KIT DE HERRAMIENTAS PARA EL TRATADO DE MAPAS EN CONFERENCIAS EN REALIDAD MIXTA

TRABAJO DE FIN DE TÍTULO

Universidad de Las Palmas de Gran Canaria

Escuela de Ingeniería Informática

Grado en Ingeniería Informática

Autor: Alberto Mejías Márquez

Tutores: José Daniel Hernández Sosa, Agustín Trujillo Pino, Antonio José Sánchez López

Las Palmas de Gran Canaria

Julio 2023

CONTENIDO

AGRADECIMIENTOS

Me gustaría empezar agradeciendo a mi familia y a mis amigos, los de la infancia, los que he conocido en estos años de universidad en la carrera y por supuesto también a ‘mi familia’ del Erasmus, sin todos ellos hubiese sido imposible llegar a este punto.

Y por supuesto agradecer también a la empresa XReality Factory SL, a mi compañero Samuel Trujillo Santana y a mis tutores José Daniel Hernández Sosa, Agustín Trujillo Pino y Antonio José Sánchez López por su disposición y sus aportaciones durante todo este tiempo para que este trabajo pudiese llegar a buen puerto.

RESUMEN

Teniendo en cuenta los tiempos en los que nos encontramos de gran evolución en el ámbito de la Realidad Virtual (RV) y Realidad Aumentada (RA), es imprescindible el desarrollo de una aplicación para conferencias en tiempo real en Realidad Mixta (RM) para que dos o más personas puedan tener una charla, ponencia o reunión. Es por ello por lo que con este trabajo queríamos dar un paso más allá y, teniendo como objetivo principal proporcionar a los usuarios una experiencia de colaboración inmersiva en un entorno tridimensional, hemos enfocado el TFT en el manejo de mapas en RM con el que poder interactuar de diversas formas como dibujar, medir o colocar pegatinas de diferentes colores y tamaños, o colocar modelos 3D como coches o ambulancias que pueden quedar flotando en el aire o quedar ligados a una localización específica del mapa.

Este trabajo ha sido realizado en colaboración con la empresa XReality Factory S.L. que ha aportado la idea de negocio y el material necesario para probar el correcto funcionamiento del proyecto.

SUMMARY

Taking into consideration the current times of evolution in Virtual Reality (VR) and Augmented Reality (AR), the development of a real-time videoconference application in Mixed Reality (MR) seems essential, enabling two or more people to have a conversation, presentation, or meeting. Therefore, with this work, we aimed to go beyond by focusing on providing users with an immersive collaborative experience in a three-dimensional environment. Our final project centers around the manipulation of maps in MR, allowing various interactive features such as drawing, measuring, and placing stickers of different colors and sizes. Additionally, users can position 3D models such as cars or ambulances that can float in the air or be anchored to a specific location on the map.

This work has been carried out in collaboration with the company XReality Factory S.L., which has provided the business idea and the necessary materials to evaluate the proper functioning of the project.

1. INTRODUCCIÓN

1.1CONTEXTO HISTÓRICO.

Las Realidad Virtual, Aumentada y Mixta comprenden unas tecnologías emergentes que han necesitado muchos avances tecnológicos en computación y visualización para alcanzar la madurez que tienen hoy en día.

A la pregunta sobre qué es la Realidad Virtual, la web www.innovae.com nos ofrece la siguiente definición: “La tecnología de Realidad Virtual consiste en la inmersión del usuario en un mundo completamente sintético generado por ordenador, en el que sus sentidos dejan de percibir el mundo real, sumergiendo al usuario en un entorno alternativo.”.

La primera referencia específica a unas gafas de Realidad Virtual data a principios del siglo XX, en los años treinta. En el cuento titulado “Las gafas del Pigmalión”, del escritor de ciencia ficción Stanley G. Weinbaum, un profesor inventa unas gafas que permiten a las personas que se las ponen transportarse a otros mundos y lugares.

En 1957 el cinematógrafo e inventor estadounidense Morton Heilig desarrolló un dispositivo que permitía a una persona disfrutar de experiencias multisensoriales. Este aparato fue conocido como el Sensorama (ver Figura 1) y fue el predecesor de las primeras gafas de Realidad Virtual que fueron creadas en 1960. Es por ello por lo que a Heilig se le conoce como el padre de la Realidad Virtual. Este primer concepto ya se acercaba en aquella época a lo que los equipos de Realidad Virtual ofrecen hoy en día, aunque en la actualidad no son tan rudimentarios como el que aparece en la Figura 1. Es por esto por lo que se considera al Sensorama como las primeras gafas de Realidad Virtual de la historia.

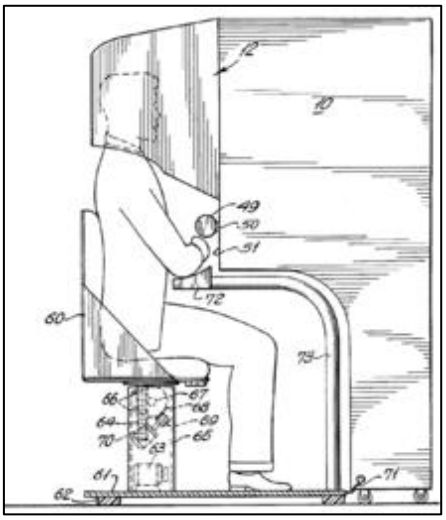

Figura 1.- Patente del Sensorama. Obtenida de https://en.wikipedia.org/wiki/Sensorama.[1]

[1] Las figuras son de creación propia, salvo que se indique lo contrario.

Por otro lado, el concepto de Realidad Aumentada fue utilizado por primera vez en 1901 por el escritor Frank Baum, quién imaginó unas gafas electrónicas para visualizar información adicional sobre las personas que uno tenía delante.

Sin embargo, no es hasta el año 1957 cuando tiene lugar la primera implementación tecnológica basada en realidad aumentada por parte del cinematógrafo Morton Heilig mencionado anteriormente. Y dieciséis años más tarde, en el año 1973, el informático Myron W. Krueger creó lo que sería conocido como la primera instalación de Realidad Aumentada. Esta mezclaba cámaras de video con un sistema de proyección (ver Figura 2) que permitía crear un entorno interactivo que respondía a los movimientos de los usuarios por medio del movimiento y de las sombras del propio usuario.

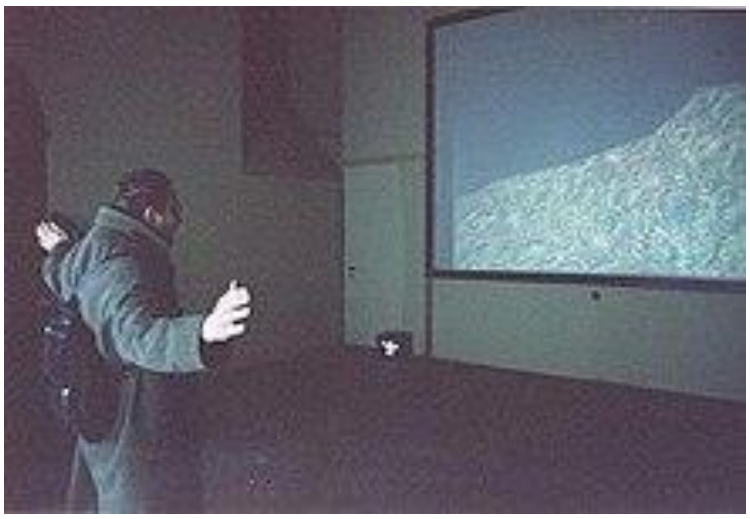

Figura 2.- Proyecto Small Planet presentado por Krueger en Mediartech '98. Obtenida de https://en.wikipedia.org/wiki/Myron_W._Krueger

Por otra parte, la realidad mixta combina la realidad virtual y la realidad aumentada de tal forma que se crean espacios en los que personas y objetos reales interactúan con los virtuales y viceversa.

Según la definición de la Realidad Mixta escrita en Wikipedia “En 1994, Paul Milgram y Fumio Kishino definieron el concepto de realidad mixta como un subconjunto de tecnologías relacionadas con la realidad virtual que implica la fusión de los mundos real y virtual en algún lugar de continuo de la virtualidad.” Esto quiere decir que la realidad mixta no es otra cosa sino un punto intermedio entre el mundo real y el mundo virtual (ver Figura 3) de forma que los elementos que componen el entorno real del usuario sirvan como elementos interactivos en el mundo virtual.

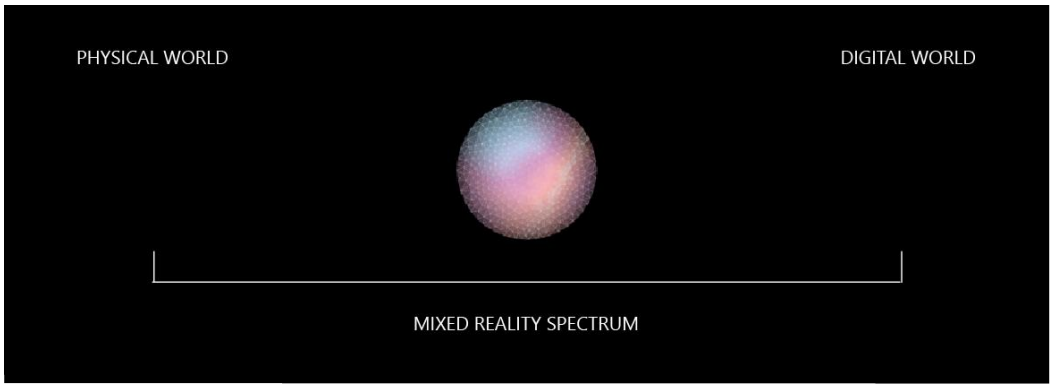

Figura 3.- Espectro de la Realidad Mixta. Obtenido de https://learn.microsoft.com/es-es/windows/mixed-reality/discover/mixed-reality

1.2 DIFERENCIAS ENTRE REALIDADES VIRTUAL, REALIDAD AUMENTADA y REALIDAD MIXTA.

La Realidad Virtual, la Realidad Aumentada y la Realidad Mixta son conceptos que han revolucionado la forma en que los seres humanos interactúan con el mundo digital. Aunque comparten algunas características y objetivos comunes, cada una de estas tecnologías presenta particularidades que las distinguen y les otorgan aplicaciones únicas en diferentes contextos.

1.2.1 REALIDAD VIRTUAL

La Realidad Virtual (RV) es una tecnología capaz de crear un entorno completamente digital que se percibe totalmente real y que sumerge completamente al usuario a través de dispositivos como cascos o gafas de RV. En un entorno de RV, los usuarios pueden interactuar con los objetos que ven como si estos realmente estuvieran allí y dejan de ser conscientes de su entorno real. En definitiva, la RV busca crear una experiencia inmersiva que aísla al usuario del mundo real.

1.2.2 REALIDAD AUMENTADA

La Realidad Aumentada (RA) es una tecnología que combina elementos virtuales con el entorno real, superponiendo información digital, como gráficos, imágenes o texto, al lugar en el mundo real en el que nos encontramos utilizando dispositivos como smartphones, tabletas o gafas especiales de RA. En resumen, la RA enriquece la percepción de la realidad existente con contenido adicional.

1.2.3 REALIDAD MIXTA

La Realidad Mixta (RM) es una combinación de la RV y la RA de forma que no sólo permite la interacción del usuario con el entorno virtual, sino que también permite que los objetos físicos del entorno del usuario sirvan como elementos de interacción con el entorno virtual. En conclusión, la RM se basa en el concepto de la “escala de la realidad”, que va desde lo completamente virtual hasta lo completamente real (ver Figura 4).

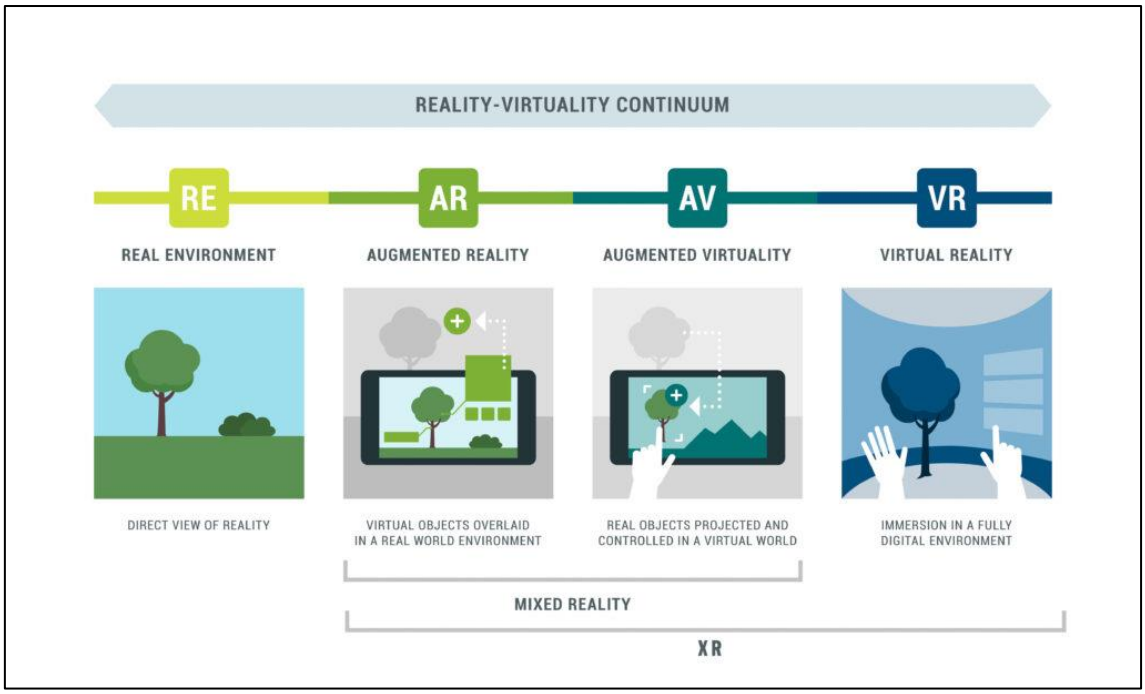

Figura 4.- Continuo de la Realidad Virtual. Obtenida de https://creatxr.com/the-virtuality-spectrum-understanding-ar-mr-vr-and-xr/

Dadas estas definiciones sobre las tres Realidades que nos conciernen podemos apuntar tres diferencias clave:

- Nivel de inmersión.

La RV ofrece una mayor inmersión, ya que los usuarios se sumergen en un mundo completamente virtual y pierden por completo la visión del mundo real. La RA, en cambio, proporciona una inmersión parcial, ya que los usuarios siguen percibiendo el entorno real con información superpuesta digitalmente. La RM por su parte combina elementos virtuales y reales de forma integrada, ofreciendo un mayor nivel de inmersión que la RA, pero menor que la RV.

- Interacción con el entorno.

En la RV, la interacción se realiza principalmente a través de dispositivos de entrada, como mandos o sensores de movimiento. La RA permite la interacción con el entorno real a través de gestos, voz u otros dispositivos. Y en la RM, la interacción se realiza tanto con elementos virtuales como con objetos físicos del entorno real, lo que proporciona una experiencia más natural y versátil.

- Aplicaciones y campos de uso.

La RV se utiliza actualmente en su mayoría en campos como el entretenimiento, la simulación y la formación. La RA por su parte encuentra su aplicación en el ámbito educativo, en el marketing y en los videojuegos de móviles. Y la RM se centra en campos como el diseño de productos, la medicina, la arquitectura y la educación donde la interacción con objetos físicos y virtuales es esencial.

En definitiva, las tres son tecnologías poderosas que tienen sus propias características y por lo tanto cada una se adecua mejor a ciertas aplicaciones que otras. Comprender las diferencias entre ellas es fundamental a la hora de elegir la tecnología más adecuada en función de los requisitos del proyecto. Además, es de destacar que son tecnologías en continuo desarrollo y evolucionan rápidamente.

1.3OBJETIVOS

El **objetivo principal** de este proyecto es conectar a diferentes personas que se encuentran en distintos lugares del mundo en una misma sala para poder así tener una reunión o charla ofreciendo una mayor cercanía en comparación con una simple videollamada. En estas reuniones, todos los usuarios podrán visualizar en la sala un mapamundi 3D con el que se puede interactuar de diferentes maneras. Algo parecido a lo que nos muestra el videojuego Halo con su mesa holográfica (ver Figura 5).

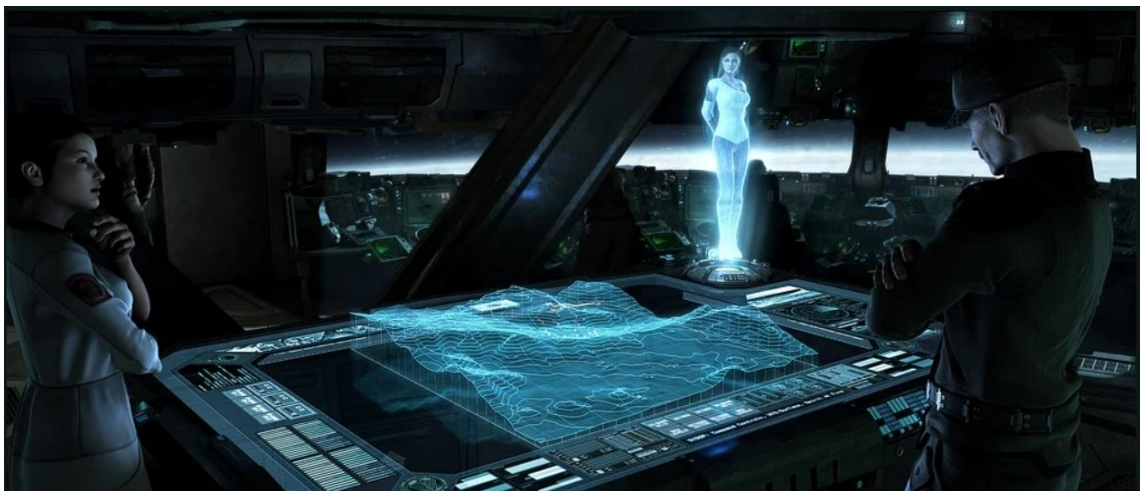

Figura 5.- Mesa holográfica en Halo. Obtenida de https://halo.fandom.com/es/wiki/Holografía

Para un futuro, el **objetivo final** de este proyecto será ofrecer a entidades tanto públicas como privadas de salas virtuales en las que puedan manejar una situación de emergencia en cualquier sitio y que sirva como un añadido a las salas físicas en las que, por esta misma característica, están limitados en espacio, número de pantallas y que requieren desplazarse a un lugar concreto. Un ejemplo podría ser las salas de control de los servicios de emergencias de una ciudad en la que tienen que coordinar diferentes servicios de forma simultánea como policía, bomberos y ambulancias.

Como **objetivo personal**, me propuse aprender a usar todas las tecnologías con las que se ha desarrollado este proyecto como el motor Unity Engine en su variante Unity 3D con el que se desarrolló la aplicación y por lo tanto también aprender a desarrollar código en su lenguaje por defecto orientado a objetos que es C#, lenguaje que por su similitud con java me resultó bastante más sencillo de lo que al principio me esperaba. Por otra parte, también era la primera vez que me enfrentaba a una aplicación multiusuario y, por lo tanto, estaba muy interesado en aprender a usar Photon Engine y la comunicación por llamadas a procedimientos remotos. Y, por último, pero no por ello menos importante, empezar a familiarizarme con el desarrollo para entornos en

realidades extendidas, en este caso realidad mixta, ya que considero que en el futuro próximo ganaran mucho protagonismo en el sector.

En resumen, con este proyecto se pretende aprender a usar nuevas tecnologías creando una aplicación con visión de futuro, pero también de presente, que consideramos que aporta mucho valor en un sector poco explotado hasta la fecha.

El trabajo presentado se ha desarrollado en colaboración con el proyecto titulado "Aplicación de realidad mixta para conferencias online, visualización e interacción conjunta de mapas, y gestión de recursos", realizado por el estudiante Samuel Trujillo Santana.

1.4METODOLOGÍAS

Antes de hablar de las metodologías empleadas para desarrollar este Trabajo de Fin de Título se detallará a continuación la distribución de tareas que se ha realizado entre los proyectos en base a los siguientes siete módulos:

- Un **módulo en conjunto** que estaba compuesto por unas tareas iniciales que consideramos primordiales antes de comenzar el desarrollo de software. En este módulo se encuentran las fases de análisis y de diseño, la creación del proyecto en Unity, integración del paquete Online Maps e instanciación del mapa en la escena, creación del primer de un primer menú principal y unos primeros controladores de la aplicación que nos permitían manejarnos por la misma.
- Un **módulo enfocado en HoloLens 2** que consistía en la integración del paquete MRTK y adaptación de la aplicación a las gafas de realidad mixta de Microsoft que fue realizado por mi compañero Samuel.
- Un **módulo enfocado en la tecnología multiusuario** del que me encargué yo y consistía en la conexión con el servidor de Photon Engine para que la aplicación permitiese que diferentes usuarios pudieran conectarse simultáneamente en la misma sala y que todo aquello que hagan en la aplicación el servidor le diga al resto de clientes lo hagan, es decir, si un usuario se mueve al norte en el mapa, el mapa de todos los demás usuarios con los que está conectado se mueva al norte.
- Un **módulo enfocado en la instanciación de activos** del que también se encargó mi compañero Samuel, el cuál consistía en crear activos dentro de la aplicación y poder manejarlos y manipularlos con ambas manos. Estos activos serían en su mayoría modelos 3D como coches de policía, ambulancias o tiendas de campaña.
- Un **módulo enfocado en herramientas** del que me encargué yo. Este módulo consistía en una serie de herramientas con las que los usuarios podían interactuar entre sí, entre ellos y el mapa y entre ellos y los objetos. Entre estas herramientas destaco el puntero, el poder dibujar con la mano, medir de un punto a otro y colocar una serie de pegatinas. Estas tres últimas herramientas además con la posibilidad de hacerlo tanto en el aire delante de cada uno y en el mapa apuntando hacia el mismo.
- Un **módulo enfocado en un sistema de guardado de salas** que realizó Samuel. Este módulo permite que un usuario guarde en la memoria de las gafas el estado de una sala tal y como este en el momento de pulsar el botón de guardado, es decir, con la

localización concreta y todos los objetos, líneas, mediciones y pegatinas en los lugares en los que estaban.

- Un **módulo enfocado en la comunicación e interacción entre usuarios** que realicé yo. En este módulo se creó un objeto prefabricado que la aplicación instancia por cada uno de los usuarios que entran en la sala, es decir, el avatar de cada usuario. Además, con el paquete Dissonance para Unity los usuarios podían comunicarse entre ellos vía voz.

Una vez comentado esto, quiero hablar de la metodología de trabajo que hemos llevado a cabo. Dentro de las metodologías ágiles utilizamos una enfocada en un desarrollo iterativo e incremental, similar a SCRUM, pero salvando las diferencias puesto que éramos dos personas y no un equipo de desarrollo. Comparándolo con SCRUM podríamos decir que nuestros tutores, tanto los académicos como el tutor de la empresa, fueron los Product Owner.

2. JUSTIFICACIÓN DE LAS COMPETENCIAS

A continuación, se describen las competencias que tienen una relación más directa con este Trabajo de Fin de Título:

I. **CII016.- Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería de software.**

Este proyecto fue realizado siguiendo una metodología ágil iterativa e incremental, similar a SCRUM, en la que cada dos semanas se desarrollaba un prototipo funcional de la aplicación.

II. **CII017.- Capacidad para diseñar y evaluar interfaces persona computador que garanticen la accesibilidad y usabilidad a los sistemas, servicios y aplicaciones informáticas.**

En esta aplicación se desarrolló un menú de botones que permite al usuario comunicarse con el sistema y que, a su vez, hace que el usuario se comunique con servidores remotos para comunicarse con el resto de los usuarios.

III. **IS01.- Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la ingeniería del software.**

Este proyecto fue desarrollado siguiendo una metodología ágil para satisfacer unos requisitos de software que están expuestos en el apartado 5.3 de este documento por medio del código y recursos utilizados para el desarrollo de una aplicación fiable.

IV. **IS03.- Capacidad de dar solución a problemas de integración en función de las estrategias, estándares y tecnologías disponibles.**

En este documento quedan documentados diferentes problemas de integración que surgieron durante el desarrollo del proyecto y las estrategias seguidas para hallar las respectivas soluciones.

V. **IS04.- Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales.**

En este proyecto se han aplicado soluciones de arquitectura de programación que han permitido un desarrollo flexible y una planificación cercana a ser óptima y que nos han hecho crear un proyecto fiable y eficiente.

3. ANÁLISIS

3.1DESCRIPCIÓN

Como se indica en el título de este proyecto, se trata de ofrecer una aplicación que contiene un kit de herramientas para mapas en tres dimensiones en conferencias en realidad mixta, desarrollado para las gafas Microsoft HoloLens 2, donde varias personas pueden conectar en una misma sala para tener reuniones e interactuar con el mapa con las diferentes herramientas de las cuales destacan dibujar, medir, colocar pegatinas o instanciar y manipular objetos.

En la aplicación los usuarios podrán conectarse a una sala y comunicarse con el resto de los usuarios que se conecten a la misma habitación. Todo aquello que hagan esta sincronizado para todo el mundo, incluso si algún usuario se conecta después.

3.2CARACTERÍSTICAS

Como ya se comentaron anteriormente, existen diferentes características o herramientas de las que dispondrán los usuarios de la aplicación que permiten que esta cumpla con los requisitos que se expondrán más adelante:

- Herramienta de Puntero.

Permite al usuario generar un rayo láser que sale en línea recta desde su mano para que todos los demás usuarios puedan ver a donde se está señalando.

- Herramienta de Dibujo.

Permite al usuario dibujar, juntando sus dedos índice y pulgar, tanto en el aire como a ras de mapa, una línea que puede ser continua o discontinua. Antes de dibujar, el usuario deberá elegir entre varios colores y seleccionar el tamaño de la línea que va a dibujar usando el menú.

- Herramienta de Medición.

Al igual que la Herramienta de Dibujo, esta característica permite medir entre dos puntos que bien pueden estar a nivel del mapa o de la propia mano del usuario. El primero de los dos puntos se establece juntando los dedos índice y pulgar, y el segundo, separándolos. También se deberá elegir el tipo de línea, color y grosor previamente a medir.

- Herramienta de Pegatinas.

Permite a los usuarios a colocar una pegatina de entre un set que viene por defecto. De la misma forma que las anteriores dos herramientas, esta también permite colocar la pegatina en el aire o a ras de mapa, hay que juntar los dedos índice y pulgar para crearla y hay que elegir la propia pegatina y el color de esta previamente a instanciarla. Sin embargo, en el caso de colocar la pegatina en el aire al juntar los dedos y mover la mano hacia arriba o derecha la pegatina se hará más grande y abajo o izquierda la hará más pequeña y en el caso de situarla en el mapa, la pegatina siempre tendrá el mismo tamaño y al mover la mano cambiará su localización en el mapa.

- Navegación por el mapa.

Esta herramienta posibilita a los usuarios a moverse por el mapa hacia el Norte, Sur, Este y Oeste además de aumentar o disminuir el zoom.

- Instanciación de activos.

Característica que permite a los usuarios crear objetos en base a modelos 3D tanto en el mapa en tamaño real como en tamaño escalado escondiendo el mapa para poder visualizar el modelo mejor.

- Guardado de escena.

Característica que permite a los usuarios guardar una escena tal y como se encuentra en el momento en la memoria de las gafas para poder instanciarla más adelante. El concepto escena hace referencia a, por ejemplo, haber instanciado en la Escuela de Ingeniería Informática de la ULPGC un coche y pintado con la Herramienta de Dibujo la trazada que tiene que recorrer hasta llegar a la facultad de Ciencias Jurídicas.

- Comunicación por voz.

Esta característica permite a los usuarios comunicarse entre ellos gracias al micrófono y los altavoces que vienen integrados en las gafas HoloLens 2. Además, los usuarios pueden apagar su micrófono y/o silenciar la aplicación si lo prefieren.

3.3REQUISITOS

En lo referente a los requisitos, para este prototipo definimos los siguientes requisitos tanto funcionales como no funcionales:

REQUISITOS FUNCIONALES	
1	El usuario debe poder elegir entre dos salas diferentes.
2	El usuario debe poder ir a cualquier coordenada geográfica en el mapa.
3	El usuario debe poder crear un acceso rápido a una coordenada geográfica.
4	El usuario debe poder aumentar o disminuir el zoom del mapa.
5	El usuario debe poder ver los avatares del resto de usuarios conectados en su sala.
6	El usuario debe poder utilizar un menú para manejar toda la aplicación
7	El usuario debe poder instanciar una serie de modelos 3D de un conjunto predefinido por el sistema.
8	El usuario debe poder activar y desactivar un puntero láser que sale de su mano.
9	El usuario debe poder dibujar líneas de diferentes colores con sus manos.
10	El usuario debe poder medir entre dos puntos que señale con sus manos.
11	El usuario debe poder colocar pegatinas con sus manos.
12	El usuario debe poder elegir si el dibujo, la medición o las pegatinas se instancian en sus dedos o a ras del mapa.
13	Los usuarios que se encuentren en la misma sala deben poder comunicarse entre ellos.
14	El usuario debe poder guardar la escena en disco.

REQUISITOS NO FUNCIONALES	
1	El sistema debe ser intuitivo y fácil de usar para usuarios sin experiencia previa en realidad virtual.
2	El sistema debe tener un rendimiento fluido y una respuesta rápida.
3	El sistema debe poder ser escalable. Que permita añadir mejoras con las posteriores versiones.
4	El sistema debe mantener la sincronización entre usuarios en todo momento.
5	El sistema guardará las teselas del mapa en caché.
6	El sistema debe comunicarse con el servidor multiusuario en un tiempo de respuesta menor a 2 segundos.

4. RECURSOS

4.1HARDWARE

4.1.1 Ordenador portátil personal.

Para el desarrollo de este proyecto he utilizado mi ordenador portátil personal con las siguientes especificaciones:

ESPECIFICACIONES

Sistema Operativo	Microsoft Windows 11 Home
Procesador	Intel Core i7-10750H CPU @ 2.60Hz
Tarjeta Gráfica	NVIDIA GeForce GTX 1650 Ti
RAM	16 GB
Tipo de Sistema	Sistema operativo de 64 bits

4.1.2 Microsoft: HoloLens 2

La aplicación ha sido desarrollada para las gafas de realidad mixta de Microsoft HoloLens 2 (ver Figura 6). De entre todos los dispositivos disponibles para realizar el proyecto se eligió este porque disponía de un ejemplar gracias a la empresa y a mi compañero Samuel.

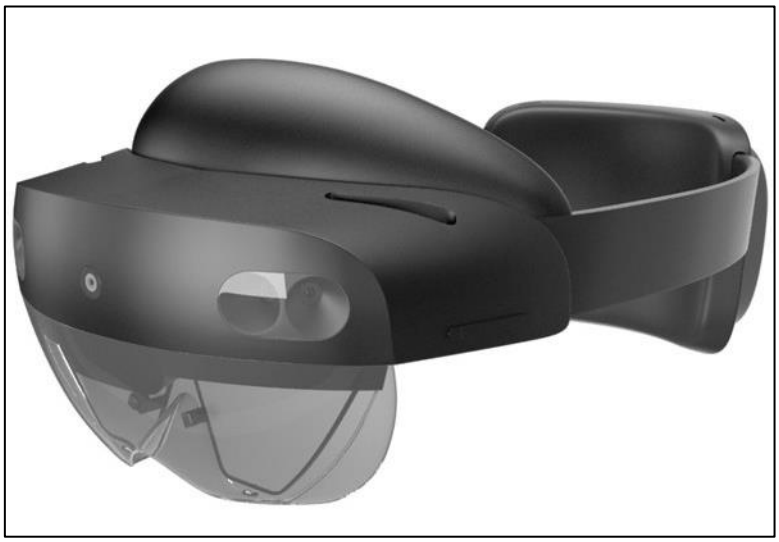

Figura 6.- Imagen de las HoloLens 2. Obtenida de https://get-it-easy.de/en/hololens-2-rent/

4.2SOFTWARE

4.2.1 Unity

Unity es un motor de videojuegos multiplataforma para Microsoft Windows, MacOS y Linux. Tiene soporte de compilación para diferentes tipos de plataformas como Web, PC, dispositivos móviles, consolas y dispositivos de realidad extendida.

Se escogió este motor en su versión de 2019 de entre todos los que hay en el mercado porque tanto yo como mi compañero teníamos destreza con el mismo con anterioridad a empezar el proyecto y es el que utiliza la empresa para desarrollar la aplicación que inspira este trabajo.

Acerca de Unity, es importante saber que es una programación en base a objetos y a componentes. Con esto quiero decir que, al utilizar como lenguaje de programación C# el código es orientado a objetos y, además, dentro de Unity cada GameObject dentro de la escena puede tener una serie de componentes que permiten a los objetos interactuar entre ellos y entre esos componentes están los scripts.

4.2.2 Microsoft Visual Studio

Este fue el IDE (Integrated Development Enviroment) que utilizamos para realizar el código de la aplicación puesto que es el que por defecto recomienda Unity.

El lenguaje de programación ha sido C# que, pese a no haberlo utilizado previamente en el Grado, debido a su similitud con Java fue muy sencillo para mi aprender a utilizarlo rápidamente.

4.2.3 Photon Engine (PUN 2)

El paquete PUN 2 de Photon Engine es el software elegido para las conexiones y sincronización multiusuario pues es un paquete gratuito que permite crear videojuegos multijugador en Unity. Dentro de sus principales características destacan el emparejamiento flexible que conecta a los usuarios a las salas donde todos los objetos pueden estar sincronizados a través de la red, la comunicación por llamadas a procedimientos remotos o propiedades personalizadas para los usuarios y salas.

La aplicación se encuentra alojada en un servidor remoto y tiene asociada una clave única. Photon crea en su servidor, asociado a esa clave única una o más salas, que también tienen su clave única, dando la posibilidad a que haya, en caso de un videojuego, varias partidas simultáneas; en nuestro caso, varias reuniones simultáneas.

La comunicación entre cliente – servidor se realiza por RPCs (en inglés Remote Procedure Calls) (ver Figura 7) que es un procedimiento por el cual el cliente es el que inicia el proceso solicitando al servidor que ejecute cierto procedimiento o función y enviando este de vuelta el resultado de dicha operación al cliente o al resto de clientes.

```
1 referencia
86  public void GoToLocation(string location)
87  {
88      PhotonView photonView = PhotonView.Get(this);
89      photonView.RPC("RPC_GoToLocation", RpcTarget.Others, location);
90  }
91
92  [PunRPC]
    0 referencias
93  void RPC_GoToLocation(string location)
94  {
95      ApplicationController.GetInstance().navigatorController.ApplyGoToLocation(location);
96  }
97
```

Figura 7.- Extracto de código RPC

Haciendo referencia a la Figura 6 ponemos un ejemplo de uso de un RPC. El método *GoToLocation (string location)* es llamado por la aplicación cuando el usuario quiere ir a una localización ya creada. Entonces se crea un objeto de la clase PhotonView que es el encargado de realizar la llamada al procedimiento remoto RPC_GoToLocation (que se encuentra justo debajo en la línea 93) con objetivo “RpcTarget.Others”, es decir, el resto

de los usuarios que no son el que hace la llamada (aquí podemos apreciar la flexibilidad que aporta el motor Photon ya que este parámetro puede modificarse para que el objetivo sean todos los jugadores o incluso que todas las llamadas a procedimientos remotos queden guardadas y se ejecuten en los usuarios que entran posteriormente a una llamada) y con parámetros extra la string que indica la localización concreta. Una vez el servidor recibe esta llamada indica al resto de usuarios que tiene que ejecuten el método RPC_GoToLocation (string location).

El uso de PUN 2 ha sido muy útil y esencial en la realización de este proyecto puesto que sin él haber hecho todas las conexiones multiusuario habrían ocupado mucho más tiempo, trabajo y esfuerzo.

4.2.4 Mixed Reality Toolkit (MRTK)

Mixed Reality Toolkit es un Proyecto controlado por Microsoft que proporciona un conjunto de componentes y características para acelerar el desarrollo de aplicaciones de Realidad Mixta multiplataforma en Unity. Es un software imprescindible en el proyecto gracias a sus principales funcionalidades:

- Proporciona el sistema de entrada multiplataforma y los bloques de reacción para las interacciones espaciales y la interfaz de usuario.
- Permite la creación rápida de prototipos gracias a la simulación en el editor, que permite ver los cambios inmediatamente.
- Funciona como plataforma extensible que permite a los desarrolladores intercambiar los componentes principales.
- Es compatible con una amplia gama de dispositivos como Microsoft HoloLens 2, Windows Mixed Reality headsets, Meta Quest, Android, iOS and SteamVR.

MRTK ha sido uno de los softwares más importantes de todo el proyecto gracias a todo lo que nos ha facilitado el trabajo, pero también es cierto que ha sido complicado de entender y de familiarizarse con su uso al completo.

4.2.5 Infinity Code - Online Maps

Online Maps es un paquete para Unity de Infinity Code que permite crear mapas en dos y tres dimensiones. Permite una gran personalización, es fácil de usar y de aprender a usar, pero al mismo tiempo es uno de los paquetes más completos de la industria.

Además, La API de Online Maps permite personalizar los comportamientos del mapa sin tener que cambiar el código fuente del paquete, sin embargo, el propio paquete permite cambiar el código fuente en caso de que necesitásemos implementar alguna funcionalidad que no pudiésemos hacer con la API por defecto.

Cabe destacar que sin la ayuda económica de la empresa a la hora de proporcionar este paquete no habría sido posible acceder al mismo puesto que tiene un coste elevado para nosotros.

4.2.6 Dreamteck Splines

El paquete de Unity Dreamteck Splines fue el escogido para realizar algunas de las herramientas principales de la aplicación, concretamente las de dibujar y medir (ver

Figura 8). Este paquete proporciona una herramienta eficiente y poderosa para trabajar con splines. Los splines son curvas suaves que se utilizan en el desarrollo de videojuegos y aplicaciones interactivas para crear trayectorias de movimiento, terrenos deformables, objetos lineales y mucho más.

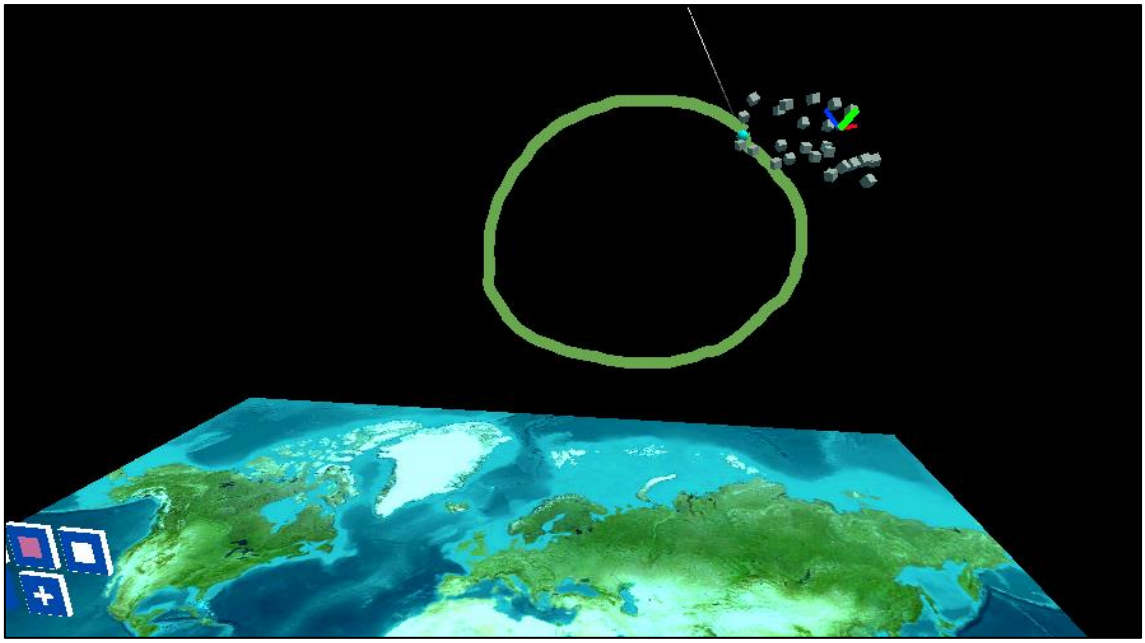

Figura 8.- Dibujo realizado con Dreamteck Splines

Una de sus principales ventajas es su facilidad de uso. Gracias a una interfaz intuitiva que permite crear y editar splines de forma rápida y sencilla con el editor de Unity y por código en tiempo real. Permite generar splines tanto en 2D como en 3D, ajustar los puntos de control, cambiar la forma de la curva y manipular los segmentos para obtener unos resultados muy precisos.

Entre todas sus características principales destacan para este proyecto las funcionalidades avanzadas con las que se puede adaptar la tensión del spline, la interpolación, el suavizado automático y la optimización del rendimiento. Además, su fácil integración con Unity3D permitiendo interactuar con otros paquetes previamente mencionados como Online Maps y MRTK y la compatibilidad multiplataforma ya que Dreamteck Splines es compatible con juegos y aplicaciones para PC, consolas y dispositivos móviles.

4.2.7 Dissonance Voice Chat

El paquete Dissonance Voice Chat fue proporcionado por la empresa y fue importado en el proyecto para facilitar la comunicación entre usuarios. Este paquete se sincroniza con Photon PUN 2 de forma que es muy sencillo de configurar en proyectos multiusuario que usen esta variante de Photon Engine.

De entre sus principales características destacan la fácil instalación en el proyecto, cancelación de ruido y de eco, posibilidad a diferentes canales de audio o playback adaptativo por el que en caso de mala conexión por parte de un usuario recrea la voz con posterioridad.

4.3OTROS RECURSOS SOFTWARE

Por otra parte, sin tener en cuenta el software utilizado dentro de la aplicación, a la hora de desarrollar la misma se han utilizado diferentes programas para coordinarme con mi compañero y la gestionar el proyecto.

4.3.1 Microsoft Teams

Software elegido para comunicarme tanto con mi compañero como con los diferentes tutores en calidad de videollamadas. Además, la usamos para enviarnos información entre nosotros que consideramos que podía ser interesante.

4.3.2 Trello

Software de gestión de proyectos usado para la administración de este. Gracias a su sistema de tableros, listas y tarjetas fuimos capaces de organizar entre mi compañero y yo las diferentes tareas que consideramos y adjudicarle una prioridad y un tiempo estimado a cada uno siguiendo una metodología de proyectos ágil.

4.3.3 GitLab

Como software de control de versiones usamos una versión basada en GitLab proporcionada por la empresa. Para toda la duración del proyecto seguimos el mismo protocolo de trabajo en el que ambos trabajábamos en ramas separadas que representaban la funcionalidad concreta en la que estábamos trabajando y una vez cada dos semanas, antes del sprint común, fusionábamos nuestras ramas con la rama de la cabecera del repositorio. De esta forma teníamos siempre una versión estable y actualizada mientras trabajábamos en paralelo en otras ramas.

5. DISEÑO

Una vez completada la fase de análisis y conociendo los recursos de los cuales disponíamos para realizar la aplicación, mi compañero y yo pasamos a la fase de diseño. En este caso decidimos destinar el tiempo para esta fase en diseñar la interfaz de usuario, un diagrama de casos de uso que haga que se cumplan los requisitos y un diagrama de clases que represente las relaciones entre las clases principales de la aplicación.

5.1DISEÑO DE LA INTERFAZ DE USUARIO

Para el diseño de interfaz de usuario se intentó hacer una versión más simple del menú que existe en la aplicación real de la empresa en la que Samuel y yo trabajamos. Antes de pasar a explicar cómo se comporta este menú, primero se debatió la paleta de colores. Estuvimos mirando diferentes Pantones para elegir colores que tuvieran conexión entre ellos y que no quedase chocante para el usuario. Tras ver varios encontramos un bonito contraste entre un color frío con otros dos más cálidos y que representarían que el botón ha sido pulsado o no respectivamente. Para el primero elegimos un ‘Cobalt Blue’ (#0047AC) y los otros dos son el ‘Cyber Yellow’ (#FFD300) que fue utilizado para resaltar el botón cuando el usuario pasaba el puntero por encima, y el color ‘Ripe Mango’ (#FFB921) que se utilizó para los botones que permanecían pulsados (ver Figura 9).

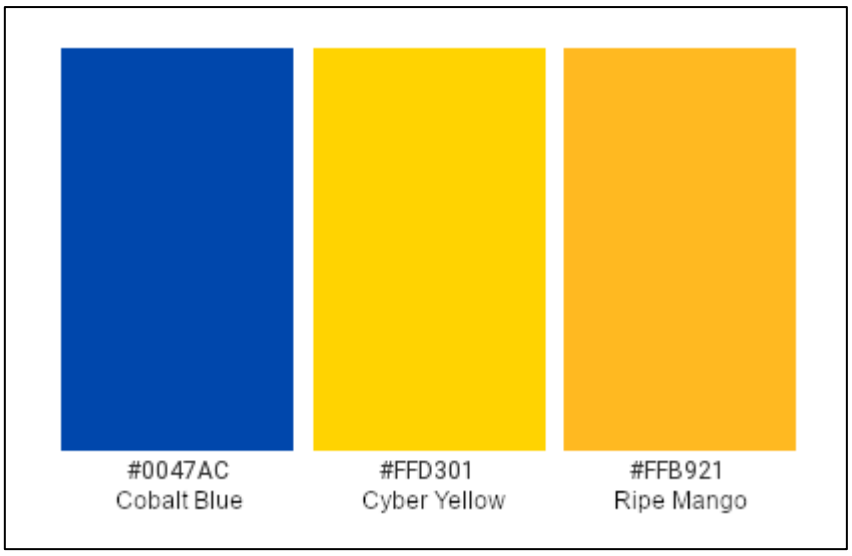

Figura 9.- Gama de colores utilizados en el menú. Obtenida de https://www.schemecolor.com/blue-yellow.php

Este menú tiene un controlador general (a partir de ahora MainMenuController) que contiene a todos los demás botones. Está compuesto por un menú principal que tiene un total de ocho botones, de los cuales siete abren diferentes submenús que contienen a su vez más botones. El octavo botón del menú principal es el botón que permite al usuario cerrar el menú. Por lo general, el menú principal crece hacia abajo y los submenús hacia la derecha y hacia abajo (ver Figura 10). El MainMenuController tiene tres principales funciones: almacenar todos los botones que se crean, crear listas de botones y actualizar los paneles de los botones.

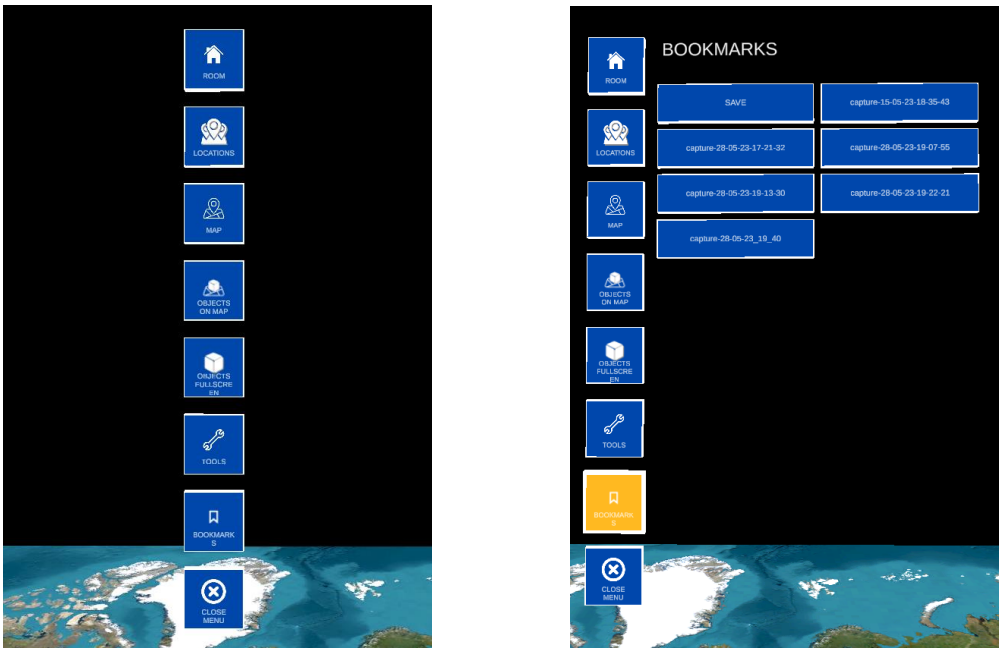

Figura 10.- Menú principal y Submenú Bookmarks

En cuanto a los paneles, me gustaría aclarar que un panel es un conjunto de botones que forman un submenú y una etiqueta que indica su nombre, por ejemplo, en la figura 9 el submenú Bookmarks sería un panel si no tuviésemos en cuenta todos los botones cuadrados que forman el menú principal. Estos paneles utilizan también el script ButtonController.

- La primera función no necesita mayor explicación, el controlador del menú debe tener acceso y control sobre todos los botones que se han creado en el menú.
- La segunda permite al sistema separar aquellos botones que tienen relación entre sí formando grupos. Gracias a una clase interna, MenuContentGroupInfo, se puede guardar una lista de botones y asignarle dos parámetros booleanos extra a esa lista que indican si todos los botones de ese grupo pueden estar desactivados y si al activar uno de los botones del grupo se llama a los métodos de desactivar los demás botones (ver Figura 11). Es decir, con esta clase lo que se propone es replicar lo que en el desarrollo web se conoce como radio-buttons y que, al pulsar, por ejemplo, el botón del menú principal de Tools se apaguen el resto de los botones del menú principal.
- La tercera y no por ello menos importante se encarga de actualizar cada una de las mallas de los paneles y botones. El funcionamiento detrás de esto es muy sencillo, cada vez que se apaga y enciende un panel se desactivan todas las mallas y se activan al

final de cada frame y esto se hace para evitar que siempre se estén reorganizando correctamente y no se active uno encima de otro y se generen fallos en el menú.

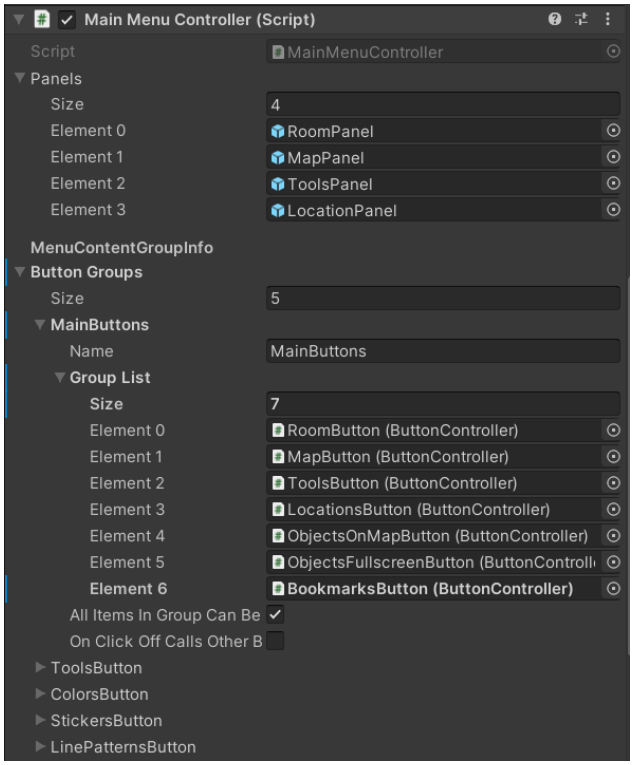

Figura 11.- Script MainMenuController

Por otra parte, el controlador de los botones (de ahora en adelante ButtonController) es un script (ver Figura 12) que tiene cada uno de los botones y que principalmente se encarga de guardar la información del propio botón como id o si se puede interactuar con el botón o no; elementos principales como lo son la etiqueta, el icono, el borde y el fondo del botón; configuración estética como color, etiqueta, icono y tamaño del borde para sus tres variantes en las que se puede encontrar que son sin pulsar, pulsado y señalado; configuración general del botón que puede indicar si por defecto aparecerá apagado o encendido, el método de accionado que puede ser de mantener pulsado, encender/apagar o click y una lista de botones que dependen de este mismo botón; y por último los métodos a los que llamará ese botón en función de su método de accionado asignado.

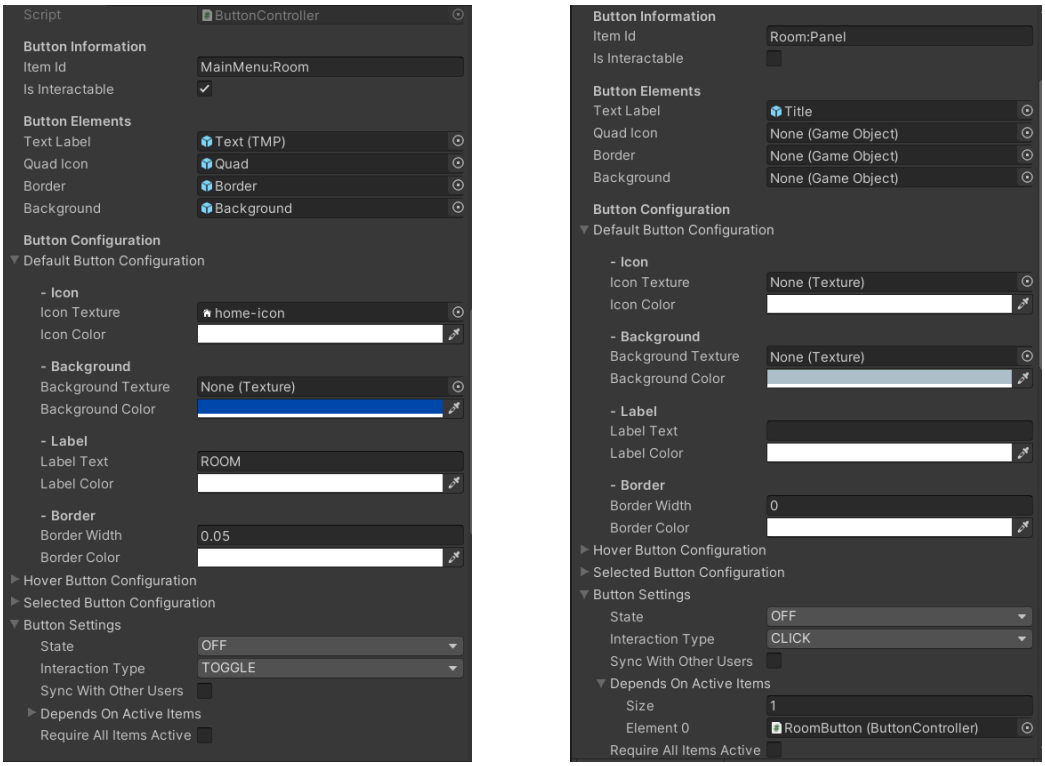

Figura 12.- Script ButtonController de Botón y Panel

Anteriormente se habló de los paneles y se indicó que también utilizaban el script ButtonController pese a no ser un botón como tal. Esto es gracias al booleano que indica si se puede interactuar o no con el objeto que tiene el script ButtonController como componente. Para los botones, este booleano es verdadero mientras que para los paneles es falso. Además, los paneles tienen otro botón dentro de la lista de botones de la que dependen. Esto es así para que una vez se pulse uno de los botones (o bien todos) se muestre o no el panel en concreto (ver Figura 12).

5.2 DIAGRAMA DE CASOS DE USO

En este diagrama de Casos de Uso quedan representadas las principales funcionalidades que puede realizar el actor 'User' (a partir de ahora referido como Usuario) dentro de la aplicación.

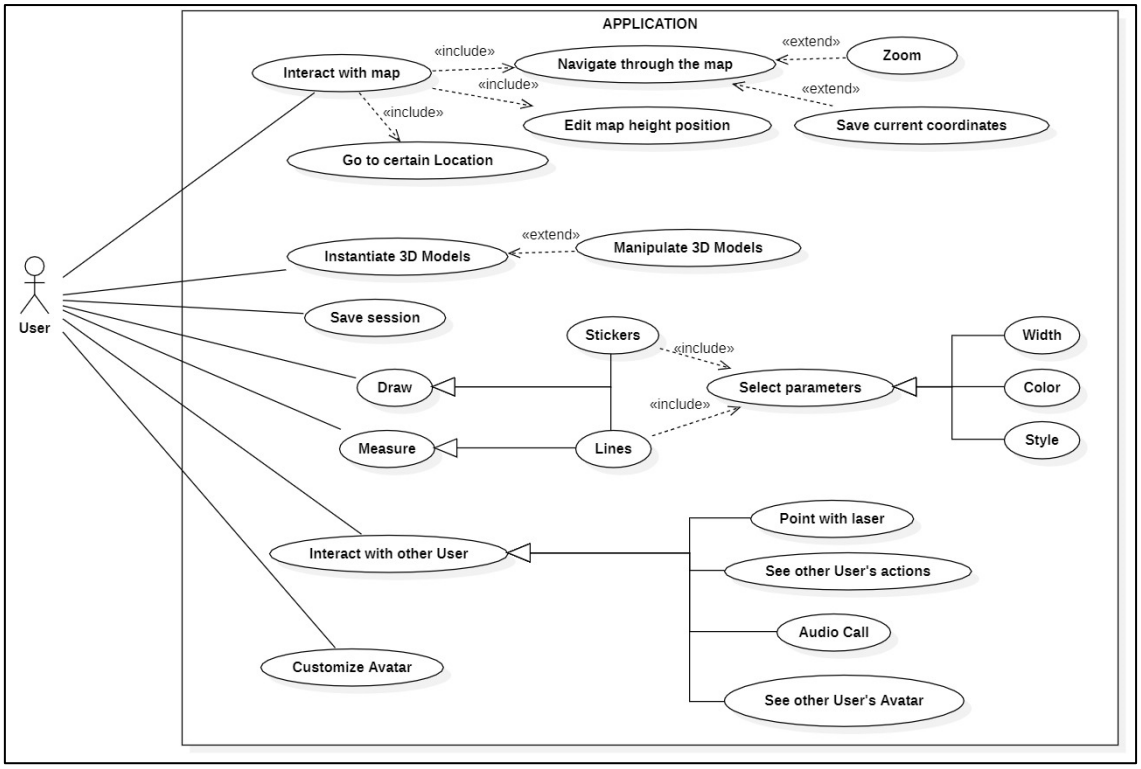

Diagrama 1.- Casos de Uso

5.3ESPECIFICACIONES DE CASOS DE USO

CU01	**INTERACT WITH MAP**
Descripción	El usuario puede interactuar con el mapa.
Precondiciones	El usuario debe haber podido conectarse a una sala.
Flujo normal	
1.	El usuario abre el menú principal.
2.	El usuario interactúa con el mapa.
Variantes	
2a.	Navegar por el mapa. INCLUDE CU02.
2b.	Editar altura del mapa. INCLUDE CU03.
2c.	Ir a una localización concreta. INCLUDE CU04.
Extensiones	
Postcondiciones	El mapa ha cambiado para todos los usuarios conectados.
Excepciones	

CU02	**NAVIGATE THROUGH THE MAP**	
Descripción	El usuario se mueve por el mapa en una dirección.	
Precondiciones	El usuario debe haber podido conectarse a una sala.	
Flujo normal		
1.	El usuario abre el submenú MAP.	
2.	El usuario mantiene pulsado durante unos segundos el botón con la etiqueta NORTH.	
3.	El mapa se mueve al norte los segundos que el usuario mantenga el botón pulsado.	
Variantes		
2a.	El usuario pulsa el botón con la etiqueta EAST.	
	2a.1	El mapa se mueve al este los segundos que el usuario mantenga el botón pulsado.
2b.	El usuario pulsa el botón con la etiqueta SOUTH.	
	2b.1	El mapa se mueve al sur los segundos que el usuario mantenga el botón pulsado.
2c.	El usuario pulsa el botón con la etiqueta WEST.	
	2c.1	El mapa se mueve al oeste los segundos que el usuario mantenga el botón pulsado.
Extensiones		
4	Hacer Zoom	CU05 ZOOM
5	Guardar Coordenadas	CU06 SAVE CURRENT COORDINATES
Postcondiciones	Las coordenadas geográficas que muestra el mapa han cambiado.	
Excepciones		

CU03	**EDIT MAP HEIGHT POSITION**	
Descripción	El usuario podrá subir y bajar el mapa verticalmente.	
Precondiciones	El usuario debe haber podido conectarse a una sala.	
Flujo normal		
1.	El usuario abre el submenú MAP.	
2.	El usuario mantiene pulsado durante unos segundos el botón con la etiqueta MAP UP.	
3.	El mapa se mueve verticalmente hacia arriba los segundos que el usuario mantenga el botón pulsado.	
Variantes		
2a.	El usuario mantiene pulsado durante unos segundos el botón con la etiqueta MAP DOWN.	
	2a.1	El mapa se mueve verticalmente hacia abajo los segundos que el usuario mantenga el botón pulsado.
Extensiones		
Postcondiciones	La posición del mapa ha cambiado.	
Excepciones		

CU04	**GO TO CERTAIN LOCATION**
Descripción	El usuario puede ir automáticamente a una localización del mapa pulsando un botón.
Precondiciones	El usuario debe haber podido conectarse a una sala.
	La localización debe haber sido guardada previamente.
Flujo normal	
1.	El usuario abre el submenú LOCATIONS.
2.	El usuario pulsa uno de los botones cuya etiqueta es el nombre de la localización.
3.	El mapa automáticamente muestra el lugar que corresponde a esa localización.
Variantes	
Extensiones	
Postcondiciones	Las coordenadas geográficas que muestra el mapa han cambiado.
Excepciones	

CU05	**ZOOM**	
Descripción	El usuario puede aumentar o disminuir el nivel de zoom en el mapa.	
Precondiciones	El usuario debe haber podido conectarse a una sala.	
Flujo normal		
1.	El usuario abre el submenú MAP.	
2.	El usuario mantiene pulsado durante unos segundos el botón con la etiqueta ZOOM IN.	
3.	El nivel del zoom del mapa aumenta y por lo tanto se acerca la imagen que muestra el mapa.	
Variantes		
2a.	El usuario mantiene pulsado durante unos segundos el botón con la etiqueta ZOOM OUT.	
	2a.1	El nivel del zoom del mapa disminuye y por lo tanto se aleja la imagen que muestra el mapa.
Extensiones		
Postcondiciones	El mapa cambia para todos los usuarios.	
Excepciones		

CU06	**SAVE CURRENT COORDINATES**	
Descripción	El usuario puede guardar la localización y zoom que muestra el mapa como una localización nueva.	
Precondiciones	El usuario debe haber podido conectarse a una sala.	
Flujo normal		
1.	El usuario abre el submenú LOCATIONS.	
2.	El usuario escribe el nombre de la etiqueta que tendrá la localización en el panel de la derecha.	
3.	El usuario pulsa el botón SAVE.	
4.	El sistema guarda en disco la localización y zoom actual del mapa con la etiqueta escrita.	
Variantes		
3a.	El usuario pulsa el botón DELETE.	
	Paso	Descripción paso
Extensiones		
Postcondiciones	Se crea un botón en el submenú LOCATIONS con la etiqueta escrita que lleva a las coordenadas guardadas	
Excepciones		
2	La etiqueta contiene la string vacía	
	Paso manejo	Descripción paso manejo

CU07	**INSTANTIATE 3D MODELS**		
Descripción	El usuario puede instanciar modelos 3D dentro de la aplicación.		
Precondiciones	El usuario debe haber podido conectarse a una sala.		
Flujo normal			
1.	El usuario abre el submenú OBJECTS ON MAP.		
2.	El usuario pulsa uno de los botones que tienen como etiqueta el objeto que se va a instanciar.		
3.	El sistema crea un objeto 3D dentro del mapa a escala real.		
Variantes			
1a.	El usuario abre el submenú OBJECTS FULLSCREEN.		
	1a.1	El usuario pulsa uno de los botones que tienen como etiqueta el objeto que se va a instanciar.	
	1a.2	El sistema esconde el mapa.	
	1a.3	El sistema crea un objeto 3D en una escala mayor a la real.	
Extensiones			
4	Manipular modelos	CU08 MANIPULATE 3D MODELS	
Postcondiciones	El objeto se instancia para todos los jugadores que se encuentran en la sala.		
Excepciones			

CU08	**MANIPULATE 3D MODELS**	
Descripción	El usuario podrá manipular los modelos 3D que instancie de tres formas diferentes: cambiar su posición, rotación y/o tamaño.	
Precondiciones	El usuario debe haber podido conectarse a una sala.	
	Alguno de los usuarios debe haber instanciado al menos un modelo 3D.	
Flujo normal		
1.	El usuario apunta con la mano al objeto 3D.	
2.	El usuario junta los dedos índice y pulgar de una de las manos como si estuviera “pellizcando” el aire.	
3.	El usuario mueve la mano mientras sigue “pellizcando”.	
4.	El sistema mueve el objeto 3D junto a la mano.	
Variantes		
3a	Rotar el objeto 3D.	
	3a.1	El usuario gira la mano mientras sigue “pellizcando”.
	3a.2	El sistema rota el objeto 3D junto a la mano.
3b	Hacer más grande el objeto 3D	
	3b.1	El usuario “pellizca” el objeto con la otra mano.
	3b.2	El usuario separa sus manos.
	3b.3	El sistema agranda el objeto 3D tanto como se han separado las manos.
3c	Hacer más pequeño el objeto 3D	
	3b.1	El usuario “pellizca” el objeto con la otra mano.
	3b.2	El usuario junta sus manos.
	3b.3	El sistema hace más pequeño el objeto 3D tanto como se han juntado las manos.
Extensiones		
Postcondiciones	Los cambios realizados al modelo se ven reflejados para todos los usuarios de la sala.	
Excepciones		

CU09	**SAVE SESSION**
Descripción	El usuario podrá guardar el estado de la sala tal y como está en la memoria del dispositivo.
Precondiciones	El usuario debe haber podido conectarse a una sala.
Flujo normal	
1.	El usuario abre el submenú BOOKMARKS.
2.	El usuario pulsa el botón SAVE.
3.	El sistema crea un archivo en formato JSON que contiene toda la información acerca de la sala, del mapa y su contenido.
4.	El sistema guarda el JSON en la memoria del dispositivo.
5.	El sistema creará un botón en el submenú BOOKMARKS con la fecha y hora en la que se guardó la sala.
Variantes	
Extensiones	
Postcondiciones	
Excepciones	

CU10	**DRAW LINES**	
Descripción	El usuario podrá dibujar con su mano líneas de diferentes colores, tamaños y estilos.	
Precondiciones	El usuario debe haber podido conectarse a una sala.	
Flujo normal		
1.	El usuario abre el submenú TOOLS.	
2.	El usuario pulsa el botón DRAW.	
3.	El usuario no pulsa el botón ON AIR	
4.	El usuario selecciona los diferentes parámetros. INCLUDE CU13	
5.	El usuario junta los dedos índice y pulgar de una de las manos como si estuviera “pellizcando” el aire.	
6.	El usuario mueve la mano.	
7.	El sistema crea una línea siguiendo la estela del dedo índice.	
8.	El usuario deja de “pellizcar” el aire con los dedos.	
9.	El sistema deja de crear la línea.	
Variantes		
3a.	Dibujo en el mapa.	
	3a.1	El usuario pulsa el botón ON AIR.
	3a.2	El sistema enciende el botón y lo cambia a ON MAP.
	3a.3	El usuario selecciona los diferentes parámetros. INCLUDE CU13.
	3a.4	El usuario junta los dedos índice y pulgar de una de las manos como si estuviera “pellizcando” el aire.
	3a.5	El usuario mueve la mano.
	3a.6	El sistema crea una línea siguiendo en el punto en el que se cruza la proyección de la mano del usuario con el mapa.
	3a.7	El usuario deja de “pellizcar” el aire con los dedos.
	3a.8	El sistema deja de crear la línea.

Extensiones	
Postcondiciones	Las líneas se dibujan para todos los usuarios conectados en la sala.
Excepciones	

CU11	**DRAW STICKERS**	
Descripción	El usuario podrá dibujar con su mano diferentes pegatinas de diferentes colores y tamaños.	
Precondiciones	El usuario debe haber podido conectarse a una sala.	
Flujo normal		
1.	El usuario abre el submenú TOOLS.	
2.	El usuario pulsa el botón STICKERS.	
3.	El usuario no pulsa el botón ON AIR	
4.	El usuario selecciona los diferentes parámetros. INCLUDE CU13	
5.	El usuario junta los dedos índice y pulgar de una de las manos como si estuviera “pellizcando” el aire.	
6.	El sistema crea una pegatina con el color y el estilo que ha seleccionado el usuario en el paso 4 en donde el usuario ha pellizcado.	
7.	El usuario mueve la mano.	
8.	El sistema varía el tamaño de la pegatina en función de lo que el usuario mueve la mano.	
9.	El usuario deja de “pellizcar” el aire con los dedos.	
10.	El sistema crea una pegatina con un tamaño en función a lo que el usuario ha movido la mano.	
Variantes		
3a	Pegatinas en el mapa.	
	3a.1	El usuario pulsa el botón ON AIR.
	3a.2	El sistema enciende el botón y lo cambia a ON MAP.
	3a.3	El usuario selecciona los diferentes parámetros. INCLUDE CU13.
	3a.4	El usuario junta los dedos índice y pulgar de una de las manos como si estuviera “pellizcando” el aire.
	3a.5	El sistema crea una pegatina con el color y el estilo que ha seleccionado el usuario en el paso 4 y de un tamaño estandarizado justo en el punto en el que se cruza la proyección de la mano del usuario con el mapa.
	3a.6	El usuario mueve la mano

	3a.7	El sistema mueve la pegatina en función de lo que el usuario mueve la mano.
	3a.8	El usuario deja de “pellizcar” el aire con los dedos.
	3a.9	El sistema deja la pegatina en el lugar en el que el usuario dejó de “pellizcar” el aire con sus dedos.
Extensiones		
Postcondiciones	Las pegatinas se crean para todos los usuarios en la sala.	
Excepciones		

CU12	**MEASURE LINES**	
Descripción	El usuario podrá medir la distancia entre dos puntos con sus manos.	
Precondiciones	El usuario debe haber podido conectarse a una sala.	
Flujo normal		
1.	El usuario abre el submenú TOOLS.	
2.	El usuario pulsa el botón MEASURE.	
3.	El usuario no pulsa el botón ON AIR	
4.	El usuario selecciona los diferentes parámetros. INCLUDE CU13	
5.	El usuario junta los dedos índice y pulgar de una de las manos como si estuviera “pellizcando” el aire.	
6.	El sistema crea un cubo del color seleccionado.	
7.	El usuario mueve la mano.	
7.	El sistema crea una línea recta siguiendo el dedo índice.	
8.	El usuario deja de “pellizcar” el aire con los dedos.	
9.	El sistema crea otro cubo y una línea que empieza en el primer cubo y acaba en este segundo junto con un texto encima de la línea que indica la distancia entre ambos cubos.	
Variantes		
3a.	Medir en el mapa.	
	3a.1	El usuario pulsa el botón ON AIR.
	3a.2	El sistema enciende el botón y lo cambia a ON MAP.
	3a.3	El usuario selecciona los diferentes parámetros. INCLUDE CU13.
	3a.4	El usuario junta los dedos índice y pulgar de una de las manos como si estuviera “pellizcando” el aire.
	3a.5	El sistema crea un cubo del color seleccionado.
	3a.6	El usuario mueve la mano.
	3a.7	El sistema crea una línea siguiendo en el punto en el que se cruza la proyección de la mano del usuario con el mapa.

	3a.8	El usuario deja de “pellizcar” el aire con los dedos.
	3a.9	El sistema crea otro cubo y una línea que empieza en el primer cubo y acaba en este segundo junto con un texto encima de la línea que indica la distancia entre ambos cubos.
Extensiones		
Postcondiciones		
Excepciones		

CU13	**SELECT PARAMETERS**	
Descripción	Seleccionar parámetros para las diferentes herramientas.	
Precondiciones	El usuario debe haber podido conectarse a una sala.	
	El usuario debe haber seleccionado una de las tres herramientas.	
Flujo normal		
1.	El sistema muestra los botones correspondientes a la herramienta seleccionada.	
2.	El usuario selecciona en el menú los diferentes parámetros.	
Variantes		
2a.	Color	
	2a.1	El usuario selecciona uno de los siete colores.
	2a.2	El sistema enciende el botón del color pulsado.
	2a.3	El sistema cambia el color de la línea de muestra.
2b.	Estilo	
	2b.1	El usuario selecciona uno de los estilos en función de la herramienta (dos para las líneas, siete para las pegatinas).
	2b.2	El sistema enciende el botón del color pulsado.
2c.	Tamaño	
	2c.1	El usuario pulsa el botón '+' o '-' si quiere hacer más grande o pequeña la línea.
	2c.2	El sistema hace más grande o pequeña la línea de muestra.
Extensiones		
Postcondiciones		
Excepciones		
2c	Tamaño de las pegatinas	
	2c.1	El usuario establece el tamaño de las pegatinas al instanciarlas moviendo la mano.

CU14	**INTERACT WITH OTHER USERS**	
Descripción	Diferentes formas de interactuar con el resto de los usuarios.	
Precondiciones	Varios usuarios deben estar conectados en la misma sala.	
Flujo normal		
1.	Abrir el menú.	
2.	Activar las herramientas correspondientes.	
Variantes		
2a.	Apuntar con el láser.	
	2a.1	El usuario activa el láser en el submenú TOOLS.
	2a.2	El sistema genera un rayo láser que sale de la mano del usuario en línea recta.
2b.	Ver las acciones del resto de usuarios.	
2c.	Llamada de audio.	
	2c.1	El usuario activa el audio y el micrófono en el submenú ME.
	2c.2	El usuario habla.
	2c.3	El sistema manda la voz al resto de usuarios en la sala.
2d.	Ver el avatar del resto de los usuarios.	
Extensiones		
Postcondiciones		
Excepciones		

CU15	**CUSTOMIZE AVATAR**
Descripción	El usuario podrá personalizar su avatar cambiando las piezas que lo componen
Precondiciones	El usuario debe haber podido conectarse a una sala.
Flujo normal	
1.	El usuario abre el submenú ME.
2.	El usuario selecciona la pieza que quiere cambiar entre cabeza, cuerpo y manos.
3.	El sistema muestra las opciones diferentes para poder cambiarlo y una paleta de colores.
4.	El usuario selecciona los cambios.
5.	El sistema guarda los cambios en disco.
Variantes	
Extensiones	
Postcondiciones	El usuario tendrá la nueva apariencia en las próximas conexiones a la aplicación.
Excepciones	

5.4DIAGRAMA DE CLASES

En lo que respecta al Diagrama de Clases, en un primer lugar consideramos que el siguiente diagrama era representativo de las clases que íbamos a crear para la aplicación y que podrían cubrir todas las necesidades y requisitos (ver Diagrama 2).

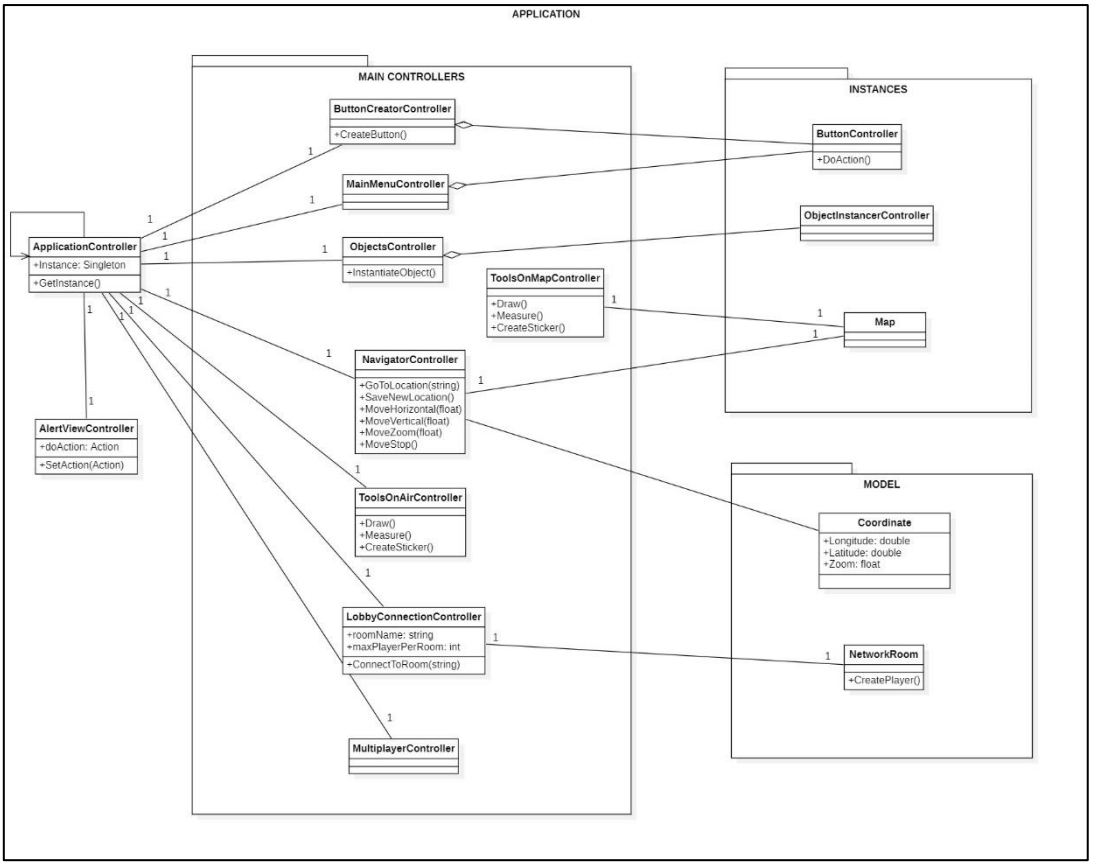

Diagrama 2.- Diagrama de Clases inicial

En él consideramos separar la aplicación en tres paquetes principales que contuviesen las demás clases. Estos tres paquetes serían formados por las clases del modelo, los principales controladores de la aplicación y los controladores de instancias. Además, fuera de estos tres paquetes se encontraban el controlador de la aplicación y un controlador especial para un pequeño panel de advertencia que nos sirve para darle conocimiento al usuario.

Con esto en mente la fase de desarrollo comenzó y tanto mi compañero como yo empezamos a desarrollar cada uno sus tareas en paralelo. Esto hizo que ambos hiciéramos cambios en el diagrama a lo largo de todos los sprints para estructurar mejor la aplicación. Por ejemplo, en el cuarto sprint decidí unificar ambas clases de herramientas y las dos que antes se llamaban *ToolsOnAirController* y *ToolsOnMapController* pasaron a ser una llamada *ToolsController*.

Finalmente, en el sexto sprint, cuando ya no se iban a hacer más cambios, decidimos unificar los dos nuevos diagramas que habíamos creado por separado dando lugar a uno en conjunto que representa las clases principales de la aplicación (ver Diagrama 3).

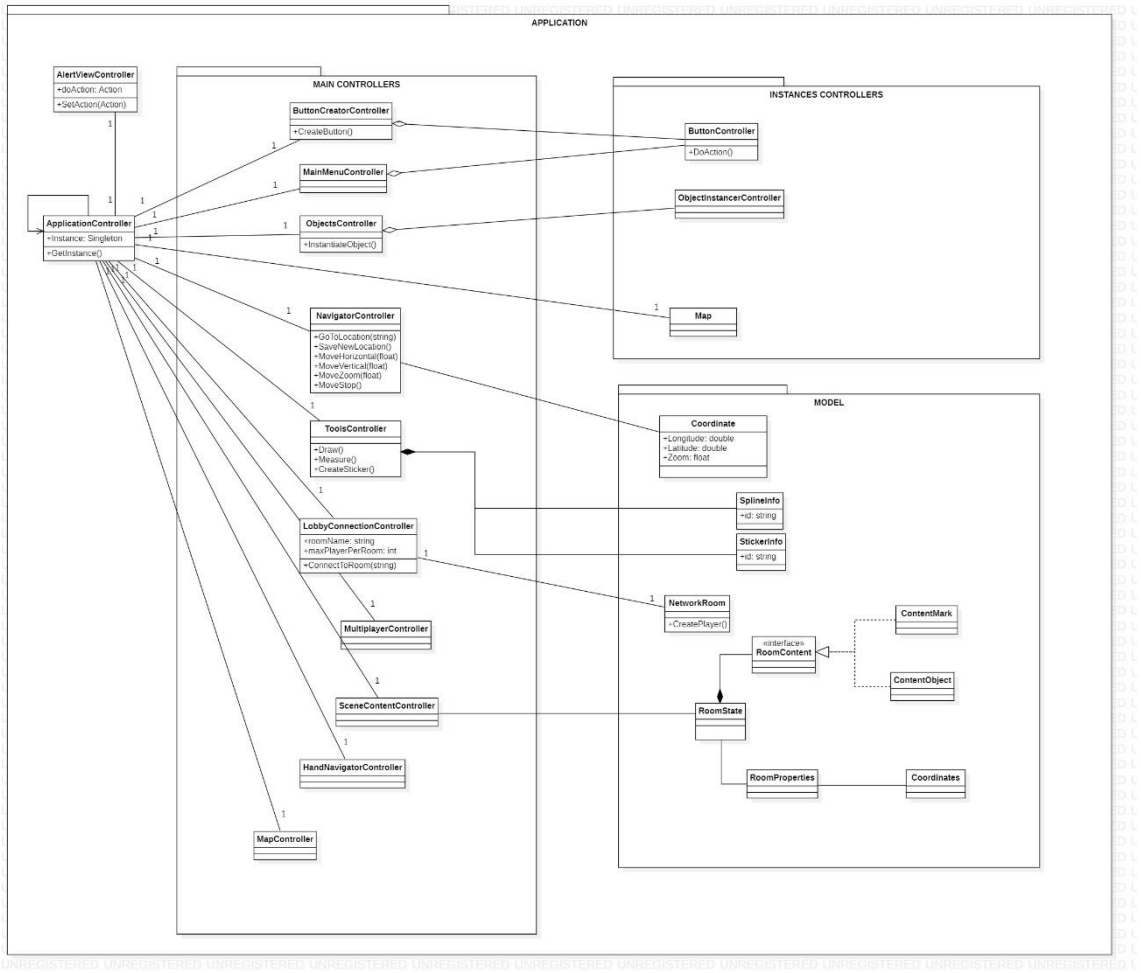

Diagrama 3.- Diagrama de Clases final

6. METODOLOGÍA ÁGIL

Como se comentó previamente en el apartado de la introducción, este proyecto se ha planificado y desarrollado siguiendo una metodología ágil siguiendo un desarrollo iterativo e incremental. Nos basamos sobre todo en las metodologías SCRUM (ver Figura 13) e intentamos en todo momento seguirla a medida de lo posible, pero en muchas ocasiones no fue posible debido a que el trabajo no fue desarrollado por un equipo de trabajo sino por una pareja y que, tanto mi compañero como yo no pudimos acatar al cien por cien los plazos establecidos por motivos laborales.

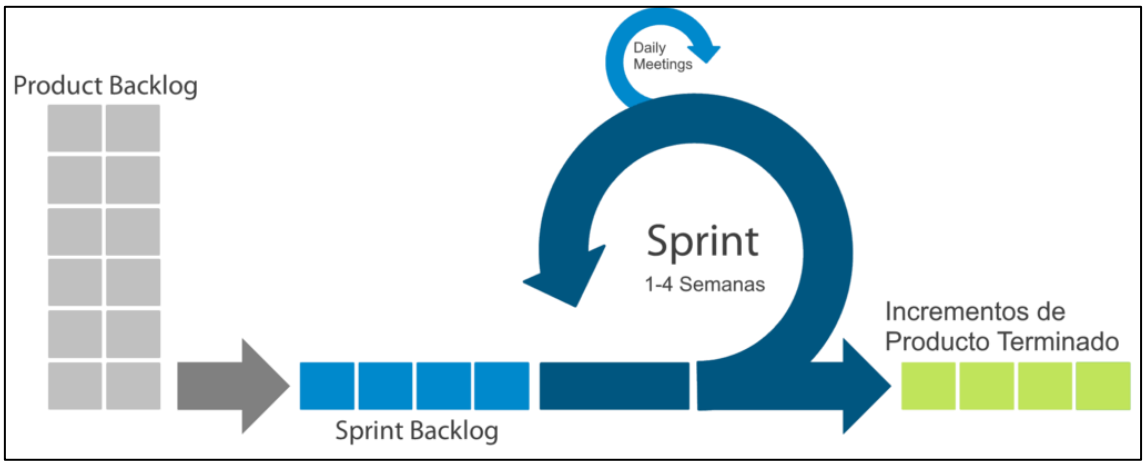

Figura 13.- Metodología SCRUM. Obtenida de https://www.iebschool.com/blog/metodologia-scrum-agile-scrum/

Pese a todo, procuramos establecer sprints de dos semanas en el que cada uno avanzaba en paralelo y al finalizar el sprint teníamos una reunión de retrospectiva entre nosotros dos donde fusionábamos todo lo que habíamos avanzado en esas dos semanas de forma que, la última versión estable tendría como mucho dos semanas de antigüedad. Además, cada dos sprints o cuatro semanas, teníamos una reunión con alguno de los tutores, bien académicos o de empresa, en la que enseñábamos los progresos de los últimos dos sprints como si estuviéramos teniendo una reunión con el Product Owner.

Para organizar el proyecto se utilizó el software de gestión de proyectos Trello. Trello utiliza tablones que se dividen en columnas de las cuales cuelgan las diferentes tarjetas que representan historias de usuario (ver Figura 14). En este tablón fuimos añadiendo historias de usuario y le asignábamos una prioridad o si correspondía a un requisito funcional o no funcional con un código de colores (ver Figura 15). Las columnas se corresponden de la siguiente manera:

- BACKLOG: Aquí se escribieron todas las historias de usuario para posteriormente ir moviéndolas a otras columnas.
- TO DO: En esta columna pasaban a estar las historias de usuario que pertenecían al sprint backlog, es decir, las que se iban a desarrollar en el sprint actual.
- IN PROCESS: Debajo de esta se encontraban aquellas historias de usuario que estaban siendo desarrolladas en ese mismo momento.

- TESTING: Aquí estaban situadas las historias de usuario que ya se habían desarrollado y estaban en fase de pruebas.
- BLOCKED: En esta columna se colocaban las historias que por algún motivo estaban bloqueadas, bien porque dependían de otras o porque no se iban a seguir desarrollando.
- DONE: En esta última estaban las historias de usuario que ya estaban completamente terminadas y probadas y que habían pasado todos los criterios de evaluación que se establecieron al principio.

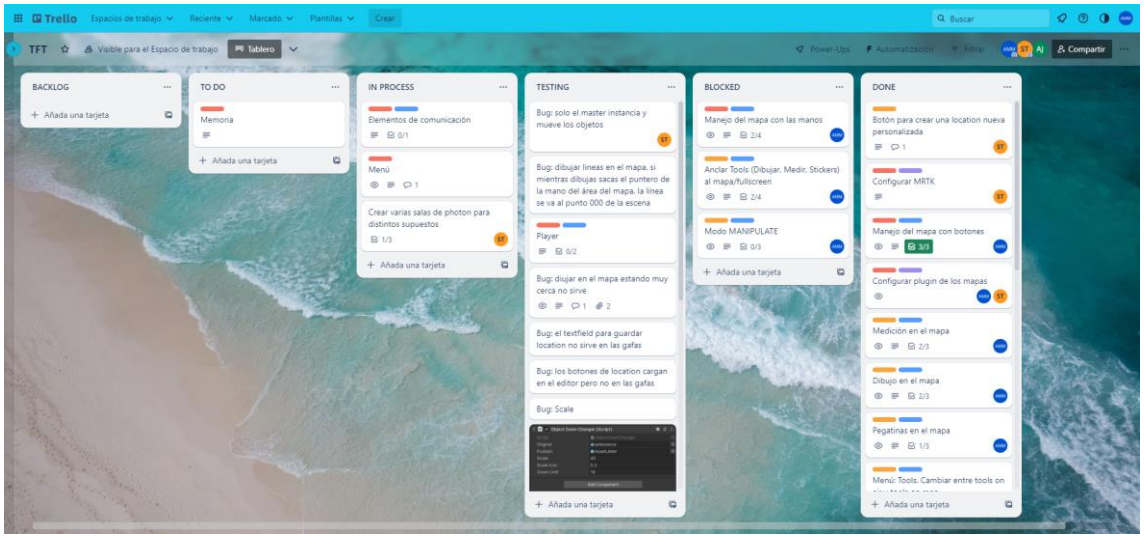

Figura 14.- Tablón Trello.

Cada dos semanas, en la reunión de retrospectiva que teníamos mi compañero y yo, aprovechábamos planificar el siguiente sprint y cambiábamos aquellas tarjetas que íbamos a desarrollar durante el siguiente sprint de la columna BACKLOG a la columna TO DO, y durante el desarrollo cada uno iba poniendo sus tarjetas bajo las columnas correspondientes.

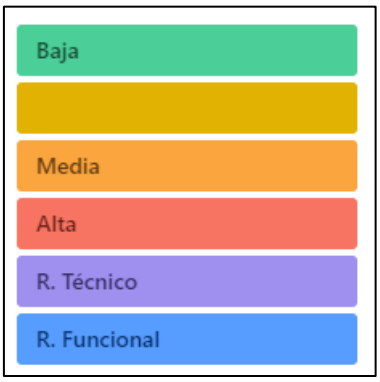

Figura 15.- Etiquetas de colores en Trello.

6.1PLANIFICACIÓN DE SPRINTS

Finalmente, terminamos el desarrollo de la aplicación en un total de seis sprints que, como explique anteriormente, fueron de entre dos y tres semanas de duración cada uno.

SPRINTS	TAREAS REALIZADAS DURANTE EL SPRINT
1.	• Fase de Análisis y de Diseño. • Instalación de Unity. • Instanciación del mapa en la escena. • Primeros controladores de la aplicación.
2.	• Instalación de paquetes externos (MRTK, Dreamteck Splines, Online Maps, etc.) • Implementación y configuración de las primeras conexiones multiusuario con Photon Engine. • Menú.
3.	• Herramienta de puntero. • Escala. • Resto de controladores del mapa (Navegación en el mapa). • Sistema de herramientas por modos. • Cambiar entre herramientas en el aire y en el mapa.
4.	• Herramienta de dibujo en el aire. • Herramienta de medir en el aire. • Herramienta de dibujo en el mapa. • Herramienta de medir en el mapa.
5.	• Herramienta de pegatinas en el aire. • Herramienta de pegatinas en el mapa. • Funcionalidad de *attach* para los dibujos y las mediciones en el aire. • Navegación en el mapa con la mano.
6.	• Avatar del usuario. • Conexión de voz. • Corrección de bugs.

De los seis sprints previamente descritos, el primero es el único que fue realizado en conjunto, es decir, en los otros cinco sprints están las tareas que yo realicé y no las que realizó mi compañero en paralelo. A pesar de ello, como al final de cada sprint nos poníamos de acuerdo para fusionar ambas ramas de trabajo en la rama de la cabecera del proyecto, ni mi compañero ni yo tuvimos ningún problema a la hora de necesitar algo que había desarrollado el otro integrante de la pareja.

7. DESARROLLO

En lo referente al desarrollo y como ya se ha comentado en varias ocasiones de este documento, existe una parte realizada en conjunto a mi compañero Samuel Trujillo Santana que consiste en la configuración del proyecto, la instanciación del mapa.

Además, existe una parte individual que ha realizado cada uno, en mi caso yo estaba más centrado en el desarrollo de herramientas de dibujo, medición y colocar pegatinas tanto en el mapa como en el aire, en las conexiones multiusuario y en otras funcionalidades que a medida que fuimos avanzando en el proyecto fueron surgiendo.

Al principio nos dedicamos los dos en conjunto a tener preparadas, en una escena y rama Git común todas las funcionalidades y configuraciones mencionadas en el primer párrafo. A partir de ahí cada uno, en diferentes ramas y escenas, fuimos desarrollando nuestras tareas.

Pese a que la gran mayoría del trabajo se lo llevan las herramientas en el aire y en el mapa, existen otras funcionalidades que son importantes de mencionar y que no se dividen en función de donde se realizan. En adelante explicaré todas estas funcionalidades que fui desarrollando a lo largo de este trabajo:

7.1CONEXIONES MULTIUSUARIO

En primer lugar, realicé todas las conexiones con el servidor de Photon Engine PUN 2 cuyas siglas provienen de Photon Unity Networking, es el producto de Photon Engine dedicado a proyectos con Unity para facilitar las conexiones multiusuario. Gracias a este producto que adquirimos en su versión gratuita fuimos capaces de crear las diferentes salas y realizar las conexiones entre los diferentes usuarios que se conectaban a las mismas.

La aplicación está conectada a una clave de aplicación, o *appkey*, única. Esta clave designa un vestíbulo, o *lobby*, y dentro de este vestíbulo se da acceso a las diferentes salas que la aplicación tenga. Nosotros en nuestra aplicación hemos seguido un patrón parecido puesto que al iniciar el usuario verá en un menú inicial las distintas salas a las que puede unirse, sin embargo, la conexión con el servidor no se realiza hasta que no se ha elegido una sala y es ahí donde entra en la propia sala de Photon.

Siguiendo el siguiente flujo, el usuario creará una sala siempre y cuando no esté ya creada una con el mismo nombre. Al crear una sala, ese usuario pasa a ser el anfitrión, o *master*, de la sala y es el encargado de mandar las ordenes al servidor para que se ejecuten todo tipo de Llamadas a Procedimientos Remotos (en inglés, Remote Procedure Call, RPC) que es una de las dos maneras, y la que utilizamos nosotros en nuestra aplicación, para comunicarse entre usuarios por el servidor Photon.

Una vez estaban hechas las conexiones cree un avatar (ver Figura 16) muy simple que consistía en una esfera y un cilindro que representasen la cabeza y el cuerpo del usuario que se había conectado a la sala junto con dos manos de color blanco. Este ‘muñeco’ tenía la cabeza enlazada a la cámara del usuario y las manos a cada uno de los controladores de las manos proporcionadas por el paquete MRTK de Microsoft para

desarrollar en HoloLens 2 y de esta forma cuando el usuario se movía en la realidad también lo hacía el muñeco y sus manos.

En esta primera versión decidimos dejarlo así, pero anotamos para futuras mejoras poder personalizar un poco el avatar de manera que cada usuario pueda elegir un color, escribir su nombre y hasta poder cambiar piezas del avatar como la cabeza, cuerpo y guantes.

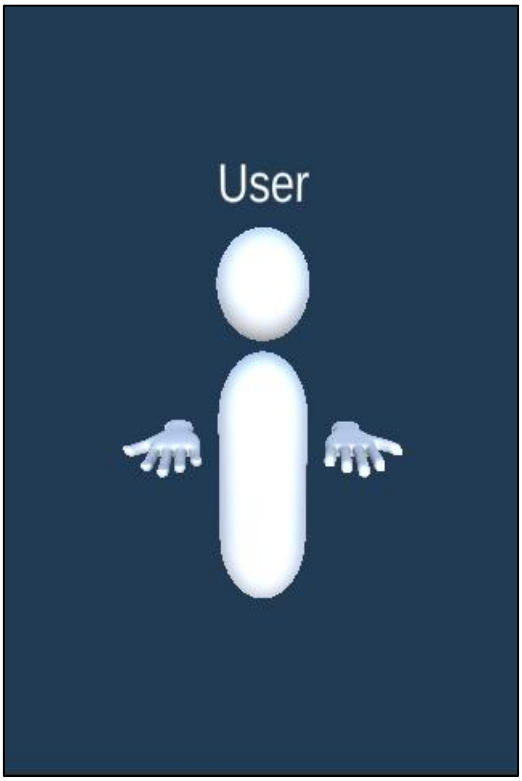

Figura 16.- Avatar del Usuario.

7.2MENÚ

Para poder manejarse en la app fue necesario desarrollar un menú principal (ver Figura 17) que activará el resto de las funcionalidades dentro de la aplicación. El menú consta de una lista de botones que componen el menú principal y que al ser pulsados abren un submenú correspondiente a cada apartado.

Para este menú tomé como referencia el que tenemos en la aplicación de la empresa y desarrollé una arquitectura parecida en el que existe un controlador del menú que se comunica en relación uno a muchos con los controladores de cada uno de los botones individuales. Y estos a su vez, también se relacionan uno a muchos con el resto de los controladores de los botones.

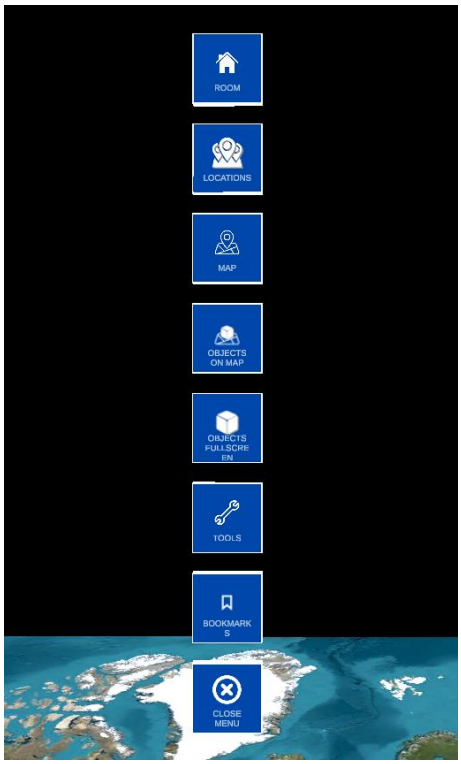

Figura 17.- Menú principal.

Esto se hizo así para que el controlador del menú tuviera completo acceso a todos los botones del menú y, de esta forma, poder acceder a ellos de manera más fácil. Los botones, a su vez, se controlan ellos mismos y únicamente se comunican con otros botones que dependen de ellos mismos, como es el caso de los botones que permiten cambiar el color de las líneas de dibujo que dependen del botón de dibujar para aparecer y desaparecer.

Los botones se pueden crear y modificar en el editor, es decir, en el espacio reservado por el motor Unity para preparar la escena y todo aquello que una vez se inicie la aplicación va a poder ver el usuario. O también se pueden crear y modificar en tiempo de ejecución por código. Entre los parámetros que permiten modificación existe la etiqueta con el nombre del botón, el icono y el color tanto por defecto como al pasar el puntero por encima y al estar seleccionado. El tipo de botón, que puede ser *click*, *hold* o *toggle*. La principal diferencia entre ellos es que el botón *click* invoca la función correspondiente cuando es pulsado; y los botones *hold* y *toggle* llaman a unas funciones al activarse y a otras al desactivarse, con la diferencia de que los de *hold* se activan al pulsarse y desactivan al dejarse de pulsar, y los de *toggle* se activan al pulsarse una vez y desactivan al pulsarse la siguiente vez. La gran mayoría de botones de la aplicación forman parte de este último conjunto.

7.3PUNTERO

Esta herramienta consiste en un rayo láser que aparece en el dedo índice del usuario (ver Figura 18) y se extiende en línea recta hasta el infinito. El principal objetivo de esta herramienta es señalar con la mano algo en el mapa o algún objeto para que el resto de

los usuarios que se encuentran en la sala sepan a lo que te refieres y puedan entenderte mejor.

El puntero en esta primera versión es de color blanco, sin embargo, dejamos anotado que para futuras versiones sería del mismo color que el avatar del usuario y es que notamos que en una reunión de muchas personas cuando varios usuarios tienen el puntero activado puede llegar a ser un poco confuso.

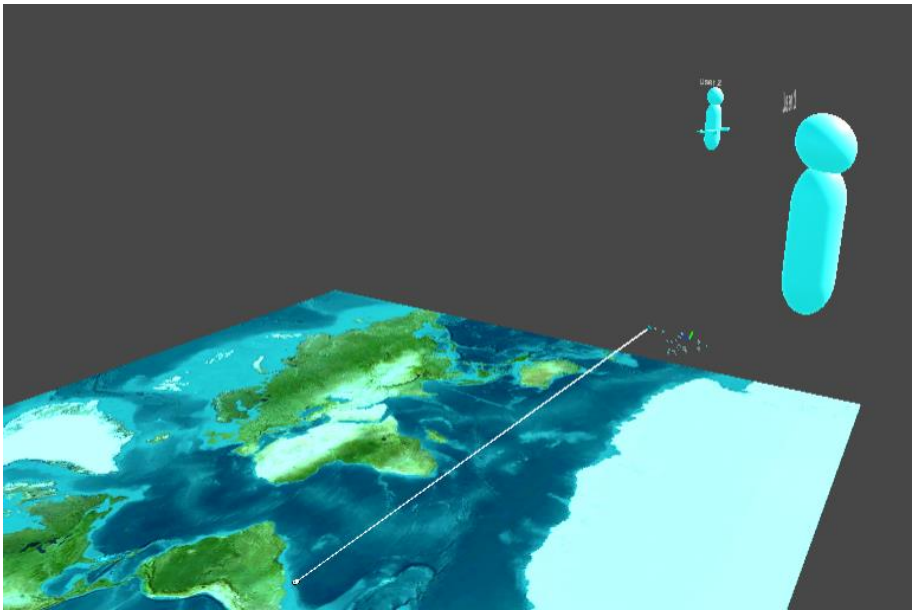

Figura 18.- Un usuario con el puntero activado y otro no.

7.4ESCALA

La herramienta de la escala está altamente ligada a la herramienta de medir tanto en el aire como en el mapa. Con este botón del menú el usuario puede decidir si ver las medidas que aparecen encima de las mediciones en una escala 1:1 o en una escala 1:X dónde X es un número calculado a partir de la siguiente fórmula matemática,

$$X = \frac{ZoomLvlScale}{nextZoomLvl - \ mapZoomFloat}$$

en la que *ZoomLvlScale* es la equivalencia de escala para cada nivel de zoom, este valor fue obtenido gracias a que la empresa disponía de un diccionario con las equivalencias (ver Figura 19) donde la clave del diccionario era el nivel del zoom y el valor la escala equivalente aproximada, *nextZoomLvl* que es el nivel de zoom siguiente al actual, es decir, si *mapZoomFloat* fuese 5.58 *nextZoomLvl* sería 6, y, por último, *mapZoomFloat* que es el zoom actual del mapa en número flotante. Gracias a esta fórmula la x varía en función al zoom del mapa en tiempo real y, por lo tanto, también varía el texto del botón.

```
zoomLevelList.Add(0f, 591657527.591555f);
zoomLevelList.Add(1f, 295828763.795777f);
zoomLevelList.Add(2f, 147914381.897889f);
zoomLevelList.Add(3f, 73957190.948944f);
zoomLevelList.Add(4f, 36978595.474472f);
zoomLevelList.Add(5f, 18489297.737236f);
zoomLevelList.Add(6f, 9244648.868618f);
zoomLevelList.Add(7f, 4622324.434309f);
zoomLevelList.Add(8f, 2311162.217155f);
zoomLevelList.Add(9f, 1155581.108577f);
zoomLevelList.Add(10f, 577790.554289f);
zoomLevelList.Add(11f, 288895.277144f);
zoomLevelList.Add(12f, 144447.638572f);
zoomLevelList.Add(13f, 72223.819286f);
zoomLevelList.Add(14f, 36111.909643f);
zoomLevelList.Add(15f, 18055.954822f);
zoomLevelList.Add(16f, 9027.977411f);
zoomLevelList.Add(17f, 4513.988705f);
zoomLevelList.Add(18f, 2256.994353f);
zoomLevelList.Add(19f, 1128.497176f);
zoomLevelList.Add(20f, 564.248588f);
```

Figura 19.- Diccionario nivel de zoom - valor de escala.

La finalidad de esta funcionalidad se podría resumir en que al pulsar el botón de la escala el usuario pueda ver, en las líneas de medición, la distancia que hay entre dos puntos tanto en el mundo de Unity como en la vida real entre dos puntos del mapa.

Cabe añadir que, en un primer lugar se intentó hacer la conversión de las mediciones entre ambas escalas de forma manual desarrollando una función que dada una distancia en escala 1:1 devolviese una distancia en escala 1:X. Sin embargo, tras estudiar la documentación de Online Maps, descubrí una función que proporcionaba el paquete previamente dicho la cual recibe como parámetros dos coordenadas geográficas y devuelve la distancia entre ellas.

Tras hacer varias pruebas de usabilidad llegué a la conclusión que no tenía mucho sentido ver las mediciones en el aire con escala 1:X puesto que el objetivo de estas mediciones no es medir distancias en el mapa sino en el aire por lo que impedí que las líneas de medición en el aire muestren su cota en escala 1:X y siempre la muestren en escala 1:1.

Finalmente, la funcionalidad de la escala cambia el texto del botón para indicar al usuario en que escala está midiendo y, en el caso de tener mediciones hechas en el mapa, también cambia su cota.

7.5CAMBIAR ENTRE HERRAMIENTAS EN EL AIRE Y EN EL MAPA.

En un principio, esta funcionalidad permitía que con un botón del menú se pudieran cambiar los tres botones que activan cada una de las herramientas del aire con los tres botones que cambian las herramientas del mapa. Esto fue simplemente por una cuestión estética para que en el menú no aparezcan los seis botones a la vez ya que son muy parecidos entre ellos.

Sin embargo, para esta funcionalidad hubo varios problemas de implementación con el resto del menú y esto es debido a que, al iniciar la aplicación, esta requiere que todos los botones estén activos y, como no quedaba bien que justo delante del usuario se desactivaran justo tres botones decidimos aparecer el menú fuera de la vista del usuario y que cuando este se mirase la palma de la mano el menú se pusiera justo delante de su vista. De esta forma se podía hacer que todos los botones estuvieran activos y se desactivaran por código antes de que el usuario conectase con la sala de Photon Engine.

Finalmente, al decidir usar el mismo script para las seis herramientas solo eran necesarios tres botones por lo que esta funcionalidad quedó descartada y el botón que la activaba ahora activa o desactiva un parámetro booleano del controlador de herramientas que indica si el usuario va a trabajar en el aire o en el mapa.

7.6LOOK AT YOUR HAND PANEL

Este panel simplemente es un mensaje que aparece delante del usuario una vez se conecta con el servidor de Photon Engine y entra en una de las diferentes salas para indicarle que para abrir el menú debe mirarse la palma de la mano (ver Figura 20). Una vez hace esto aparece un círculo que se rellena con el tiempo de forma radial y una vez está completo, un menú inicial que permite elegir entre diferentes salas pasa a estar delante del usuario.

Esta funcionalidad puede parecer redundante pero realmente es muy importante ya que dota a la aplicación de varios segundos desde que el usuario ha entrado a la sala y hasta que el menú aparece delante suya para realizar todos los arreglos comentados en el anterior apartado.

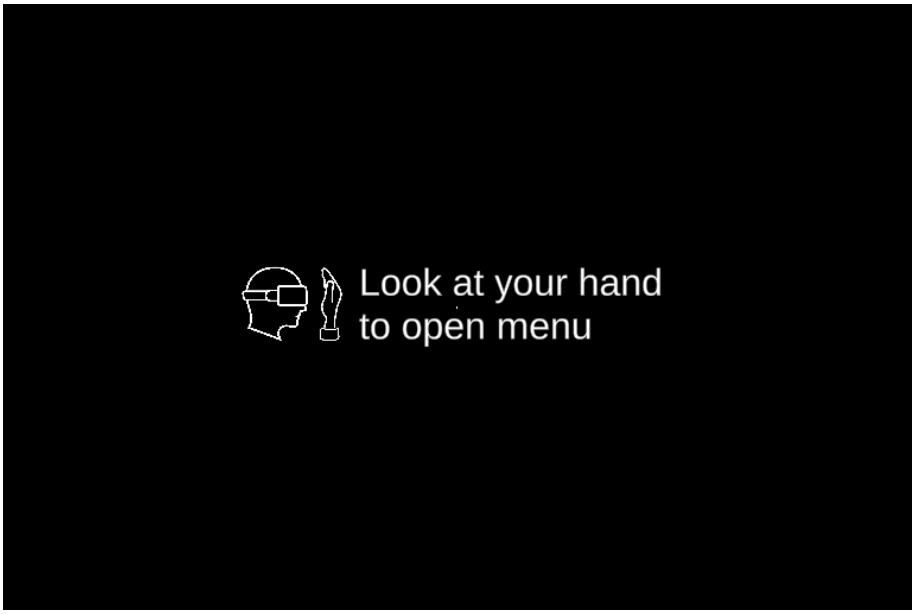

Figura 20.- LookAtYourHand Panel.

7.7NAVEGACIÓN POR EL MAPA

Esta funcionalidad permite al usuario desplegar un submenú con ocho botones (ver Figura 21), cuatro que permiten moverse por el mapa hacia el norte, sur, este y oeste

respectivamente. Otros dos que permiten hacer zoom in y zoom out y otros dos que mueven el mapa verticalmente.

Todos estos botones son de tipo *hold*, es decir, mientras los mantienes pulsado ejecutan su llamada. Esto permite darle un valor numérico a la función en cuestión que llamen y sumarlo o restarlo al valor actual que contenga el mapa. Por ejemplo, sumarle 0,5 a la coordenada de latitud permite viajar al norte, y restarle, al sur. Esto funciona de igual forma para el zoom y para el desplazamiento vertical del mapa.

Al ser funciones que se ejecutan mientras un botón está siendo pulsado, el sistema está continuamente calculando la coordenada de destino y la está enviando con un pequeño retraso de menos de un segundo al resto de usuarios conectados en la sala. Cuando el botón deja de ser pulsado, ya una vez el sistema ha calculado una coordenada final, los controladores de los usuarios receptores tienen un tiempo de gracia para seguir moviendo el mapa hacia la dirección o zoom concreto.

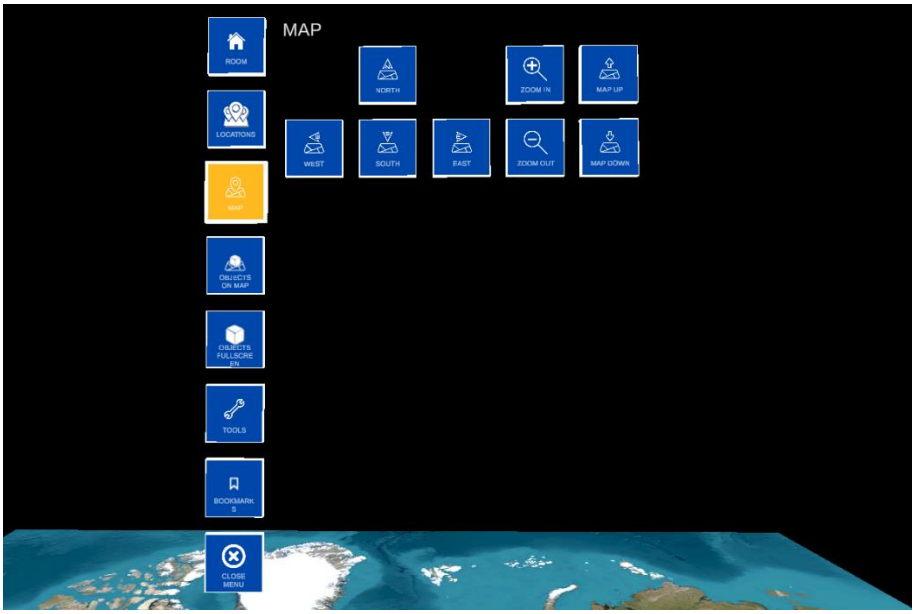

Figura 21.- Submenú Map.

7.8SISTEMA DE HERRAMIENTAS POR MODOS

Para desarrollar un sistema en el que, utilizando los mismos gestos, el usuario pudiera realizar diferentes acciones se me ocurrió un sistema por lo que yo denomine “Modos”. Estos modos serian exclusivos entre sí, es decir, al activar uno desactivas los demás y estos son activados mediante los botones en el menú.

En un primer lugar iban a ser seis modos: tres correspondientes a las herramientas en el aire y otros tres para las herramientas en el mapa. Sin embargo, a efectos prácticos estos seis eran realmente tres: Modo Dibujo, Modo Medición y Modo Pegatinas, solo que se diferenciaban en donde se colocarían. Tras pensarlo de nuevo decidí que fueran solamente estos tres y no los seis que iba a poner en un principio y que trabajasen en conjunto con un parámetro booleano que indicase si el lienzo era el aire o el mapa. Finalmente añadí el Modo Manipulación que explicaré más adelante.

7.9HERRAMIENTAS EN EL AIRE

La peculiaridad de estas herramientas (dibujar, medir y colocar pegatinas) es que se instancian en la punta de tus dedos una vez juntas los dedos índices y pulgar siempre y cuando tengas seleccionado en el menú la opción de la herramienta correspondiente. Para ello, coloqué un prisma cuadrangular en la cabeza del usuario y si el punto de colisión entre los dedos se encuentra dentro de este cubo se llevará a cabo la previamente elegida herramienta.

Para añadir más variedad a las líneas de dibujar y medir, al seleccionar estos botones en el menú aparecen otra serie de botones que permiten al usuario elegir entre dos tipos diferentes de líneas (continua o discontinua), seis diferentes colores (verde, amarillo, rojo, azul, rosa y blanco) y cambiar el grosor de la línea dentro de unos límites. Las pegatinas por su parte solamente permiten cambiar el color y elegir entre siete diferentes pegatinas cual quieres colocar.

Tanto dibujar como medir fueron posibles gracias al paquete Dreamteck Splines que fue facilitado por la empresa, mientras que las pegatinas fueron realizadas con objetos primitivos de Unity e imágenes en formato PNG.

7.9.1 DIBUJAR.

Como ya expliqué previamente el desarrollo de dibujar consiste en comprobar si, primero, está seleccionada la opción de dibujar y, segundo, los dedos índice y pulgar se están tocando, haciendo alusión a cuando escribes en una pizarra con una tiza. Si se cumplen ambas condiciones, un script se dedica a recoger los puntos en el aire por donde pasa el dedo índice y mediante un método del paquete Dreamteck Splines se puede crear una línea que atraviese todos esos puntos uno detrás de otro (ver Figura 22).

El funcionamiento de la línea discontinua es el mismo salvo que al crear el propio *spline* se crea con un material que acepta una textura. Esta textura es una imagen de cuatro puntos que después es modificada en el eje horizontal para que se repitan uno detrás de otro y formen así una línea de puntos.

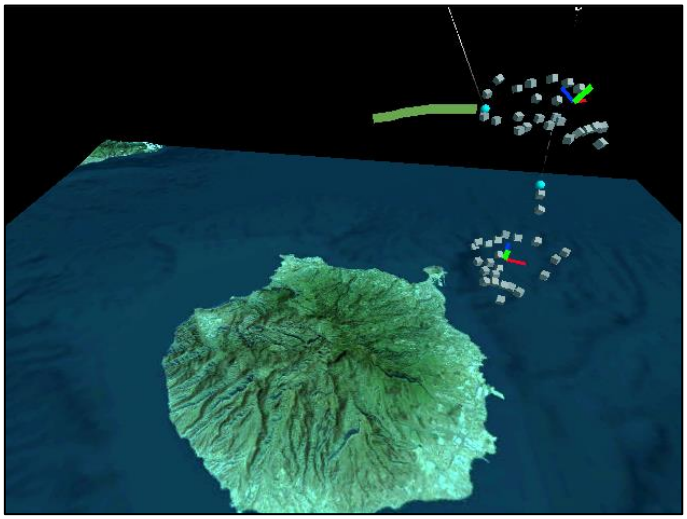

Figura 22.- Usuario dibujando en el aire.

7.9.2 MEDIR

La herramienta de medir se comporta de manera muy similar a la de dibujar salvo por unos pocos cambios. En primer lugar, no recoge todos los puntos del mundo por el que pasan el dedo índice sino solamente dos, en el que se juntan los dedos y en el que se separan. En estos dos puntos se crean dos cubos, haciendo alusión a una cota. Por encima de la línea y en el punto intermedio entre ambos cubos se crea un texto que indica la distancia entre los mismos, o lo que es lo mismo, lo que mide la línea (ver Figura 23).

Esta línea también se puede cambiar de color, grosor y patrón de la misma forma que con la herramienta de dibujar.

Figura 23.- Usuario midiendo el largo del activo de una ambulancia.

7.9.3 COLOCAR PEGATINAS

Para la herramienta de las pegatinas seguí un procedimiento parecido al de ambas líneas previamente explicadas salvo por el detalle que al arrastrar la mano por delante de ti en lugar de pintar cambias el tamaño de la pegatina (ver Figura 24) y esta se establece en el punto en donde juntaste tus dedos previamente.

Además, en el caso de las pegatinas no aparece el submenú con el que se puede cambiar el tamaño en las otras herramientas, sin embargo, aparece otro submenú con el que elegir la pegatina que quieres colocar en el aire de entre unas predeterminadas (√, X, O, ←, ↑, →, ↓).

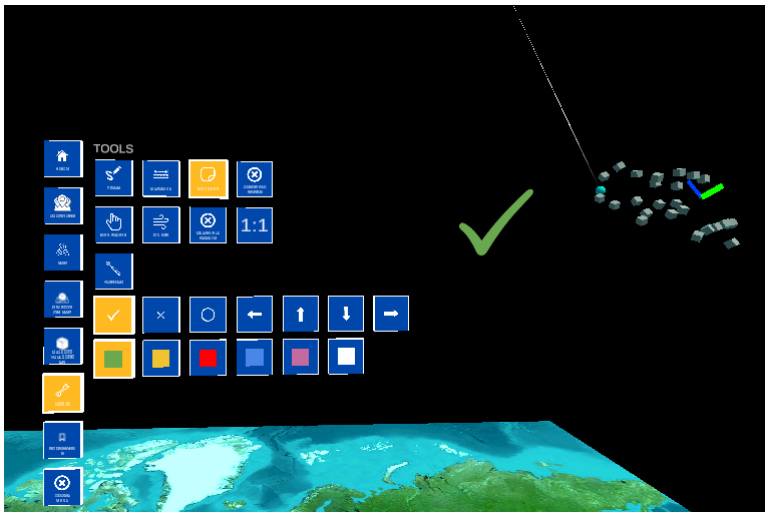

Figura 24.- Submenú Tools con las pegatinas activas y un Usuario poniendo una de ellas.

7.10 HERRAMIENTAS EN EL MAPA

Al desarrollar estas herramientas, como son las mismas que en el aire, pero en lugar de quedar flotando se enganchan al mapa y quedan ancladas al mismo, en una primera instancia, se valoró adaptar el mismo script usado para las herramientas en el aire para estas herramientas pudiendo así ahorrar tiempo y esfuerzo. Tras un primer intento se llegó a la conclusión que no era tan buena idea e iba a acarrear mucho más tiempo de adaptar el script para poder funcionar correctamente con ambos 'lienzos' (aire y mapa). Sin embargo, tras comentarlo con mi compañero y el tutor de empresa, se tomó la decisión de realizarlo tal y como en primera instancia comenté y se adaptó el script de las herramientas en el aire para que también manejase las herramientas en el mapa.

A pesar de ello, existen algunas diferencias que cabe mencionar. La más notoria es que al hacer el gesto de pinza con los dedos para dibujar, medir o colocar pegatinas, estas se instancian en el punto en el que un rayo que aparece de la punta de los dedos del usuario colisiona con el mapa.

7.10.1 DIBUJAR.

El factor diferencial que incluyen estas líneas en comparación con las del aire es que al quedarse ancladas en el mapa quedan ancladas tanto en tamaño y escala, por lo que si se dibuja algo y posteriormente el usuario hace zoom o se desplaza al norte el dibujo quedará en el mismo sitio y con un tamaño proporcional a como se realizó (ver Figura 25).

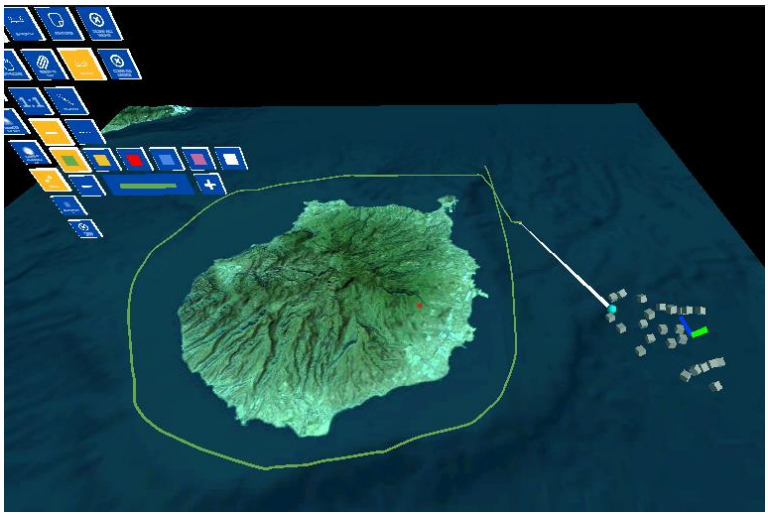

Figura 25.- Usuario dibujando en el mapa.

7.10.2 MEDIR.

Medir en el mapa funciona exactamente igual que en el aire, salvo que, como se mencionó anteriormente la línea y los cubos quedan anclados al mapa (ver Figura 26).

Figura 26.- Usuario midiendo en el mapa.

7.10.3 COLOCAR PEGATINAS.

El caso de las pegatinas si es un poco diferente con respecto a las pegatinas en el aire. A diferencia de aquellas que se quedan en el aire flotando, las pegatinas en el mapa se instancian al juntar los dedos apuntando al mapa y al mover la mano mientras se mantienen los dedos juntos la pegatina se mueve junto al rayo. Al despegar los dedos, esta queda anclada a la posición en la que estaba y tendrá siempre el mismo tamaño, haciendo alusión a una chincheta de Google Maps, que por mucho que te acerques o alejes en el mapa la chincheta tiene el mismo tamaño. Además, esta pegatina tiene un soporte que indica exactamente donde se encuentra la misma evitando de esta forma que quede flotando sobre el mapa y pueda causar confusión en lo que respecta a su posición exacta (ver Figura 27).

Cabe mencionar que para que todo esto funcione para varios usuarios simultáneamente, el master de la sala comunica al servidor por RPC que se cree, ya bien sea una línea o una pegatina, en la posición del mundo de Unity o del mapa, con el color y tamaño correspondiente y el servidor se lo indica a todos los demás usuarios simultáneamente.

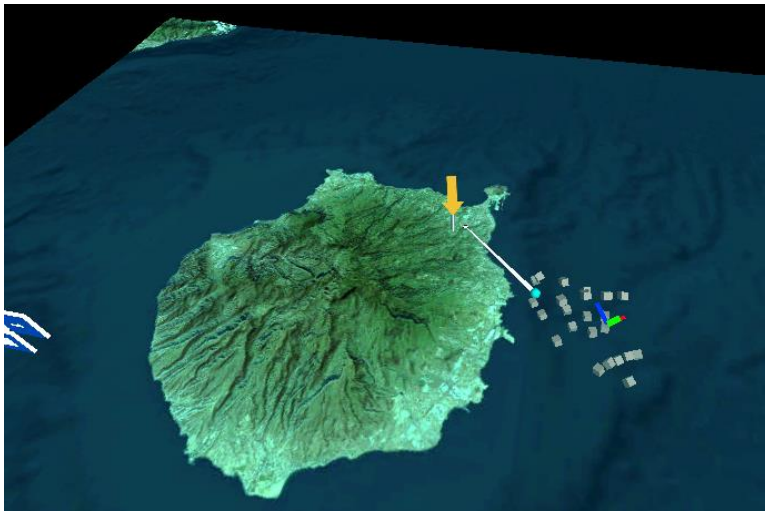

Figura 27.- Usuario poniendo una pegatina en el mapa.

7.11 MODO MANIPULATE

El Modo Manipulate (de aquí en adelante Modo Manipulación) es una funcionalidad que implementé para que las diferentes herramientas tanto en el aire como en el mapa puedan funcionar en sincronía con la parte que hizo mi compañero (ver Figura 28). Esto surgió ya que al estar usando la aplicación nos dimos cuenta de que existía la posibilidad de poner un activo en el mapa, por ejemplo, un coche, y activar en el menú el Modo Dibujar. De esta forma al agarrar el coche e ir a moverlo, el usuario empezaba a dibujar a la vez que movía el coche a otra localización. Esto ocurre ya que la forma en la que se ejecutan las acciones de dibujar y de manipular un activo es la misma, juntar los dedos índice y pulgar como si se pellizcase el aire.

Añadir este nuevo Modo Manipulación era una forma de indicarle a la aplicación que no quería dibujar, sino que quería manipular los activos. En un principio, siempre que la aplicación se encuentre en este modo aparecerá una caja de color naranja alrededor del activo en cuestión cuando el puntero del usuario pase por encima del mismo. Y, por contraparte, si la aplicación se encuentra en Modo Dibujar o Modo Medir, esta caja naranja no aparecerá y por lo tanto no se podrá mover ni rotar ni escalar el activo en cuestión. Sin embargo, la caja naranja fue descartada por mi compañero y es que, al instanciar activos en el mapa, la caja era mucho más grande que el propio activo y solamente se apreciaba el cubo y no el activo en sí.

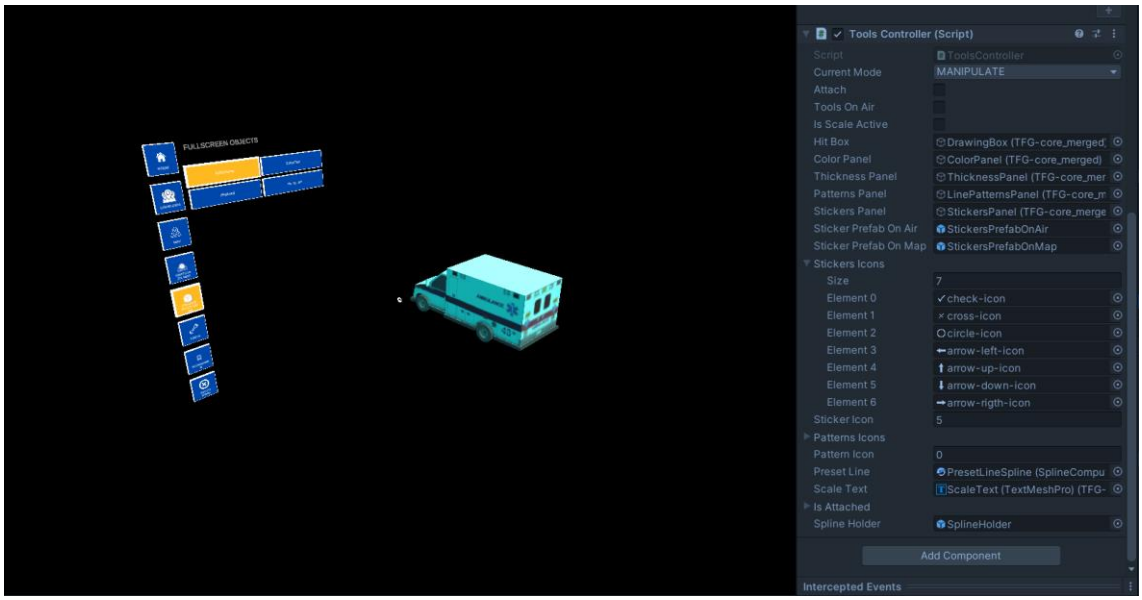

Figura 28.- Usuario instanciando un activo y se activa el Modo Manipulación.

7.12 FUNCIÓN ATTACH

La función *attach* o anclaje es una funcionalidad que no estaba en los planes iniciales y que fue propuesta por el tutor académico. Esta funcionalidad consiste en poder anclar las líneas que quedan flotando en el aire a una posición del mapa para que, por ejemplo, si el usuario dibuja o mide en el aire por encima de una localización, al moverte por el mapa o hacer zoom esta línea varie su posición y/o tamaño en relación con cómo estaba cuando se dibujó (ver Figura 29).

Por la propia naturaleza de la funcionalidad no tiene sentido que se pueda activar esta característica para las líneas dibujadas en el mapa y es por ello por lo que cuando en el menú se pulsa en el botón que cambia entre el lienzo de trabajo, el botón de anclaje queda oculto. Lo mismo ocurre para el caso de colocar pegatinas y es que tras varios intentos de colocar pegatinas en el aire que queden ancladas el resultado no fue el esperado y quedó apuntado y notificado como futura mejora.

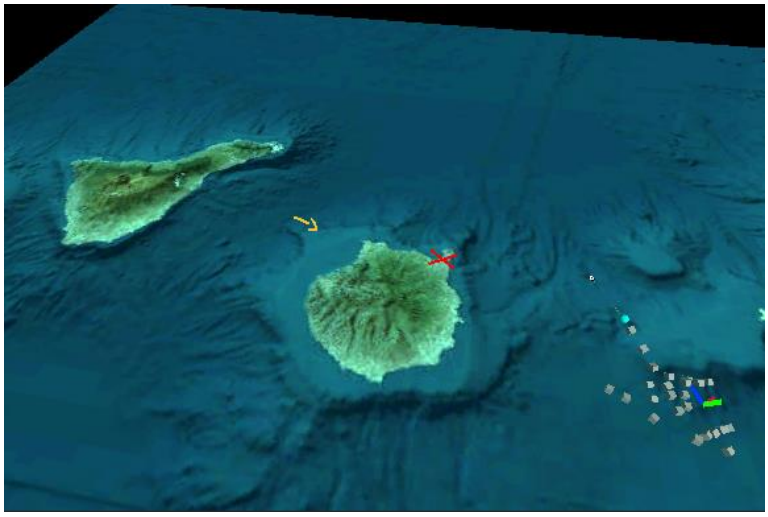

Figura 29.- Líneas ancladas en el aire al mapa.

7.13 HAND NAVIGATOR CONTROLLER

La funcionalidad del *HandNavigatorController* se nos ocurrió a Samuel y a mí mientras desarrollábamos otra de las herramientas y es que nos dimos cuenta de que sería, además de muy útil para la manejabilidad de la aplicación, muy intuitivo para el usuario poder mover el mapa con las manos al pulsar en el mismo.

Es por ello por lo que comencé a desarrollar esta funcionalidad en la que el usuario era capaz de moverse por el mapa siempre y cuando ninguna de las herramientas estuviera seleccionada si pinchaba el mapa con una de las dos manos y hacer zoom in y zoom out al pinchar el mapa con ambas.

Pesé a todas las pruebas que hice, finalmente se descartó esta opción ya que no quedaba del todo fluido y generaba bastantes bugs. Quedó anotada y documentada para próximas mejoras.

7.14 PERSONALIZACIÓN DEL AVATAR

El desarrollo de este caso de usuario nunca se comenzó por motivos de priorización y de tiempo en lo que respecta a la entrega del proyecto. Pese a todo, la intención era permitir al usuario personalizar cada una de las tres partes que componen el avatar.

De la forma que estaba enfocado era, en esta primera versión de la aplicación sería elegir un color que sería común para todas las partes y que fuese exclusivo para un usuario en una sala ya que estas estaban capadas a cuatro usuarios simultáneos. Este color también cambiaría el color del rayo que sale del puntero.

Sin embargo, también pensamos que, en una futura versión de la aplicación, esta podría conectarse con un portal web donde los usuarios pudieran elegir diferentes modelos y hacer sus propias combinaciones como si de un armario se tratase.

8. PRUEBAS

En lo referente a las pruebas, todas las funcionalidades fueron en un primer lugar probadas en un entorno local en el motor Unity ya que permite ejecutar la aplicación sin tener que compilar una versión. Para ello se introdujeron varios puentes en el código que permitían al usuario interactuar con la aplicación como, por ejemplo, el botón principal y secundario del ratón permitían juntar los dedos índice y pulgar de las manos y girar la cabeza respectivamente; las teclas W, A, S y D permitían al usuario moverse hacia adelante, izquierda, detrás y derecha respectivamente; las teclas de la barra espaciadora y las mayúsculas izquierda enseñaban y ocultaban las manos derecha e izquierda respectivamente; y la tecla 'C' permitía abrir el menú sin necesidad de mirarse la mano (ver Figura 30).

```
    Mensaje de Unity | 0 referencias
    private void Update()
    {
#if UNITY_EDITOR
        if (Input.GetKeyDown(KeyCode.C))
        {
            menuContent.GetComponent<HandMenuInitializer>().SetVisible();
        }
#endif
    }
```

Figura 30.- Extracto de código para abrir el menú en el editor de Unity.

Retomando lo que comentaba anteriormente, una vez se desarrollaba una funcionalidad se realizaban las pertinentes pruebas en local. Habiendo pasado esta primera fase de pruebas se comprobaba que funcionase con diferentes usuarios. Para ello o bien nos conectábamos desde el entorno de Unity mi compañero y yo en la misma sala o lo comprobaba yo mismo con dos clientes de Unity abiertos simultáneamente (ver Figura 31).

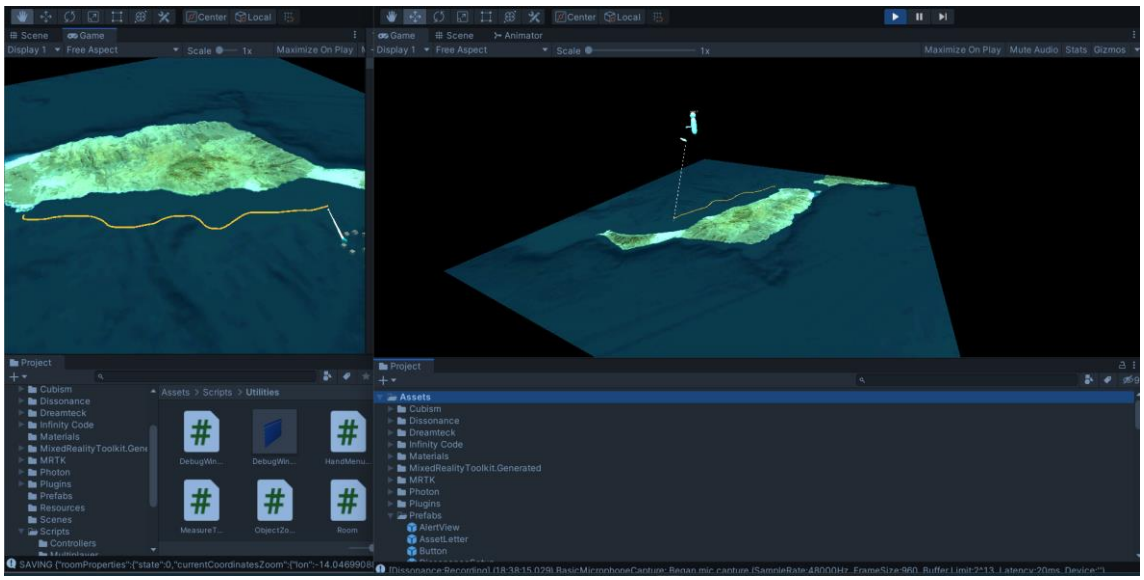

Figura 31.- Dos clientes de Unity abiertos simultáneamente.

Por último, en una tercera fase, se compilaba una versión de la aplicación y se probaba en las gafas tanto un solo usuario como múltiples usuarios (ver Figura 32). Lamentablemente solo contábamos con un juego de gafas por lo que no se podían conectar diferentes personas con las gafas para probar, pero sí que comprobábamos, estando mi compañero con las gafas y yo con el ordenador que todo funcionase bien en la aplicación compilada dentro de las HoloLens 2.

Figura 32.- Captura de la aplicación en las gafas.

9. CONCLUSIÓN

Durante la realización de este proyecto he conseguido aprender y desarrollar nuevas tecnologías que nunca había podido practicar durante mi etapa como estudiante pero que, gracias a ella, he conseguido unos resultados que considero más que buenos. Considero este hecho algo que me define como una persona que nunca está satisfecha en su zona de confort y que siempre intenta aprender cosas nuevas y ser más ambicioso, en este caso, desarrollar una aplicación en realidad mixta multiusuario usando tecnologías y patrones de desarrollo que no había visto previamente.

Aun así, a lo largo del desarrollo de este proyecto también he podido utilizar muchas otras técnicas de desarrollo que estudie en la carrera, pero que por motivos lógicos no todas fueron desarrolladas como se merecen. Este proyecto me ha dado la visión de forma más practica necesaria para adquirir por completo los conocimientos sobre estas técnicas.

Considero que este proyecto, junto con el trabajo a diario con la empresa que dio idea para el mismo, XReality Factory S.L., a la que le estoy muy agradecido, ha contribuido enormemente en mi desarrollo como ingeniero de software a lo largo de este último curso académico.

9.1MEJORAS FUTURAS

Para finalizar, consideramos que este proyecto es un prototipo de lo que podría ser una aplicación revolucionaria en diferentes sectores de los que se hablará en el siguiente apartado. Entre las diferentes mejoras que se podrían desarrollar en un futuro destacan:

- Sistema de usuarios con diferentes permisos.
- Sistema de inicio de sesión.
- Personalización del avatar.
- Portal web donde cada usuario, en función de su permiso, pueda realizar diferentes acciones como, por ejemplo, subir nuevos modelos 3D, crear diferentes salas o invitar a otros usuarios vía email.
- Aumentar el número de usuarios por sala.
- Soporte para diferentes formatos que se puedan visualizar en un entorno en realidad mixta como pueden ser vídeos, presentaciones de diapositivas.
- Soporte para visualizar capas ArcGIS.

9.2REPERCUSIÓN DEL TRABAJO

En cuanto a sus posibles aplicaciones al mundo laboral, la aplicación podría ser una buena herramienta para empresas de transporte y otro tipo de logística, con las que podrán visualizar nuevas rutas y calcular las más óptimas; ejércitos nacionales, que pueden visualizar el terreno previo a una misión o para formar a los nuevos soldados en las maniobras; en entornos de rodajes de series o películas se puede no solo visualizar el terreno donde querer grabar sino que, gracias a la herramienta para colocar objetos puedes incluso crear un pequeño boceto del set que finalmente necesitas para una toma en concreto.

Pero sin duda, en lo que creemos que más puede destacar es en el apartado de gestión de emergencias y es que estas, en muchas ocasiones, no avisan ni se pueden prever y el tiempo en el que se toman las decisiones y se comunican puede ser crucial para salvar vidas humanas. Con esta aplicación este paso intermedio se puede obviar ya que las personas que toman las decisiones en una sala de gestión de emergencias pueden estar conectados con las personas que se encuentran a pie de campo para intentar paliarlas y tener una comunicación e intercambio de información de forma más fluida.

Por ejemplo, en una situación como la de la erupción volcánica de La Palma en el pasado 2021, la aplicación podría conectarse con drones que sobrevuelen la colada de lava y aquellos que controlan la emergencia junto con el equipo de bomberos podrían ver las imágenes de los drones y tomar decisiones en función de estas estando los bomberos ya preparados cerca de la propia colada para una vez se toma la decisión actuar de inmediato.

ANEXO I: MANUAL DE USUARIO

1. ENTRADA DEL USUARIO

Al iniciar la aplicación el usuario verá delante de sus ojos las diferentes salas a las que puede entrar (ver Figura 33).

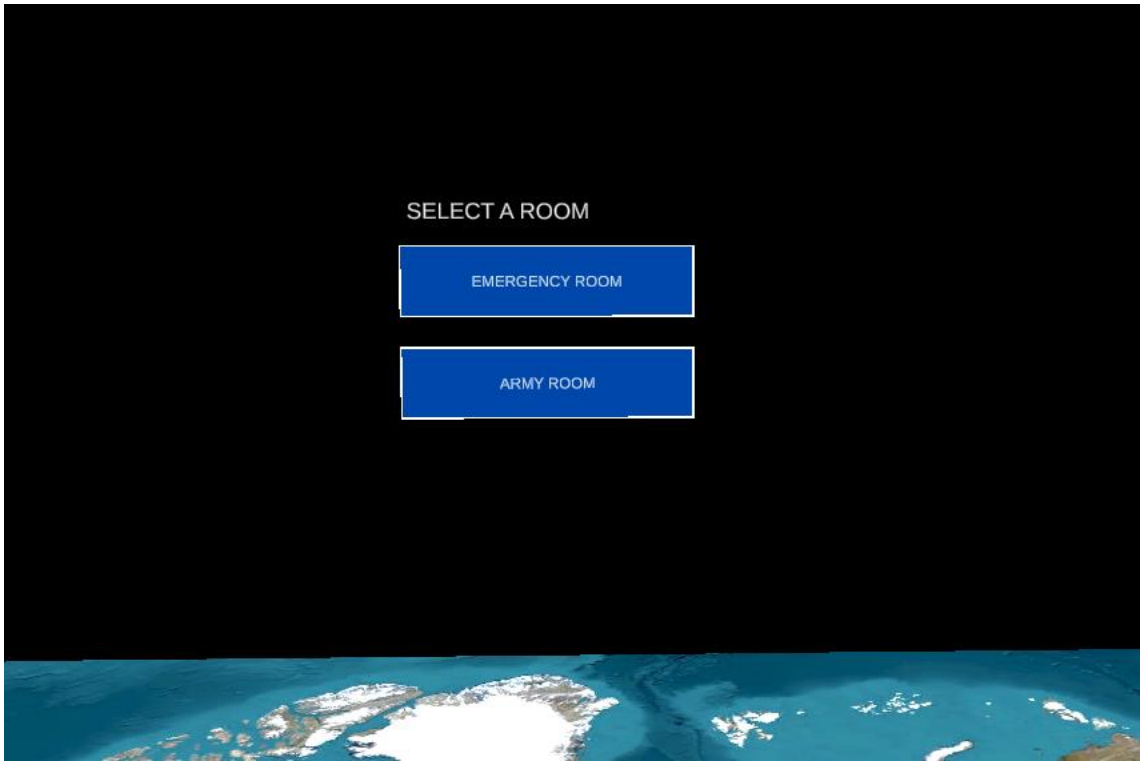

Figura 33.- Menú de selección de salas inicial.

Al pulsar con el dedo una de ellas, entrará y deberá esperar unos segundos hasta que un panel aparezca delante de su vista que le indica como abrir el menú principal (ver Figura 20). Es en este momento cuando el usuario está dentro de la sala.

Para abrir el menú principal el usuario debe poner su mano delante de su cara de forma que mire a la palma de su mano.

2. ACTIVAR AUDIO

El usuario puede activar y desactivar el audio de la aplicación y su propio micrófono, en función de si quiere escuchar y/o que le escuchen el resto de los usuarios (ver Figura 34).

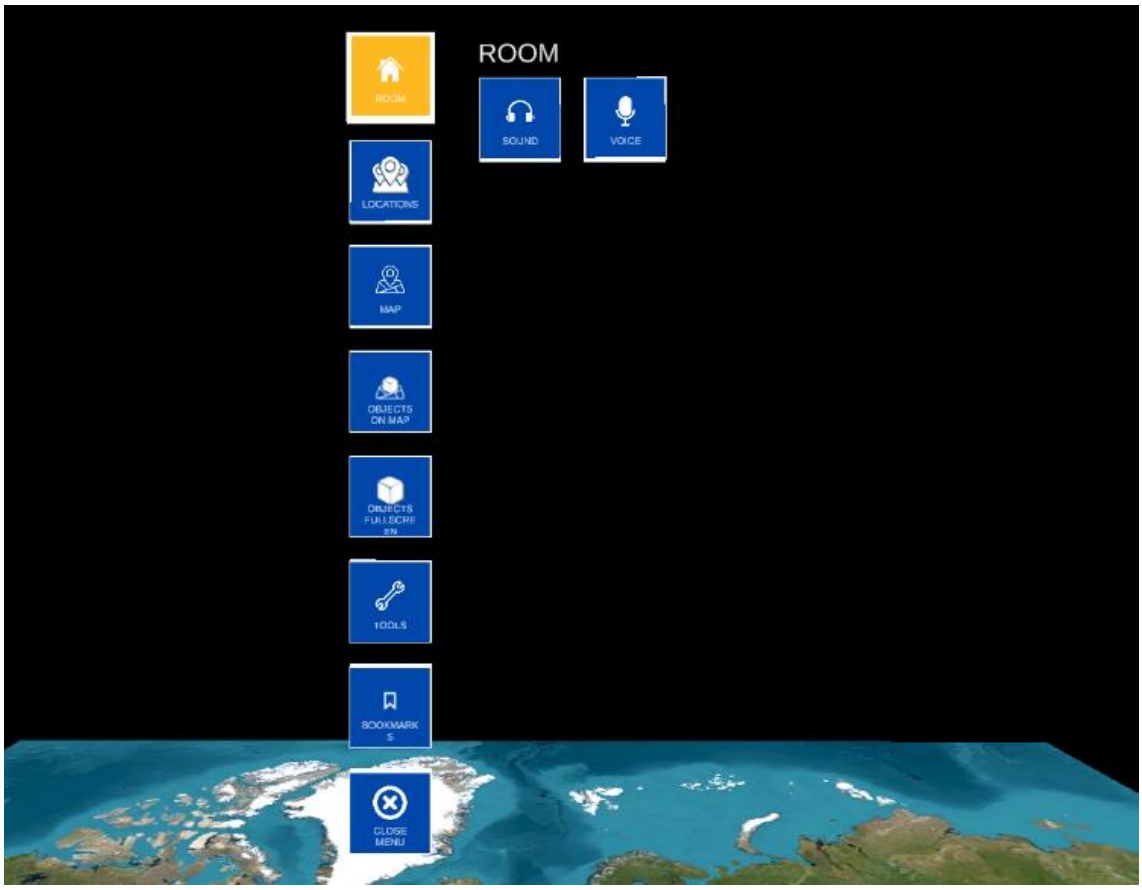

Figura 34.- Submenú Room.

3. IR A UNA LOCALIZACIÓN

Existen tres formas de ir a una localización:

- La primera consiste en abrir el submenú MAP y con los cuatro botones de dirección y los dos de hacer zoom el usuario puede moverse hasta la localización del mapa que desee (ver Figura 21).
- Para la segunda el usuario deberá abrir el submenú LOCATIONS y se mostrarán aquellas localizaciones previamente guardadas junto con algunas que ya trae por defecto la propia aplicación (ver Figura 35).

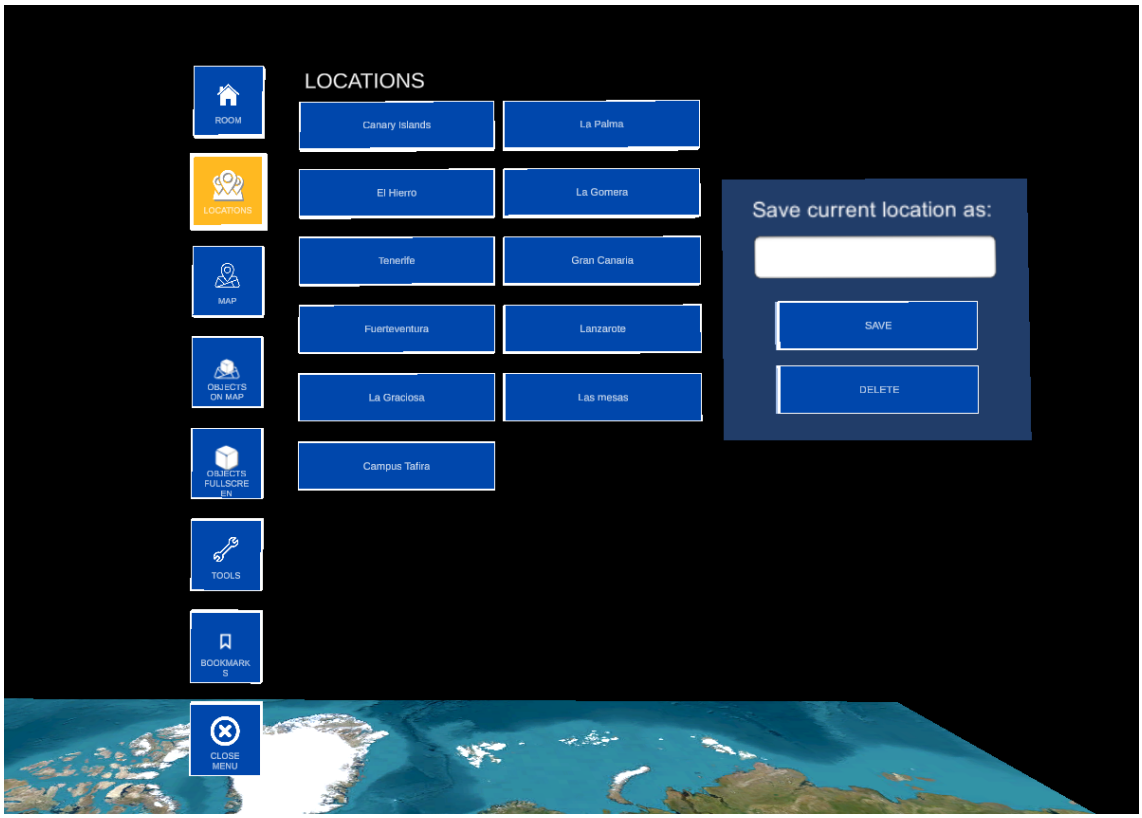

Figura 35.- Submenú Locations.

- La tercera trata de pedirle a otro usuario conectado en la sala que haga una de las dos anteriores. Entonces el mapa se moverá para todos los usuarios conectados (ver Figura 36).

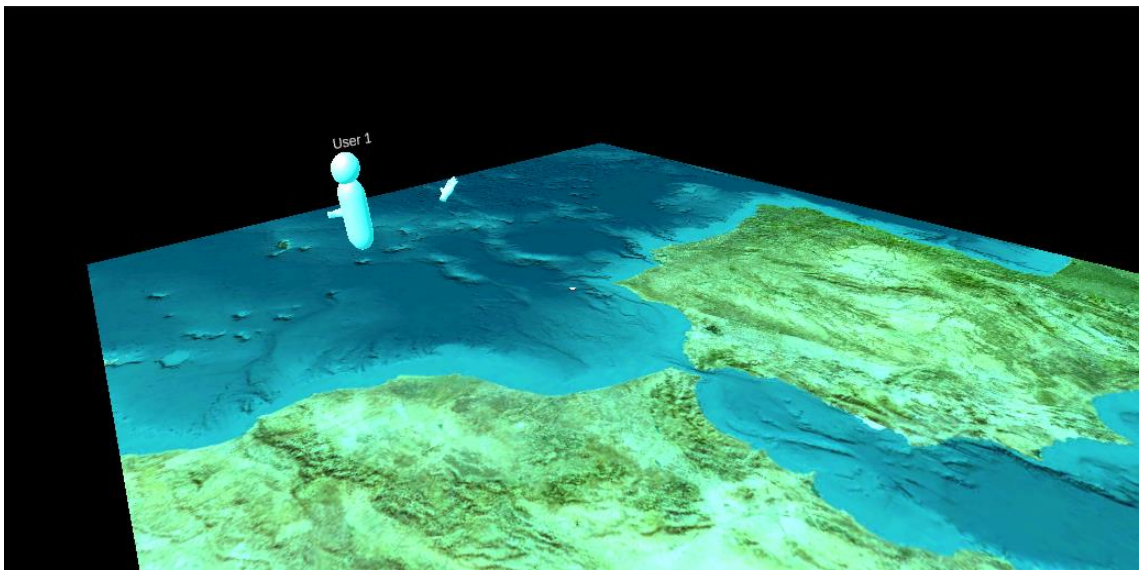

Figura 36.- Usuario observando a 'User 1' mover el mapa.

4. INSTANCIAR ACTIVOS

Los activos pueden ser instanciados de dos formas diferentes. Para ambas existen las mismas opciones. El primero de los botones del menú con la etiqueta MAP OBJECTS abre los activos que se instancian en el mapa en escala real (ver Figura 37).

Figura 37.- Objetos en el mapa.

El segundo de los botones que tiene la etiqueta FULLSCREEN OBJECTS instancia el activo delante del usuario en una escala mucho mayor a la real y permite al usuario verlo con mayor detalle (ver Figura 38).

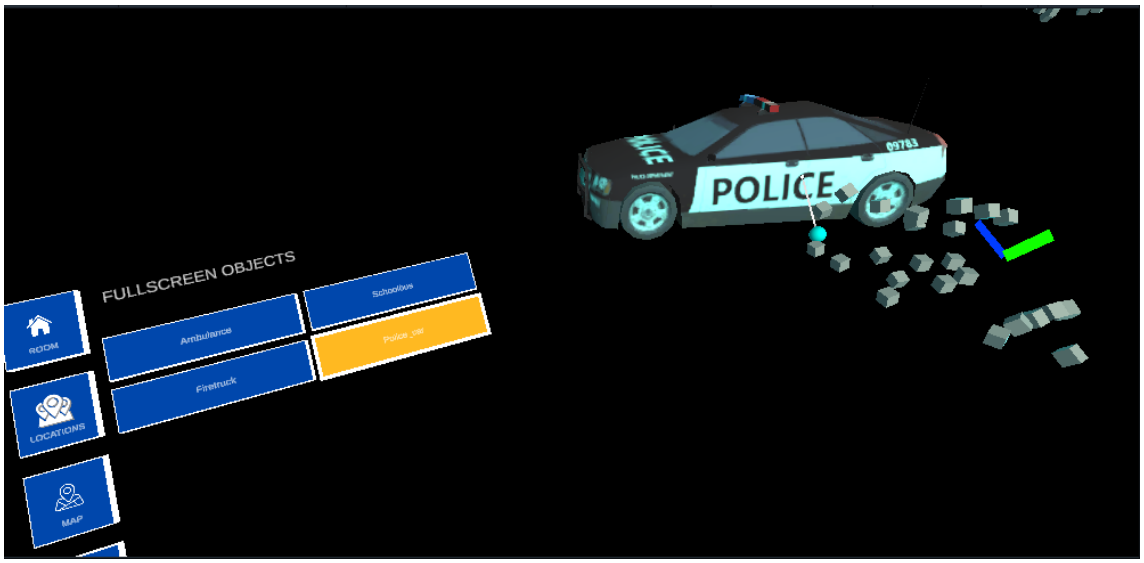

Figura 38.- Objetos en pantalla completa.

5. INTERACTUAR CON LAS HERRAMIENTAS

Al pulsar en el botón del menú TOOLS se abre el panel de herramientas. Estás herramientas permiten a los usuarios interactuar entre ellos y con el mapa (ver Figura 39).

La primera fila se corresponde con las herramientas de dibujo. Por orden se corresponden con dibujar líneas, medir y colocar pegatinas. El cuarto y último botón borra todo lo que se haya hecho con las tres anteriores herramientas.

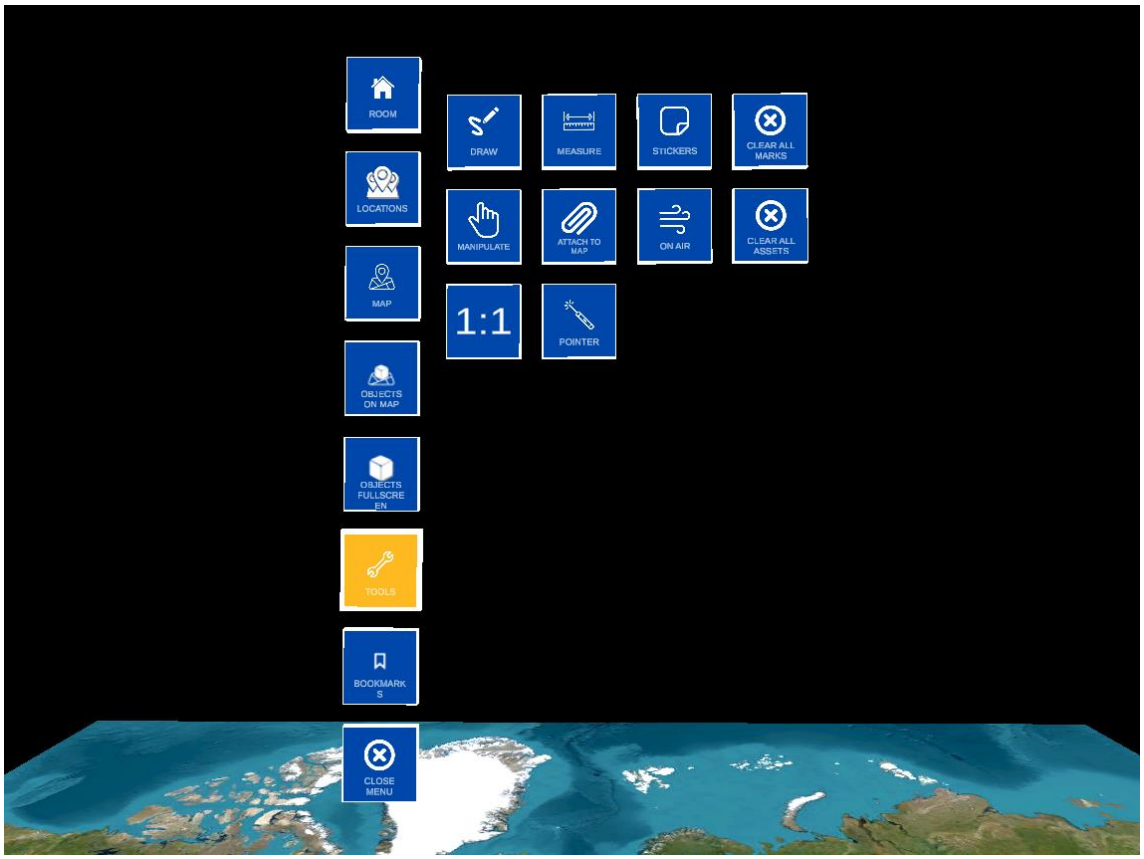

Figura 39.- Submenú Tools.

El botón con la etiqueta MANIPULATE permite al usuario elegir si quiere manipular un activo instanciado. Esto hace que no pueda utilizar ninguna de las tres primeras herramientas de la primera fila ya que son exclusivas. Además, existe un botón que se muestra y se esconde en función de otros botones pulsados. Este es el botón ATTACH cuya función permite al usuario anclar las líneas de dibujo o de medición al mapa sin dibujar encima del mapa.

Por orden, el siguiente botón es un interruptor que permite cambiar el "lienzo" en donde se "pintaran" las tres primeras herramientas, cambia entre el aire y el mapa. El siguiente sirve para borrar todos los activos que se hayan instanciado en el mapa. El penúltimo de todos estos botones indica la escala y permite al usuario intercambiar entre la escala del mundo real y la escala virtual que equivaldría a la que este mostrando el zoom del mapa.

El último de los botones principales de las herramientas es el botón del puntero que permite al usuario activar y desactivar un puntero láser que aparece de su mano (ver Figura 18).

Por último, los botones de dibujar, medir y crear pegatinas muestran unos botones secundarios que permiten editar las líneas y las propias pegatinas (ver Figuras 40 y 41).

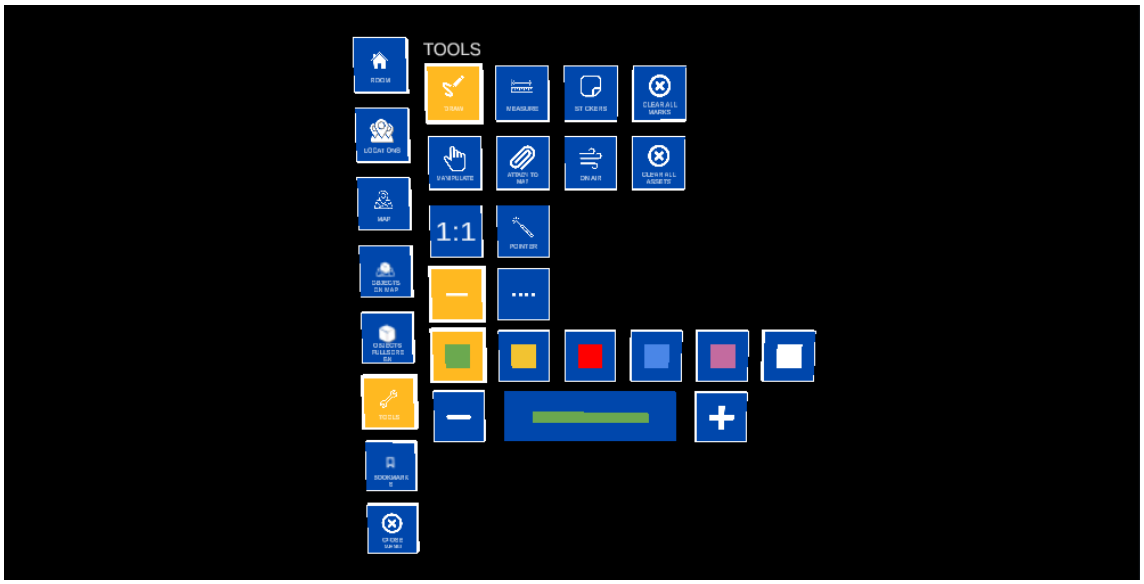

Figura 40.- Submenú Tools con las opciones de personalización de las líneas.

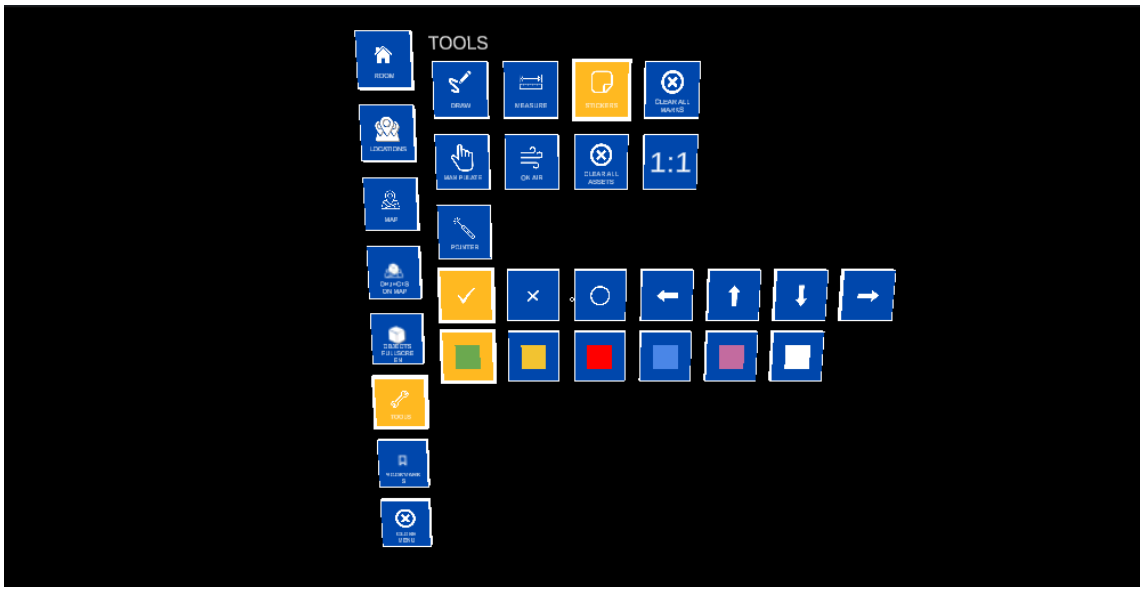

Figura 41.- Submenú Tools con las opciones de personalización de las pegatinas.

6. CARGAR/GUARDAR UNA SALA

En este último submenú el usuario puede pulsar el botón con la etiqueta SAVE para guardar en la memoria de las gafas la sala tal y como está y en un futuro poder cargarla y no tener que replicarla manualmente (ver Figura 42).

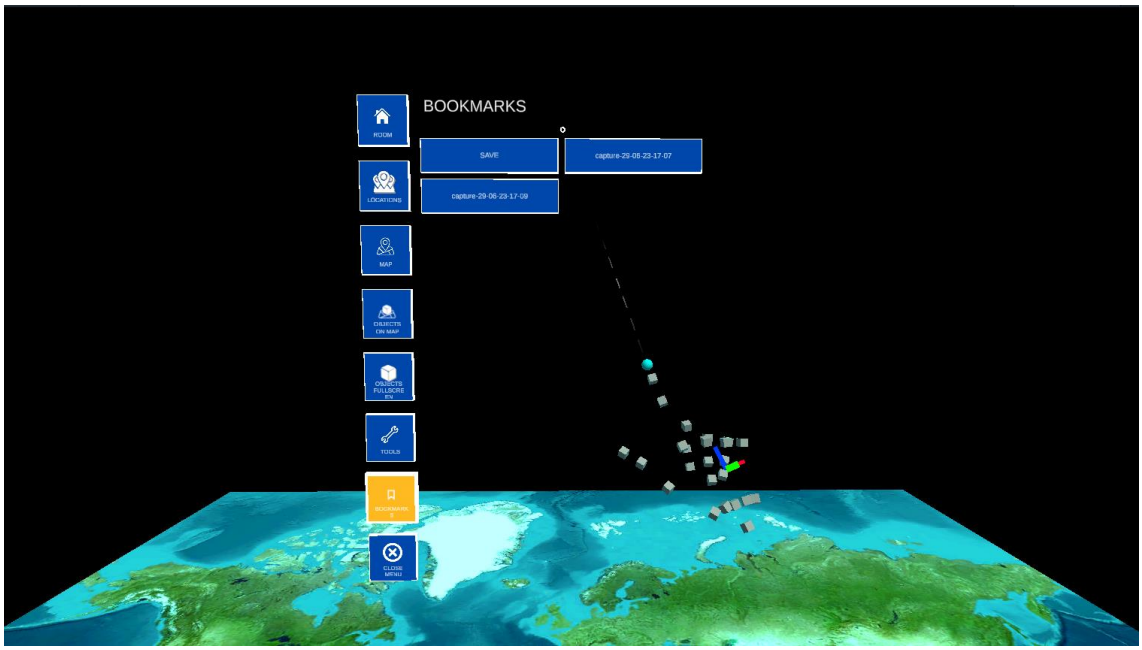

Figura 42.- Submenú Bookmarks.

ANEXO II: REFERENCIAS

Innovae. (s.f.). La tecnología de realidad virtual. Innovae. Recuperado de https://www.innovae.com/la-tecnologia-de-realidad-virtual/

Innovae. (s.f.). La realidad aumentada. Innovae. Recuperado de https://www.innovae.com/la-realidad-aumentada/

CreatXR. (s.f.). The virtuality spectrum: Understanding AR, MR, VR and XR. En CreatXR. Recuperado de https://creatxr.com/the-virtuality-spectrum-understanding-ar-mr-vr-and-xr/

Halopedia. (s.f.). Halografía. En Halopedia. Recuperado de https://halo.fandom.com/es/wiki/Holografía

Get it Easy. Recuperado de https://get-it-easy.de/en/hololens-2-rent/

Scheme Color. Blue and Yellow Color Scheme. Recuperado de https://www.schemecolor.com/blue-yellow.php

IEBSchool (s.f.). Metodología Scrum: Agile Scrum. Recuperado de https://www.iebschool.com/blog/metodologia-scrum-agile-scrum/

Wikipedia. (s.f.). Realidad mixta. En Wikipedia, la enciclopedia libre. Recuperado de https://es.wikipedia.org/wiki/Realidad_mixta

Microsoft. (s.f.). Mixed Reality. En Microsoft Learn. Recuperado de https://learn.microsoft.com/es-es/windows/mixed-reality/discover/mixed-reality

Wikipedia. (2023, 1 de julio). Myron W. Krueger. En Wikipedia, la enciclopedia libre. Recuperado de https://en.wikipedia.org/wiki/Myron_W._Krueger

Wikipedia. (2021, 26 de abril). Sensorama. En Wikipedia, la enciclopedia libre. Recuperado de https://en.wikipedia.org/wiki/Sensorama

Dreamteck. (s.f.). Dreamteck Splines: Manual de usuario. Recuperado de https://dreamteck.io/page/dreamteck_splines/user_manual.pdf

Dissonance. (s.f.). Dissonance: Documentación. Recuperado de https://placeholder-software.github.io/Dissonance/

Milgram, P., & Kishino, F. (1994). A taxonomy of mixed reality visual displays. IEICE Transactions on Information Systems, 77(12), 1321-1329.

Azuma, R. T. (1997). A survey of augmented reality. Presence: Teleoperators and Virtual Environments, 6(4), 355-385.

Sherman, W. R., & Craig, A. B. (2003). Understanding virtual reality: Interface, application, and design. Morgan Kaufmann.

Wikipedia. (2023, 2 de julio). Unity (motor de videojuego). En Wikipedia, la enciclopedia libre. Recuperado de https://es.wikipedia.org/wiki/Unity_(motor_de_videojuego)

Photon Engine. (s.f.). Photon Engine: Documentación. Recuperado de https://doc.photonengine.com/pun/current/getting-started/pun-intro

Microsoft. (2023, 21 de marzo). Mixed Reality Toolkit 2: Documentación. Recuperado de https://learn.microsoft.com/es-es/windows/mixed-reality/mrtk-unity/mrtk2/?view=mrtkunity-2022-05

Infinity Code. (s.f.). Infinity Code Online Maps: Documentación. Recuperado de https://infinity-code.com/assets/online-maps

Proyectos ágiles. (s.f.). Desarrollo iterativo e incremental. Recuperado de https://proyectosagiles.org/desarrollo-iterativo-incremental/