E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Herramienta Colaborativa en Realidad Virtual y mixta de diseño y prototipado de dispositivos interactivos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Pablo Iván Albito Tapia

Oscar Ardaiz Villanueva

Pamplona, 07/05/2022

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Agradecimientos y dedicatorias

He aquí el fin de una etapa y el inicio de otra. Muchos son los retos que hemos superado y muchos serán los que superaremos.

Por eso, antes de partir, quiero agradecer en primer lugar a las personas que colaboraron conmigo tanto en las pruebas como en las conceptualizaciones, a ellos, quien ya saben quiénes son, ¡Muchas gracias!

En segundo lugar, a mis amigos, que me han apoyado en todo momento, incluso en los momentos más difíciles de la carrera, y sobre todo en este último tramo, gracias a vosotros he llegado hasta aquí.

En tercer lugar, cómo no, agradecer a Oscar la oportunidad que ha decidido darme para trabajar con él codo con codo en este proyecto, y su compromiso de estar atento día tras día, “you’re breathtaking”.

En cuarto lugar, quiero agradecer a mi familia su eterno apoyo, tanto en las buenas como en las malas, sin vosotros nada de esto hubiese podido ser real. Y en especial agradecimiento a mi padre, que ha sido mi leal sujeto de pruebas durante toda la etapa de desarrollo :D. Os quiero mucho.

En quinto lugar, a los lectores de esta memoria, por ser tan valientes para adentrarse en estos lares.

Y, por último, a mi mismo. Si, por fin lo has logrado, puedes sentirte orgulloso ahora.

Resumen

Este trabajo se enmarca en un proyecto de investigación sobre aplicaciones de soporte a la creatividad para el diseño de objetos y artefactos en espacios de trabajo que tienen en cuenta la posición y movimientos del cuerpo construidas con tecnologías de realidad virtual y/o mixta. El trabajo consiste en una aplicación de realidad virtual y realidad mixta que permite conectar a 2 usuarios en una misma sala de forma online en donde podrán utilizar una serie de herramientas creativas tales como diferentes tipos de pinceles 3D para pintar sus ideas y poder realizar prototipos en tiempo real mediante herramientas de animación que permitan simular la interacción espacial con los objetos creados. El trabajo se ha desarrollado en el motor gráfico Unity usando C# como lenguaje y ha sido desplegado en las gafas Oculus Quest 2. En el trabajo, además se ha evaluado y demostrado la funcionalidad en escenarios con usuarios reales.

Índice

Contenido

Introducción

Realizar el TFG ha sido todo un desafío y un privilegio para mí. Desde pequeño siempre me ha interesado el mundo de los videojuegos, y esperaba algún día poder realizar algún proyecto en el que pudiese aprender a utilizar algún motor gráfico, y que posteriormente pudiese utilizar esos conocimientos para crear algo único y propio.

Fue entonces en noviembre cuando hablé con algunos colegas de la universidad que ya se habían graduado, que surgió la idea de crear algún proyecto interesante utilizando la última moda en tecnologías vanguardistas, conocida como realidad virtual o VR para los internacionales. Una tecnología por la que compañías como Meta (antiguamente Facebook) han metido todas sus papeletas, y ¿quién podría resistirse a tan emocionante e inigualable sensación de inmersión que surge la primera vez que te pones unas gafas VR encima?

La verdad es que el futuro de esta nueva herramienta está lleno de innumerables oportunidades y posibilidades y emociona poder ser de las primeras generaciones que pueden explorar y descubrir libremente todo este mundo que se abre ante nuestros pies.

Dicho esto, me fue muy fácil saber a qué persona pedir ser mi director de TFG. Cuando entré al despacho de Oscar, fue casi inmediato el visto bueno por parte de ambos. Y a pesar de que, por mi parte, no tenía conocimientos previos de ningún motor gráfico, ni de materias relacionadas con desarrollo de videojuegos (el campo más parecido a VR), ambos nos arriesgamos yendo a la antigua usanza, a aprender sobre la marcha.

Y después de hacer algunas pruebas con la reciente incorporación de la MR (Realidad Mixta) en el último dispositivo de Meta, las Oculus Quest 2, una idea ambiciosa surgió, y a partir de entonces, Oscar y yo empezamos a trabajar juntos para poder plasmar lo que en nuestra imaginación se hallaba, y como se suele decir, el resto es historia.

Justificación y objetivos

El prototipado tradicional de aplicaciones siempre ha tenido un problema fundamental intrínseco en las tecnologías usadas, ya sea mediante algo tan sencillo como dibujar en un papel, utilizar herramientas no específicas como el clásico Microsoft Power Point, o algo más enfocado a este fin, como Figma, Adobe XD, Marvel App, etc. Y es que todas ellas tienen en común la limitación a nivel de representación 2D. Si bien es cierto que muchas de ellas no intentan reproducir un escenario 3D puesto que normalmente sus escenarios están enfocados a dispositivos móviles o de escritorio que transmiten su información a través de pantallas con contenido 2D, hay escenarios en donde la creatividad se ve limitada debido a las necesidades de un entorno 3D donde se puedan tomar en cuenta el factor de profundidad. Aplicaciones como videojuegos, herramientas de diseño, herramientas audiovisuales, aplicaciones de realidad virtual y numerosas más, se benefician profundamente de esta capacidad.

Y qué mejor forma de prototipar un escenario en 3 dimensiones que utilizando una herramienta que permita sumergirte por completo en un mundo tridimensional. Aquí es donde entra nuestra tecnología número uno en el ámbito, la realidad virtual.

Y es que, gracias a los últimos avances en el campo, tenemos cada vez mayor acceso a dispositivos que permiten la inmersión en mundos virtuales. Tanto desde un mismo smartphone utilizando las famosas Google Carboards, como los dispositivos más destinados a este propósito como los HTC Vive, o en nuestro caso, gracia al apoyo de la universidad, las Oculus Quest 2.

Debido a esta necesidad y estas herramientas disponibles, se decidió idear una aplicación que permitiese a los usuarios con y sin experiencia en la realidad virtual, prototipar sus ideas de aplicaciones en un entorno tridimensional, dándoles la posibilidad de desatar su imaginación a través utilidades como pinceles, objetos, animaciones y más.

Pensamiento espacial

Claramente, la entrada de esta nueva inmersión que ofrece la realidad virtual y la mixta, presenta nuevos desafíos a los diseñadores ya que los elementos que conformarán nuestra experiencia de usuario no se limitan a una pantalla plana. Para ello debemos cambiar nuestra forma de pensar y de orientar la experiencia de usuario, tal y como menciona M.Pell:

> Para poder expresarnos bien dentro de este medio tridimensional, se necesita tener comodidad pensado y actuando en un espacio tridimensional. Eso no significa tener que aprender necesariamente modelado 3D. Tiene más que ver con adoptar el espacio como un lienzo en vez de una pantalla plana. El mundo alrededor es ahora un nuevo escenario para las ideas, no el

> espacio de los papeles o ventanas del ordenador. Considera todo el espacio que te rodea. El contenido puede ser colocado y trabajado en cualquier sitio de la habitación en la que te encuentras.
>
> Como es de esperar, el pensamiento espacial respecto al tradicional es un cambio grande, y ciertamente más difícil poner en práctica de lo que suena para la mayoría de nosotros.
>
> Casi todo lo hecho antes de este punto en la historia del diseño de la experiencia de usuario ha sido acerca de cómo controlar lo que está detrás de una pantalla como ventanas, menús, iconos, diálogos y botones. [1]

Ahora que las herramientas de desarrollo de realidad virtual están al alcance de todos, es nuestra oportunidad de aprovechar este nuevo entorno emergente para introducir un nuevo mundo de interacciones y experiencias que transformarán la manera de visualizar nuestras ideas en el entorno virtual.

Unity en VR

Con estas primeras ideas, surge el concepto de un Unity en VR, y es que los conceptos son muy similares si se compara con el editor en particular del que dispone Unity, en donde puedes crear muy rápidamente un prototipo de tu aplicación, arrastrando objetos clásicos de Unity como un simple cubo, u otros más complejos como cualquier modelo o prefab hecho con alguna herramienta como blender. Además de la capacidad de editar las dimensiones de estos elementos, la posición, rotación, materiales, físicas y animaciones.

El resultado entonces es una herramienta que empieza a tener cierta complejidad para un usuario estándar y al que, si bien tiene un multipropósito y es de uso más profesional, se aleja en parte de la audiencia a la que está destinada. Sin embargo, parece interesante profundizar en esta más complicada y a su vez completa versión de la aplicación, por lo que después de darle algunas vueltas, se decidió que no era una mala idea incorporar varias mecánicas a la aplicación y llamarla modo experto.

Modo experto

Este modo experto sería una versión más completa y orientada a un uso profesional, en donde el usuario tendría acceso a todas las herramientas anteriormente mencionadas y por lo tanto su nivel de habilidad y conocimiento debía ser mayor en este tipo de aplicaciones.

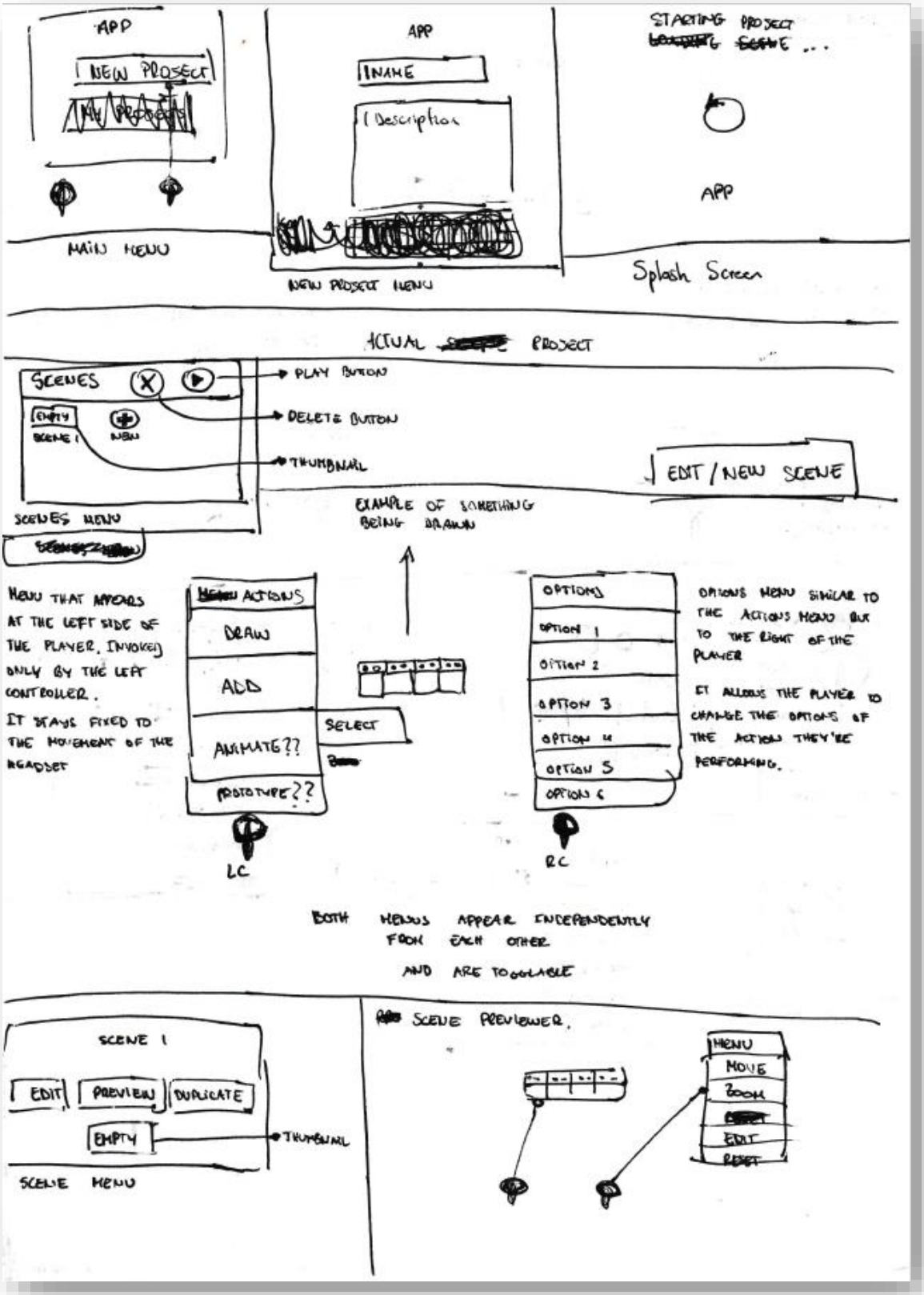


Ilustración 1:Sketch del modo experto I

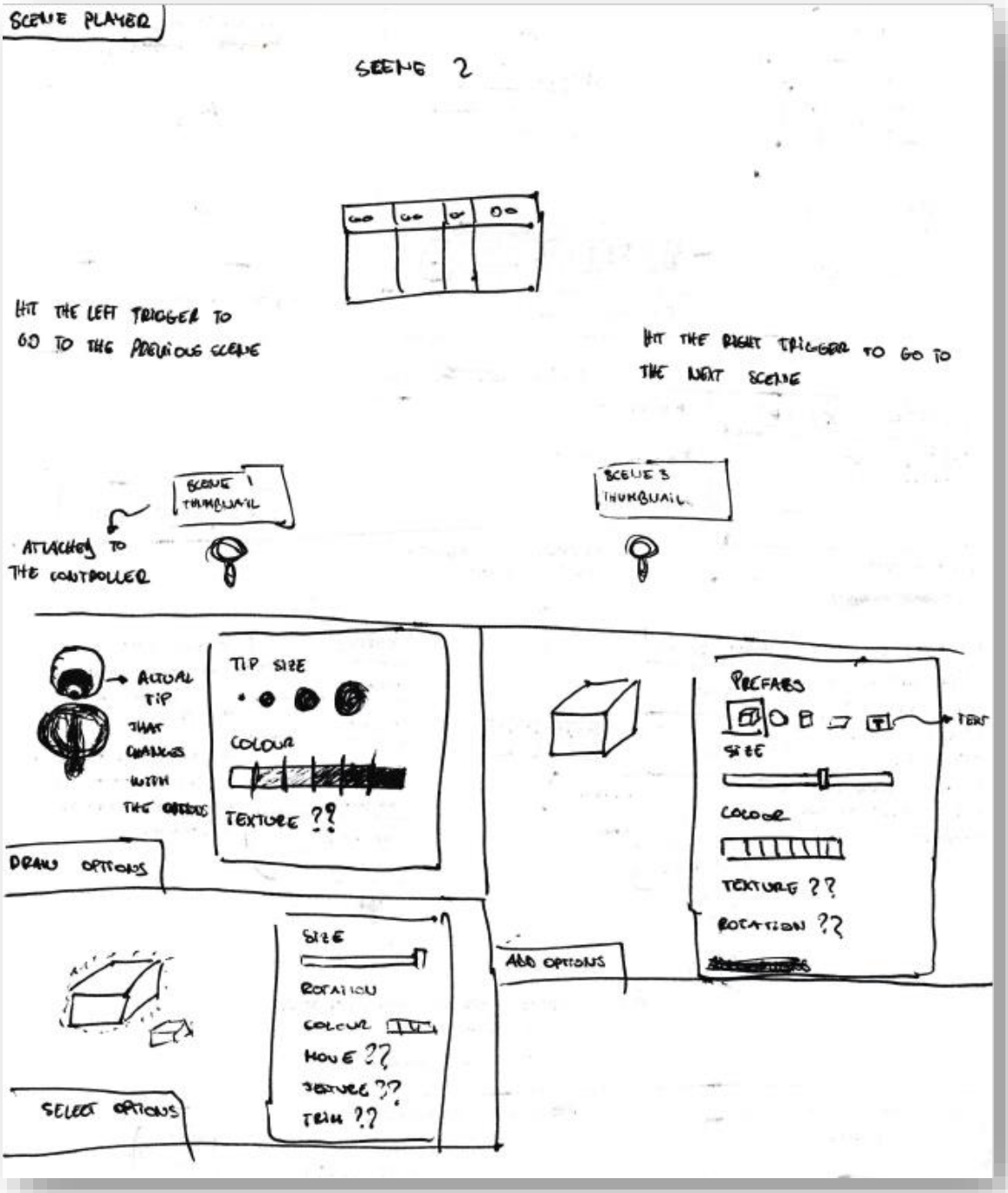


Ilustración 2: Sketch del modo experto II

Como podemos ver al principio, el modo experto tiene un sistema de escenas muy parecido al de Unity, en donde podemos diseñar y crear nuestro prototipo de aplicación y guardarlo fácilmente, para posteriormente ser editado, eliminado en caso de que el usuario lo desee, y mostrarlo como si de una presentación se tratase

Dentro de la creación de una escena, tenemos múltiples herramientas mostradas en el menú de la izquierda, y en la derecha las opciones de las mismas. Entre estas herramientas se encuentra un pincel para dibujar con diferentes tamaños, colores y texturas, un selector de objetos para añadir con diversas opciones de posición, tamaño, color, textura y rotación, un animador, y un sistema de prototipado.

Además de las funcionalidades propuestas en el diseño anterior, se ha buscado la posibilidad de añadir físicas y colisiones a nuestros objetos para que puedan interactuar entre ellos, la capacidad de seleccionar objetos y poder moverlos libremente y por último tener una lista de jerarquía de objetos existentes en la escena para poder agruparlos tal y cómo existe en Unity.

Ejemplo de caso de uso:

Nuestro objetivo es prototipar un juego de memorización que consiste en que un jugador dibuje una serie de elementos en distintas posiciones y que vayan apareciendo uno a uno de forma ordenada, y que luego desaparezcan totalmente. Un segundo usuario mientras tanto deberá memorizar las posiciones y el orden en el que hayan aparecido y deberá intentar colocar un objeto donde crea que van a ir las piezas en orden. Para ello se necesitará un pincel que dibuje los trazos, una agrupación de los trazos (listado objetos), una línea temporal para saber cuándo se ha pintado un elemento y cuando debe desaparecer (animaciones). Una serie de objetos (objetos básicos de Unity) que el segundo usuario colocará para intentar adivinar las posiciones de los trazos y un sistema para saber cuándo el segundo usuario ha acertado con las posiciones (colisiones).

Modo Fácil

Después de analizar el modo experto, todavía nos queda explorar un modo más cómodo y fácil de usar para usuario no tan expertos en la materia y que quieran prototipar de manera rápida y simple sus ideas con esta nueva herramienta.

Así que la siguiente pregunta es, ¿Qué es lo más básico que se necesita para poder prototipar una aplicación tridimensional?

El primer concepto de modo fácil fue reducir la cantidad de elementos utilizables y la cantidad de opciones que se pueden realizar. Para ello se optó por cambiar totalmente la interfaz y apostar por un diseño que aprovechase más la inmersión que la realidad virtual nos ofrece. Por lo tanto, se cambiaron los menús por un objeto con interacción, botones que se aprietan en vez de ser clicados, objetos que se cogen en vez de ser seleccionados, un modo interactivo de cambiar de color que consiste en sumergir el elemento que tienes en la mano en el color que deseas y un nuevo sistema de redimensión que pretende cambiar modificar el tamaño de un objeto mediante gestos en vez de selectores.

En este modo se han eliminado las animaciones, se han reducido las opciones que tiene cada elemento y se han quitado las físicas, colisiones y listado de elementos existentes.

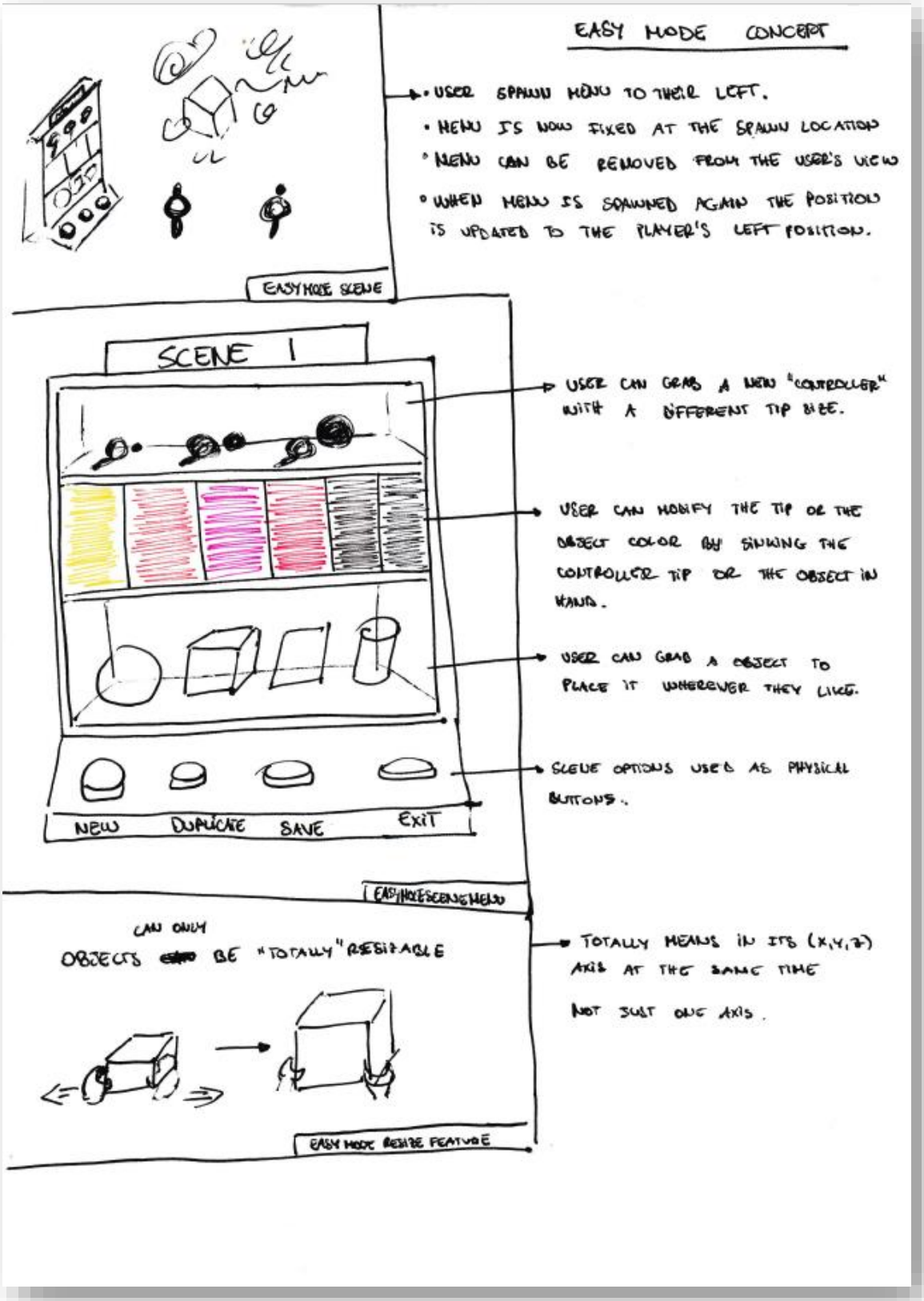


Ilustración 3:Sketch del modo sencillo

Pero después de contemplar durante un tiempo, se llegó a la conclusión de que el modo fácil propuesto no era lo más sencillo posible. El modo fácil debía tener algo que poder usarse inmediatamente y de forma casi evidente después de entrar a la aplicación, por lo que sus funcionalidades debían ser aún más reducidas, pero conservando la esencia y la funcionalidad al máximo posible

A la cuestión se nos ocurren 2 elementos indispensables como primera aproximación:

La posibilidad de utilizar un pincel para dibujar rápidamente el boceto inicial de prototipo, y la capacidad de animar de forma secuencial estos trazos.

El pincel o en general una herramienta con la que poder dibujar, garabatear o escribir, es por excelencia la mejor herramienta disponible para poder empezar un diseño rápido y efectivo. ¿Quién no ha creado su primera aplicación a partir de unos primeros bocetos a papel? Por lo tanto, se ha decido tenerla como uno de los imprescindibles.

El segundo elemento de animación es algo más complejo, lo que permitirá es convertir estos trazos simples en un prototipo. En líneas generales, lo que se pretende conseguir es que se pueda seguir una línea de tiempo con respecto al orden en el que se han dibujado los trazos y poder reproducir de nuevo estos trazos, dando libertad al usuario para elegir cuando pintar el siguiente grupo de trazos y de qué manera.

Con estas dos simples herramientas se empiezan a generar ideas sobre cómo se puede utilizar la herramienta de formas creativas, como por ejemplo ¿Se puede diseñar un minijuego jugable con sólo estas dos utilidades? ¿Se puede aplicar metodologías de prototipado como la de Mago de Oz? ¿Se puede actuar de forma interactiva con una segunda persona sin necesidad de un segundo par de dispositivos? ¿Y qué pasaría si se desease incluir un segundo jugador con un segundo par de gafas?

Multiusuario

Como respuesta a una de las preguntas hechas en el anterior punto, se ha pensado la posibilidad de no solo crear un prototipo de forma sencilla, si no de la capacidad de poder mostrar tu prototipo a otras personas en tiempo real, y no solo eso, sino también poder aportar a la creación de este. De aquí se han obtenido 2 ideas principales.

La primera es poder mediante sólo la parte de monousuario de la aplicación poder compartir los controles y las gafas con 2 usuarios al mismo tiempo. Esto significa poder dejar que un usuario prototipe su idea de aplicación con un mando y las gafas y que una vez finalizado, le deje las gafas al segundo usuario y que este primero vaya controlando las animaciones con el segundo mando.

La segunda idea es crear un sistema multiusuario o multijugador online en tiempo real para que 2 usuarios con 2 gafas puedan compartir sus prototipos en diferentes localizaciones con alta precisión.

Integración de realidad mixta o MR

Un sistema multiusuario en VR es perfecto para poder eliminar las barreras de la distancia cuando se quiere compartir la creación del prototipo de una aplicación con alguien que no está cerca físicamente de tu ubicación. Pero ¿qué hay de aquellas personas que trabajan junto a ti, esas personas que están en tu misma oficina, departamento o en tu misma clase?

Sería extraño tener que compartir una sala virtual en un entorno virtual totalmente alejado de la realidad cuando ambos usuarios se encuentran en la misma sala, ¿no?

Pues para responder a estos problemas, se ha decidido pensar en la última de las integraciones de Meta en su dispositivo estrella. La capacidad de añadir un passthrough en sus gafas, o, en otras palabras, poder ver a través de sus cámaras, haciendo que el escenario de nuestra aplicación sea el mismo sitio donde se encuentren los usuarios. Y esto permita que la experiencia de compartir y prototipar no sólo sea realista, sino que además se aproveche de elementos del entorno para poder dar más vida a los prototipos.

Trabajos relacionados

XRDirector

Es un sistema colaborativo de “authoring” inmersivo basada en roles que permite a los diseñadores crear escenas de interacción usando diferentes técnicas originarias de la industria del cine con dispositivos de realidad virtual y realidad aumentada como elementos primarios para manipular objetos virtuales. Principalmente creada para diseñar prototipos de escenas complejas y simular sistemas interactivos al estilo de Wizard of Oz testing sin necesidad de programar. [2]

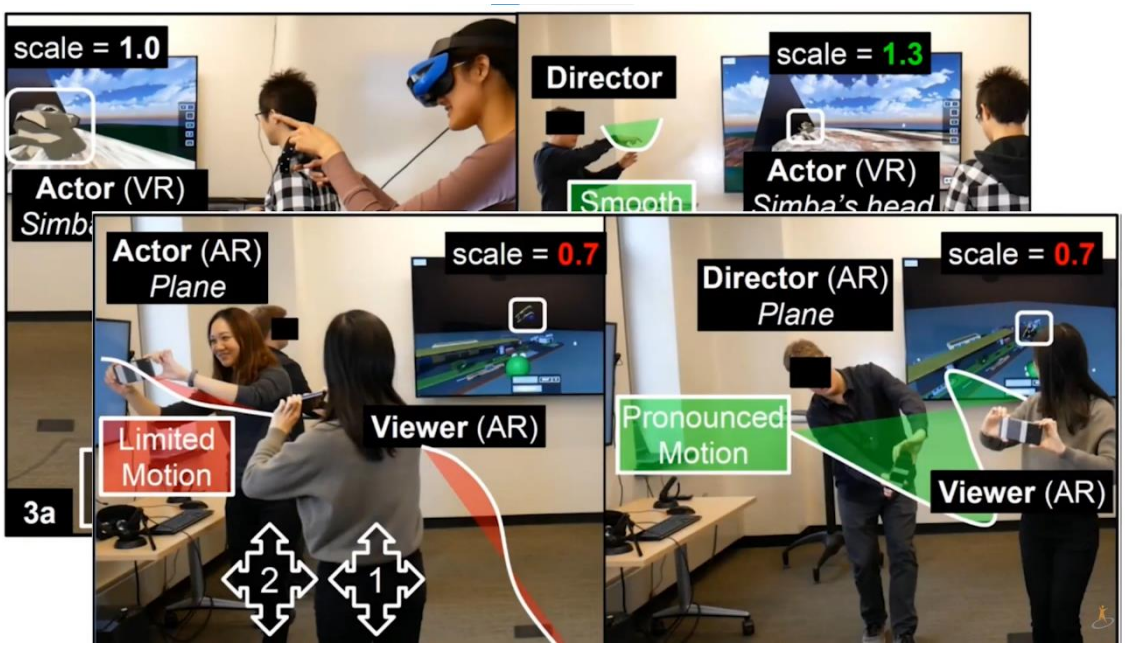


Ilustración 4: Elementos de XRDirector

MagicalHands

Es un sistema de "authoring" de animaciones basado en realidad virtual que pretende demostrar cómo los gestos manuales permiten crear mejores animaciones a partir de la generación y edición de fenómenos físicos como sistemas de partículas, deformaciones, acoplamiento, etc. [3]

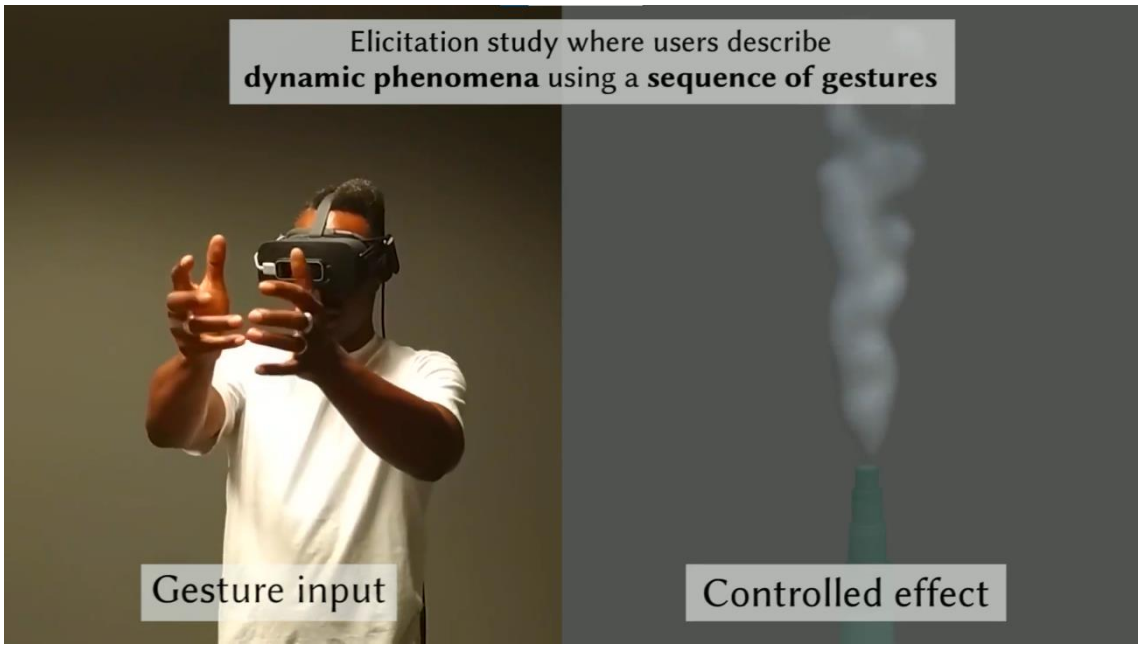


Ilustración 5: Ejemplo de uso de Magical Hands

Contexto tecnológico

El arte del prototipado

El prototipado es una técnica de diseño que permite crear una versión inicial del producto, en nuestro caso software, de manera que nos permita probar, evaluar, y validar frente a usuarios, jefes de proyecto, y en general a los stakeholders, antes de empezar a construir nuestra versión final, o durante diferentes etapas de desarrollo para obtener una retroalimentación constante.

Un prototipado además puede servir para definir o descubrir requisitos, resolver una integración técnica, uso de algún framework o comprobar rendimiento, probar interfaces para estudiar la usabilidad o conseguir inversores.

En un proyecto agile se entrega una solución al cliente tan pronto y tan a menudo como sea posible. Esta entrega temprana es especialmente valiosa cuando hay alguna duda de las necesidades reales del cliente. La funcionalidad de los primeros ciclos del proyecto puede ser muy limitada, pero útil, en cualquier caso. En algunos casos, la primera iteración puede ser una prueba de concepto. Debe aportar valor de negocio, aunque su funcionalidad sea muy limitada. El cliente recibe una primera impresión de los resultados finales y tiene la oportunidad de trabajar con algo, evitando el concepto vago

TIPOS DE PROTOTIPO

Rápido/Desechable: Después de haber obtenido una versión preliminar de los requisitos de usuario se crear una versión rápida y sencilla para mostrar visualmente a los usuarios cómo se vería el software acabado. Una vez cumplido el objetivo, este elemento será desechado en vez de formar parte de la solución final.

Evolucionable: Un prototipo robusto que sea perfeccionado constantemente para poder construir mejoras y requisitos adicionales a partir de este.

Producción: Un prototipo que cumple con las características necesarias para poder incorporarse como parte del producto final o como una primera solución.

PROTOTIPO TRADICIONAL

La creación de prototipos es especialmente útil para diseñar interfaces persona-ordenador y comprobar la forma de interaccionar. Se pueden clasificar según su fidelidad con el resultado final del producto de la siguiente manera:

Sketching

Se trata de realizar un dibujo rápido y sencillo para poder visualizar el concepto en mente y poder ver con claridad los pain points de los usuarios.

Ilustración 6: Sketching de una aplicación

Prototipo de baja fidelidad (Wireframes)

En esta categoría hablamos de evolucionar el sketch anterior y dotarle de contenido de manera que podamos centrarnos más en el diseño y ubicación del mismo sin ir a un nivel más preciso

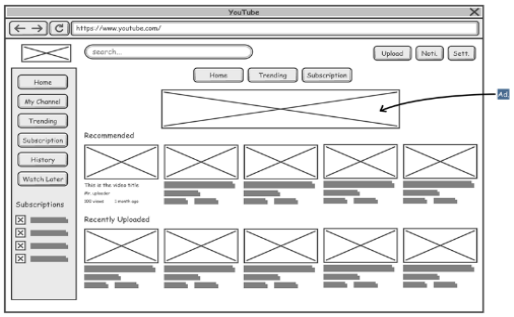


Ilustración 7: Wireframe de una aplicación

Prototipo de media fidelidad (Mockups)

Este prototipo incluye la parte de precisión que en la anterior fase se ignoraba y es que en este nivel se busca principalmente la fidelidad con el diseño final. Es el prototipo que mejor representa el producto final, pero sin incluir interacciones.

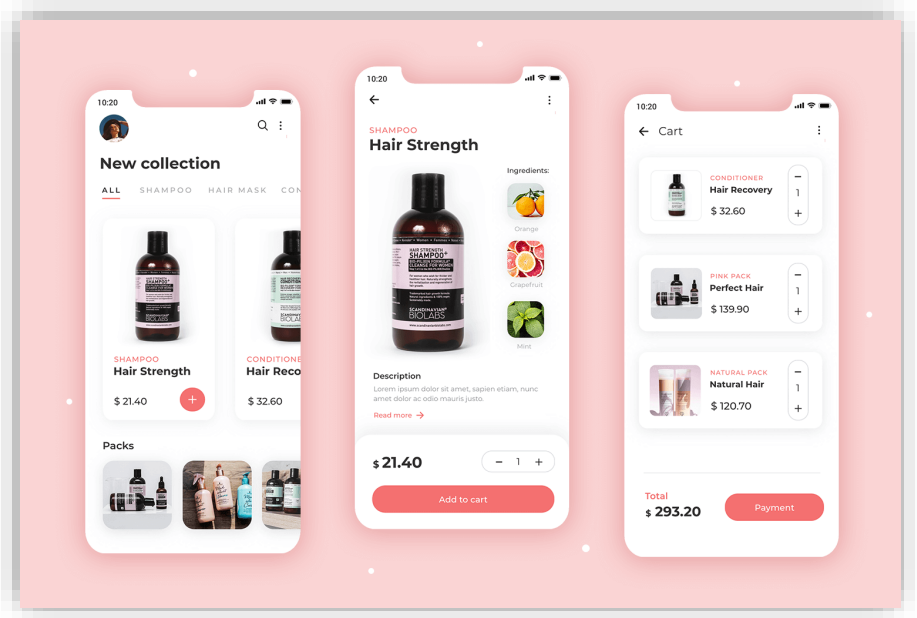


Ilustración 8: Maqueta de una aplicación

Prototipo de alta fidelidad (Maqueta)

Con las maquetas añadimos el último nivel faltante de interacciones que nos permitirán detectar problemas de usabilidad a través de varias pruebas con usuarios. Normalmente, es en este punto donde se acaba determinando si la idea final funciona. [4] y [5]

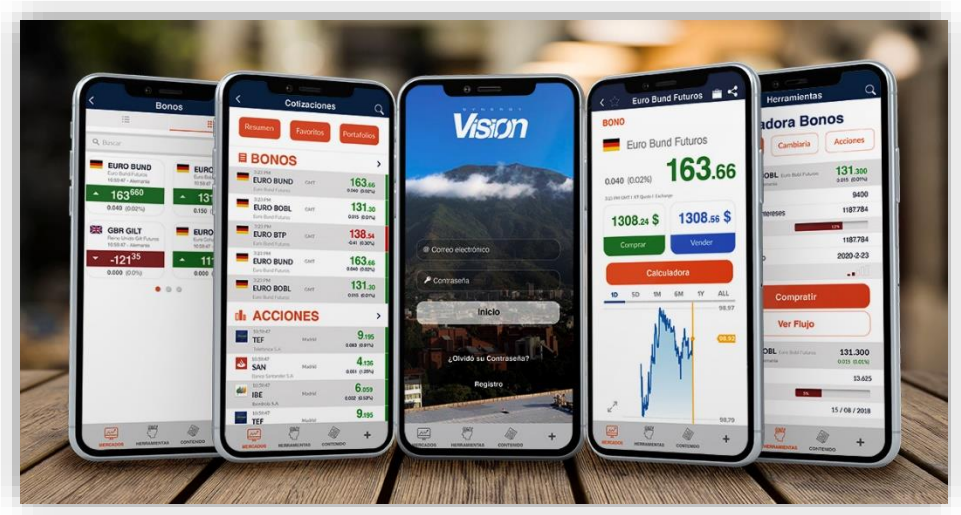


Ilustración 9: Maqueta de una aplicación

Wizard of Oz Testing

El método de Wizard of Oz es un proceso que permite al usuario interactuar con una interfaz sin saber que las respuestas son generadas por un humano en vez de por un ordenador.

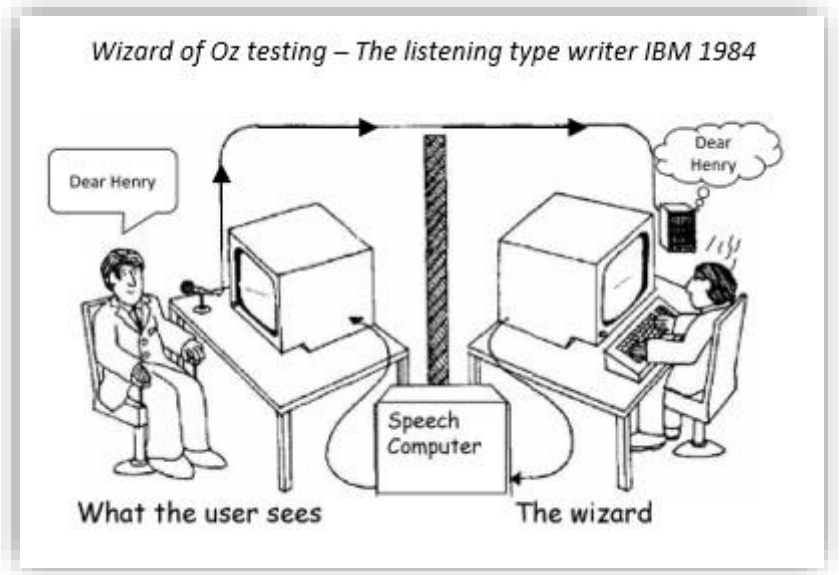


Ilustración 10: Demostración de Wizard of Oz Testing por IBM

Este método permite a los investigadores testar un concepto teniendo a un moderador liderando una sesión cara a cara con cada usuario, mientras un practicante – "el Mago" – controla las respuestas enviadas al usuario a través del dispositivo elegido. En la imagen superior, vemos un ejemplo para probar el concepto de "escritura por dictado". El usuario está sentado en una sala hablando por un micrófono mientras "el Mago" se sienta detrás de la escena escribiendo lo que el usuario está diciendo para que aparezca en la pantalla del usuario como si estuviera siendo hecho por un ordenador.

El porqué de esta prueba es sencillo, esta prueba permite a negocio testar las reacciones de los usuarios al sistema antes de si quiera pensar en el desarrollo. Esto puede ahorrar mucho tiempo, dinero, y prevenir que un producto salga al mercado antes de que sea apto y listo para los usuarios.

La estrategia a seguir para desarrollar este tipo de pruebas es crear prototipos de forma rápida y fácil de usar, de tal manear que se pueda falsificar de forma sencilla las respuestas dirigidas al usuario. [6]

Authoring inmersivo

"Authoring" inmersivo es un estilo de desarrollo en el que el sujeto que está creando el contenido está dentro de el entorno en el que desarrolla. En particular, dentro de los entornos de la realidad virtual y la realidad aumentada, esta técnica permite experimentar y revisar de primera mano todo el contenido que se está desarrollando, para de esa manera tener un feedback instantáneo de las sensaciones de inmersión, y usabilidad, siendo especialmente efectivo para usuarios no técnicos que diseñan escenas a través de interacciones entre elementos en vez de lógica programada. [7]

Realidad Virtual

La Realidad Virtual es un entorno de escenas y objetos de apariencia real generado mediante tecnología informática que crea en el usuario la sensación de estar inmerso en él. Dicho entorno se contempla a través de un dispositivo conocido como gafas de realidad virtual. Gracias a la realidad virtual podemos sumergirnos en videojuegos como si fuéramos los propios personajes,

aprender a operar un corazón o mejorar la calidad de un entrenamiento deportivo para obtener el máximo rendimiento.

La realidad virtual comprende dos elementos principales: el entorno del usuario y el entrono virtual. Mientras el usuario interactúa con el sistema de realidad virtual, los dos entornos se comunican e intercambian información a través de una barrera llamada interfaz. La interfaz puede considerarse como un traductor entre el usuario y el sistema de realidad virtual. Cuando el usuario aplica acciones de entrada (por ejemplo, movimiento generación de fuerza, voz, etc.), la interfaz traduce estas acciones en señales digitales, que pueden ser procesadas e interpretadas por el sistema. Por otro lado, las reacciones calculadas del sistema también se traducen por la interfaz en cantidades físicas, que el usuario puede percibir mediante el uso de diferentes tecnologías de pantalla y actuador (por ejemplo, imágenes, sonidos, olores, etc.). Finalmente, el usuario interpreta esta información y reacciona al sistema en consecuencia.

Usos

Educación: La realidad virtual puede mejorar la educación proveyendo a los estudiantes con experiencias memorables e inmersivas que de otra forma no sería posibles y además todo esto toma lugar dentro del aula.

Psicoterapia: En este campo, el uso de la realidad virtual ha sido bastante novedoso, ya que esta logra que el sujeto ya no se encuentre en una posición pasiva, permitiendo moverse por el entorno e interactuar con él de diferentes maneras logrando que la interacción se haga más íntima y con ello ganar ergonomía. Las aplicaciones que se han desarrollado hasta el momento tienen que ver con técnicas de exposición empleadas habitualmente para el tratamiento de las fobias, como la aerofobia, fobia social, agorafobia, pero se ha avanzado también en otros campos como los trastornos alimentarios. También existen numerosas aplicaciones de la realidad virtual para la rehabilitación psíquica y psicomotora.

Medicina: Se aplica la tecnología de realidad virtual por ejemplo para el aprendizaje de la anatomía y sobre todo en el área clínica: especialmente para el entrenamiento quirúrgico de los residentes en formación, y para los pacientes en el manejo del dolor, rehabilitación física y el tratamiento terapéutico de enfermedades mentales.

Entretenimiento: A lo largo de la década se han incorporado medios cuya finalidad es la de entretener al usuario abriendo un nuevo nivel de canal comunicativo exponencial, los ejemplos

más destacados son los cortometrajes realizados por Google Spotlight Stories destacado por su cualidad de storytelling para realidad virtual.

Videojuegos: Los videojuegos han evolucionado considerables. La calidad gráfica ha llegado al punto de igualar a la de consolas como en los ya vistos Doom VR y Fallout 4 VR haciendo un desplazamiento virtual por medio de teletransportación dentro del mapa. La mezcla entre experiencia de realidad virtual y videojuego VR culminan en un híbrido de las experiencias como se puede apreciar en Rick and Morty: Virtual Rick-ality o Job Simulator haciendo que el usuario pueda actuar de manera sencilla en el ambiente sin necesidad de desplazarse por medio de teletransportación con la finalidad de evitar fatiga visual.

Software: Actualmente, tras la incorporación de HDM compatibles con Steam VR han permitido incorporar dicha tecnología a software de diseño aportando una nueva perspectiva y manipulación del entorno para la creación de contenido digital. Software como Autodesk Maya, Maxon Cinema 4D, entre otros, han integrado la opción de edición mediante dispositivos de realidad virtual ya sea para reproducir o crear contenido para la misma plataforma. [8]

Realidad Mixta

La realidad mixta reúne elementos del mundo real y elementos digitales. En la realidad mixta se interactúa y se manipulan objetos y entornos tanto físicos como virtuales, mediante el uso de tecnologías de imagen y detección de última generación. La realidad mixta permite observar y sumergirse en el mundo que le rodea incluso mientras interactúa con un entorno virtual con sus propias manos, todo ello sin tener que quitarse el visor. Brinda la posibilidad de tener un pie en el mundo real y otro en un lugar imaginario, derribando los principios básicos entre lo real y lo ficticio, y ofrece una experiencia que puede cambiar la forma en que se juega y se trabaja hoy en día.

Los términos realidad mixta y realidad aumentada suelen utilizarse como sinónimos. Según la definición original de realidad mixta, esta abarca claramente muchos más aspectos que la realidad aumentada porque pone de relieve la interacción entre la realidad y los objetos virtuales: los objetos o sujetos del mundo real pueden interactuar con objetos o sujetos del mundo virtual. Por ejemplo, si colocas un objeto virtual en una mesa real, las aplicaciones de realidad mixta lo registrarán y moverán el elemento virtual con la mesa si tú desplazas la mesa

a otro lugar. Las aplicaciones de realidad aumentada más comunes no suelen responder a este tipo de cambios y, en casos así, el objeto virtual quedaría suspendido en el aire o desaparecería por completo.

La realidad mixta se utiliza hoy en día en sectores muy diversos, como en el de los videojuegos, las tiendas de venta en línea, la industria o el ámbito militar. En muchos casos, se trata de aplicaciones de realidad aumentada que solo permiten algunos aspectos de la interacción entre el mundo real y el virtual. Sin embargo, la realidad mixta desempeñará, sin duda, un papel mucho más decisivo en las aplicaciones del futuro. [9] y [10]

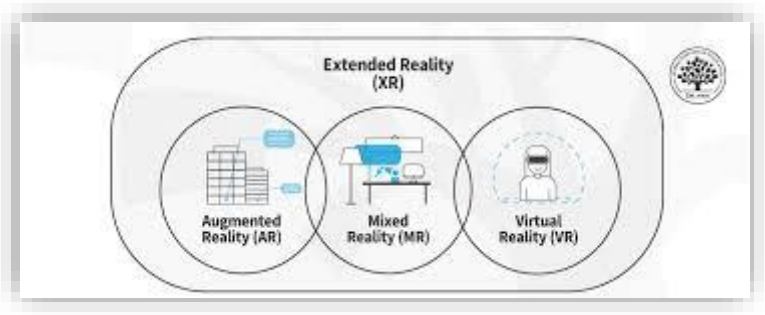


Ilustración 11: Representación del espectro de la realidad extendida o XR

Herramientas

UNITY

Ilustración 12: Logo de Unity

Unity es motor de juegos desarrollado por Unity Technologies, usado principalmente para desarrollar videojuegos y simulaciones para ordenadores, consolas y dispositivos móviles capaz de soportar gráficos 2D y 3D y una funcionalidad crear scripts a través de C#

Además, Unity ofrece servicios a desarrolladores como anuncios, analíticas, certificaciones, cloud build, y pone en disposición a los desarrolladores una tienda de assets tanto comerciales como gratuitas de terceros con recursos como texturas, modelos, plugins, extensiones del editor, incluso juegos enteros de ejemplo. [11]

Unity y VR

Unity tiene la capacidad de conectarse directamente con dispositivos de realidad virtual directamente desde la aplicación, sin ningún plugin externo en un proyecto. Provee una API base y funcionalidades con compatibilidad para múltiples dispositivos tanto en el presente como en un futuro con nuevos dispositivos y software. Por el momento la superficie de la VR API es mínima por diseño, pero se expandirá a medida que lo haga el campo de la realidad virtual. [12]

Unity y Oculus

Oculus Integration Package y sus componentes permiten desarrollar experiencias inmersivas específicas para Oculus incluyendo un pack de componentes, el ovrCameraRig, controles, un guardian, retroalimentación háptica y traqueo de manos, además de añadir múltiples herramientas como Audio SDK, build tools, métricas y un sample framework con el que poder observar escenas construidas a modo de tutorial y de descubrimiento de las capacidades del api que ofrece Oculus.

Competidores de Unity

Unreal Engine 4 (UE4):

- Acceso a código fuente para que los desarrolladores puedan estudiar el motor y puedan aprender cómo se usa
- Un framework multijugador, visualización a tiempo real y un flexible editor.
- Un modo de blueprints visual scripting que permite crear prototipos rápidos
- Uso sencillo de herramientas de animación, efectos de sonido, simulaciones, etc.

CRYENGINE:

- Excelentes visuales
- Sandbox y más herramientas destinadas a VR
- Soluciones de audio incorporadas
- Construir visualizaciones e interacciones a tiempo real.

Amazon Sumerian: Solución de amazon para la realidad virtual no orientado a programación.

- Sumerian editor
- Sumerian host
- Gestion de assets
- Capacidad de crear scripts para generar lógicas de escena

[13]

VISUAL STUDIO 2019

Ilustración 13: Logo de Visual Studio 2019

Visual Studio trae consigo funcionalidades específicas para desarrollar programas en C# ayudándose de la velocidad y precisión de IntelliSense. Además, este editor trae un sistema de debugeado para el motor grafico de Unity, permitiendo añadir breakpoints, evaluar variables y expresiones complejas. [14]

BLENDER

Ilustración 14: Logo de Blender

Con Blender se pueden crear visualizaciones 3D como imágenes, animaciones 3D, disparos VFX, y edición de video. Encaja perfectamente para individuos y pequeños estudios que se benefician de su pipeline unificada y su responsivo proceso de desarrollo. Es además una aplicación multiplataforma con una relativamente baja memoria y unos requerimientos comparada con otras herramientas de diseño 3D, perfecta para la creación rápida de assets para poder ser utilizados posteriormente en Unity. [15]

OCULUS QUEST 2

Ilustración 15: Oculus Quest 2

Son unos cascos de realidad virtual desarrollados por Facebook Reality Labs. Tal y como su predecesor, Quest 2 es capaz de correr en un sistema operativo basado en Android y en el software compatible con Oculus de escritorio cuando está conectado via usb o WI-FI. Es una actualización del origina Oculus Quest con un diseño similar, pero con menor peso, especificaciones internas mejoradas, una pantalla con mejor tiempo de refresco y resolución por ojo, y unos controles mejorados. Estas características, junto a una disponibilidad y asequibilidad en el mercado, conviertieron a las Oculus Quest 2 en el dispositivo más vendido de realidad virtual en el 2021. [16] y [17]

PHOTON ENGINE

Ilustración 16: Logo de Photon Engine

Es un motor de juegos especializado en desarrollo de juegos multijugador. Contiene una serie de productos, software, tecnología y componentes de redes que trae consigo una gran velocidad, rendimiento y más a los juegos en línea. [18]

Photon y Unity

Photon Unity Networking (PUN) es un paquete de Unity para juegos multijugador. Contiene un emparejamiento flexible que permite introducir a jugadores en salas en donde los objetos pueden ser sincronizados a través de la red. RPCs, Custom Properties o eventos de Photon de bajo nivel son algunas de sus funcionalidades. La comunicación rápida y opcional es hecha a través de servidores dedicados de Photon, para que los clientes no tengan que conectarse uno con uno.

PUN exporta a básicamente todas las plataformas soportadas por Unity y tiene dos opciones:

PUN FREE: Paquete sin coste con varias demos, scripts de prueba y varia documentación de referencia. Escogido por preferencia.

PUN PLUS: Tiene el mismo contenido que el PUN FREE añadiendo un plan de 100 usuarios concurrentes en el Photon Cloud. [19]

Cuerpo del trabajo

Gestión del Proyecto

Para poder construir este producto software se ha empleado una metodología ágil dentro del marco de gestión de proyectos que ha facilitado la realización del mismo. El siguiente es el backlog del proyecto:

Nº	Como	Quiero	Para	Estimación	
1	Usuario	Dibujar un trazo en realidad virtual	Realizar un diseño rápido	1	Épica 1: Experimentación
2	Usuario	Dibujar múltiples trazos	Realizar un diseño más elaborado	2	
3	Usuario	Borrar el último trazo creado	Poder eliminar trazos mal hechos	1	
4	Usuario	Borrar todos los trazos creados	Poder empezar a diseñar desde cero	1	
5	Usuario	Utilizar el passthrough de Oculus	Observar mi entorno dentro de la aplicación	2	
6	Usuario	Conectarme a un servidor multijugador	Poder conectarme con más usuarios	3	
7	Usuario	Crear una sala	Permitir que otros usuarios puedan entrar a mi partida	1	
8	Usuario	Unirme a una sala	Poder entrar en las partidas de otros usuarios	1	
9	Usuario	Tener un menú donde pueda elegir unirme a una sala o crear una sala	Facilitar a los usuarios conectarse a una sala	3	
10	Usuario	Visualizar otros usuarios	Observar su posición en tiempo real	3	
11	Usuario	Dibujar un trazo en una sala online	Permitir a otros usuarios ver el trazo en tiempo real	2	
12	Usuario	Eliminar el último trazo en una sala online	Permitir ver a otros usuarios como se borra el último trazo creado	1	
13	Usuario	Eliminar todos los trazos en una sala online	Permitir ver a otros usuarios como se reinicia el diseño	1	
14	Usuario	Aplicar el passthrough al multijugador	Permitir al usuario visualizar a un segundo usuario en su propio entorno	1	
15	Usuario	Sincronizar el posicionamiento del usuario en la sala online	Que los usuarios puedan ver los trazos de otros usuarios en una posición relativa al mundo real y no al virtual	13	Épica 3: Drawing Animator / Épica 2: MR Online Drawing
16	Usuario	Crear una animación al dibujar un trazo	Reproducir la realización de los trazos desde que se han creado hasta que se ha terminado de diseñar	5	

17	Usuario	Tener dos tipos de trazos, una línea estática y una dinámica	Tener líneas que no cambien durante el proceso de animación y otras que se difuminen una vez acabado su tiempo de vida	3		
18	Usuario	Tener una línea temporal de animaciones	Permitir que los trazos dinámicos puedan desaparecer y los estáticos permanecer	3		
19	Usuario	Difuminar la línea dinámica cuando se esté dibujando y cambiando de línea temporal	Tener una referencia de la última línea temporal y poder hacer más trazos a partir de ella	3		
20	Usuario	Tener diferentes tipos de velocidad en la que se reproduzca la animación	Reproducir más deprisa o más lento una animación dependiendo de lo que el usuario desee	3		
21	Usuario	Reproducir de nuevo la animación	Observar cuantas veces se quiera toda la animación	2		
22	Usuario	No poder dibujar mientras se está reproduciendo una animación	Prevenir obtener comportamientos inesperados	2		
23	Usuario	Dibujar trazos una vez acabada la animación	Seguir prototipando si todavía no se había acabado	3		
24	Usuario	Elegir entre 8 diferentes colores de trazo.	Permitir al usuario tener más capacidad creativa	3	Épica 4: Mejoras Drawing Animator	
25	Usuario	Tener un modo de reproducción instantáneo	Tener trazos que no dependan de la velocidad en la que se han pintado	2		
26	Usuario	Tener indicadores de progreso de la animación	Saber cuándo se está reproduciendo una animación, cuando se puede pasar a la siguiente animación y cuando se ha acabado para poder seguir pintando o reproducir de nuevo	3		
27	Usuario	Cambiar el tamaño de la punta del pincel	Realizar trazos con diferentes tamaños	2		
28	Usuario	Tener el modo de reproducción de animación clásico de forma ininterrumpida en vez de secuencial	Tener una forma alternativa de visualizar las animaciones	2		
29	Usuario	Tener un menú mejorado	Tener una mejor experiencia de usuario	2		
30	Usuario	Tener un pincel más visual	Tener la sensación de estar usando un pincel real	2		
31	Usuario	Cambiar de línea temporal cada vez que dibuje una línea dinámica	Tener los controles más básicos posibles	2		
32	Usuario	Integrar el modo multiusuario con el modo monousuario	Tener todas las funcionalidades nuevas en el modo multijugador	8		

Desarrollo de la aplicación

Épica I: Experimentación

Antes de empezar a desarrollar el modo fácil, primer se realizaron varias pruebas sobre distintas herramientas de las que se disponía y de las que la aplicación se va a basar. Esto, en otras palabras, es realizar varias pruebas de concepto para saber cuál es el camino más adecuado a seguir para alcanzar los objetivos propuestos. En esta parte se explicarán los elementos que se han estudiado y algunas de las pruebas que se han realizado con ellos.

OvrCameraRig

Los objetos tipo CameraRig son de vital importancia para el desarrollo de aplicaciones de realidad virtual. Son aquellos que introducen la comunicación entre el dispositivo de realidad virtual y el objeto en la escena de Unity. Es decir, contienen la posición de las gafas y de los mandos y la trasladan a una cámara que se podrá ubicar dentro de una escena y los mandos que podrán interactuar con diferentes elementos de la misma.

En específico se utiliza el CameraRig diseñado por Oculus llamado OvrCameraRig, que, a diferencia de otros CameraRig, este tiene la capacidad de comunicarse con las opciones específicas y experimentales de los dispositivos de Oculus, como grabaciones, hand tracking, y la más importante para este proyecto, el acceso al passthrough que nos permitirá trasladar la aplicación a un entorno de realidad mixta.

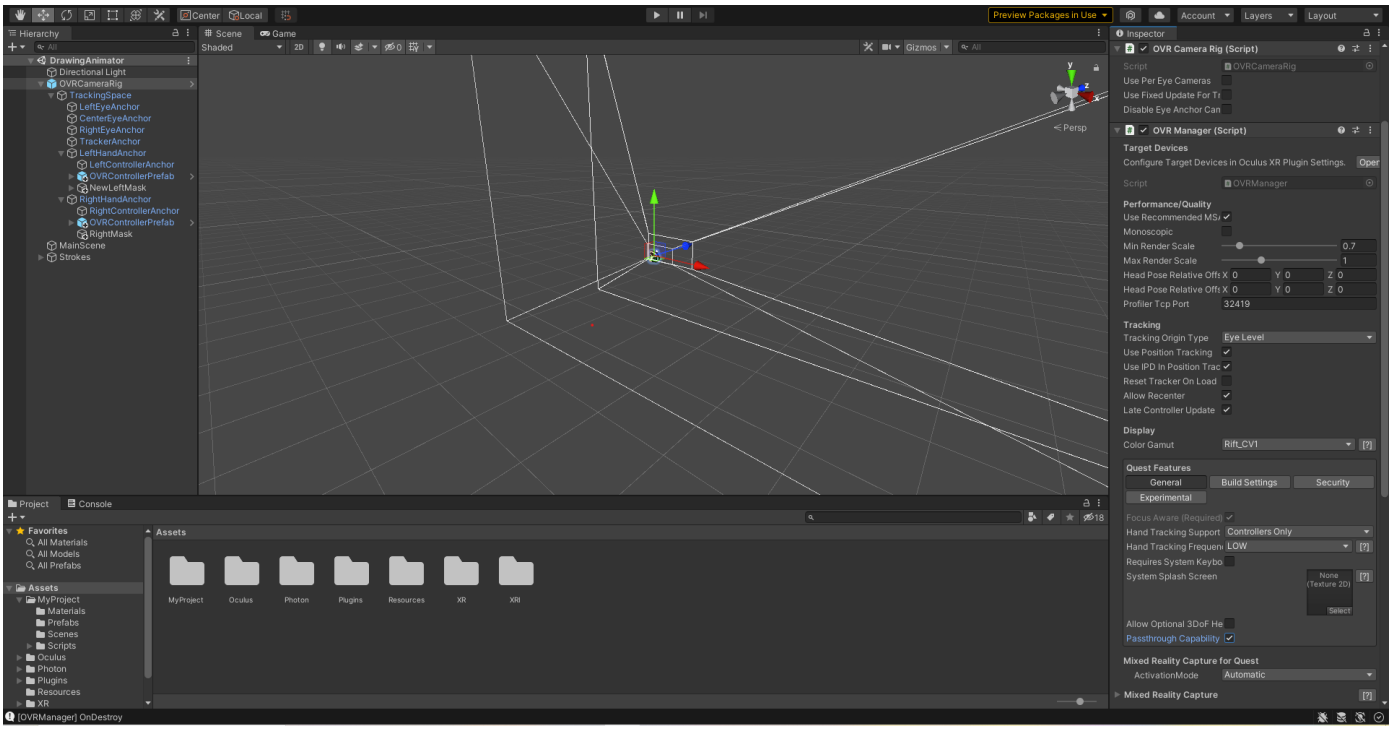


Ilustración 17: ovrCameraRig en Unity

Limitaciones del OvrCameraRig

Una de las principales limitaciones de las que se hablará más profundamente en posteriores pasos es el capado de movimientos posicionales y rotacionales de las cámaras y mandos del ovrCameraRig, que, en otras palabras, representan la cabeza y las manos del usuario. Esto en otras palabras, limita la capacidad de mover al usuario a únicamente poder hacerlo de forma externa, y deja opciones importantes como el reposicionamiento y el recentrado totalmente fuera del alcance de los desarrolladores.

Line Renderer

El Line Renderer es un objeto que permite almacenar posiciones de vectores tridimensionales y renderizar estas siguiendo diferentes parámetros configurables

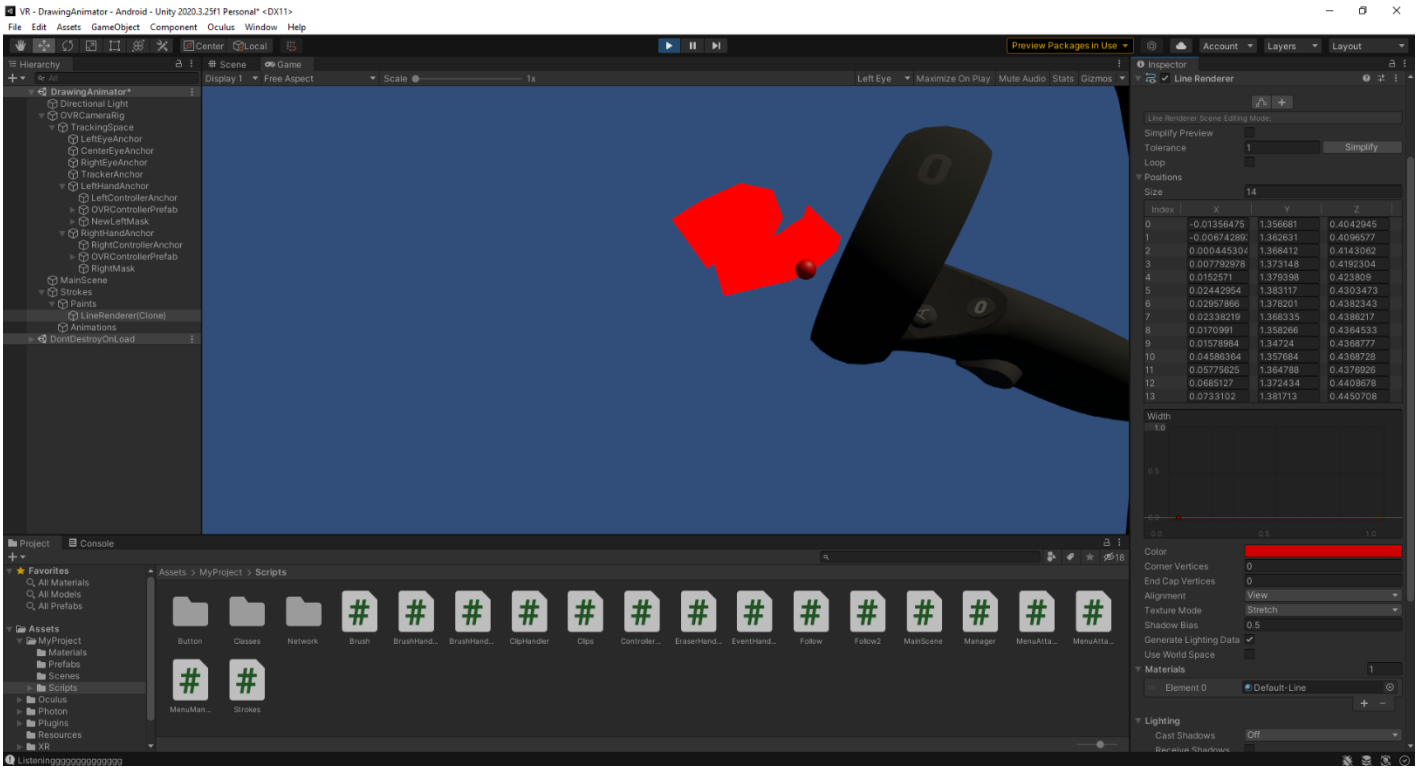


Ilustración 18: LineRenderer en Unity

El modo de operar es sencillo, basta con crear un script que permita obtener la posición del mando y crear un evento tal que cuando pulses un botón se empiecen a almacenar los vectores del mando.

El color, iluminación y textura dependen del material que se le asigne al Line Renderer, en nuestro caso, se deja el valor por defecto ya que tiene propiedades idóneas para tener iluminación propia y de este modo, no se verá afectado por las condiciones de luminosidad de la escena.

Otra configuración interesante es la posición local o global de los vectores almacenados, es decir, que las posiciones obtenidas de los vectores se tomen de forma global, o relativa al objeto que contiene al Line Renderer. Por ejemplo, si el vector de referencia que tengo es (1, 1, 1) y el objeto tiene una posición global, la línea se pintará en la posición (1, 1, 1), pero si cambiamos a la configuración a local, y este objeto está contenido en un objeto padre cuya posición es (1, 0, 1) ahora el vector resultante contenido en este objeto padre no se pintará en la posición (1, 1, 1), si no en la posición (1, 0 ,1) + (1, 1, 1) = (2, 0, 2). Esto será de vital importancia para siguientes pasos.

Photon

La integración de Photon con Unity es muy sencilla. Sólo hay que entrar a Photon, crear una cuenta y añadir una aplicación a la que se le asignará un servidor de manera gratuita y limitada para 20 usuarios simultáneos con una APP ID para que puedas conectarte desde Unity

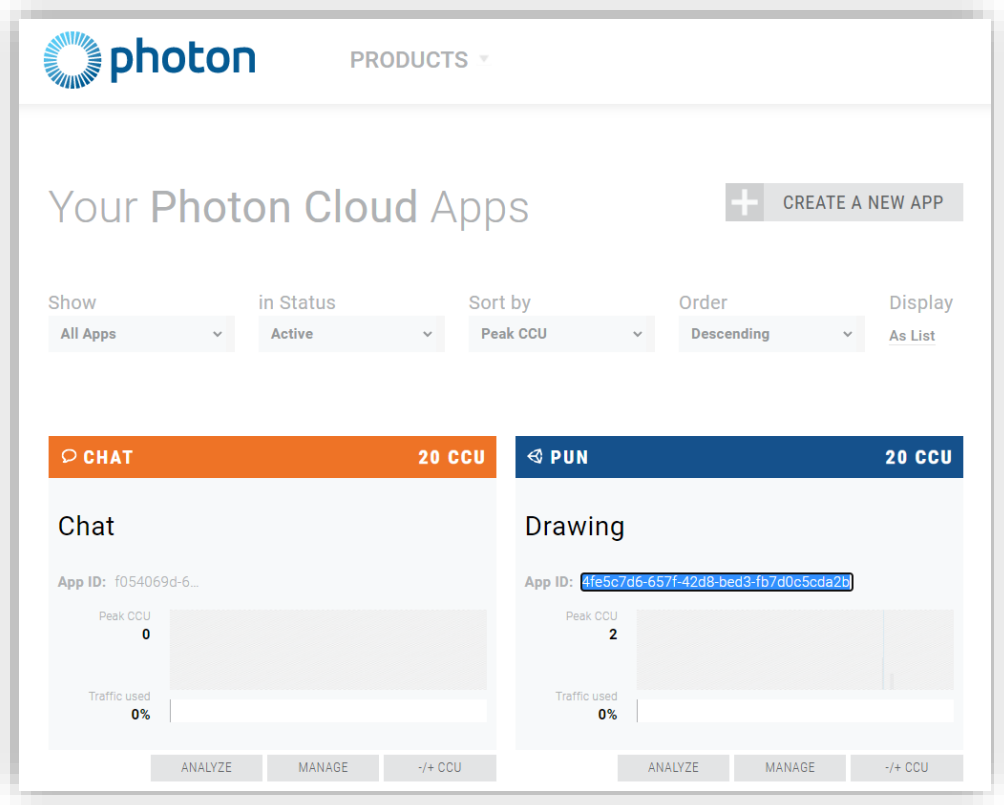


Ilustración 19: Workspace de Photon

Luego hay que entrar a Unity, descargar e instalar los paquetes necesarios y configurar el script para conectarte a tu servidor.

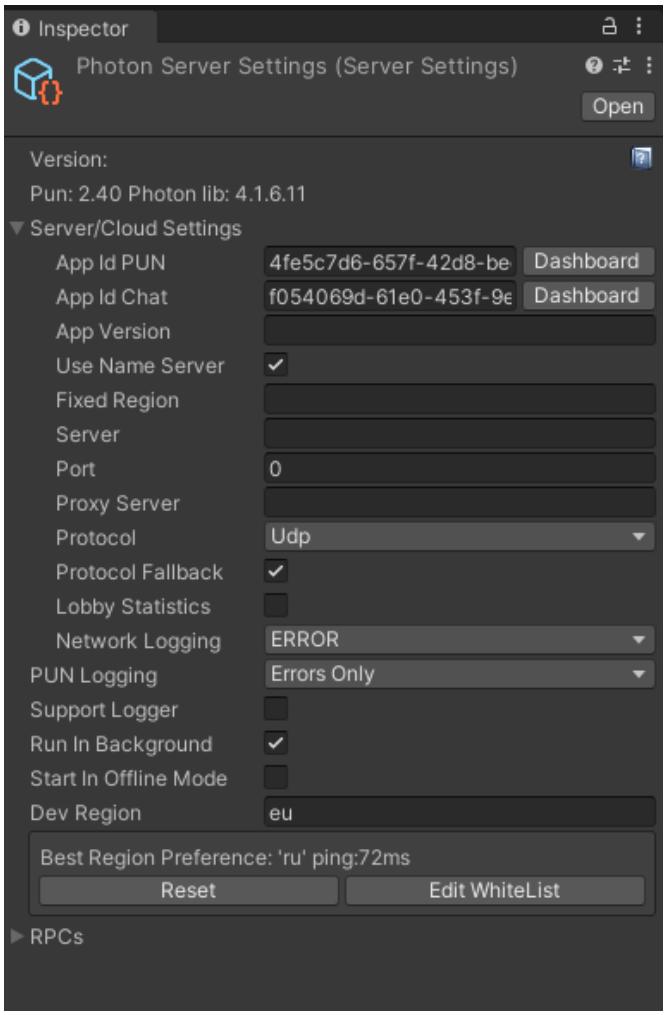


Ilustración 20: Configuración de la conexión con photon en Unity

Una vez conectado el funcionamiento es simple. Photon te permite extender de objetos tipo MonoBehaviourPunCallbacks que te permitirá acceder a callbacks que se ejecutan durante las interacciones con el servidor, al crear una sala, al unirte a una sala, y realizar acciones que reaccionan a este tipo de eventos como poder crear un objeto que represente a un jugador cuando un usuario se haya conectado.

```
Unity Script (2 asset references) | 0 references
public class NetworkManager : MonoBehaviourPunCallbacks
{
    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        this.ConnectToServer();
    }

    1 reference
    private void ConnectToServer()
    {
        PhotonNetwork.ConnectUsingSettings();
        Debug.Log("Try Connect to Server...");
    }

    3 references
    public override void OnConnectedToMaster()
    {
        Debug.Log("Connected TO Server");
        base.OnConnectedToMaster();

        RoomOptions roomOptions = new RoomOptions();
        roomOptions.MaxPlayers = 10;
        roomOptions.IsVisible = true;
        roomOptions.IsOpen = true;

        PhotonNetwork.JoinOrCreateRoom("Room 1", roomOptions, TypedLobby.Default);
    }
```

Ilustración 21: Ejemplo de MonoBehaviourPunCallbacks

Además, photon trae consigo scripts llamados PhotonViews que permiten sincronizar un objeto simple de Unity a través de la red, es decir, que crea y mantiene la posición, rotación, tamaño y otras propiedades básicas de un objecto para todos los usuarios de forma simultánea. Por ejemplo, Puedo crear un objeto simple que represente a un jugador, añadirle un PhotonView y hacer que este objeto siga los movimientos de las gafas. De este modo cada uno de los jugadores que se unan a la sala verá este objeto moviéndose en tiempo real y sincronizada dentro de su escena.

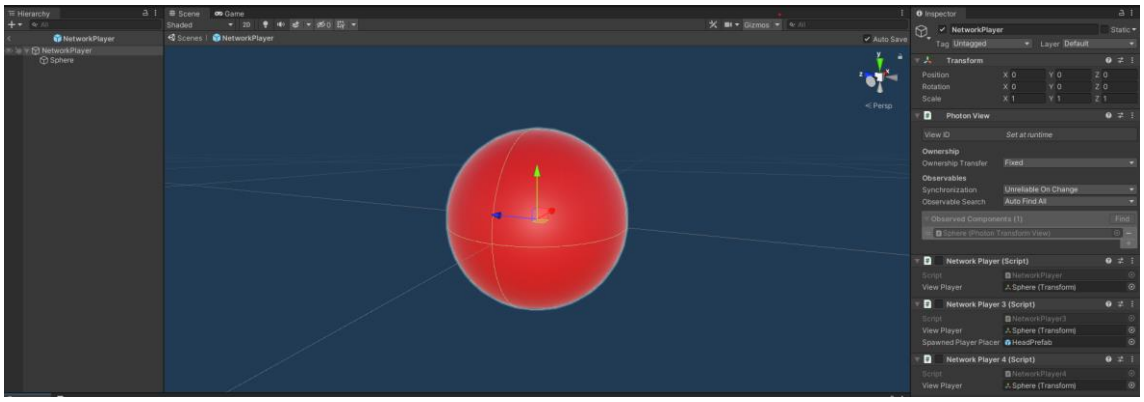
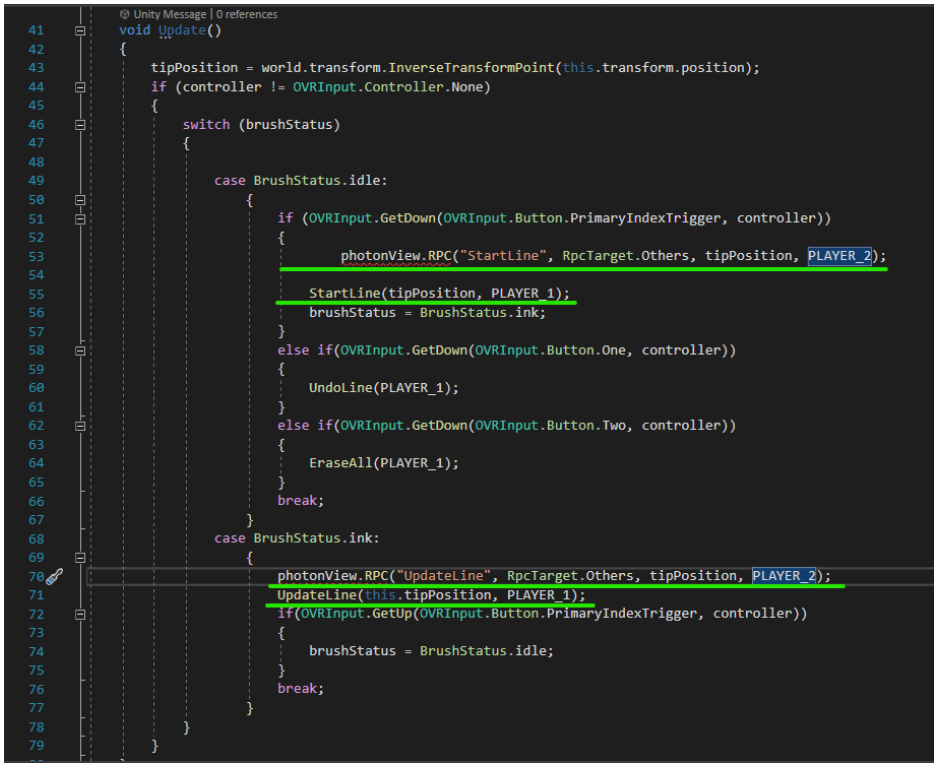


Ilustración 22: Ejemplo de PhotonView en Unity

Una de las limitaciones de estos scripts es que pueden enviar información limitada a través de la red ya que esta comunicación debe ser breve y eficiente, por lo que sirven únicamente para objetos simples. ¿Pero qué hay de objetos más complejos? Para poder realizar interacciones complejas con objetos complejos se debe utilizar combinaciones de técnicas diferentes para mantener la red poco saturada. En el caso de nuestra aplicación, para poder los trazos en tiempo real a diferentes usuarios, en vez de enviarles el objeto del trazo, se ha utilizado la siguiente herramienta clave que incluye photon, los RPC.

RPC es la sigla para los Remote Procedure Calls, que son un protocolo de comunicación a través de la red mediante paquetes TCP o UDP que permite llamar a funciones remotas en otros ordenadores conectados como si se llamasen de forma local, lo que permite distribuir carga computacional, comunicar equipos, entro otros. [20]

Con Photon, las RPC sirven para llamar a que se ejecuten funciones desde un cliente a otro cliente conectado en el servidor. Por ejemplo, el usuario que está dibujando, puede utilizar una RPC para hacer que la función que dibuja los trazos en su equipo se ejecute en los equipos de otros usuarios simultáneamente y de este modo, se renderice el trazo en todos los clientes de la aplicación sin tener que compartir el objeto de tipo trazo que de otra forma hubiese generado mucha más carga en la red.



```
Unity Message | 0 references
void Update()
{
    tipPosition = world.transform.InverseTransformPoint(this.transform.position);
    if (controller != OVRInput.Controller.None)
    {
        switch (brushStatus)
        {

            case BrushStatus.idle:
                {
                    if (OVRInput.GetDown(OVRInput.Button.PrimaryIndexTrigger, controller))
                    {
                        photonView.RPC("StartLine", RpcTarget.Others, tipPosition, PLAYER_2);

                        StartLine(tipPosition, PLAYER_1);
                        brushStatus = BrushStatus.ink;
                    }
                    else if(OVRInput.GetDown(OVRInput.Button.One, controller))
                    {
                        UndoLine(PLAYER_1);
                    }
                    else if(OVRInput.GetDown(OVRInput.Button.Two, controller))
                    {
                        EraseAll(PLAYER_1);
                    }
                    break;
                }
            case BrushStatus.ink:
                {
                    photonView.RPC("UpdateLine", RpcTarget.Others, tipPosition, PLAYER_2);
                    UpdateLine(this.tipPosition, PLAYER_1);
                    if(OVRInput.GetUp(OVRInput.Button.PrimaryIndexTrigger, controller))
                    {
                        brushStatus = BrushStatus.idle;
                    }
                    break;
                }
        }
    }
```

Ilustración 23: Ejemplo de RPC en scripts

Canvas + UI Helpers

Canvas es un sistema de interfaces gráficas en 2D y 3D que permite crear un lienzo gigante que se muestra en la cámara de usuarios a modo de un menú, en donde se puede añadir elementos interactivos como botones y sliders, checks, scrolls, etiquetas y demás. Para nosotros es un sistema fácil e intuitivo sirve para mostrar al usuario un menú sencillo con el que pueda interactuar para poder conectarse o crear una sala en el modo multijugador.

De la interacción con el menú se encargan los UI Helpers que son en otras palabras punteros que salen de los mandos y que cumplen la función de hacer clic en los botones del menú cuando la superficie de este puntero toca un botón y a su vez el usuario pulsa un botón de interacción.

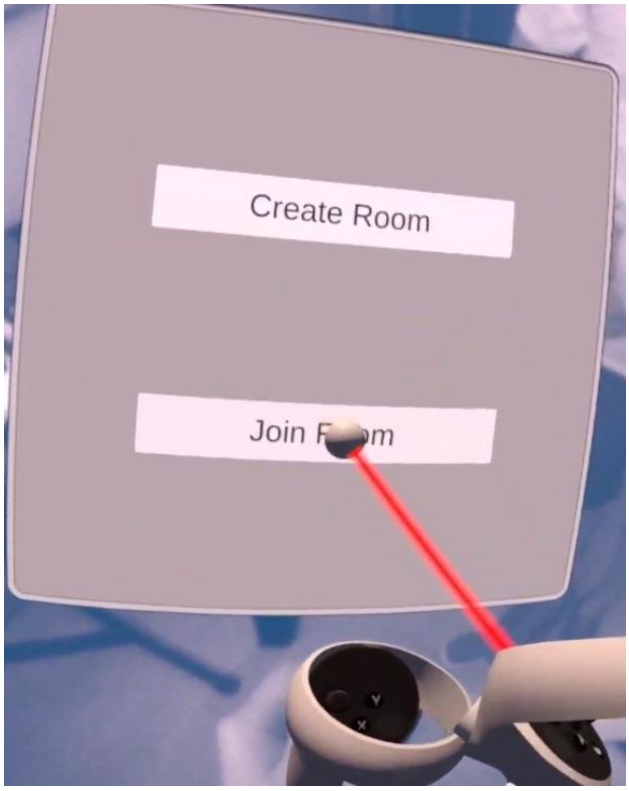


Ilustración 24: Menú creado con canvas + UI Helpers

Épica II: MR Online Drawing

Una vez claros y probados los conceptos básicos, se ha comenzado a implementar la aplicación a partir de las anteriores y separando en 2 épicas y por lo tanto en 2 partes diferentes según la estrategia seguida. La parte multiusuario online y la parte local en mono usuario.

La primera de ellas, llamada dentro de desarrollo MROnlineDrawing, consiste en llevar las pruebas de multijugador online con Photon aplicando la capa del passthrough de Oculus.

Esto presenta un desafío muy importante que es la sincronización espacial entre el escenario virtual y el mundo físico. Y es que tal y como se ver reflejado en los resultados de los sprints, esta es la tarea que más tiempo ha llevado. Consequentemente, para poder abordarla, se decidió resolver esta problemática a partir de diferentes aproximaciones.

Primera aproximación

Para la primera aproximación se intenta hacer lo más básico posible para tratar de comprender el problema que rodea a esta épica. Para ello se intenta responder a la siguiente pregunta:

¿Qué es lo que sucede si simplemente se activa el passthrough?

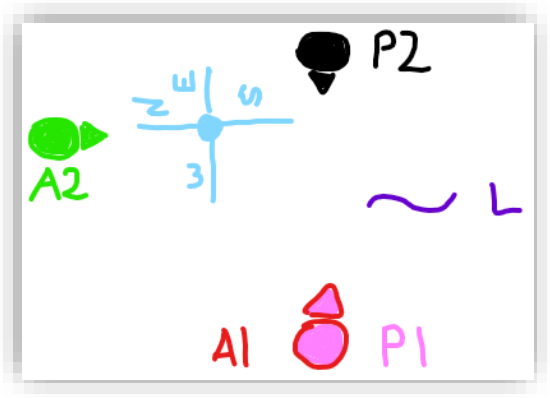


Ilustración 25:Perspectiva de la persona 1

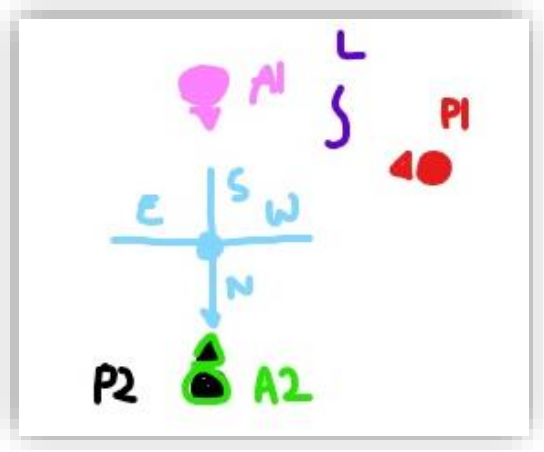


Ilustración 26:Perspectiva de la persona 2

- P1, P2: representan a la persona 1, la persona 2 y a dónde miran
- A1, A2: representan al avatar de la persona 1, de la persona 2 y a dónde miran
- L: es la línea que se pinta en el mundo virtual, en este caso, por la persona 1.
- La cruz representa el centro del mapa y su orientación, detallado como una brújula para saber dónde está el norte del mapa.

La conclusión es que efectivamente, los avatares se sincronizan con respecto al centro del mapa en el entorno virtual que no tiene nada que ver con las posiciones de las personas en el entorno real.

Segunda aproximación

La segunda aproximación fue intentar que el primer jugador pudiese elegir en la posición y rotación del avatar del segundo jugador mediante un objeto auxiliar. La problemática ahora viene dada por una de las limitaciones por parte del api que se mencionaban durante la primera épica, y es que Oculus sólo nos permite mover el ovrCameraRig y no a los elementos internos que contienen la posición real del jugador como el centerEyeAnchor. ¿Qué significa esto?

Cómo caso ideal tenemos al centerEyeAnchor posicionado a la par con el ovrCameraRig, cuando movemos el ovrCameraRig de posición el centerEyeAnchor se moverá exactamente donde esté el ovrCameraRig.

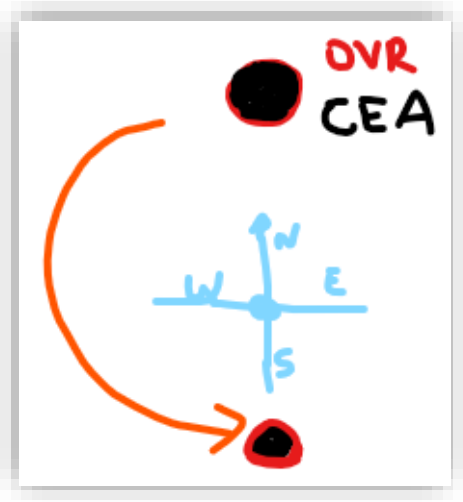


Ilustración 27:Ejemplo de movimiento del OVR cuando tiene la misma posición que el CEA

- OVR: ovrCameraRig
- CEA: centerEyeAnchor

Pero ¿Qué pasa en caso de que el jugador decida moverse? Si el jugador se mueve, el centerEyeAnchor cambiará su posición, pero el ovrCameraRig se mantendrá estático. ¿Y si ahora movemos el ovrCameraRig? Ambos objetos se moverán la misma distancia, pero conservarán su separación tal y como se ilustra en la siguiente imagen.

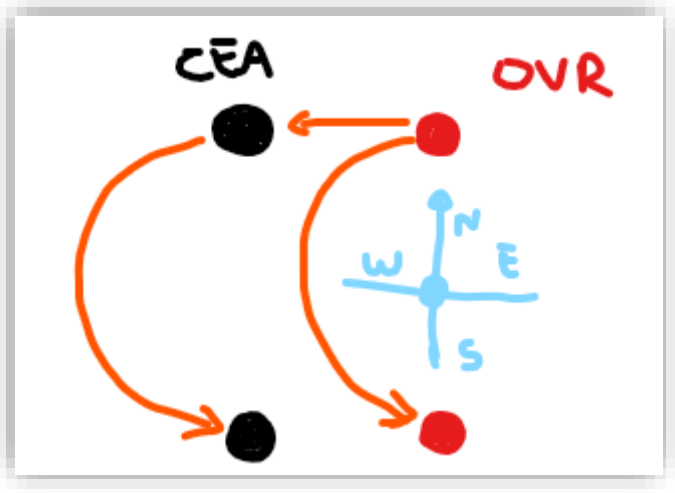


Ilustración 28:Ejemplo de movimiento del OVR cuando NO tiene la misma posición que el CEA

¿Se puede corregir esto? Durante las pruebas, se optó por crear un script que hiciese que el ovrCameraRig siguiese todo el tiempo la posición del centerEyeAnchor de forma controlada para que no se desplace de forma infinita, y aunque funcionaba con algún fallo de precisión, lo que no podía solventar este script es la rotación del ovrCameraRig que será una problemática que aparecerá más adelante.

Tercera aproximación

Para la tercera aproximación se realizó un experimento para centrar a los jugadores de forma no programática, es decir, se utilizó las herramientas de centrado que dispone todos los dispositivos Oculus integradas en el sistema operativo. Esta forma no tan elegante resolvía el problema de la rotación y de cierta manera el problema de la posición, pero no era algo que se podía considerar factible y usable para el público ya que se tenía que depender de una función externa de la aplicación y que además no daba resultados especialmente precisos.

Cuarta aproximación

Con la cuarta aproximación, se utilizó una técnica popular para sincronizar objetos virtuales con objetos reales. La idea es simple, cambiar la posición y rotación de un objeto con respecto a la posición del mando. Pero aplicado a nuestro caso parece que esta idea necesita muchos retoques.

Para desarrollar nuestra propia versión de sincronización a partir de este concepto base, se parte del hecho de que ahora nuestro mapa es el mapa global de la escena de Unity. Por lo tanto, la idea es transformar ese mapa global en un objeto local que contenga todos o casi todos los elementos de la escena. El motivo es simple, mover este objeto hará que se muevan todos sus objetos hijos y, por lo tanto, haremos que este mundo local sea el que se deba sincronizar con el usuario y no al revés.

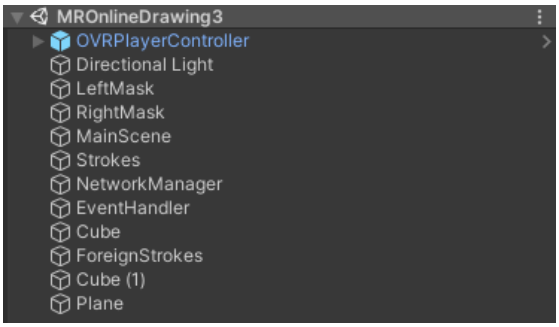


Ilustración 29: Antes del cambio

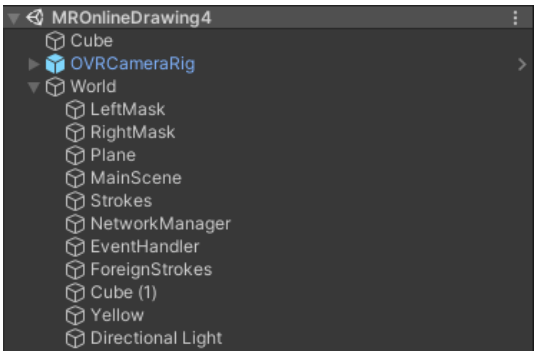


Ilustración 30: Después del cambio

Por lo tanto, la primera prueba de concepto de esta aproximación consistía en trasladar todos los objetos a una posición especifica relativa al centerEyeAnchor en vez de al ovrCameraRig, que por supuesto, tuvo éxito.

El siguiente problema de la cuarta aproximación es, como ya se había mencionado previamente, tener que girar elementos. Y es que, con la primera prueba de esta casuística, se probó a girar directamente el ovrCameraRig en vez del mapa ya que, si funcionaba bien, la complejidad se reducía drásticamente tal y como se pensaba inicialmente. Pero, de nuevo, la problemática del cameraRig hacía que el avatar del jugador girase en torno al ovrCameraRig, y no en su propio eje, como cabía de esperarse.

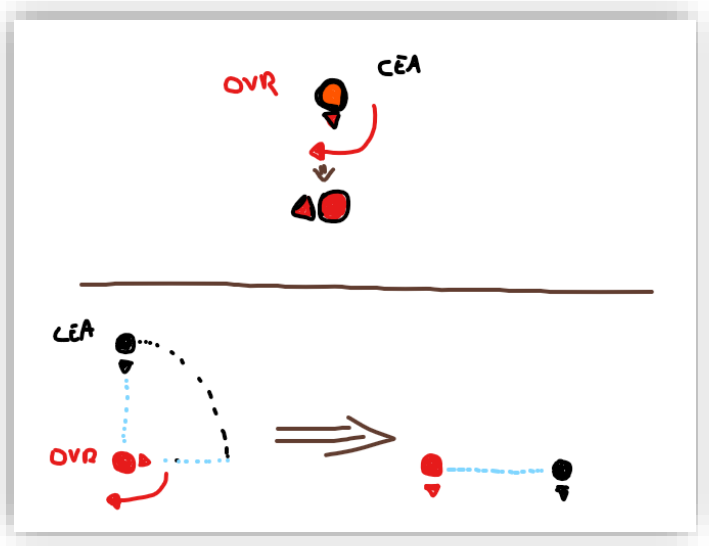


Ilustración 31:Rotación cuando el OVR y el CEA tienen la misma posición VS cuando tienen una posición diferente

Pero para la siguiente prueba se descubre la función clave de rotateAround(), que permite rotar elementos en un eje que el programador puede especificar, que en nuestro caso sería rotar el

mundo local entorno al centerEyeAnchor. De esta forma se ahorra hacer algunos cálculos intermediarios para realizar la rotación.

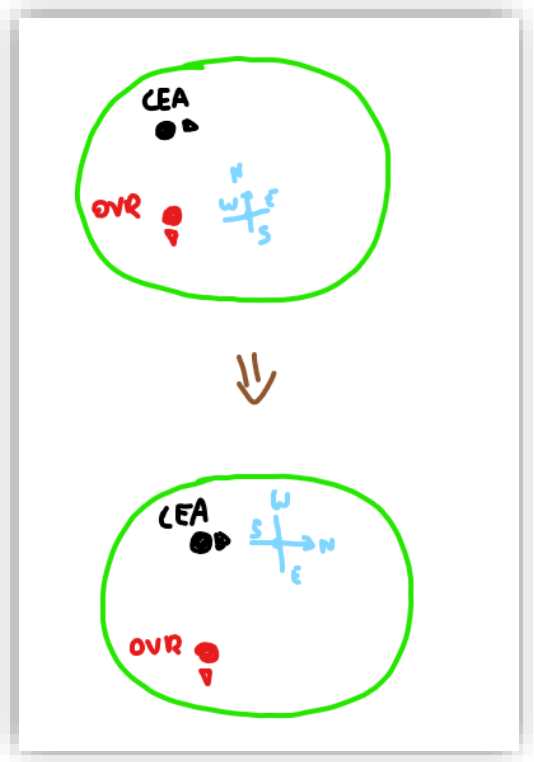


Ilustración 32: Solución final

Como se puede observar, el ovrCameraRig está no se encuentra en la misma posición del centerEyeAnchor, pero en este paso, hemos decidió desacoplarnos del ovrCameraRig y hacer todas las operaciones de forma relativa al centerEyeAnchor que, a fin de cuentas, es el elemento que representa al usuario.

Una vez implementada la solución final que resuelve el problema de la sincronización, se ha añadido un sistema complementario que ayude al usuario a tener referencias visuales para realizar la sincronización de una forma más sencilla.

Y de esta forma concluye la primera parte de la épica, la segunda parte consiste en agregar los elementos que se desarrollarán en la siguiente épica a la parte del multijugador online, pero como se puede observar en los anexos de los sprints, esta primera parte ha requerido mucho más tiempo del estimado durante su tallaje, por lo que la segunda parte se ha quedado pendiente por desarrollar.

Épica III: DrawingAnimator

La idea detrás de esta épica, como ya se ha mencionado anteriormente, es dotar a la aplicación la capacidad de poder crear diseños rápidos con un pincel para luego poder animarlos a través de varias líneas temporales diferenciando dos tipos de trazos: Un **trazo dinámico** que desaparece cuando se cambia de línea temporal y un **trazo estático** que permanece cuando se cambia de línea temporal.

Si separamos las tareas del modo monousuario podemos observar varios comportamientos como dibujar, animar, almacenar y borrar líneas. Para ello se ha creado diferentes scripts que se encarguen de un solo objetivo, de esta manera se separan responsabilidades y se crea un código más mantenible

BrushHandler

El script responsable de dibujar los trazos se ha llamado BrushHandler, que contiene las funciones necesarias para poder gestionar los trazos que se van a dibujar.

Por un lado, contiene una máquina de estados que permite diferenciar que función se está ejecutando actualmente. Los estados son:

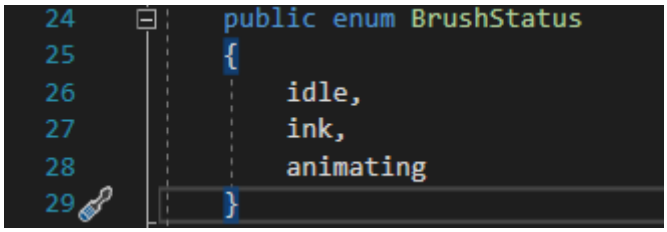


Ilustración 33: Máquina de estados de BrushHandler

- Idle: Cuando el pincel no hace nada. Este estado sirve para preparar el pincel para un nuevo trazo. Esto quiere decir que se crea el objeto de un nuevo trazo que puede ser dinámico o estático y se almacena en una lista compartida de objetos de tipo trazo como metadatos y adicionalmente se almacena una nueva línea en forma de Line Renderer en un objeto de Unity llamado Paints que contendrá y mostrará todos los trazos que pinte el usuario. Además, está a la escucha del cambio de línea temporal para que cuando esto suceda, se difumine el último conjunto de trazos línea temporal, y se transparente totalmente el resto de los trazos de otras líneas temporales. Durante este estado el usuario puede cambiar el color del pincel.

- Ink: Cuando el pincel está dibujando. Este estado sirve para actualizar el trazo que se está dibujando
- Animating: Cuando se está ejecutando una animación. Este estado sirve para bloquear la acción del pincel cuando una animación está en curso.

ClipHandler

Es el script responsable de gestionar todo el proceso de almacenado de animaciones y reproducción de las mismas.

Contiene también una máquina de estados que permite gestionar todas las acciones de este script

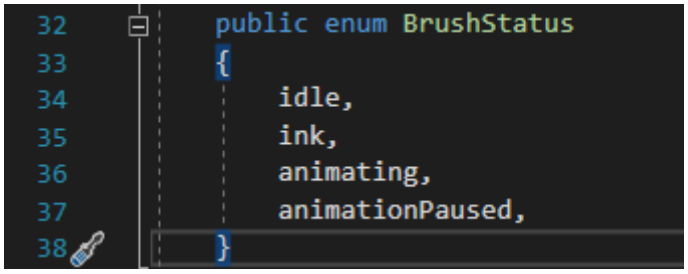


Ilustración 34: Máquina de estados de ClipHandler

- Idle: Cuando el pincel no hace nada. Este estado sirve para iniciar un contador del tiempo que transcurre entre que se empieza un trazo y se acaba para saber cuánto tiempo debe durar la animación. Además, en caso de que se vaya a iniciar la reproducción de una animación llama a las corrutinas creadas para que ejecuten las animaciones guardadas en orden y a la velocidad que haya sido registrada al momento de ser pintada por el usuario.
- Ink: Cuando el pincel está dibujando. Este estado sirve para aguardar mientras se está dibujando un trazo para evitar comportamientos inesperados.
- Animating: Cuando se está ejecutando una animación. Este estado sirve para aguardar mientras se está realizando una animación para evitar comportamientos inesperados.
- AnimationPaused: Cuando se ha acabado de animar un grupo de trazos de una línea temporal. Este estado sirve para reproducir el siguiente conjunto de trazos de la próxima línea temporal y la velocidad de reproducción.

EraserHandler

Es el script responsable de reiniciar la escena. Cuando recibe el input correspondiente, se encarga de vaciar los objetos que contienen los trazos y las animaciones y de reiniciar las variables compartidas entre los distintos scripts.

Strokes

Es el script que contiene el listado de trazos, y que comparte la información con el resto de los scripts. Este script se adjunta al objeto de Unity llamado Strokes que contiene los objetos Paints y Animations que contienen respectivamente los trazos y las animaciones representadas por Line Renderers.

Épica IV: Mejoras App

En esta última parte del desarrollo el objetivo es realizar varias pequeñas mejoras a la aplicación actual para las funcionalidades que cumplen las expectativas de los objetivos planteados y replantear, rediseñar y desarrollar de nuevo funcionalidades que se ha visto que pueden encajar mejor a los usuarios si se adoptan de otra forma. De entre los cambios más destacados se encuentran los siguientes:

Sistema de cambio de colores del pincel

Consiste en una lista de colores en memoria de la cual el pincel toma los colores de forma rotativa cada vez que el usuario ejecuta una acción determinada.

Modo de animación clásico de forma ininterrumpida

Tal y como estaba implementado el modo de animación clásico, las líneas se pintaban secuencialmente dentro de su propia línea temporal, y una vez se acababan de animar todas estas líneas, el usuario tenía que avanzar con un botón a la siguiente línea temporal para que se difuminaran las líneas dinámicas de la actual línea temporal y se pintaran las líneas de la nueva línea temporal. Pero ahora, la nueva lógica nos dice que todo se pinte de corrido, es decir, que el usuario no tenga el control sobre si se pinta la siguiente línea temporal, si no que se anime directamente.

Menú mejorado

Tal y como se aplica la lógica de la punta del pincel, el menú ahora es un objeto que se ata al mando del jugador para que este siga su trayectoria y que de esta forma no sea un objeto flotante del mapa que, depende de cómo se mueva el usuario, acabe dificultando su uso.

Mejoras visuales

Para poder mostrar de una forma simple al usuario que acción se está realizando actualmente, se han añadido diferentes indicadores visuales a los mandos que se han creado manualmente a través de blender.

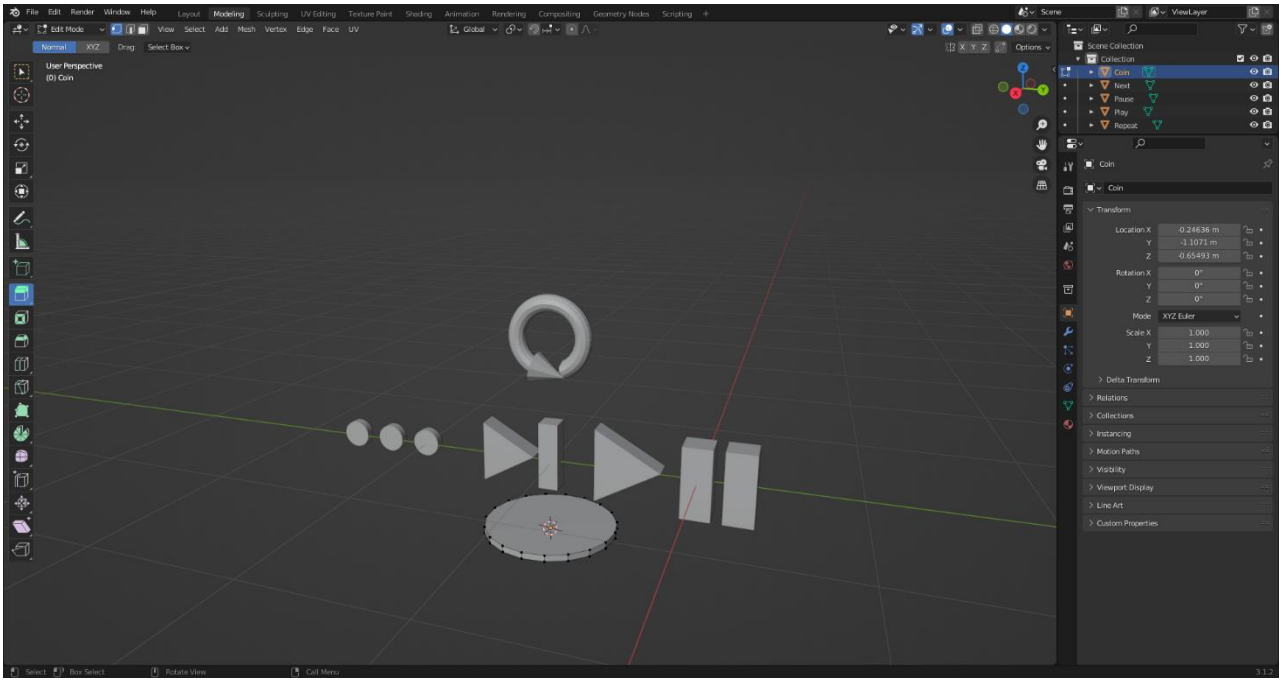


Ilustración 35: Assets personalizados creados en Blender

Una línea temporal por cada trazo dinámico

Consiste en que ahora cada vez que se pinta un trazo, este cree una nueva línea temporal difuminando así los trazos dinámicos previos sin esperar a que el usuario le dé al botón de cambiar de línea temporal.

Manual de usuario

Modo monousuario

CONTROLES

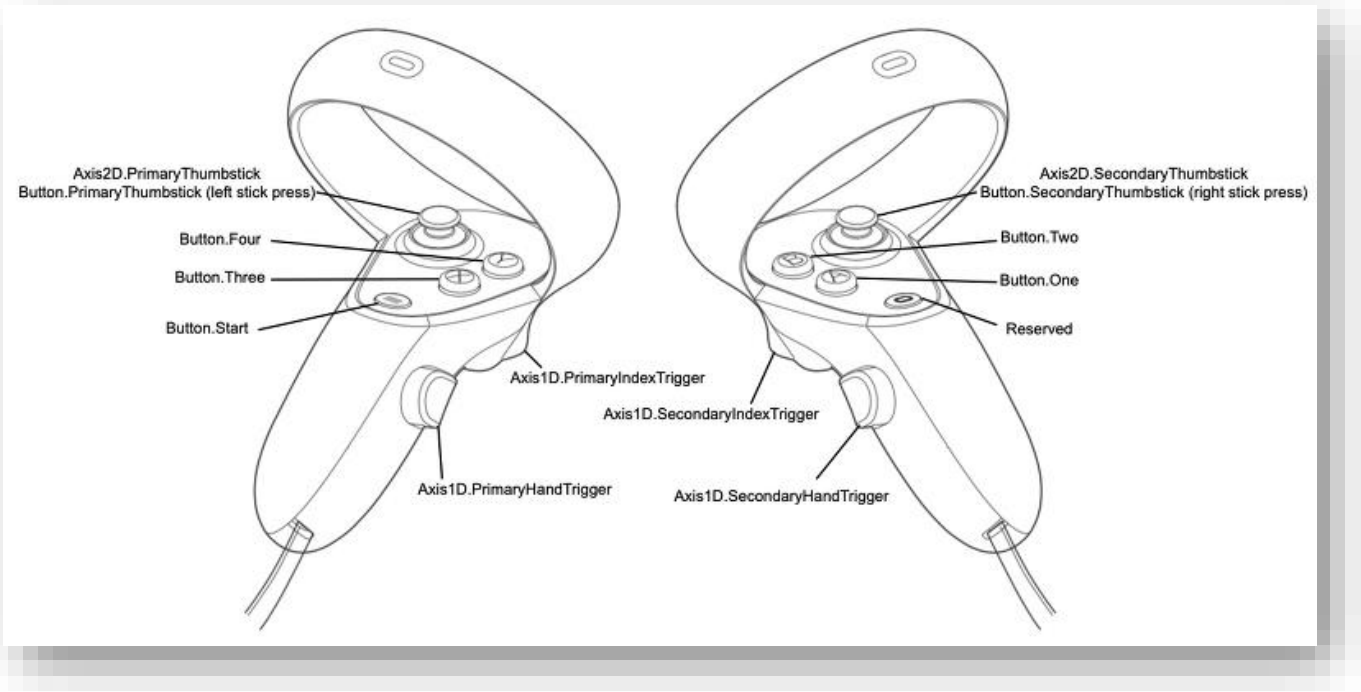


Ilustración 36: Ilustración de los mandos y controles de Oculus Quest 2

Mando Derecho:

- **SecondaryIndexTrigger:** Sirve para dibujar un trazo dinámico. Cuando presionas él botón ocurren 2 eventos. El primero es que se crea una línea temporal que almacena la línea que se va a dibujar a continuación. La segunda es la pintada de la línea. Cuando se suelta el botón se guarda la línea que se ha pintado y automáticamente se difumina ligeramente para dar sensación de que corresponde a otra línea temporal. Después de soltar el botón se puede dibujar otra línea estática o dinámica.
- **SecondaryHandTrigger:** Sirve para dibujar un trazo estático. Al igual que en el primer caso cuando se presiona el botón ocurren 2 acciones. La primera es introducir el nuevo trazo en una línea temporal común a otras líneas estáticas. Y lo segundo es pintar la línea. Cuando se suelta el botón se guarda la línea. Acto seguido se puede tanto pintar

otra línea estática que entere en la misma línea temporal como pintar una línea dinámica en otra línea temporal.

- **SecondaryThumbstick**: Depende de la dirección a la que se mueva ocurren diferentes eventos.
    - Hacia arriba y hacia abajo: Cambiar el color del pincel.
    - Izquierda y derecha: Agrandar y empequeñecer el pincel.

Mando Izquierdo:

- **PrimaryIndexTrigger:** Inicialmente desactivado si no se ha pintado previamente ningún trazo. Una vez se haya pintado algo en la escena, al pulsarse se inicia la animación instantánea, es decir, se pintarán instantáneamente todos los trazos pertenecientes a una línea temporal. Si se le vuelve a dar al botón, se pasará a pintar los trazos de la siguiente línea temporal eliminando todos los trazos dinámicos de la actual línea temporal. En caso de que no haya más animaciones se repetirá la animación desde el principio.
- **PrimaryHandTrigger**: Al igual que con el anterior botón está inicialmente desactivado si no se ha pintado previamente ningún trazo. Una vez se haya pintado algo en la escena, al pulsarse se inicia la animación clásica, es decir, se pintarán paulatinamente todos los trazos pertenecientes a una línea temporal. Si se le vuelve a dar al botón, se pasará a pintar los trazos de la siguiente línea temporal eliminando todos los trazos dinámicos de la actual línea temporal. En caso de que no haya más animaciones se repetirá la animación desde el principio.
- **Button.Four**: Sirve para reiniciar la escena. Puede pulsarse en cualquier momento.

USO

Video demo: https://youtu.be/n5Ghz2Uf6ZA

Mini demo de la reproducción clásica: https://youtu.be/6RBy8zcYSDg

Estado inicial:

Mando izquierdo con indicador de que se podría reproducir una escena, pero como no hay ningún trazo dibujado al iniciarse la aplicación no hace nada.

Mando derecho con el pincel preparado para dibujar.

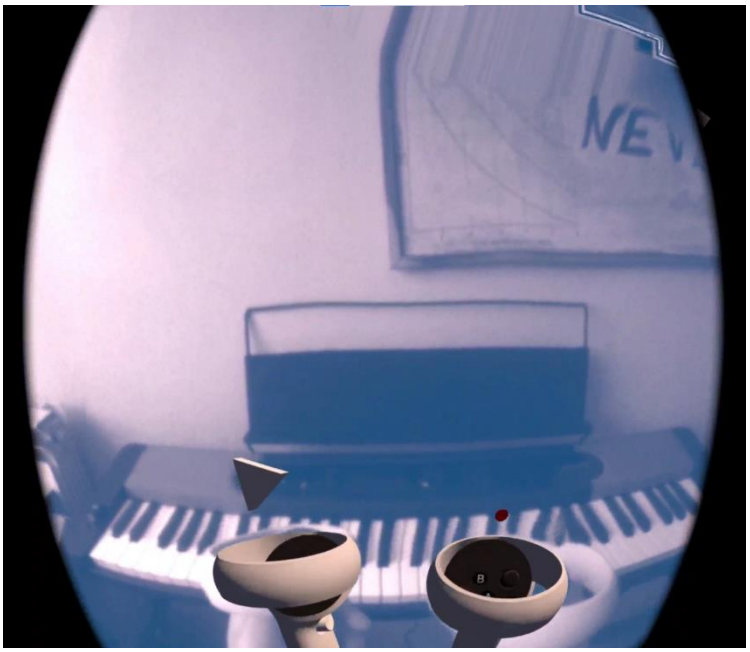


Ilustración 37: Estado inicial de la aplicación

Realizar un trazo dinámico:

Con el mando derecho presionando el **SecondaryIndexTrigger** se dibuja un trazo dinámico que en este caso forma el número 3.

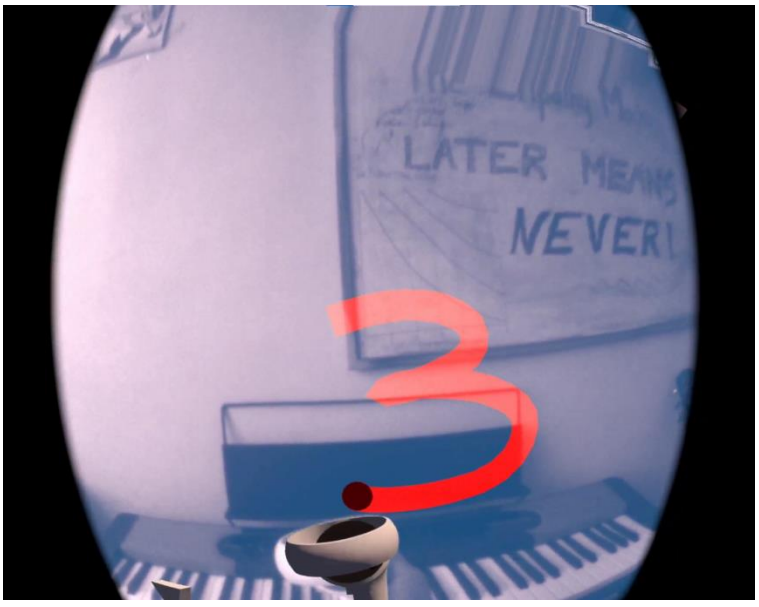


Ilustración 38: Se pinta una línea dinámica que representa el número 3

Una vez que se suelta el botón de pintar, se almacena el trazo en una nueva línea temporal y se vuelve de un material semitransparente.

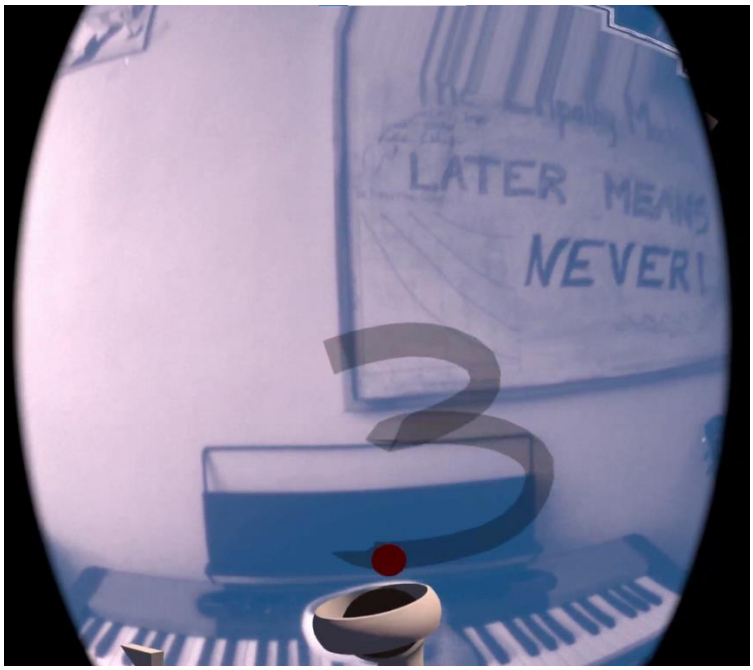


Ilustración 39: Trazo dinámico finalizado

Realizar un trazo estático:

Con el mando derecho presionando el **SecondaryHandTrigger,** se dibuja un trazo estático que, en este caso, forma una pista de bailes con luces. Este trazo permanecerá durante toda la escena y **no** se difuminará una vez acabado de pintar.

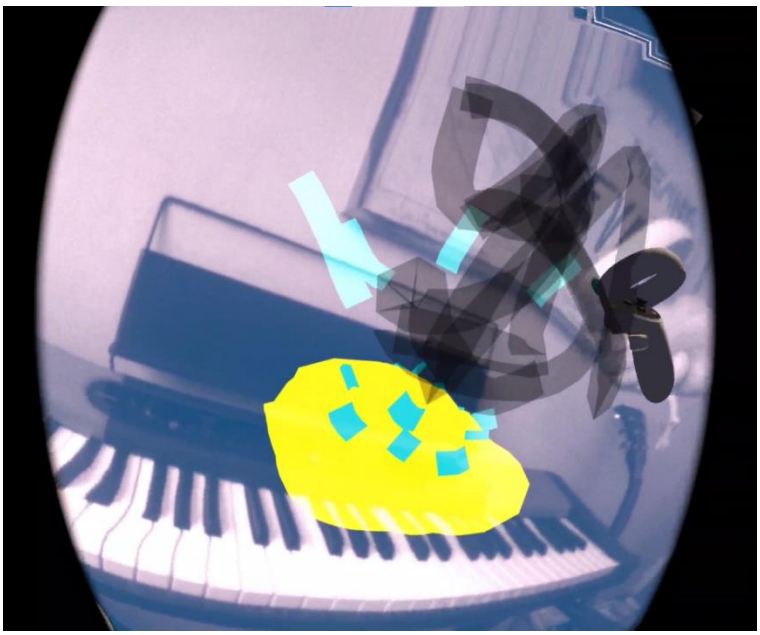


Ilustración 40: Pista de baile con luces dibujada con trazos estáticos

Cambio de color y tamaño de pincel:

Con el **SecondaryThumbstick** del mando derecho, si se presiona hacia arriba o hacia abajo se cambia el color del pincel y de derecha a izquierda el tamaño del mismo.

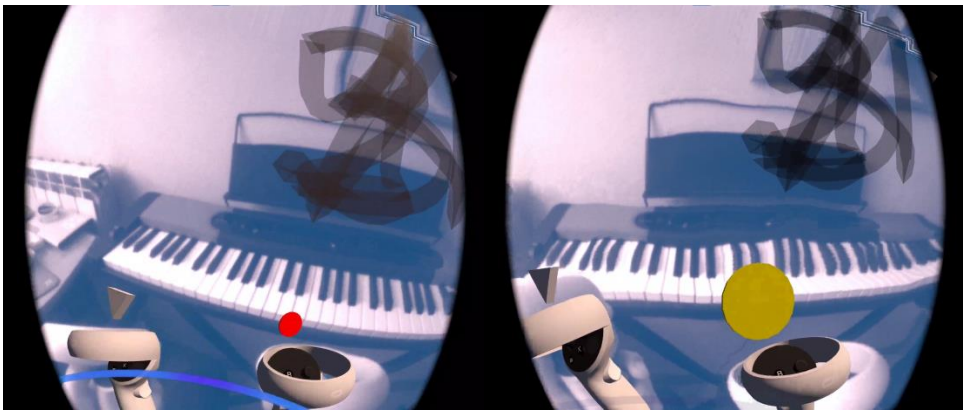


Ilustración 41: El color y el tamaño del pincel han cambiado

Iniciar la reproducción de la animación:

Con el mando izquierdo y con un trazo dibujado se procede a presionar el botón **PrimaryIndexTrigger** para iniciar la animación.

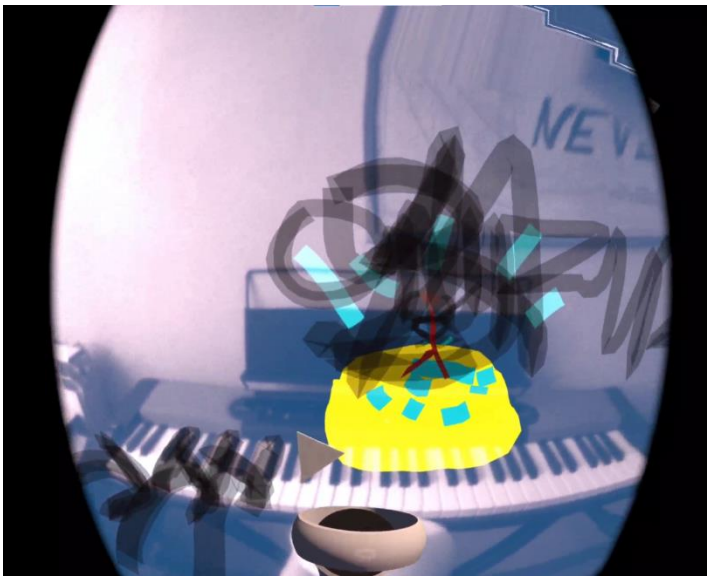


Ilustración 42:Mando izquierdo con indicador de que se pueden reproducir las animaciones

Siguiente animación:

Cuando aparezca el indicador de que se puede avanzar a la siguiente animación, se presionará el mismo botón de **PrimaryIndexTrigger** para iniciar la siguiente animación.

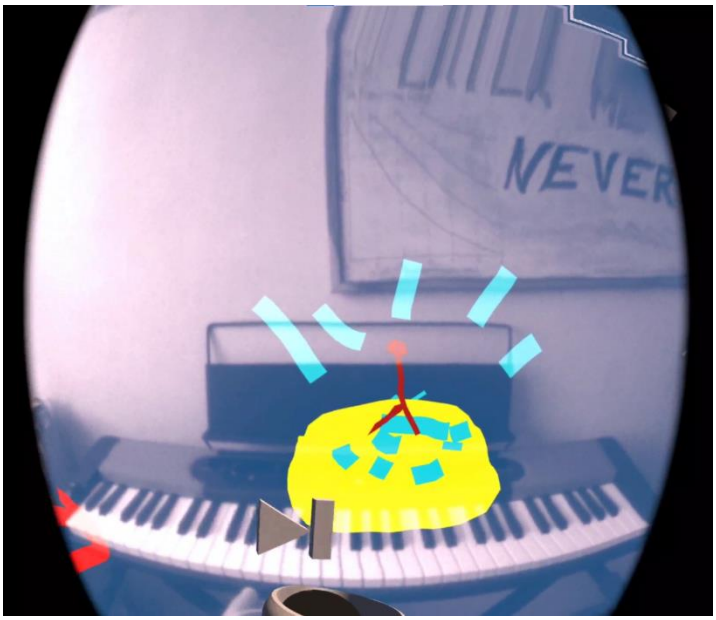


Ilustración 43: Mando izquierdo con indicador de que se puede reproducir la siguiente animación

Reproducir desde el principio:

Una vez acabas todas las animaciones, aparecerá un indicador de volver a reproducir la escena, para poder reiniciar todas las animaciones y empezar a animar desde 0. Para ello, volver a presionar botón **PrimaryIndexTrigger**.

Ilustración 44: Mando izquierdo con el indicador de reproducir de nuevo en la escena final

Pintar una vez acabada las animaciones:

Una vez aparezca el indicador de reproducir de nuevo, la segunda opción a parte de repetir las animaciones es seguir prototipando. En este caso se utilizarán los controles del mando derecho previamente mencionados para seguir realizando trazos.

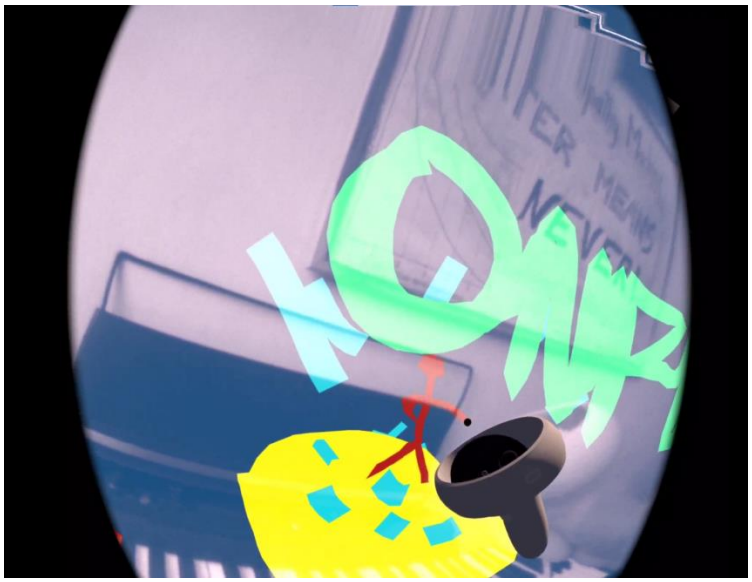


Ilustración 45: Se realiza unos trazos adicionales después del fin de la reproducción de las animaciones

Modo multiusuario

CONTROLES

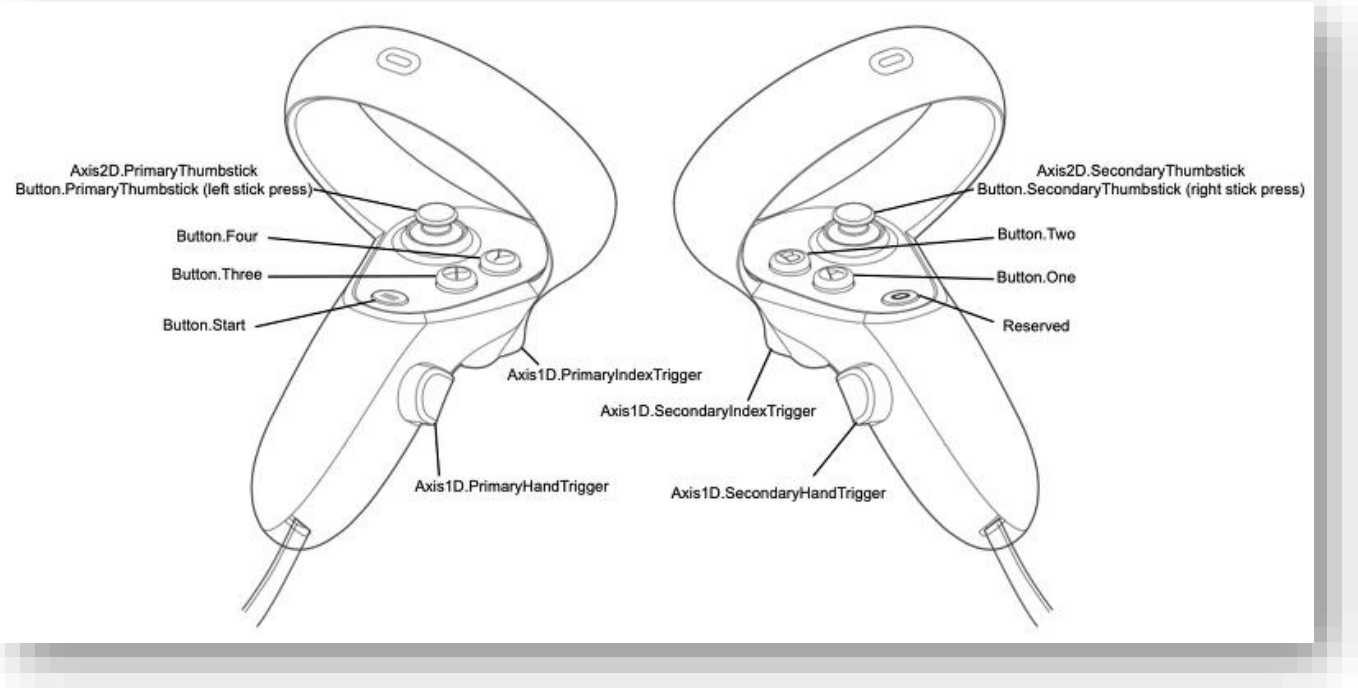


Ilustración 46: Ilustración de los mandos y controles de Oculus Quest 2

Mando Derecho:

- **SecondaryIndexTrigger:** Sirve para dibujar un trazo simple mientras se mantiene pulsado. Una vez se suelte, el trazo se termina. Si previamente se ha pulsado el botón del menú, sirve para seleccionar la opción del menú a la que se desea acceder.

Mando Izquierdo:

- **Start:** Sirve para activar o desactivar el menú. En el menú se puede seleccionar la opción de unirse o crear una sala.
- **PrimaryHandTrigger:** Desactivado si el usuario no se ha unido o creado previamente a una sala. Una vez activado sirve para rotar y trasladar el mapa a una posición y dirección correspondiente a la posición y rotación de la cabeza del usuario.

USO

Video Demo: https://youtu.be/Ih7kTu3rEJQ

Estado inicial:

Cada usuario está en un mundo poblado de 2 cubos que servirán posteriormente para la sincronización de posicionamiento y dirección.

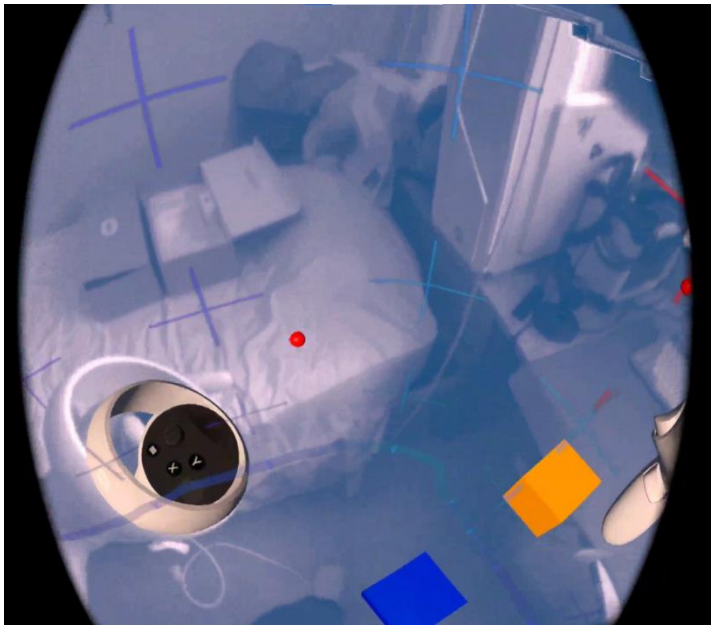


Ilustración 47: Estado inicial con cubos de sincronización

Ilustración 48: Segundo jugador con la aplicación arrancada simultáneamente

Menú de conexión:

Al pulsar el botón **Start** se activa el menú donde se podrá seleccionar si se crea una sala o si se desea unir a una ya existente. Para poder estar en la misma sala, uno de los jugadores tiene que crear una sala, y el otro jugador debe unirse esta.

Ilustración 49: Menú básico

Desincronización inicial:

Una vez ambos jugadores están en la misma sala, cada jugador puede observar una esfera de color rojo en el centro de su mundo virtual que representa la posición en tiempo real del otro jugador, que como se ha visto anteriormente, no tiene nada que ver con la posición del mundo real.

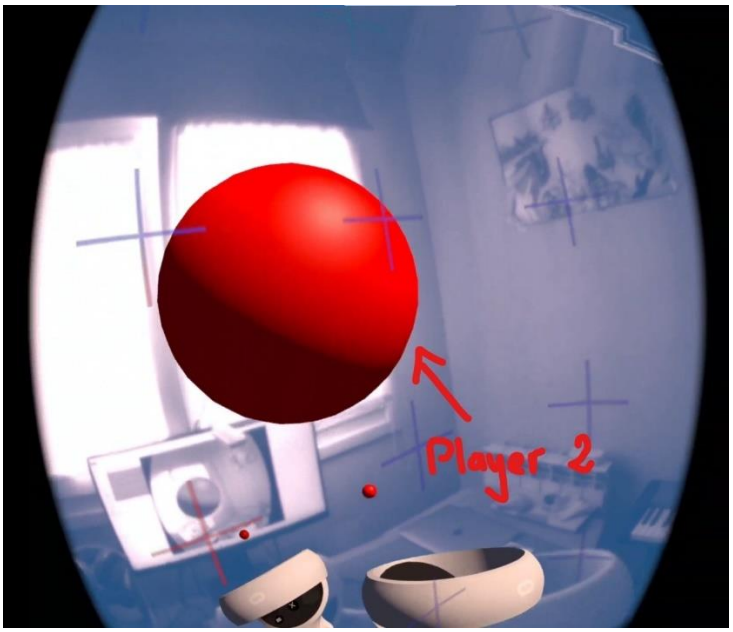


Ilustración 50: Objeto que representa al segundo jugador en el mundo del primer jugador

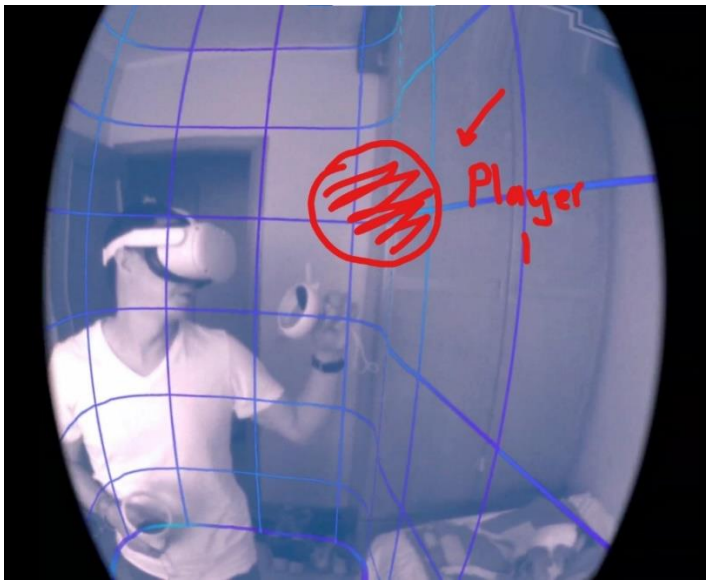


Ilustración 51: Representación de la posición del primer jugador en el mundo del segundo jugador de acuerdo con lo señalado por el segundo jugador

Cómo sincronizar:

Gracias a los cubos de sincronización que ambos jugadores tienen, se puede mover el mundo de forma relativa a la posición del jugador con el botón **PrimaryHandTrigger** del mando izquierdo de cada jugador. Este movimiento siempre será colocar el cubo azul debajo del jugador y el amarillo enfrente. En este caso, para sincronizar tanto la posición como la dirección de los jugadores se ha hecho lo siguiente:

1. El segundo jugador mueve los sus cubos de forma que se posicionen en el centro del área física donde se está realizando las pruebas, de este modo tendrán un espacio generoso para moverse.
2. Acto seguido el primer jugador deberá pedir instrucciones al segundo jugador para saber en qué parte del mundo físico está ubicado el cubo azul y el cubo amarillo.

Ilustración 52: Segundo jugador enseña en que posición el primer jugador debe colocarse

3. Una vez se sepa la localización física, el primer jugador tendrá que ubicarse encima del cubo azul y luego mover sus cubos para que coincidan con la posición tanto del cubo azul como el del cubo amarillo que ha indicado el segundo jugador.

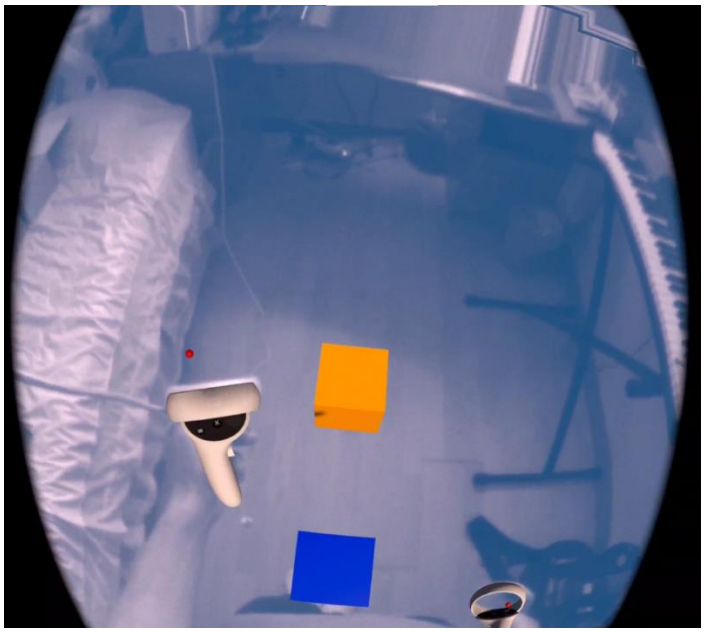


Ilustración 53: El primer jugador se ha ubicado donde el segundo jugador le ha señalado y ha movido sus cubos.

4. Después de estos pasos se procederá a mirar si las esferas ahora se encuentran en la posición de cada jugador.

Ilustración 54: La esfera del segundo jugador ahora está encima del segundo jugador

5. En caso de que haya algo de descuadre se repite el proceso.

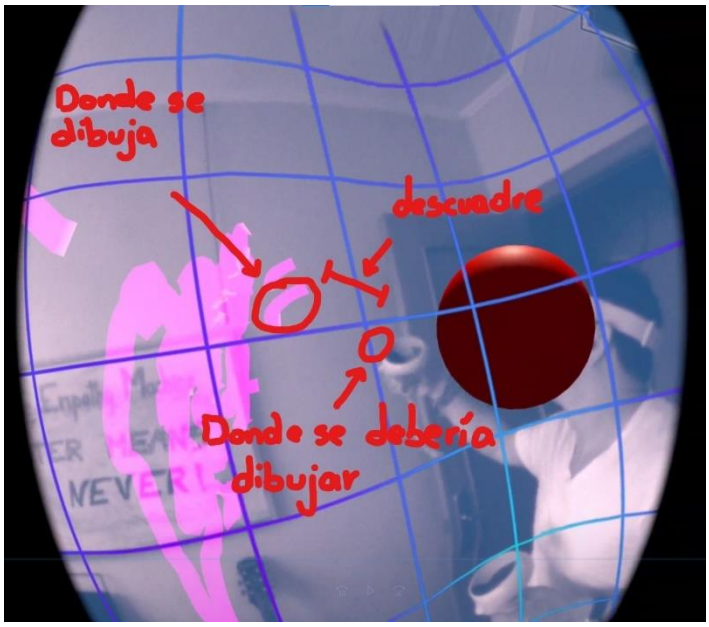


Ilustración 55: Hay un pequeño descuadre entre la posición real y la virtual

Prototipado colaborativo simple:

Una vez ambos jugadores estén sincronizados con respecto a sus avatares en el mundo virtual, pueden realizar trazos de forma colaborativa y simultánea

Ilustración 56:Jugador 1 y jugador 2 prototipando simultáneamente

Pruebas de usuario

El último capítulo de este proyecto es una serie de pruebas que se hicieron a diferentes usuarios para demostrar el uso de la aplicación y comprobar que el objetivo de la usabilidad y a la vez, la utilidad de la aplicación, se estaban cumpliendo.

Las pruebas consistían en una primera demostración de cómo funciona la aplicación a partir de un ejemplo simple. Luego hacer que el usuario replique ese ejemplo para hacerse con los controles. Lo siguiente hacer que replique la aplicación de Beat Saber (popular juego de VR) para entender de que maneras diferentes los usuarios pueden utilizar la aplicación con un ejemplo conocido. Después que prototipe una aplicación que tenga en mente para demostrar si realmente la aplicación está cumpliendo su objetivo de permitir que los usuarios puedan prototipar fácilmente aplicaciones de todo tipo. Luego aplicar la técnica de Wizard of Oz para demostrar que la aplicación puede utilizarse para realizar pruebas de usabilidad. Y, por último, que aproveche la capacidad de realidad mixta para que prototipe otra aplicación que tenga en mente, pero esta vez aprovechando el entorno que le rodea.

RESULTADOS

Primer sujeto

Notas de la prueba: Esta parte fue realizada previa a la épica de Mejora App

Aplicaciones prototipadas:

- Beat Saber
- Aplicación libre
- Aplicación libre usando MR: Se ha intentado prototipar el clásico juego de Tower Defense a escala real utilizando diferentes elementos del entorno como mesas, sillas y el propio suelo para ubicar los diferentes elementos del juego.

Comentarios:

- Los controles son intuitivos, pero necesitan una mejora visual para saber cuándo se acaba la animación y cuando se puede dar al siguiente y cuando se puede volver a dibujar.
- La reproducción rápida debería ser más rápida o instantánea para tener mayor sensación de transición.

- Estaría bien continuar dibujando una vez acabada la animación.
- Añadir múltiples colores al pincel tendría un gran valor añadido.

Conclusiones:

A partir de esta demo salieron muchas ideas a implementar durante la épica de mejoras app con el fin de mejorar la experiencia de usuario y la usabilidad.

Segundo Sujeto

Notas de la prueba: Durante esta prueba se ha logrado recopilar una serie de fotos de muestra pero que debido a que Oculus es muy restrictivo con las medidas de privacidad la parte del passthrough está muy limitada y en vez de ver el entorno real en las capturas, el fondo es el azul estándar. Además, estas pruebas se han realizado después de implementar la épica de Mejoras App.

Aplicaciones prototipadas:

- Aplicación del ventilador: La idea de la aplicación del ventilador suele ser el primer ejemplo a presentar para que la gente se haga con los controles. Consiste en una aplicación que a partir de un guante con sensores, se puede regular el encendido o apagado del mismo.

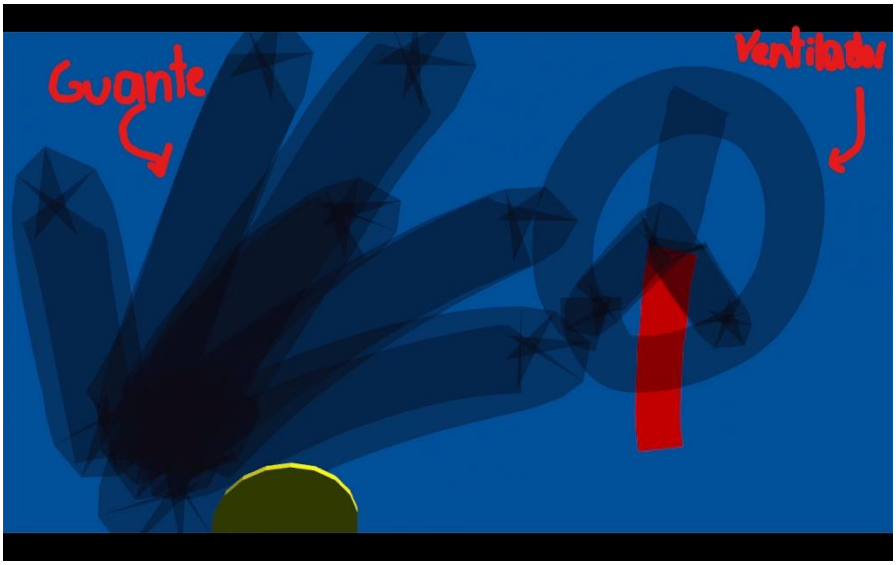


Ilustración 57: Preparación del prototipo. En este caso el guante y el ventilador están en posición inicial

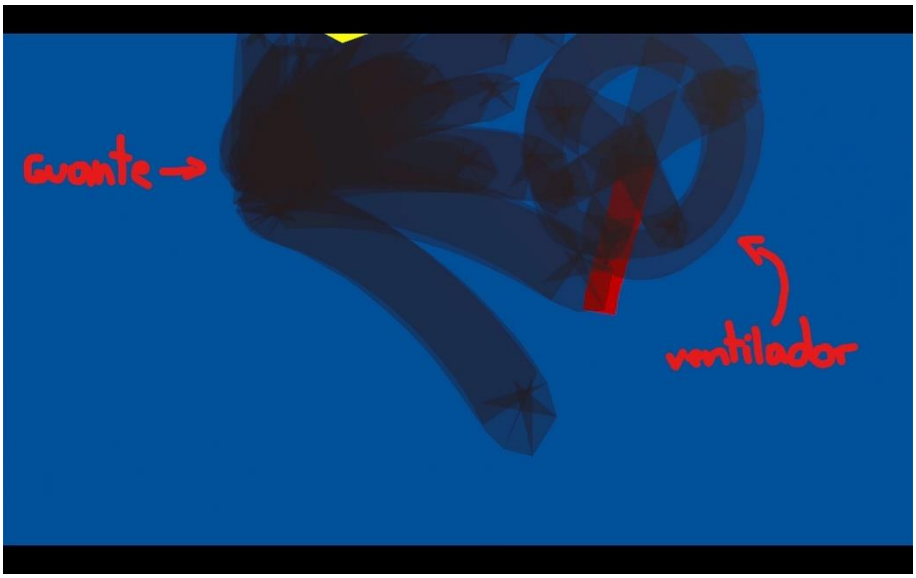


Ilustración 58: Cambio de línea temporal. Ahora se prototipa que el guante y el ventilador se cambien de posición inicial a activa cuando se animen las líneas.

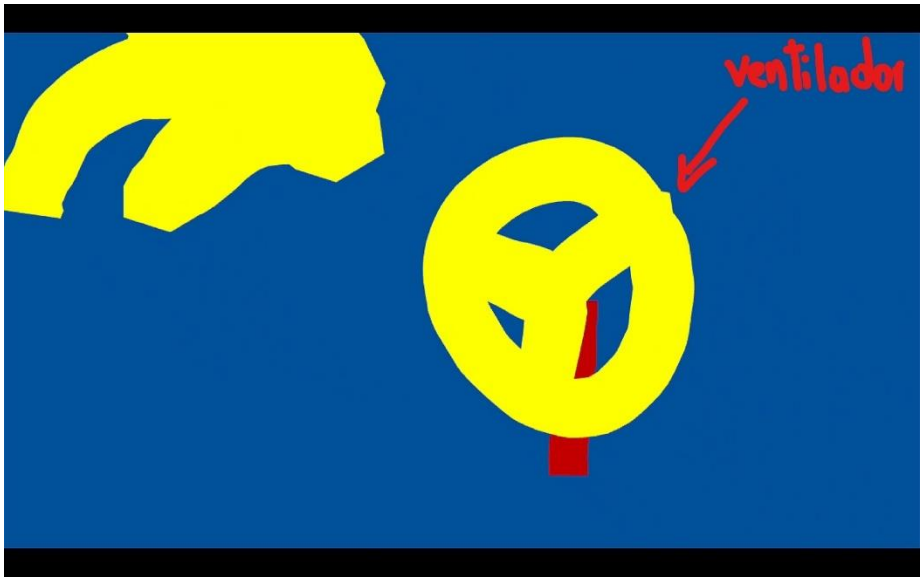


Ilustración 59: Animación final. El guante y el ventilador están en posición activa y la animación se ha acabado mostrando cómo las aspas del ventilador han cambiado de posición.

- Beat Saber
- Prueba de Wizard of Oz: Se dibujan una serie de elementos sin que el sujeto de prueba sepa donde se encuentran. Luego, se colocan las gafas al sujeto y se pide que interactúe con los elementos existentes. En este caso es una espiral y el usuario interpreta que la acción a realizar es introducirse en ella.
- Aplicación libre: Se trata de un juego de lucha con unos mandos especiales. En una pantalla aparecen 2 personajes pertenecientes a 2 jugadores. Los controles son atacar

y esquivar y cada vez que un personaje es atacado un mecanismo del mando es activado y un elemento de este salta de su posición fija en el mando.

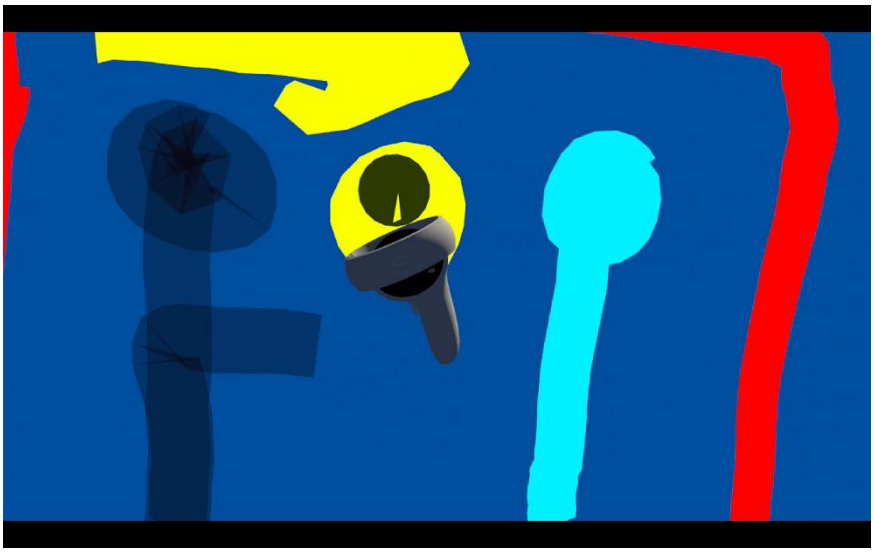


Ilustración 60: Se muestra como el usuario prepara la escena con dos personajes en una pantalla

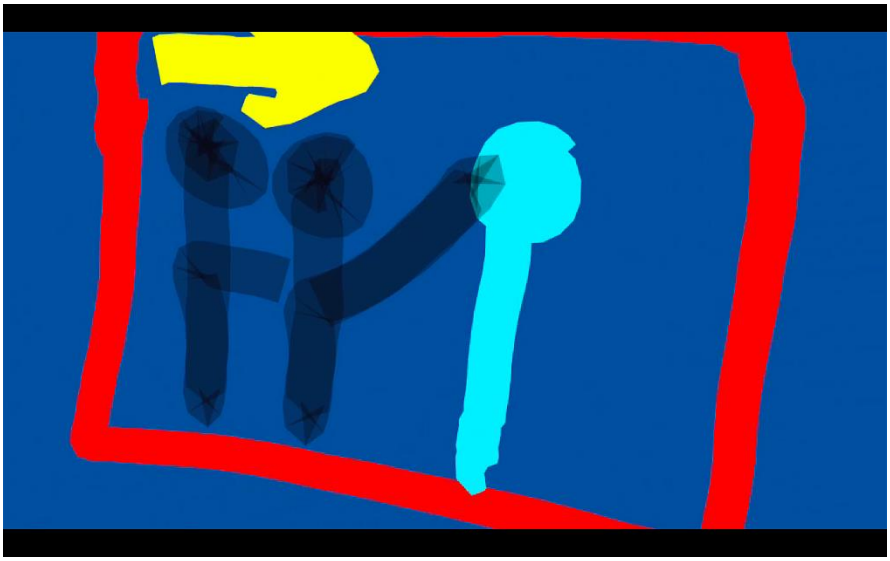


Ilustración 61: Se crea el efecto de que un personaje golpea a otro

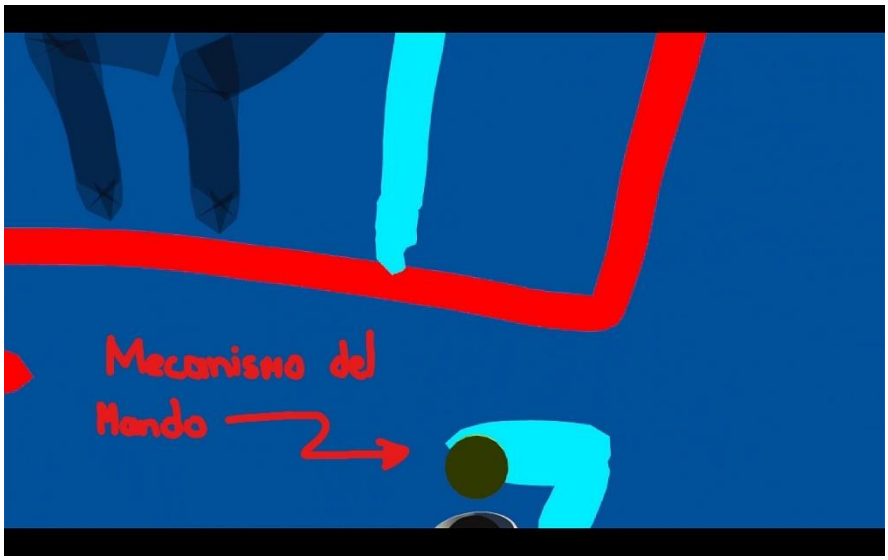


Ilustración 62: Se prototipa cómo el mecanismo del mando físico se activa.

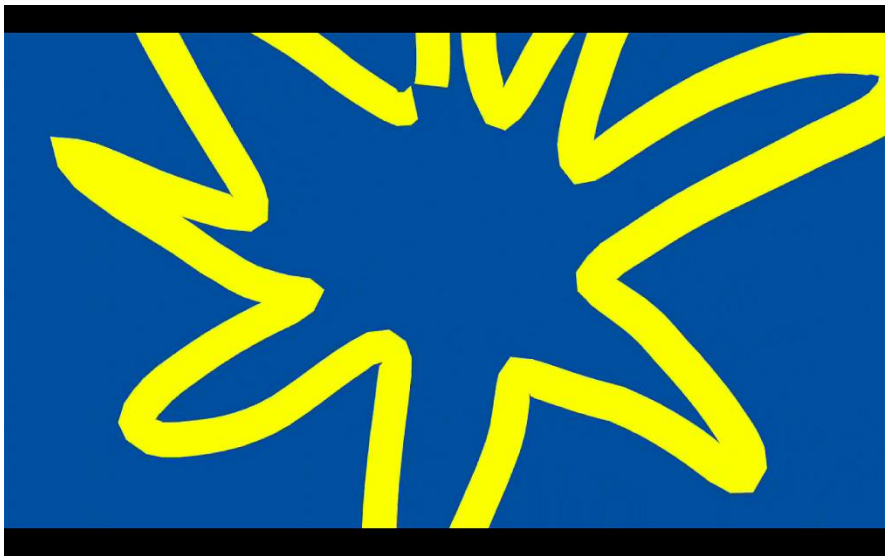


Ilustración 63: Fin de la escena

- Aplicación libre usando MR: Se ha creado una simple, pero muy explicativa y orientativa aplicación que trata de indicar a los usuarios que acciones deberían realizar en el entorno. En este caso, al usuario, mediante este prototipo, se le indica que debe coger una caja ubicada encima de una mesa, luego colocarla encima de una silla en posición horizontal y por último colocarla en posición vertical.

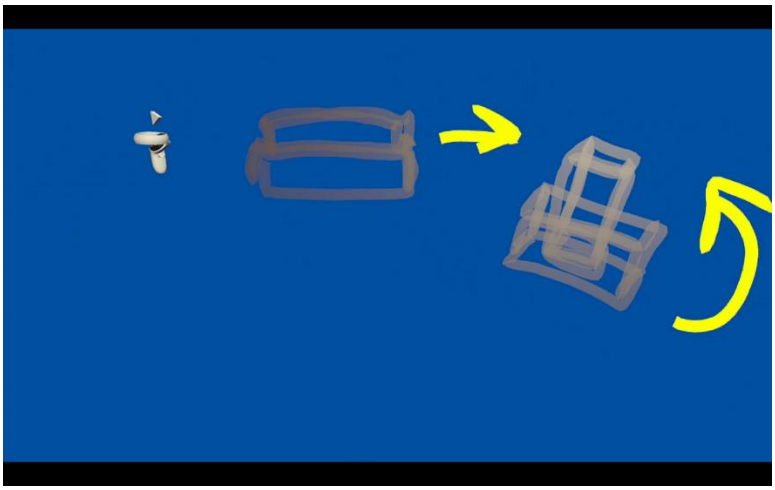


Ilustración 64: Se prototipa las instrucciones que deber realizar el usuario en el entorno real.

Ilustración 65:Caja en posición inicial

Ilustración 66: Caja en segunda posición siguiendo las instrucciones del prototipo

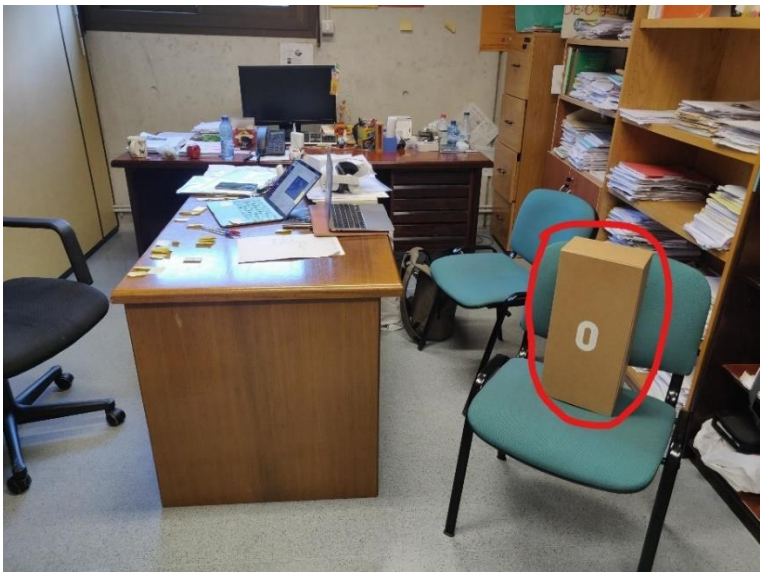


Ilustración 67:Caja en posición final siguiendo las instrucciones del prototipo

Comentarios:

- Cuesta hacerse a los controles al principio, pero luego se vuelven muy fáciles de utilizar

Conclusiones:

Gracias a la última prueba se logró demostrar uno de los grandes potenciales de la aplicación. El sujeto logró demostrar con su simple ejemplo de las cajas cómo se puede utilizar estas herramientas para realizar distintos prototipos aprovechando el entorno, en este caso a modo de guía de interacción con objetos reales.

Tercer y cuarto sujeto

Notas de la prueba: Estas pruebas se realizaron en conjunto con el tercer y cuarto usuario y bajo las mismas condiciones que la segunda prueba y también utilizando la versión de monousuario.

Aplicaciones prototipadas:

- Beat Saber: En este caso un usuario prototipaba el juego con un solo mando y una vez acabado se le colocaba las mismas gafas al segundo usuario para que intentase jugar con el prototipo creado sin mandos mientras que el primer jugador con el otro mando reproducía las animaciones que se veían en las gafas.
- Aplicación libre:
    - El primer usuario realizó un prototipo de un juego basado en un escenario en donde varios jugadores divididos en dos equipos deben conquistar el mapa pintando todo el territorio posible con el color de su equipo.
    - El segundo usuario realizó un prototipo sobre otro juego basado en una carrera de barcos de vela cuya mecánica principal es colocar la vela en una dirección favorable al viento para poder ganar más velocidad.

Conclusiones:

En este paso se pretendía y se ha logrado demostrar que la aplicación puede llegar a ser útil con más de un usuario aun usando sólo unas gafas de realidad virtual con mandos compartidos.

Próximos pasos

Como próximos pasos a corto plazo se tiene en cuenta terminar de implementar todas las funcionalidades de la aplicación monousuario en la de multiusuario, para así poder tener la experiencia completa con varios jugadores de forma simultánea.

A largo plazo se piensa más en la forma de evolucionar la aplicación a algo más parecido a un Unity en VR. Tal y como se explica en los objetivos del proyecto, esta será una versión más avanzada que incluya las primeras funcionalidades desarrolladas y sumando la capacidad de añadir a la escena objetos tridimensionales además de dotarles de formas de interaccionar con otros objetos mediante un sistema de colisiones y físicas, y como último añadido, poder editar sus dimensiones, posición y rotación.

Con estos futuros cambios, se podrá tener una aplicación capaz de crear prototipos de forma algo menos rápida e intuitiva pero mucho más precisa.

Bibliografía y referencias

[1] “Envisioning Holograms”, M.Pell, Woodinville Washington USA, 2017.

[2] “XRDirector: A Role-Based Collaborative Immersive Authoring System” [online], Michael Nebeling, Katy Lewia, Yu-Cheng Chang, Lihan Zhu, Michelle Ching, Piaoyang Wang, Janet Nebeling, 2020, disponible en https://dl.acm.org/doi/10.1145/3313831.3376637

[3] “MagicalHands: Mid-Air Hand Gestures for Animating in VR” [online], Rahul Arora, Rubaiat Habib Kazi, Danny M. Kaufman, Wilmot Li, Karan Singh, 2019, disponible en https://dl.acm.org/doi/10.1145/3332165.3347942

[4] “¿Qué es el prototipado y cómo prototipar un producto?” [online], Elena Bello, 2021, disponible en https://www.iebschool.com/blog/que-es-el-prototipado-digital-business/.

[5] “Software prototyping” [online], de Wikipedia disponible en https://en.wikipedia.org/wiki/Software_prototyping

[6] “Wizard of Oz testing – a method of testing a system that does not yet exist.” [online] de Fiona Harwood disponible en https://www.simpleusability.com/inspiration/2018/08/wizard-of-oz-testing-a-method-of-testing-a-system-that-does-not-yet-exist/

[7] “Immersive authoring of Tangible Augmented Reality content: A user study” [online] Gnu A Lee, Gerard J. Kim, 2008, disponible en https://www.sciencedirect.com/science/article/abs/pii/S1045926X08000402#:~:text=Immersive%20authoring%20refers%20to%20the,checked%20first%20hand%2C%20in%20situ

[8] “Realidad virtual” [online] de Wikipedia disponible en https://es.wikipedia.org/wiki/Realidad_virtual

[9] “¿Qué es la realidad mixta” [online], Qian Wen, Dhurata Jahiu, Tim Sherer, Vinnie Tieto, Harrison Ferrone, Kimberly Frazier y David Coulter, disponible en https://docs.microsoft.com/es-es/windows/mixed-reality/discover/mixed-reality

[10] “¿Qué es la realidad mixta?” [online], 2020, disponible en https://www.ionos.es/digitalguide/online-marketing/vender-en-internet/realidad-mixta/

[11] “Unity Game Engine Guide: How to Get Started with the Most Popular Game Engine Out There” [online], 2020, disponible en https://www.freecodecamp.org/news/unity-game-engine-guide-how-to-get-started-with-the-most-popular-game-engine-out-there/

[12] “VR overview” [online], disponible en https://docs.unity3d.com/540/Documentation/Manual/VROverview.html

[13] “10 Great Tools For VR Development” [online], Aran Davies, https://www.devteam.space/blog/10-great-tools-for-vr-development/

[14] “Build Unity Games with Visual Studio” [online], disponible en https://visualstudio.microsoft.com/vs/unity-tools/

[15] “Blender 3.1 Manual – Introduction” [online], disponible en https://docs.blender.org/manual/en/latest/getting_started/about/introduction.html

[16] “Oculus Quest 2” [online] de Wikipedia, disponible en https://en.wikipedia.org/wiki/Oculus_Quest_2

[17] “Almost 80% of VR headsets sold in 2021 were an Oculus Quest 2” [online], Nicholas Sutrich, 2022, disponible en https://www.androidcentral.com/quest-2-2021-most-sold

[18] “Photon Engine For Multiplayer Games: Try it for Free (What You Need To Know)” [online], Bryan Wirtz, 2021, disponible en https://www.gamedesigning.org/engines/photon-multiplayer/

[19] “PUN V2 – Getting Started – Introduction” [online], disponible en https://doc.photonengine.com/en-us/pun/current/getting-started/pun-intro

[20] “Remote Procedure Call” [online], de IBM, 2022, disponible en https://www.ibm.com/docs/en/aix/7.1?topic=concepts-remote-procedure-call