágenes que quieran especificarse como *target* para ser identificadas posteriormente. Lo más relevante de estas imágenes es que han de tener muchas zonas de contraste para poder crear *keypoints* los cuales permiten detectar y rastrear la imagen.

Por último, se ha de crear un plano en la escena al que se le ha de incorporar el vídeo que se quiera reproducir sobre el mismo. Al hacerlo, se habilita un componente en la ventana del inspector llamado *Video Player*. A dicho plano se le añade también un material con la imagen ya que tarda unos segundos que el vídeo comience a reproducirse, por lo que es más estético si el plano muestra una imagen de base en vez de un plano en 3D vacío. Solo queda hacer del plano un objeto *prefab* en el proyecto de Unity y referenciarlo en la lista.

Este proyecto fue empleado en las posteriores exposiciones para la recolección de *feedback* para el proyecto, de manera que acercaba la tecnología AR a los potenciales usuarios y les ayudaba a conectar mejor con la visión del proyecto.

3.1.5. Iteración 5

En esta quinta iteración se realizaron cuatro acciones principales:

1. Mejoras en la interfaz de la aplicación y del cuestionario.
2. Corrección de la carga y texturizado de los objetos 3D en Blender. Creación del objeto 3D de fuego.
3. Programación de la interactividad. *(Este apartado se verá con mayor profundidad en la iteración siguiente ya que fue en la que se consiguió poner en funcionamiento).*
4. Investigar y realizar tests con ManoMotion como posible nueva implementación.

3.1.5.1. Interfaz

Este subapartado trata sobre el diseño de la interfaz de la aplicación. Se llevó a cabo a lo largo de esta 5ª iteración y de la 6ª, mas se va a explicar en esta por mantener una coherencia.

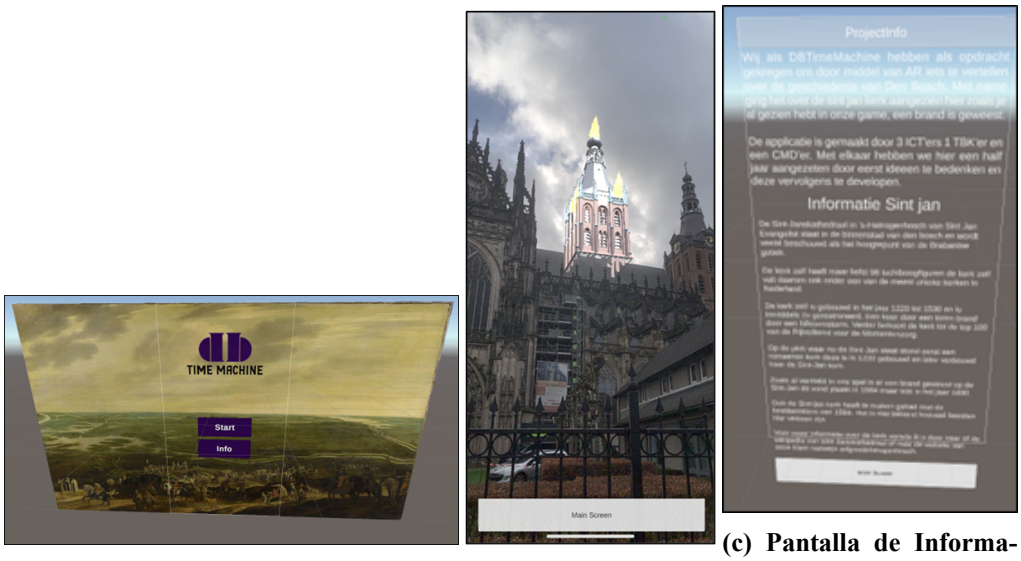
Para la estructura de la aplicación se establecieron varias escenas. Las escenas realizadas fueron las siguientes:

- ***MainScreen***: esta escena es la primera pantalla a la cual los usuarios accederían. Su principal y única función es la de dar la bienvenida al usuario a la aplicación.
- ***ArScreen***: esta escena, por otro lado, es la que contiene la '*ARSession*', es decir, la torre en AR en las coordenadas marcadas y la interactividad mediante los focos de fuego y el cuestionario.

Por un lado, la *MainScreen* aportaba un elemento estético mientras cargaba la aplicación, con una imagen de la ciudad de Den Bosch de fondo y una estructura intuitiva en cuanto a su funcionamiento, mientras que por otro lado, también servía como un elemento práctico, ya que esta pantalla de

inicio servía para dar paso al usuario a la escena del juego o a una pantalla con información sobre la el momento histórico sobre el que estaba desarrollado esta aplicación.

Por último, en la *ArScreen* se añadió finalmente un botón para poder volver a la pantalla inicial si se deseaba, igual que en la pantalla de información.



(a) ***MainScreen*** **(b)** ***ArScreen*** **(c) Pantalla de Información**

Figura 3.15: Imágenes extraídas del prototipo en pruebas

3.1.5.2. Objetos 3D: Corrección y creación

Con los objetos 3D el proyecto se topó con varios obstáculos. Como pone en el título de este apartado estos obstáculos se dividen principalmente en dos bloques: creación de objetos y corrección de los existentes.

- **Creación**: para la creación de los focos de fuego situados sobre la torre en AR ocurrieron varios sucesos. En primer lugar, se estimó que, debido a que el objeto iba a aparecer repetidas veces, era conveniente que este objeto no tuviese mucho peso a la hora de exportarse. Además, ya que iba a estar enlazada una animación al mismo, esta decisión mejoraba el rendimiento de la animación también. Para ello, se realizaron varios focos de fuego *lowpoly*, esto es, de bajo poligonaje, objetos con pocas caras y pocos vértices.

  Los primeros modelos realizados se desarrollaron en Blender, junto a su animación y texturizado. Esto ocasionó problemas de compatibilidad entre Blender y Unity, ya que pese a sus grandes compatibilidades, en algunas ocasiones ocurre que los modelos realizados cargan con texturas y *shaders* incompatibles con el procesado en Unity. Algunas fuentes recomendaban realizar los *shaders* directamente en Unity y aplicarlos en el modelo y otras realizar el modelo directamente en Unity. Primero se intentó crear los *shaders* con emisión de luz que eran los que más causaban incompatibilidad, sin embargo, la animación del objeto no se consiguió corregir, por lo que finalmente se decidió crear un foco de fuego desde el programa de Unity, a detrimento de la calidad resultante del mismo. Teniendo en cuenta el tiempo que

se consumía en la creación y corrección de cada objeto, se consideró prioritaria la creación de uno que fuese funcional a uno más estéticamente atractivo.

A través de las capacidades que ofrecía Unity, se consiguió modelar un objeto *lowpoly* animado de un foco de fuego.

Figura 3.16: Ejemplo del objeto del foco de fuego final

- **Corrección**: el cliente entregó modelos 3D que poseía el archivo de la ciudad. Desgraciadamente, los modelos proporcionados se encontraban corruptos. Algunos perdían las texturas y *shaders* y/o aparecían en blanco mientras que otros ralentizaban el programa debido al tamaño de los archivos. En la figura 4.1 del documento se puede ver un ejemplo de estos errores.

  Se consideró el empleo de otros modelos similares y darle la oportunidad posteriormente al cliente de reemplazar los modelos empleados por los modelos reales, sin embargo, se consiguió resolver, mediante el programa de Blender, las transformaciones incorrectas y los enlaces de los *shaders* de los modelos que causaban incompatibilidades y al hacerlo se descubrió que estos no eran incompatibles con los procesados de Unity.

  Estos objetos no eran *lowpoly* como los focos de fuego, por lo que tardaban algo más en cargar, sin embargo, al no tener animaciones enlazadas y poseer unas texturas bastante simples conseguían compensar su gran tamaño de poligonaje.

3.1.5.3. Posibles implementaciones III: *ManoMotion*

Este es el último apartado sobre posibles implementaciones adicionales que eran interesantes para el prototipo a entregar. En esta ocasión se consideró interesante la tecnología de seguimiento de manos o *Hand Tracking* y de control mediante gestos o *gesture control* que proporcionaba ManoMotion.

En el capítulo 2 apartado 5, se explica cómo ManoMotion mediante una cámara RGB rastrea y reconoce los movimientos de las manos del usuario y le permite interactuar con los modelos virtuales en tiempo real. Esto resultó atractivo, desde el punto de vista del consumidor, el poder interactuar de forma más similar a la vida real con objetos y modelos en AR con las manos.

Para poder trabajar con esta API el desarrollador se ha de registrar en su página web [30], lo cual le dará acceso a las claves y el SDK necesarios. Según lo que se desee trabajar se adquirirá una u otra licencia. Para esta implementación se necesitaba la segunda, *SDK Pro (Non-Commercial)*, esta es más completa que la básica y también es gratuita. Además, la característica principal por la

que se ha de trabajar con esta es el método de rastreo ya que la licencia básica tiene un método de rastreo por medio de una caja, mientras que la segunda licencia, la adquirida, se hace por medio de un esqueleto aplicado a la mano. Se descargará el paquete con la versión Pro y con ARFoundation.

Para el proyecto de Unity se ha de tener en cuenta que se ha de seleccionar la plantilla URP ya que ManoMotion emplea sus propios *shaders* para representar el esqueleto de la mano.

Continuando con la configuración del proyecto, una vez más se han de aplicar los paquetes de ARCore (para Android) o ARKit (para iOS) y ARFoundation. Igual que en otros proyectos, seguidamente se selecciona y se hace el traspaso a la plataforma para la que se va a desarrollar. A continuación, en *Player Settings* se habrá de eliminar *Vulkan* para que no cause incompatibilidades con los plugins de ARCore. En la misma pestaña, más abajo, en *Identification* se ha de habilitar la pestaña de *Override Default Package Name* para que el SDK de ManoMotion funcione. En la página de ManoMotion, en la pestaña de licencias, se podrá adquirir el nombre del paquete necesario para reescribir el que viene por defecto en Unity. En la configuración del *scripting backend* se cambia de Mono a IL2CPP y su nivel de compatibilidad a .NET 4x. Además, se ha de habilitar el recuadro de ARM64 y en la pestaña de *XR Plug-in Management* se habilita el recuadro de ARCore.

Para la configuración de Unity para poder importar adecuadamente ManoMotion, únicamente queda por especificar el renderizado. En la carpeta de *Settings*, se selecciona el archivo llamado *ForwardRenderer* y en la pestaña del inspector se ha de pulsar el botón que dice *Add Renderer Feature* → *AR Background Renderer Feature*. Tras esto ya se puede importar el paquete del SDK anteriormente descargado.

Para poder emplear AR se habrá de eliminar la cámara que va por defecto en la jerarquía y agregar *AR Session* y *AR Session Origin*, que contiene la nueva cámara. Además, habrá que agregar los componentes *ARManomotionManager* y *ManomotionCanvas*. Dentro de *ARManomotionManager* habrá que referenciar todos los campos vacíos que contiene y que hacen referencia a componentes de el prefab *ManomotionCanvas* tal y como se puede observar en la figura 3.17.

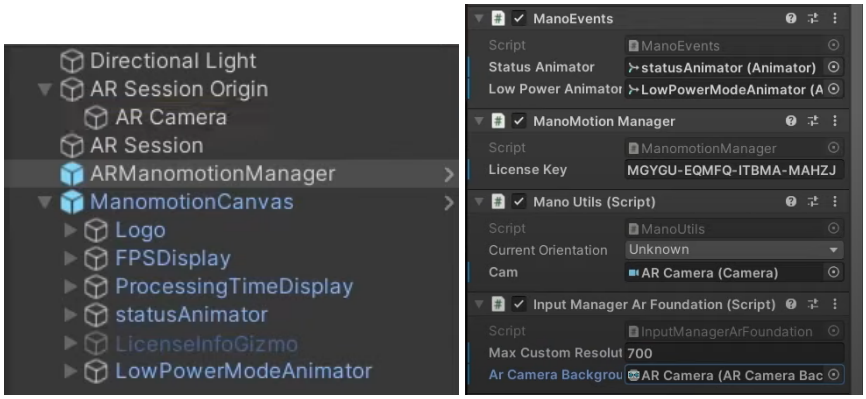


Figura 3.17: Especificaciones en *ARManomotionManager*

Una vez realizada toda la configuración base del proyecto se comienza a dar cuerpo a lo que realmente va a consistir en sí. En este caso, se hará que una esfera, que rastrea y sigue la mano del usuario, colisione con los objetos virtuales 3D de su entorno y sea capaz de, detectando cierto gesto (el de agarre), mover el modelo 3D a voluntad. Para ello, en la ventana de jerarquía se añade un objeto esfera, a la que se le denominará *HandPointer*, y se le cambia el tag a *Player* y se escribirán los *scripts* que permitirán la interacción entre la esfera y los modelos 3D. A esta esfera también se

le han de eliminar los componentes *Mesh Filter* y *Mesh Renderer* para que así la esfera no se vea en el prototipo final. Es recomendable, sin embargo, que para los tests de funcionamiento estos componentes sigan presentes para comprobar el funcionamiento correcto del proyecto.

El primer archivo se llamará *GrabInteraction.cs* y el segundo *ObjectManipulator.cs*. En el primer *script* mencionado se programará el funcionamiento del gesto de agarre. Este *script* será asignado a un nuevo objeto vacío creado en la jerarquía y tras asignarlo se referenciará a la esfera en él.

En el segundo, se definirán los comportamientos de los objetos al colisionar con la esfera y detectar el gesto de agarre. Se le añade un *BoxCollider* y un *RigidBody* y si estos hacen contacto con el *HandPointer* cambiarán de color, para que el usuario sepa que están siendo detectados. Si el usuario realiza el gesto de agarre el objeto detectado pasará a ser hijo de la esfera, de otra manera, si el usuario suelta el objeto, desde el *script GrabInteraction.cs* se eliminarán todos los hijos que se le hayan adherido.

Tras exportar la escena y la aplicación se puso en periodo de prueba. Tras múltiples tests se detectó que este proyecto, pese a su componente atractivo en cuanto al tipo de interactividad, reunía más fallos de funcionamiento que de provecho para el usuario final. La cámara no rastreaba bien el movimiento ni el gesto realizado y perdía constantemente el agarre sobre los objetos virtuales, por lo que se acabó decidiendo que no se implementaría en el prototipo final.

3.1.6. Iteración 6

Para esta sexta iteración el equipo se centró principalmente en mejorar la interfaz ya expuesta en la iteración anterior, por lo cual no se va a ahondar en ello, y en la corrección de la interactividad.

3.1.6.1. Interactividad

Para la interactividad se afrontaron diversos desafíos. El proyecto disponía de interactividad en dos puntos importantes:

- **Botones**: los botones localizados en la *MainScreen*, los botones de las respuestas del cuestionario y los botones de retroceso a la pantalla de inicio.

  Entre estos los que mayor dificultad supusieron fueron los del cuestionario, ya que debían extraer la información del archivo `JSON` y reconocer la que se pulsaba para aumentar el contador de puntaje en caso de que la respuesta fuese la correcta. Para ello se desarrolló la función de `CheckAnswer` del *script* GameRequest.cs que, como ya se ha mencionado, se encuentra dispuesto en los anexos.

  Esta función reconoce el nombre de cada botón pulsado y comprueba si el nombre coincide con el que alberga la respuesta correcta, especificada también en el archivo `JSON`. Tras reconocer la variable, esta entra en una estructura condicional `if` y si es positiva hace aumentar el contador, sino el cuestionario simplemente pasa a la siguiente pregunta.

- **Fuegos AR**: estos objetos tenían que ser interactivos ya que eran los que permitían al usuario acceder al cuestionario. Estos debían actuar como botones de enlace entre la escena con la torre en AR en las coordenadas especificadas y el *pop-up* del cuestionario.

En este tipo de 'botones' se afrontaron más dificultades, ya que como tal en Unity no se estaban procesando como tales sino como simples *prefabs* u modelos 3D. Para averiguar esto, los desarrolladores del equipo investigaron múltiples técnicas. Viendo la dificultad y los consecutivos resultados infructuosos se consideró que lo más efectivo sería contactar con desarrolladores expertos. Se contactó con un desarrollador de habla hispana y con otro desarrollador de habla neerlandesa. Gracias a ellos se detectó que Unity estaba detectando los modelos como simples objetos y explicó como podría solucionarse mediante *RayTracing*.

El *RayTracing* o trazado de rayos, se emplea en esta aplicación para detectar la colisión de una pulsación con el espacio en el que coincide un objeto virtual 3D. Esto funciona mediante un algoritmo que calcula de manera aproximada el camino de la luz como píxeles en un plano y simula sus efectos sobre los modelos virtuales. Si esta pulsación se producía sobre uno de los modelos 3D el *script* cargaría la pantalla del cuestionario con la primera pregunta a responder sobre la escena *ArScreen*.

Tras conseguir resolver la programación correcta para lograr contar con una aplicación plenamente funcional el prototipo se dio por finalizado. La aplicación funcionaba correctamente y se habían alcanzado todos los objetivos establecidos.

3.1.7. Iteración 7

Esta fue la iteración final por lo que se debía recopilar toda la información y documentación realizada, repasarla, corregirla y dejarla lista para la entrega. Además, también era necesario organizar una exposición final, más compleja que las anteriores y a la cual se debía invitar a los clientes de los proyectos para ver el resultado final.

Por lo que se refiere a la documentación de traspaso se decidió que, empleando los documentos que ya se habían realizado, se reescribiría y entregaría finalmente en limpio los 3 documentos más relevantes:

1. ***Project Progress***: en este se explicaba a grandes rasgos las decisiones tomadas respecto a las implementaciones a lo largo del proceso de creación y desarrollo del prototipo. Decisiones como la no implementación de la tecnología de *Hand Tracking* debido a su malfuncionamiento o el descarte de otras como una base de datos en la que los usuarios se registrasen para comparar sus puntuaciones con otros jugadores debido a la decisión de no aplicarlo del *stakeholder* o cliente principal.
2. ***Setup development environment***: en este archivo se explicaba qué pasos había de seguirse para instalar y configurar de manera correcta el entorno de Unity para poder trabajar el proyecto adecuadamente. Así como cómo descargar de GitHub el proyecto base para poder trabajar sobre ello y no empezar de cero.
3. ***Development documentation***: finalmente, en este documento se explicaba cómo se había desarrollado la aplicación, los pasos que se habían llevado a cabo para la programación y explicaciones pertenecientes a la misma, tales como por ejemplo el funcionamiento de los sistemas WGS84 de Google Earth y EGM96 de ARCore, entre otras.

En otro orden de cosas, adicionalmente en esta iteración se había de estructurar y organizar una presentación final enfocada a los clientes. Para ello, se tenia que organizar un estand similar al

de las iteraciones pasadas, sin embargo, esta vez se había de reflejar el progreso completo y los resultados finales obtenidos en el proyecto.

Se organizó en tres partes:

- **Línea del tiempo**: se resumía a grandes rasgos las fases del método de *design thinking* que se habían ido completando. También contaba con carteles en los que se exponían momentos y decisiones destacables del proceso.
- **Características principales**: se mostraban en un grupo de carteles las características más importantes con las que contaba el prototipo final desarrollado y se leía una breve explicación sobre cada una de ellas, como por ejemplo, la geolocalización de objetos o los objetos pulsables y sus animaciones.
- **Prototipo final y exposición de pruebas**: este apartado se consideró interesante mostrar todos los prototipos realizados además del entregable final que destacaba sobre el resto. Esto fue así debido a que se quería reflejar, mediante vídeos y pósters, la complejidad del proyecto y todo el trabajo realizado a lo largo de las iteraciones.

Además, también se ponía a disposición del público el prototipo final, con las coordenadas cambiadas al lugar de la exposición, para que se pudiera reflejar su puesta en funcionamiento en tiempo real y hacer más comprensibles los resultados obtenidos.

A modo de resumen de este proyecto se puede visitar esta presentación [37] que se entregó como portafolio personal de recopilación de trabajo y decisiones realizadas.

Capítulo 4

Resultados, conclusiones y perspectivas futuras

En este capítulo final se va a comentar cuáles fueron los resultados obtenidos con el producto final y, a su vez, se van a extraer conclusiones sobre los mismos.

Por otra parte, como el propio título de capítulo indica, también se van a exponer las perspectivas futuras para este proyecto y las posibles aplicaciones que puedan atribuirse al mismo y a la experiencia que se ha extraído de su desarrollo.

4.1. Resultados

Tal y como se ha comentado a lo largo de las iteraciones del capítulo anterior, han habido resultados tanto positivos como negativos en el desarrollo de este prototipo. A continuación, se van a mostrar los más relevantes en cada caso y se van a elaborar conclusiones entorno a su impacto en el prototipo entregado.

4.1.1. Resultados negativos

Seguidamente se muestran y se explican los resultados negativos extraídos:

- **Primeras metodologías de recolección de feedback**: este aspecto es importante destacarlo pues, sin un buen feedback, esto es, sin un buen método para recolectar opiniones de potenciales usuarios, no se puede tener en cuenta si el producto que se está diseñando va a ser útil o empleado a futuro.

  Este resultado se considera negativo debido a que alguna de las metodologías fueron:

  - Preguntar de manera directa (sin el beneficio del anonimato) a los usuarios sus opiniones, y no apuntar la pregunta realizada ni recolectar las respuestas por escrito.
  - Selección de la idea mediante un juego con pegatinas colocadas en los pósteres del estand de exposición. Este, aunque en inicio fue novedoso y entretenido, pronto mostró

sus fallos: dar la opción a selección múltiple era contraproducente, pues lo que se pretendía era filtrar una única opción; y, al hacerlo de manera visual, el usuario promedio elegía su opción en base a lo que la mayoría anterior ya había elegido, por lo que no era una decisión real sino en busca de la aceptación del resto y por tanto, no era válida.

- **API de ManoMotion no funcional para el usuario**: esta tecnología no pudo finalmente ponerse en práctica debido a la cantidad de errores que su ejecución producía. El resultado fue negativo debido a que, como se ha comentado en las iteraciones 4 y 5, la cámara no ejercía un buen seguimiento o *tracking* de la mano del usuario y por tanto las acciones realizadas, como el agarre de objetos en AR no era efectivo.

  Además, también se consideraron otros factores como la necesidad de un tipo específico de configuración del proyecto de Unity, incompatible con la geolocalización de los objetos con ARCore, o el desconocimiento, por parte de los usuarios, del funcionamiento de la tecnología. Ocurrió que, mediante el testeo de esta implementación, el equipo se percató de que los usuarios tendían a apretar mucho las manos para intentar que la aplicación captase mejor su gesto y acababan con las manos doloridas e incluso con hendiduras auto-infligidas con sus uñas en sus palmas. Si en principio este último factor comentado puede parecer una tontería, a largo plazo es un inconveniente puesto que los usuarios no querrían usar la aplicación si con ella se producen dolor, y además no recomendarían el uso de la aplicación, por lo que se perdería gran cantidad de público potencial.

  Debido a estos motivos esta tecnología no se implementó finalmente en el prototipo resultante.

- **Modelos 3D incompatibles**: esto fue un gran inconveniente. El cliente nos proporcionó modelos 3D que poseía el archivo de la ciudad *Erfgoed 's-Hertogenbosch* [38]. No obstante, los modelos proporcionados no eran compatibles con los *softwares* empleados en el proyecto. Esto ocasionó un retraso importante en el progreso de la aplicación ya que eran necesarios los modelos para una implementación válida.

  Por otro lado, algunos perdían las texturas y *shaders* y aparecían como bloques en blanco, otros ralentizaban el programa debido al tamaño de los archivos. Otros modelos que causaron problemas fueron los focos de fuego. Estos fueron creados, modelados y animados en Blender pero al traspasarlos a Unity perdían la animación adjunta y pasaban de una textura etérea y que emitía luz a una textura rígida y sin color. Se probó con varios modelos de fuego, modelados con diferentes técnicas pero se acabó concluyendo que un modelado, aunque mucho más simplificado, realizado en el propio Unity era más adecuado para evitar un mayor consumo de tiempo.

Figura 4.1: Ejemplos de escala, textura y rotación incorrectas

- **Organización temporal mejorable**: debido a que no se disponía de un equipo con experiencia previa en este tipo de desarrollos de producto, se partía de una base muy inestable. La organización intentó simular la que se emplea en los métodos de *design thinking*, lo cual fue realmente útil para realizar un proyecto más detallado y adecuado para el usuario final. Sin embargo, este se considera resultado negativo, al menos de manera parcial, pues se podía haber aplicado una mejor organización, más eficiente y teniendo más en consideración aspectos, por ejemplo, en el campo de la formación de los integrantes para mejorar el resultado o en el campo de organización de los archivos que se iban obteniendo y de la documentación a traspasar, pues esta podría haberse visto más completa y limpia de lo que fue entregada.

4.1.2. Resultados positivos

Amén de los negativos, también se produjeron resultados positivos en el proceso y los más significativos fueron:

- **VPS y geolocalización funcionales**: tras las primeras pruebas de testeo de esta tecnología se comprobó su eficacia y su margen de error. Este último se consideró lo suficientemente manejable como para incorporarlo en la implementación final de la aplicación. Mediante las coordenadas incorporadas en el proyecto de Unity y el sistema VPS de google, que emplea *Google Lens* para orientarse a través de la cámara mediante *tracking* de imágenes, se juzgó adecuado su uso en pos de la creatividad y la introducción de innovación en el producto.

  Si bien es cierto que en el inicio hubieron ciertos obstáculos en su implementación, como fueron las pruebas de coordenadas dentro de edificios, ya que *Google Maps* no posee coordenadas exactas dentro de los mismos ni puede emplear su sistema VPS, se consiguió manejar, así como otros desafíos que esta técnica opuso, tales como los modelos 3D corruptos.

Figura 4.2: Ejemplos de objetos en coordenadas diferentes

- **Recolección de feedback efectiva**: al comprender en qué fallaba la metodología que se estaba empleando se replanteó la forma de recolección del feedback.

  El primer paso era acercar al usuario a la tecnología que se quería consumir en el producto en desarrollo. Para esto se empleó la aplicación de detección de imágenes que había desarrollado de manera personal para la entrega del *portfolio*, como se ha comentado en la iteración 4 del capítulo tercero de este documento. De esta forma, el usuario podía sentirse más cerca del producto final, pues esta aplicación mostraba un **video en AR** sobre la imagen detectada. Además, así el equipo así disponía de la **pregunta previamente escrita** y no había variaciones en la misma. Y por último, se recolectaban las respuestas en un papel del que los usuarios disponían, con tal de mantener su **anonimato**.

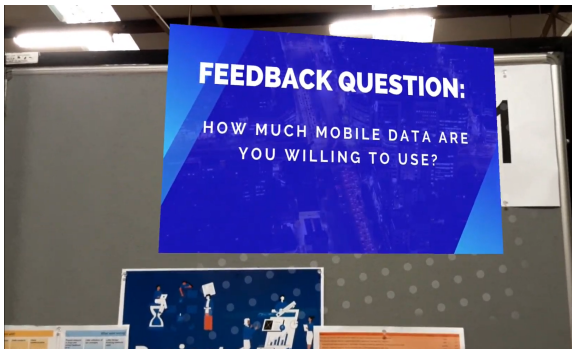


Figura 4.3: Ejemplo de recolección de *feedback* con detección de imágenes

- **Corrección de modelos 3D**: este resultado positivo fue adquirido tras los múltiples resultados negativos comentados anteriormente. Es importante mencionar este resultado como positivo porque refleja el compromiso y el nivel de involucre que se puso en conseguir enmendar este inconveniente.

  Se solucionó el modelado de objetos animados y con emisión mediante la realización de los mismos en el software de Unity directamente, software en el que se compila la aplicación

para su testeo y posterior distribución. También se solucionaron las escalas incorrectas de los modelos 3D que aparecían obstruyendo la cámara AR. Asimismo, se solucionó el texturizado de los modelos de iglesias que aparecían sin colores mediante la correcta conexión de los nodos de materiales que tenían adjuntos. Por último, se consiguió rectificar también los problemas de orientación de los objetos, que en algunos casos se mostraban con transformaciones espaciales como rotaciones e inclinaciones incorrectas o desplazamientos en el espacio lejos de su centro de masas, entre otras.

En la figura 4.1 se puede ver un ejemplo de un modelo original sin correcciones, mientras que en la figura 4.2 se puede ver el modelo con sus texturas aplicadas correctamente y su rotación corregida (escala aún por modificar).

- **Contacto con profesionales**: por último, es remarcable el resultado positivo obtenido tras la puesta en contacto con desarrolladores profesionales. Al afrontar problemas en la programación de la interactividad de los objetos digitales y su '*clicabilidad*', esto es, la capacidad de un objeto 3D para responder al evento de ser pulsado por el usuario, y tras tratar de realizar el programa en múltiples ocasiones, el equipo coincidió que era momento de pedir ayuda a uno o varios programadores *seniors* con más experiencia.

  Se interpreta esto como un resultado muy positivo, no solo por los resultados funcionales que se obtuvieron tras esta puesta en contacto sino por la habilidad del equipo para reconocer el momento en el que el proceso se encontraba atascado y era necesario consultar a profesionales con mayor experiencia en el campo.

En conclusión, en todo proyecto finalizado se espera obtener resultados favorables y positivos, sin embargo, también es importante reconocer los fallos y errores cometidos a lo largo del desarrollo, las mejoras producidas y las que no se han llegado a alcanzar. Esto debe ser así con tal de diseñar mejores productos en el futuro y poder aplicar las metodologías más competentes para cada fase.

La experiencia extraída de este proceso de diseño y creación de producto refleja que, mediante el empleo de metodologías como las de *design thinking* entre otras, se puede realizar un buen estudio de mercado para recopilar información suficiente como para desarrollar un producto útil y con valor. A medida que estas metodologías se emplean y se va adquiriendo conocimientos en ellas y en el mercado al que se quiere enfocar el futuro producto se van produciendo mejoras en el proceso y se avanza de manera segura sobre una base informada. Esto, como en la historia, es importante, pues como se cita al principio de la memoria "*aquellos que no pueden recordar el pasado están condenados a repetirlo*". Tener una base bien fundamentada es, por tanto, tan necesario en la creación de cualquier producto como en cualquier otro aspecto, sea este mundano, político o incluso comercial.

4.2. Perspectivas Futuras

Esta última sección consiste en una reflexión sobre las posibles aplicaciones futuras que esta aplicación y el proyecto en general en las que puede verse reflejado.

Por una parte, este prototipo puede verse extrapolado a otros sucesos históricos similares, situaciones en las que el edificio original haya sufrido un cambio de estilo arquitectónico y se quiera ver reflejado el antiguo respecto al actual de manera simultánea. Además, también puede proporcionar una fuente de información más interesante que la lectura de dicho suceso.

Por otra parte, el prototipo, como se ha mencionado en el capítulo de introducción de este documento, forma parte de un proyecto mayor, un proyecto europeo, siendo este el consorcio europeo de máquinas del tiempo *Time Machine Europe*, con más de 30 países involucrados y cientos de organizaciones. Más de 250 *partners* europeos, siendo estos universidades, archivos y compañías privadas así como historiadores, ingenieros, geógrafos, desarrolladores, emprendedores, investigadores y ciudadanos.

Es por ello que este proyecto puede ser considerado valioso para, principalmente, dos de los objetivos de desarrollo sostenible establecidos para el 2030:

- El objetivo **nº4**: educación de calidad.
- El objetivo **nº17**: alianzas para lograr los objetivos.

No obstante, de manera colateral este proyecto puede encajar en otros objetivos como el nº9, *industria, innovación e infraestructura*, por su parte novedosa en la aproximación a la información con ámbitos históricos y la conservación de la misma; entre otros objetivos mucho más secundarios.

Este producto realizado forma parte, por tanto, de un proyecto de gran envergadura que permite a la sociedad compartir y conservar una historia común sin dejar que esta desaparezca y se olvide. Y esto, a la sociedad europea y global, aporta la oportunidad de construir un futuro conjunto y copartícipe.

Bibliografía

[1] *Polimata RAE*. Del lat. mod. polymathes 'que sabe mucho', y este del gr. πολυμαθής polymathḗs. 4 de Agosto de 2020. URL: https://dle.rae.es/pol%C3%ADmata.

[2] *George Santayana*. 11 de Enero de 2008. URL: https://es.wikipedia.org/wiki/George_Santayana.

[3] *La vida de la razón*. 5 de Junio de 2015. URL: https://es.wikipedia.org/wiki/La_vida_de_la_raz%C3%B3n.

[4] *Den Bosch Time Machine*. 2 de Diciembre de 2020. URL: https://www.erfgoedshertogenbosch.nl/activiteiten/den-bosch-time-machine-en.

[5] *Time Machine Europe*. 14 de Abril de 2018. URL: https://www.timemachine.eu/.

[6] *Design Thinking Phases*. 13 de Abril de 2017. URL: https://www.interaction-design.org/literature/topics/design-thinking.

[7] *Plataforma de desarrollo en Tiempo Real de Unity*. 14 de Marzo de 2019. URL: https://unity.com/es.

[8] *GooBall - Wikipedia*. 13 de Septiembre de 2006. URL: https://en.wikipedia.org/wiki/GooBall.

[9] *Unity - Manual: Barra de herramientas*. 14 de Diciembre de 2020. URL: https://docs.unity3d.com/es/530/Manual/Toolbar.html.

[10] *Unity - Manual: Jerarquia*. 14 de Diciembre de 2020. URL: https://docs.unity3d.com/es/530/Manual/Hierarchy.html.

[11] *Unity - Manual: Vista de Escena*. 14 de Diciembre de 2020. URL: https://docs.unity3d.com/es/530/Manual/UsingTheSceneView.html.

[12] *Unity - Manual: Vista de Juego*. 14 de Diciembre de 2020. URL: https://docs.unity3d.com/es/530/Manual/GameView.html.

[13] *Unity - Manual: Inspector*. 14 de Diciembre de 2020. URL: https://docs.unity3d.com/es/530/Manual/UsingTheInspector.html.

[14] *Unity - Manual: Ventana de proyecto*. 14 de Diciembre de 2020. URL: https://docs.unity3d.com/es/530/Manual/ProjectView.html.

[15] *Unity - Manual: Ventana de Consola*. 14 de Diciembre de 2020. URL: https://docs.unity3d.com/es/2017.4/Manual/Console.html.

[16] *Unity - Manual: Aprendiendo la interfaz*. 14 de Diciembre de 2020. URL: https://docs.unity3d.com/es/530/Manual/LearningtheInterface.html.

[17] *Unity (motor de videojuego)*. Artículo actualizado y con mayor profundización en todas las plataformas admitidas que se tratan en este apartado. 23 de mayo de 2023. URL: https://es.wikipedia.org/wiki/Unity_(motor_de_videojuego)#Plataformas_objetivo.

[18] *Unity - Manual: Unity 2D*. 14 de Diciembre de 2020. URL: https://docs.unity3d.com/es/530/Manual/Unity2D.html.

[19] *Blender*. Actualización más reciente el 11 de Julio de 2023. 24 de Noviembre de 2002. URL: https://www.blender.org/.

[20] *TopBar*. 25 de Noviembre de 2020. URL: https://docs.blender.org/manual/es/2.91/interface/window_system/topbar.html.

[21] *Areas*. 25 de Noviembre de 2020. URL: https://docs.blender.org/manual/es/2.91/interface/window_system/areas.html.

[22] *Barra de Estado*. 25 de Noviembre de 2020. URL: https://docs.blender.org/manual/es/2.91/interface/window_system/status_bar.html.

[23] *Blender - Introducción al sistema de Ventanas*. 25 de Noviembre de 2020. URL: https://docs.blender.org/manual/es/2.91/interface/window_system/introduction.html.

[24] *Google Developers*. 16 de Abril de 2015. URL: https://developers.google.com/?hl=es-419.

[25] *Vuforia*. 30 de Diciembre de 2012. URL: https://developer.vuforia.com/.

[26] *ARCore*. 19 de Agosto de 2022. URL: https://developers.google.com/ar?hl=es-419.

[27] *Unity ARKit Plugin*. 17 de Junio de 2019. URL: https://developer.apple.com/augmented-reality/arkit/.

[28] *Unity MARS*. 29 de Noviembre de 2019. URL: https://unity.com/products/mars/get-started.

[29] *API de ARCore*. 29 de Junio de 2023. URL: https://developers.google.com/ar/develop/geospatial?hl=es-419.

[30] *ManoMotion*. 7 de Abril de 2015. URL: https://www.manomotion.com/.

[31] *Estrategia: Análisis SWOT como herramienta para los Sistemas de Gestión de la Calidad*. 26 de Febrero de 2021. URL: https://www.linkedin.com/pulse/estrategia-an%C3%A1lisis-swot-como-herramienta-de-para-los-ram%C3%ADrez/?originalSubdomain=es.

[32] *5 porqués*. 20 de Agosto de 2022. URL: https://safetyculture.com/es/listas-de-verificacion/5-porques/.

[33] *Definición RAE - ad hoc*. 2 de Agosto de 2020. URL: https://dle.rae.es/ad%20hoc.

[34] *COCD Box - Design Methods*. 20 de Mayo de 2023. URL: https://designforedpolicy.org/cocd-box#:~:text=The%20COCD%20box%20is%20a,and%20ideas%20for%20the%20future..

[35] *ARCore Extensions*. 6 de Diciembre del 2022. URL: https://developers.google.com/ar/develop/unity-arf/getting-started-extensions?hl=es-419.

[36] *Conversor GeoidEval*. 25 de Mayo del 2017. URL: https://geographiclib.sourceforge.io/cgi-bin/GeoidEval.

[37] *Final Overview Presentation*. 26 de Enero de 2023. URL: https://1drv.ms/b/s!AoUkmeuBPjoegx16TvFwSbSv9S4P?e=mLsXSZ.

[38] *Erfgoed s'-Hertogenbosch - Heritage*. 24 de Abril de 2016. URL: https://www.erfgoedshertogenbosch.nl/.

Parte II

Anexos

Capítulo 1

VPSManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR.ARFoundation;
using UnityEngine.XR.ARSubsystems;
using Google.XR.ARCoreExtensions;
using System;
using UnityEngine.UI;

public class VPSManager : MonoBehaviour
{
    [SerializeField] private AREarthManager earthManager;

    [Serializable]
    public struct GeospatialObject
    {
        public GameObject ObjectPrefab;
        public EarthPosition EarthPosition;
    }

    [Serializable]
    public struct EarthPosition
    {
        public double Latitude;
        public double Longitude;
        public double Altitude;
    }

    [SerializeField] private ARAnchorManager aRAnchorManager;
    [SerializeField] private List<GeospatialObject> geospatialObjects = new
↪ List<GeospatialObject>();

    // Start is called before the first frame update
    void Start()
    {
        VerifyGeospatialSupport();
```

```
        }

        void OnDispose()
        {
            Input.location.Stop();
        }

        private void VerifyGeospatialSupport()
        {
            var result =
↪   earthManager.IsGeospatialModeSupported(GeospatialMode.Enabled);
            switch (result)
            {
                // the device DOES support this service
                case FeatureSupported.Supported:
                    Input.location.Start();
                    Debug.Log("Ready to use VPS");
                    PlaceObjects();
                    break;

                // the device MAY or may NOT support this service
                case FeatureSupported.Unknown:
                    Debug.Log("Unknown...");
                    Invoke("VerifyGeospatialSupport", 5.0f);
                    break;

                // the device do NOT support the geospatial mode
                case FeatureSupported.Unsupported:
                    Debug.Log("VPS Unsupported");
                    break;
            }
        }

        private void PlaceObjects()
        {
            if (earthManager.EarthTrackingState == TrackingState.Tracking)
            {
                foreach (var obj in geospatialObjects)
                {
                    var earthPosition = obj.EarthPosition;
                    var objAnchor =
↪   ARAnchorManagerExtensions.AddAnchor(aRAnchorManager, earthPosition.Latitude,
↪   earthPosition.Longitude, earthPosition.Altitude, Quaternion.identity);
                    Instantiate(obj.ObjectPrefab, objAnchor.transform);
                }
            }
            else if (earthManager.EarthTrackingState == TrackingState.None)
            {
                Invoke("PlaceObjects", 5.0f);
            }
        }
    }
```

Capítulo 2

GameRequest.cs

```
using Newtonsoft.Json;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;
using TMPro;
using Unity.VisualScripting.Dependencies.NCalc;
using UnityEngine;
using UnityEngine.Networking;
using UnityEngine.UI;
using static VPSManagerWOA;

public class GameRequest : MonoBehaviour
{
    private string url = "https://jeroene2000.github.io/DBTimeMachine.json";
↪ //This URL has to be changed by the developers to their own JSON
    public TMP_Text question;
    public Text answer;
    public Button findButton;
    public int gameScore = 0;
    public List<GameDataObject> questionsJson;
    public string currentNameOfObject;
    public int fireNumber = 0;

    public void Start()
    {
        StartCoroutine(MakeGameRequest());
    }

    IEnumerator MakeGameRequest()
```

```
        {
            UnityWebRequest request = UnityWebRequest.Get(url);
            yield return request.SendWebRequest();

            if (request.isNetworkError || request.isHttpError)
                Debug.Log(request.error);
            else
            {
                questionsJson =
↪   JsonConvert.DeserializeObject<List<GameDataObject>>(request.downloadHandler.text);
                question = GameObject.Find("Question").GetComponent<TMP_Text>();
                AdvanceQuestion();
            }
        }
        public void AdvanceQuestion()
        {
            currentNameOfObject = StaticClass.objectName;
            fireNumber = Int32.Parse(Regex.Match(currentNameOfObject,
↪   @"\d+").Value);
            question.text =
↪   questionsJson.ToArray()[0].game.questions.ToArray()[fireNumber].question.ToString();
            //get the answers from the questionsJson from the current
↪   question.text
            List<string> answers = new List<string>();
            foreach (var answer in
↪   questionsJson.ToArray()[0].game.questions.ToArray()[fireNumber].answers)
            {
                answers.Add(answer);
            }

            foreach (var item in questionsJson)
            {
                MakeAnswerButtons(answers,
↪   item.game.questions[fireNumber].correctAnswer);
            }

        }

        public void RemoveButton(string buttonName)
        {
            GameObject button = GameObject.Find(buttonName);
            Destroy(button);
        }
        public void MakeAnswerButtons(List<string> answers , int
↪   correctAnswerIndex)
```

```
        {
            if(answers.Count == 0)
            {
                return;
            }
            if (correctAnswerIndex <= answers.Count && correctAnswerIndex >= 0)
            {
                for (int i = 0; i < answers.Count; i++)
                {
                    answer = GameObject.Find("Answer" + i).GetComponent<Text>();
                    answer.text = answers[i];
                    findButton = GameObject.Find("Button" +
↪   i).GetComponent<Button>();
                    findButton.onClick.RemoveAllListeners();
                    findButton.onClick.AddListener(() => CheckAnswer());
                }
            }
        }
        public void CheckAnswer()
        {
            //get the name of which button is clicked and check if it is the
↪   correct answer
            string buttonName =
↪   UnityEngine.EventSystems.EventSystem.current.currentSelectedGameObject.name;
            GameObject quizScreen = GameObject.Find("QuestionCanvas");
            int buttonNumber = Int32.Parse(Regex.Match(buttonName,
↪   @"\d+").Value);
            if (buttonNumber ==
↪   questionsJson.ToArray()[0].game.questions.ToArray()[fireNumber].correctAnswer)
            {
                gameScore++;
                quizScreen.SetActive(false);
            }
            //check if last question is called from question in questionJSon
            var debug =
↪   questionsJson.ToArray()[0].game.questions.ToArray().Last();
            if
↪   (questionsJson.ToArray()[0].game.questions.ToArray()[fireNumber].question.ToString()
↪   == debug.question)
            {
                question.text = "your score is " + gameScore;
                RemoveButton("Button0");
                RemoveButton("Button1");
                RemoveButton("Button2");
```

```
                RemoveButton("Button3");
                return;
            }
            quizScreen.SetActive(false);

        }
    }
```

Capítulo 3

ObjectClick.cs

```
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
using UnityEngine.SceneManagement;

public class ObjectClick : MonoBehaviour
{
    public LayerMask Surfaces;
    public Scene scene;

    void Start() // Start is called before the first frame update
    {
    }

    void Update() // Update is called once per frame
    {
        if (Input.GetMouseButtonDown(0))
        {
            var camera = Camera.allCameras.First(e => e.name == "AR Camera");
            Ray ray = camera.ScreenPointToRay(Input.mousePosition);

            if (!Physics.Raycast(ray, out RaycastHit hit, 10000f, Surfaces))
↪   return;

            if (hit.transform.gameObject.tag == "AR Object")
            {
                StaticClass.objectName = hit.transform.gameObject.name;
                SceneManager.LoadScene("QuizScreen", LoadSceneMode.Additive);
            }
        }
    }
}
```