



**UNIVERSIDAD
DE GRANADA**

**TRABAJO FIN DE GRADO
INGENIERÍA INFORMÁTICA**

GodPic

Un juego *Point & Click* en Realidad Virtual

Autor

Antonio Galdó Seiquer

Director

Francisco Luis Gutiérrez Vela



Escuela Técnica Superior de Ingenierías Informática y de
Telecomunicación

—
Granada, septiembre de 2020



GodPic

Un juego *Point & Click* en Realidad Virtual.

Autor

Antonio Galdó Seiquer

Director

Francisco Luis Gutiérrez Vela

Godpic: Un juego Point & Click en Realidad Virtual

Antonio Galdó Seiquer

Palabras clave: videojuego, realidad virtual, Point & Click, software, Unity.

Resumen

El proyecto consiste en un videojuego en tres dimensiones para realidad virtual basado en el clásico estilo de aventuras Point & Click. Se han adaptado las mecánicas y los distintos tipos de retos a un mundo 3D, donde las escenas estarán en cuadros como si se tratase de un museo, y se interactúa con objetos y personajes que hay en ellas.

Para avanzar en la aventura, el personaje deberá moverse en el entorno 3D, acercándose a las distintas escenas e investigando salas, las cuales representarán distintos escenarios.

El juego se ha desarrollado con un plan de entregas, con una metodología ágil basada en Scrum. Se ha avanzado por etapas, centrándose en el usuario y en las continuas evaluaciones del software.

GodPic: A Virtual Reality *Point & Click* game

Antonio Galdó Seiquer

Keywords: videogame, virtual reality, Point & Click, software, Unity.

Abstract

The project consists of a three-dimensional video game for virtual reality based on the classic Point & Click adventure style. The mechanics and the different types of challenges have been adapted to a 3D world, where the scenes will be in images as if it were a museum, and there is interaction with objects and characters that are in them.

To advance in the adventure, the character must move in the 3D environment, approaching the different scenes and investigating rooms, which will represent different scenarios.

The game has been developed with a delivery plan, with an agile methodology based on Scrum. Progress has been made in stages, focusing on the user and continuous evaluations of the software.

Yo, **Antonio Galdó Seiquer**, alumno de la titulación **TITULACIÓN de la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 77558481C, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Antonio Galdó Seiquer

Granada a 8 de septiembre de 2020.

D. **Francisco Luis Gutiérrez Vela**, Profesor del Área de XXXX del Departamento YYYY de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***Godpic, Un juego Point & Click en Realidad Virtual***, ha sido realizado bajo su supervisión por **Antonio Galdó Seiquer**, y autorizo la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 8 de septiembre de 2020.

El director:

Francisco Luis Gutiérrez Vela

Agradecimientos

Quiero agradecer a Guillermo Díaz (Willyrex) por ser la piedra angular de gran parte del entretenimiento audiovisual de habla hispana, lo cual ha sido una inspiración y ha marcado mis gustos y aficiones, haciendo posible este trabajo.

También doy gracias a mis padres, a mi novia y amigos por ser un gran apoyo, aunque unos más que otros.



UNIVERSIDAD
DE GRANADA



GodPic



Antonio Galdó Seiquer

I. Introducción	21
II. Estado del arte	22
III. Análisis inicial del problema	30
IV. Tecnología a usar	34
V. Metodologías a usar en el proyecto	40
VI. Plan de entregas	43
VII. Desarrollo. Entregas e iteraciones	45
VIII. Conclusiones y trabajo futuro	62
IX. Bibliografía	63

I. INTRODUCCIÓN

El trabajo será un videojuego en realidad virtual basado en el estilo *Point & Click* de juegos 2D, por lo que se interactuará directamente sobre las escenas con los controles de RV.

Anteriormente he trabajado en Unity haciendo otro videojuego, y pese a que me parezca un mundo de gran complejidad, me gusta ver que tu trabajo se puede ver reflejado en un entretenimiento para alguien. Todo el mundo que haya probado la realidad virtual sabe perfectamente que es el futuro, pese a que exista desde hace bastantes años, pero aún cuesta que se implante del todo en la industria.

La industria de los videojuegos ha avanzado enormemente en los últimos años, llegando a mover más dinero que el cine o la música, así que no es algo que se deba tomar a la ligera. En 2018 el CEO de Netflix reconoció que los videojuegos eran la principal competencia de las plataformas como la suya. No hay más que ver la última estrategia de Amazon: permitir reproducir contenidos de Amazon Prime a través de la plataforma Twitch, puesto que esta es mucho más rentable que su plataforma de películas y series, ya que se transmiten principalmente videojuegos.

Por todo esto, hacer un juego en RV me parece una opción perfecta para el trabajo, y puede ser muy interesante hacer un juego de aventuras con unas mecánicas sin demasiada complejidad, como puede ser un *Point & Click*.

El objetivo es hacer un juego de estilo *Point & Click* en realidad virtual con su documento de diseño de juego (GDD), a parte de esta memoria. Primero empezaré con la documentación y la memoria, después empezaré a desarrollar el mundo 3D en Unity, y más tarde añadiré las mecánicas de realidad Virtual. Por último quedará la narrativa y diseñar a fondo todos los niveles.

II. ESTADO DEL ARTE

II.1. REALIDAD VIRTUAL

La Realidad Virtual es la percepción de un entorno, de objetos o escenas, generado a partir de un ordenador.

Para entender la realidad virtual tenemos que saber que se encuentra en un espectro de lo que llamamos realidad mixta. Esto engloba todo lo que hay entre el mundo real y el mundo virtual, mezclando estos aspectos en mayor o menor medida. Muy cerca del mundo virtual tenemos la Realidad Virtual (que no debe tener elementos del mundo real al ser un entorno completamente aislado) y, en el otro lado podemos encontrar la realidad aumentada.



Figura 1: Diagrama de realidad-virtualidad de Paul Milgram (1994)

La realidad aumentada es la percepción de un entorno real (no virtual), el cual realzamos con información generada por ordenador. Esta información debe ser interactiva en tiempo real, para simular que los elementos pertenecen al mundo real.

Solemos tener la idea de que esto afecta solo para elementos visuales o audiovisuales, pero también es posible usar realidad aumentada con elementos de más tipos como exclusivamente audio. La percepción de esta realidad puede ser directa o indirecta según el tipo de elemento que usamos para percibir esta información “extra”, que puede ser una pantalla, o un proyector que muestra imágenes sobre los objetos directamente.

Estos conceptos mayormente no tienen sentido en realidad virtual puesto que aquí simulamos un entorno completo (con imagen, audio, e incluso tacto si nos lo permite la tecnología). Se da por hecho que en la Realidad Virtual solo hay una manera de percibir este entorno, y es por medio de algún dispositivo que nos introduzca en esta realidad directamente, aislándonos del “mundo real”.

En la realidad virtual podemos distinguir entre varios tipos: completamente inmersiva (con unas gafas y unos mandos para interactuar, y aislarnos completamente del mundo real), parcialmente inmersiva (como un simulador de vuelo, en el que solamente se ve una pantalla curva), o no inmersiva (algun tipo de entretenimiento social entre varios usuarios en el que participemos, como un videojuego multijugador).

Usualmente cuando la gente habla de realidad virtual, se refiere a la que es completamente inmersiva, y es la que se tratará aquí también, y con la que se ha trabajado para hacer el juego. Para conseguir inmersión es necesario un dispositivo como unas gafas de RV, que es precisamente lo que se ha usado.

En 2010 se creó el primer prototipo de unas gafas de RV modernas y destinadas al posible uso de todo el mundo (originalmente basadas en un casco de realidad virtual anterior) que fueron las Oculus Rift; en 2012 se lanzó un [Kickstarter del proyecto](#) y consiguió mucho más del objetivo inicial. El modelo fue evolucionando y en 2014 Facebook compró Oculus por 2300 millones de dólares.



Figura 2: Primer modelo de Oculus Rift



Figura 3: Oculus Quest

Oculus ha seguido innovando en gafas de realidad virtual, llegando a sacar por ejemplo las Oculus Quest, que son totalmente independientes de un ordenador. Teniendo una capacidad de almacenaje de 64Gb o 128Gb, con su sistema operativo se vuelven en un dispositivo completamente autónomo, y sin cables ni sensores externos. Se podría decir que una versión económica de este modelo es el Oculus Go, también autónomo, pero con mucha menos capacidad gráfica y de almacenamiento, y un solo mando menos potente.

En abril de 2016 la empresa HTC sacó a la venta VIVE, diseñado para usarse de forma recreativa, muy orientado a la plataforma Steam (de Valve) para juegos de ordenador. Entre este año y el siguiente se sacaron las versiones definitivas de Playstation VR (antes conocido como proyecto Morpheus), que era el equivalente de HTC VIVE para ordenador, pero exclusivo para su uso con PlayStation. Recientemente la compañía Valve ha lanzado al mercado su propio sistema Valve Index, que es compatible con HTC VIVE y tiene una calidad gráfica realmente alta.



La mayoría de estos modelos de gafas o cascos son un sistema de reproducción de imagen y una interacción por movimiento, por lo que para ejecutar un programa (juego o película) necesitamos un ordenador (la mayoría necesitan Windows) con una potencia considerable, sobre todo en el apartado gráfico.



Figura 5: HTC VIVE

*Figura 6: HTC VIVE PRO
(último modelo)*

También es interesante mencionar Google Cardboard, que es un dispositivo armable de Google de muy bajo coste, y que funciona introduciendo el móvil, por lo que no depende de ningún ordenador. Esto hace que sea mucho más accesible para cualquier persona, pero permite mucha menos interacción y calidad.



Figura 7: Google Cardboard

La interacción para realidad virtual es un tema especial, puesto que no podemos usar un mando normal de consola, ni mucho menos un teclado y ratón, porque nos sacarían de esa realidad al abstraernos en el manejo del personaje. Para realidad virtual se suelen usar mandos especiales (como los que podemos ver en las imágenes) con sensores de posición muy potentes. Esto permite simular nuestras manos en un videojuego, llegando a poder manejar cada dedo en algunas ocasiones.

Hay diferencias según el modelo, podemos ver por ejemplo que con Oculus Quest no se necesitan sensores extra porque los llevan incorporados los mandos. En otros sistemas de RV como Rift o Vive, los mandos envían una señal de posición pero necesitan sensores externos. Los mandos son siempre inalámbricos, puesto que a veces es necesario un movimiento bastante agresivo, no así como los sensores, que suelen ir conectados al ordenador por USB.

II.2. VIDEOJUEGOS: POINT & CLICK

El proyecto va a ser un videojuego de estilo Point & Click, pero ¿qué significa esto?

Todo el mundo sabe lo que es un videojuego: un juego donde una o más personas interactúan con una interfaz representada por imágenes. Dentro de toda la

variedad que podemos ver en los videojuegos encontramos los juegos de tipo aventura.

Un juego de tipo aventura es aquel en el que el usuario toma el protagonismo de una historia, y tiene que resolver algunos problemas o puzzles para poder completar la historia del personaje en cuestión. Este tipo de videojuegos empezaron en 1976, siendo Colossal Cave Adventure el primero de este tipo que contenía un componente narrativo.

Podría decirse que casi todos los juegos actuales son en parte de aventura, pero la mayoría tienen componentes de otros géneros como acción o rol, por lo que no es tan fácil encontrar puramente juegos de aventura. Actualmente el tipo de juegos de aventura que tiene más popularidad son las aventuras gráficas. Dentro de estas podemos encontrar desde juegos en 2 dimensiones sin ningún diálogo como Limbo (Playdead – 2010) hasta películas interactivas como son los últimos juegos de la empresa Quantic Dream: Fahrenheit (2005), Heavy Rain (2010), Beyond: Two Souls (2013), y Detroit: Become Human (2018).

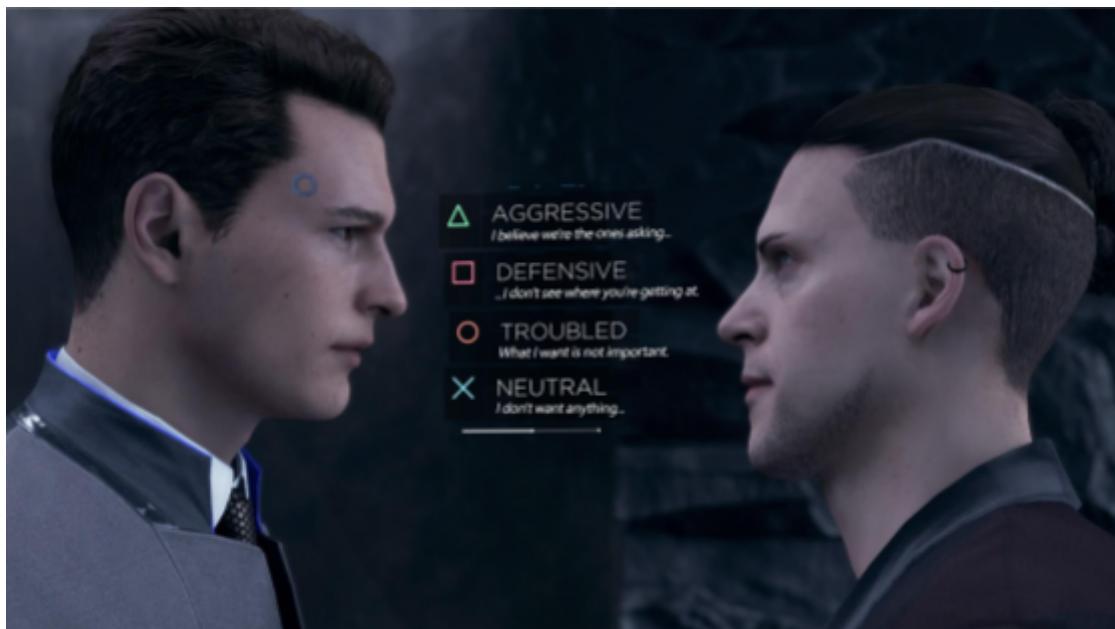


Figura 8: diálogo de Detroit: Become Human

Uno de los subgéneros más abundantes de este tipo son las aventuras Point & Click, las cuales se basan en una mecánica muy sencilla: mover el cursor y hacer clic donde corresponda para ir completando una serie de puzzles y avanzar por diferentes pantallas o escenas. Aquí se pueden encontrar maravillas como King's Quest (saga de Sierra On-Line) la cual está considerada la primera aventura gráfica de la historia, pero en las primeras ediciones aún no era del estilo Point & Click, o Monkey Island (saga de Lucasfilm Games), la cual perfecciona la esencia de las aventuras Point & Click con el motor SCUMM (creado por ellos para su anterior juego en 1987).



Figura 9: escena de *The Secret of Monkey Island Special Edition*

En una aventura *Point & Click* suele haber siempre unas mecánicas y un estilo de jugar similar con estos elementos:

- Se ve al personaje en tercera persona: la cámara está externa a la escena, ya sea con una perspectiva isométrica o paralela si es más bien en 2 dimensiones.
- Se hace clic en el suelo para mover al personaje, y este, si puede, se moverá automáticamente a la posición.
- Objetos: se puede interactuar con ellos ya sea actuando directamente sobre ellos (como un botón) o cogiéndolos y guardándolos en el inventario. También se suele poder arrastrar objetos del inventario (o seleccionarlos de alguna manera) para que interactúen sobre otros objetos o personas.
- Personas: siempre está la opción del diálogo, en la que tendremos que elegir entre varias opciones, o ver un diálogo de la historia del juego; o en su defecto, podremos interactuar con ellas de alguna manera con algún objeto (bien seleccionándolo o que se detecte automáticamente).

Los **tipos de retos** que suele haber en estas aventuras se basan en encontrar un objeto clave, ya sea adivinando cuál nos hace falta, buscando dónde está o haciendo algún desafío intermedio. Es muy usual que necesitemos varios objetos para juntarlos y obtener un objeto nuevo, o que tengamos que resolver algún acertijo para que nos den cierto objeto o nos dejen avanzar.

También puede haber mecánicas de interactuar con un personaje a través del diálogo, y tener que elegir una opción con una información que hayamos descubierto previamente, ya sea con un objeto o hablando con otro personaje. En muchos de estos juegos hay distintas opciones para interactuar con los personajes y los objetos, para darle un añadido de complejidad ("mirar", "hablar", "dar", "usar", "tirar", "abrir", etc).

Son muy populares las aventuras Point & Click en las que el jugador tiene que escapar de un sitio, pero no sabe lo que tiene que hacer a priori. En este tipo de juegos tendrá el reto añadido de descubrir qué es lo que tiene que hacer, o qué objetos tiene que buscar.

A continuación un ejemplo de un caso típico: nosotros necesitamos una llave para avanzar, y un personaje nos dice que nos la proporcionará si le entregamos un objeto, pero para obtener ese objeto tenemos que ir a la sala anterior y buscarlo, porque había pasado desapercibido antes. En cuanto interactuemos con el objeto sobre el personaje, se abrirá un nuevo diálogo y nos dará la llave, con la cual podremos interactuar con la puerta y avanzar.

II.3. ANTECEDENTES Y EJEMPLOS SIMILARES

Antes de nada, cabe mencionar que al juntar realidad virtual y Point & Click, se va más allá de la tecnología clásica Point & Click porque hay más maneras posibles de interactuar, pero que en el fondo siguen siendo interactuar con un botón.

Tabletop Simulator VR (Berserk Games - 2016)

El famoso simulador de juegos de mesa de Steam sacó en 2016 la compatibilidad con HTC Vive y Oculus Rift. Es interesante ver qué tipo de mecánicas se pueden adaptar a la realidad virtual en un juego como este. Tiene una serie de menús con diferentes opciones para crear objetos y modificarlos, y en el mundo 3D, con los objetos (dados, cartas...) se puede interactuar de varias formas. Destaca la mecánica de agitar un mazo de cartas para barajar, o de hacer clic en un dado con un botón determinado para lanzarlo hacia arriba automáticamente (a parte se puede coger y lanzar de manera normal).



Figura 10: Captura de Tabletop Simulator con HTC Vive

Psychonauts in the Rhombus of Ruin (Double Fine Productions - 2017)

Sigue la historia de Psychonauts y con mecánicas similares al juego anterior, que son interactuar haciendo clic pero con el jugador moviéndose en un mundo 3D e interactuando con todo ese mundo. Está disponible en PlayStation 4 y Steam.



Figura 11: Captura de Psychonauts in the Rhombus of Ruin

Wayward sky (Uber Entertainment - 2016)

Juego exclusivo de Playstation VR donde puedes jugar desde la perspectiva del personaje (interactuando con objetos) o en tercera persona con una vista desde arriba, donde debes hacer clic e indicar a dónde tiene que ir el personaje.



Figura 12: Captura de Wayward sky (parte Point & Click)

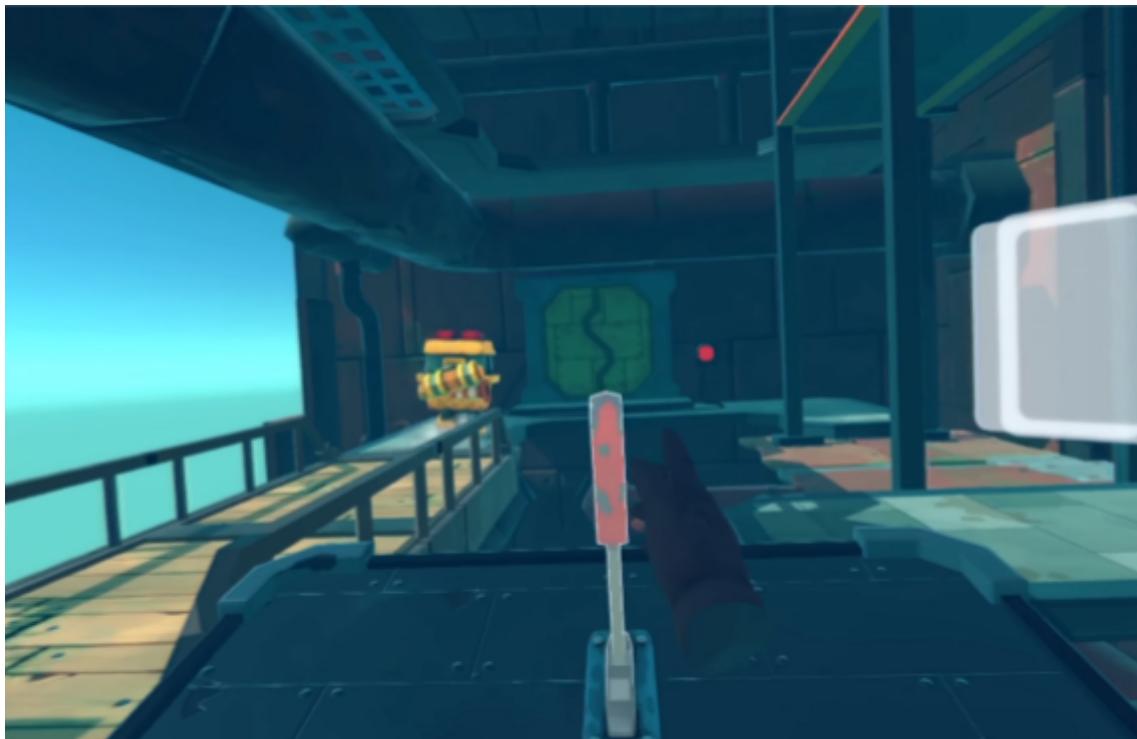


Figura 13: Captura de Wayward sky (interacción en primera persona)

Los desafíos o puzzles que presenta son interesantes a tener en cuenta, como el de tirar de una palanca para mover una plataforma, por ejemplo. Podría ser interesante tener mecánicas similares en nuestro juego, pero en este juego las mecánicas son extremadamente sencillas, adaptándose a los controles de PlayStation, y el juego se centra en los diferentes puzzles a resolver.

Google Earth VR

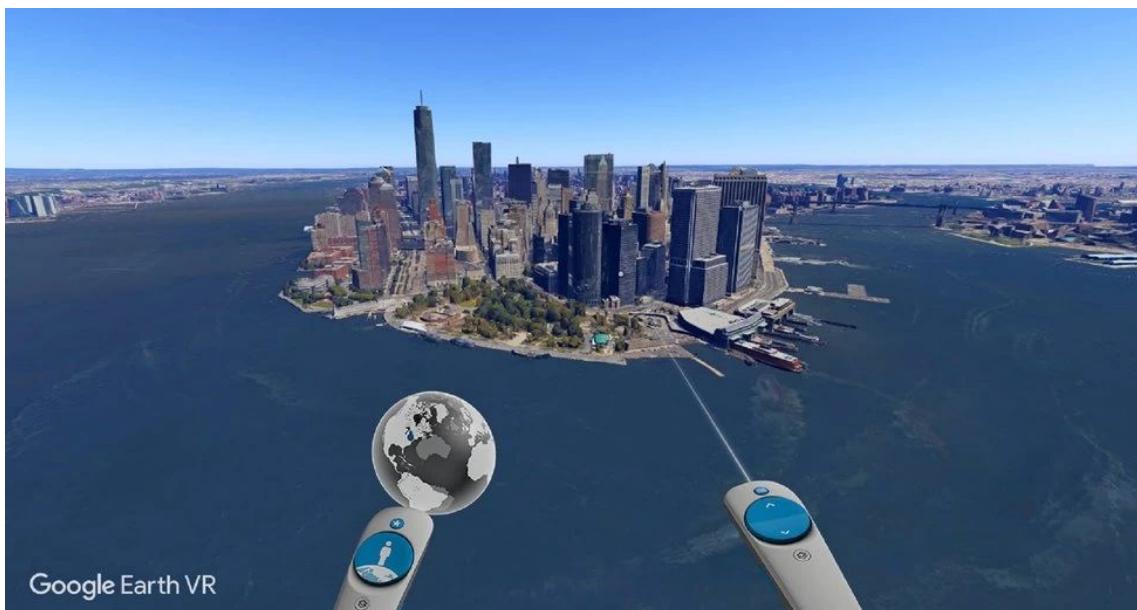


Figura 14: Captura de Google Earth VR

No es un videojuego, pero con esta aplicación podemos viajar por todo el mundo haciendo clic para movernos, ya sea viendo las ciudades desde el espacio, o desde

el suelo como en Google Street View. Con las mecánicas de teletransportación típicas de la realidad virtual, podemos ver que tiene similitudes con un juego *Point & Click* en el hecho de que el personaje va a donde apuntamos (con la diferencia de que nosotros somos el personaje y el mundo es en tres dimensiones).

4. CONCLUSIONES

No hay muchos trabajos similares hasta ahora, puesto que todo lo que se hace en realidad virtual suele ser para interactuar directamente sobre un entorno 3D, ya que no está pensado para un mundo 2D como son originalmente las aventuras *Point & Click*. Por esto tendré que adaptar los desafíos y el modo de juego a un estilo más dinámico, teniendo en cuenta que el jugador no tendrá limitaciones de movimiento ni de ningún tipo a la hora de informarse e interactuar sobre las escenas. El desafío más difícil está en hacer que el juego no sea trivial, puesto que la mayoría de puzzles en un juego del estilo se basa en interactuar con objetos, entonces aquí será prácticamente inmediato con los controles de VR.

En VR al interactuar sobre un personaje no se debería cambiar la pantalla completamente (como es usual en los *Point & Click*) para mostrar el diálogo, por lo que habrá que mostrarlo directamente sobre la escena, o quizás reproducir el sonido y alguna animación simple. Por lo demás debería ser exactamente igual en el tipo de desafíos, excepto que se usará el control en vez de un ratón sobre una pantalla.

III. ANÁLISIS INICIAL

III.I. CÓMO TRASLADAMOS LAS MECÁNICAS DE *POINT & CLICK* A UNA REALIDAD VIRTUAL

El juego va a ser en un entorno de realidad virtual donde podremos interactuar con los mandos y nos podremos desplazar.

El jugador estará situado en una habitación, que representará un escenario de la historia en sí, ya sea un determinado momento o un espacio concreto. Dentro de esta habitación habrá diferentes cuadros o fotografías que representarán escenas, exactamente igual a las escenas 2D que puede tener un *Point & Click* clásico.

En la figura podemos ver la idea inicial para nuestro juego. A medida que el jugador interactúa con las escenas (cuadros o imágenes 2D) podrá ir avanzando por distintas salas.



Figura 15: Representación esquemática del juego

Mecánicas en detalle:

- Moverse entre escenas: Las escenas serán representadas por cuadros o imágenes en las paredes, y los escenarios serán habitaciones o grupos de habitaciones. En un *Point & Click* clásico se suele pasar de escena haciendo clic en el borde de la pantalla, o interactuando con un camino, puerta o flecha. En nuestro juego esa interacción será el equivalente a girar la cabeza y mirar hacia otra escena, (o moverse físicamente hacia ella).
- Agarrar objetos: En el formato original se hace clic sobre un objeto para que el personaje lo guarde en su inventario. Nosotros actualizaremos esta mecánica para que haya que agarrar el objeto con el mando de VR, y arrastrarlo hacia el personaje. En principio no habrá inventario como tal, puesto que todos los objetos estarán en el mundo y se tendrá un acceso muy fácil a ellos, sin necesidad de acumular muchos de ellos.
- Usar objetos: En lugar de mover el cursor al objeto en el inventario y arrastrarlo (o hacer clic en usar), lo cogeremos físicamente con el control, y lo introduciremos en la imagen (el escenario). Cuando se pueda interactuar con el objeto (ya sea con un personaje o con otro objeto) se resaltará y solo habrá que soltar el objeto.
- No será necesario hacer clic para mover al personaje, puesto que seremos nosotros mismos que nos moveremos con los controles de realidad virtual (en un mundo 3D).
- Se podrá interactuar con personas acercando el control (la mano) hacia ellas y pulsando el botón del control, pero no habrá un diálogo como tal, puesto que nuestro personaje no está en la escena (simplemente veremos lo que tienen que decir, que podrá cambiar según interactuemos con el escenario).
- Avance típico en un juego de aventuras: para conseguir avanzar en nuestro juego, tendremos que ir desbloqueando nuevas habitaciones, y eso lo

conseguiremos interactuando con los escenarios y los objetos que haya en él, moviéndolos de un escenario a otro y/o usándolos sobre una persona o sobre otro objeto.

Para conseguir desbloquear la siguiente habitación habrá que abrir una puerta, lo cual se hará accionando una palanca.

- Explicación de la narrativa: al entrar en un nuevo escenario veremos un texto, el cual representará nuestros pensamientos (el diálogo de nuestro personaje), pero será (en principio) independiente de la interacción con las escenas.

III.II. NARRATIVA

El personaje se despertará sin ningún contexto en una sala oscura con un orbe flotante en el centro. Al interactuar con él, verá en las paredes de la habitación una representación del Big Bang, con lo cual podrá avanzar a la siguiente habitación.

En la siguiente habitación verá una escena de unos neandertales reunidos, y otra escena de una tormenta. Si interactúa con los neandertales verá que necesitan calentarse, por lo que la solución para este pequeño puzzle será agarrar un rayo de la tormenta y soltarlo junto a los seres de la otra escena, creando una llamarada y abriendo la puerta.

La siguiente sala también consistirá de dos escenas, una con un paisaje desértico y varios objetos, y otra con un fondo de pirámides egipcias y lo que parece ser un faraón. Este necesita transmitir sus ideas de una forma que no sea hablando, y con la ayuda de los objetos de la otra escena, hay que crear un jeroglífico. Esto es una referencia a la invención de la escritura, en el antiguo Egipto, alrededor del año 3500 ac. Al darle un objeto con un jeroglífico al faraón se completa el puzzle y el personaje puede avanzar.

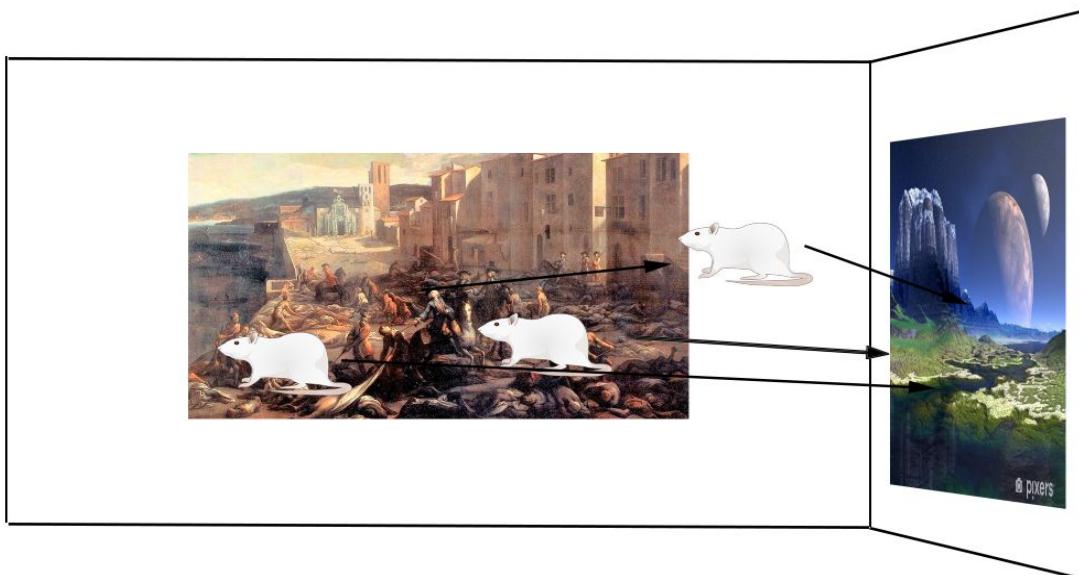


Figura 16: Esquema de la cuarta sala del juego

El cuarto escenario trasladará al jugador a la Edad Media, concretamente en Florencia, durante la epidemia de la Peste Negra. Una escena contendría a varios

enfermos y un médico, que no da más de sí y no sabe lo que hacer, y también se verían algunas ratas. El desafío consistirá en mover las ratas de esa escena a otra escena donde no haya nadie

El último puzzle tendrá varios escenarios (varias salas) y representará la actualidad (el autor del trabajo desarrollando este videojuego). Las salas representarán distintos lugares importantes a la hora de hacer este trabajo (una habitación y la ETSIIT), y los desafíos consistirán en preparar todos los materiales y reunir toda la información.

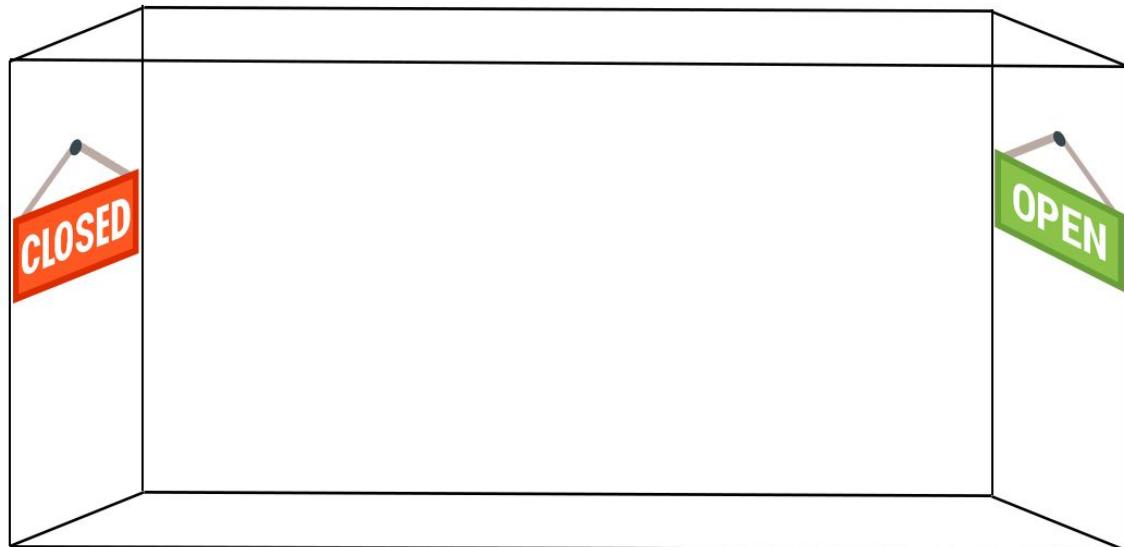


Figura 17: Esquema de entrada al último nivel

El jugador se encontrará con un pasillo, con una puerta abierta y la otra no. La puerta abierta dará a la habitación, en la cual habrá que resolver un puzzle para poder abrir la otra puerta. Tras esto, en la sala de la ETSIIT, el jugador se verá obligado a volver a la habitación (verá todo cerrado a causa de una pandemia).

Finalmente el personaje tendrá que interactuar con el desarrollador del juego de una escena. El jugador tendrá que intentar que nada de esto pase, matando al creador del juego, y se reproducirá una animación donde entramos dentro de la escena, puesto que el jugador está acabando consigo mismo.

IV. TECNOLOGÍA A USAR

IV.I. UNITY:

Unity es una plataforma de desarrollo de software, que usa el motor de juego que lleva su mismo nombre. Versión usada para el proyecto: 2019.3.15f1. [Esta es su página oficial.](#)

Unity se creó con el objetivo de hacer llegar a todos los pequeños desarrolladores una herramienta para poder crear un videojuego complejo sin tener que crear un motor gráfico y todas las mecánicas desde cero. Es capaz de trabajar tanto en 2D como en 3D, y los guiones que afectan a los objetos se programan en C#.

Para un entorno 3D, Unity permite crear formas y diseños 3D, trabajar con texturas, sonidos, elementos gráficos 2D para la interfaz, y más objetos, a los cuales se les pueden aplicar guiones que creamos en C#, así como otros ya predefinidos como el de control de personaje. Esto se suma al sistema de físicas del mundo de Unity, que tiene ciertos valores por defecto, así como una documentación con recomendaciones como la multiplicación por una constante para que el tiempo se aplique de igual manera en todos los dispositivos.

Unity es un motor de juego independiente de la plataforma, permitiendo que el proyecto se exporte para Windows, MacOS, Linux, Android, IOS, WebGL, Playstation o Nintendo, entre otros.

Para usar Unity se necesita crear una cuenta y adquirir una licencia. Unity proporciona licencias gratuitas de uso personal, pero su licencia profesional sí es de pago. Unity tiene un servicio para estudiantes de licencias profesionales durante un año, que es la que se está usando para este proyecto, pero para lo que se está realizando, es exactamente lo mismo.

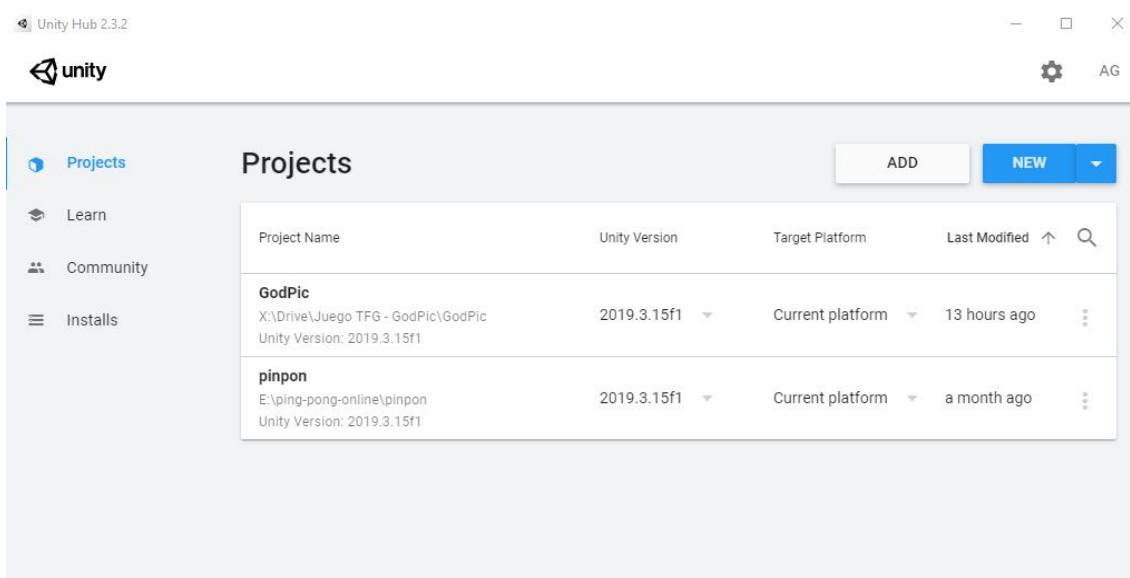


Figura 18: Captura de Unity Hub

A finales de 2018, Unity lanzó Unity Hub (figura 18), un servicio que nos servirá para organizar nuestros proyectos, nuestras versiones de Unity y nuestra licencia.

En este proyecto se usará la versión Unity 2019.3.15f1, puesto que es una versión estable, ya se ha trabajado anteriormente con ella y es totalmente compatible con XR Interaction Toolkit.

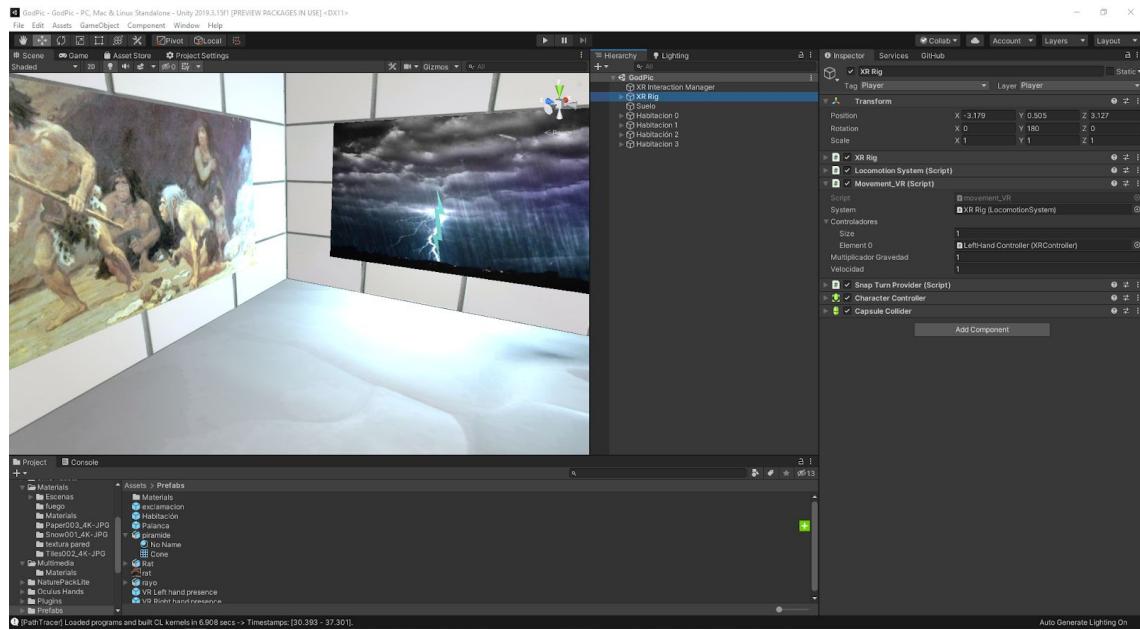


Figura 19: Captura de Unity (tema oscuro)

Una vez instalamos y abrimos Unity, se puede ver una pantalla similar a la de la figura 19, con varias ventanas. Abajo podemos ver la estructura del proyecto, con todos los archivos que contiene, y también la ventana de la consola, que nos mostrará mensajes de información y errores cuando ejecutemos o compilamos los scripts.

Arriba en el centro podemos ver la pantalla principal, que por defecto contendrá la vista de la escena, donde podremos ver el mundo e interactuar con los objetos (ayudándonos de la barra de herramientas de arriba, o con atajos de teclado). A continuación tiene la vista *Game*, para poder probar el juego en el mismo editor, sin necesidad de compilar todo en un ejecutable. Esto se puede hacer con el botón *Play* de arriba del todo.

También encontraremos una vista de la *Asset Store*, que es un navegador donde podremos descargar *assets* para nuestro proyecto (explicación más adelante). La ventana de *Project Settings*, por defecto no se encuentra ahí, pero es la configuración de nuestro juego, donde también encontraremos la configuración del plugin *XR Interaction Toolkit*.

En las ventanas de la derecha se puede ver que hay dos columnas. En la primera nos encontramos la Jerarquía, que contiene la escena y todos sus objetos. Si seleccionamos cualquiera de estos objetos podremos interactuar con él en la vista de la escena (moviéndolo, redimensionándolo, etc.). Los objetos pueden tener

objetos “hijos”, heredando su *transform* (componente que tienen todos los objetos, indicando su posición, rotación y escala) y creando un *transform* relativo al padre.

En el extremo derecho tenemos el inspector, donde aparecerá toda la información de los objetos que seleccionemos, y donde podremos configurar sus componentes, así como añadirlos o eliminarlos. Por ejemplo, un objeto podría ser un cubo, y podría tener como componentes un material, una caja de colisión (*Box Collider*), o un reproductor de sonido.

Si creamos un objeto con unos determinados componentes, y/o varios objetos “hijos”, podremos guardarlo arrastrándolo a una carpeta de nuestro proyecto (ventana de abajo) creando así, un *prefab*. Esto será un objeto plantilla, que podremos usar para replicar más veces, o para pasar como argumento a algún *script* que pida un objeto (tipo *GameObject*).

Se puede ver que la ventana de abajo tiene una carpeta general llamada *Assets* que lo engloba todo, esto es así porque cualquier cosa que podamos usar en nuestro proyecto será un *asset*. Una imagen, un modelo 3D, un sonido, un *script* o incluso una escena son ejemplos de assets que se pueden guardar y compartir a través de la tienda. Por esta razón es muy común usar assets externos en cualquier proyecto de cierta envergadura.

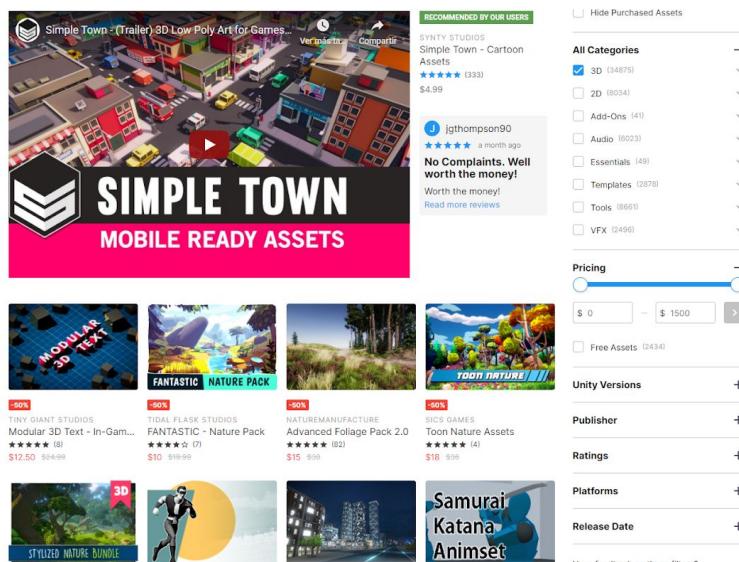


Figura 20: Assets para proyectos 3D más populares en la Asset Store de Unity

Se ha escogido Unity por su amplia comunidad y su larga trayectoria, siendo una de las opciones más fiables y prácticamente un estándar en la industria de los videojuegos.

Otros juegos famosos que usan Unity: Ori and the Blind Forest, Cuphead, Inside, Subnautica, Hollow Knight, Boneworks (juego de RV) o incluso Pokémon GO.

IV.II. XR INTERACTION TOOLKIT:

Plugin de Unity para adaptar los controles y las mecánicas del videojuego a Realidad Virtual. Es oficialmente una versión preliminar puesto que está en la

versión 0.9.4, pero ya ofrece los servicios para crear una experiencia completa en realidad virtual o aumentada.

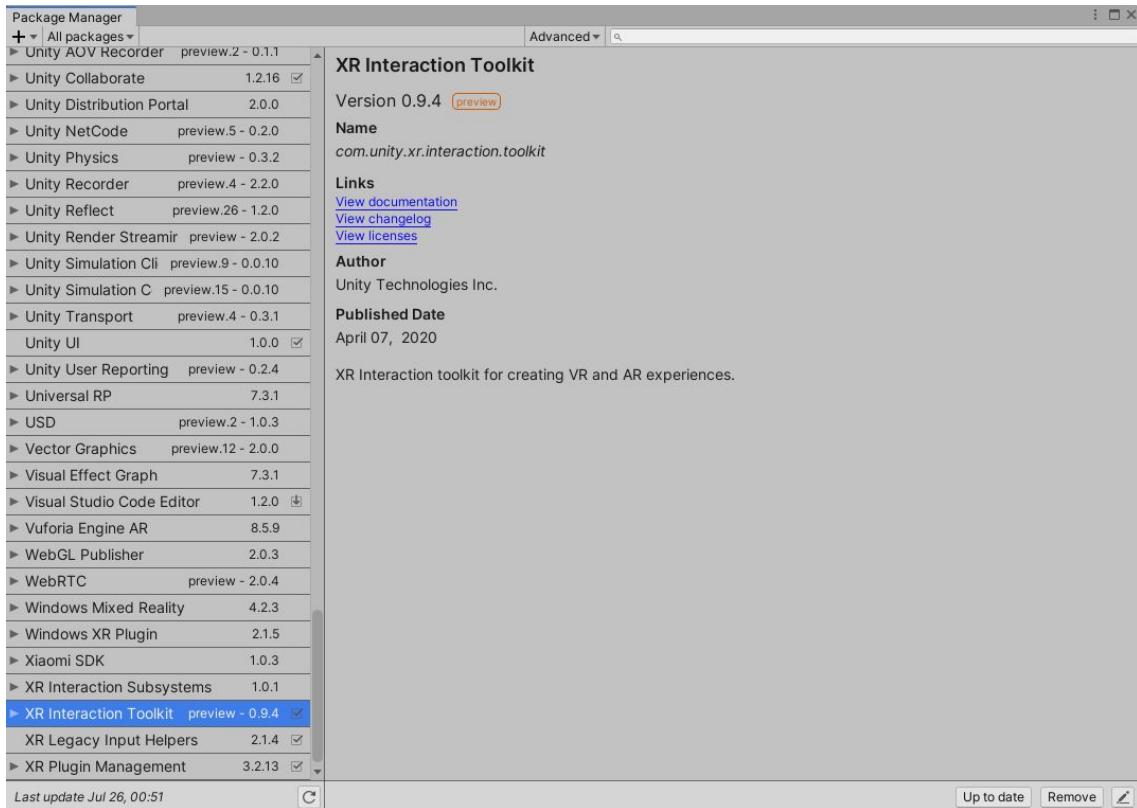


Figura 21: Gestor de paquetes de Unity, XR Interaction Toolkit seleccionado

Este plugin sirve tanto para realidad virtual como para realidad aumentada. Contiene las herramientas básicas para agarrar objetos, seleccionar o interactuar sobre ellos, hacer que interactúen los objetos unos con otros, así como medios para interactuar con interfaces gráficas (botones o menús). También provee los medios para teletransportación o snap-turn (girar 90º lateralmente pulsando un joystick).

Para poder usarlo es necesario *XR Plugin Management*, que permite usar la compatibilidad con los dispositivos. Como en el desarrollo hemos usado Oculus, tenemos que añadir el paquete *Oculus VR* desde los ajustes.

He elegido esta manera de usar realidad virtual porque es la que nos recomienda Unity, siendo compatible con todos los dispositivos posibles de RV y de AR. Quizá no es la que tiene más ejemplos y documentación en Internet, pero tiene un apoyo enorme, incluso aunque no es la versión definitiva. En Unity 2020 otros plugins como *Oculus VR* (nativo) quedan obsoletos.

IV.III. OCULUS RIFT:

Casco de realidad virtual con las siguientes características:

- Dos pantallas OLED con una resolución de 1080x1200 píxeles, cubriendo un campo de visión de 110° a 90 Hz.
- Auriculares 3D desarrollados por Visisonics.

- Sistema de seguimiento de la posición (sensores) con 6 grados de libertad. Inicialmente se usaba solamente un sensor, puesto que no había controles para las manos (solo el casco) o se usaba un mando de XBox, pero desde que se añadieron los controladores Oculus Touch, son necesarios 2 sensores por si unos dispositivos interfieren en la trayectoria hacia el sensor.
- Controladores con 5 botones y Joystick, siendo dos de estos progresivos (agarre y gatillo) y los demás (botones A, B y Menú), activadores de *sí* o *no*. Como he comentado anteriormente, en primer lugar se usaba un mando de XBox, y para algunas acciones especiales un mando básico llamado Oculus Remote, el cual ya fue descartado.

Para poder usar este dispositivo ha sido necesario instalar el Software de Oculus, directamente desde su web. Tras instalarse el programa, hay que colocar los dos sensores por USB, y el casco debe estar conectado en un USB 3.0 y por HDMI (tiene 2 cables). Tras configurarlo correctamente, pasaremos al lobby principal de Oculus, con el menú de la tienda y donde nos saldrán las aplicaciones compatibles que tengamos.

Una vez lo tengamos configurado, solo tendremos que tener instalado en nuestro proyecto de Unity el Oculus Xr Plugin. En el proyecto tendremos que eliminar la cámara principal (si la hay) y crearemos dos objetos de los tipos XR Interaction Manager y XR Rig, respectivamente.



Figura 22: objetos XR Interaction Manager y XR Rig en un proyecto de Unity

El objeto XR Rig representará nuestro jugador, y contendrá la cámara y los controladores. Se ha añadido al XR Rig un script de controlador de personaje, así como un colisionador de cápsula, para que no se puedan atravesar objetos del entorno.

IV.IV. MATERIAL EXTERNO USADO:

A continuación voy a referenciar varios materiales que he usado en el proyecto, ya que no me he dedicado a la parte de diseño, y he tenido que usar material externo.

Un asset es la representación de cualquier objeto que puedes usar en Unity, ya sea un modelo 3D, una imagen, una escena, o un script, entre otros. Una gran cantidad de ellos se pueden encontrar en la Unity Asset Store (algunos gratuitos y otros no), siendo posible descargar partes de proyectos de otra gente.

- Asset de Unity con un conjunto de libros en tres dimensiones: Books Pack - Vertex Studio. Disponible en <https://assetstore.unity.com/packages/3d/props/interior/books-pack-5484>.
- Asset de Unity con partículas y efectos: Cartoon FX Free - Jean Moreno. Efecto de electricidad usado para el rayo, y efecto mágico morado usado en la sala del Big Bang. Disponible en <https://assetstore.unity.com/packages/vfx/particles/cartoon-fx-free-109565>.
- Asset de Unity con fuego procedural de distintos colores: Procedural fire - Hovl Studio. Usado en el fuego de la escena de la prehistoria. Disponible en <https://assetstore.unity.com/packages/vfx/particles/fire-explosions/procedural-fire-141496>.
- Asset de Unity de resaltar objetos: Quick Outline - Chris Nolet. Asset especialmente útil, puesto que nos soluciona el problema de añadir un *shader* a un objeto para resaltar sus bordes (incluso a través de otros objetos) del color que queramos. Se usará con un script que creará un componente del *asset* llamado *Outline* para el objeto padre. Disponible en <https://assetstore.unity.com/packages/tools/particles-effects/quick-outline-115488>.
- Modelos de manos 3D gratuitos de Oculus con tutorial por Valem disponible en <https://www.youtube.com/watch?v=VdT0zMcggTQ>.
- Modelo 3D de rata por Aurelien Gatineau en CGTrader. Disponible en <https://www.cgtrader.com/free-3d-models/animals/mammal/simple-rats-base-poly>.
- Efecto de sonido de fuego - n Beats. Disponible en <https://www.youtube.com/watch?v=mz9ftpHTWTM>.
- Efecto de sonido de trueno - MrSoundtabel. Disponible en <https://www.youtube.com/watch?v=T-BOPr7NXME>.
- Efecto de sonido de puerta - Bit Lab. Disponible en <https://www.youtube.com/watch?v=M8V7ATOjsE>.
- Efectos de sonido de explosión - Creative Film. Disponible en <https://www.youtube.com/watch?v=f0gB369NYFA>.
- Efecto de sonido de frase árabe - Nemoapps. Disponible en <http://nemoapps.com/phrasebooks/arabic>.
- Efecto de sonido de cremallera - Audio Effects. Disponible en https://www.youtube.com/watch?v=bGVZTPHS1_k.
- Imagen PNG de roca - FreePNG. Disponible en <https://www.freepng.es/png-emxtdd/download.html>.

- Imagen PNG de tronco - PNGEgg. Disponible en <https://www.pngegg.com/es/png-blpu>.
- Imagen PNG de caja de diálogo de HiClipart. Disponible en <https://www.hiclipart.com/free-transparent-background-png-clipart-mclug>.
- Imagen PNG de faraón de PNGOcean. Disponible en <https://www.pngocean.com/gratis-png-clipart-znyor>.
- Texturas para distintos materiales - CC0 Textures. Disponibles en <https://cc0textures.com>.
- Tipo de letra Future TimeSplitters - 1001Fonts. Disponible en <https://www.1001fonts.com/future-timesplitters-font.html>.
- Loquendo TTS para hacer los audios explicativos.

V. METODOLOGÍAS A USAR EN EL PROYECTO

V.I. DISEÑO CENTRADO EN EL USUARIO

Este enfoque en el diseño se centra completamente en la gente que va a usar el producto final. Como debería ser en cualquier videojuego, se buscará la mejor experiencia del usuario con el menor esfuerzo posible por su parte. La principal diferencia con otros métodos de trabajo es el hecho de adaptar todas las etapas de producción del producto para que el usuario no tenga que modificar su comportamiento de ninguna manera.



Figura 23: Proceso del Diseño Centrado en el Usuario

El proceso de desarrollo es cíclico, puesto que se van haciendo continuas evaluaciones hasta que se llegan a los resultados esperados. Como se indica en la figura 23, el proceso se compone de 4 etapas principales. Primero se realiza una

planificación a largo plazo, y, a continuación, se pasa a identificar al cliente, y para qué se va a usar el producto y cómo. Tras esto se deben buscar unos objetivos a cumplir y la manera de la que nuestro producto podrá alcanzar esas expectativas.

Es entonces cuando se puede empezar con el desarrollo como tal. Ahora se deben realizar todas las etapas de crear el producto (crear el concepto y llegar hasta la solución definitiva). Finalmente se llega a la etapa de evaluación, en la que se evalúa si se cumplen los requisitos con usuarios potenciales. Esta última etapa es la más importante en este proceso, puesto que determina si hemos llegado al objetivo, o tenemos que realizar otra iteración del bucle.

Al ser el videojuego una experiencia para el usuario, más allá de la utilidad o productividad de la aplicación, lo que hay que valorar son las emociones que provoca en la persona que lo juega. Es por esto que todo el proceso debe estar centrado en el usuario, y usaremos el desarrollo ágil para poder adaptarnos mejor según vayamos avanzando en el proyecto.

V.II. DESARROLLO ÁGIL

Es un tipo de metodología para desarrollar software que se basa en iteraciones, e ir incrementando el valor del producto completo. Surgió para anteponerse a las típicas metodologías en cascada de desarrollo rígido, donde era prácticamente imposible volver varias etapas atrás en el proceso de desarrollo.

Sus fundamentos están en el Manifiesto por el Desarrollo Ágil de Software, cuyos 12 principios son los siguientes:

1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7. El software funcionando es la medida principal de progreso.
8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.

11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

Las metodologías ágiles más importantes son Scrum y Kanban. Ambas permiten tener un feedback, y tener una continua planeación y evaluación, sobre todo para fomentar un buen trabajo en grupo.

- Scrum: El equipo de trabajo se organiza en distintos roles (*Product Owner*, *Scrum Master* y equipo de desarrollo), entre los cuales tendrán total transparencia, un buen nivel de evaluación y capacidad de adaptación.
- Kaban: Divide las tareas por tarjetas, por lo que es sencilla de utilizar, muy visual, y fácil de actualizar para los demás miembro del equipo

A continuación se explica más en detalle la metodología Scrum, ya que es la metodología ágil más usada a nivel mundial.

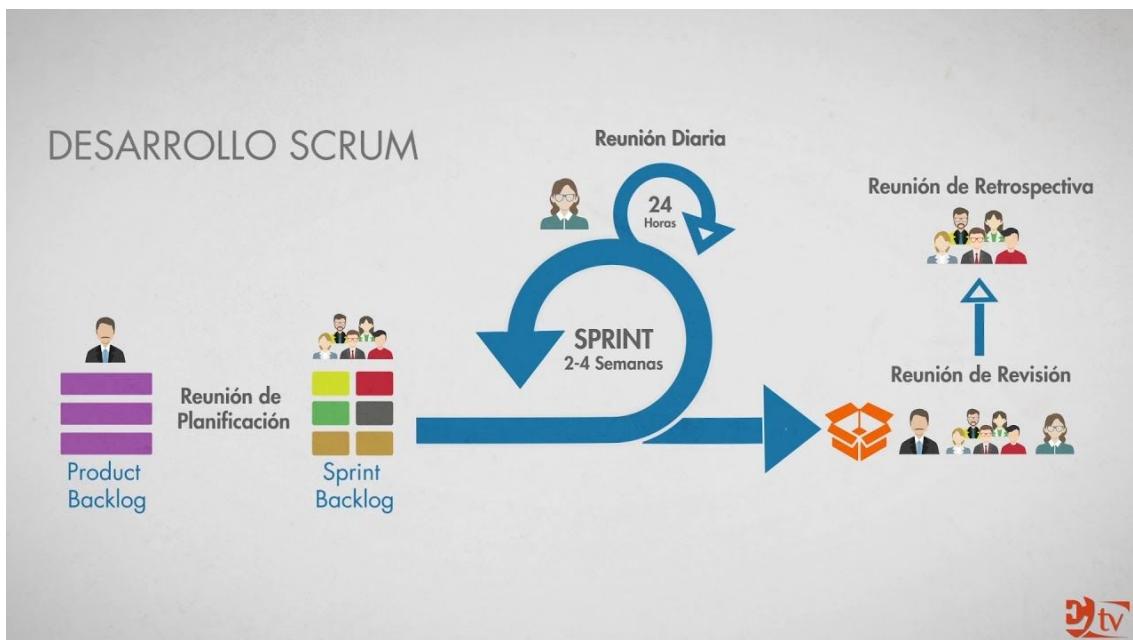


Figura 24: esquema del proceso de SCRUM

La metodología Scrum no es posible sin que el *product owner* organice bien el *product backlog*. El *product backlog* es donde se reúnen todas las historias de usuario de nuestro proyecto a nivel global (tareas poco detalladas y muy detalladas, que van englobando unas a otras). En la reunión de planificación, el *product owner* comunicará bien las historias de usuario al equipo, y se decidirá qué tareas se podrán hacer en el sprint. Así se forma el *sprint backlog*.

Con ayuda del *Scrum master* (una especie de guía y coordinador) y el *sprint backlog*, se puede hacer un sprint que suele ser de 2 semanas, con reuniones diarias, y una de refinamiento sobre la mitad del sprint. En las reuniones diarias se responden estas preguntas lo más rápido posible: “¿Qué hice ayer?”, “¿Qué voy a hacer hoy?”, “¿Tengo algún impedimento que necesito que me solucionen?”.

Finalmente se hace una reunión relajada de revisión, donde se muestran los resultados, pudiendo ser con gente externa al equipo, incluso. Tras esto se realiza la reunión de retrospectiva con objetivo de mejorar u optimizar el siguiente *sprint*. Después simplemente se enlaza con la siguiente reunión de planificación, y se vuelve a iniciar el ciclo.

V.III. PREPRODUCCIÓN, PRODUCCIÓN Y POSTPRODUCCIÓN

Habrá una diferenciación clara entre estas etapas, como si se tratara de una película, puesto que hay crear una historia y una obra audiovisual para el usuario.

En la etapa de preproducción, se creará toda la documentación posible, incluyendo todos los puntos anteriores de esta memoria. Cuando termina esta etapa, el juego debe estar pensado en su mayoría, pero al ser una metodología ágil, la producción tomará mucha más relevancia que si fuera una metodología común en cascada.

La producción será realmente la etapa más importante, puesto que se realizarán todas las iteraciones, y se creará el juego desde 0, añadiendo ideas y conceptos poco a poco pero de una forma organizada.

En la etapa de postproducción tocará evaluar el juego, completar la documentación y realizar los últimos cambios sobre la versión global. Nuevamente nos afecta el tipo de metodología, puesto que las líneas fronterizas entre etapas son mucho más difusas, teniendo varias evaluaciones en todo el proceso de creación del videojuego.

VI. Plan de entregas

Un plan de entregas es un contrato que se puede tener con el cliente, para ir viendo el progreso del desarrollo según se va generando. Esto es extremadamente útil con una metodología ágil, ya que se mantiene un continuo *feedback* con el cliente o usuario. Para una metodología como Scrum, esto beneficiaría enormemente las reuniones de retrospectiva y la planificación, teniendo el *product owner* mucha información asegurada.

En este trabajo se usa un plan de entregas con una metodología ágil (similar a Scrum, pero como no hay un equipo como tal, no se ha seguido una estructura tan estricta). Esto se hace por varias razones muy sencillas, que se suelen repetir en la mayoría de desarrollos de videojuegos: Se tiene que diseñar la historia, pensar en las mecánicas, y, lo más importante, ir aprendiendo sobre las tecnologías a usar, según se va desarrollando el software.

Por estos factores, nadie podría predecir exactamente cómo va a ser el desarrollo del juego, así que se fijan unas entregas orientativas, con objetivos más o menos generales como orientación. Cada entrega sirve

para agrupar varias iteraciones de trabajo, e ir teniendo *milestones* para marcar el desarrollo.

Entrega	Objetivo	Fecha prevista
1	<p>Tener un prototipo del juego con dos habitaciones y dos escenas en cada una, mostrando las mecánicas básicas que se usarán en el juego (mover objetos de una escena a otra, interactuar con objetos y abrir diálogos de personajes).</p> <p>Iteración 1: Crear el software. Tener una demo jugable de una habitación con un desafío a completar para abrir una puerta y superar el nivel. El desafío debe consistir en interactuar con objetos y personajes de escenas que haya en las paredes.</p> <p>Iteración 2: Evaluar la demo con varios usuarios y solucionar errores.</p>	1/08/2020
2	<p>Diseñar toda la narrativa y tener todas las mecánicas implementadas completamente. También debe de estar definida la estética del diseño de juego.</p> <p>Iteración 1: Diseñar la narrativa general de todos los escenarios que contendrá el juego, así como los diálogos que habrá con los personajes. Unir las historias de las escenas con la historia del jugador para que todo tenga un sentido y no sean puzzles independientes.</p> <p>Iteración 2: Homogeneizar el estilo visual del juego y elegir las versiones definitivas de las texturas e iluminación.</p>	15/08/2020
3	<p>Haber creado todos los niveles del juego y que sea jugable al 100%</p> <p>Iteración 1: Implementar todas las mecánicas que el jugador necesitará para completar los desafíos.</p> <p>Iteración 2: Crear los demás escenarios del juego con las mecánicas creadas, a excepción del último.</p>	5/09/2020
4	<p>Completar el último nivel (múltiple) y hacer un menú para entrar o salir del juego.</p> <p>Iteración 1: Desarrollar el último nivel (dos salas y habrá que volver de una a otra) con nuevas mecánicas.</p> <p>Iteración 2: Tener una escena previa al juego con un menú y dos opciones: "Jugar" o "Salir".</p>	15/09/2020

VII. Desarrollo

VII.I ENTREGA 1: PROTOTIPO INICIAL.

✓ Iteración 1 - crear el software

Como se ha explicado anteriormente, la escena de Unity debe contener un XR Rig que será nuestro personaje. Para que se pueda mover con el joystick del mando, se le ha añadido un componente *Character Controller* (ya definido por Unity) y un script que se ha creado, llamado *movement_VR*, que se actualizará a cada frame, comprobando la entrada que le envíemos con el mando, como vemos a continuación.

```
void ComprobarInput()
{
    foreach(XRController controlador in controladores)
    {
        if (controlador.enableInputActions)
            if (controlador.inputDevice.TryGetFeatureValue (CommonUsages.primary2DAxis, out Vector2 posicion))
                Mover(posicion);
    }
}

void Mover(Vector2 posicion)
{
    Vector3 direccion = new Vector3(posicion.x, 0f, posicion.y);
    Vector3 rotacion = new Vector3(0, cabeza.transform.eulerAngles.y, 0);
    direccion = Quaternion.Euler(rotacion) * direccion;

    controladorPersonaje.Move(direccion * velocidad * Time.deltaTime);
}
```

Adicionalmente también se comprueba la dirección de la cabeza, la altura y se aplica gravedad, por si se quisiera implementar alguna mecánica que necesitase movimiento vertical. También se ha añadido la capacidad de girar la cámara 45 grados hacia derecha o izquierda moviendo el joystick derecho. Para esto se ha añadido el componente *Snap Turn Provider (Script)* que nos proporciona el plugin de XR.

El jugador aparece directamente en una habitación cúbica viendo las dos imágenes superpuestas en dos de las cuatro paredes. En una hay una escena con unos neandertales y en otra una tormenta. Sobre la escena de la tormenta hay un objeto de un rayo, con el cual se tendrá que interactuar, por lo que tiene el componente *XR Grab Interactable* (otorgado también por el plugin de XR).

El rayo se ha creado en Blender, modelando en 3D a partir de 3 cubos. Una vez se ha exportado el modelo en formato fbx, se puede introducir el

archivo en un directorio del proyecto de Unity y se usa como un objeto 3D cualquiera.

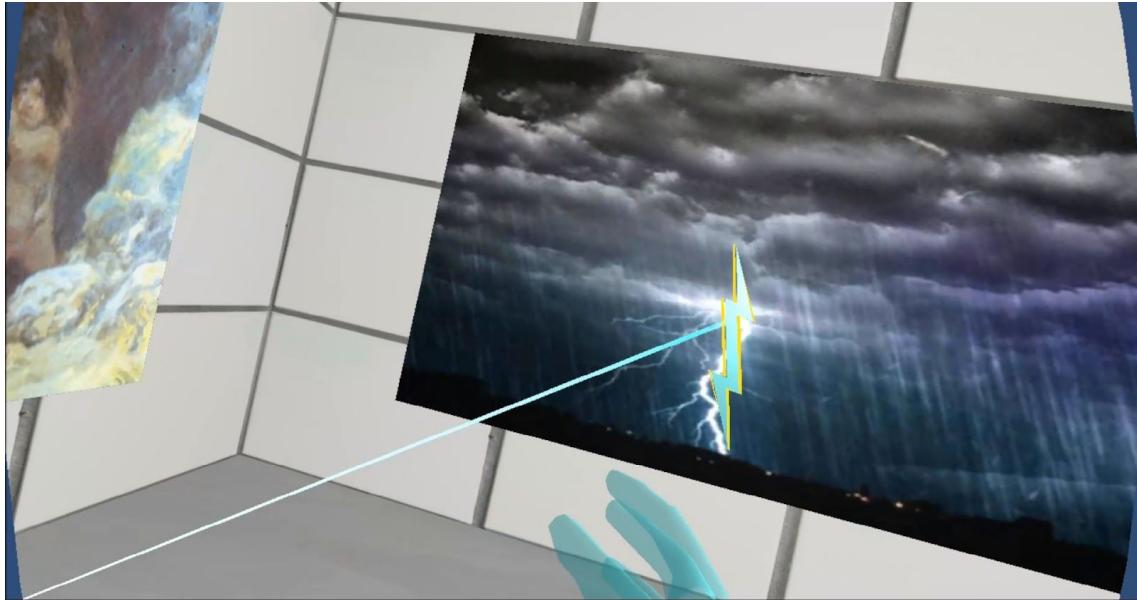


Figura 25: Rayo en escena de tormenta, destacado por apuntarle (nivel de prueba)

Si se interactúa con el neandertal de la escena dirá que necesitan calentarse, por lo que la solución del puzzle será mover el rayo desde la tormenta hacia la otra escena. Al soltar el rayo en el lugar indicado aparecerá un fuego y se abrirá la puerta a la siguiente habitación, la cual está vacía de momento.

Con el plugin de XR es automática la interacción de agarrar un objeto, si este contiene el componente mencionado anteriormente, así que no hay mayor problema. El colisionador del objeto rayo (y de todos los objetos agarrables en el juego) se ha puesto en una capa que hemos creado, llamada *XR Interactable*.

		Layer Collision Matrix						
		Default	TransparentFX	Ignore Raycast	Water	UI	Player	XR Interactable
Default	Default	✓	✓	✓	✓	✓	✓	✓
	TransparentFX	✓	✓	✓	✓	✓	✓	✓
Ignore Raycast	✓	✓	✓	✓	✓	✓		
Water	✓	✓	✓	✓				
UI	✓			✓				
Player		✓						
XR Interactable	✓							

Figura 26: Matriz de colisiones por capas (nivel de prueba)

Luego, en el apartado *Physics* de los ajustes del proyecto, se ha desactivado la interacción entre esta capa y la capa *Player* (como vemos en la figura), que se ha asignado al *XR Rig*.

Para resaltar el objeto, con un borde de color que pondremos cambiar, se ha usado el Asset *Quick Outline* de Chris Nolet, con el cual se ha creado un script llamado *Resaltar* que se añadirá a cualquier objeto que se desee resaltar. El guión contiene la siguiente clase:

```
public class Resaltar : MonoBehaviour
{
    public GameObject activador = null;

    private Outline outline;

    void Awake()
    {
        outline = gameObject.AddComponent<Outline>();
        outline.enabled = false;
        outline.OutlineMode = Outline.Mode.OutlineAll;
        outline.OutlineColor = Color.yellow;
        outline.OutlineWidth = 5f;
    }

    public void ResaltarObjeto(bool siono)
    {
        outline.enabled = siono;
    }

    void OnTriggerEnter(Collider objeto)
    {
        if (activador != null && objeto.gameObject == activador)
        {
            outline.OutlineColor = Color.red;
        }
    }

    void OnTriggerExit(Collider objeto)
    {
        if (activador != null && objeto.gameObject == activador)
        {
            outline.OutlineColor = Color.yellow;
        }
    }
}
```

Con ayuda de la gestión de eventos del componente *XR Grab Interactable*, se puede llamar a la función *ResaltarObjeto* pasándole el parámetro booleano correspondiente (true en caso de estar en contacto con el objeto, y false cuando se deje de estarlo). También contiene un objeto *activador*, que, si existe y el objeto entra en contacto con él, cambiará el color del borde a rojo.

Hay dos interacciones posibles con los objetos: con la mano derecha se agarra directamente por contacto (componente *XR Direct Interactor*) y con la izquierda se agarra a distancia (componente *XR Ray Interactor*). Hay un rayo de luz que cambia de color en la mano izquierda e indica cuándo se puede interactuar, y que se puede personalizar en el componente *Line Renderer*.



Figura 27: diálogo de personaje Neandertal (nivel de prueba)

Los diálogos se muestran directamente al apuntar al personaje en cuestión, por lo que no se puede interactuar y solo sirven como información. Esto se ha conseguido creando un objeto esférico donde está el personaje, desactivando el *Mesh renderer* (para que sea invisible), y añadiendo un componente *XR Simple Interactable*. En el evento de entrar en contacto (*Hover*) se puede activar el objeto del texto en cuestión.

Los modelos de las manos que se han usado son los que nos proporciona Oculus gratuitamente, igual que sus animaciones. Para poder activarlas se ha creado una secuencia de animación con dos parámetros (botón de agarre y gatillo) y se ha añadido un script llamado *HandsPresence* al Prefab de cada mano. La siguiente función es la que detecta los botones del mando y envía el parámetro al animador.

```
void UpdateHandAnimation()
{
    if(targetDevice.TryGetFeatureValue(CommonUsages.trigger, out float triggerValue))
    {
        handAnimator.SetFloat("Trigger", triggerValue);
        if (triggerValue > 0.5f)
            handModelPrefab.GetComponent<MeshRenderer>().enabled = false;
    }
    else handAnimator.SetFloat("Trigger", 0);
}
```

```
if (targetDevice.TryGetFeatureValue(CommonUsages.grip, out float gripValue))
    handAnimator.SetFloat("Grip", gripValue);
else handAnimator.SetFloat("Grip", 0);
}
```

La implementación de las manos y su animación está originalmente realizada por ValemVR en su serie de tutoriales.

✓ Iteración 2 - evaluación

Se ha probado el programa con dos personas, con el objetivo de ver si se cumplían las expectativas. Se ha preguntado específicamente sobre cuatro características del juego: estética, historia, mecánicas y opinión general. Las opiniones han sido constructivas puesto que el programa aún es muy sencillo, con muchas cosas por añadir y que mejorar.

Uno de los usuarios ha sugerido añadir más opciones a la hora de jugar, para que haya una recompensa directa por hacer bien los desafíos, y un castigo o perjuicio por realizarlo de otra manera. Se ha propuesto la idea de añadir más imágenes con objetos perjudiciales que puedan causar el reinicio del nivel.

Con respecto a la estética, se han destacado los fallos en iluminación (porque no estaban correctamente configuradas las sombras de las fuentes de luz) y la simpleza de las imágenes y texturas. La narrativa de esta demo es casi inexistente, pero en todos los casos se ha echado en falta algo más que explicara qué debe hacer el personaje o cuál es el sentido de las imágenes.

Para versiones futuras se intentará explicar mejor por el contexto o será necesaria la ayuda de audios o textos explicativos en cada sala o escena. Las mecánicas, en cambio, se han entendido en todos los casos. Aunque son bastante escasas, al ser realidad virtual se han realizado de manera intuitiva, excepto en un caso, problema que con la ayuda comentada anteriormente estaría solucionado.

El hecho de que los diálogos no sirvan para interactuar con la escena de ninguna manera ha llamado la atención y se ha indicado que podría mejorarse, aunque no estropea la experiencia de juego.

VII.II ENTREGA 2: DISEÑO DE ESTÉTICA Y NARRATIVA.

✓ Iteración 1 - crear la historia

El jugador representará a un tipo de dios que guía a la humanidad durante varios momentos de la historia. Se ha seguido un orden cronológico mostrando acontecimientos importantes hasta llegar al momento actual.

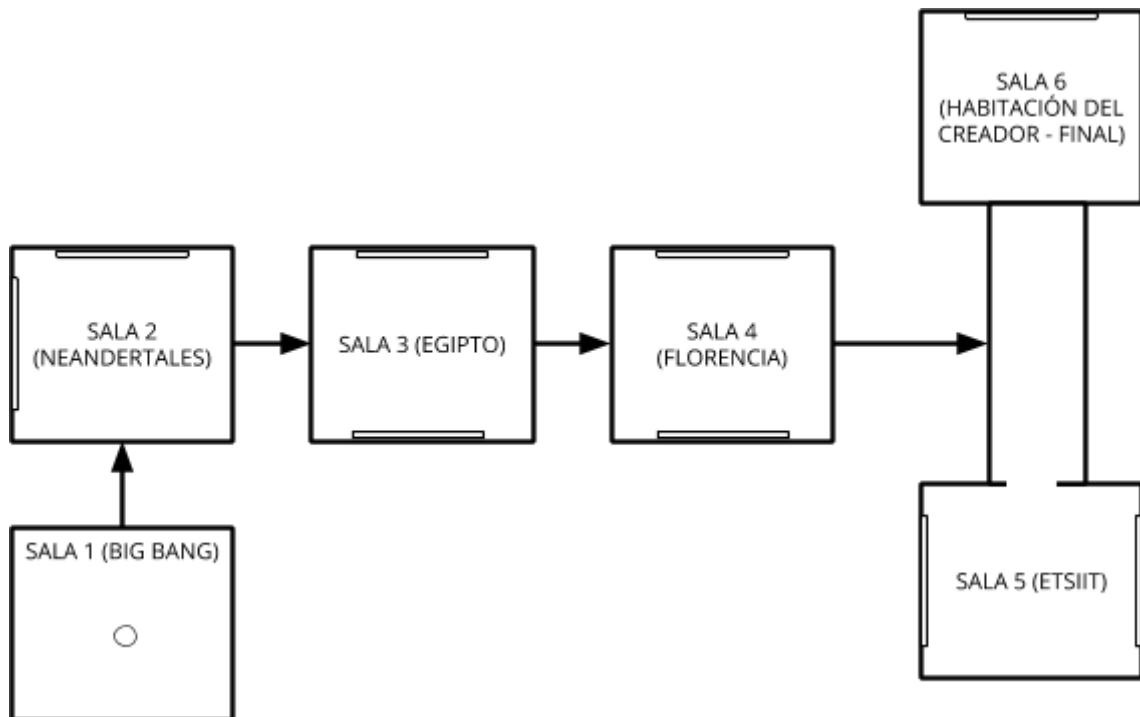


Figura 28: diagrama de habitaciones

El primero de todos será el Big Bang, que será donde aparece el jugador y se introducirá el juego, y es la sala previa a las que creamos en la demo anterior.

La segunda sala ya la hemos creado previamente, será el descubrimiento del fuego por el Homo erectus. Se representará que uno de los neandertales habla en castellano (aunque no tenga sentido) para poder decirle al jugador que necesitan calentarse o protegerse. El hecho de que los diálogos sean en castellano se hará con todos los posibles personajes, que es el recurso normal que se suele usar en este tipo de aventuras.

En la tercera habitación el jugador viaja al antiguo Egipto, donde un faraón querrá transmitir sus ideas y se representará la invención de la escritura a través de jeroglíficos.

En el cuarto escenario, también representado por una sala, estaremos en Florencia en el siglo XIV, y tendremos que lidiar con la Peste Negra. Habrá un médico quejándose de que los pacientes no cesan, y lo que será necesario hacer, será mover las ratas de su escena a otra, la cual representará un planeta alienígena, donde no podrían dañar a nadie.

El quinto escenario consta de dos salas, y representarán cada una un lugar actual, que serán la vivienda del creador de este trabajo, y la Etsiit de la Universidad de Granada, respectivamente. Habrá que reunir objetos que representarán información para este trabajo, y otros que representen la información, para juntarlos e introducirlos en la escena del desarrollo del trabajo.

✓ Iteración 2 - Arreglar la iluminación y modificar texturas y estilos

En primer lugar se ha cambiado el material de las manos por uno de color azul semi transparente.

Se ha elegido una textura de paneles blancos para las paredes, y una de papel para el suelo (texturas sacadas de CC0 Textures, disponibles en <https://cc0textures.com/>). Había un error de iluminación que se ha solucionado haciendo que las fuentes de luz tengan en cuenta las sombras de los objetos con los que se encuentran (*Cast shadows* ha sido activado con *hard shadows*).

VII.III ENTREGA 3: CREAR EL JUEGO

✓ Iteración 1 - Crear las mecánicas definitivas que se necesitarán a lo largo del juego

- Las mecánicas de interactuar con los objetos se han modificado con respecto a la versión de prueba. En este primer nivel, se podía agarrar con ambas manos pulsando el botón *grip*, pero con el control izquierdo era a distancia, y con el derecho, por proximidad (cuando los *colliders* de la mano y el objeto entraban en contacto).

Ahora ambas manos deberán agarrar por proximidad, pero se ha añadido un *XR Ray Interactor* a parte para cada mano, seleccionando en *controller Node*, la mano izquierda y derecha, respectivamente. Este *interactor* a distancia nos servirá para ver los diálogos de los personajes, pero no para agarrar objetos, puesto que estos tendrán la opción *trigger* activada.



Figura 29: puntero de exclamación en diálogo

Como vemos en la figura, también se ha añadido un objeto con forma de exclamación como puntero al apuntar a estos personajes. El objeto es una combinación de una esfera, una cápsula, y un cilindro.

- Se ha creado un objeto palanca que se necesitará mover para abrir las puertas y pasar de sala. En los puzzles, en vez de abrir la puerta al completar el desafío, simplemente se habilitará el movimiento de la palanca. Se detectará la rotación de la palanca mediante un script, que accionará la puerta moviéndola hacia abajo.

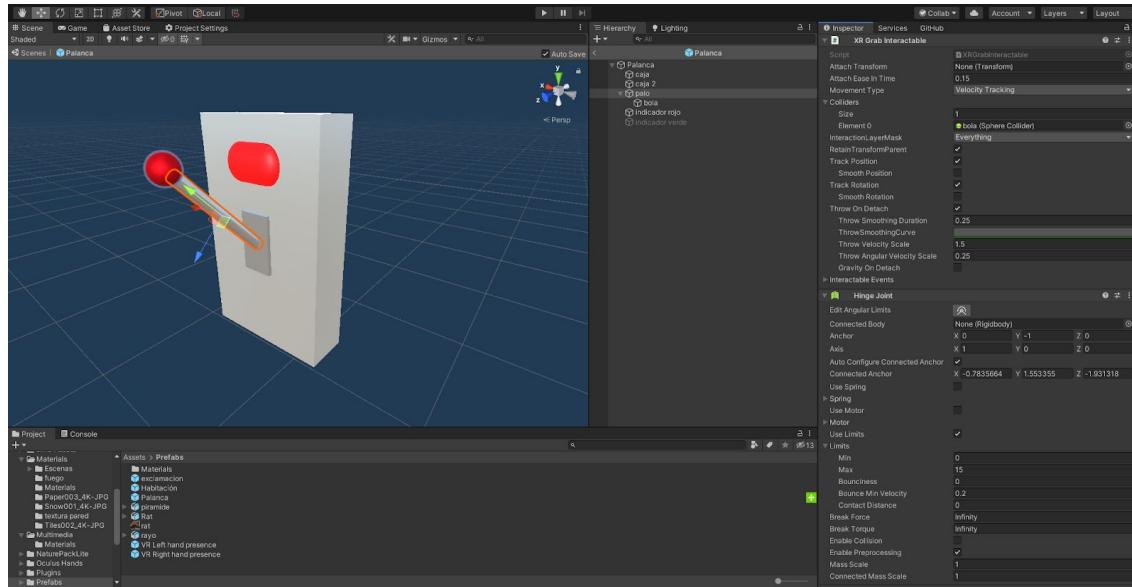


Figura 30: Prefab de palanca

Como se puede ver en la figura, el objeto *palo* contiene los componentes *XR Grab Interactor* y *Hinge Joint*, los cuales nos sirven para la interacción. En el primero, se ha cambiado el tipo de movimiento de *Kinematic* a *Velocity Tracking*, para que no sea un movimiento directo, si no que aplique la fuerza sobre el objeto.

Este cambio, junto al segundo componente, que fija un extremo del cilindro (y su objeto hijo) en el mundo, hace que el movimiento sea realista. Se han definido unos límites de movimiento en este componente, y para que no haya problemas de colisión ni de gravedad, se ha desactivado la gravedad del cilindro y se han eliminado las *box collider* de las cajas.

Por último tiene dos cápsulas con materiales rojo y verde (uno activado y otro desactivado, respectivamente), los cuales se cambiarán uno por otro cuando se libere el movimiento para abrir la puerta.

- Abrir y cerrar una puerta cuando se interactúe o al entrar en una sala. Por norma general, las puertas se abren cuando se realiza cierta interacción y se completa el puzzle del escenario, y se cierran cuando el jugador pasa al siguiente.

Esto se ha realizado creando un objeto vacío con la característica *Trigger* al inicio de la sala, conteniendo un script llamado *DoorController*, con funciones para bajar y subir la puerta. A continuación podemos ver una parte del código:

```
public class DoorController : MonoBehaviour
{
    public GameObject puerta, jugador, palancaAnterior;

    private bool doorIsOpening = false, doorIsClosing = false,
palancaBajada = false;
    private Vector3 posicionOriginal;

    // Update is called once per frame
    void Update()
    {
        if (doorIsOpening)
            puerta.transform.Translate(Vector3.down * Time.deltaTime * 10);

        if (doorIsClosing)
            puerta.transform.Translate(Vector3.up * Time.deltaTime * 10);

        if (puerta.transform.position.y <= -3) //Se termina de abrir
            doorIsOpening = false;

        if (puerta.transform.position.y >= posicionOriginal.y &&
doorIsClosing) //Se termina de cerrar
        {
            puerta.transform.position = posicionOriginal;
            doorIsClosing = false;
            puerta.GetComponent< AudioSource >().enabled = false;
        }

        if (palancaAnterior.transform.eulerAngles.y > 90 && !palancaBajada)
//Detectar palanca y abrir
        {
            AbrirPuerta();
            palancaBajada = true;
        }
    }

    void Start()
    {
        posicionOriginal = puerta.transform.position;
    }

    private void OnTriggerEnter(Collider other)
    {
        if(other.tag == "Player")
        {
            CerrarPuerta();
            //GameObject.Destroy(gameObject);
        }
    }
}
```

```

public void AbrirPuerta()
{
    doorIsOpening = true;
    puerta.GetComponent< AudioSource >().Play();
}

public void CerrarPuerta()
{
    doorIsOpening = false;
    doorIsClosing = true;
    puerta.GetComponent< AudioSource >().Play();
}

```

- Para la sala de la Peste se ha realizado una mecánica especial en el traslado de los objetos. Estos, al entrar en contacto con el plano objetivo, desaparecen y aumentan el contador del *script* (asociado a la imagen objetivo, en este caso la del planeta alienígena). Cuando este contador llega a cuatro, se libera la palanca y se cambia el diálogo como hemos hecho anteriormente.
- Se ha creado un *prefab* de un botón, que consiste en varios objetos sencillos (como se puede ver en la figura 31), donde la cápsula roja tiene un componente *Configurable Joint*, otro *XR Grab Interactable*, y un *script* llamado *Boton*. Se ha definido un límite de movimiento vertical de 0.05 puntos, teniendo un objeto activador justo abajo, para que al bajar se reproduzca el sonido que tiene asociado.

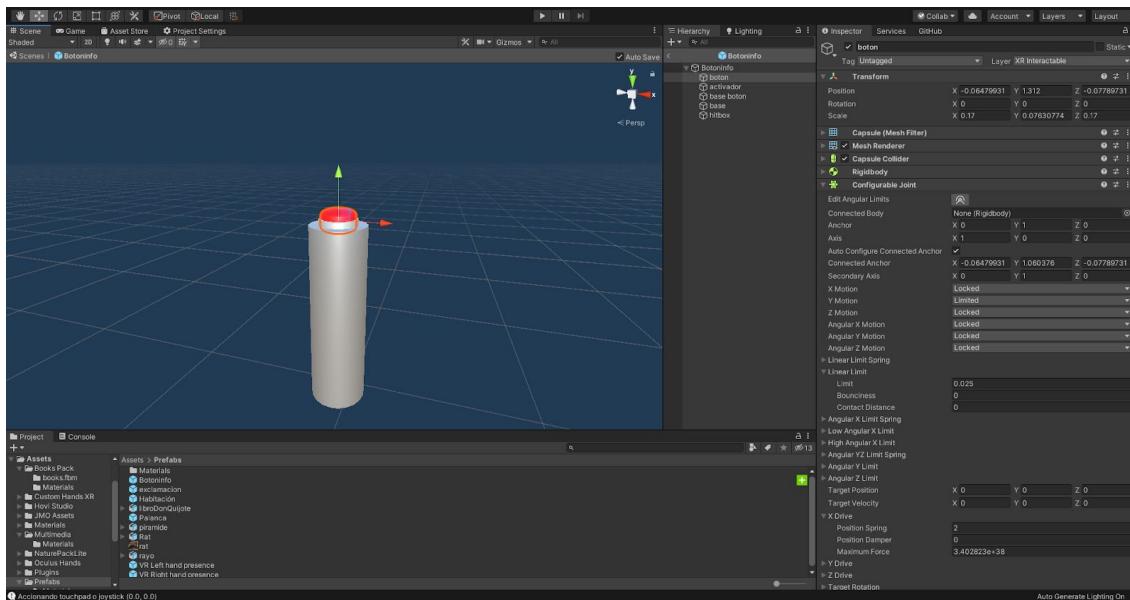


Figura 31: Prefab del botón

Para que el objeto del botón esté levantado, en el *script* se ha definido un impulso inicial hacia arriba, pero como inicialmente está en contacto con el activador, el componente *Audio Source* no está activo desde el principio.

Al llegar a la primera sala hay un objeto invisible que activará los sonidos de todos los botones del juego, estos ya levantados previamente.

✓ Iteración 2 - Crear los otros niveles

- El **primer nivel** se ha creado con una sala y una esfera en el centro. No hay ningún cuadro o escena, simplemente las paredes y el techo de la sala tienen un vídeo de representación del Big Bang que se reproduce al interactuar con la esfera (que tiene un efecto de luces alrededor).

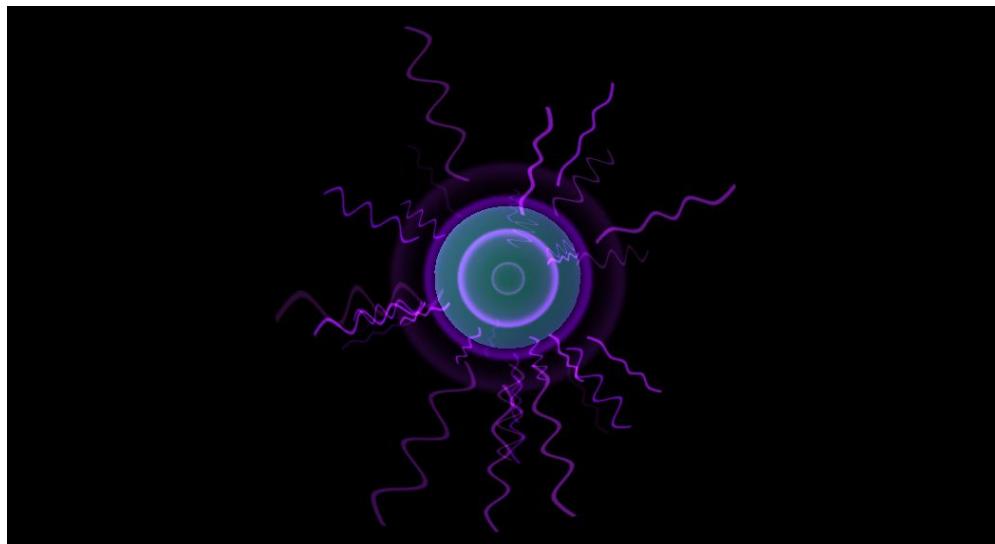


Figura 32: esfera interactiva en el centro de la primera sala

Al interactuar se llama a la función de abrir la puerta de la siguiente sala, por lo que se abrirá reproduciendo el sonido, y, cuando el jugador pase, se cerrará automáticamente.

- El **segundo nivel**, que consiste de dos imágenes y un objeto rayo, ya se ha explicado previamente. Se le ha añadido la palanca para abrir la puerta, por lo que en el script del rayo creando el fuego, se ha activado el movimiento de la palanca activando su luz verde y desactivando la roja. Al hacer esto, no tiene sentido abrir la puerta así que se ha desactivado esta función, que se hará cuando se detecte la rotación de la palanca.
- El **tercer escenario** tendrá una escena con las pirámides de fondo, y en primer plano la imagen de un faraón. Cuando se le apunte abrirá un diálogo de la misma manera que el personaje del neandertal. En la pared contraria hay una escena con dos objetos (un trozo de madera y una piedra).

Estos objetos se ven en 2 dimensiones porque tienen un objeto imagen enlazado a ellos, y la primera vez que se agarran, esta imagen se desactiva y queda el objeto 3D (esto está hecho con los eventos del componente *XR Grab Interactable*).

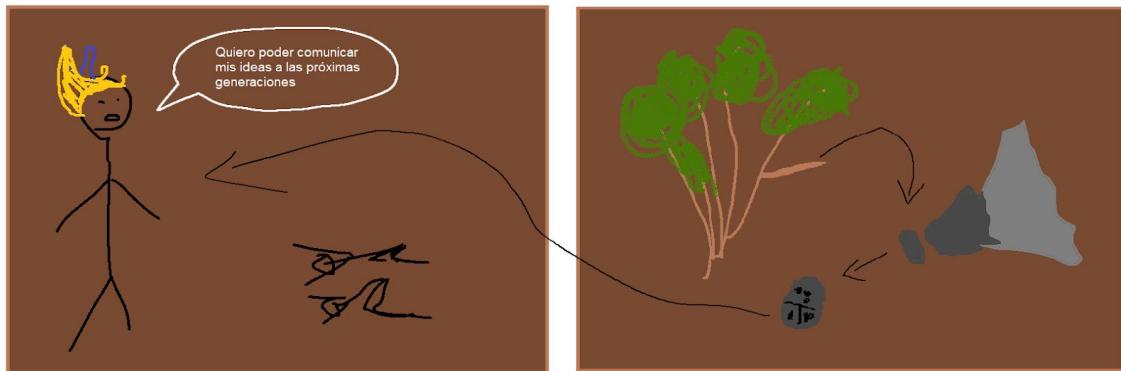


Figura 33: Esquema de tercera sala



Figura 34: Escena de Egipto en Unity

Como vemos en la figura 32, estos objetos (*Roca* y *Madero*) tienen un objeto hijo llamado *guía*, que nos servirá para transformar su posición cuando se agarren con el control de realidad virtual. Simplemente hay que modificar la posición del objeto, y arrastrarlo al apartado *Attach Transform* del *XR Grab Interactable*.

- En la **siguiente sala** hay otro personaje con un diálogo de manera similar, pero en este caso hay 4 objetos con los que podemos interactuar. Estos objetos son cuatro ratas iguales, pero su imagen de previsualización es distinta para cada una.

El objetivo es moverlas al otro cuadro de la pared de enfrente, así que se ha creado un script llamado *Ratas* que se ha añadido al plano que contiene esta imagen. Tiene dos listas de parámetros, así que en una he metido los cuatro objetos “rata” y en otra lista, cuatro objetos “imagen” idénticos a los que tienen cada uno. Estas imágenes estarán ya posicionadas en el segundo cuadro, pero desactivadas, así que en este script se detecta la posición para desactivar el objeto rata y activar la imagen.

Cuando se ha realizado esta acción las cuatro veces, automáticamente se abre la puerta y se cambia el diálogo del personaje (esta mecánica se ha

implementado en las escenas anteriores también). A continuación se muestran los métodos clave del script mencionado:

```
void OnTriggerEnter(Collider objeto)
{
    for (int i = 0; i < 4; i++)
    {
        if (objeto.gameObject == ratas[i])
        {
            ratasBien++;
            ratas[i].SetActive(false);
            imagenesRata[i].SetActive(true);
            Terminar();
        }
    }
}

public void Terminar()
{
    if (ratasBien == ratasTotales)
    {
        dialogoACambiar.text = "Parece ser que los pacientes están mejorando poco a poco";
        doorIsOpening = true;
        door.GetComponent< AudioSource >().enabled = true;
    }
}
```

VII.IV ENTREGA 4: ÚLTIMO NIVEL Y TEST

✓ Iteración 1 - Crear todos los objetos, desafíos y mecánicas del último nivel

Tras pasar la puerta del anterior escenario se pasará a un pasillo, con un extremo abierto y el otro no. A cada extremo habrá una habitación con una o dos escenas, similar a las vistas anteriormente. El **tercer escenario** tendrá una mecánica añadida, y es que será necesario una interacción en un sitio que hayamos visitado anteriormente.

Primero habrá un desafío para abrir la puerta a la otra sala, y en esta sala, podremos obtener el objeto necesario para terminar el juego.



Figura 35: Escena de la habitación

Se ha creado un pasillo, y al final un giro a la derecha con esta vista. Como se ve en la figura 33, hay una escena que ocupa toda una pared, una pantalla de ordenador a la derecha y una palanca a la izquierda. El perro es el personaje informativo de esta escena, diciendo al jugador que debe empezar a trabajar en su tfg. En la pantalla de ordenador (un sprite de un PNG) se puede ver una imagen igual a la figura 34.

Figura 36: captura de la plataforma Swad y objeto del último escenario.

El jugador tendrá que llevar este objeto a la pantalla del ordenador de la escena (de manera similar a la del rayo y los neandertales). Cuando se deja el objeto imagen en la pantalla, se activa un *sprite* desactivado en la escena, que es la misma imagen distorsionada, y se hace desaparecer el objeto original, dando una sensación de que se ha introducido la imagen

en la propia pantalla de la escena. A la vez que esto, se libera el movimiento de la palanca para que se abra la pared detrás del jugador, dejando ver la última escena (la biblioteca).

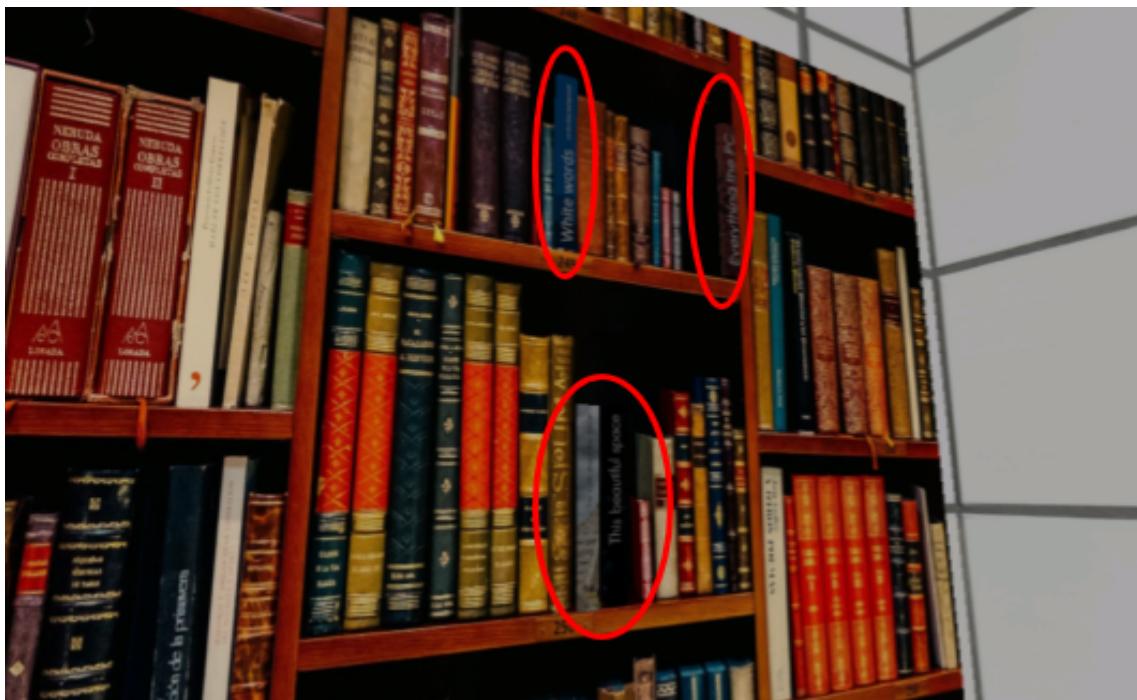


Figura 37: Escena de biblioteca (señalados en rojo los libros para interactuar)

En la figura 35 se han señalado en rojo los libros que son objetos 3D con textura (del asset citado anteriormente, Books Pack, de Vertex Studio) y que el jugador podrá agarrar como cualquier objeto. El personaje del perro indicará que tenemos que reunir “cuatro piezas de conocimiento”, lo que significa que el personaje tendrá que guardarse estos cuatro libros.

Para hacer esto posible se ha implementado una mecánica típica en los videojuegos Point & Click: el inventario. De una forma parecida a la usada con el escenario de la Peste, enlazado al jugador hay un objeto que se activa al agarrar el primer libro que va haciendo desaparecer los objetos “libro” y va aumentando un contador. A continuación vemos el código de la clase que maneja esta mecánica, llamada *mochilaLibros* (asociada al objeto *Mochila*).

```
public class mochilaLibros : MonoBehaviour
{
    public List<GameObject> libros;
    public TextMesh dialogoACambiar;
    public GameObject paredParaQuitar, perro;

    private int librosBien = 0;
    private const int librosTotales = 4;

    void OnTriggerEnter(Collider objeto)
```

```

{
    for (int i = 0; i < 4; i++)
    {
        if (objeto.gameObject == libros[i])
        {
            librosBien++;
            libros[i].SetActive(false);
            GetComponent< AudioSource >().Play();
            Terminar();
        }
    }
}

public void Terminar()
{
    if (librosBien == librosTotales)
    {
        perro.GetComponent< AudioSource >().Play();
        dialogoACambiar.text = "Ya tienes lo necesario...\\nA trabajar.";
        paredParaQuitar.SetActive(false);
    }
}
}

```

Como se puede ver en el *script*, cuando se interactúa con los cuatro libros, se reproduce el sonido del perro como aviso, y se elimina una pared. El jugador no verá ningún cambio a priori, puesto que se ha eliminado la pared de detrás de la escena de la habitación. Haciendo que el último desafío sea entrar a través de la escena, hacia el lugar de trabajo.

Una vez se cruza el cuadro, el jugador se encontrará con una sala completamente negra (un objeto invisible activará una pared negra detrás de nosotros según entremos) y se activará un objeto de texto indicativo de que se ha llegado al final del juego.

✓ Iteración 2 - Crear un menú al inicio juego.

Se ha creado una escena nueva en Unity, con los elementos por defecto de *XR Interaction Toolkit*, y a la cámara se ha asociado un grupo de objetos. El primero, el logo del juego como imagen, y a continuación dos objetos *TextMeshPro*, que con un componente *XR Simple Interactable*, activaban respectivamente las funciones del *script* creado llamado *menuOption*, que contenía lo siguiente:

```
public class menuOption : MonoBehaviour
{
    public void Jugar()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
    }

    public void Salir()
    {
        Application.Quit();
    }
}
```



Figura 38: Captura del menú inicial

VIII. CONCLUSIONES Y TRABAJO FUTURO

1. Este trabajo ha consistido en hacer un juego en tres dimensiones, aplicando las mecánicas y el estilo *Point & Click* a la realidad virtual.
2. El juego se ha presentado en 4 entregas, empezando por hacer un prototipo, a continuación el diseño de la narrativa, seguido de las mecánicas y del resto de escenarios del juego.
3. Como argumento del juego, se ha intentado transmitir que el protagonista es un dios, y ve cómo la historia transcurre desde el principio de los tiempos hasta la actualidad.
4. El desarrollo de este juego, así como su continua evaluación, han permitido observar que la estética del juego, tanto visual como auditiva, es una parte importante en la experiencia del usuario. De cara a un futuro, sería conveniente incidir en dichos aspectos.

IX. BIBLIOGRAFÍA

- Anfossi, G. (1985) La programmation des jeux d'aventure, Editions du PSI, Paris
- Asensio, I. (2019), Unity, ¿qué es y para qué sirve?. Disponible en <https://www.masterd.es/blog/que-es-unity-3d-tutorial/>.
- Bailón Morillas, A. B. (2020), Computación Ubiuba e Inteligencia Ambiental, Apuntes. Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada.
- Bockholt, N. (2017), Realidad virtual, realidad aumentada, realidad mixta. y ¿qué significa "inmersión" realmente? Disponible en <https://www.thinkwithgoogle.com/intl/es-es/canales-de-publicidad/tecnologia-emergente/realidad-virtual-aumentada-mixta-que-significa-inmersion-realmente/>.
- Cinematrix (2020), Tutorial de Blender para modelado 3D. Disponible en <https://www.youtube.com/watch?v=-P398I78NJg>.
- Conell, A. (2019), Tutoriales de XR Interaction Toolkit (VR en Unity). Disponible en https://www.youtube.com/watch?v=6N_0jeg6k0.
- Facebook Technologies (2020), LLC. Oculus. Disponible en <https://www.oculus.com/setup/>.
- Gómez, S. (2017), Instituto de Marketing Ágil, Qué es el Desarrollo ágil. Disponible en <https://www.institutodemarketingagil.com/single-post/Que-es-el-Desarrollo-agil>.
- Google LLC (2020), Google Earth VR. Disponible en <https://arvr.google.com/earth/>.
- Honduras Digital Challenge, Staff (2020), Metodología Scrum, una herramienta útil para agilizar tus proyectos. Disponible en <https://hondurasdigitalchallenge.com/2020/05/21/metodologia-scrum-una-herramienta-util-para-agilizar-tus-proyectos/>.
- Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas (2016), Manifiesto por el Desarrollo Ágil de Software. Disponible en <https://agilemanifesto.org/>.

- Mitchell, D. (1986) An Adventure in Programming Techniques, Addison-Wesley Publishing Co., Londres
- [Script de seguir movimiento sin emparentar por Alex Fryer](#)
- Unity Technologies (2020), Documentación de XR Interaction Toolkit. Disponible en <https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@0.9/manual/index.html>.
- Unity Technologies (2020), Documentación de *XR Interaction Toolkit*. Disponible en <https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@0.9>.
- Unity Technologies (2020), Unity. Disponible en <https://unity.com/>.
- Yusef Hassan Montero y Sergio Ortega Santamaría (2009), Informe APEI sobre usabilidad, punto 4: Diseño Centrado en el Usuario (DCU).