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

Visita virtual a los búnkeres de Erratzu

Grado en Ingeniería Informática

Trabajo Fin de Grado

Ricardo Manuel Garcia Marreros

Jesús Villadangos Alonso

Pamplona, Lunes 07 de Septiembre de 2020

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Índice:

1 Introducción

La realidad virtual es una tecnología en auge basada en técnicas informáticas que combinan imágenes, sonidos y espacios simulados para generar experiencias multisensoriales extraordinarias.

Su aplicación es muy variada, desde el entretenimiento u ocio, pasando por el educacional hasta el estudio del comportamiento humano a nivel psicológico, todo ello creando una ilusión de presencia, partiendo de entradas visuales y táctiles, para así lograr engañar al sistema perceptual.

Este proyecto se basa en el desarrollo de un viaje en realidad virtual a través de los búnkeres más conocidos de Navarra, en este caso los de Erratzu, construidos después de la guerra civil a lo largo de todo el pirineo para defender de posibles invasiones.

Con la información proporcionada por espeleólogos e historiadores, este proyecto permitirá a las personas conocer las maravillas culturales e históricas que guardan estos lugares desde sus casas o algún museo donde se proyecte esta visita virtual, por medio de interacciones y un entorno digitalizado y simulado que proporcione una sensación de inmersión casi total.

Proyectos de este estilo permitirán a personas que, por alguna diversidad funcional, no les resulte fácil acceder a estas instalaciones, también se dirige a personas que deseen adquirir esta experiencia pero que no cuenten con los medios necesarios para visitarlas y por ultimo para nivel educativo.

2 Objetivos

2.1 Objetivo general:

Con el desarrollo de este software pretendemos realizar una visita virtual guiada por el bunker de Erratzu ubicado al norte de Navarra, el diseño del entorno debe ser lo más fiel posible a la realidad, de carácter histórico/educativo, para ello contaremos con mapas del lugar, grabaciones y testimonios del lugar.

Para un correcto funcionamiento nos centraremos en la calidad de la experiencia para el usuario, proporcionando una facilidad intuitiva del dispositivo a usar y además permitiendo la accesibilidad a personas con alguna diversidad funcional.

El acceso a este tipo de lugares es muy complicado, ya sea por su estructura arquitectónica o por preservación cultural, no todas las personas pueden llegar a conocer estos lugares aun teniendo un gran interés por visitarlo, por esta razón el uso de la realidad virtual puede crear esa ilusión de encontrarse en ese lugar y conocerlo, ya sea desde la comodidad de casa o desde un museo o instituto de enseñanza.

2.2 Objetivo particular:

Con este proyecto no pretendemos obtener un producto funcional para una empresa o para su venta, sino la investigación de todos los aspectos necesarios para desarrollar una aplicación de realidad virtual con todos los componentes necesarios para tener una inmersión casi completa.

3 Requisitos

3.1 Alcance

El software tiene como objetivo el uso educacional a nivel histórico, aportando así información real sobre los búnkeres, acontecimientos históricos, relevancia, experiencias, testimonios y anécdotas de la época en la que se construyeron.

Además, se pretende generar una experiencia agradable dentro del entorno virtual que anime a las personas a visitar el lugar y a vivir esa experiencia.

Como extensión se ha añadido herramientas que permiten el uso a personas con alguna disfuncionalidad motora, para que así su disfrute e inmersión sea mayor dado que el lugar real no tiene esa accesibilidad.

Cueva de Altamira

3.2 Requerimientos funcionales

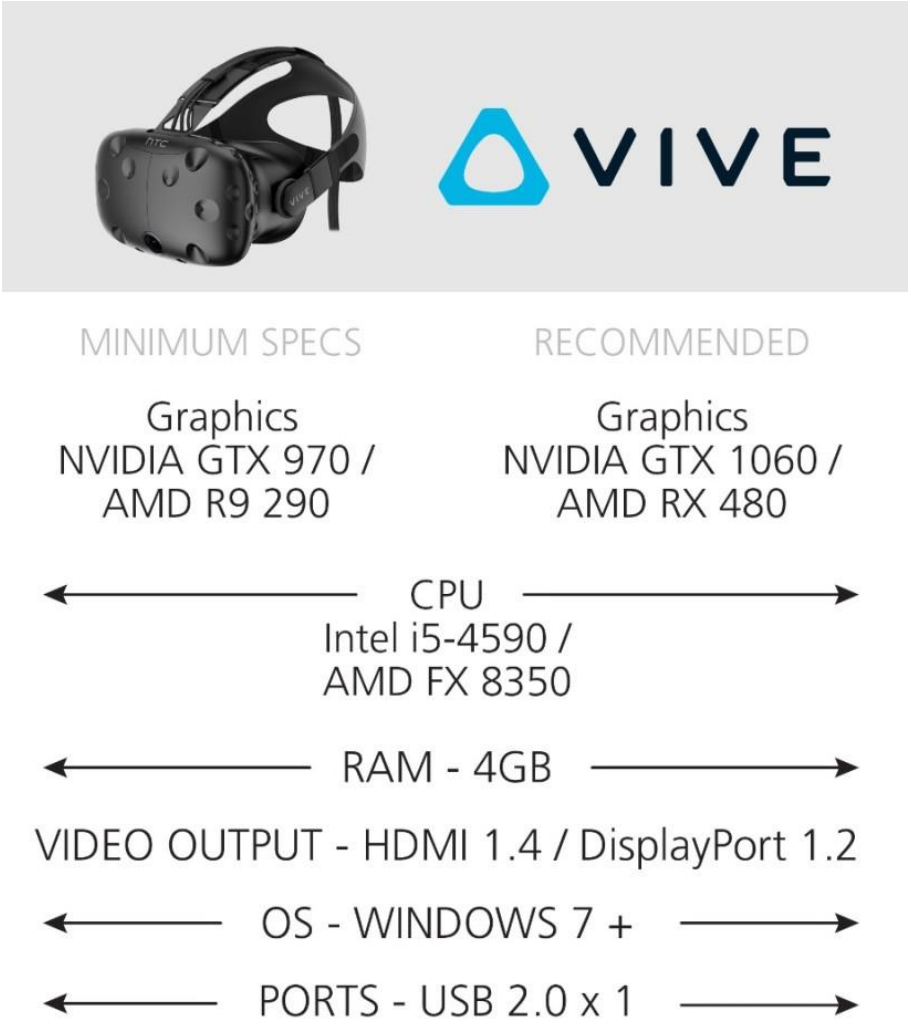
- El entorno virtual debe dar al usuario la posibilidad de interactuar con los objetos dentro de la simulación.
- La simulación debe ser capaz de generar eventos basándose en el lugar en el que se encuentre el usuario dentro del entorno virtual.
- El entorno virtual debe proporcionar una experiencia realista, basándose en la solidez y la textura de los elementos que la conforman, mientras que la simulación debe proporcionar los conocimientos educacionales correctos.

3.3 Requerimientos no funcionales

- El correcto funcionamiento del proyecto necesitará de unas gafas de realidad virtual, en nuestro caso VIVE PRO y unos dispositivos de control de movimiento.
- Requisitos mínimos del aparato de realidad virtual

3.4 Funcionalidad Global

- El software permitirá un aprendizaje fluido y de una duración de 10 a 15 minutos aproximadamente.
- No habrá textos, el aprendizaje será auditivo.



Especificaciones del visor [1]

4 Análisis y diseño

4.1 Evaluación y síntesis

Partimos de la inexperiencia de no haber realizado antes proyectos relacionados con la realidad virtual, es el primer contacto con Unity, pero con algo de experiencia en el uso de aplicación de diseño gráfico o 3D, como lo es Blender.

A nivel personal la selección de este proyecto se debe al interés por establecer nuevos conceptos de la realidad y como un entorno correctamente simulado puede generar nuevas experiencias a personas que jamás en su vida podrán vivirlas, ya sea por alguna dificultad motora o por alguna característica especifica del comportamiento, por ejemplo, autismo o personalidad introvertida.

Basándonos en los objetivos que buscamos con este proyecto, podemos establecer una serie de características que debe cumplir la simulación, por ello, dividiremos este proyecto en distintos aspectos importantes que debemos seguir:

- La gestión del proyecto
- El análisis del flujo del software
- El modelado 3D
- La implementación en Unity

4.1.1 Gestión de proyecto

1. Análisis y diseño
   1.1. Especificaciones
   1.2. Requerimientos
   1.3. Documentación
      1.3.1. Diseño funcional
      1.3.2. Documentación histórica
2. Diseños
   2.1. Modelado en Blender
3. Desarrollo
   3.1. Implementación en Unity
4. Pruebas
   4.1. Prototipo 1
   4.2. Beta
   4.3. Versión final
5. Instalación y despliegue
   5.1. Entorno virtual
6. Seguimiento y cierre

Una vez identificadas las funcionalidades del proyecto:

- El usuario tiene acceso a una interfaz que le proporciona una lista de áreas de los búnkeres que puede seleccionar para realizar la ruta.
- El usuario puede interactuar con objetos (coger y soltar).
- Existen objetos con otras funcionalidades (una linterna se enciende y apaga).
- El usuario recibe información auditiva cuando se acerca a distintos lugares dentro de la ruta.

Y haciendo uso de la documentación respectiva de la zona, pasamos a realizar el diseño del modelo 3D, el cual representa todo el entorno en el que el usuario se desenvolverá.

Es importante tener todo el material necesario para realizar el modelo con las medidas exactas, procurando la mayor semejanza con la realidad, el trabajo con Blender ha supuesto un gran tiempo de aprendizaje de las herramientas que proporciona y el modela ha supuesto un trabajo extra, ya que se partió desde cero y sin conocimiento del uso de la aplicación, sin embargo el desarrollo en realidad virtual va de la mano del diseño gráfico, por ello este trabajo ha sido enriquecedor para posteriores proyectos.

Una vez tenemos un modelo consistente, lo implementaremos en Unity corrigiendo errores gráficos y solventándolos, para posteriormente pasar al desarrollo del jugador, el cual más adelante explicare los distintos métodos para hacerlo.

La primera prueba del entorno se basa en un usuario, que una vez puesto las gafas de realidad virtual puede apreciar el modelo en 3D creado, el cual aún no posee las texturas necesarias.

Una vez conocemos el tipo de “Player” que queremos desarrollar, pasaremos a realizar las acciones de inmersión, en este caso las acciones motoras del jugador, moverse hacia delante, atrás, izquierda y derecha, dado que las librerías de gafas de realidad virtual que vamos a usar ya nos permiten el control de movimiento de la cabeza y las manos (partiendo de los mandos de control), todo ello gracias a los sensores de movimiento que poseen y a las estaciones base que las detectan.

Otra acción que posee el jugador será la de coger y soltar objetos que encontrará por el entorno (linternas, tablas, bichos, etc.), algunos de ellos también tendrán una funcionalidad, en el caso de una linterna, encender o apagar, permitiendo así al usuario tener visibilidad dentro del búnker, todo ello muy intuitivo y mejorando la inmersión del usuario.

La segunda prueba permite al usuario dentro del entorno el interactuar con distintos elementos colocados ahí, además de moverse desde un punto inicial hasta uno final del búnker.

La falta de ciertos materiales como las descripciones y testimonios auditivos, retrasaron la implementación de ciertas características, como eventos guías en distintos puntos del búnker, sin embargo se realizó un prototipo de prueba, usando una esfera

que reprodujese un audio aleatorio de información sobre el bunker, obtenida de un reportaje en internet.

Los eventos se han establecido como coordenadas centro en las que un usuario al acercarse en un radio establecido, esta reproducirá un audio informativo y si el usuario se aleja, esta se pausara y en cuanto vuelve a aproximarse, el audio se reproducirá desde donde se quedó.

Hay un tiempo estándar establecido para que una persona mantenga información auditiva y pueda reestablecerla, como si fuese un tema dentro de una conversación, teniendo esto en cuenta, si el usuario está fuera de la zona por mucho tiempo, el audio volverá a iniciarse desde el principio si este volviese a ese lugar, de la misma manera, si entra en otra zona distinta dentro del bunker, el nuevo audio se reproduciría y el anterior se establecería desde el inicio en caso volviese para así mantener la coherencia en la captación de información.

La última prueba, versión final, dado que muchos materiales se obtuvieron muy tarde, esto no permitió la posibilidad de rehacer el búnker con los nuevos datos, por ello se realizó otro espacio sesgado del búnker al que se le aplico las texturas y correcciones correspondientes, dando así la posibilidad de hacer una interfaz de selección.

La interfaz de selección presenta una lista de áreas de los búnkeres, está creada para posicionarse dentro de una de ella. En total en la zona de Erratzu hay nueve búnkeres de distinta longitud y estructura. La lista permitiría ir a uno de estos búnkeres y realizar la ruta, sin embargo en esta versión se ha adecuado a la selección del bunker con texturas básicas y con sus fallos estructurales (lucernas situadas en el techo, que no se apreciaban en los mapas, además de dibujos y escritos en las paredes) y una versión reducida del búnker con las texturas correspondientes, bien situadas, y con una estructura más acorde a la realidad.

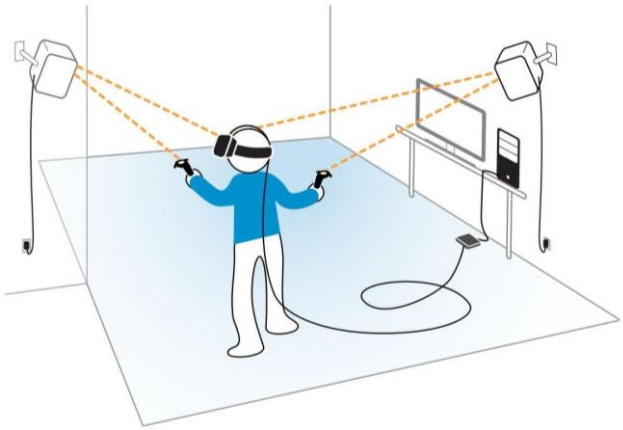
De esta manera la versión final es más funcional y abre la posibilidad de hacer algunas mejoras al modelo y añadir otras zonas de los búnkeres para darle un uso educacional.

La versión final no es un prototipo que pueda salir al uso público ya que aún posee fallos y posibles mejoras, las cuales más adelante podrían mejorarse:

- Adición de otras zonas de los búnkeres con sus respectivas texturas
- Adición de elementos que mejoren la inmersión, hojas en el suelo, bichos, ranas, etc.
- Una interfaz más llamativa, con un mapa desplegable del área que se va a seleccionar.
- Un concepto más ambicioso seria el representar todo el monte con los búnkeres en su interior y además de la interfaz de selección, poder usar la herramienta de transportarse para moverse directamente a puntos del monte donde estén ubicadas las entradas de las áreas.

4.1.2 Flujo del software

El usuario tiene que estar informado del correcto uso de los dispositivos de control de movimiento, también debe conocer su entorno real de movilidad y tener asistencia para evitar accidentes.



Una vez dentro de la simulación, el usuario tendrá una pequeña interfaz que le indicará donde se encuentra y que áreas del bunker puede visitar.

Tras haber seleccionado el área de visita, el usuario estará ubicado frente al bunker listo para empezar el recorrido.

Si el usuario está atento, sobre un muro o en el suelo encontrará una linterna que podrá usar para moverse por dentro.

Es el mismo entorno oscuro que motiva al usuario a buscar un objeto que le permita iluminar el camino que desea realizar.

Una vez dentro del bunker hay unas salas, las cuales al ingresar activaran unos eventos informativos, los cuales dan una explicación del uso que se les daba a esas estructuras, características del bunker, hechos históricos, anécdotas de personas de la zona y curiosidades.

Al finalizar el recorrido, el usuario volverá a ver la interfaz y tendrá de la opción de seleccionar otra área de los búnkeres de la zona o de realizar el mismo recorrido.

Diagrama de Flujo

Proceso de interacción del usuario con el entorno simulado, desde el instante en el que se pone los cascos y arranca la aplicación hasta el momento en el que desee finalizarla.

La aplicación permite al usuario libertad para realizar la ruta como el desee, sin embargo tendremos que realizar distintas actividades dependiendo de sus decisiones.

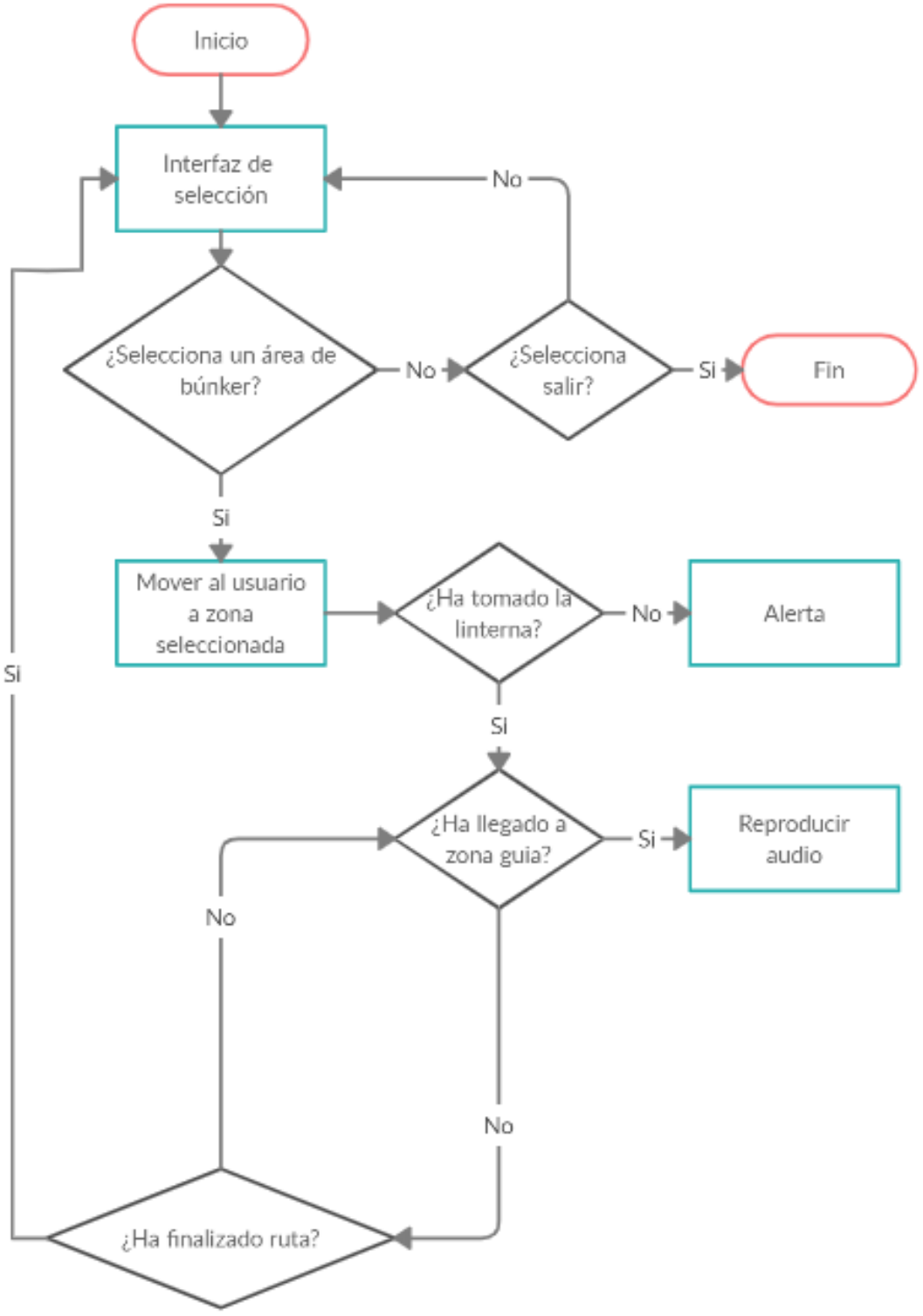


Diagrama de Clases

Establecido en base a los componentes de las clases creadas en Unity, además de la incorporación de funcionalidades creadas en los scripts correspondientes, para facilitar el uso estas operaciones, se añaden los scripts como componentes de los objetos creados.

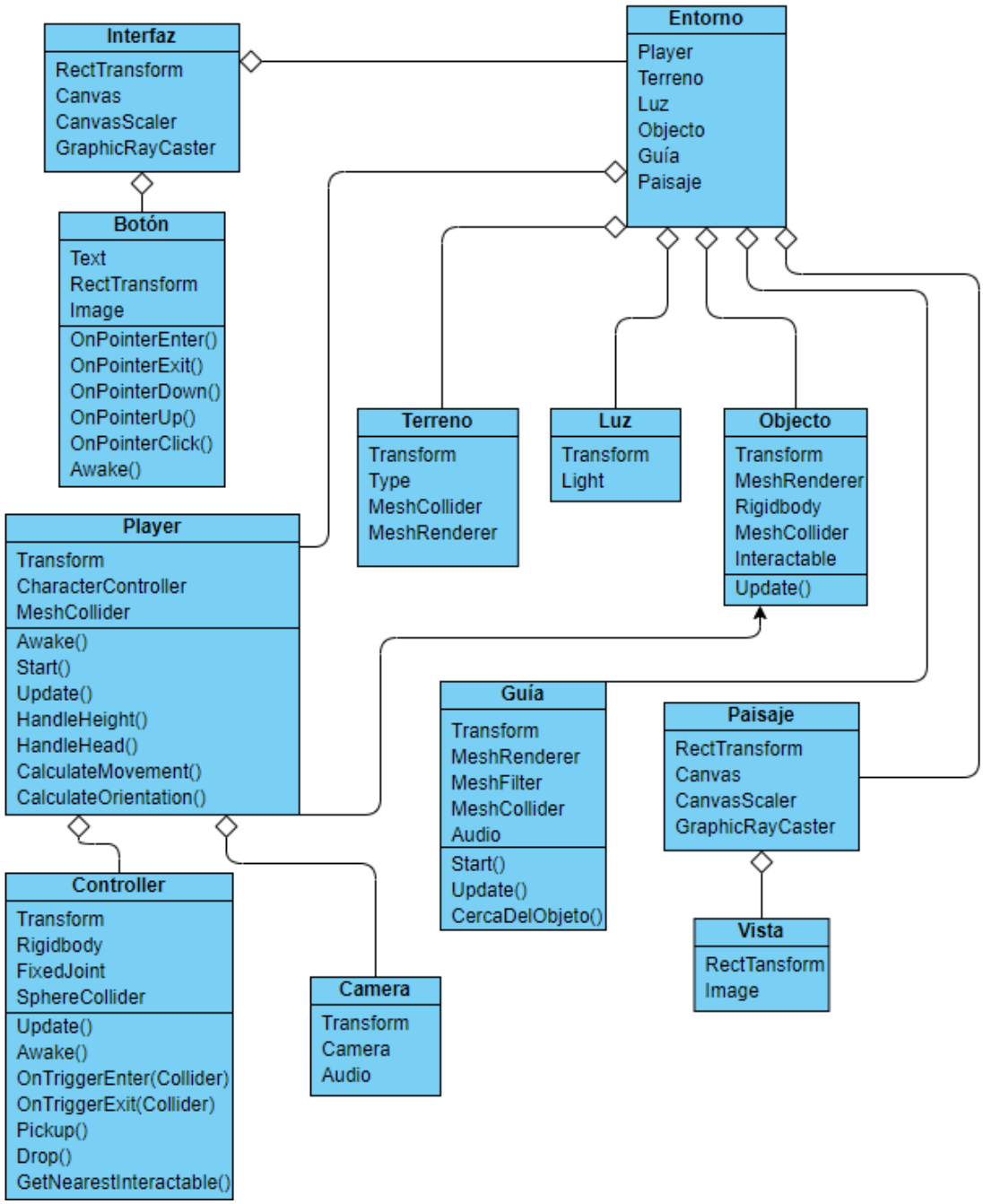
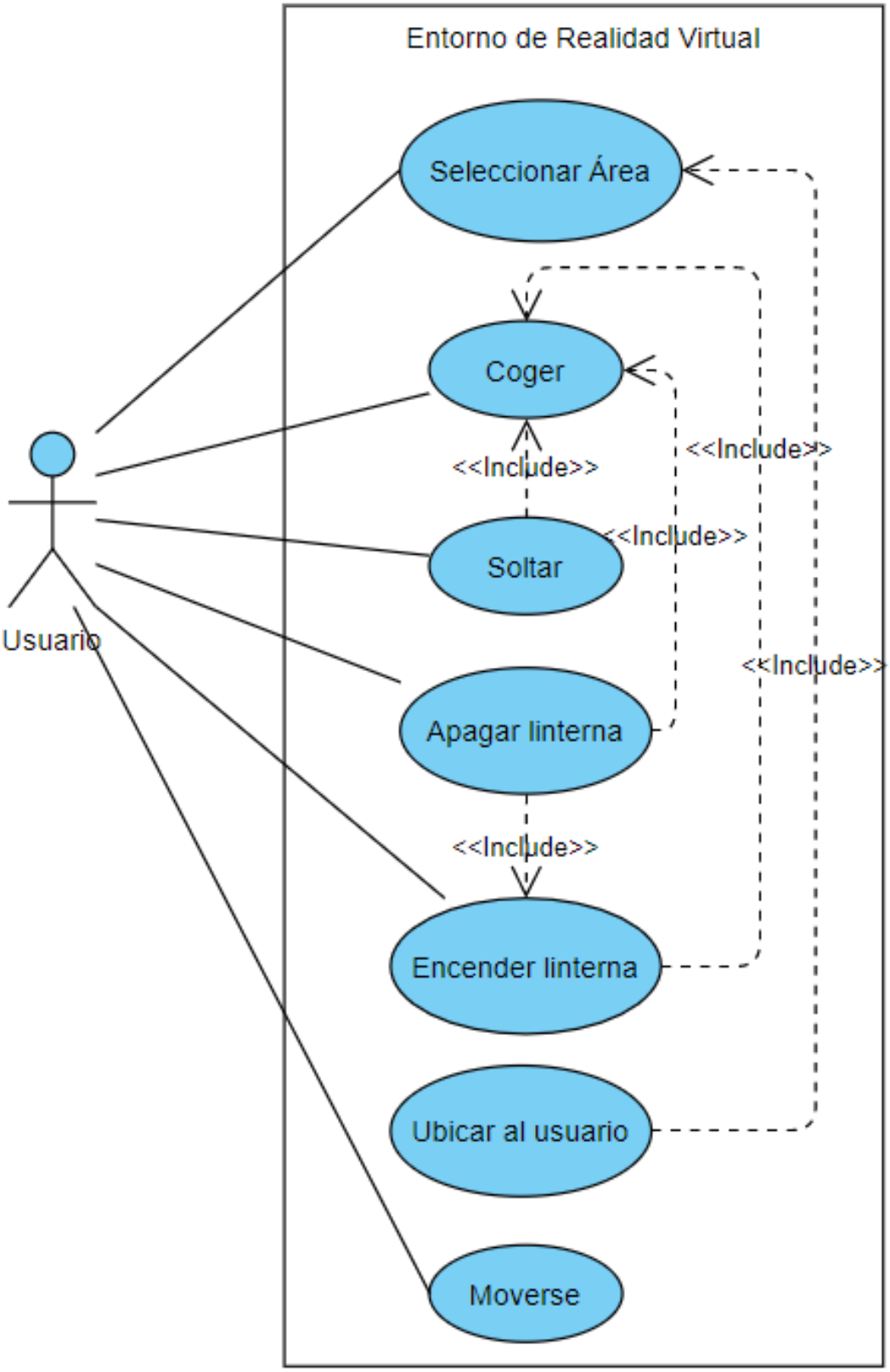


Diagrama de casos de uso

Solo hay un usuario que interactúa con el entorno virtual, las demás acciones se ejecutaran en base a las decisiones que tome el usuario.



4.1.3 Modelado 3D

Para todo proyecto de un ingeniero informático será necesario tener herramientas o información que no estén dentro de sus parámetros de conocimiento, en este caso la realidad virtual trabaja con modelos 3D, el cual simula el entorno que queremos desarrollar.

De la infinidad de aplicaciones para crear espacios y modelos 3D, hemos escogido:

BLENDER: Es un programa informático multiplataforma, dedicado especialmente al modelado, iluminación, renderizado, animación y creación de gráficos tridimensionales. También de composición digital utilizando la técnica procesal de nodos, edición de vídeo, escultura (incluye topología dinámica) y pintura digital.

El programa fue inicialmente distribuido de forma gratuita, pero sin el código fuente, con un manual disponible para la venta, aunque posteriormente pasó a ser software libre. Actualmente es compatible con todas las versiones de Windows, macOS, GNU/Linux (incluyendo Android), Solaris, FreeBSD e IRIX.

Blender está pensado para representar, o dibujar, escenas 3D, generando al final una imagen 2D. Esa representación se realiza mediante motores gráficos, los cuales pueden ser de varios tipos. Blender trae por defecto tres motores gráficos de pre-renderizado y uno de tiempo real.

Los tres de pre-renderizado se pueden dividir en dos realistas (orientados a la creación de imágenes de aspecto foto realista) y uno de representación de dibujo a mano. Los dos realistas son el llamado 'motor interno', que es el original de Blender y aún es el seleccionado por defecto al ejecutar por primera vez la aplicación, y Cycles, que es un motor más reciente y basado en el trazado de rayos de luz. [2]

Dentro de sus simulaciones físicas podemos encontrar la solides rígida, solides deformables, fluidos y humo o vapores.

Otras aplicaciones de modelado 3D:

- Autodesk Maya
- Muvizu
- 3D Builder
- 3DModel Maker
- DAZ 3D

La selección de Blender se debe a la gran cantidad de recomendaciones hacia una buena aplicación de modelado, por su formato gratuito y a nivel ultra-profesional, aunque requiera de una formación exhaustiva, usarlo en este proyecto era una buena oportunidad para iniciar en el uso de esta aplicación.

4.1.4 Implementación en Unity

VIPE-PRO posee librerías y plugins compatibles con Unity, la cual además cuenta con una Asset Store que posee innumerables herramientas para el desarrollo de un proyecto.

UNITY: Es un motor de desarrollo o motor de videojuego multiplataforma creado por Unity Technologies. Unity está disponible como plataforma de desarrollo para Microsoft Windows, Mac OS, Linux. La plataforma de desarrollo tiene soporte de compilación con diferentes tipos de plataformas (Véase la sección Plataformas objetivo). A partir de su versión 5.4.0 ya no soporta el desarrollo de contenido para navegador a través de su plugin web, en su lugar se utiliza WebGL. Unity tiene dos versiones: Unity Professional (pro) y Unity Personal. [3]

Funcionalidades típicas que tiene un motor de videojuegos, son las siguientes:

- Motor gráfico para renderizar gráficos 2D y 3D
- Motor físico que simule las leyes de la física
- Animaciones
- Sonidos
- Inteligencia Artificial
- Programación o scripting
- Etc.

Alternativas a Unity

- Unreal Engine. Por Epic Games.
- Buildbox. Por AppOnboard.
- Visual Studio. Por Microsoft.
- Construct 3. Por Scirra.
- GameMaker: Studio. Por YoYo Games.
- Cooladata. Por Cooladata. ...
- GameSparks. Por Wrike

La selección de Unity se debe a la alta popularidad que tiene y la gran comunidad de usuarios que trabajan con ella, esto permite acceder a multitud de documentación, foros y comunidades donde puedan resolver dudas durante el aprendizaje del uso de la plataforma.

La gran variedad de metodologías y técnicas a la hora de desarrollar permiten que el aprendizaje sea más ameno e interesante, sumado a ello la compatibilidad con otros entornos de desarrollo.

4.2 Clasificación de las actividades

El rango de uso del software es muy amplio, partiendo de edades mayores a los 10 años de edad, por recomendación.

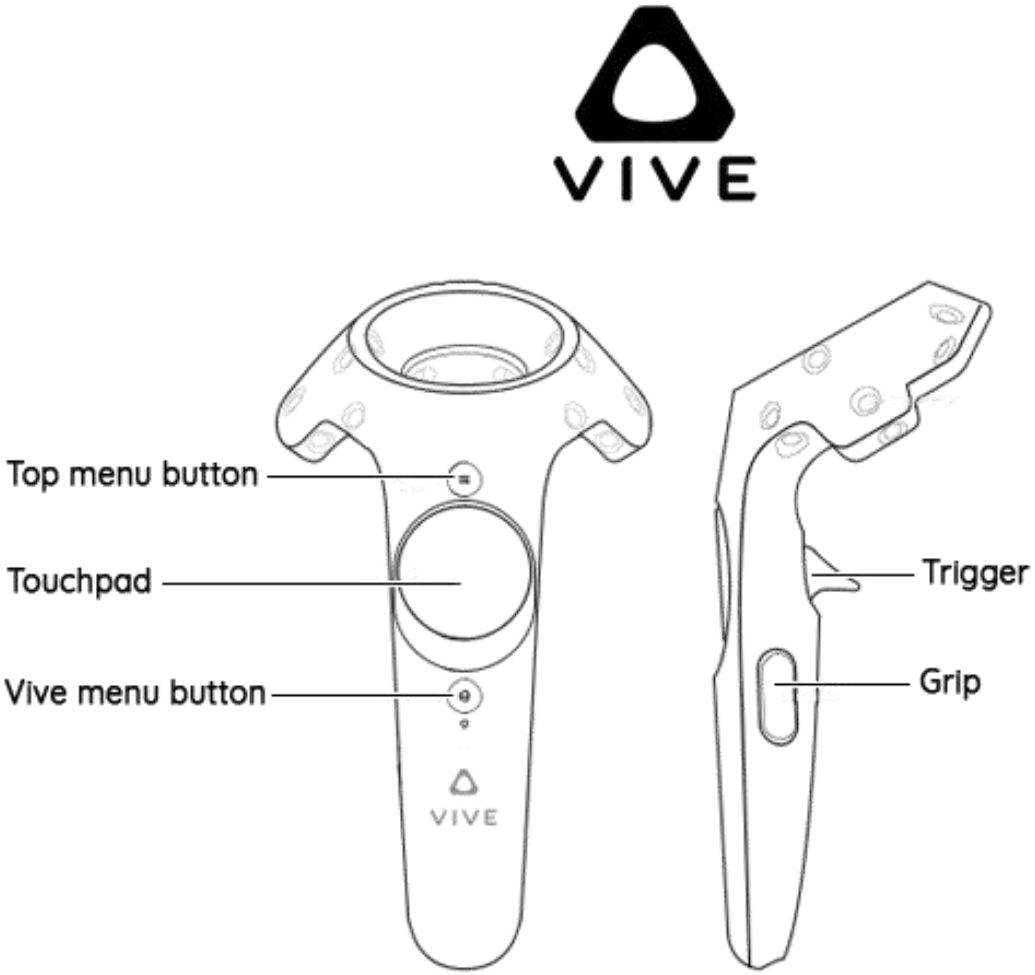
Es aplicable a personas con intereses educativos, personas deseosas de nuevas experiencias o personas que por alguna dificultad no puedan visitar la zona.

4.3 Instrucciones claras

El usuario se encontrará en un espacio adaptado para las gafas de realidad virtual, en este caso la tecnología en la que desarrollaremos será VIVE-PRO.

Es necesario proporcionar al usuario instrucciones de uso, para las distintas herramientas del dispositivo en el que van a utilizar el software.

En este caso usaremos los mandos HTC VIVE PRO, los cuales al igual que las gafas, poseen unos sensores que junto a las estaciones base, localizaran al usuario dentro del entorno y las interacciones que realice.



Mando izquierdo:

- TouchPad: Controla el movimiento del usuario, hacia delante, atrás, izquierda y derecha.

Mando derecho:

- Grip: Controla la sujeción de los objetos del entorno, por ejemplo, una linterna, tablas de madera, etc…
- TouchPad (click): Permite realizar acciones con determinados objetos del entorno.
- Trigger: selección de un elemento en una lista.

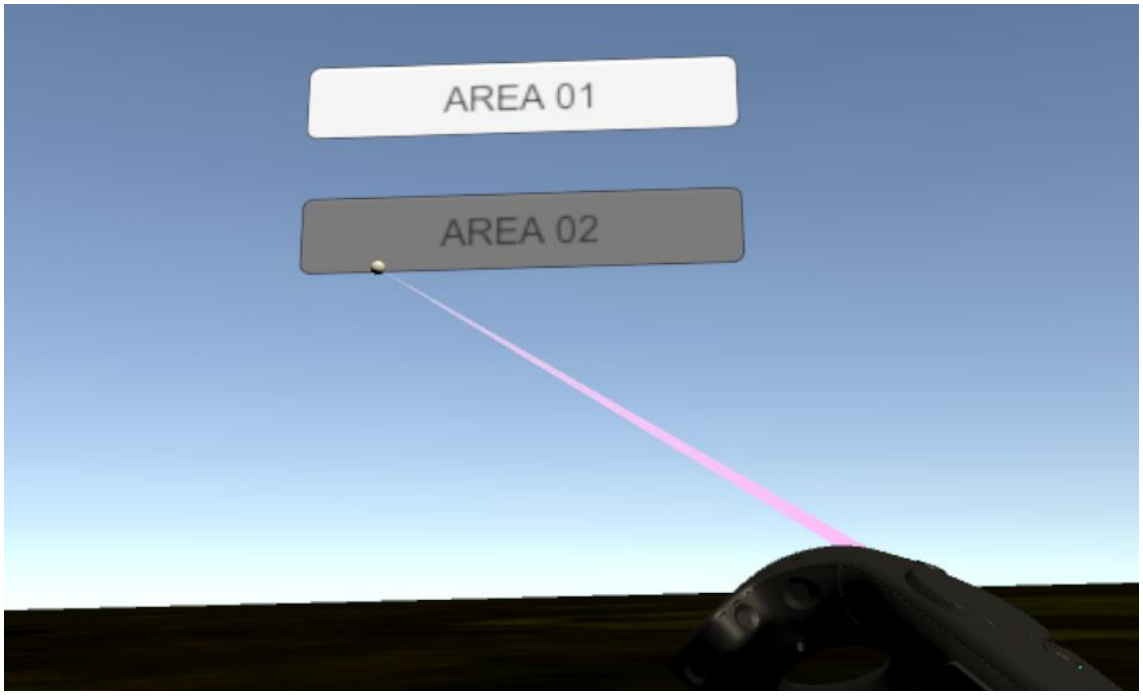
4.4 Interfaz

Menú de selección: el usuario partirá de un espacio cerrado donde encontrará una pequeña pantalla, la cual le establecerá los lugares que puede visitar. La pantalla mostrará un listado de las áreas de los búnkeres que puede visitar, tras la selección el usuario será ubicado en la entrada del bunker correspondiente.

El desplazamiento el usuario dentro del entorno será en una dirección y guiado por las narraciones dentro de la simulación, podrá gestionar por decisión propia a dónde quiere ir, pero siempre dentro del bunker, con una entrada y una salida, lo que facilita el proporcionarle la siguiente área del bunker que desee visitar al acabar el recorrido.

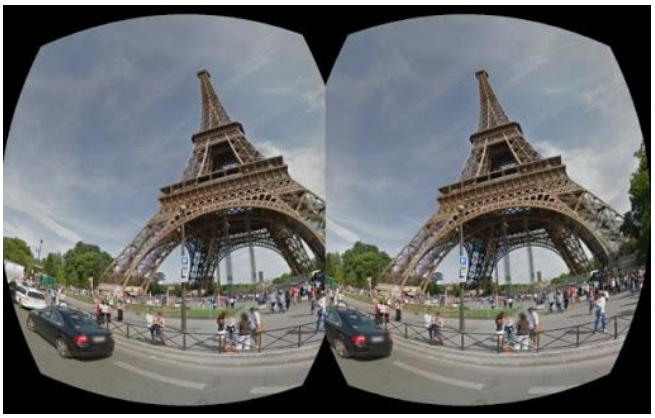
Al finalizar el recorrido, el menú de selección será el mismo que el del inicio, con la particularidad de permitir realizar nuevamente ese recorrido.

Herramienta de soporte para el entorno virtual: Unity



4.5 Diseño

Los proyectos que están basados en realidad virtual pueden ser muy creativos, permitiendo a los usuarios vivir experiencia que escapen de la lógica para su entretenimiento o estudio del comportamiento, o pueden estar basados en su totalidad a la realidad tal y como la conocemos, para vivir experiencias o para la enseñanza.



Google Cardboard

Algunos proyectos que estudian estos aspectos:

Investigando el comportamiento de los peatones con VR y AR a través de edificios virtuales

Aunque la situación en la que se encontraron los participantes no era real, algunos de los ellos mostraron signos de estrés. Según Feng, "algunas personas entraron en pánico y comenzaron a maldecir. Una de las participantes se asustó un poco porque había estado en un terremoto real en su país. Sentir pánico no es muy agradable para los participantes, pero muestra lo realista e inversiva que puede ser el entorno virtual. [4]

Half-Life: Alyx

Es el último título que ha sido lanzado de la mano de Valve, la exitosa saga Half-Life regresa para recordarnos el porqué de su gran éxito con una aventura protagonizada por Alyx Vance. Todos aquellos que lo han probado coinciden en que se trata de una gran experiencia, imprescindible para todos los que quieran descubrir las posibilidades que proporciona la tecnología que nos ocupa a día de hoy. [5]

Con el desarrollo de este proyecto pretendemos generar una ilusión que engañe a nuestra mente partiendo de los sentidos y para ello tenemos que tener en cuenta dos conceptos básicos que son:

Presencia e inmersión.

4.5.1 La presencia

La desarrollaremos en la operatividad del usuario dentro del entorno virtual y como interactúa con él.

Aspectos como el movimiento, son bastante complejos de desarrollar en base a la inmersión que queremos establecer, el usuario al moverse por un túnel puede hacerlo con distintas herramientas:

- A través de los mandos:
  - Utilizando el TouchPad o un Joystick para moverse basándose en las direcciones.
  - Utilizando un gatillo y apuntando a la localización a la que quiere desplazarse.
- A través de herramientas como VIRTUIX OMNI que permiten al usuario caminar sobre una plataforma de desplazamiento, aumentando la inmersión e intensificando la experiencia.

Virtuix Omni podríamos definirlo como un sistema de control complementario a otros elementos como cascos de realidad virtual, que en los últimos tiempos están evolucionando de forma importante en coste y experiencia de uso.

Omni encaja en el puzle ofreciendo un sistema para movernos en el espacio con nuestro propio esfuerzo físico, caminando, e incluso corriendo. La idea es la de poder movernos en todas las direcciones libremente sobre una superficie, y que el sistema reconozca nuestro movimiento y los interprete en el juego. [6]

En este proyecto, dado que no contamos con una plataforma de desplazamiento, lo desarrollaremos en base a un TouchPad, dado que el lugar es muy estrecho y con cruces cerrados, por lo que no tiene sentido establecer la funcionalidad de transporte a un punto.

4.5.2 La inmersión

Con las tecnologías desarrolladas en la actualidad, podemos crear un entorno virtual que interactúe con estos tres sentidos: vista, oído y tacto.

En este proyecto desarrollaremos un entorno virtual que juegue con el sentido de la vista y del oído, dado que para el sentido del tacto necesitaríamos hardware más avanzados, los **cuales** trasmiten el sentido del tacto a través de vibraciones en la palma de la mano, utilizando motores que intensifican gradualmente la vibración dependiendo del uso que se esté dando en el entorno virtual y así estimulando la zona de la mano que hace que el sentido del tacto sea real para nuestro cerebro.

Snippet: El sistema engaña al cerebro para hacerle creer que en realidad está agarrando lo que el casco de realidad virtual le está enseñando.

El guante envía una señal eléctrica que hace que el material se contraiga cada vez que alguien lo toca en un espacio de realidad virtual.

En consecuencia, el dispositivo permite a las personas no sólo sentir los objetos virtuales, sino también que pueden asirlos y sostenerlos, e incluso sentir su forma mientras lo mantienen en la mano.

Se trata de un guante ligero y flexible que imita en el mundo virtual las sensaciones de manipular objetos reales. [7]

La implementación partiendo de la vista:

- Mapeado
- Texturas
- Coherencia
- Colisión
- Leyes

Herramientas utilizadas para generar el modelo 3D: **Blender**

La implementación partiendo del oído.

- Ambiente
- Coherencia
- Diálogos o guías

Herramientas usadas que aumentan la inmersión: **Google Resonance Audio SDK**

5 Implementación

Lo que buscamos en este proyecto es la máxima inmersión posible, utilizando todas las herramientas de las que dispongamos.

5.1 Trabajo con Blender

5.1.1 Mapeado

Para establecer un correcto mapeado, es necesario utilizar datos que se acerquen lo más posible a la realidad, para ello podemos hacer uso de mapas de las estructuras que queremos modelar o tomando las medidas en la misma localidad.

También podemos hacer uso de empresas externas que nos proporcionen los modelos 3D que deseamos utilizar.

Con la ayuda de la empresa Gabinete TRAMA S.L. hemos obtenido los mapas e imágenes de la estructura de los búnkeres de Erratzu, los cuales se han modelado usando la herramienta Blender, partiendo únicamente de los mapas.

Las medidas que se han tomado han sido a escala del mapa, debido a la situación actual no se ha podido visitar el lugar hasta últimas fechas del proyecto, por lo que el primer prototipo no estaba correctamente basado en la realidad, pero una vez se ha podido visitar la localización, se ha pasado a realizar las correcciones necesarias.

Aun así, el desarrollo del modelo 3D ha sido tedioso, debido a los escasos conocimientos de diseño gráfico, realizar el modelo partiendo de cero a consumido mucho tiempo del proyecto, obviamente las correcciones implicarían un uso extra del tiempo restante, por lo que se ha pasado a realizar dos prototipos distintos, la versión antigua con esos pequeños fallos y otra versión de una zona específica del bunker con los nuevos datos corregidos.

1. Prototipo

Medidas basadas en la puerta

Medida real 1.9 metros de alto y 1 metro de ancho
Arqueada en la parte superior con un radio de 0.5 metros

Grosor de los muros: 0.4 metros

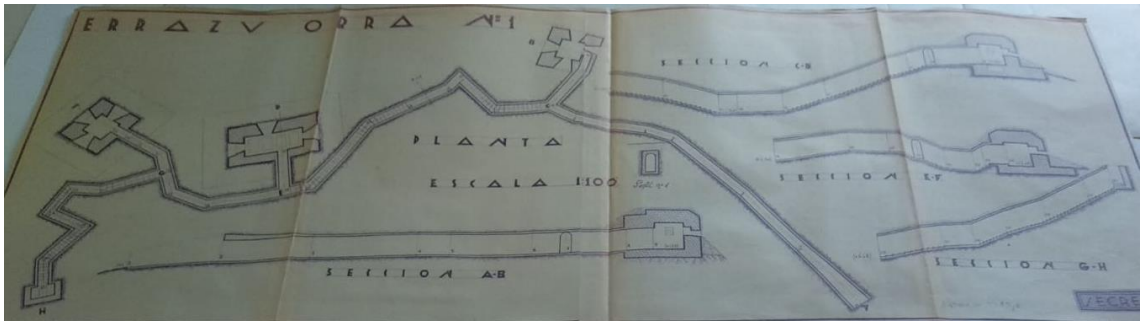
Dimensión de los escalones: 0.3 x 0.18 x 1 metro

2. Prototipo con correcciones

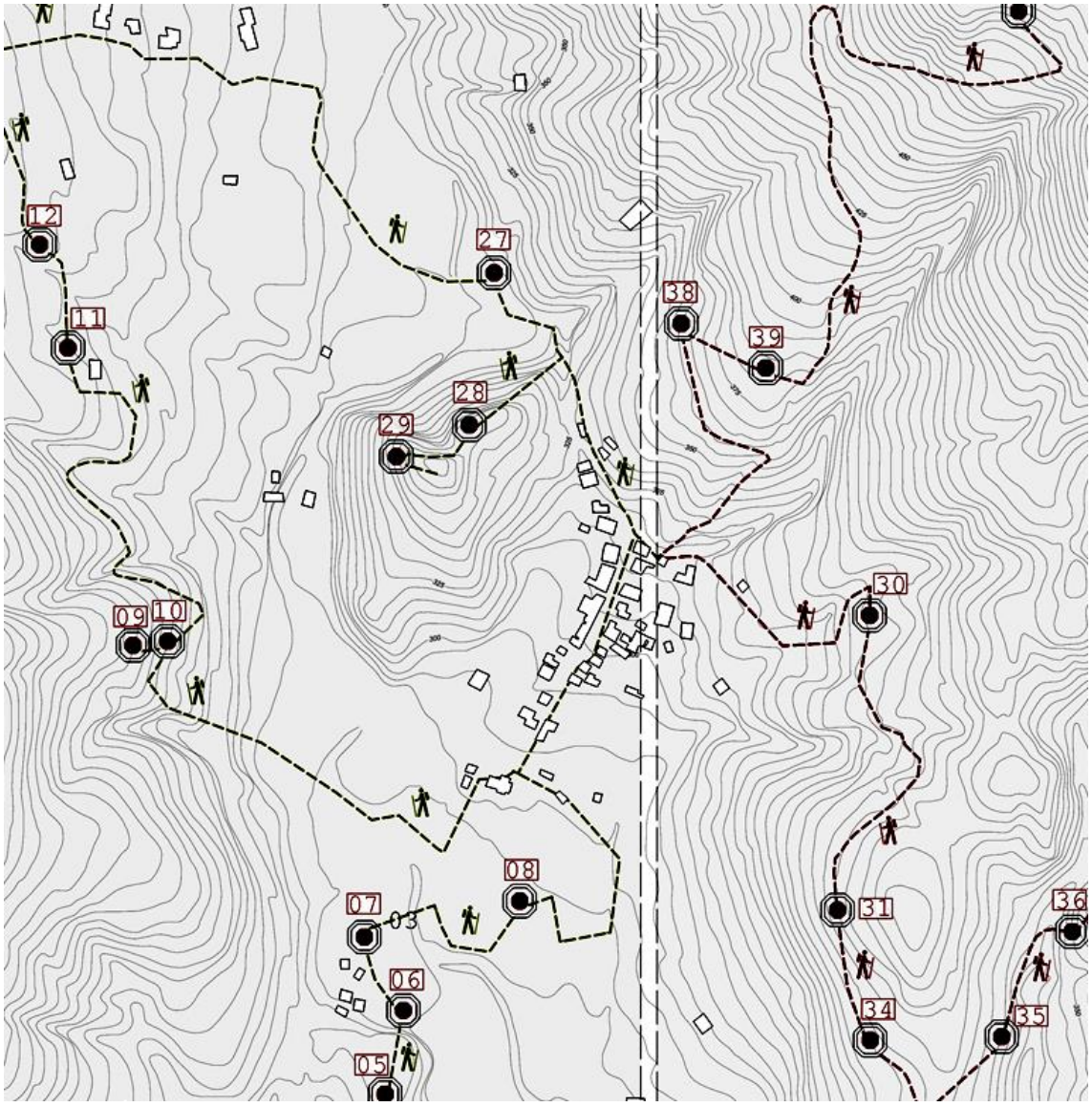
Los escalones tienen distintas medidas, varían en base del lugar, siendo de 0.5 x 0.18 x 1 metros en algunas zonas.

Existen aperturas de 0.25 x 0.15 metros en los techos que actúan como ventilación, los cuales están no siguen un patrón y no se contemplaban en los mapas.

Los arcos en los cruces de los túneles tienen un comportamiento distinto al del mapa, formando un arco en pico.

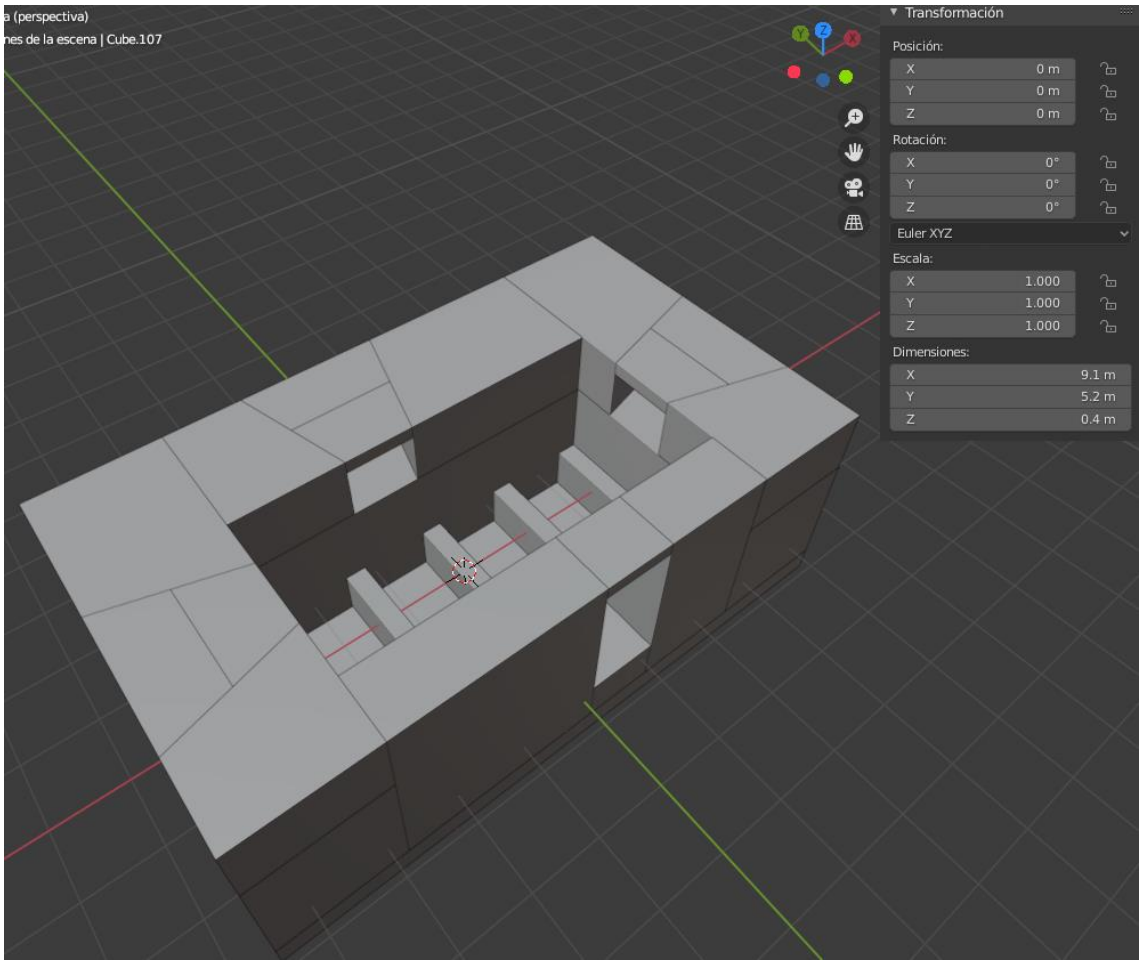


Mapa de la Obra nº1 de los búnkeres Erratzu



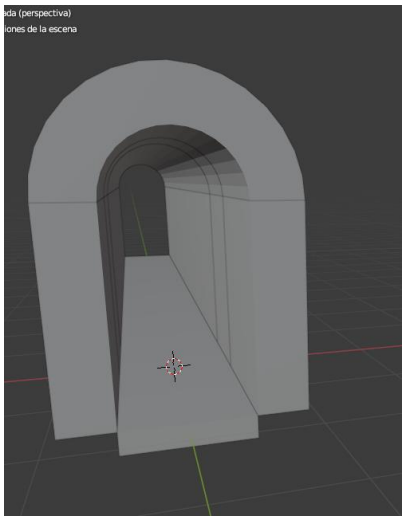
Rutas por el valle de Baztan

El primer prototipo del mapa se basa en una de las casetas dentro del bunker, en la obra nº 1, para iniciar el aprendizaje, trabajamos a partir de cubos los cuales iremos modificando en base a las medidas establecidas en el mapa.



Un gran inconveniente se generó al tener solo una fotografía del mapa, de esta manera no teníamos las escalas reales del lugar, por ello en la siguiente versión adaptamos las medidas.

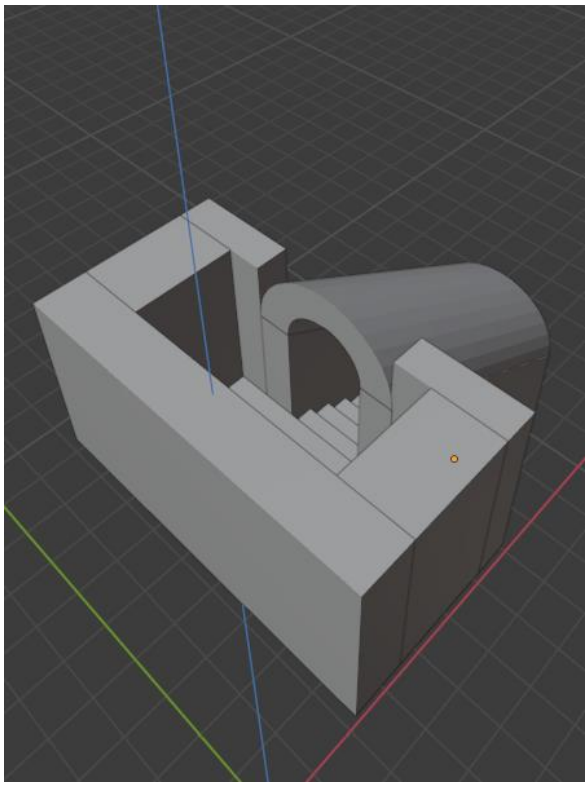
Blender trabaja con escalas y posiciones globales y locales, ello genero muchas dudas al principio del proyecto, las cuales derivaron en un mayor tiempo de modelado.



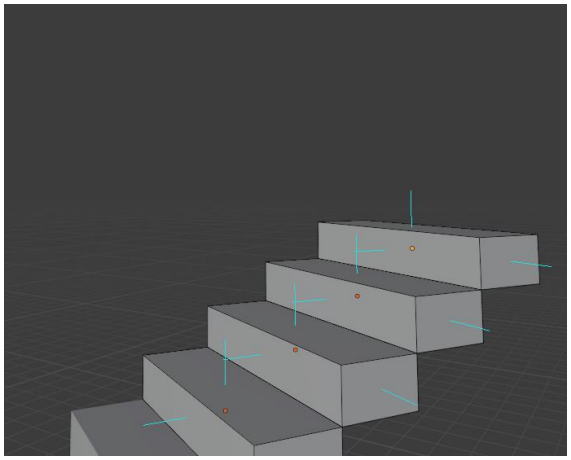
Al generar los arcos, surgieron distintos modos de realizarlos, a través de un cilindro hueco o a partir de la extensión de medio circulo.

Había que valorar también las medidas del radio del arco y al unirse con las paredes establezcan la altura correcta del túnel.

Otro caso importante que había que valorar era la distribución de las escaleras. El desarrollo del modelado del bunker partía por secciones, de esta manera se iban posicionando los elementos uno a uno, a medida se iba dibujando el camino, la desventaja de esto suponía que no debía haber errores tras la incorporación de una nueva pieza.

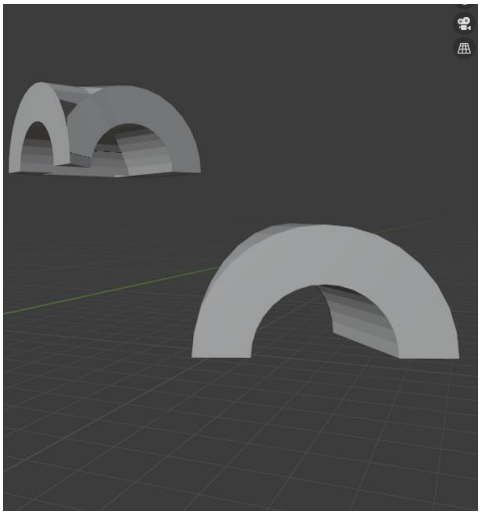


Las ventajas que tenía esta manera de trabajar, afectaban de manera posteriori, es decir, cada elemento como tal tiene una malla, que rodea toda la superficie, entonces al realizar una pieza hueca, el mallado no tenía en cuenta el hueco y establecía el mallado en la superficie más exterior de la pieza, de esta manera un cilindro no se detectaría como hueco si lo tuviese.



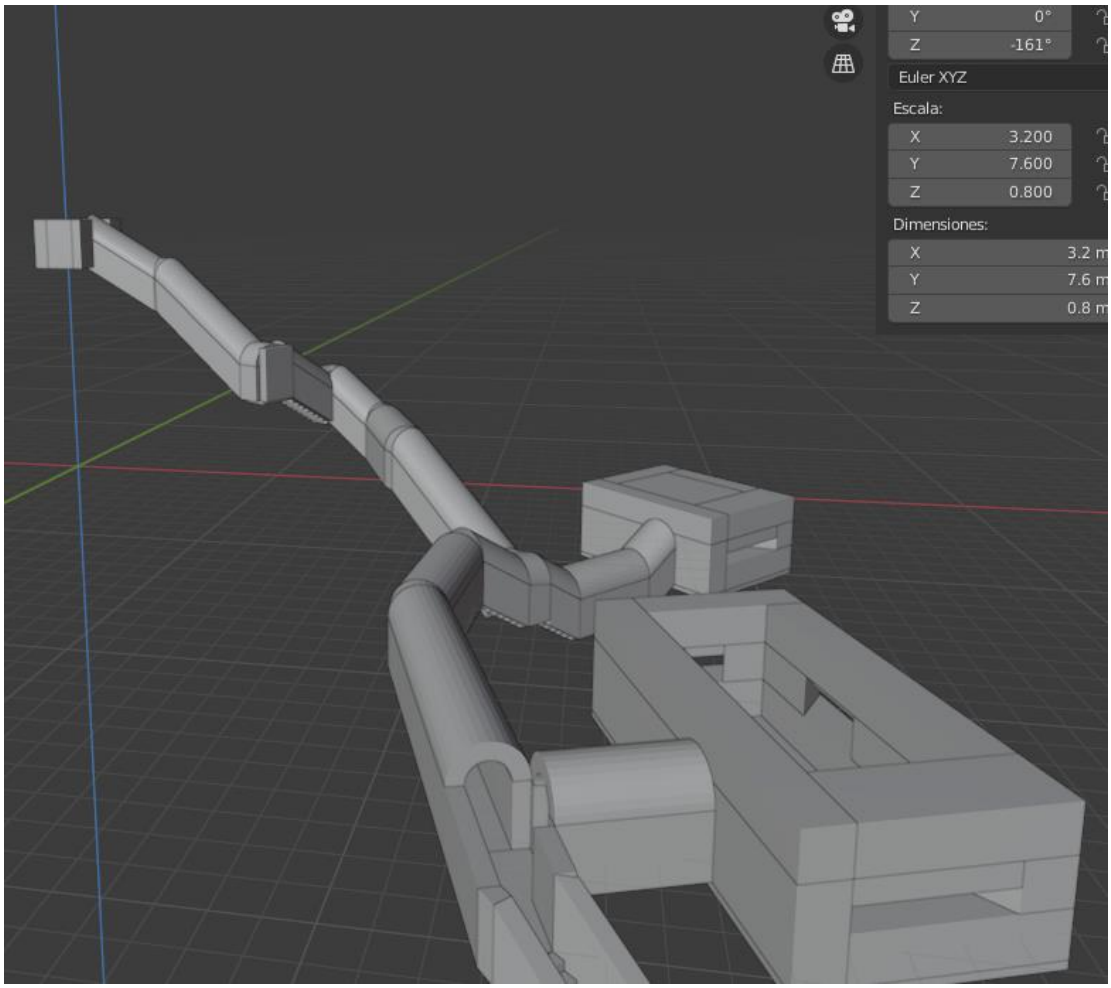
La falta de conocimientos en el uso de esta aplicación genero un retraso en el desarrollo en Unity, debido a que varias piezas al ser exportadas al formato .fbx desaparecían algunas caras, el rehacerlas a veces solucionaba el problema, sin embargo con más investigación se descubrió que cada cara de una pieza posee una dirección normal, la cual para poder verse debe estar orientada hacia afuera.

Por otro lado, en base a la arquitectura de la zona, existían piezas mucho más complejas de realizar, cuyas características suponían varias horas de trabajo extra, las cuales no suponían la labor de un ingeniero informático, sino la de un diseñador gráfico o una empresa externa.

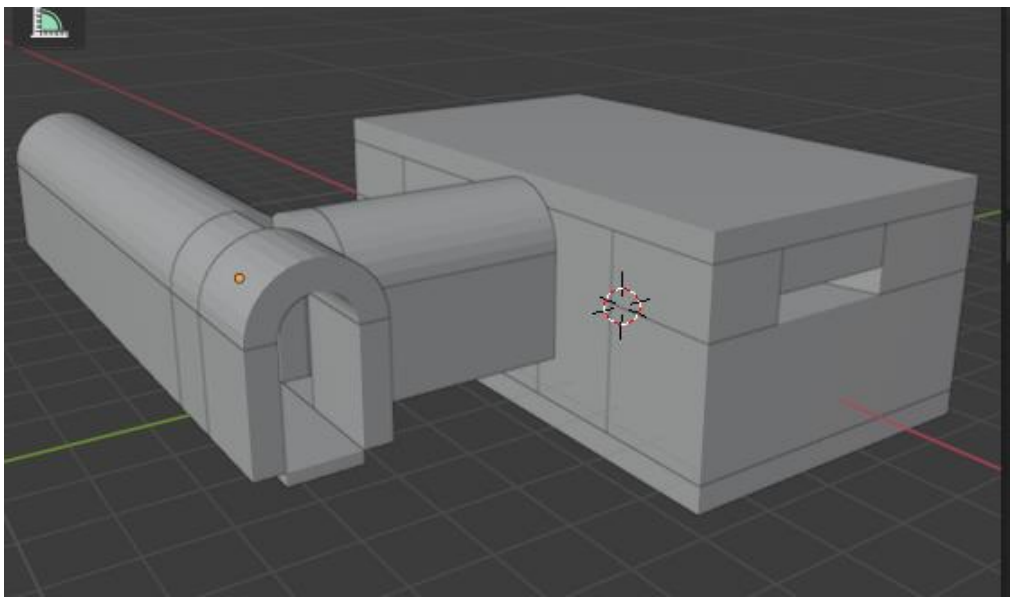


Aun así, el trabajo con Blender ha supuesto un gran aprendizaje y practica la hora de realizar un modelo 3D, para que en futuros proyecto tenga un amplio abanico de herramientas y opciones a la hora de desarrollar un entorno virtual.

Acabado del primer prototipo del bunker, con un 40% de semejanza al entorno real.



Segundo prototipo, con un 90% de semejanza al entorno real, reducido solo a un pasillo y la segunda caseta visitable de la obra nº 1 del bunker.



5.1.2 Texturas

Las texturas son importantes a la hora de aumentar la inmersión del usuario en nuestro entorno, si pretendemos educar al usuario con el entorno que estamos desarrollando, debemos mostrarle un aspecto acorde a ello.

Si se muestra una cueva, las texturas deben presentar humedad, material por el que están compuesto las paredes, vegetación, antigüedad, etc. Todo ello permitirá al usuario una mayor presencia de su persona en el entorno virtual.

Este proyecto se basa en búnkeres construidos durante la postguerra civil, entre los años de 1939 y 1960, hay que tener mucho en cuenta el material que se usó para construirlos, las texturas también cambiaran en base al bunker en el que nos encontramos, dado que en ciertas obras una vez realizado el encofrado se pasó a realizar el pulido, sin embargo, en otras no, ello ayuda al usuario a entender que las prisas y el estado en alerta de aquella época no permitieron acabar ciertos trabajos.

De esta manera en dos búnkeres distintos, para dos mismos tipos de muros, se usarán distintas texturas unas que marquen la separación de las tablas que se utilizaron para el encofrado y otras que muestres una pared lisa de concreto.

Otro aspecto que hay que tener en cuenta son las condiciones actuales del bunker; La lluvia y la vegetación cambian el estado del bunker, estableciendo un suelo que además de ser de concreto también muestre húmeda y hojas encharcadas.

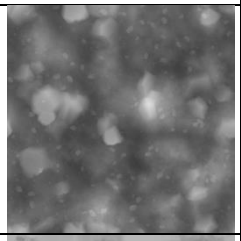
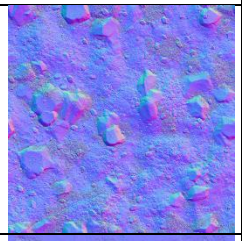
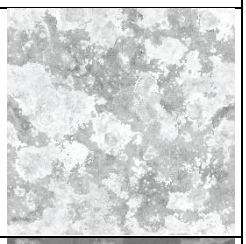
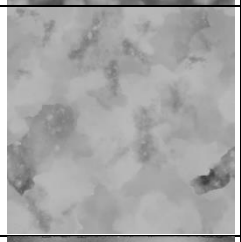
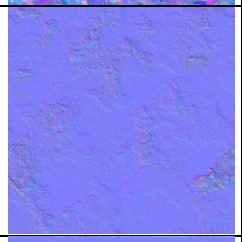
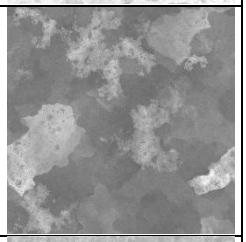
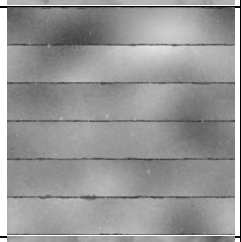
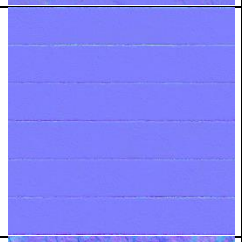
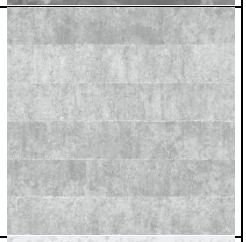
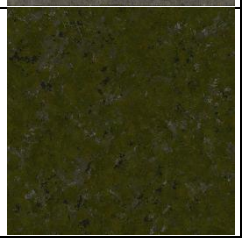
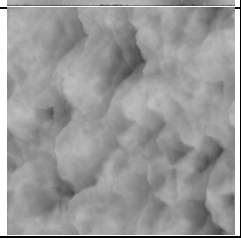
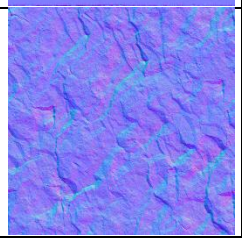
Algunos detalles, a nivel cultural, los soldados durante la construcción hicieron escritos o dibujos que representaban a su unidad en las paredes y techos del bunker, a través de fotografías, se pueden implementar sobre las zonas correspondientes, de tal forma que añade toques más realistas al entorno.

Para el desarrollo aplicaremos distintos mapas de texturas:

- Albedo: Similar a un mapa difuso, pero en el cual, eliminamos las sombras y las luces, este puede ser más realista dado que a veces las sombras y luces pueden ser demasiado pesadas o incluso innecesarias si se utiliza una textura de desplazamiento en malla.
- Normal: Es perfecto para el software 3D, porque cada color representa un eje de direcciones diferente, el ordenador puede comprender la forma además de las protuberancias pequeñas y medianas.

- Height: Mapa de relieve o desplazamiento, se utiliza para deformar la malla real, para suelos, rocas, acantilados, donde la forma del material es importante ya que requiere mucho más detalle de malla.
- Specularity: Solo se usa para flujos de trabajo especulares que no son PBR.
- Roughness: Este mapa controla la nitidez de los reflejos y debe conectarse a una entrada de brillo o rugosidad.
- AO: Mapa de oclusión ambiental: son mascaras que ocluyen la luz donde sea menos visible. Cuanto más oscuro sea el lugar del mapa, menos luz se verá.
- Diffuse: Prácticamente es una foto estándar del material real.

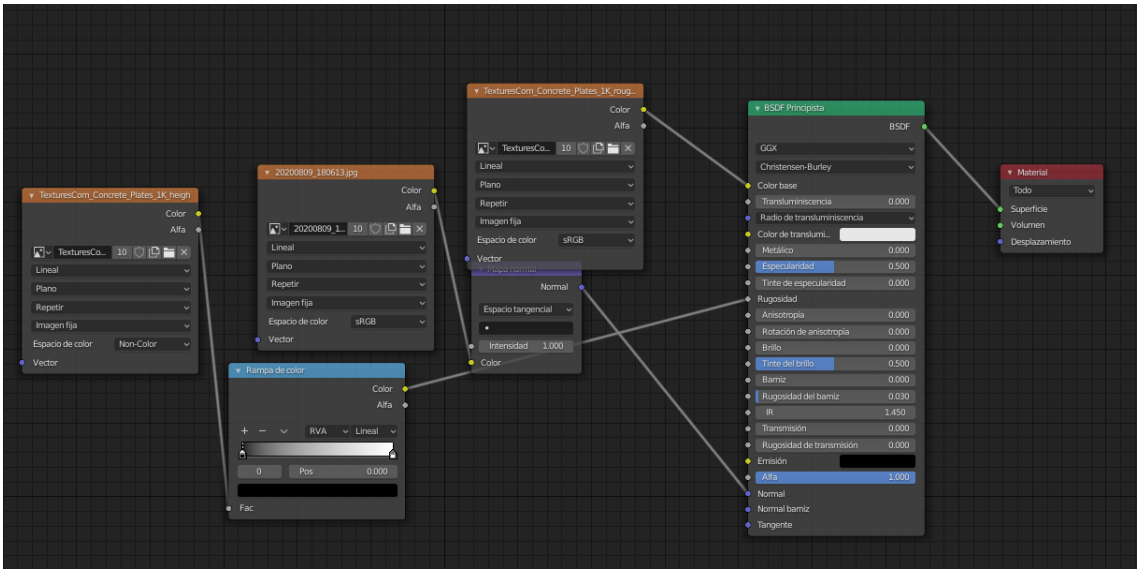
Nosotros usaremos los siguientes mapas de texturas y materiales obtenidos de www.textures.com

	Albedo	Height	Normal	Roughness
Rock Soil				
Concrete Wood Grain				
Concrete Plates				
Rock Mossy				

Y como material fotográfico, hemos ido a la zona de los búnkeres reales a fotografiar las texturas del lugar y luego las aplicaremos en nuestro modelo realizando distintas combinaciones para que se adapte lo más posible a la realidad.

Si queremos aumentar la inmersión del usuario en el entorno, habría que cambiar el modelo en base a las texturas, por ejemplo, en lugar de hacer una pared lisa y añadir ranuras a través de texturas, también se podrían añadir las ranuras en el modelo y tras aplicarle las texturas y hacer las modificaciones correspondientes, tendríamos una pared cuyas ranuras encajasen con las texturas y añadiese profundidad a los detalles.

Podemos utilizar el editor UV para cambiar las texturas que queremos realizar.



5.1.3 Coherencia

Atado a esto último, la coherencia es otro aspecto que debemos desarrollar en nuestro entorno virtual basado en la realidad, es un aspecto que hay que tener mucho en cuenta si queremos hacer un entorno educativo y realista.

En el entorno podemos añadir objetos que permita al usuario desenvolverse mejor e interactuar con ellos, en una pradera o bosque con la flora, en un edificio con los utensilios correspondientes y a nivel más avanzado, en el Renacimiento con obras de la época y no posteriores o fuera de la época.

Si el usuario posee niveles de conocimiento avanzado puede perder esa inmersión al ver objetos que no corresponden con el entorno en el que está establecido.

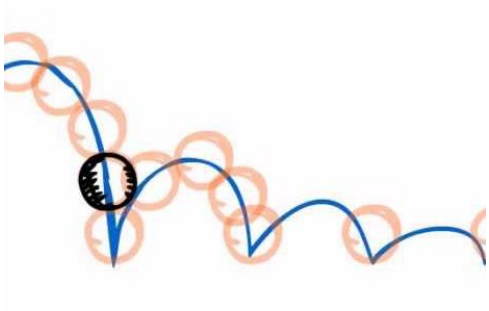
En esta zona por ejemplo se encontrarían arboles tales como Fresno de hoja ancha, Roble común, avellano, castaño y aliso; El hecho de no tener estos modelos no implicaría meter modelos que tengamos, como por ejemplo una palmera, porque reducirían la inmersión de un usuario especializado además de dar información errónea a usuarios en plena educación.

		
Quercus Robur	Castanea Sativa	Corylus Avellana

Al incorporar estos modelos en relieve a la zona del bunker, manteniendo la lógica de posicionamiento, permitirá tener un mayor concepto del entorno.

Además, el añadir movimiento al modelo provocaría la ilusión de viento o aire suave. [8]

5.1.4 Colisión



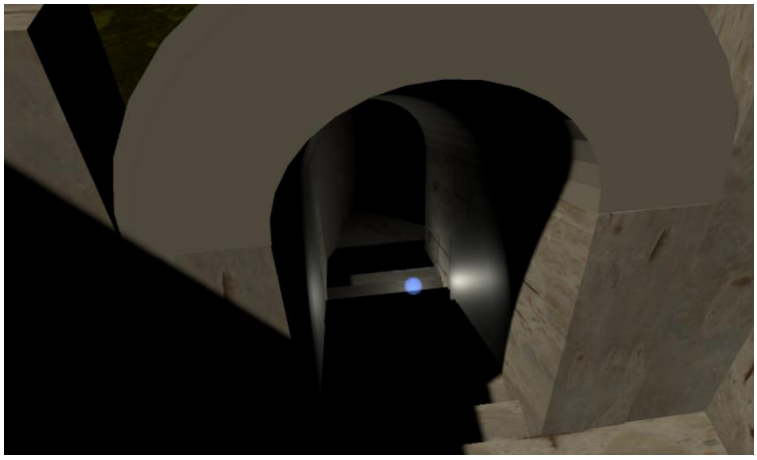
Dado que no tengamos las herramientas táctiles que nos permitan sentir a través de las manos objetos del entorno virtual, no implica el no desarrollar la interacción entre otros elementos de la simulación.

Dos o más objetos solidos deben poder colisionar entre sí, en este proyecto, si yo tengo una linterna en la mano, esta no puede atravesar las paredes, al igual que mi propia persona no debe cruzar las paredes o caer atravesando el suelo, es importante darle una masa y una superficie, en este caso nos ayudamos de Unity para establecer un mallado a los objetos, metiéndolos en unos contenedores que cubren toda la superficie del objeto.

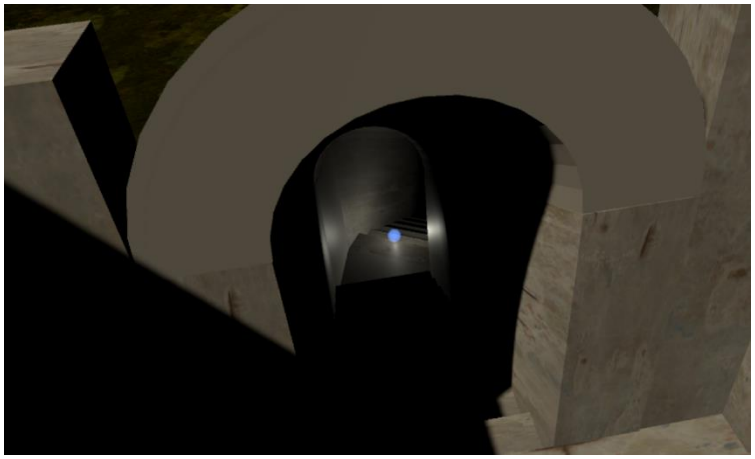
De esta manera dos objetos pueden interactuar entre sí, otras posibilidades añadidas son las de generar una vibración al chocar dos objetos, en cuanto la superficie de uno interactúe con la superficie de otro se genere una vibración en los mandos, de tal manera que sea más fuerte cuanto más cerca estén el uno del otro o cuanto más rápido hayan interactuado, de esta manera aumentamos la percepción de velocidad y fuerza que se aplican en los objetos con los que interactuamos.

5.1.5 Leyes

Partiendo de lo que se comentaba antes, hay que seguir las leyes físicas si se desea establecer un entorno virtual lo más realista posible, Unity permite esto al integrar masa a los objetos, teniendo así un peso de caída.



La linterna de la simulación debe emitir luz al encenderse (al presionar un botón o el gatillo del mando), la intensidad debe ser correspondiente a la de una linterna real y además ser unidireccional, debe poder lanzarse y al caer sobre una superficie, tener un pequeño movimiento aleatorio antes de detenerse.

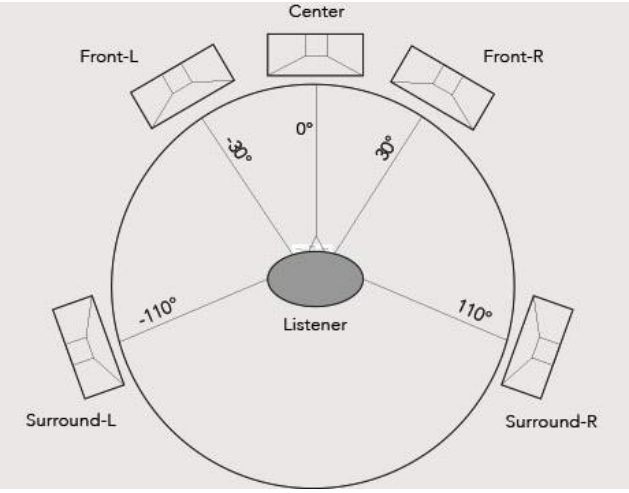


Esfera luminosa cayendo por las escaleras

Si la luz se emite en todas las direcciones, esta debe ser coherente ante el movimiento del objeto que la emite, si es un objeto esférico y al encontrarse en una pendiente, el objeto al rodar cambiara su comportamiento y el del entorno.

5.2 Audio

Otro aspecto que mejora la inmersión en la realidad virtual son los sonidos, que desarrolla el ambiente, la coherencia y los diálogos, unas herramientas que usaremos en este proyecto.



Dado que la salida del sonido es a través de auriculares, esto permitirá aislar al usuario de la realidad y lo deja listo para aplicarle sonidos dentro de la simulación que lo engañen, es algo que debemos aprovechar.

El trabajar con conceptos tales como el sonido en 3D permitirá ubicar al usuario dentro del espacio generado y así se orientará usando como referencia sonidos que se generaran de distintas direcciones, o que se encuentren en movimiento.

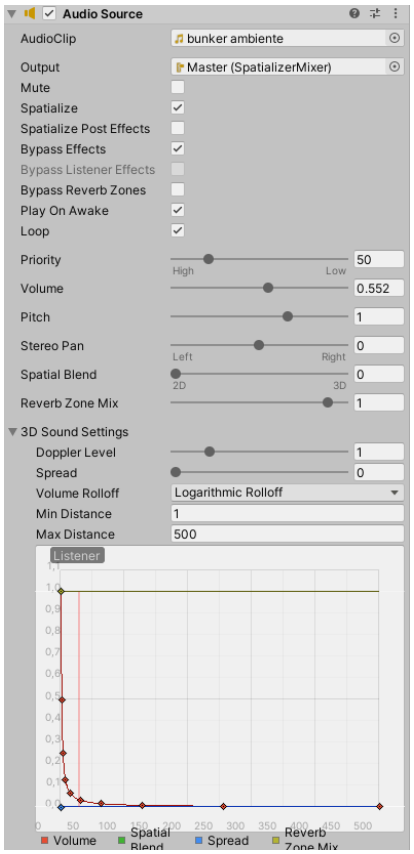
Al pasar unos minutos el usuario se acostumbrará y sentirá que se encuentra en esa zona, dado que el cerebro captará la información que le estamos transmitiendo, rellenando pequeños aspectos e interpretándola como un espacio real.

Google Resonance Audio SDK: Se trata de un kit de desarrollo de software de audio especial (SDK) basada en su tecnología propia que funciona a escala de plataformas móviles y de escritorio.

Ofrece entornos de audio dinámicos y ricos en sus experiencias de VR, AR, juegos o video sin

afectar al rendimiento, utiliza algoritmos de procesamiento de señal digital altamente optimizados basados en Ambisonics de orden superior para estabilizar cientos de fuentes de sonido 3D simultáneos, sin comprometer la calidad del audio. [9]

5.2.1 Ambiente



El usuario recibirá un sonido de fondo que mesclará el viento que entra por el túnel, el crujido de los árboles de fuera, algún animal de la zona, etc.

El nivel del volumen será bajo para no se distraiga en ese sonido y solo sea de fondo, estará alejado para que el sonido se disperse y no venga de una sola dirección.

Hemos buscado un sonido adecuado para el entorno, de librerías de sonido o grabando un audio largo en la zona, sin interferencia de otras personas y reproduciéndolo en bucle.

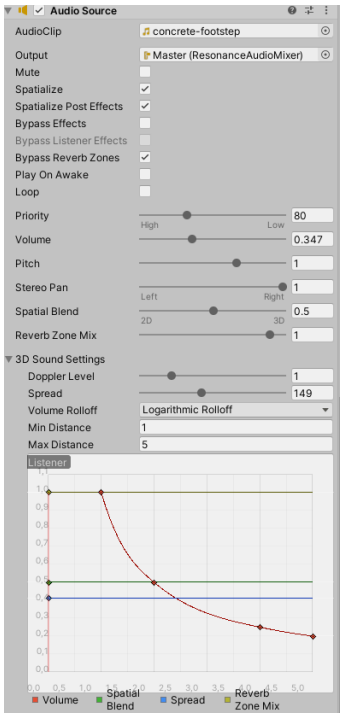
Es importante entender que el sonido se tiene que reproducir en todas las direcciones, dado que pertenece a todo el entorno, no solo a un objeto y el hecho de no darle movimiento no confundirá al usuario si en un momento dado se genera un sonido puntual y luego recibe un desplazamiento.

5.2.2 Coherencia

Los ruidos deben acoplarse al movimiento del usuario, si este camina, debe poder oír sus pasos sobre el suelo encharcado, para ello se creará un script que detecte el movimiento de usuario y genere el ruido cuando ciertas variables de posicionamiento cambien.

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

public class FootStep : MonoBehaviour
{
    private CharacterController m_CharacterController = null;
    public AudioSource au1;
    // Start is called before the first frame update
    void Start()
    {
        m_CharacterController = GetComponent<CharacterController>();
    }

    // Update is called once per frame
    void Update()
    {
        if (m_CharacterController.isGrounded == true && m_CharacterController.velocity.magnitude > 1.1f && au1.isPlaying == false) {
            //audio.volume = Random.Range(0.8f, 1);
            //audio.pitch = Random.Range(0.8f,1.1f);
            au1.Play();
        }
    }
}
```

Los sonidos deben ser acordes a la zona, además la herramienta de sonido 3D de google nos permite ubicar por ejemplo unas ovejas fuera del bunker y saber dónde se encuentras gracias a la dirección del sonido emitido por los auriculares, esta tecnología ya está desarrollada por lo que es solo un paquete que debemos incorporar a nuestro proyecto.



Para realizar este efecto, hemos buscado sonidos de pasos y hemos creado un "Audio Source" el cual se reproduce cuando el usuario realiza un cambio de posición, el audio se reproduce en su misma posición partiendo del sueldo que está pisando, también cambiamos la curvatura del sonido 3D para hacerlo más realista y no sea como un audio más que se esté reproduciendo.

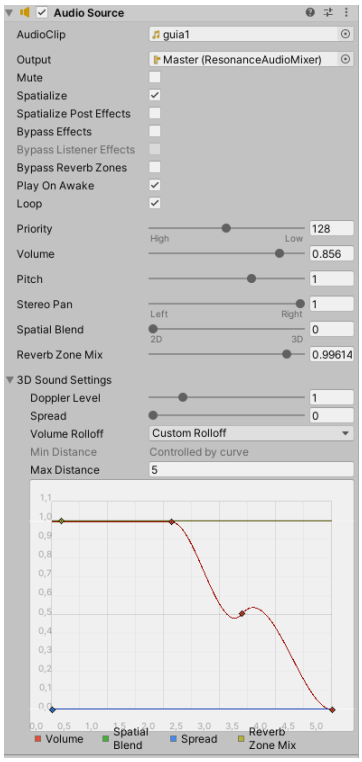
Hay varias herramientas en las que se puede profundizar más, pero la escasez de tiempo no lo permite hasta futuros proyectos.

5.2.3 Diálogos o guías

Este proyecto procura cumplir un nivel educacional histórico, por lo que contamos con audios y testimonios que se reproducirán en distintas zonas del bunker, proporcionando datos interesantes del lugar, esto se debe a que, revisado algunos estudios, la mejor forma de captar información es a través del oído y no usando escritos.

```
public class Guia1 : MonoBehaviour
{
    public AudioSource au1;
    public Transform player;
    public Transform sphere;
    int activa;

    float timeLeft = 30.0f;

    // Start is called before the first frame update
    void Start()
    {
        au1.Stop();
    }

    // Update is called once per frame
    void Update()
    {
        if (CercaDelObejto(player, sphere, 3) && !au1.isPlaying)
        {
            au1.Play();
        }

        if (!CercaDelObejto(player, sphere, 3) && au1.isPlaying)
        {
            au1.Pause();
            activa = 1;
        }

        if (!CercaDelObejto(player, sphere, 3) && !au1.isPlaying && timeLeft < 0)
        {
            au1.Stop();
            activa = 0;
        }

        if (activa == 1)
        {
            timeLeft -= Time.deltaTime;
        }

    }

    private bool CercaDelObejto(Transform a, Transform b, int d)
    {
        bool cx, cy, cz;
        cx = (a.position.x < b.position.x + d && a.position.x > b.position.x - d);
        cy = (a.position.y < b.position.y + d && a.position.y > b.position.y - d);
        cz = (a.position.z < b.position.z + d && a.position.z > b.position.z - d);
        return cx && cy && cz;
    }
```

Nuestro sistema visual está más atento a los detalles del entorno y nos descentrar de cualquier tipo de lectura, por ello resulta más interesante dar a conocer los datos del bunker, tales como su uso, fechas de construcción, anécdotas de lugareños, etc., a través del oído.



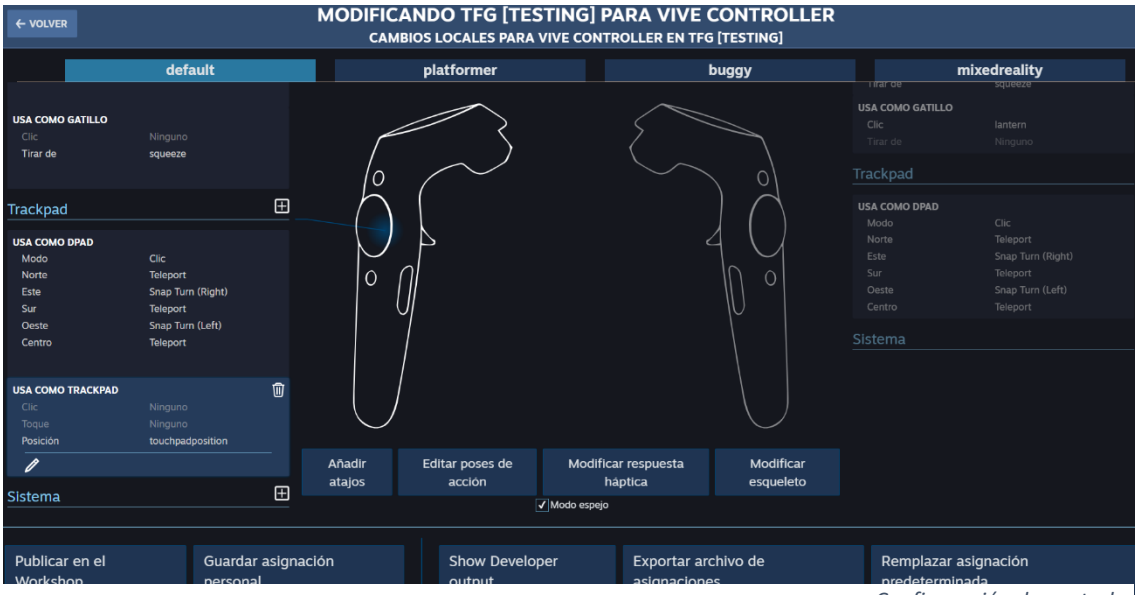
Dado que no tenemos modelos de personas reales, hemos decidido establecer una esfera con una textura futurista, la cual reproduce el audio informativo del área en el que se encuentra, una vez el usuario este cerca de ella de la misma manera cuando este alejado o acceda a otra zona con audio, la anterior se detendrá.

En este caso si hemos utilizado la herramienta de sonido 3D, es decir a medid que nos movemos por la habitación, el guía nos va contando cosas sobre el lugar y si nos movemos alrededor suyo, el sonido se distribuye de manera distinta por los auriculares, dando así un efecto de profundidad y de vida al objeto que representa al guía.

La inmersión se completaría si añadimos el modelo de una persona real al objeto que genera el sonido, de esta misma manera el incorporar movimiento al andar o movimiento de los labios, provocaría un mejor entorno realista.

5.3 Operatividad

El usuario se desenvuelve dentro del entorno virtual y para ello necesitaremos las herramientas que nos proporciona la plataforma de desarrollo que hemos escogido, Unity, además de los mecanismos por defecto de las gafas de realidad virtual, plugins que incorporaremos a nuestro proyecto y scripts que tendremos que desarrollar para permitir que el usuario se mueva de manera fluida y pueda interactuar con objetos en su interior, como por ejemplo una linterna, una tabla, etc.



Configuración de controles

Siempre que vayamos a añadir una nueva opción que no se encuentre en los controles por defecto, tendremos que crearla y establecer su funcionamiento a través de un script.

En este caso:

- TouchPadPosition, establece el movimiento del usuario, dado que no utilizamos los plugins estándar de VIVE-PRO.
- Lantern, usa el gatillo derecho para encender la linterna esta también establecida en su correspondiente script.

5.4 Trabajo con Unity

5.4.1 Iluminación

El entorno virtual contará con una luz que actuará como sol, dado que el recorrido no dura más de 10 - 15 minutos, no será necesario darle movimiento a la iluminación, sin embargo, en otro tipo de proyecto, si se basa en el pasar del tiempo, sería necesario establecer un movimiento periódico a la luz.

Algunas investigaciones establecen esto para mejorar la percepción que se tenga del tiempo, por otro lado, también se quiere comprobar si el largo uso de un entorno virtual con un horario del pasar del día cambiaria nuestra perspectiva del tiempo, tan cómo se da en los polos de la tierra, en el espacio y en la zona centro del planeta.

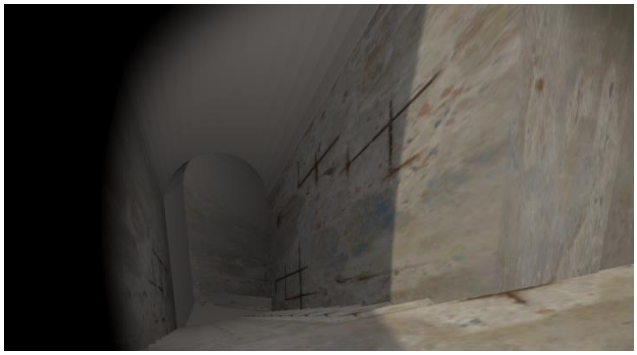
Además, el sujeto encontrará una linterna en el suelo antes de ingresar al bunker que previamente se encuentra a oscuras, la linterna se podrá encender o apagar, esto permitirá una inmersión más realista.

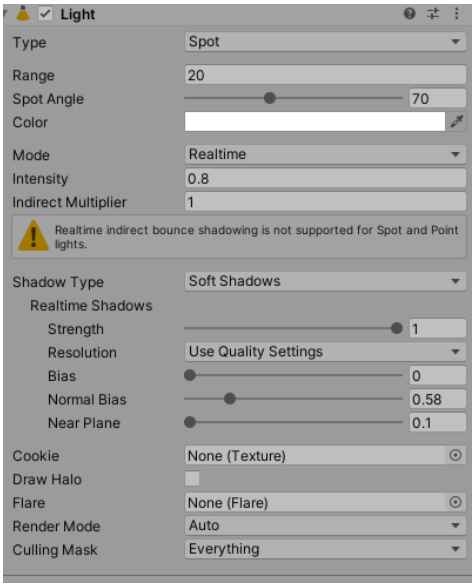
Dependiendo del lugar en el que se encuentre el objeto que generara la iluminación será diferente y el tipo de iluminación también, por ejemplo, si fuese una antorcha, la iluminación tendría movimiento propio, mientras que, si fuera una lámpara, la iluminación se realizaría por los laterales.

El script correspondiente al uso de la linterna es muy básico, tras la interacción con la opción correspondiente del mando, encenderá o apagará la luz, comprobando previamente su estado.

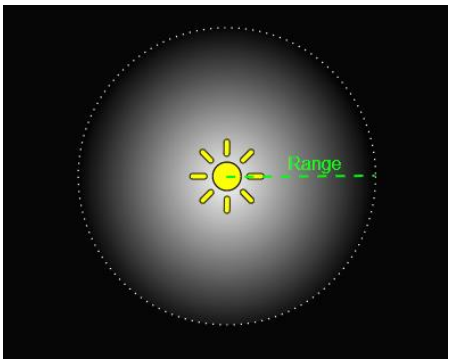
```
public class Lantern : MonoBehaviour
{
    public Light myLantern;

    public SteamVR_Action_Boolean m_Trigger;

    //private Interactable interactable;

    // Start is called before the first frame update
    void Start()
    {
        myLantern.enabled = false;
        //interactable = GetComponent<Interactable>();
    }

    // Update is called once per frame
    void Update()
    {
        //if (interactable.attachedToHand != null)
        //{
        //SteamVR_Input_Sources source = interactable.attachedToHand.handType;
        if (m_Trigger.stateDown)
            OnOff();
        // }

        if (Input.GetKeyDown(KeyCode.F))
            OnOff();
    }

    void OnOff()
    {
        myLantern.enabled = !myLantern.enabled;
    }
}
```

La luz tendrá distintas características, las cuales iremos modificando en base a nuestra necesidad para obtener un resultado final con el que estemos conformes.



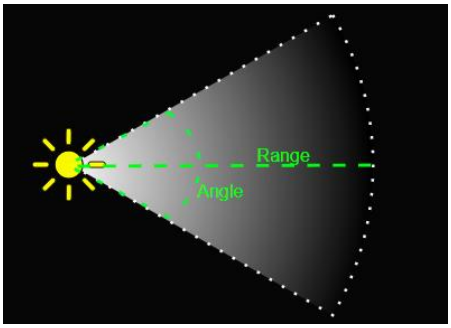


Point Lights



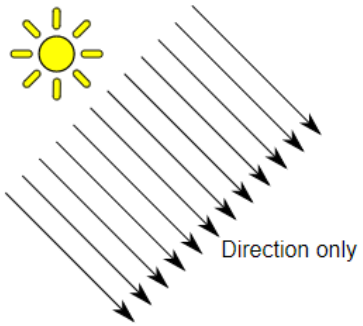
Un Point Light se encuentra en un punto en el espacio y emite luz en todas las direcciones por igual. La dirección de la luz que alcanza una superficie es la línea desde el punto de contacto hasta el centro del punto del Light Object. La intensidad disminuye con la distancia, llegando a cero en un rango especificado.

Spot Lights



Un Point Light tiene una posición específica y un rango sobre el cual la luz incide. Sin embargo, el Spot Light está limitado a un ángulo, resultando en una región de iluminación con forma cónica. El centro del cono apunta en la dirección de avance (Z) del Light Object.

Directional Lights



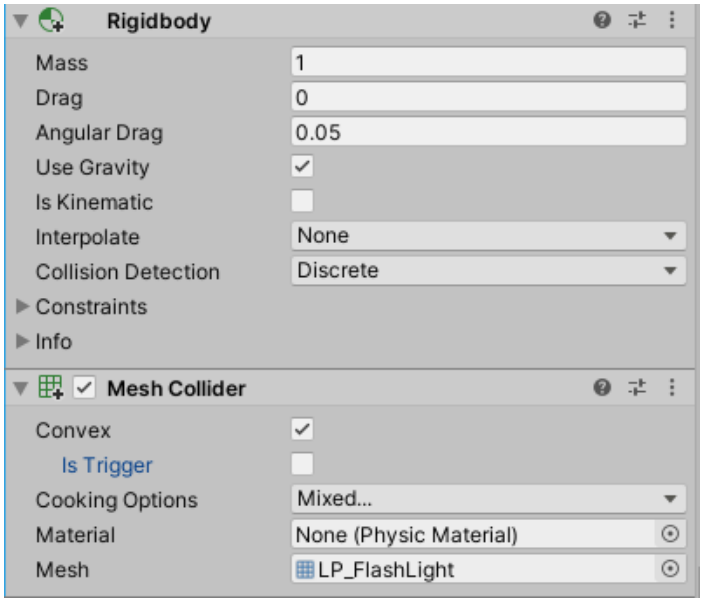
Un Direccional Light no tiene ninguna posición identificable de su fuente por lo que puede colocarse en cualquier lugar de la escena. Todos los objetos de la escena son iluminados desde la misma dirección por este objeto. La distancia de la luz respecto al objeto destino no está definida y por lo tanto la luz no disminuye.

Area Lights

Un Area Light está definido por un rectángulo en el espacio. La luz se emite en todas direcciones, pero solo por un lado del rectángulo. La luz incide sobre un rango específico. Dado que el cálculo de iluminación requiere mucho tiempo de procesador, los Area Lights no están disponibles en tiempo de ejecución y únicamente puede procesarse en Lightmaps. [10]

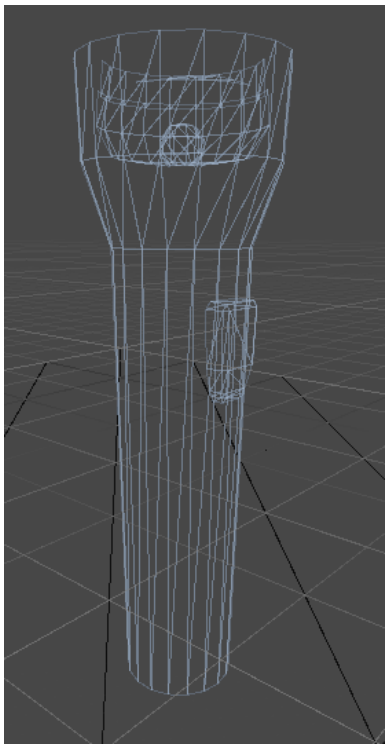
5.4.2 Solides y gravedad

El bunker y los objetos dentro del entorno virtual tendrán solides, es decir, no serán traspasables y se podrán coger e interactuar con ellos, esto mejora la inmersión del usuario.

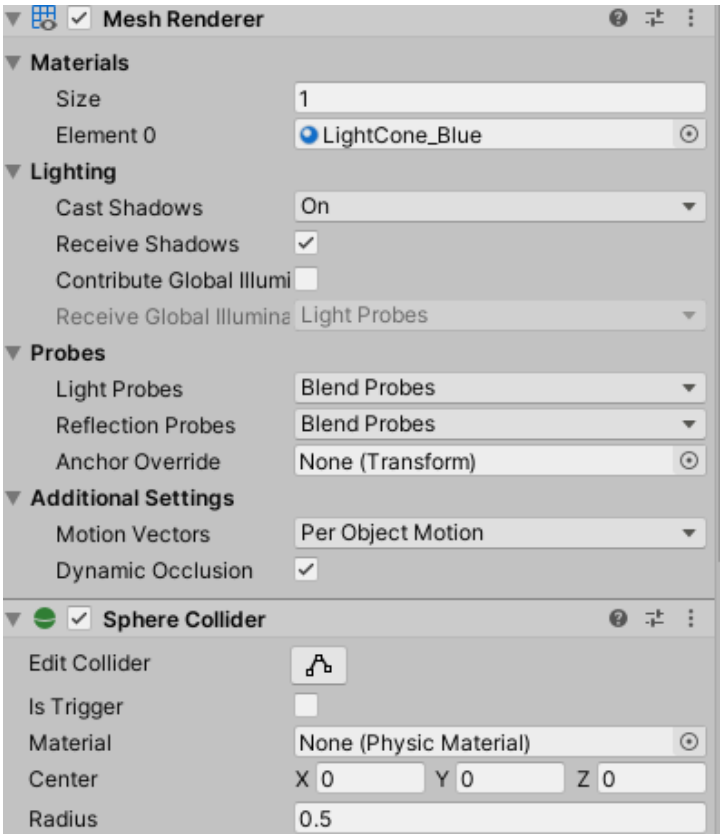


La gravedad se aplicará al usuario, permitiéndole un movimiento realista por el bunker, en zonas altas poder caer y a su vez bajar escalones, también se aplicará a los objetos, de manera que tras tomar un objeto y soltarlo, este caerá por inercia.

La herramienta Unity permite añadir complementos que dispongan de estas características tales como “Mesh Collider” o “Rigidbody” aplicable a los objetos, el terreno y al usuario, facilitando así la interacción entre los elementos y de esta manera puedan colisionar y no se atraviesen entre sí, sin embargo, también tendremos que hacer modificaciones que permitan una distinta interacción.



Contenedor de linterna



5.4.3 Movimiento del Usuario

Paquetes y librerías usadas:

Terrain Tools Sample Asset Pack: El paquete contiene una colección de herramientas para impulsar el desarrollo de territorios, cuenta con 35 cepillos de terreno además de terrenos de alta calidad, cañones, crestas, montañas, etc.

Además de seis capas de terreno con texturas listas para PBR de alta calidad, para así pintar tierra, musgo, roca, arena, pedregal y nieve. [11]

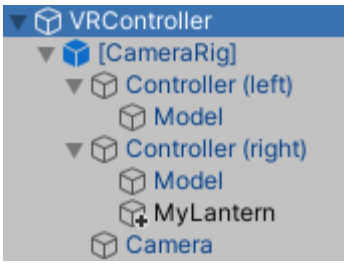
SteamVR Plugin: Valve mantiene un complemento de Unity para interactuar sin problemas SteamVR con Unity. Con SteamVR, los desarrolladores pueden apuntar a una API a la que se puedan conectar todos los auriculares VR populares. El moderno SteamVR Unity Plugin gestiona tres cosas principales para los desarrolladores: cargar modelos 3D para controladores de realidad virtual, manejar la entrada de esos controladores y estimar cómo se ve su mano mientras usa esos controladores. Además de administrar esas cosas, tenemos un ejemplo de Interacción de sistema para ayudarlo a que su aplicación de realidad virtual despegue. Proporcionar ejemplos concretos de interacción con el mundo virtual y nuestras API. [12]

Oculus Integration: Unity proporciona compatibilidad con realidad virtual integrada para dispositivos Oculus. El paquete de integración de Oculus agrega scripts, prefabricados, muestras y otros recursos para complementar el soporte integrado de Unity. El paquete incluye una interfaz para controlar el comportamiento de la cámara de realidad virtual, un prefabricado de control en primera persona, una API de entrada unificada para controladores, funciones de renderizado, herramientas de depuración y más. [13]

Se moverá a través del bunker usando el touchpad, con las direcciones indicadas, la altura del usuario se ajustará a su altura real, previamente se especificará la posición del suelo en la configuración de las estaciones base y luego ellas mismas establecerán la altura del usuario en base a los sensores de las gafas de realidad virtual.

```
public class VRcontroller : MonoBehaviour
{
    public float m_Gravity = 30.0f;
    public float m_Sentitivity = 0.1f;
    public float m_MaxSpeed = 0.1f;
    public float m_RotateIncrement = 90;

    public SteamVR_Action_Boolean mm_RotatePress = null;
    public SteamVR_Action_Boolean m_MovePress = null;
    public SteamVR_Action_Vector2 m_MoveValue = null;

    private float m_Speed = 0.0f;

    private CharacterController m_CharacterController = null;
    private Transform m_CameraRig = null;
    private Transform m_Head = null;

    private void Awake()
    {
        m_CharacterController = GetComponent<CharacterController>();

    }
    // Start is called before the first frame update
    private void Start()
    {
        m_CameraRig = SteamVR_Render.Top().origin;
        m_Head = SteamVR_Render.Top().head;
    }

    // Update is called once per frame
    private void Update()
    {
        //HandleHead();
        HandleHeight();
        CalculateMovement();

    }
```

VRcontroller: es un script que se encargará del desplazamiento, SteamVR plugin posee una interfaz “Player” para realizar actividades dentro de un entorno virtual, utiliza comandos ya estandarizados, como transportación a un punto, sin embargo, posee algunos inconvenientes, no genera su gravedad, no valora la altura del usuario y la opción de transportarse a un punto no es validad en este proyecto.



Partiendo de esto último crearemos desde cero un script “Player” que controle todo el movimiento que deseamos realizar a través de un touchpad, para esto enlazamos la cámara con el componente “Steam VR_Camera (script)” de la librería incorporada, ya que nos dará control del visor de VIVE PRO, esto es muy importante.

Awake() se llama cuando un GameObject activo que contiene el script se inicializa, cuando se carga una escena, o cuando un GameObject previamente inactivo

se establece en activo, o después de que se inicializa un GameObject creado con Object.Instantiate . Utilizamos Awake para inicializar variables o estados antes de que se inicie la aplicación.

Unity llama a Awake solo una vez durante la vida útil de la instancia del script. Su vida útil dura hasta que se descarga la escena que lo contiene. Si la escena se vuelve a cargar, Unity vuelve a cargar la instancia del script, por lo que se volverá a llamar a Awake. [14]

En este caso inicializa la variable m_CharacterController devolviéndole el componente del tipo del objeto.

Start() se llama cuando se habilita un script justo antes de que se llame por primera vez a cualquiera de los métodos de actualización.

Al igual que la función Awake, Start se llama exactamente una vez durante la vida útil del script. Sin embargo, se llama a Awake cuando se inicializa el objeto de secuencia de comandos, independientemente de si la secuencia de comandos está habilitada o no.

Nosotros lo utilizamos para inicializar las variables correspondientes a la cámara y a la posición de la cabeza original, usando los parámetros que devuelve el plugin de steamVR. [15]

Update() se llama por cada fotograma, si MonoBehaviour está habilitado, es la función más utilizada para implementar cualquier tipo de script de juego. No todos los scripts MonoBehaviour necesitan de Update. [16]

Aprovecharemos esto para llamar continuamente a las funciones que necesitaremos que se vayan actualizando a medida que interactuamos con el entorno.

Previamente inicializaremos las variables que necesitaremos a lo largo del programa, tales como la sensibilidad, la velocidad de desplazamiento, la gravedad y la rotación.

De la misma manera inicializaremos las variables correspondientes a los mandos de control que usaremos, estos dentro de las acciones del plugin SteamVR, incorporaremos la librería “Valve.VR”.

HandleHead(): Su propósito es la de controlar la posición y la rotación de la cámara, sin embargo

HandHeight(): Basándose en el movimiento de la mano calcula el nuevo posicionamiento del “Player” desplazando el centro de la capsula que lo contiene.

Calculate movement(): Toma la orientación devuelta por “CalculateOrientation” y aplica el movimiento en base a los datos aportados por el Touchpad en tiempo real.

CalculateOrientation(): Calcula la orientación basada en la rotación y usando los ángulos de Euler, especificando la orientación de un sistema de referencia de ejes ortogonales en movimiento con otro estático.

```
private void HandleHead()
{
    // Store current
    Vector3 oldPosition = m_CameraRig.position;
    Quaternion oldRotation = m_CameraRig.rotation;
    // Rotation
    transform.eulerAngles = new Vector3(0.0f, m_Head.rotation.eulerAngles.y, 0.0f);
    // Restore
    m_CameraRig.position = oldPosition;
    m_CameraRig.rotation = oldRotation;
}

private void HandleHeight()
{
    // Get the head in local space
    float headHeight = Mathf.Clamp(m_Head.localPosition.y, 1, 2);
    m_CharacterController.height = headHeight;

    // Cut in half
    Vector3 newCenter = Vector3.zero;
    newCenter.y = m_CharacterController.height / 2;
    newCenter.y += m_CharacterController.skinWidth;

    // Move capsule in local space
    newCenter.x = m_Head.localPosition.x;
    newCenter.z = m_Head.localPosition.z;

    // Rotate
    //newCenter = Quaternion.Euler(0, -transform.eulerAngles.y, 0) * newCenter;

    // Apply
    m_CharacterController.center = newCenter;

}
```

```
private void CalculateMovement()
{
    // figure out movement orientation
    //Vector3 orientationEuler = new Vector3(0, m_Head.eulerAngles.y, 0);
    Quaternion orientation = CalculateOrientation();
    Vector3 movement = Vector3.zero;

    // If not moving
    if (m_MoveValue.axis.magnitude == 0)
        m_Speed = 0;

    // add, clamp
    m_Speed += m_MoveValue.axis.magnitude * m_Sentitivity;
    m_Speed = Mathf.Clamp(m_Speed, -m_MaxSpeed, m_MaxSpeed);
    // Orientation
    movement += orientation * (m_Speed * Vector3.forward);
    // Gravity

    movement.y -= m_Gravity * Time.deltaTime;

    // Apply
    m_CharacterController.Move(movement * Time.deltaTime);

}
private Quaternion CalculateOrientation()
{
    float rotation = Mathf.Atan2(m_MoveValue.axis.x, m_MoveValue.axis.y);
    rotation *= Mathf.Rad2Deg;

    Vector3 orientationEuler = new Vector3(0, m_Head.eulerAngles.y + rotation, 0);
    return Quaternion.Euler(orientationEuler);
}
```

5.4.4 Interacción

Para este aspecto es necesario tener cuenta que un modelo de mano para nuestra simulación permitiría tener una inmersión más completa. Ya que podemos ver el tipo de acción que estamos generando en los mandos de control.

Debido a que no estamos usando el plugin estándar de SteamVR, lo que haremos será incorporar el modelo y las acciones de modelo a nuestro VRcontroller.

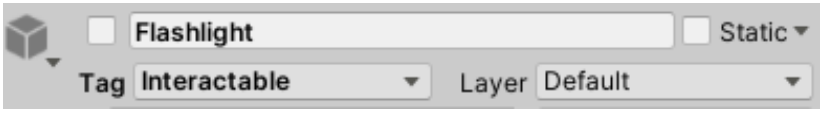
La interacción del usuario con el entorno se basa en coger y soltar cosas, además de la interacción con algún elemento en particular, por ejemplo, presionar un gatillo para encender una luz, este elemento ya ha sido explicado antes.

Partimos de la idea de dar una característica particular a los objetos, dándoles la etiqueta de “Interactable”, lo cual permite al usuario, interactuar con ese objeto.

También tenemos que dar la característica de uso, a las manos del usuario para que este pueda coger los objetos que hemos establecido como “Interactable”, para ello desarrollaremos el script “Hand”.

Previamente tendremos que establecer los controles que utilizaremos para esta acción, en nuestro caso el “Grip”, el cual tendremos que definir en el “SteamVR input”.

Interactable: hace que una variable no se muestre en el inspector, pero si sea serializada, esto permitirá darle la característica a los objetos que interaccionaran con “hand”.




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

[RequireComponent(typeof(Rigidbody))]
public class Interactable : MonoBehaviour
{
    [HideInInspector]
    public Hand m_ActiveHand = null;
}
```

Hand: Da la opción de interactuar con los objetos, tras definirlo lo incorporaremos a los modelos de mano izquierda y mano derecha.

Parte de la inicialización de los componentes de acción de los mandos de control establecidos en SteamVR plugin, para ello incorpora la librería "Valve.VR" y posteriormente incorpora una lista de objetos con los que se podrá interactuar.

Awake(): Inicializa las variables con los tipos correspondientes a los objetos que utilizara, en este caso "steamvr_behaviour_pose", el cual establece automáticamente la posición y la rotación de transformación en cada actualización para que coincida con la pose y "FixedJoint" que restringe el movimiento de un objeto para que dependa de otro objeto.

Update(): Se encarga de realizar las acciones coger y soltar basadas en el comportamiento de los mando, si presionamos el "Grip" y estamos en la posición del objeto, lo cogemos, si tenemos el objeto en la mano y soltamos el "Grip", el objeto caerá.

```
public class Hand : MonoBehaviour
{
    //
    public SteamVR_Action_Boolean m_GrabAction = null;

    private SteamVR_Behaviour_Pose m_Pose = null;
    private FixedJoint m_Joint = null;

    private Interactable m_CurrentInteractable = null;
    public List<Interactable> m_ContactInteractables = new List<Interactable>();

    void Awake()
    {
        m_Pose = GetComponent<SteamVR_Behaviour_Pose>();
        m_Joint = GetComponent<FixedJoint>();
    }

    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        // Down
        if (m_GrabAction.GetStateDown(m_Pose.inputSource))
        {
            print(m_Pose.inputSource + "Trigger Down");
            Pickup();
        }
        if (m_GrabAction.GetStateUp(m_Pose.inputSource))
        {
            print(m_Pose.inputSource + "Trigger Up");
            Drop();
        }
    }
```

Pickup(): comprueba cual es el objeto más cercano con el que podemos interactuar, al realizar la acción con los mando lo coge dándole así la propiedad m_ActiveHand definida en “Interactable”, de esta manera podemos tomar el objeto dándole solides junto a nuestra mano.

Drop(): previamente parte de tener un objeto en la mano, de esta manera al soltar el “Grip”, soltamos el objeto, pero no solo eso, podemos aplicarle una velocidad y dirección a la hora de soltarlo, de esta manera lo estamos lanzando hacia otro lugar.

```
private void OnTriggerEnter(Collider other)
{
    if (!other.gameObject.CompareTag("Interactable")) return;
    m_ContactInteractables.Add(other.gameObject.GetComponent<Interactable>());
}

private void OnTriggerExit(Collider other)
{
    if (!other.gameObject.CompareTag("Interactable")) return;
    m_ContactInteractables.Remove(other.gameObject.GetComponent<Interactable>());
}

public void Pickup()
{
    //nearest
    m_CurrentInteractable = GetNearestInteractable();
    //
    if (!m_CurrentInteractable) return;
    //
    if (m_CurrentInteractable.m_ActiveHand) m_CurrentInteractable.m_ActiveHand.Drop();
    //
    m_CurrentInteractable.transform.position = transform.position;
    //
    Rigidbody targetBody = m_CurrentInteractable.GetComponent<Rigidbody>();
    m_Joint.connectedBody = targetBody;
    //
    m_CurrentInteractable.m_ActiveHand = this;
}
public void Drop()
{
    //
    if (!m_CurrentInteractable) return;
    //
    Rigidbody targetBody = m_CurrentInteractable.GetComponent<Rigidbody>();
    targetBody.velocity = m_Pose.GetVelocity();
    targetBody.angularVelocity = m_Pose.GetAngularVelocity();
    //
    m_Joint.connectedBody = null;
    //
    m_CurrentInteractable.m_ActiveHand = null;
    m_CurrentInteractable = null;


}
```

GetNearestInteractable(): Nos permite interactuar con el objeto más cercano, con una distancia mínima.

```
private Interactable GetNearestInteractable()
{
    Interactable nearest = null;
    float minDistance = float.MaxValue;
    float distance = 0.0f;

    foreach (Interactable interactable in m_ContactInteractables)
    {
        distance = (interactable.transform.position - transform.position).sqrMagnitude;
        if (distance < minDistance)
        {
            minDistance = distance;
            nearest = interactable;
        }
    }
    return nearest;
}
```

5.4.5 Eventos

Los eventos son un aspecto muy importante a la hora de realizar una simulación en un entorno virtual, siempre y cuando queramos hacerlo interactivo.

Que el lugar en el que se encuentre el usuario establezca distintos tipos de eventos esto crea en el usuario la ilusión de ser parte de los acontecimientos que se están generando, en este proyecto el único evento que se genera en base al lugar del usuario son las guías auditivas de información sobre la zona y curiosidades del terreno o la estructura.

Otros eventos a destacar serían los generados por decisión propia del usuario, por ejemplo, al seleccionar el bunker al que desea ir, ello activa una serie de acontecimientos que ocurrirán en el lugar deseado.

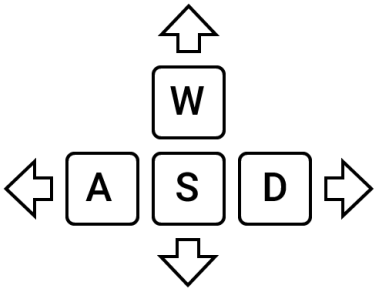
Varios de los eventos pueden ser audiovisuales, generando un aprendizaje por medio de interacciones con otras personas simuladas que actúen bajo un criterio establecido, por aprendizaje o por un árbol de estados.

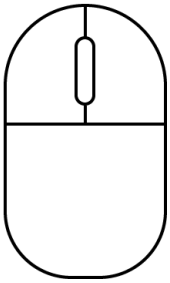
Estos eventos pueden cambiar la perspectiva del tiempo y aprender por una simulación temporal, por ejemplo, al romper una estalactita dentro de una cueva, podemos retroceder en cámara rápida mientras observamos todo el proceso que ha tenido que pasar para que se crease esa estalactita, permitiendo así enseñar la importancia y el peso que tiene el tiempo en la creación de las cosas.

6 Pruebas

6.1 Prototipo 1

Durante los primeros meses de proyecto a falta de las gafas de realidad virtual, se tuvo que enfocar la simulación en un entorno 2D, usando el teclado para movilizarnos por el bunker, teniendo como referencia las teclas estándar de movimiento.





"W": movimiento hacia delante.

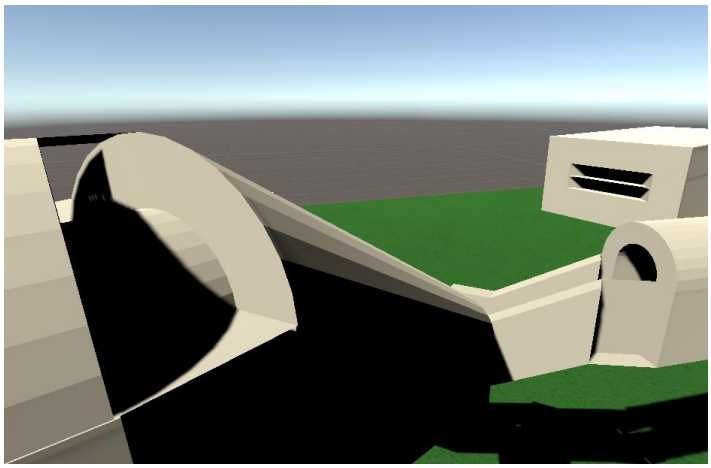
"S": movimiento hacia atrás.

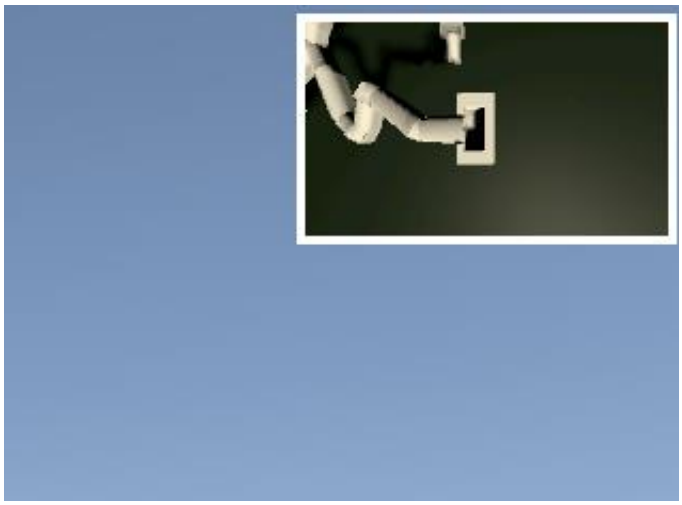
"A": movimiento hacia la izquierda.

"D": movimiento hacia la derecha.

Ratón: como objetivo de la vista, orienta el rumbo que se quiere tomar.

El modelo 3D del bunker tenía muchos errores de diseño, creando espacios vacíos debido a la mala integración de la pieza en Blender, las cuales una vez llevadas a Unity, se notaban en la entrada de luz por zonas cerradas, estas piezas tenían fallos debido a la dirección errónea de la normal de algunas caras, tras conocer el error posteriormente se pasó a corregirlo para futuras versiones.





Otra herramienta que se desarrollo fue la incorporación de un mapa en la posición superior derecha de la pantalla, utilizando canvas y captura de imagen a tiempo real, sin embargo, una vez desplegado en las gafas de realidad virtual, esta opción dejo de tener sentido, dado que no se apreciaba y a su vez desconcertaba al usuario, era una opción más útil en un entorno 2D.

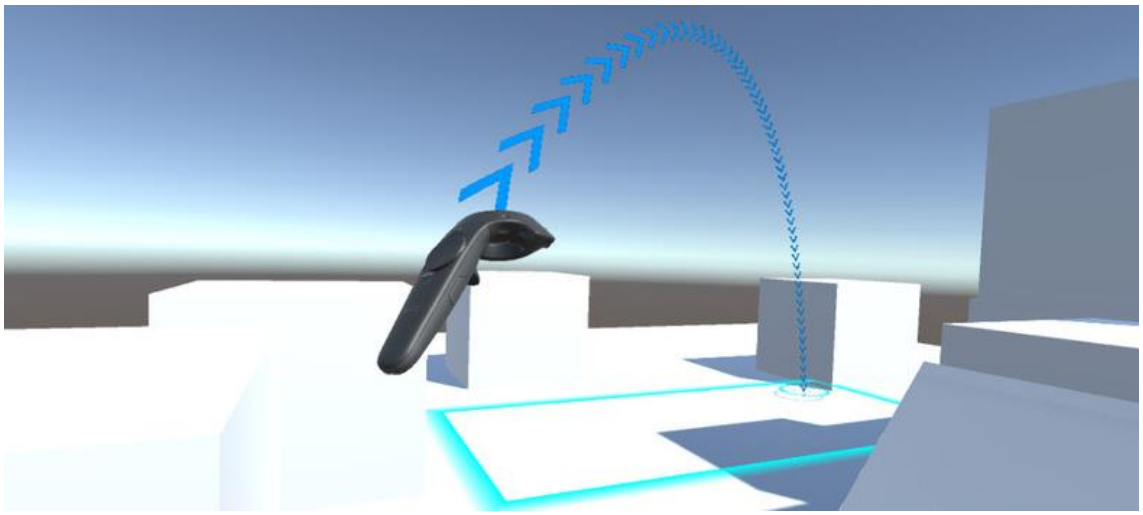
Aun así, estaría bien desarrollar esta opción en un entorno 3D como un desplegable del brazo izquierdo o parte de un menú interno.

6.2 *Prototipo beta*

El prototipo beta se desarrolló primero con la librería de Oculus, la cual ya contaba con un elemento "OVRplayerController" que cumplía con la función de usuario, contaba con los modelos de manos, distintas interacciones, un movimiento correcto, aspectos de velocidad de desplazamiento y gravedad, además permitía usar la aplicación en modo teclado en caso no detectara las gafas de realidad virtual.

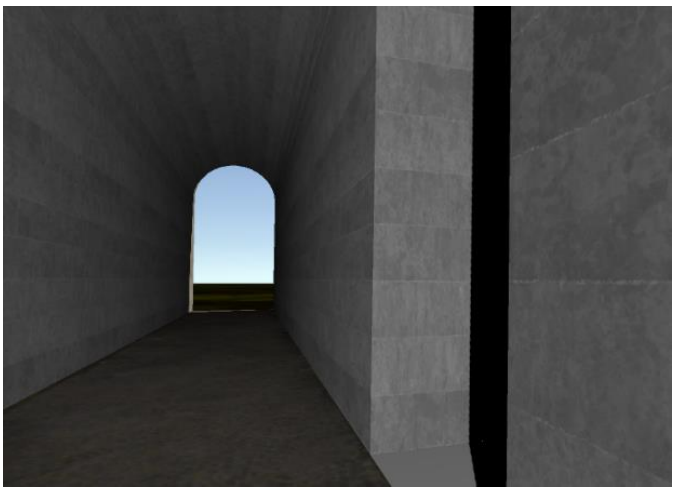
Más adelante con la adquisición de las VIVE PRO pasamos a desarrollar la aplicación con el plugin SteamVR para adaptarlo lo más posible al hardware que poseíamos, sin embargo, este no contaba con aspectos tales como la velocidad, se movía por transportación a un punto y no tenía gravedad.

El modelo 3D ya estaba casi completo, pero con ausencia de texturas, las cuales no podíamos establecer debido a la problemática de desplazamiento de la situación actual.



VR teleporter

6.3 *Versión final*



La versión final cuenta con las texturas correspondientes a la zona, dado que se logró ir al lugar en el que está basado el entorno virtual, se tomó fotos de los muros, la flora, los suelos, el entorno y detalles varios que, al incorporarse, aumentó la inmersión del usuario.

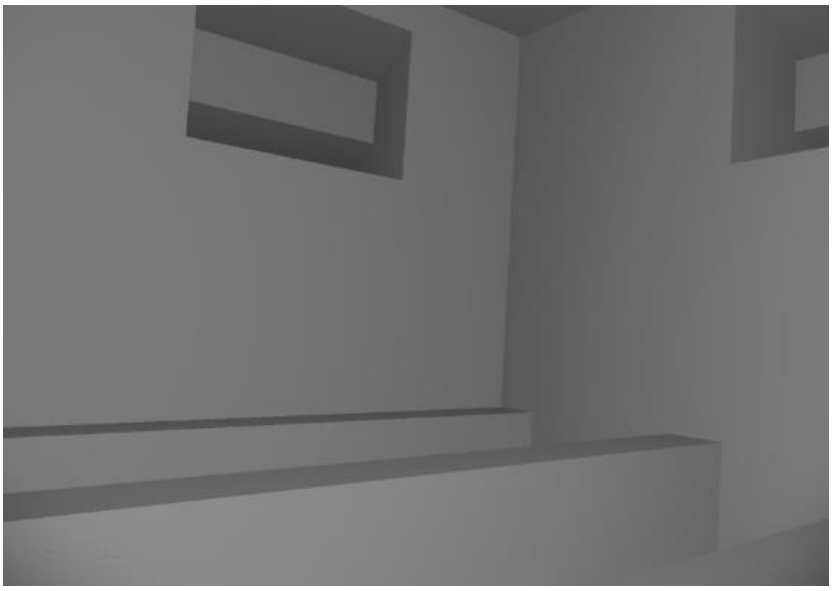
Aunque los guías están establecidos en el entorno virtual los audios no corresponden en su totalidad a la información que queríamos establecer, debido a que la persona que nos iba a proporcionar esa información, no ha podido contactar con

nosotros tras escribirle, por ello se ha usado fragmentos de otros videos explicativos de la zona, entrevistas y reportajes.

Esta versión trabaja con un VRController creado desde cero, que cumple con las especificaciones que queríamos, la interacción con los objetos y el desplazamiento por touchpad.

Se han añadido imágenes del entorno apreciables desde dentro de las casetas del bunker.

Una vez hemos conseguido ir al lugar, tenemos una mejor recopilación de datos y medidas que nos ha permitido establecer la estructura, lo más fiel posible a la realidad, incluso llevándolo a un estilo de la época de la posguerra civil.



7 Conclusiones

7.1 Impacto

Considero que este proyecto podría abrir puertas a actividades turísticas o de índole educativo en Navarra, son muchas las situaciones actuales por las que es más difícil realizar una visita a algún lugar de la zona, ya sea por dificultad de movilidad, por preservación arqueología/histórica, por dificultad económica o desinterés educativo.

El turismo cultural es uno de los tipos de turismo que más gente mueve alrededor del mundo. Los núcleos receptores de este turismo suelen ser las ciudades históricas o que tengan algún valor patrimonial, con lo que es necesario trasladarse físicamente a estos lugares.

La gran afluencia de gente pone en peligro muchas veces la conservación de estos espacios o monumentos. Y en otras ocasiones la accesibilidad o el entendimiento de los conjuntos no está suficientemente bien adaptada. [17]

7.2 Problemática

Posiblemente aumento de visitas turísticas en realidad virtual podría generar desinterés por visitar esos lugares físicamente, esto supondría la reducción en inversión en hostelería y otros negocios que se nutren del turismo.

Sin embargo, una correcta orinecían podría llevar a las personas a quedarse con más gana de visitar esos lugares, un ejemplo podría ser el hacer un recorrido por un museo y que cierta zona sea de acceso exclusivo, en el caso de montes o cuevas, el hermoso paisaje ínsita a las personas a estar en ese lugar físicamente y vivir esa experiencia.

Otros problemas sí que están más relacionados con el uso del dispositivo, el exceso de uso de las gafas de realidad virtual si estas no están correctamente graduadas, generando así mareos y nauseas, si no se controla su exceso de uso puede generar problemas de salud por la falta de movimiento y adicción a la realidad virtual, sin embargo, si se es consciente de ello las mismas gafas provocan un cansancio de las mismas, asiendo así por decisión propia el tomarse un descanso de ellas.

Aun así, este tipo de tecnologías sigue en desarrollo y tiene un amplio abanico de aspectos saludables que se pueden desarrollar, a nivel médico, mejorando la concentración y los reflejos, a nivel psicológico, estudiando el comportamiento social y situacional, etc.

seco.org (realidad virtual al servicio de la medicina)

8 Líneas futuras

8.1 Hacia el proyecto

- Añadir herramientas de accesibilidad a personas con dificultad de movilidad.

Aunque una persona con dificultad de movilidad pueda realizar la visita, hay componentes que podrían mejorar su interacción, por ejemplo, añadir la opción “elevador”, de esta manera si es una persona en silla de ruedas, podría usar un botón para aumentar su altura y poder ver por las aperturas de las casetas del bunker.

- Permitir que el usuario tenga un mapa del bunker y visite todas las zonas del lugar simplemente dándole a la zona del bunker que desea visitar, y aparezca automáticamente allí.

- Implementar una versión para móvil y requiera un joystick.

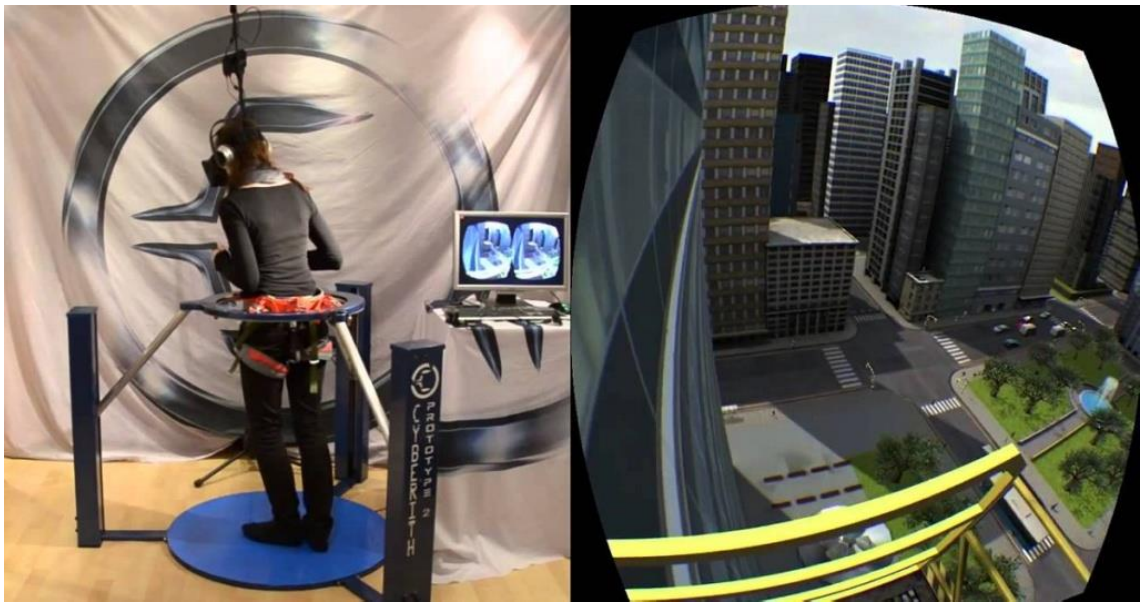
Móviles compatibles con la realidad virtual

8.2 Hacia futuras investigaciones

Me gustaría seguir esta rama de la informática y dedicarme a la investigación de más forma de aumentar la inmersión en un entorno de realidad virtual, reduciendo también las herramientas físicas que se requerirían para ello.

Ya hay proyectos que buscan que la interacción con el entorno requiera de menos artefactos y se base en un aprendizaje del individuo que lo usa, ya sea por un mapeado neuronal de las zonas que se utilizan al realizar actividades o basándose en un software de aprendizaje que interactúa en base al entrenamiento de las acciones del individuo y logrando adaptar los eventos a sus gustos o nivel.

Otro aspecto de esta rama que me gustaría investigar y desarrollar es el comportamiento humano ante distintas situaciones simulas, ya que al trabajar en el proyecto de fin de grado he descubierto que cosas tan simples como un error de posicionamiento al encontrarte en una zona elevada puede hacer que tengas la sensación de caída, aumentando tu vértigo, podría usarse de la misma manera para tratar fobias o manías de las personas.



Combate fobias a través de la realidad virtual

9 Bibliografía

9.1 Material

- Imágenes y mapas otorgados por Gabinete Trama S.L.
- Audios y testimonios por el profesor Fernando Mendiola
- Texturas de edificaciones y ambiente por Textures.com y Cgtrader.com

9.2 Referencias

[1] Página oficial de VIVE-PRO:

(https://www.vive.com/mx/product/vive-pro/)

[2] Blender

(https://es.wikipedia.org/wiki/Blender)

[3] Unity

(https://es.wikipedia.org/wiki/Unity_(motor_de_videojuego))

[4] Investigando el comportamiento de los peatones con VR y AR (edificios virtuales)

(https://www.thenewnow.es/innovacion/comportamiento-peatones-vr-ar-edificios-virtuales/)

[5] Half-Life: Alyx:

(https://www.half-life.com/es/alyx)

[6] Virtuix Omni

(https://www.xataka.com/realidad-virtual-aumentada/virtuix-omni-el-complemento-para-movernos-y-sudar-en-realidad-virtual)

[7] Snippet:

(https://www.nature.com/articles/s41598-019-45422-6)

[8] Imágenes usadas para establecer las texturas del medio ambiente:

(https://www.cgtrader.com/es/3d-modelos)

[9] Google Resonance Audio SDK:

(https://sevilla.abc.es/mobility/noticia/android/noticias-android/resonance-audio-el-nuevo-sistema-de-sonido-de-google-para-vr/)

[10] Manual de luces en unity:

(https://docs.unity3d.com/Manual/Lighting.html)

[11] Terrain Tools Sample Asset Pack:

(https://assetstore.unity.com/packages/2d/textures-materials/nature/terrain-tools-sample-asset-pack-145808)

[12] SteamVR Plugin:

(https://assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647)

[13] Oculus Integration:

(https://assetstore.unity.com/packages/tools/integration/oculus-integration-82022)

[14] Manual de función Awake:

(https://docs.unity3d.com/ScriptReference/MonoBehaviour.Awake.html)

[15] Manual de función Start:

(https://docs.unity3d.com/ScriptReference/MonoBehaviour.Start.html)

[16] Manual de función Update:

(https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html)

[17] Realidad virtual aplicada al turismo

(https://editeca.com/realidad-virtual-turismo/)

9.3 Asesoría

- Profesor Jesús Villadangos Alonso (Tutor)
  - Aportación: información, contacto y relación con Gabinete Trama S.L., entorno de trabajo, materiales y dispositivos para el desarrollo (kit de realidad virtual VIVE PRO, ordenador de trabajo con especificaciones de componentes necesarios), apoyo e interés por el proyecto y tutoría en relación al trabajo de fin de grado.

9.4 Influencia

- DR. Mar Gonzalez Franco, investigadora del equipo EPIC de Microsoft Research.