Producción de un Videojuego en Realidad Virtual con Seguimiento de Movimiento en Visor y Controladores Manuales utilizando Metodología Ágil

Alejandro Blázquez, Carlos Casado y Juan Antonio Palacios

FACULTAD DE INFORMÁTICA
DEPARTAMENTO DE INGENIERÍA DEL SOFTWARE
E INTELIGENCIA ARTIFICIAL
UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo Fin de Grado en Ingeniería Informática

Madrid, 16 de junio de 2017

Director: Prof. Dr. Federico Peinado Gil
Codirector: Dr. Nahum Álvarez Ayerza

Autorización de difusión y utilización

Los alumnos Alejandro Blázquez, Carlos Casado y Juan Antonio Palacios, junto al director de este Trabajo de Fin de Grado (TFG), Prof. Dr. Federico Peinado Gil, y al codirector Dr. Nahum Álvarez Ayerza autorizamos a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores tanto la propia memoria de este trabajo, como el código, los contenidos audiovisuales (incluso si incluyen imágenes de los autores), la documentación y el prototipo desarrollado.

Así mismo autorizamos a la UCM a que este trabajo sea depositado en acceso abierto en el repositorio institucional con el objeto de incrementar la difusión, uso e impacto del TFG en Internet y garantizar su preservación y acceso a largo plazo.

Fdo. Alejandro Blázquez Fdo. Carlos Casado Fdo. Juan Antonio Palacios

Fdo. Prof. Dr. Federico Peinado Gil Fdo. Dr. Nahum Álvarez Ayerza

Índice de contenido

RESUMEN

El presente Trabajo de Fin de Grado aborda la producción de videojuegos en Realidad Virtual desde el punto de vista de las metodologías ágiles en Ingeniería de Software. Analizamos el estado de la técnica en Realidad Virtual comenzando por una breve introducción a su historia, desde sus inicios a mediados del siglo XX hasta la aparición de los modernos cascos de Realidad Virtual en el estado actual de la técnica. Sobre estos últimos detallamos especialmente dos de los dispositivos de alta gama disponibles en el mercado: Oculus Rift y HTC Vive, con el que hemos trabajado. También revisamos entornos de desarrollo y herramientas disponibles, como Unity o Unreal Engine, con el que nos quedamos finalmente para aprovechar su potencia y robusta integración con Vive.

El objetivo general de este proyecto es crear una experiencia en primera persona muy inmersiva mediante el uso del visor de Vive, con seguimiento de movimiento, y de los controladores de mano correspondientes, también capaces de hallar su posición y orientación. Estos controladores avanzados nos permiten experimentar con muy diferentes movimientos e interacciones dentro de la Realidad Virtual, de manera que podemos comprobar cuáles de ellos resultan más naturales y satisfactorios para los usuarios, y cuales son más factibles para implementar. Hemos elegido como hilo conductor de la experiencia el género de terror, concretamente la temática de las enfermedades mentales y las fobias, por considerarlo uno de los más emoción puede llegar a provocar entre los jugadores de Realidad Virtual.

Con el fin de lograr nuestro objetivo de manera eficiente, tomamos una serie de decisiones por las cuales comenzamos a utilizar la metodología ágil conocida como Scrum para modelar el proceso de desarrollo de nuestro proyecto. El resultado de las distintas fases de planificación, especificación, análisis, diseño, implementación y pruebas, así como la creación de un completo diario de desarrollo generada durante el mismo proporciona una sólida base, un videojuego en Realidad Virtual de nuestra autoría y con la mayoría de las características deseadas. Gracias a esa base podemos obtener unas conclusiones finales y valorar satisfactoriamente tanto los resultados obtenidos como la aplicación de la citada metodología en este proyecto. Consideramos que ahora está mucho más cerca la posibilidad de crear un producto comercial innovador que se apoye en las contribuciones de este trabajo.

PALABRAS CLAVE

Desarrollo de Software, Desarrollo de Videojuegos, Informática del Entretenimiento, Interacción Persona-Ordenador, Experiencia de Usuario, Realidad Virtual, Tecnología Háptica, Ingeniería del Software, Metodología Scrum

ABSTRACT

This final degree project looks at video game development from the point of view of Software Engineering. We analyze the state of Virtual Reality, beginning with a brief introduction to its history, from its starting moments in mid twentieth century to the emergence of the moderns HMD (Head Mounted Display). We'll emphasize on this ones, specially on the market giants: HTC Vive and Oculus Rift.

Our main goal is to develop an immersive experience using HTC Vive HMD and its motion controllers. These controllers will allow us to experiment with different movement types inside Virtual Reality, so that we can observe which ones of them are the most intuitives or more complicated in implementation matters. We chose as a conductive thread the survival horror game genre, due to considering it one of the most immersive that we know.

In order to achieve our goal in the most effective way, we carry out an intense study of the state of the art, where we considered all the information we can collect about devices, development tools, or problems and applications of Virtual Reality. We decided to use HTC Vive, given its OpenVR philosophy; and Unreal Engine for developing our videogame.

With all the decisions made, we began the real project using agile methodology for software development. More concretely, we used the one known as SCRUM. The extensive documentation generated during the development process provided us a solid basis for the creation of a complete development journal, which serves to provide final conclusions and decide if the experience with this methodology has been satisfactory and helpful to create a fully functional final product.

KEYWORDS

Software Development, Video Game Development, Entertainment Computing, Human-Computer Interaction, User Experience, Virtual Reality, Haptic Technology, Software Engineering, Scrum Methodology

1. Introducción

Nos encontramos ante el inicio de una nueva época para la Realidad Virtual (RV o VR en sus siglas inglesas). La aparición en el mercado de los últimos dispositivos supone una corriente innovadora que parece no tener límites y que va a tener aplicaciones en todos los campos de interés para nuestro día a día.

Lo primero que debemos hacer en este capítulo introductorio es explicar qué se entiende por "**Realidad Virtual",** término acuñado en 1989 por el director de VPL Research, Jaron Lanier, quien la definió como sigue:

> *"La ilusión de tomar parte en un ambiente sintético simplemente con el mero hecho de observar de forma externa tal ambiente. La realidad virtual se basa en tres dimensiones: estereoscópico, monitorización de cabeza (Head Tracked Display) y de cuerpo/manos, y sonidos. La RV es una experiencia inmersiva y multisensorial."* (Steuer, 1992)

Una vez que nos hemos acercado a la primera definición de RV, conviene conocer los hitos que marcaron un antes y un después en el desarrollo de esta tecnología. Hitos asociados a cuatro elementos muy importantes:

1. **Ivan Sutherland.** Profesor del MIT y pionero en el campo de la computación gráfica, siendo probablemente una de las personas que más contribuyó al desarrollo de este área. En 1965 habló de un sistema (The Ultimate Display) en el cual llegaba a asegurar que se podía "sentir el juego". Más adelante, en 1968, creó el primer casco de Realidad Virtual (Head Mounted Display o HMD)[1], conocido como **La Espada de Damocles** (Sutherland, 1965). Se trataba de un dispositivo fijado al techo mediante un brazo mecánico que sostiene el sistema de visualización (dos pantallas CRT). En las articulaciones de dicho brazo se encontraban potenciómetros encargados de medir los cambios de orientación de la cabeza del usuario.

Figura 1.1: La espada de Damocles, primer casco de Realidad Virtual de la Historia

2. **Simuladores de vuelo.** Gran parte de la tecnología que se utiliza para la RV proviene de los simuladores de vuelo. Muchos de estos avances se deben a Thomas Furness, quien en 1982 desarrolló el sistema **VCASS** (Visually Coupled Airborne Systems Simulator), un simulador de vuelo muy avanzado, en el que el piloto llevaba un visor que aumentaba la visión con gráficos utilizados para orientarse y mostrar la trayectoria óptima de vuelo.

[1] Casco de RV por sus siglas en inglés: Head Mounted Display.

3. **NASA Ames.** En 1984 en este centro de investigación de la NASA, se creó la contemporánea Realidad Virtual gracias al **VIVED** (Virtual Visual Environment Display), desarrollado por el doctor Mike McGreevy y Jim Humphries(Earnshaw, 2014)). Gracias a este proyecto consiguieron evaluar el potencial del sistema monocromático del casco de VR para futuros astronautas.

4. **CAVE y Realidad Aumentada.** En 1992 se presentó el proyecto CAVE (Cave Automatic Virtual Environment), que sustituyó al visor utilizado hasta el momento por un conjunto de imágenes estereoscópicas proyectadas en las paredes de la habitación. La sala utilizada para esta CAVE tenía unas dimensiones de 3x3x3 metros y una sala oscura a su alrededor de 10x8x4. Esta sala oscura es necesaria para que alrededor de la CAVE no pueda haber luz mientras se llevan a cabo proyecciones. Este sistema se puede considerar como un precursor de la Realidad Aumentada (RA) que ha llegado hasta nuestros días, consistente en visualizar elementos virtuales dentro de una realidad física a través de un dispositivo tecnológico.

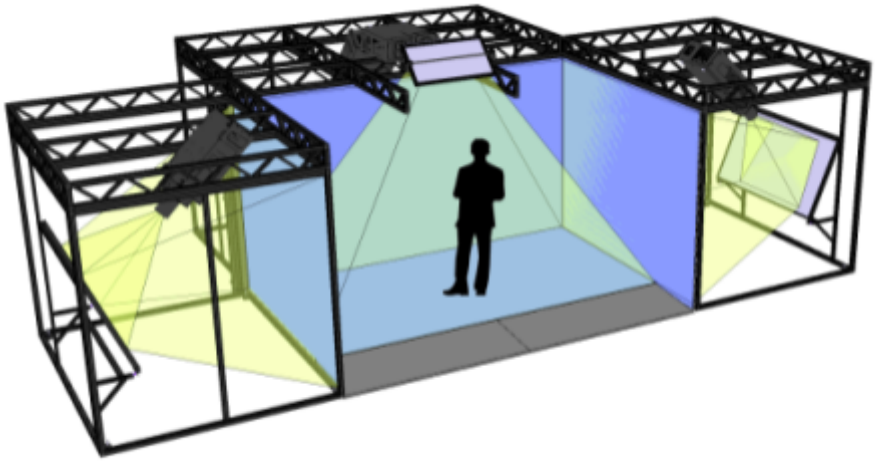

Figura 1.2: Sistema CAVE de Realidad Aumentada.

A día de hoy la RV es una tecnología en pleno auge y desarrollo y, aunque como hemos visto encuentra aplicaciones en multitud de campos, es en la industria del videojuego donde originalmente comienza su expansión y crecimiento.

1.1. La Realidad Virtual en los videojuegos

Basándose en la capacidad de inmersión que proporciona la RV se pueden recrear experiencias en primera persona que permitien al usuario introducirse casi completamente en la atmósfera imaginada por el diseñador de un videojuego. Es por esto que aquellos juegos inmersivos como los pertenecientes al género de terror, entre otros, están empezando a apostar por esta tecnología.

Realizaremos, como parte del presente trabajo, un estudio del estado de la técnica que nos permitirá ahondar en este aspecto más adelante. Por ahora podemos adelantar el hecho de

que grandes sagas, que ya triunfaron antes de este fenómeno, están lanzando nuevos títulos orientados al mercado de RV (p. e. *Resident Evil*[2]), así como de nuevos nombres como *Outlast 2*, el cual cuenta con una gran expectación y una muy buena acogida como ha demostrado su primera demo[3]. Si bien es verdad que este último no ha sido desarrollado íntegramente para RV, sí lo fue la citada demo, que gustó mucho al público. El factor común que comparten estas dos experiencias de RV es que se enfocan mucho en la narrativa para contar una historia de terror, dejando de lado dinámicas de juego[4] espectaculares propias de videojuegos de acción como los tiroteos, las carreras de coches, etc. y centrándose en acciones más cercanas a la realidad que doten a la experiencia de una mayor verosimilitud para el jugador.

1.2. Actualidad de la industria de la Realidad Virtual

Uno de los grandes motores de la industria tecnológica es la capacidad de generar beneficios económicos tanto para las empresas como para los países. Sin ahondar demasiado en este tema por salirse de las competencias del presente trabajo, comentaremos ahora por encima algunas cifras significativas.

Según el primer primer informe sobre el estado de la RV de **The App Date** (The App Date, 2016), ya hay 150 empresas dedicadas a este sector en nuestro país. Además, dicho informe asegura que se han vendido más de 7 millones de visores de RV a finales de 2016 y que se prevé que esta cifra aumente hasta los 100 millones en 2020.

A día de hoy, se estima que en 2017 la venta de cascos de RV tendrá un valor de unos 5.700 millones de euros y que llegará hasta los 11.800 millones de euros en 2018. Con estos datos podemos ver el auge que se espera tenga la RV en la economía global.

En el ámbito nacional, y según el mismo informe, cabe destacar que se estima que el 95% de las empresas españolas que trabajan con este tipo de tecnología está pensando en contratar personal para 2017. De este porcentaje, un 46% prevé contratar entre 1 y 5 profesionales en el próximo año. Hay más de 500 proyectos basados en Realidad Virtual creados en España y ya el 48% de las compañías ha realizado entre 5 y 10 proyectos de éxito de este tipo en el último semestre de 2016.

1.3. Motivación y alcance del proyecto

Con el escenario ya dibujado, nos interesamos por el presente de la RV. La motivación de este trabajo no es otra que afrontar un proyecto englobado dentro de esta tecnología, con el fin de conocer de primera mano las dificultades que se deben afrontar a la hora de desarrollar con éxito un producto de vanguardia. Más concretamente, dicho producto

[2] También conocido como *Biohazard* en Japón, actualmente es una exitosa franquicia de medios desarrollada por la empresa Capcom. Su trama gira en torno a un virus que asola la Humanidad y que convierte a los infectados en seres monstruosos. Se trata por tanto de una trama terrorífica basada en el fenómeno *zombi*.

[3] http://store.steampowered.com/app/414700/Outlast_2/

[4] Entendemos por dinámica del juego al planteamiento y al conjunto de estrategias que puede realizar un jugador para producir una consecuencia dentro del juego, generalmente la consecución de alguno de sus objetivos.

buscará potenciar emociones tan básicas como el miedo y la angustia, tan características de las fobias y fácilmente potenciables en las experiencias inmersivas, por lo que nos acercaremos al género de terror. Para ello realizaremos un estudio del estado actual de la tecnología y de este tipo de juegos en particular, buscando información que nos ayude a tomar las distintas decisiones previas a la realización de dicho proyecto.

Durante las siguientes páginas se presentará el trabajo de investigación realizado previo al diseño del proyecto (Capítulo 2). Se detallarán los objetivos concretos del proyecto, así como la metodología utilizada, Scrum, una de las metodologías ágiles aprendidas durante nuestra formación universitaria (Capítulo 3), y cómo se ha aplicado de manera práctica a la producción de un videojuego en RV (Capítulos 4, planificación, 5, diseño e implementación, y 6, pruebas con usuarios), donde se hace especial énfasis en el seguimiento de movimiento tanto de la cabeza como de las manos del jugador. Finalmente, recogeremos por escrito en esta memoria las conclusiones obtenidas de la experiencia, y daremos una serie de directrices a seguir para trabajos futuros sobre el proyecto (Capítulo 7).

1.4. Vinculación del tema elegido con las competencias del Grado

En este punto vamos a ir detallando qué competencias del Grado en Ingeniería Informática[5] de la Universidad Complutense de Madrid hemos puesto en práctica con este trabajo.

CG19. Conocimiento y aplicación de los principios fundamentales y técnicas básicas de la programación paralela, concurrente, distribuida y de tiempo real.

En nuestro proyecto utilizaremos un entorno de desarrollo llamado Unreal Engine y basado en Blueprints. Hemos podido aplicar los conocimientos adquiridos en las asignaturas de programación (**Fundamentos de la Programación**) a la hora de desarrollar nuestro software de una manera eficiente y sostenible en un futuro.

Además hemos aplicado nociones de **Programación Orientada a Objetos** en la creación de los numerosos actores que tienen cabida en nuestro software. Cada uno de ellos ha sido definido de manera abstracta e implementado según las necesidades de cada nivel, utilizando así ejemplares de objetos con comportamientos similares pero personalizados.

CG6. Conocimiento adecuado del concepto de empresa, marco institucional y jurídico de la empresa. Organización y gestión de empresas.

Gracias a lo aprendido en la asignatura de **Gestión Empresarial**, podemos realizar un estudio de mercado y un presupuesto sencillo sobre lo que costaría desarrollar un software de estas características y ver la viabilidad o no del proyecto con los recursos disponibles.

[5] Detalladas en la web de la UCM: https://www.ucm.es/estudios/grado-ingenieriainformatica-estudios-competencias

CG8. Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.

La metodología de desarrollo software que hemos aplicado a nuestro proyecto, Scrum, aprendida durante el segundo curso en la asignatura de **Ingeniería del Software,** nos permite gestionar y dirigir nuestro proyecto de manera efectiva de principio a fin. Con esto, podemos adecuarnos a los objetivos planteados desde el principio y mejorarlos sin necesidad de poner en peligro la integridad del mismo.

CG12.Conocimiento, diseño y utilización de forma eficiente de los tipos y estructuras de datos más adecuados a la resolución de un problema.

Gracias a los conocimientos adquiridos durante la asignatura de **Estructura de Datos y Algoritmos** podremos implementar una carga eficiente de niveles dentro de nuestro videojuego. Diseñaremos una estructura de basada en diccionarios para que el juego sea modular y, simplemente agregando una clave y el nombre del nivel, podrá añadirse al ciclo del juego de una manera rápida y sencilla.

CG20. Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería de software.

Como ya hemos comentado, utilizaremos una metodología ágil de desarrollo basada en Scrum. Aplicaremos los principios de dicha metodología respetando lo esencial del ciclo de vida de la misma. Todos estos conocimientos también fueron adquiridos durante nuestra formación en Ingeniería de software y en **prácticas en empresas.**

CT1. Capacidad de comunicación oral y escrita, en inglés y español utilizando los medios audiovisuales habituales, y para trabajar en equipos multidisciplinares y en contextos internacionales.

Durante nuestra formación, hemos tenido la oportunidad de trabajar en equipo formando desde grupos pequeños de dos personas hasta grupos de seis o siete miembros. Como nuestro proyecto consta de tres miembros hemos podido aplicar las directrices aprendidas en dichas asignaturas sobre trabajo en equipo.

Por otra parte, hemos tenido la oportunidad de poder realizar presentaciones con diversos medios audiovisuales y con colaboradores (revisores) de otras nacionalidades, al igual que aprendimos en la asignatura de **Cloud y Big Data.**

CT2. Capacidad de análisis y síntesis en la resolución de problemas.

Durante nuestra formación nos han enseñado a pensar de una manera diferente. Nos enseñaron cómo un ingeniero debe abordar un proyecto dependiendo de la naturaleza del mismo. Sin este pensamiento analítico este proyecto nos habría parecido inabarcable y de alguna forma el proyecto también nos sirve para afianzar ese “computational thinking” por el que tanto se aboga ahora.

2. Estado de la técnica

Pasaremos en este punto a detallar el estado de la técnica y del trabajo previo a la realización del proyecto. Cabe destacar que dicho estudio se llevó a cabo al comienzo del proyecto, durante el de noviembre de 2016 y, tratándose la RV de una tecnología en alza, alguna de la información aquí detallada podría haberse quedado desactualizada a la fecha de publicación de esta memoria. Sin embargo podemos asegurar que, a grandes rasgos, lo que sigue a este párrafo es una fotografía bastante precisa del estado de la RV circa 2017.

Comenzaremos con un desglose del hardware capaz de hacernos experimentar RV para, posteriormente, analizar los pros y los contras actualmente encontrados en el uso de esta para crear videojuegos y otras aplicaciones inmersivas. Finalmente, analizaremos los distintos entornos de desarrollo de videojuegos y el software publicado hasta el momento de iniciar nuestro proyecto. Esto nos permitirá conocer mejor el estado de maduración de la tecnología con el fin de tomar las decisiones más acertadas de cara al diseño y desarrollo final.

2.1. Hardware de Realidad Virtual

En esta sección se enumeran y analizan las diferentes propuestas tecnológicas existentes. El análisis incluye tecnologías actualmente en el mercado y futuros dispositivos ya anunciados de los que se conocen suficientes detalles como para ser incluídos en el presente documento.

En la actualidad podemos hacer una división, principalmente, según cuatro categorías diferentes de dispositivos de RV: **visores de 360 grados** o **cascos** (**head-mounted displays**, o **headsets**) con pantallas en su interior; **visores ópticos** (**optical head-mounted displays**, **OHMDs**), que parecen gafas especiales; **visores para teléfonos móviles** que sólo requieren acoplarse a uno estos aparatos para funcionar; y otros **complementos** como controladores manuales con seguimiento de movimiento, las propias cámaras o sensores, o dispositivos especiales de seguimiento para otras partes del cuerpo.

2.1.1. Head-Mounted Display

Son dispositivos sujetos a la cabeza para mayor estabilidad y comodidad, que cuentan habitualmente con una pantalla en su interior situada frente a cada ojo, o con una pantalla mayor frente a ambos ojos.

Hemos realizado un estudio en profundidad de estos dispositivos por tratarse del hardware que utilizaremos en el desarrollo del proyecto, dado que debemos conocer lo máximo posible sobre la herramienta con la que vamos a trabajar.

Parámetros principales de rendimiento y experiencia

Hablaremos ahora de los parámetros fundamentales de los cascos de VR, que son: **capacidad para proyectar imágenes estereoscópicas** (ya sea mediante dos salidas de vídeo, una sola salida secuencial o los métodos *side-by-side* y *top-and-bottom*), **distancia**

interpupilar, campo de visión, orientación, latencia y **tasa de refresco.** Por último explicaremos lo que es el llamado **efecto rejilla.**

Capacidad para proyectar imágenes estereoscópicas

La capacidad del visor y su paneles interiores para producir sensación de profundidad o relieve viene dado mediante los métodos estereoscópicos que generan imágenes diferentes para cada ojo. No se trata de una técnica nueva y llegó a alcanzar cierta popularidad con, por ejemplo, los anaglifos.

Figura 2.1: Ejemplo de anaglifo

Los anaglifos más populares superponen dos imágenes de diferente color y deben ser observadas con unas gafas que cuentan con un filtro para cada ojo del color que queremos excluir en la visualización. Así, en el ojo que tiene el filtro azul, por eliminación, veremos la imagen roja y en el ojo que tiene frente a él el filtro rojo, veremos la imagen azul. Aunque lo ideal es que cada una de las imágenes sea tomada desde diferentes posiciones, puede crearse la ilusión sin necesidad de ello.

Existen otras variantes populares de los anaglifos, como la superposición de imágenes con diferente polaridad, muy utilizado hoy día en el cine y la televisión con capacidad 3D, que permiten la utilización de una gama mayor de colores mediante el uso de gafas polarizadas activas o pasivas.

La polarización activa requiere de unos filtros alimentados con capacidad para bloquear la visión de cada ojo a medida que se proyectan los fotogramas correspondiente al ojo contrario en la pantalla.

La polarización pasiva, más parecida al método de los anaglifos consiste en filtrar las imágenes que se proyectan al mismo tiempo en pantalla con unos filtros sin alimentar, que no realizan más que su función de filtrado.

Hasta ahora, existen principalmente los siguientes métodos para la reproducción de imágenes en cada ojo tanto de forma activa como pasiva:

Dos salidas de vídeo, una para cada ojo

Provee a cada ojo de una señal diferente de vídeo. En la **Figura 2.2** podemos observar un panel AMOLED con dos salidas de vídeo, una a cada lado del ojo, aisladas mediante los visores.

Los dispositivos con estas características suelen tener mayor resolución en su salida de vídeo y ven incrementada su capacidad para emitir imágenes a una mayor tasa de refresco.

Figura 2.2: A la izquierda, visor HMD de PlayStation VR desmontado. A la derecha, el panel AMOLED que se encuentra en su interior.

Salida secuencial

Combina de forma secuencial dos señales de vídeo en una sola señal alternando fotogramas del ojo derecho y el izquierdo durante su reproducción.

Este método preserva la resolución máxima de los fotogramas pero reduce a la mitad la tasa de refresco de los mismos. De esta manera si la frecuencia de la señal original es de 60 Hz se reproducirá a 30 Hz en cada ojo. El menor rendimiento en la tasa de refresco de imágenes se convierte así en un lastre para su uso en cierto tipo de entretenimiento, como videojuegos en RV.

Side-by-side* y *top-and-bottom

Estos últimos métodos, los más utilizados en los sistemas 3D del hogar, consisten en la reproducción de un vídeo dividido por la mitad, con cada una de las mitades para su correspondiente ojo. El reproductor tendrá la capacidad de juntar las imágenes y reescalarlas en una sola salida superpuesta a la vista, en la que se reproducirá de forma secuencial la imagen de cada ojo, que permitirá su visionado, sobre todo, mediante filtros polarizados activos o pasivos.

Los dos formatos más comunes son *side by side* (SBS) y *top and bottom* (TAB), en función de la disposición de los fotogramas correspondientes a cada ojo en la salida original de vídeo. *Side-by-side* coloca las imágenes a izquierda y derecha y *top-and-bottom* arriba y abajo.

La división en dos de del fotograma original y la necesidad de redimensionar dos imágenes en una sola salida reduce la resolución original a la mitad, lo que hay que tener en cuenta como futuro inconveniente.

Campo de visión

El campo de visión[6] (*field of view*, o FOV) es la extensión (vertical y horizontal) del mundo que se puede observar. En el caso del ojo humano cuenta con aproximadamente 114º de campo de visión horizontal (en el caso de la visión binocular), no siendo así en la mayoría de dispositivos de RV, que ofrecen un campo de visión algo menor.

En el campo de los videojuegos adquiere vital importancia, ya que los movimientos rápidos de ciertos tipos de juegos podrían desembocar en problemas e incomodidades para el usuario si lo que visiona en la pantalla, ya sea en un dispositivo de RV como en un monitor externo, tiene un campo de visión insuficiente.

El campo de visión de la mayoría de los videojuegos ha evolucionado al mismo tiempo que los diferentes aspect ratio de los monitores.

Orientación

Para crear mayor sensación de inmersión, cuando el usuario usa un dispositivo de RV verá reflejado su orientación de la cabeza respecto del cuerpo y posición absoluta en el espacio.

La orientación se controla mediante unos sensores incrustados en el visor, que permiten conocer el eje en el que se inclina la cabeza durante su uso, así como la velocidad o fuerza con la que se mueve. Para ello el dispositivo hará uso de un giroscopio, un magnetómetro y un acelerómetro.

El seguimiento de la posición en el espacio lo incorporan los dispositivos más avanzados y permite registrar en tiempo real la posición del usuario en un espacio concreto. Con esto se pretende conseguir un experiencia menos pasiva y que en algunos casos integre el entorno, en cuyo caso los periféricos deberán mapear la habitación.

Latencia

La latencia es el retraso experimentado entre la realización de un movimiento, el momento en que es captado por los dispositivos y periféricos de medición para la orientación, y finalmente el instante en que el usuario percibe el resultado de dicho movimiento en el visor del dispositivo. Una alta latencia empeoraría notablemente la inmersión y aumentaría la incomodidad.

A esta medida de tiempo se le conoce como **latencia de movimiento a fotón** (motion-to-photon latency) y se ve influenciado por la propagación de la latencia en diferentes estadios entre que se realiza el movimiento y se percibe el nuevo fotograma de vuelta. Una baja latencia es necesaria para convencer a la mente de que el usuario se encuentra realmente en un mundo virtual. A esta sensación se le llama presencia y se estima que la latencia para garantizar este estado debe ser menor de 20ms (Abrash, M (2014).

[6] Fuente: Howard, Ian P.; Rogers, Brian J. (1995). *Binocular vision and stereopsis*. New York: Oxford University Press. p.32. ISBN 0-19-508476-4. Retrieved 3 June 2014.

La latencia se ve incrementada por los siguientes estadios:

- Movimiento del casco de realidad virtual.
- Retraso de los sensores.
- USB.
- Procesamiento y fusión.
- Renderizado.
- Transmisión.
- Tiempo que tarda la pantalla en sustituir un pixel.
- Persistencia del pixel (tiempo que tarda la pantalla en apagar el pixel).

Giokaris, P. (2014, January 16)

Tasa de refresco

La tasa de refresco de la pantalla del dispositivo es el la capacidad del dispositivo reflejada en el número de fotogramas por segundo que puede mostrar.

En un videojuego en los sistemas habituales, se considera cómodo para jugar que se aprovechen los 30 Hz de la pantalla, pero en el caso de la RV lo mínimo para que no se experimenten efectos adversos e incomodidad es a partir de los 60 Hz.

Los dispositivos de RV más populares oscilan entre los 90 Hz y los 120 Hz , considerándose una experiencia óptima para el uso en casos de RV.

Giokaris, P. (2014, January 16)

Efecto rejilla

Para terminar esta subsección explicaremos brevemente qué es el efecto rejilla. Se trata de un defecto de visionado común cuando se coloca la vista demasiado cerca de una pantalla en la que se percibe la separación de los píxeles como si fuese una matriz.

El efecto rejilla empeora notablemente al aumentar el campo de visión en una pantalla con baja resolución y densidad de píxeles. Es uno de los problemas más comunes y que más recursos han dedicado a paliar los fabricantes.

Abrash, M (2014)

Principales modelos

En los últimos años se ha experimentado un incremento en el interés del público en la RV, lo que ha dado lugar a una amplia competencia y múltiples opciones tecnológicas y comerciales en el mercado de los visores para RV. En la siguiente sección se desglosan las principales características de los modelos de más repercusión o cuya tecnología se espera se convierta en un estándar en un futuro cercano, así como un breve resumen de la historia de su desarrollo.

Oculus Rift

Oculus fue fundada por el ingeniero Palmer Luckey, el cual gestó su idea como miembro activo de unos foros de discusión en abril de 2009: desarrollar un casco de RV más avanzado y barato que los que por aquel entonces estaban en el mercado.

Luckey, P. (2008, August 21)

Mientras Luckey se encontraba desarrollando un prototipo inicial de su proyecto, el cofundador de *id Software*[7], John D. Carmack, mantenía una investigación paralela con el fin de desarrollar lo mismo.

Finalmente, Carmack daría el pistoletazo de salida a la tecnología presentando un precario prototipo basado en el Oculus Rift de Palmer. Dicho prototipo ejecutaba software propio en la feria E3[8] de junio de 2012, sirviéndose para el cálculo de la orientación de una unidad de medición universal compuesta por un acelerómetro, una pantalla LCD de 5,6 pulgadas y un campo de visión horizontal de 90º y vertical de 110º.

Rift Experiences | Oculus. (n.d.). Retrieved June 09, 2017

DK1

Tras la formación de Oculus como empresa, en 2012 el proyecto se presenta en formato de *crowdfunding* mediante la plataforma *Kickstarter.* El ***Development Kit 1*** **(DK1)** sería enviado a todos aquellos que participaron del proyecto aportando un mínimo de 300 dólares para su desarrollo.

El kit para desarrolladores incluiría una pantalla LCD de 7 pulgadas, sustituyendo la anterior de 5,6 por falta de existencias e incrementando así su volumen y peso. Esta nueva pantalla ayudará a la generación de imágenes estereoscópicas al no producirse superposición, además de a reducir la latencia y el desenfoque en el movimiento al girar la cabeza de forma rápida. Más brillo y una profundidad de color de 24 bits por píxel mejorarán notablemente la experiencia, además de reducir en gran medida el efecto rejilla gracias a la mayor densidad de píxeles.

Página Oficial Oculus DK1. (n.d.)

[7] Compañía estadounidense de desarrollo de videojuegos, fundad en 1991 y responsables del clásico juego de disparos en primera persona *Doom*.

[8] Electronic Entertainment Expo, una de las citas anuales más importantes del entretenimiento electrónico: https://www.e3expo.com/

Figura 2.3: Kit de desarrollo Oculus Rift DK1

El rastreador de movimiento incluido en esta primera versión de desarrollo será el nuevo ***Adjacent Reality Tracker*** a 1000 Hz. Este rastreador incluye una combinación de tres giroscopios, acelerómetros y magnetómetro que hace posible orientar el visor de forma absoluta.

El primer kit de desarrollo salió rumbo a las casas de los mecenas del proyecto en septiembre de 2014, dando así comienzo a la carrera por la RV. Este kit incluía el visor propiamente dicho y una caja de control con conexiones DVI y HDMI.

Patel, N., & Cober, D. (n.d.)

Prototipos intermedios y DK2

En junio de 2013 se presentó un nuevo prototipo llamado ***HD Prototype*** que hacía evolucionar al DK1 hacia la alta definición. Para ello se incluyó un nuevo panel LCD con resolución 1080p que mejoraba notablemente la experiencia reduciendo el efecto rejilla al duplicar el número de píxeles.

Oculus at E3 2013. (2013, June 11)

El siguiente paso de la compañía para mejorar su DK1 pasará por el nuevo ***Crystal Cove Prototype***, que utiliza un nuevo sistema basado en una cámara externa al visor para rastrear la posición del usuario. Dicho sistema se basa en una serie de puntos infrarrojos situados en el visor, los cuales son rastreados por la citada cámara. Esto permite detectar nuevos movimientos como, por ejemplo, el de agacharse o sentarse.

Crystal Cove Debut, CES Recap, and Steam Dev Days. (2014, January 24)

Finalmente, todas estas mejoras desembocaron en un nuevo modelo al que sí tendrían acceso los usuarios: el ***Development Kit 2*** **(DK2).** Este modelo incluye una pantalla que proporciona una resolución final de 960 x 1080 por cada ojo y una mayor tasa de refresco,

además de utilizar el mismo sistema que el Crystal Cove Prototype para el rastreo del usuario.

Página Oficial DK2. (n.d.)

Último prototipo y versión del consumidor

En Septiembre de 2014 Oculus presenta su último prototipo antes de dar el salto al mercado mundial. En este nuevo prototipo, denominado ***Crescent Bay Prototype*** , se realiza una mejora sustancial de la resolución, se mejora su ergonomía reduciendo su peso y se le incorpora audio mediante auriculares integrados en el visor. Se mejora el visionado de la imagen al incluir por primera vez dos pantallas en vez de una y dan el último paso en el rastreo de la posición del usuario, añadiendo un rastreo de 360º mediante el seguimiento de LEDs en el visor. Pulir el último diseño era todo lo que necesitaban para sacar a la luz su versión final, la llamada *Consumer Versión* de Oculus Rift.

Oculus Connect 2014. (2014, September 20)

Figura 2.4: Versión para el consumidor de Oculus Rift

La versión final del Oculus Rift lo compone una pantalla OLED de 2160 x 1200 de resolución total (1080 x 1200) con una tasa de refresco de 90 Hz y un campo de visión de 110º. Cuenta con una capacidad de rastreo de la posición de 360º en un área de 1,52x3,35m mediante acelerómetro, giroscopio, magnetómetro y cámara de rastreo, además de salida de audio mediante auriculares integrados y entrada mediante un micrófono incluido en la carcasa del propio visor.

Las conexiones incluidas en el dispositivo se componen de una conexión HDMI, USB 2.0 y USB 3.0 y utiliza como plataforma software y tienda digital de contenidos la aplicación ***Oculus Home***.

HTC Vive

El HTC Vive se ha convertido, junto con Oculus, en la referencia tecnológica de gama alta para los cascos de RV modernos. Gran parte de su desarrollo ha sido opaco al público y a los medios, entre otros motivos por formar parte de la estrategia, primero en paralelo y después en conjunto, de dos grandes empresas tecnológicas para su incursión en la ola de

la RV: el fabricante de smartphones **HTC** y la desarrolladora de videojuegos **Valve Corporation .**

Vive | Discover Virtual Reality Beyond Imagination. (n.d.).
Valve. (n.d.).

En el año 2012 y mientras Oculus se encontraba reuniendo fondos en Kickstarter, Valve ya dedicaba recursos a investigar la major manera de llevar los cascos de RV a los hogares. Para ello desarrolló un prototipo basado en el uso de una cámara y el rastreo de posición mediante el uso de lo que ellos llamaron ***April Tags*** (Olson, E. (2011)), que no eran más que una primera aproximación basada en los códigos QR. Estos códigos ya habían sido utilizados con anterioridad en circunstancias similares para su uso en RA (Fino, Martín-Gutiérrez, Fernández & Davara; 2013)

El principal inconveniente de este dispositivo, según sus propios desarrolladores, se centraba en la borrosidad de la imagen mostrada por pantalla, la alta latencia de los movimiento y su diseño poco ergonómico propio de una fase tan temprana del desarrollo.

En el año 2013 Valve sigue trabajando en la búsqueda de soluciones y asume su clara competencia con Oculus tras un corto periodo de cooperación. De forma paralela, HTC empieza a interesarse por la RA y la RV, decidiendo apostar finalmente por esta última y descartando también fabricar un visor para móviles como el de Samsung.

La adquisición de Oculus por parte de Facebook en marzo de 2014[9] obliga a Valve a tomar una decisión que culmina en la alianza de Valve con HTC para el desarrollo de HTC Vive a mediados de ese mismo año.

Figura 2.5: Visor HTC Vive (centro) con sus controladores (abajo) y las estaciones de rastreo (arriba).

El primer prototipo de HTC Vive no ha cambiado demasiado desde su concepción. Decidieron enfocar el rastreo de posición de otra manera y lo reinterpretaron mediante el uso de emisores láser con puntos de detección y sistema de rastreo situados en el propio visor. Complementan el diseño unos controladores con pocos botones y una tecnología basada en la precisión y la rapidez del rastreo de la posición.

[9] http://tecnologia.elpais.com/tecnologia/2014/03/26/actualidad/1395796446_034242.html

El primer kit de desarrollo aparece en diciembre de 2014 y sólo podía ser utilizado bajo la supervisión de los empleados de Valve. Habría que esperar al ***Mobile World Congress*** *(*Mobile World Congress. (n.d.)) de 2015 para que tuviera lugar el gran anuncio de HTC Vive. (Página oficila HTC Vive. (n.d.))

La versión final de HTC Vive se presenta por fin en febrero de 2016 con ligeros cambios respecto a la tecnología mostrada en la versión previa: es más pequeña y mueve el centro de gravedad más cerca de la cara, lo que permite, junto con su reducción de peso, un aumento de ergonomía y más estabilidad durante el movimiento.

Se añade una cámara frontal que permite mostrar lo que ocurre frente al usuario en el mundo real mediante la pulsación de un botón. También se incluye un micrófono incorporado al visor.

Las especificaciones finales del visor constan de una pantalla OLED de resolución 2160 x 1200 (1080 x 1200 para cada ojo) con una tasa de refresco de 90 Hz y un campo de visión de 110º.

El sistema de rastreo seguirá siendo mediante tecnología láser, a lo que bautizaron como sistema ***Lighthouse*** (Lighthouse, Manual de usuario de HTC Vive. (n.d.))por la similitud con el funcionamiento de un faro. Dos cajas emiten haces de luz láser y sirven como referencia para el rastreo de la posición de los fotosensores del visor y los controladores, permitiendo así el movimiento libre en un área de hasta 4,57x4,57m. Para ayudar a determinar la posición del jugador, el visor contará también con un acelerómetro, un giroscopio y una cámara frontal.

Las conexiones incluídas en la versión final son HDMI, USB 2.0 y USB 3.0.

PlayStation VR

La gran apuesta de Sony por la RV para su PlayStation 4 pasa por lo que nació con el nombre de ***Project Morpheus*** *(*Project Morpheus (n.d.)) en su etapa de desarrollo y que finalmente pasó a llamarse ***PlayStation VR (***Página oficial de PlayStation VR (n.d.))***.**

Figura 2.6: Modelo final de PlayStation VR

Sony ya demostró mostrarse interesado por la RV en el año 1997, año en que salió a la venta su primer casco de RV[10]. Pero hubo que esperar hasta 2014 para que uno de los desarrolladores de Sony, Anton Mikhailov**,** confirmase que la empresa llevaba tres años desarrollando un prototipo del citado Project Morpheus.

En septiembre de 2015 Sony anuncia el nuevo nombre del dispositivo, PlayStation VR, y, tras la adquisición de ***SoftKinetic***

[10] Conocido como Sony Glasstron. Sus especificaciones pueden consultarse en el siguiente enlace: https://www.cnet.com/products/sony-glasstron-plm-a35-head-mounted-display-0-55/specs/

(Página oficial SoftKinetic (n.d.)), una startup tecnológica dedicada, entre otras materias, al rastreo de movimiento y reconocimiento de gestos, el nuevo casco de VR de Sony salió a la venta en octubre de 2016.

El visor cuenta con una pantalla OLED de 5,7 pulgadas, una resolución de 1920 x 1080 (960 x 1080 en cada ojo), una tasa de refresco máxima de 120 Hz y un campo de visión de 100º. Aunque su tasa de refresco es mayor que la de la competencia, cabe resaltar que la capacidad de procesamiento de la PlayStation 4 haría difícil mover juegos a 120 fps, por ello Sony ha desarrollado una tecnología de interpolación de movimiento que permite crear la sensación de mayor tasa de refresco y menor latencia al emitir fotogramas intermedios mediante la llamada reproyección asíncrona[11].

El modelo final realiza el rastreo de la posición y la captura de movimientos mediante el uso de una cámara doble que rastrea las luces que se encienden en el casco. Incluye además una unidad de procesamiento que no añade potencia de procesamiento a la PS4, sólo sirve para bifurcar la salida de vídeo del juego a la pantalla exterior y el visor de PlayStation VR.

Otros visores

En la actualidad, múltiples empresas se encuentran invirtiendo en RV y en concreto, muchas de ellas se encuentran en desarrollo de cascos que, si bien no tienen la repercusión de los anteriores comentados, suelen ser más accesibles al público por su precio.

Open Source Virtual Reality

Proyecto open-source fundado con la intención de crear un ecosistema libre de RV que permita la posibilidad de crear juegos compatibles con cualquier dispositivo.

La filosofía open-source es aplicada en este proyecto tanto en hardware como en software, por lo que proporcionan un casco de RV totalmente libre para desarrolladores llamado HDK. Por un precio reducido se tiene acceso a un casco con rastreo de de posición con un módulo con giroscopio, acelerómetro y magnetómetro (cámara incluída) y que ofrece un refresco de la posición de 100 Hz.

La pantalla del visor es una OLED con una resolución de 2160 x 1200 (1680 x 1200 cada ojo) con una tasa de refresco de 90 Hz (Página oficial Open Source VR (n.d.).

Star VR

Se trata del primer dispositivo de RV con una resolución de 5k, es decir, 5120 x 1440 (2560 x 1440 por ojo). Para ello cuenta con una pantalla dual de 5.5” en formato panorámico (32:9) y un excepcionalmente alto campo de visión de 210º horizontales y 130º verticales.

[11] Esta tecnología intenta prever cuándo se perderá un fotograma, insertando uno de forma inteligente en el intervalo.

El proyecto ha pasado por diferentes fases e inició su andadura en 2011, con el desarrollo de un visor estereoscópico de resolución 1024x728 y campo de visión de 120º.

Con las versiones han mejorado sobre todo la resolución, el campo de visión y el formato panorámico de la pantalla, así como el rastreo de la posición, que no fue añadido hasta 2015.

En 2016 anuncian su colaboración con empresas como Acer o IMAX y presentan la actualización 1.4 en la feria E3. Actualización en la que se pudo ver una reducción del peso, mejoras en el aspecto y ergonomía y optimizaciones de hardware y software variadas (Página oficial Star VR (n.d.)).

Fove VR

Empresa fundada por Yuka Jokima, una antigua productora de videojuegos de Sony y el desarrollador Lochlainn Wilson que se han propuesto desarrollar un visor de RV multipropósito. Su principal atractivo radica en el rastreo de la mirada del usuario mediante un sistema de seguimiento ocular.

Este registro de la mirada permitirá la creación de experiencias en la que podamos mantener contacto visual con interlocutores, enfocar en profundidad de forma natural y, en definitiva, hacer la experiencia mucho más realista y con mayor sensación de presencia.

Para lograr este ambicioso objetivo, el visor incorporará una pantalla OLED de resolución 2560 x 1440 con una tasa de refresco de 70 Hz y un campo de visión de 100º, además de un sistema de rastreo y orientación óptico para el que será necesario una cámara de infrarrojos interna para cada ojo. Para el rastreo de la mirada se utilizan dos rastreadores por láser con una tasa de refresco de 120 Hz.

Figura 2.7: De izquierda a derecha, OSVR, Star VR y Fove VR

2.1.2. Visores ópticos

Los visores ópticos (optical head-mounted displays) usan un sistema de lentes espejadas que reflejan información e imágenes artificiales y dejan pasar la luz del exterior. Por tanto recibimos imágenes del mundo real, donde se superpondrán las generadas por ordenador (Página oficial Fove VR (n.d.)).

HoloLens

La apuesta de Microsoft por la Realidad Mixta se ve reflejada en el desarrollo de HoloLens, un nuevo visor repleto de sensores y dotado de un sistema computacional con capacidad para mapear el entorno real que rodea al usuario. Posteriormente se incluyen elementos virtuales en la imagen que puede verse de dicho entorno.

Figura 2.8: HoloLens de Microsoft

Para el rastreo se incluye un acelerómetro, giroscopio y un magnetómetro, además de cuatro sensores extra que ayudan a HoloLens a entender el entorno, cuatro micrófonos incorporados y un sensor de luz ambiental.

El visionado se realizará a través de tecnología holográfica mediante la combinación de lentes transparentes en las que se proyectarán imágenes y se ajustarán automáticamente a la distancia pupilar del usuario. Incluye audio integrado.

Las hololens contienen un procesador Intel Atrom de 32 bits con GPU integrada, además de una Unidad de Procesamiento Holográfica hecha a medida y 2 GiB de RAM.

Este visor vendrá con Windows 10 instalado junto a la plataforma Windows Holographic y, aunque todavía no se conoce una fecha exacta de lanzamiento, el evento está previsto para finales de 2017(Página oficial Holo Lens (n.d.)).

Google Glass y otros

Aunque hemos incluido HoloLens en esta misma sección, la mayoría de los visores ópticos tienen un formato más reducido, integrados en una especie de lente en la que se proyectan imágenes holográficas en sólo uno de los ojos. Así, es común que los OHMDs se centren más en mostrar información que en cómo se muestra, abandonando el objetivo estereoscópico y de sensación real de presencia que proponen los cascos de VR, y orientándose principalmente a la RA.

Al tratarse por tanto de una tecnología basada en la proyección holográfica que se sale del propósito del presente trabajo, no incidiremos tanto en ella como sí hemos hecho en los cascos de RV. Aún así es necesario nombrar en este apartado las ***Google Glass*** (Página oficial Google Glass (n.d.)) del archiconocido “gigante de Mountain View”. La propuesta de Google es un visor óptico centrado en la RA y construidas sobre las monturas de lo que podrían ser unas lentes graduadas clásicas. El sistema corre bajo el sistema operativo Android 4.4.

Figura 2.10: Google Glass

La pantalla de visualización consiste en un cristal líquido sobre silicio (LCoS), basado en la utilización de partículas de cristal líquido aplicadas a un dispositivo de silicio de resolución

640 x 360 y una cámara frontal incorporada con capacidad de grabar vídeo. 16 GiB de almacenamiento, 1 GiB RAM (2 GiB en la versión para desarrolladores) y una unidad de medición universal compuesta por tres giroscopios, acelerómetros y magnetómetros. El audio se transmite por conducción ósea (Tech specs, Google Glass Help. (n.d.)).

Además, varias marcas aparecieron a la postre del anuncio o salida de las Google Glass aprovechando la liberación del núcleo de su código. Algunos de estos modelos son accesibles al gran público y otros futuribles anunciados por marcas de los que se desconocen gran parte de sus detalles técnicos.

Empresas como Vuzix (Vuzix (n.d.)) o Sony han desarrollado sus propios productos. Vuzix ya tiene en el mercado un modelo M100 de smart glasses y permite la reserva de su nuevo modelo M300. Se centran por ahora en ofrecer soluciones a profesionales de la industria, sector médico, o cualquier aspecto de los negocios. Al llevar instalado Android ofrecen facilidades para crear nuestras propias aplicaciones.

En el caso de Sony, presentó en 2015 sus Smarteyeglass, un prototipo con una tecnología de visualización parecida a las Google Glass pero anunciaron que pretendían que la imagen proyectada ocupe todo el cristal de las gafas. La versión final incorpora también un controlador de mano comunicado por cable a la montura.

2.1.3. Visores para teléfonos móviles

El auge de la RV ha generado un extenso mercado, con suficiente competencia como para haber avanzado tecnológicamente de manera notable en muy poco tiempo. Aún así, el precio de los visores más avanzados sigue siendo un lastre importante para la penetración en el público general de estos dispositivos, por no hablar de la capacidad de procesamiento, que hace necesario el tener ordenadores de gama alta, con procesadores avanzados y tarjetas gráficas de última generación.

Aunque con la salida al mercado de la nueva gama de tarjetas de tecnología Pascal de Nvidia y Polaris de AMD se ha estrechado ese margen, la respuesta económica a los casos de RV de sobremesa sigue pasando por las tecnologías móviles. En concreto, los teléfonos inteligentes de gama alta integran una capacidad de procesamiento, calidad de pantalla y capacidad de rastrear al usuario suficiente como para crear experiencias inmersivas de RV que acercan la tecnología al público no especializado ni conocedor en profundidad del tema.

Prácticamente todos los fabricantes de *smartphones* han sacado al mercado un modelo compatible con sus dispositivos, así que a continuación detallaremos sólo alguno de los visores (que son prácticamente carcasas con lentes) pioneros, accesibles y de gran éxito comercial en el mercado.

La principal diferencia con un casco de VR convencional es que la pantalla y la capacidad de procesamiento dependen totalmente del dispositivo móvil, que puede acoplarse a la carcasa con lentes y, en algunos casos, también dotada de botones o accesos directos a las aplicaciones que está ejecutando el propio móvil.

Samsung Gear VR

Casco de RV desarrollado por Samsung en colaboración con Oculus, lanzado para consumidores en el año 2015 fruto de la investigación iniciada por la compañía en 2013 (Samsung Gear VR (n.d.)).

La resolución, densidad de píxeles y capacidad de procesamiento necesaria hacen su funcionamiento exclusivo de la gama más alta de sus smartphones, en concreto los pertenecientes a la gama Galaxy y algunos modelos de Note.

Figura 2.11: Samsung Gear VR

El Gear VR incorpora acelerómetro, giroscopio y sensor de proximidad para mejorar la inmersión, además de facilitar el apuntado y el movimiento por la aplicación y el sistema operativo del móvil mediante accesos directos ("inicio" y "atrás"), un touchpad y controles de volumen. La última versión incorpora también mejoras en la ergonomía y una rueda manual que ajusta las lentes a la vista del usuario.

La popularidad de la marca y distintas promociones que lo regalaban junto con la compra de smartphones compatibles lo han convertido en uno de los productos de RV con mayor penetración en el mercado.

Google Cardboard

La más barata de las opciones que tiene el usuario de disfrutar de la RV son las Google Cardboard, aparecidas de la mano de Google en 2014 y basadas en un armazón de cartón recortado y montable en pocos pasos el mismo usuario. Las piezas pueden ser compradas, manufacturadas o fabricadas en casa con materiales baratos y las instrucciones que Google publica en su página web[12].

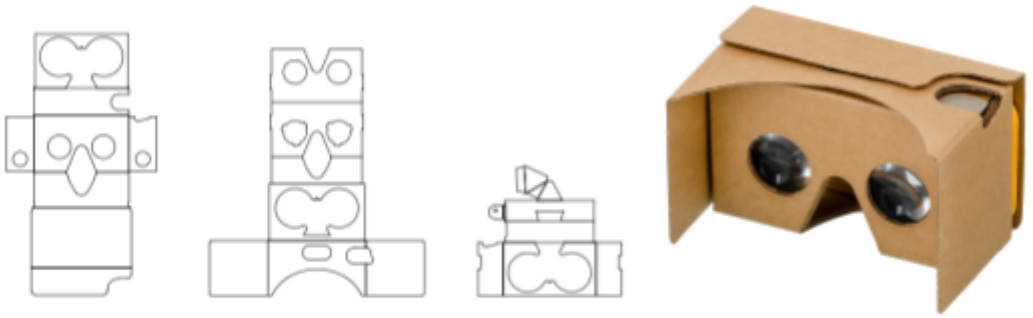

Figura 2.12: A la izquierda, imagen extraída de las instrucciones de Google Cardboard. A la derecha, una Google Cardboard ya montada.

[12] https://static.googleusercontent.com/media/vr.google.com/es//cardboard/downloads/manufacturing-guidelines.pdf

El resultado es un visor de VR que permite utilizar nuestro smartphone como pantalla. La división de la imagen en dos gracias a las aplicaciones del dispositivo, las lentes y la separación entre ojos permiten la creación de una imagen estereoscópica con un amplio campo de visión.

En enero de 2016 Google anunció que hasta ese momento habían distribuído cinco millones de Cardboards (Página Google CardBoard (n.d.)).

Google Daydream View

La respuesta de google a las carcasas de mayor gama no se hizo esperar y en noviembre de 2016 lanzó Google Daydream (Google Daydream View (n.d.)), una plataforma para dispositivos Android 7.1 junto con su primer visor diseñado para teléfonos móviles. Fue diseñada especialmente para los nuevos *smartphones* de Google llamados Pixel y Pixel XL, fabricados por HTC.

Uno de los principales atractivos del visor es la inclusión de un controlador con accesos directos útiles para moverse por la interfaz en forma de botón y *touchpad*. El controlador ofrece tres grados de libertad, en lugar de los seis que, por ejemplo, ofrecen los controladores de HTC Vive.

2.1.4. Complementos

Existen multitud de periféricos capaces de reflejar aún más acciones del usuario dentro de un entorno de RV. HTC abrió el camino con sus controladores incluidos de serie en la venta de HTC Vive y supuso que el resto de desarrolladores tuvieran que trabajar a marchas forzadas en mejorar su tecnología para igualar la experiencia de este casco de RV concreto.

Cada controlador cubre unas necesidades diferentes, aunque puede hacerse un desglose en función de su uso, plataforma, tecnología y experiencia del usuario.

Controladores clásicos: ratón, teclado y mando de juego

Exceptuando el caso de HTC Vive, que desde el inicio apareció en el mercado con sus controladores de mano incluidos y hasta la actual proliferación de controladores manuales específicos para su uso en RV, la mayoría de los cascos de RV basaban sus experiencias en meras presentaciones con poco movimiento e interactividad utilizando para ello el uso del ratón y del teclado o un mando de juego estándar. Las complicaciones de un movimiento rápido al que están acostumbrados los jugadores al usar, por ejemplo, teclado y ratón en juegos de primera persona suponía un serio problema de comodidad durante el visionado de las imágenes.

También se encontraban grandes limitaciones en la inmersión, puesto que los movimientos reflejados en el visor, con los que el usuario debía sentir empatía, terminaban siendo forzados y antinaturales.

En los casos en los que los desarrolladores optaron por que el usuario tomase un papel más activo y tuviera que moverse por un escenario virtual, se optaba por mecanismos de movimiento menos fluidas. El usuario debía rotar su avatar, y por tanto la cámara, en segmentos predefinidos de ángulos fijos, lo que reduce considerablemente el mareo provocado por el llamado *Simulator Sickness* y ayuda a orientarse.

El ratón y el teclado son de uso común todavía hoy con los dispositivos que usan como base de procesamiento el ordenador personal.

Controladores específicos de Realidad Virtual

A continuación se detallarán controladores más avanzados, específicamente creados para su uso con visores de RV.

HTC Vive Motion Controllers

Utilizados en sustitución de las manos en el mundo virtual, sirven para interactuar con con objetos presentados en pantalla. Los mandos incluyen sensores para el rastreo de la señal enviada por las estaciones emisoras y sus controles se basan en un gatillo analógico, botones de menú y sistema y un trackpad en la parte superior (HTC Vive Motion Controllers (n.d.)).

Figura 2.13: Arriba, detalle de los controladores HTC Vive. Abajo, usuaria utilizando los mismos.

Los controladores están fabricados en plástico y cuentan con unas dimensiones aproximadas de 220 x 118mm y un peso de 207 g. Su conexión se realiza mediante tecnología inalámbrica propietaria y la batería útil de litio se aproxima a las 5h pudiendo cargarse mediante conexiones micro USB.
En octubre de 2016 Valve presentó un nuevo prototipo de controladores en los que está trabajando actualmente y que permiten detectar cuándo abrimos y cerramos las manos con el fin de mejorar la sensación al agarrar objetos.

Oculus Touch

En diciembre de 2016 salieron a la venta por separado los nuevos controladores que complementarán Oculus Rift (Información para desarrolladores de Oculus Touch (n.d.)).

Acompañado de un sensor de movimientos pensando en la VR[13] a escala de habitación (room-scale), ofrecen una experiencia más real e inmersiva.

Figura 2.14: Controlador Oculus Touch

El mando incluye un *stick* analógico, dos gatillos también analógicos, dos botones de acción, una palanca analógica y respuesta háptica mediante vibración que reacciona a nuestras acciones. Además, permitirá el reconocimiento de algunos gestos de nuestros dedos. Su rastreo permite 6 grados de libertad en el movimiento (es decir, en todo el espacio tridimensional).

Su alimentación se realiza mediante baterías intercambiables y recargables AA en una carcasa de plástico del que resulta un controlador de 110wx100hx98L mm y un peso de 156 g con batería incluida.

PlayStation Move

Controladores de PlayStation inicialmente lanzados en 2010 para su uso con PlayStation 3, pero que han extendido su compatibilidad a PlayStation VR (PlayStation Move (n.d.)).

Para el rastreo de los movimientos del jugador, los sensores inerciales detectan el movimiento del jugador y PlayStation Camera rastrea su posición.

El controlador principal cuenta con botones de acción típicos de PlayStation, un botón de movimiento, una esfera iluminada para su rastreo y en la parte trasera un gatillo analógico y un botón de “select”.

Su conexión es inalámbrica mediante Bluetooth y se alimenta mediante una batería de ion-litio cargada vía mini USB.

Desde su salida han aparecido algunos periféricos acoplables al PlayStation Move que mejoran la experiencia de juego y permiten mayor inmersión como el “PlayStation Move shooting attachment”, que permite adaptar el movimiento al de un subfusil. PlayStation Move incluye también otro controlador para su uso en la navegación de la interfaz, con controles clásicos de cruceta, gatillo analógico y botones de menú.

Otros controladores

Existen más controladores para otras plataformas normalmente más simplificados, como por ejemplo los controladores para visores móviles como Daydream, que sirven básicamente como herramienta de apuntado y tienen sólo dos botones y un trackpad en la parte superior. No ahondaremos más en el estudio de estos por encontrarse fuera del eje central de nuestro proyecto, que pretende enfocarse a una experiencia lo más inmersiva posible sólo alcanzable mediante el uso de controladores manuales avanzados con seguimiento de movimiento.

[13] RV a escala de habitación: experiencias pensadas dentro de un habitáculo que debe ser mapeado previamente mediante el uso de los controladores

Cámaras y sensores

Terminamos este estudio echando un vistazo al rastreo de movimiento y posición del usuario. Esta es una de las características más a tener en cuenta en la RV, algo básico en la mayoría de visores disponibles, e inseparable de ellos, por lo que a continuación detallaremos algunos de los dispositivos ocupados de este cometido.

Constellation

El sistema de rastreo de Oculus usa una cámara conectada a un puerto USB 3.0 de nuestra máquina cerca de nuestro espacio de uso apuntando a nuestra posición.

Utiliza un sistema óptico de 360 grados de rastreo de LEDs infrarrojos combinado con la unidad de medición inercial del visor para el rastreo y pueden combinarse más de una para conseguir una mayor precisión en conjunto (Constellation, Información para desarrolladores del sistema de sensores de Oculus Rift (n.d.)..

Lighthouse

Este sistema de HTC Vive fue inventado por el miembro de Valve Alan Yates. Basa su rastreo del usuario en la combinación de la unidad de medición inercial y un sistema de láseres infrarrojos de 360º gracias a sus bases repartidas en el habitáculo. En principio para lograr RV a escala de habitación es suficiente con las dos bases incluidas en el pack de HTC Vive (Lighthouse, Manual de usuario de HTC Vive (n.d.)).

Su sistema de rastreo permitió desde el principio mapear la habitación para que el jugador vea una malla indicativa que le indica una referencia al mundo real dentro de su mundo virtual. A este sistema se le conoce como ***Chaperone*** *(* Chaperone. (n.d.))

PlayStation Camera

Junto con la salida de PlayStation VR vino una actualización de PlayStation Camera, en la que se incluye una segunda cámara para percibir profundidad que se coloca frente al jugador .

El sistema, al igual que el de Oculus Rift, parte de la combinación de la unidad de medición inercial y un rastreo óptico de 360º por LEDs situados en los controladores PlayStation Move y el visor de PlayStation VR.

Figura 2.15: De izquierda a derecha, Oculus, Vive Lighthouse y PlayStation Camera.

Otras cámaras

El mercado de los dispositivos móviles sigue avanzando y Samsung ha sacado a la venta una cámara 360º, la llamada Gear360, con conexiones WiFi, Bluetooh y NFC que permiten sincronizarla con el dispositivo Samsung Gear VR y poder ver así lo grabado en forma de experiencia de RV.

2.2. Entornos de desarrollo de videojuegos

Se llama **entorno de desarrollo de videojuegos** a la aplicación informática que tiene como fin ofrecer un conjunto de herramientas, generalmente integradas entre sí (de ahí el llamarlo **Integrated Development Environment**, o **IDE**), que permiten hacer su trabajo al equipo responsable de crear un videojuego hasta el punto de poder generar las versiones finales ejecutables del mismo. También conocidos como **motores** *(engines)* de videojuegos, estos entornos suelen incluir varios módulos destinados a diferentes funcionalidades. Por ello, es común que cada entorno cuente con su propio motor (un subsistema especializado) de renderizado de gráficos 2D o 3D, su propio motor de físicas y colisiones, control de sonido, gestor de animaciones, de inteligencia artificial, de comunicaciones por red, gestión de ficheros y recursos en general y sistema de creación de secuencias cinemáticas.

El uso de entornos de desarrollo avanzados y profesionales para videojuegos, como los que comentaremos a continuación, permite economizar el tiempo de desarrollo y flexibilizar el posterior despliegue en diferentes plataformas.

En los últimos años algunos de estos entornos han sido liberados para su uso de forma gratuita, creando comunidades implicadas en su desarrollo y favoreciendo creaciones independientes gracias a su fácil acceso.

Entre ellos existen características comunes, pero encontramos algunas diferencias que resaltamos a continuación.

2.2.1. Unity

Unity es un entorno de videojuegos multiplataforma desarrollado por la empresa Unity Technologies en 2004. Pensado sobre todo para el desarrollo de videojuegos para PC, Unity está disponible para plataformas Windows, OS X y Linux (Página oficial Unity. (n.d.)).

Su salida inicial fue el fruto del fracaso de la empresa en el desarrollo de un videojuego llamado ***GooBall*** *(*Página oficial GooBall. (n.d.))***,** con el que crearon potentes herramientas que más tarde abrirían al uso de aquellos que lo deseasen. El objetivo: democratizar el desarrollo de videojuegos, permitiendo que tanto grandes como pequeñas empresas pudieran llevarlo a cabo con éxito. Se trata de un entorno amigable para varios perfiles dentro de un equipo de desarrollo, siendo este uno de sus puntos fuertes y motivo por el que alcanzó tanta popularidad.

El motor gráfico de Unity utiliza Direct3D para plataformas Windows, OpenGL para OS X y Linux, OpenGL ES para dispositivos Android e iOS, e interfaces propietarias propias de

algunas consolas como Wii. Unity, además, utiliza PhysX como motor de físicas. **Sus lenguajes de programación principales son UnityScript[14], C# y Boo.**

El entorno incluye una tienda de recursos variados para su uso tanto de pago como gratuitos. Estos recursos han sido creados por la comunidad de desarrolladores, así como por el propio equipo de desarrollo de Unity. Modelados, plantillas, animaciones, paquetes de audio, extensiones del editor, proyectos completos y un largo etcétera que supone una gran ayuda durante el desarrollo.

Unity ofrece a sus usuarios y clientes varios modelos en función de los ingresos y necesidades de la empresa. Unity Personal es totalmente gratuito para aquellas empresas que tengan menos de 100.000$ anuales en ingresos o fondos recaudados. El modelo escala de precio en función de los ingresos y necesidades de la empresa y añade servicios de consultoría y herramientas de monetización (Información sobre precios y condiciones de Unity. (n.d.)).

La gratuidad o no de la edición del entorno de desarrollo que se use no tiene influencia en temas de propiedad intelectual, el desarrollador seguirá siendo dueño de todo lo que cree además de no tener que pagar regalías a Unity Technologies.

2.2.2. Unreal Engine

Unreal Engine es un entorno de desarrollo de videojuegos multiplataforma desarrollado por Epic Games y utilizado por primera vez en 1998 para el videojuego Unreal[15]. Desde entonces este entorno se ha convertido en la base de múltiples juegos y plataformas (Página oficial Unreal Engine. (n.d.)).

En su primera versión integraba renderizado, detección de colisiones, inteligencia artificial, funciones de red y gestión de sistema de ficheros. También contaba con un sistema de rasterizado y acelerador de hardware que usaba la Glide API, especialmente desarrollada para GPUs 3dfx y compatibilidad para OpenGL y Direct3D.

La inclusión del lenguaje de scripting UnrealScript ayudó a popularizar esta herramienta, sobre todo por la facilidad que este lenguaje daba para realizar *mods*.

Su segunda versión vino de la mano del lanzamiento del videojuego America's Army en 2002 y supuso la reprogramación completa del entorno añadiendo un editor de niveles y una actualización del motor de físicas. Durante el tiempo de vida de esta versión del entorno fueron añadiendo compatibilidad con otras plataformas y consolas.

Unreal Engine 3 hizo su primera aparición en 2004 con cambios en el sistema de sombreado, como por ejemplo el sistema de iluminación por píxel, en contraposición al sistema por vertex que utilizaba hasta ese momento.

[14] Lenguaje propietario similar a JavaScript, pero con varias diferencias semánticas: http://wiki.unity3d.com/index.php/UnityScript_versus_JavaScript
[15] http://www.imdb.com/title/tt0282232/

Su soporte se reducía a Windows, Xbox360 y PlayStation 3 en su salida, pero en 2010 se añadió compatibilidad con Android e iOS. También se mejoró el sistema de iluminación, junto con la compatibilidad con luces dinámicas o soporte para DirectX 11.

La vida de Unreal Engine 3 fue muy larga y este estuvo sometido a múltiples actualizaciones que añadieron un sistema de destrucción de objetos, simulación de multitudes, compatibilidad con Ipod Touch, e integración con SteamWorks.

Unreal Engine 4 nacería para cubrir las necesidades de la octava generación de consolas, PCs y dispositivos móviles Android basados en la Nvidia Tegra. Incluye mejoras en la productividad del equipo y permite actualizar el código C++ en el editor de código mientras el propio entorno de desarrollo está en ejecución. Sustituye el antiguo sistema Kismet de UE3 de programación visual por ***Blueprints*** (Figura 2.18) para desarrollo de la lógica del juego de forma gráfica sin necesidad de usar C++, incluyendo la posibilidad de depurar directamente la ejecución sobre los diagramas.

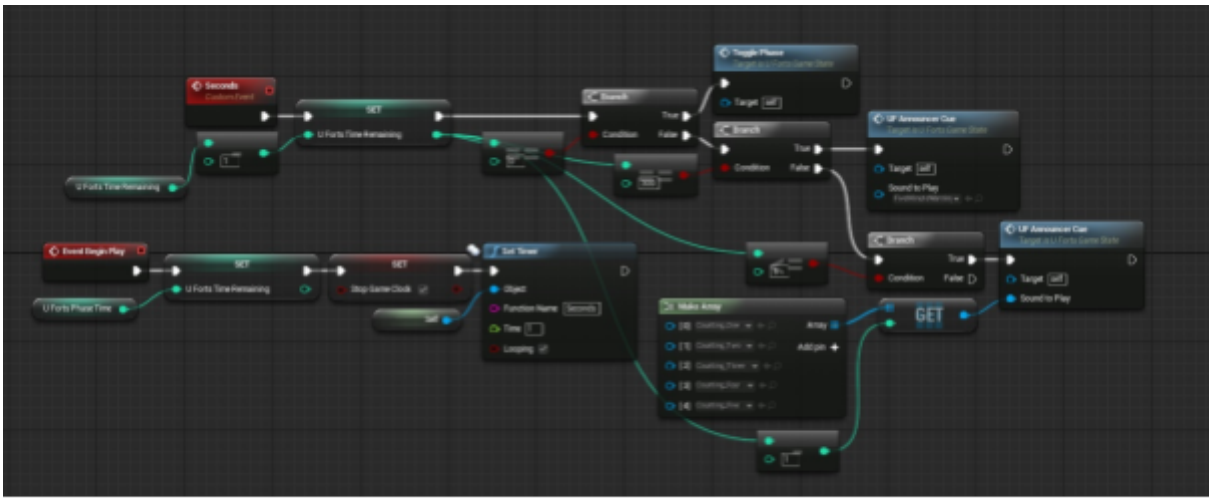

Figura 2.18: Ejemplo de Blueprint en Unreal Engine 4

Fue presentado en 2014 en la **Game Development Conference** (Game Development Conference. (n.d.)) anunciando un nuevo modelo de suscripciones en el que cualquiera puede utilizar UE4 y comercializar su juego pagando 19$ al mes y cediendo un 5% de los ingresos del mismo.

Tras el lanzamiento de su tienda de recursos Marketplace, Epic decidió que Unreal Engine 4 tendría licencias gratuitas para universidades, lo que fue el germen para que en marzo de 2015 se anunciase que era de uso gratuito para todo el mundo siempre y cuando un 5% de los beneficios a partir de los 3000$ del videojuego creado fuesen a parar a sus arcas.

Cabe destacar que Oculus ha propuesto a desarrolladores pagarles lo correspondiente a las regalías que corresponden a Epic si los desarrolladores eligen poner a la venta su juego de RV en la tienda digital de la que son propietarios, la Oculus Store.

2.2.3. CryEngine

CryEngine es un entorno de desarrollo creado por la desarrolladora alemana Crytek utilizado en múltiples videojuegos que apareció por primera vez en un videojuego de disparos en primera persona titulado Far Cry. Dicho juego nació como una demo técnica para tarjetas gráficas Nvidia y, tras ver su éxito, se decidió utilizar el entorno para desarrollar nuevos juegos (Página oficial CryEngine. (n.d.)).

Tras algunas actualizaciones gráficas en su primera versión, la segunda versión del entorno es utilizada para desarrollar otro de los juegos más famosos de la compañía: Crysis. Más tarde, los títulos de la franquicia se convertirían en una suerte de estándar para medir el rendimiento máximo para videojuegos en un PC.

En 2009 Crytek anuncia su tercera versión del entorno, que permitirá el desarrollo para Windows, PlayStation 3, Xbox 360 y Wii U. También añadirán soporte para Directx 9, 10 y 11. Su tercera versión recibirá más actualizaciones, sobre todo mejoras del motor gráfico y la aparición de un SDK para *mods* que permitía la creación de mapas y añadidos para Crysis 2. También aparecería una versión gratuíta de CryEngine para uso no comercial llamada CryEngine Free SDK.

A partir de esta versión, Crytek rebautizan su entorno como CryEngine (a partir de la versión 3.6.0) y no vuelve a aparecer una versión numerada hasta CryEngine 5. Hasta la salida de éste, se añadirán nuevas actualizaciones y soporte para Linux y la nueva generación de consolas PlayStation4, XboxOne y compatibilidad con los sistemas de RV.

CryEngine 5 soporta de forma nativa DirectX 12, Vulkan y RV y fue liberada al uso público de forma gratuita para el desarrollo de videojuegos y acceso al código fuente mediante un sistema de donaciones voluntarias.

CryEngine 5 fue liberado en marzo de 2016, así que la comunidad y la documentación para formación son aún escasas.

2.3. Software desarrollado: experiencias y videojuegos

Para poder fijar con precisión los objetivos específicos de este proyecto, es necesario revisar el estado de la técnica, los proyectos que ya han sido desarrollados para RV. De esta forma podremos tener un mayor conocimiento del área y aprovechar o descartar aquello que se haya probado como válido o no.

Este apartado se hace especialmente difícil, pues aunque como hemos visto el hardware ya no es un problema, nos encontramos en un terreno poco explotado hasta la fecha en lo que a variedad se refiere. No es de extrañar, si tenemos en cuenta que buscamos trabajar con una tecnología realmente joven. No obstante, 2016 es considerado por muchos como “el año de la RV”, y es precisamente la novedad de la tecnología lo que ha propiciado que numerosas empresas desarrolladoras se lancen a la creación de experiencias para esta tecnología.

Haremos en el presente documento varias distinciones antes de entrar en materia. Por un lado analizaremos las experiencias centradas en la narración y en sumergir al usuario en un entorno virtual, sin interactuar con el mismo más allá del movimiento de la cámara. La mayoría de estas experiencias han sido desarrolladas para Oculus Rift o pensadas para ser reproducidas con smartphones mediante la ayuda de una carcasa acoplada como Google Cardboard o Samsung Gear VR. Seguidamente nos centraremos en aquellos desarrollos que requieren una mayor interacción por parte del usuario, dividiendo además estos últimos en dos categorías diferentes: aquellos que se sirven de un controlador genérico o clásico (como el teclado, el ratón o los controladores de juego tradicionales), y los que utilizan controladores específicos como ocurre con los controladores de HTC Vive, Oculus Touch o PlayStation Move.

2.3.1. Experiencias no interactivas

Con el lanzamiento de Oculus Rift llegaron una serie de experiencias no interactivas que buscaban, principalmente, mostrar al gran público de lo que era capaz el dispositivo. Es por esta razón que los primeros pasos de la RV actual resultan puramente comerciales. Ejemplos reconocidos de estas primeras demostraciones son ***Oculus Rollercoaster*** Página oficial Oculus Rollercoaster. (n.d.)), una simple montaña rusa en la que el usuario sólo puede mirar el paisaje desde el vagón, y ***Oculus Dream Deck (***Página oficial Oculus Dream Deck. (n.d.)) , una colección de experiencias sin ninguna conexión entre ellas que sirvieron de introducción a la RV para muchos.

Figura 2.19: Usuaria de Oculus Rift experimentando Oculus Rollercoaster

Gracias a la impresión inicial, estas experiencias calaron en el público, abriendo las puertas a desarrollos más elaborados. Sin embargo, el elevado precio de Oculus Rift dejaba estas demostraciones confinadas en salones especializados y exposiciones. Es aquí donde entran en escena los visores para móviles.

Vídeos de 360º en Realidad Virtual

La llegada al mercado de las carcasas VR para *smartphones* supone una gran explosión de nuevas experiencias, generalmente no interactivas, de RV. Al dejar dichas carcasas toda la

capacidad de procesamiento al dispositivo móvil, estas experiencias resultan simples, pero efectivas en su mayoría. Dentro de este campo podemos hacer varias divisiones:

- **Experiencia promocional:** Entendida por aquella experiencia que pretende servir como puerta a otra experiencia. Es el caso de ***Insidious VR,*** parte de la campaña promocional de la película ***Insidous: Chapter 3*** (2015); la carrera de cuádrigas de ***Ben-Hur*** (remake 2016); o ***Planet of the Couches,*** un gag realizado para la conocida serie de televisión *Los Simpson.*
- **Experiencia artística:** Aquellas experiencias desarrolladas por el simple hecho de disfrutarlas o con la intención de comunicar algo. En este apartado cabe destacar ***Google Spotlight Stories,*** un proyecto de Google que busca narrar historias sirviéndose de la RV.

- **Experiencia amateur:** Generalmente se trata de vídeos rodados con la ayuda de una cámara en 360 grados como las citadas en apartados anteriores (2.1.4.3. Cámaras y sensores). Se suelen utilizar para compartir experiencias de la vida real, como deportes extremos, aunque se están empezando a utilizar también para informar.

El auge del género de terror

Antes de pasar a detallar las que hemos clasificado como experiencias interactivas, es importante destacar lo bien que han sabido aprovechar los creadores de historias terroríficas, el género en el que nos centramos en este trabajo, la inmersión proporcionada por la RV. Dentro de las experiencias no interactivas encontramos gran número de estas historias, siendo incluso las más exitosas algunas creadas por desarrolladores independientes como, por ejemplo, ***The Dark Inside*** para Google Cardboard.

2.3.2. Experiencias interactivas

Entendemos como experiencia interactiva aquella en la que se requiere la participación del jugador más allá de la simple observación. Es decir, deberá intervenir con sus acciones en los sucesos que ocurren en el mundo virtual. Es por esto que no hemos considerado las experiencias hasta ahora mencionadas como interactivas, ya que ninguna de ellas requería ninguna acción por parte del usuario para **reproducirse** hasta el final. Si bien es cierto que éste podía mover la cabeza y observar la escena de diferentes maneras, se convertía en un simple espectador durante la experiencia.

Las experiencias que detallamos a continuación, y que destacamos debido a su mayor relación con nuestro proyecto, esperan del usuario algún tipo de acción para continuar con su ejecución; o bien ofrecen al mismo la oportunidad de alterar la experiencia de alguna manera aunque la interacción no sea obligatoria. Dichas acciones pueden ir desde las más simples, como caminar o fijar la mirada en un punto concreto para producir una consecuencia; hasta otras más complejas como la resolución de puzzles y acertijos mediante el uso de algún tipo de controlador hardware (2.1.4. Periféricos). Estas últimas generalmente requieren de varias acciones más simples, como pueden ser accionar una palanca o mover algún tipo de objeto dentro de la experiencia. Tras estas acciones, se

debe responder de alguna manera al usuario, dándole a entender que su acción ha ocasionado alguna reacción.

Interacción con controladores clásicos

Controladores clásicos en videojuegos son el teclado, el ratón y dispositivos como el controlador de juego (gamepad), cuyo uso requiere unas acciones físicas por parte del jugador (pulsar botones o accionar palancas) que distan mucho de las acciones que su avatar realiza supuestamente en el mundo (saltar, correr, lanzar bolas de fuego, ...).

Dentro de esta categoría englobamos todas aquellas experiencias que utilizan los dispositivos de RV como una manera de presentar el entorno al usuario y ofreciendo casi las mismas posibilidades de interacción que en un juego sin RV a los controladores antes mencionados. Así, el visor actúa únicamente como una mejor interfaz, proporcionando una mayor sensación de presencia que las pantallas tradicionales.

Adaptaciones de experiencias inicialmente no diseñadas para RV

Algunos desarrolladores con productos ya consolidados en el mercado, como CCP Games y su ***EVE: Online*** (Página oficial EVE: Online. (n.d.)) vieron en la RV la oportunidad de ofrecer al usuario final una mayor inmersión en sus títulos, ya de por sí bastante impresionantes. El mencionado *EVE: Online* es famoso en el sector por ser el videojuego multijugador masivo en cuyo mundo virtual han tenido lugar las batallas entre jugadores más multitudinarias que se han visto hasta la fecha, llegando a unir en el mismo momento a varios miles del jugadores al mando de sus naves[16].

Dado el carácter épico de este videojuego concreto, no es de extrañar que la empresa que lo desarrolló hace más de una década se fijara en la RV, dando lugar a ***EVE: Valkyrie***. Se trata de una experiencia en la que el usuario debe pilotar una nave en medio de una batalla interestelar. Además de ser una experiencia intensa, nos sirve para comenzar a adentrarnos en la RV interactiva, pues lo cierto es que el jugador no necesita realizar ningún movimiento especial para disfrutarla. Dado que nos encontramos sentados tanto virtualmente como en la realidad, la credibilidad y sensación de presencia es realmente alta, pues los mandos de la nave también se han diseñado buscando la similitud con el controlador que tendremos en la mano.

Entrando ya en el género de terror cabe destacar ***The Kitchen***[17], una experiencia para PlayStation VR, desarrollada por Capcom como parte de la campaña publicitaria de su lanzamiento *Resident Evil 7* (enero de 2017). La interactividad de dicha experiencia es limitada, dado que sitúa al usuario atado en una silla y, a pesar de que éste puede utilizar el controlador de PlayStation para mover las manos, no puede hacer nada más para intervenir en la trama de la historia que se presenta.

[16] http://www.bbc.com/mundo/noticias/2013/07/130730_tecnologia_eve_online_guerra_espacial_dp

[17] Si se sigue el siguiente enlace, se puede ver la experiencia completa en YouTube: https://www.youtube.com/watch?v=NYnpDN0QCZU

En un terreno más independiente, nos encontramos con ***Emily Wants To Play (***Página oficial Emily Wants To Play. (n.d.)), un videojuego creado por el desarrollador independiente Shawn Hitchcock. Dicho juego nos pone en la piel de un repartidor de pizza que llega a una casa aparentemente abandonada y se ve atrapado en ella, teniendo que resolver el misterio que allí se encierra para poder escapar. Este proyecto vio la luz como videojuego convencional en diciembre de 2015, siendo adaptado para RV en agosto de 2016.

Figura 2.20: Captura de pantalla de EVE: Valkyrie, una de las primeras y más exitosas adaptaciones de videojuegos a la Realidad Virtual.

También con este ambiente tétrico tenemos la demo de ***Outlast 2 (***Página oficial Outlast 2. (n.d.))***,*** videojuego desarrollado por Red Barrels Games (2017). Aunque el juego completo no es compatible con ningún casco de VR, sí lo es la mencionada demo, a la que podemos jugar utilizando un visor y un controlador clásico de juegos.

Por último, como ejemplo de experiencia plenamente interactiva encontramos ***Please, don't touch anything (***Página oficial Please, don't touch anything. (n.d.)), un juego de puzzles inicialmente desarrollado para móviles y PC con una estética en dos dimensiones. El jugador se encuentra ante un panel con un gran botón rojo y se le pide que "no toque nada". Obviamente el juego pretende todo lo contrario, y el objetivo es el de tocarlo todo para encontrar los numerosos finales alternativos del juego. Esta experiencia fue adaptada más tarde a la RV, redibujando todo el entorno para convertirlo en un entorno en tres dimensiones, pero manteniendo la dinámica de juego y los objetivos. La interacción con dicho entorno se realiza con el visor, desplazando la vista hacia el objeto con el que queremos interactuar.

En esta sección hemos podido comprobar que la RV se ha utilizado ya tanto para promocionar otras experiencias ajenas a la misma, como para adaptar experiencias que ya funcionaban sin necesidad ella.

Experiencias diseñadas exclusivamente para Realidad Virtual

Aunque algunas de las experiencias antes mencionadas fueron rediseñadas para la RV, entendemos por experiencias **exclusivas** aquellas donde no existe necesidad de un contexto previo como *EVE: Valkyrie*, ni complementan una experiencia ajena a la RV como ocurre con *The Kitchen* y *Resident Evil 7*. Es decir, se diseñaron con el único objetivo de

ofrecer una experiencia completa en RV, y sus mecanismos no están relacionados con ningún videojuego clásico o película.

Es el caso de ***Technolust*** *(*Página oficial Technolust. (n.d.)) , una experiencia de algo más de una hora de duración lanzada junto con Oculus Rift en abril de 2016. En ella se introduce al jugador en un futuro distópico y se le anima a resolver una serie de puzzles y acertijos para llegar al final. Su escasa hora de duración hace que la historia que pretende contar sepa a poco, pero los niveles y la experiencia general resultaron algo bastante novedoso en el momento de su lanzamiento. Además, resultó una de las primeras experiencias que trataban de unir narración e interactividad.

Sin embargo, la experiencia que hemos encontrado más reseñable en este aspecto, por parecernos original además de divertida, es ***I expect you to die*** (Página oficial I expect you to die. (n.d.)). En este videojuego, se introduce al jugador en la piel de un espía internacional al estilo de James Bond y se le invita a escapar de las trampas de algún tipo de supervillano. El juego combina narración con una serie de puzzles suficientemente cortos como para poder evitar la sensación de mareo que se produce durante sesiones de RV demasiado largas. Volveremos más tarde sobre él, pues aunque originalmente se diseñó para controladores clásicos, fue adaptado para jugarse con Oculus Touch.

Interacción con nuevos controladores

Entendemos por nuevos controladores aquellos que permiten al usuario interactuar con la experiencia mediante el uso de su cuerpo, creando una ilusión más realista que con el uso de los controladores clásicos. Así, si el usuario mueve una mano o un brazo, o incluso si se mueve dentro del espacio de la habitación en la que está disfrutando de la experiencia, dicho movimiento será reconocido y representado en RV (se puede encontrar más información a este respecto en la sección 2.1.4.2 de este trabajo).

De lo clásico a lo moderno

Empezaremos este apartado hablando de nuevo de adaptaciones: juegos y experiencias originalmente concebidas para controladores clásicos que terminaron siendo adaptadas a los nuevos controladores. Dentro de este campo, sin embargo, también cabe diferenciar aquellas experiencias diseñadas directamente para RV, de aquellas que no lo fueron. Empezaremos por estas últimas, ya que en su mayoría no se trata de experiencias completas, sino más bien de una suerte de minijuegos que pretenden complementar la experiencia original.

Es el caso de ***Universe Sandbox 2*** *(*Página oficial Universe Sandbox 2. (n.d.)). El juego original invita al usuario a experimentar con la física del universo, colocando planetas y estrellas en un espacio abierto para comprobar más tarde cómo se comportan en su interacción con el resto de cuerpos celestes. Se trata de un videojuego basado en lo que se conoce comúnmente como *point-and-click*, dicho de otra manera, se sirve fundamentalmente del uso del ratón para experimentarlo por completo. Por eso su adaptación a RV utilizando los controladores de HTC Vive resultó sencilla y no se perdió casi ningún elemento de jugabilidad e interacción. Los usuarios del juego se mostraron

encantados con la adaptación, pues la RV permitía experimentar la sensación de estar en el espacio de una manera ciertamente más realista.

Similar a *Universe Sanbox 2* por su mecanismo *point-and-click*, aunque más enfocado hacia el humor y con un tipo de interacción aún más simplificado, encontramos ***Surgeon Simulator (***Página oficial Surgeon Simulator. (n.d.))***.*** En este caso la experiencia se centra en un movimiento bastante simple, como es el de coger y mover objetos de un lugar a otro.

Por último encontramos una serie de juegos diseñados desde un principio para RV, pero que no tuvieron en cuenta los nuevos controladores en el momento de su creación y más tarde sufrieron una actualización para adaptarse a ellos. Como ejemplo particularmente bien ejecutado de este tipo encontramos el ya mencionado ***I expect you to die***. Originalmente el usuario utilizaba el ratón para interactuar con el entorno, haciendo click en aquellos elementos que quería alcanzar, y era la propia experiencia la que acercaba estos elementos a la vista del jugador. Este método resulta intuitivo, pues las aventuras gráficas[18] se han servido de él a lo largo de toda la historia de los videojuegos y se ha probado como modelo de interacción eficiente hasta en usuarios no familiarizados con los videojuegos.

Sin embargo, este sistema resulta altamente incompatible con la búsqueda del realismo que se persigue cuando se utilizan los controladores modernos como son los de HTC Vive por una sencilla razón: el ratón se utiliza con una mano y los controladores con dos. Esto plantea una disyuntiva a la hora de adaptar la mecánica de juego. ¿El jugador debe utilizar sólo un controlador y dejar su mano izquierda inmóvil durante la experiencia? Tratándose como se trata de una narración en la que el usuario encarna a un agente secreto, lo ideal sería evitar esto, pues puede crear la ilusión de que el espía no sabe utilizar sus dos manos, algo totalmente incompatible con la idea de la experiencia.

Así pues, se incluyó un mecanismo de juego nuevo que permitiera al jugador atraer hacia sí aquellos objetos lejos de su alcance: la telequinesis. Apuntando con cualquiera de los dos controles hacia un objeto, éste queda resaltado, y el jugador puede entonces utilizar el controlador para moverlo por la estancia y recogerlo[19].

Nuevo paradigma de diseño

Con la aparición de HTC Vive y, más adelante, de los controladores Oculus Touch, así como el lanzamiento de un PlayStation VR compatible con los ya conocidos PlayStation Move, el diseño de videojuegos para RV comienza a desligarse del desarrollo original, dando lugar a nuevos mecanismos de juego y patrones diferentes a los ya conocidos. Nos encontramos ya en un punto en el que la interacción comienza a coger mayor peso en la experiencia de RV. Un punto en cierto modo desconocido para la mayoría de

[18] Género de videojuegos basado en la consecución de puzzles mediante acciones cotidianas como hablar con otros personajes o recoger y examinar objetos. Generalmente su control se basa únicamente en el uso del ratón.

[19] Además de este problema, los desarrolladores de Schell Games encontraron otros que se pueden leer más detallados en este artículo escrito por John Kolencheryl, director de tecnología de *I Expect You To Die*: http://www.gamasutra.com/blogs/JohnKolencheryl/20161214/287450/Engineering_a_Super_Spy_in_I_Expect_You_To_Die.php

desarrolladores experimentados que abre las puertas a una industria completamente distinta a la que estamos acostumbrados.

Como ocurriera con los primeros videojuegos tradicionales, los desarrollos en RV se centran más en los mecanismos del juego que en el guión o la historia que desean contar, entendiendo por mecanismos aquellas acciones que el jugador debe realizar para la consecución de uno o varios objetivos. El desarrollo para RV es un terreno poco explorado y se hace necesario desarrollar una dinámica básica y encontrarse cómodo con ellas antes de pasar a desarrollos más complicados. Se debe crear una base similar a lo que fue el primitivo *Super Mario Bros (*Página oficial Super Mario Bross. (n.d.)*).* (1985) para los sucesivos videojuegos de género plataformas que vinieron después; o los clásicos como *Wolfenstein 3D (*Página oficial Wolfenstein. (n.d.)) (1992) y *DOOM* (Página oficial DOOM. (n.d.)) (1993) para los shooters actuales.

Figura 2.21: Super Mario Bros., Wolfenstein 3D y Doom. Todos ellos presentaron dinámicas y mecanismos de juego innovadores que más tarde fueron asumidos en sus respectivos géneros.

En este sentido se hace realmente destacable el título ***The Lab*** (Página oficial The Lab. (n.d.).), un videojuego desarrollado por Valve que convierte al usuario en un científico de Aperture Lab, una empresa imaginaria que realiza una serie de experimentos de dudosa naturaleza ética. Si bien es cierto que Aperture Lab forma parte del imaginario de una de las franquicias más exitosas de la desarrolladora (*Portal,* 2007), resulta más un sencillo guiño que un punto importante de guión.

Valve ha intentado plasmar en *The Lab* varias de las interacciones que se pueden realizar con los controladores de HTC Vive. El juego resulta ser una especie de “caja de arena” (o laboratorio, de ahí el nombre) en el que los desarrolladores han querido experimentar con total libertad cómo funcionan dichos controladores en un entorno virtual. Encontramos pues varios niveles sin ninguna conexión entre ellos más allá de que todos se encuentran en el mismo laboratorio, que hace las veces de menú conceptual para navegar entre los mismos. Lo único que buscan estos niveles es obligar al usuario a realizar diferentes acciones con los controladores. Para el movimiento dentro del entorno se sirve de un sistema que llamaremos de **teletransporte** (al igual que muchos otros videojuegos que nombraremos más adelante). En este sistema, el usuario simplemente apunta al lugar al que se quiere mover (este lugar siempre tiene que estar en el campo de visión del jugador) acciona el controlador y enseguida se ve teletransportado allí. Para que la experiencia no pierda por completo el realismo, existe una distancia máxima a la que el jugador puede teletransportarse siempre limitada al .

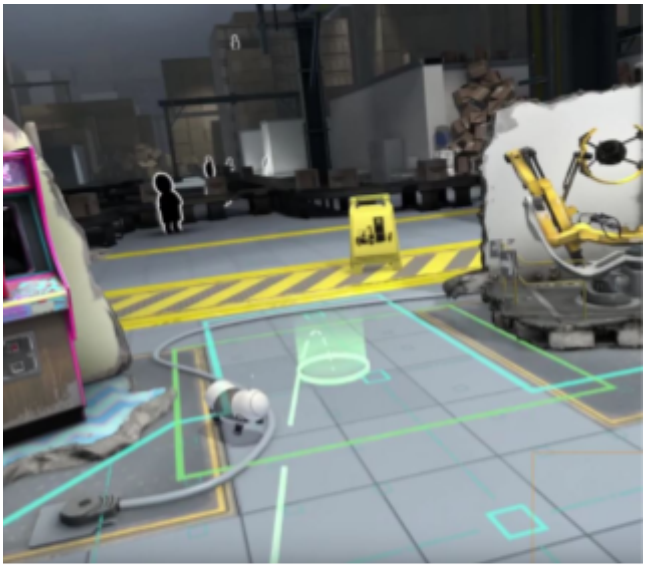

Figura 2.22: Ejemplo de movimiento por teletransporte en *The Lab*. Tras dejar de presionar el botón en el controlador, la escena se trasladará al lugar apuntado. La malla dibujada en el suelo delimita el espacio al que el jugador puede teletransportarse.

Cabe destacar que Valve también ha querido explorar la interacción del entorno con el jugador, y no sólo en el sentido opuesto. También se muestra un ligero desarrollo en inteligencia artificial, como una suerte de perro robot que es capaz de perseguir un palo lanzado por el jugador en uno de los niveles; o una extraña criatura que sigue el controlador con la mirada en otro.

Interacción dirigida

Antes de centrarnos en el estado de los videojuegos narrativos con componentes interactivos, cabe destacar también las experiencias que invitan al jugador a superarse a sí mismo, ya sea creando su propia experiencia o superando una puntuación.

Este tipo de juegos podrían ser el futuro de lo que conocemos como *arcade*: juegos pensados para ser disfrutados en sesiones cortas y originalmente diseñados para colocarse en algún salón de juego. Si bien es cierto que dichos salones tienden a desaparecer en la actualidad, no así los juegos que hemos decidido nombrar como de interacción dirigida.

Entendemos por juegos de interacción dirigida aquellos cuyo objetivo está fijado desde un principio, y todas las acciones que realiza el jugador deben ir encaminadas a lograr ese objetivo. Ejemplos de estos juegos pueden ser juegos de conducción, de sincronización (baile, simuladores como *Guitar Hero* (2005), etcétera) o juegos de construcción.

Precisamente en este último género encontramos *Fantastic Contraption*(Página oficial Fantastic Contraption. (n.d.)), título que invita al usuario a construir elementos con formas dispares a las que deberá dar forma; o *Tilt Brush (*Página oficial Tilt Brush. (n.d.)) una experiencia puramente creativa desarrollada por Google que pone al alcance del jugador una serie de herramientas con el fin de permitirle, simplemente, “pintar” una especie de escultura tridimencional en el aire.

Algunos de estos mecánicos son realmente interesantes por el uso que hacen de los controladores, pero no ahondaremos mucho más en este terreno por no tener tanta relación con el propósito de con nuestro proyecto como lo ya mencionado anteriormente.

Interacción libre

Es cierto que la práctica totalidad de empresas y desarrolladores independientes parecen más centrados actualmente en entender cómo lograr que la interacción con la RV sea lo más fluida posible, pero algunos desarrollos actuales comienzan a derivar hacia experiencias narrativas que buscan algo más que simplemente llevar al jugador a mundos alternativos.

En este sentido es reseñable ***Budget Cuts*** (Página oficial Budget Cuts. (n.d.)), producto actualmente en desarrollo por la empresa Neat Corporation. Si bien a fecha de diciembre de 2016 no se conocía mucho acerca de la trama que intentará desarrollar dicho título, lo que la desarrolladora ha mostrado hasta ahora nos deja ver un videojuego de sigilo en el que el jugador tendrá que superar una serie de objetivos concretos. Sus mecanismos aprovechan mucho de lo ya mencionado -movimiento por teletransporte, posibilidad de interactuar con el entorno- pero hace una diferenciación clave entre los dos controladores: el izquierdo se utiliza para moverse y previsualizar el entorno al que queremos desplazarnos (se trata de una especie de *gadget* espía), mientras que el derecho sirve para llevar a cabo todas las interacciones (coger, lanzar, accionar botones y palancas, abrir puertas, etcétera). Además añade un componente de inteligencia artificial para crear una serie de enemigos recorriendo el entorno en busca del jugador, que deberá permanecer oculto el mayor tiempo posible. Es destacable que dichos enemigos reaccionen a las interacciones del usuario con el entorno, como cuando este lanza algún objeto para despistarlos mientras él se escabulle por otro camino.

Budget Cuts también aprovecha los sensores de HTC Vive para utilizar el propio entorno real como controlador de la experiencia a escala de habitación (room-scale). El jugador

puede agacharse, asomarse a ventanas para obtener mejor visión del entorno virtual e incluso realizar interacciones complejas en cualquiera de estas posiciones.

Pero quizá lo más destacable hasta el momento en materia de narración sea ***The Gallery: Call of the Starseed*** (Página oficial The Gallery: Call of the Starseed. (n.d.)), un videojuego episódico que narra la historia de un hombre en busca de su hermana perdida. Comprobamos en este título que también utiliza el sistema de movimiento por teletransporte, pero de una manera quizá más "creíble" que en el citado *Budget Cuts.* El objetivo del juego será el de seguir una serie de pistas que nos llevarán hasta la hermana perdida, descubriendo por el camino la historia de un mundo fantástico y olvidado.

Como podemos ver, el guión de esta experiencia es bastante profundo para lo que nos tiene acostumbrados hasta el momento la RV, pero el título no olvida los mecanismos de juego típicos como el ya citado movimiento por teletransporte, o la posibilidad de agarrar objetos. Añade además algo muy interesante, y es que el jugador podrá llevar un inventario de objetos almacenado en una mochila. Dicha mochila puede abrirse mediante un movimiento muy característico, que es llevarse la mano a la espalda. El título introduce así una forma intuitiva de llevar a cabo acciones muy típicas de los videojuegos, y abre la puerta a la creación de mecanismos básicos y repetibles en futuros títulos que ya mencionábamos al principio de este apartado.

El estado del género de terror en Realidad Virtual con interacción libre

Como ya mencionamos anteriormente, el género de terror fue un activo en alza durante los comienzos de la RV, siendo todas las experiencias planteadas en este campo de un alto contenido narrativo. Dichas experiencias se centran en la generación de sensaciones en el usuario, dejando un poco de lado la interacción de este mismo con el entorno.

En el campo interactivo que hemos analizado en esta última parte encontramos pocas experiencias reseñables, aunque podemos fijar nuestra vista en ***A Chair in a Room*** (Página oficial A Chair in a Room. (n.d.)) para comprobarlo. En dicha experiencia, el jugador se mete en la piel de un supuesto paciente en un psiquiátrico y debe pasar por distintas pruebas. El jugador no sabe muy bien dónde está ni por qué el protagonista acabó en dicho psiquiátrico en un principio, teniendo que averiguar la historia mediante la investigación del entorno. En el aspecto técnico encontramos algunos fallos de ejecución -podemos atravesar paredes, algunos objetos son difíciles de coger, etcétera-. Estos fallos son tristemente comunes entre los juegos que buscan mayor profundidad en su trama, pues los desarrolladores de los mismos parecen centrarse en las tareas concretas que el jugador debe realizar, olvidando la libertad de movimientos que proporciona la RV. Sin duda se trata de una experiencia que nos enseña muchos detalles a tener en cuenta en nuestro proyecto.

2.4. Aplicaciones y retos de la Realidad Virtual

Tras conocer los dispositivos utilizados en la presentación de mundos virtuales, pasamos ahora a investigar tanto los desafíos asociados a la tecnología, los problemas más

comunes y la manera en que se están solventando, como los beneficios de sus aplicaciones que han sido encontrados hasta el momento.

Centraremos este estudio en los síntomas que produce la RV **en el individuo,** ya que se trata de una tecnología preferentemente enfocada hacia el usuario. No existen visores de RV que se puedan utilizar en grupo, por lo que la experiencia está enfocada al uso individual.

2.4.1. Problemas compartidos con los videojuegos clásicos

A la RV se le asocian muchos de los síntomas que padecen las personas que abusan del uso de videojuegos. Entre ellos destacan que tiene una alta componente adictiva y esto hace que el individuo se distancie del mundo real, pudiendo llegar a aislarse de sus relaciones personales. Muchos científicos han rebatido esta hipótesis, ya que sólo un uso excesivo de la experiencia produciría este tipo de síntomas, exactamente igual que ocurre con los videojuegos, como podemos apreciar en el estudio de Andrea Berger **(**Berger,Karpel, Lejbowicz & Racki; 2013).

Además, aunque también es común al resto de videojuegos, los problemas de visión el jugador dificultan en mayor grado la experiencia con la RV. Al tratarse de un mundo virtual por imágenes en 3D, el usuario con estos problemas puede no llegar a disfrutar al 100% y, en casos muy acentuados, puede que ni siquiera sea capaz de utilizar un visor. Cabe destacar en este punto que la mayoría de visores pueden ser utilizados con lentillas y en ocasiones incluso con gafas. Algunos de ellos disponen de ajustes para miopía, hipermetropía, al igual que ajuste de distancias entre pupilas, como veremos más adelante en este mismo estudio.

2.4.2. Problemas específicos de la Realidad Virtual

Hemos comentado ya algunos de los problemas más típicos dentro del mundo de los videojuegos que, de algún modo, se acentúan con el uso de la RV. Sin embargo, existen problemas extendidos dentro de esta tecnología como, por ejemplo, la **cinetosis.**

La cinetosis es un trastorno del equilibrio debido al movimiento en las personas. En este caso particular hablamos del subtipo llamado “mal del simulador” (*simulator sickness*), en el que los sentidos no se encuentran en armonía y se producen incongruencias por faltas de sincronización entre la información que estos reciben. El oído no detecta movimiento porque permanecemos estáticos, aunque la vista nos dice lo contrario ya que hay movimiento en el visor de RV, por lo que el cerebro responde con náuseas y mareos.

Este problema aparece especialmente cuando se lleva al usuario a realizar movimientos bruscos o la experiencia presenta excesiva acción (Abubakra,Herrero, Fernández, Sanz., & Zabala; 2015).

Oculus Rift plantea una serie de recomendaciones a seguir para tratar de reducir o prevenir los síntomas del mal del simulador (Oculus Rift VR Motion Sickness - 11 Ways to Prevent It! 2015):

- Utiliza el giro automático de personaje en tercera persona, siempre y cuando sea posible.
- Centra tus ojos en un punto fijo durante la partida.
- Mueve tu personaje a la menor velocidad posible, reduciendo el campo de visión.
- Juega partidas cortas y toma breve descanso antes de volver a empezar otra nueva partida.
- Cierra los ojos cuando la cámara se mueva sola en lugar de responder a tus movimientos de tu cabeza.
- No te fuerces utilizando juegos mal optimizados.
- Asegúrate que la distancia interpupilar en tu visor es correcta.
- Baja el brillo al mínimo.
- Come o bebe productos que contengan jengibre, se utiliza desde hace siglos contra el mareo.
- Intenta no mover mucho la cabeza.
- Intentar evitar caminar hacia atrás durante el juego en la medida de lo posible.
- Asegurarte que tu ordenador soporta una tasa de frames elevada y estable. De lo contrario considera cambiar las componentes que limitan los fps.
- Descarga también los parches capaces de reducir los mareos que proponen los distintos fabricantes. Alguna vez tienen sus contras ya que pueden reducir la inmersión.

2.4.3. Aplicaciones de la Realidad Virtual

Hasta ahora hemos estado hablando de los problemas que podemos encontrar al utilizar la RV. Estos problemas no impiden que existan numerosas aplicaciones y beneficios potenciales que la RV puede llegar a ofrecer en los distintos aspectos de nuestra vida.

A lo largo de las siguientes líneas vamos a comentar alguno de los casos donde los mundos virtuales han resultado determinantes para mejorar la calidad de vida tanto a nivel profesional como a nivel personal en diferentes sectores.

Para empezar, vamos hablar de uno de los sectores que más nos preocupa a todos y cuyo avance nos interesa: la sanidad. Durante el último años la aplicación de la RV en este ámbito se ha disparado, pero de ahora en adelante se prevé que va a seguir aumentando todavía más.

En primer lugar, ya en muchos hospitales se está empezando a implantar salas con dispositivos de RV para ayudar a las personas con algún tipo de fobia a intentar superarla en un entorno controlado. Lo hace, por ejemplo, la empresa española **Psious** (Página oficial Psious. (n.d.)) , cuyo objetivo es ayudar a superar los problemas de salud mental utilizando la tecnología como instrumento. Todo surge con el fundador de la empresa, Daniel Roig, el cual padecía miedo a volar. Consiste en una terapia de exposición gradual en la que el usuario va enfrentándose a sus miedos durante las sesiones supervisadas, tanto por médicos como por psicólogos. Patologías sencillas como la de Daniel se pueden superar en tan solo 5 sesiones (Página oficial Psious-Sanitas, (n.d), 2017). Gracias a esto, tanto el paciente como sus doctores se pueden ahorrar una cantidad muy alta de dinero y, además, la RV puede ayudar a dar diagnósticos más certeros y encontrar remedios más efectivos.

Por otra parte, empresas como la de Daniel están teniendo grandes beneficios y siendo reconocidas con premios lo que incentiva ambos campos: el económico y el de la sanidad.

Siguiendo por este hilo, todos conocemos los casos en los que niños hospitalizados no pueden ni siquiera salir a la calle y apenas recibir visitas, por lo que pierden toda relación con el mundo real por culpa de su enfermedad. Hospitales como el del Niño Jesús de Madrid se han dispuesto a cambiar esto, obteniendo por ahora buenos resultados con la RV.

Figura 2.16. Uso de HTC Vive en el Hospital Niño Jesús de Madrid

La iniciativa es común entre el hospital y **Voluntechies** (Página oficial Voluntechies. (n.d.)) y consiste en recrear un paseo por el mundo exterior, por calles y lugares utilizando Google Earth, los que permite a los niños que no pueden salir de su habitación recrear vivencias que ellos tenían antes de entrar al hospital y, de esta manera, no alejarse demasiado de su vida y entorno real. También utilizan videojuegos en RV para aquellos casos que el niño no se pueda ni mover de la cama.

Uso en formación de profesionales

Por último, para acabar con el sector de la sanidad, cabe destacar los avances tanto para la formación de nuevos profesionales como la mejora en su trabajo de los ya profesionales.

La RV permite a los médicos y personal sanitario recrear escenarios de emergencia. Un ejemplo de los más comunes, es enseñar cómo deben responder los profesionales en casos que entren en la UCI, ya que para este tipo de trabajos hay que actuar rápida y eficientemente. Gracias a estos entrenamientos se puede practicar en un entorno recreado pero con problemas reales sin poner en peligro la vida de nadie.

Otro punto en auge actualmente, es el entrenamiento de cirugías, consistente en el aprendizaje para los estudiantes de medicina y cirujanos noveles. Este entrenamiento generalmente implica la práctica con cadáveres y un proceso gradual de apoyo a los médicos con más experiencia antes de asumir tareas y mayores responsabilidades en cirugía (Rodríguez-García, 2006). La RV proporciona otro medio complementario para aprender y prácticar, muy accesible y sin ningún tipo de riesgo para pacientes reales.

Además existen proyectos para la formación de distintos profesionales, como el proyecto

SICEMAM (Proyecto SICEMAM. (n.d.)), un sistema cibernético experto que busca reinventar el paradigma del mantenimiento actual de aeronaves.

Entretenimiento y Realidad Virtual

Una vez vistas aplicaciones el campo de la sanidad y la formación, nos vamos a meter de lleno con otro sector también muy importante y puntero, el cual queremos explotar a lo largo de esta memoria: el entretenimiento.

Hoy día en la sociedad moderna se consumen películas, series, videojuegos, etcétera con una frecuencia más alta que años atrás. De un tiempo a esta parte, los avances tecnológicos marcan la manera de consumir y utilizar estos medios y, en estos últimos años, comienza a cobrar relevancia el papel que puede jugar la RV en todo esto.

En primer lugar vamos hablar de la RV orientadas a películas, cortometrajes, videos de Internet, etc. La forma en que veíamos las películas y la experiencia de usuario al verlas ha cambiado. En marzo de 2016 se estrenó en Amsterdam un espacio llamado **VR Cinema** (Página oficial VR Cinema. (n.d.)) donde veremos a decenas de personas disfrutando del séptimo arte de una forma diferente. La compañía tiene previsto abrir nuevas salas en Londres, París, Berlín y Madrid.

Otro ejemplo de la potencia de la RV en este área es la asociación de IMAX con Google. En el marco de Google I/O 2016 (Página Google I/O 2016. (n.d.)), IMAX anunció que se asociará con la compañía de Mountain View para crear una cámara de RV, con calidad cinematográfica: IMAX VR camera. Evidentemente, el equipo de IMAX aportará su conocimiento en cuanto a cámaras, mientras que Google contribuirá con todo lo referente a la tecnología software (Google And IMAX Partner To Develop Groundbreaking Virtual Reality Camera, 2016).

Figura 2.17: Previsión de ingresos en RV según The App Date.

Para terminar este apartado quisiéramos adentrarnos en la industria del videojuego, la cual ya hemos tratado en detalle en secciones anteriores, aunque sin habernos detenido en sus cifras concretas. En la Figura 2.17 se encuentra un gráfico con los datos previstos de ventas de visores según el evento The App Date de la Fundación Telefónica ya citado anteriormente.

Como podemos observar y venimos hablando en las líneas anteriores, la RV se mueve en todos los ámbitos y tiene un futuro muy esperanzador tanto en lo económico como en lo tecnológico.

2.5. Conclusiones extraídas del estudio

Tras la revisión del estado de la técnica realizado, extraemos varias conclusiones que nos sirven de punto de partida para nuestro desarrollo:

- La RV lleva siendo un objetivo que parecía inalcanzable durante mucho tiempo. La tecnología actual ha permitido desarrollar nuevo hardware que permite acercarnos más a ese sueño de vivir en mundos virtuales, propiciando una carrera desenfrenada entre varias empresas por desarrollar la mejor experiencia.
- Existe una división palpable entre los visores de alta gama y aquellos que se sirven de la tecnología móvil para generar un entorno virtual. Estos últimos parecen centrarse principalmente en el desarrollo de experiencias no interactivas, por lo que es lógico suponer que los visores con pantalla propia y orientados al ordenador de sobremesa llevarán la voz cantante de la RV durante un tiempo, si no en lo comercial al menos en lo referente al avance de la tecnología.
- Existe mucho software experimental y sólo algunos se han atrevido a desarrollar experiencias habituales en videojuegos, con narrativas realmente cerradas y enfocadas a la consecución de objetivos para ir avanzando en la historia. No existe por el momento consenso bien definido en lo que a movimientos y mecanismos de juego básicos se refiere, aunque se empiezan a repetir patrones como el teletransporte y el uso de objetos en vez de menús, para mantener la sensación de presencia y maximizar la inmersión.
- El género de terror ha experimentado un auge durante los inicios de la RV, posiblemente por la facilidad que tiene este género a la hora de provocar emociones en el usuario. Con la aparición de controladores específicos para RV como los de HTC Vive y Oculus Touch, los desarrolladores parecen haber enfocado su trabajo a encontrar una manera adecuada de interactuar con el entorno, sin todavía entrar de lleno en la narración y en las emociones más sutiles.

Por todo esto decidimos enfocar nuestro proyecto a la realización de un videojuego en RV en el que el jugador deberá alcanzar unos objetivos concretos, resolviendo una serie de pruebas que le obliguen a sacar provecho del seguimiento de movimiento del visor y de los controladores manuales específicos. En estas pruebas buscaremos abarcar una variedad amplia de movimientos para comprobar cómo de difícil es desarrollar un producto funcional con estas características, así como averiguar hasta qué punto con nuestros recursos somos capaces de ofrecer una experiencia con sensación de presencia e inmersión suficiente. Pretendemos provocar el mayor impacto emocional posible, por lo que nos

hemos centrado en el género de terror, y concretamente en la temática de las fobias, para la historia y la ambientación, con la intención de que la imaginación del jugador ayude a potenciar también la experiencia.

3. Objetivos y metodología

En este capítulo concretamos los objetivos que deseamos alcanzar con la realización de este trabajo y exponemos la metodología y el modelo de proceso de desarrollo a seguir, que es una adaptación a nuestras condiciones de trabajo de la popular metodología ágil *Scrum*.

3.1. Objetivo general

Al comienzo del proyecto el objetivo no era otro que el de experimentar en primera persona en qué consistía el desarrollo de una experiencia interactiva para un entorno de RV, pensando en las posibilidades que tendríamos de crear un producto completo y listo para ser comercializado con estas características. La novedad de los controladores incluidos con el dispositivo HTC Vive nos motivó a conocer qué tipo de interacciones se podrían diseñar para los mismos, así como a investigar cuáles de esas interacciones resultan más o menos intuitivas para la experiencia completa que queríamos crear.

Aunque habíamos recibido formación acerca del desarrollo de software, no sabíamos si las metodologías aprendidas podrían ser aplicadas al caso concreto de los videojuegos, y videojuegos en RV además, por lo que decidimos añadir a ese objetivo inicial el de poner a prueba una metodología que consideramos adecuada a tal efecto. Por ello decidimos afrontar el proyecto de una manera ágil, diseñando y planificando todo el proceso de manera que pudiéramos identificar qué tareas resultan más difíciles o son más relevantes a la hora de desarrollar una experiencia de estas características. Esto nos permitiría hacernos una idea del esfuerzo y del coste necesarios para desarrollar proyectos de mayor envergadura en el futuro e identificar, si existiera, la mejor manera de iterar e ir desarrollando incrementos “atómicos” y bien definidos hasta alcanzar el objetivo.

En resumen, el objetivo general del presente trabajo es intentar adaptar la metodología ágil conocida como Scrum a nuestras condiciones de trabajo durante el desarrollo de una experiencia interactiva en RV, un videojuego de terror, utilizando tanto el visor como los controladores manuales de HTC Vive. Se pretende experimentar con una amplia variedad de formas de interacción dentro de la RV, para saber cuales resultan más satisfactorias para el jugador, en el sentido de proporcionar una experiencia lo más inmersiva y emocional posible. Este es un ámbito poco frecuentado en el desarrollo de software debido a la novedad de esta tecnología, y por eso consideramos importante documentar escrupulosamente todo el proceso para luego poder identificar las fortalezas y debilidades de aplicar la metodología Scrum en este desarrollo concreto.

3.2. Objetivos específicos

Tratándose la RV de una propuesta tecnológica nueva, y a pesar de que en este último año varias empresas han desarrollado aplicaciones y experimentado con ella, no existen convenios sobre lo que debe incluir o no una experiencia interactiva en RV.

Durante la revisión del estado de la técnica identificamos, por ejemplo, distintas formas de presentar información al usuario dentro del entorno virtual, o incluso maneras muy diferentes de desplazarse por dicho entorno. Entre nuestros objetivos específicos se encuentra conocer las múltiples posibilidades que ofrece la tecnología y comprobar por nosotros mismos cuáles de estas opciones nos resultan más intuitivas a la hora de disfrutar de la experiencia creada. Es por ello que en la especificación del proyecto incluiremos requisitos como:

- Tener libertad para mirar el entorno en sus 360 grados
- Moverse a cualquier punto (accesible) del entorno
- Coger objetos de distintos tamaños
- Manipular estos objetos con movimientos de las manos
- Consultar información a través de objetos que se pueden coger y manipular
- Hacer que objetos que tenemos cogidos interactúen con nuestro cuerpo
- Hacer que objetos que tenemos cogidos interactúen con otros objetos
- Utilizar las manos también para interactuar con el propio escenario
- Situar objetos de manera precisa dentro del entorno

3.3. Elección del dispositivo de Realidad Virtual

Como ya hemos mencionado antes, desarrollaremos el proyecto utilizando el dispositivo HTC Vive, que trae incluidos el visor de RV, los controladores manuales y emisores de señal para el posicionamiento.

Esta elección se debe principalmente a la disponibilidad del mismo por parte del Departamento de Ingeniería del Software e Inteligencia Artificial, así como del laboratorio de investigación al que pertenece nuestro director. Aunque también hubiera sido posible utilizar un sistema Oculus Rift disponible, o adquirir otro diferente, consideramos que se trata de un dispositivo que se adecúa más a nuestras necesidades. Realizamos nuestras pruebas y pudimos comprobar que mediante SteamVR, la plataforma ideal para HTC Vive que ofrece la propia Valve), este dispositivo de RV se encuentra totalmente integrado con los entornos de desarrollo más populares como son Unity y Unreal Engine.

3.4. Elección del entorno de desarrollo

Entre los múltiples entornos de desarrollo de videojuegos existentes, y concretamente de los estudiados en el capítulo del estado de la técnica, hay dos que como decimos son especialmente populares: Unity y Unreal Engine.

Tras un breve análisis de nuestras opciones nos decidimos por este último debido a la facilidad que presenta a la hora de integrar un desarrollo en RV con controladores específicos, ya que existen numerosos plugins que facilitan dicha integración. Esto, junto con la posibilidad de trabajar con una herramienta de programación visual como Blueprints, permite desarrollar niveles enteros del juego sin necesidad de acceso continuado al dispositivo de RV y pudiendo dividir más fácilmente el trabajo entre los tres integrantes del equipo.

Además, Unreal cuenta con una serie de tutoriales incluidos dentro del mismo programa que nos han parecido de gran utilidad para formarnos en la herramienta, y también dispone de una tienda digital donde al poco de comenzar el proyecto encontramos un pack de elementos prediseñados para principiantes que nos proporcionaban la posibilidad de incluir recursos audiovisuales muy efectivos para la atmósfera que queríamos crear en el juego.

3.5. Plan de trabajo

Durante la primera fase del proyecto sufrimos varios cambios en los objetivos y en los requisitos principales del mismo. Inicialmente nuestra idea era aprovechar un desarrollo previo perteneciente a un Trabajo de Fin de Grado de años anteriores y continuar desde allí, pero no contábamos con un permiso expreso de los autores para hacerlo, y su planteamiento de juego limitaba mucho nuestra libertad creativa a la hora de incluir situaciones en las que se requiriese una variedad de movimientos y el uso de controladores manuales específicos de VR, pues se trataba de una experiencia breve y pensada para controladores clásicos de juego.

Una vez establecidos los nuevos objetivos, comenzamos a reunirnos con el director del proyecto cada dos semanas aproximadamente para realizar con él el seguimiento del proyecto e irle transmitiendo los resultados del estudio y del trabajo que hemos detallado en esta memoria[20]. En estas primeras reuniones fueron surgiendo requisitos que incluir en la especificación, lo que acabaron cristalizando en un **videojuego modular de terror compuesto por varios niveles autoconclusivos en forma de mini-juegos de ingenio y habilidad.**

3.6. Modelo de proceso de desarrollo utilizado

El presente trabajo consiste en aplicar la metodología ágil Scrum, adaptada a nuestras necesidades, al desarrollo de un videojuego en RV. La idea surge de una propuesta del director, ante la dificultad que nos encontramos nosotros a la hora de plantear el desarrollo del videojuego desde nuestra posición de estudiantes sin experiencia en este tipo de aplicaciones. Al contar con una especificación muy inestable (que al ser nosotros mismos los clientes, se iba construyendo según progresábamos), y no poder dedicar todo nuestro tiempo al desarrollo del videojuego, consideramos buena idea utilizar una metodología flexible que nos permitiera ir alcanzando objetivos poco a poco, a la vez que estaba adaptada a nuestras capacidades y recurso como equipo de desarrollo.

Estudiando los distintos tipos de metodología a nuestro alcance vimos que la que más se adecuaba a nuestro proyecto y a nuestras necesidades era la metodología ágil. Elegimos esta filosofía ya que con su sistema de trabajo basado en entregas periódicas de software funcional y centrado en ir haciendo realidad las “historias” del cliente, nos permite percibir el avance del proyecto en todo momento y en caso de no poder alcanzar todos los objetivos, disponer al menos de un software funcional que entregar al final del curso. Tomada esta decisión, teníamos que elegir cual de todas las metodologías ágiles

[20] Las actas de estas reuniones pueden encontrarse en el ANEXO III de este documento.

utilizaríamos. Considerando distintos tipos, y viendo que la que más se utiliza en la el Grado y el Máster en Desarrollo de Videojuegos de la Facultad, nos decantamos por Scrum.

Scrum se basa en la adaptabilidad a los cambios (constantes en el desarrollo de un videojuego en RV, dada la naturaleza cambiante de la tecnología actual) como punto fuerte para aumentar las probabilidades de éxito en los proyectos. Su propósito principal es simplificar el proceso de desarrollo y poner todo el esfuerzo en lo que realmente importa: satisfacer las necesidades del cliente. Por otra parte, con el método de entregas periódicas y muy frecuentes de software funcional es más sencillo adaptarse a los cambios según surgen y ajustar los objetivos a corto-medio plazo si es necesario.

Las fases en las dividiremos el desarrollo son: planificación, especificación, análisis, diseño, implementación y pruebas. Utilizaremos diferentes herramientas para llevar a cabo nuestro trabajo en cada una de ellas, como detallaremos más adelante.

Pese a las pequeñas desviaciones que se produjeron en las entregas, las cuales resultaron asumibles, podemos decir que los objetivos planteados en cada uno de los hitos (*sprints* en la nomenclatura de Scrum) fueron cumplidos de forma razonablemente satisfactoria. Gracias a este cumplimiento, al final del plazo pactado conseguimos entregar una versión jugable y bastante completa de nuestro videojuego y pudimos sacar conclusiones bien fundamentadas sobre el esfuerzo y los recursos que necesitaríamos para desarrollar el proyecto a una escala comercial.

A continuación detallamos cómo adaptamos la metodología Scrum a las particularidades de nuestro proyecto y entorno de trabajo. Para obtener una descripción más extensa sobre la metodología utilizada se puede consultar el ANEXO IV de este mismo documento.

3.6.1. Adaptación de Scrum para este proyecto

Para poder utilizar la metodología Scrum y tratar de garantizar la consecución y calidad del proyecto, necesitamos aclarar bien los términos de nuestro proceso de desarrollo.

Como hemos mencionado, plantearemos un proceso de desarrollo ágil, que es iterativo, incremental y pone el foco en obtener resultados que satisfagan las necesidades del cliente. Este proceso se puede dividir en seis fases, que se ordenan de manera consecutiva. Estas fases sirven generalmente para estructurar un proyecto completo cuando se utilizan metodologías tradicionales, aunque en una metodología como Scrum cada *sprint* (cada desarrollo de un incremento a lo largo de dos o tres semanas), funciona como un mini-proyecto autónomo y de alguna manera es como si incluyera dentro estas mismas fases.

- Planificación: Consiste en organizar el trabajo, ver qué hay que conseguir, con qué equipo se cuenta, qué tareas habrá que abordar y estimar el tiempo que nos puede llevar hacerlo. Sobre todo al principio, requiere ponerse de acuerdo en los métodos y herramientas que se van a usar para llevarlo a cabo. En las metodologías tradicionales la planificación es una actividad de protección de las demás actividades estructurales del proyecto. Debe ser lo más extensa y precisa posible y se realiza al comienzo del proyecto, por ejemplo haciendo explícito un documento

con la Estructura de Descomposición del Trabajo (EDT). La EDT es una descomposición jerárquica, orientada al producto entregable y completo del trabajo de desarrollo que realizaremos. Una vez realizadas todas las tareas del EDT con éxito, podremos decir que el software está finalizado.A diferencia de las metodologías tradicionales, en Scrum los sprints se planifican individualmente, abordándolos de uno en uno, según se van realizando.

- Especificación: Se ocupa de detallar lo más posible la necesidad del cliente, definir con precisión los requisitos, funcionales o no, que se espera vaya a tener el software a desarrollar.

- Análisis: En esta fase hay que estudiar las alternativas a la hora de construir la aplicación software deseada, como por ejemplo las herramientas, los plugins y las técnicas que pueden ser de utilidad en el desarrollo del mismo. También es el momento de reflexionar y subdividir la tareas mayores en tareas más pequeñas. Además, por cada sprint debemos subdividir en tareas el objetivo final del mismo. Esta subdivisión queda reflejada en el ANEXO VI.

- Diseño: Durante esta fase se inventan soluciones al problema, desde el punto de vista de la arquitectura software y el diseño orientado a objetos, como ocurre en lenguajes visuales como Blueprints. Se tendrán que definir los interfaces que conectan unos componentes con otros, por ejemplo.

- Implementación: Es la fase de codificación del software, lo que permite ir teniendo, incremento a incremento, la aplicación construida y funcionando en el entorno de desarrollo al menos.

- Pruebas: En esta última fase se realiza un despliegue de la aplicación (se construye la “build” del videojuego) y al igual que se hace sobre el proyecto ejecutable desde el entorno de desarrollo, se da uso al software y se realizan una serie de ejercicios para comprobar que la funcionalidad, el rendimiento e incluso la experiencia resultante se ajusta a los objetivos que se buscaban.

Estas fases, aunque no aparezcan literalmente reflejadas así en los nombres de los *sprints*, han servido de base para crear nuestra plantilla de desarrollo de los mismos.

Al ser un proceso iterativo, recibimos retroalimentación de información y requisitos una vez empezado el desarrollo del proyecto por parte de nuestro director, así como la validación continua de nuestros avances. Gracias a esto tendremos varios ciclos de iteración completos y cortos para las distintas fases del proyecto. Por otra parte, seguimos un desarrollo incremental en el sentido de añadir nuevas funcionalidades, una a una, a nuestro software de acuerdo con las prioridades del cliente y del proyecto.

Combinando las iteraciones con el desarrollo incremental, obtenemos nuevas versiones cada pocas semanas, con mejoras respecto de las versiones anteriores y así hemos trabajado hasta llegar a la versión estable del software que es la que se entrega finalmente con esta memoria y queda a disposición del público.

Por último, es importante destacar que con esta metodología el equipo y el cliente se mantendrán especialmente motivados, ya que se aprecia un continuo avance del proyecto.

Roles de los miembros del equipo

A continuación, repartimos los roles definidos por defecto en la metodología Scrum entre los miembros de nuestro equipo. Al ser un proyecto que no es comercializado, hay ciertos aspecto definidos en los roles que no tienen representación en nuestra organización.

Product Owner

El director Federico Peinado y el co-director Nahum Álvarez: Son los principales interesados, junto con los alumnos, de que el proyecto salga adelante. Su principal responsabilidad es la de garantizar la viabilidad del mismo y guiar todo el proceso. Aunque no fuese posible acceder libremente al material del Departamento, nuestro director se ha preocupado de conseguir un dispositivo de RV aunque fuese a través de su laboratorio de invetigación. Con él hemos fijado los requisitos del proyecto y registrado dichos cambios en actas y en el propio backlog para hacer los sprints.

Scrum master

Alejandro Blázquez: Ha sido el responsable del proceso Scrum. Encargado de comprobar que se cumplen las iteraciones de forma correcta definidas en la planificación. También es el encargado de gestionar las reuniones internas de equipo y con el product owner.

Team

Alejandro Blázquez, Carlos Casado y Juan Antonio Palacios: Son el conjunto de desarrolladores. Cada uno de ellos será el responsable del desarrollo de un nivel concreto, así como de generar la documentación asociada al mismo. A pesar de que cada Sprint tiene un desarrollador principal, el resto del equipo aporta ideas y ayuda al principal. Todas las aportaciones, modificaciones y resolución de impedimentos que surgen en cualquier Sprint, tienen capacidad como equipo de autogestionarse para poder solventarlos.

Herramientas utilizadas

A continuación dejamos una lista detallada de todas las herramientas que utilizaremos para el correcto desarrollo de nuestro proyecto, ordenadas por orden de importancia o nivel de utilización.

Unreal Engine: Entorno para el desarrollo de videojuegos creado por la compañía Epic Games. Utilizaremos este programa, concretamente su versión 4, para desarrollar nuestro videojuego.

HTC Vive: Visor de RV desarrollado por HTC y Valve, con controladores manuales especializados. Con este dispositivo podremos sentirnos dentro del mundo virtual de nuestro videojuego creado en Unreal Engine. Los controladores de mano nos permitirá

utilizar la última tecnología en seguimiento de movimiento. Con estos controladores podremos coger objetos y realizar acciones básicas dentro de nuestro videojuego[21].

Trello: Herramienta gratuita de gestor de proyectos online. Esta herramienta la utilizaremos para la organización interna del proyecto y compartir el estado de las tareas asignadas a cada uno de los miembros (tarjetas). También nos permite ponernos alarmas o citas en el calendario para llevar al día las entregas de cada uno de los sprints del proyecto (Página oficial de Trello. (n.d.)).

Google Drive: Servicio de almacenamiento de datos en la nube que ha resultado vital para el desarrollo del proyecto. Lo utilizamos para llevar un repositorio común donde compartiremos información generada durante el desarrollo del proyecto y como copia de seguridad de todos los datos e información redactada para está memoria. También y como modo provisional, la utilizaremos como repositorio común para descargarnos las versiones del videojuego. Debido al tamaño limitado de datos que se pueden almacenar se buscará otro repositorio para guardar las versiones del videojuego (Página oficial Google Drive. (n.d.)).

Google Docs. El editor de textos utilizado para trabajar en la documentación del proyecto y en la redacción de la memoria final. También permite integrar figuras con composiciones de imágenes realizadas en Google Drawing.

Mensajería instantánea en línea (Slack y WhatsApp) y correo electrónico: La mensajería instantánea nos permitirá estar en contacto con nuestros *product owners*, así como con el resto de los compañeros del equipo. Nos permitirá recibir notificaciones importantes en tiempo real, como pudieran ser eventos interesantes, información adicional para nuestra memoria o estado de las tareas de nuestros compañeros (Página oficial Slack. (n.d.); Página oficial WhatsApp. (n.d.)).

Skype: Aprovechando las ventajas de la tecnología, utilizaremos Skype para realizar reuniones rápidas de forma remota. Nos permitirá realizar reuniones prácticamente a diario en la que compartiremos avances y aclararemos objetivos a corto medio y largo plazo. Gracias a esto en caso de que el equipo no se pueda juntar de manera personal, se podrá realizar la reunión de manera virtual sin necesidad de modificar planificación. Además, la funcionalidad de compartir pantalla nos permite colaborar en tiempo real sin necesidad de reunirnos físicamente (Página oficial Skype. (n.d.)).

Blender: Es una herramienta multiplataforma, dedicada especialmente al modelado, iluminación, renderizado, animación y creación de gráficos tridimensionales. Haremos pequeñas incursiones en este programa para modelar algunos aspectos puntuales de objetos y decoración de nuestro videojuego. (Página oficial Blender. (n.d.))

Excel: Software que permite realizar tareas contables y de cálculo gracias a sus funciones, desarrolladas específicamente para ayudar a crear y trabajar con las llamadas hojas de

[21] Para más información sobre UE4, HTC Vive y sus controladores se pueden consultar las secciones y anexos pertinentes.

cálculo. También nos permitirá crear gráficos y estadísticas para tener una representación más visual de la información. (Página oficial Windows Office (Excel). (n.d.))

Open Broadcaster Software: Utilizaremos este grabador de videos para poder crear y enseñar el trabajo en reuniones en las que no dispongamos del equipo necesario para realizar una prueba física. Guardaremos cada grabación en nuestro drive como bitácora audiovisual de los avances. Estos vídeos serán incluidos en presentaciones posteriores para mostrar el avance del trabajo.

YouTube: Portal de Internet para la subida y visualización de videos. Utilizaremos esta plataforma para subir breves videos con capturas del gameplay de nuestro videojuego y así ir dando a conocer nuestro trabajo poco a poco tanto al público como a nuestro propio director. (Página oficial Youtube. (n.d.))

Salas de trabajo: Para una mayor facilidad a la hora de desarrollar el trabajo de forma conjunta, se realizará la reserva de espacios dentro de la Facultad de Informática de la Universidad Complutense de Madrid. Lo incluimos en esta sección de herramientas, no sólo porque para este tipo de metodología de desarrollo es vital disponer de un espacio donde realizar las reuniones de inicio y fin de los sprints y de equipo, sino porque la RV requiere un espacio considerable donde trabajar. En este último caso lo cierto es que el espacio habilitado para el uso de RV fue en casa de uno de los integrantes del grupo, por no disponer de dicho espacio en la Facultad.

Audacity: Editor de audio gratuito. Permite grabar sonidos, reproducir sonidos, importar y exportar archivos WAV, AIFF, y MP3, y más. Lo utilizaremos para editar a nuestro gusto los sonidos que se reproducen durante el juego. (Página oficial Audacity. (n.d.))

Photoshop: Editor de gráficos rasterizados desarrollado por Adobe Systems Incorporated. Usado principalmente para el retoque de fotografías y gráficos. Lo utilizaremos para editar las imágenes del videojuego a nuestro gusto. (Página oficial Photoshop. (n.d.))

4. Planificación y especificación

Es necesario planificar y especificar lo más formalmente posible el desarrollo de nuestro proyecto antes de poder comenzar con las fases de análisis y diseño.

Incluimos en este apartado las fases de preparación del proyecto, partiendo desde el punto en el que comenzamos a planificar por sprints, utilizando la metodología Scrum, hasta que documentamos cómo deseamos que sea el juego. Todo lo trabajado y acordado previamente a la adopción de la metodología ágil, quedó reflejado en las actas de reunión que pueden encontrarse en las actas del ANEXO III

4.1. Lluvia de ideas inicial

Una vez decidimos iniciar el proyecto de desarrollo de un videojuego desde cero partiendo de una idea propia, era el momento de buscar cual debía ser esa idea. Como ya hemos explicado en el estado de la técnica, el género de terror parece funcionar realmente bien en este tipo de experiencias, y por eso decidimos que ésta sería la temática del juego.

Por su modularidad nos llamó la atención la saga de películas **SAW**. En esta saga un asesino en serie obliga a sus víctimas a superar un puzzle que pone en peligro su vida, suponiendo la muerte en caso de fracasar. Decidimos tomar esta idea y hacerla nuestra, diseñando puzzles que permitieran al usuario realizar distintas interacciones con los controladores manuales dotados de seguimiento de movimiento.

A pesar de que la idea nos gustaba, pronto nos dimos cuenta de que una de las características de las películas de SAW es que las víctimas suelen encontrarse atadas, lo que impide el movimiento, con lo que tuvimos que seguir depurando la idea. Por esto, y para favorecer distintos estados de ánimo durante las interacciones, llegamos también a la conclusión de que deberíamos introducir un nivel de “calma”, una especie de *hub*, que conectara los distintos niveles de “estrés” (puzles) del juego.

Así pues, nuestro juego se encontraría dividido por los niveles principales, en los que la interacción debería ser rápida y precisa, que se conectarían entre sí por un nivel diferente, más tranquilo y pausado en el que el jugador podría tomarse su tiempo para interactuar con el entorno de una manera más calmada. Esto nos permitirá experimentar con distintos tipos de ritmo de juego.

Finalmente, para simplificar aún más el desarrollo del juego y de los escenarios, tomamos la decisión de que toda la acción tuviera lugar dentro de una misma celda, situada en un hospital psiquiátrico y de la que el jugador no podrá escapar en ningún momento. El hilo conductor de la historia será el de un medicamento experimental que produce alucinaciones, siendo estas alucinaciones los distintos puzzles a resolver. Esto también nos permitirá jugar con los volúmenes de la ya citada celda, dándonos cierta libertad a la hora de diseñar niveles con cierta diversidad de interacciones.

Para la historia que queremos contar tomamos referencias de la película **Shutter Island**, cuyo argumento tiene lugar dentro de un hospital psiquiátrico. Esta historia se contará

presumiblemente por medio de notas que el jugador deberá leer y que se le presentarán en el citado "periodo de calma" entre niveles.

Como toque final, buscamos inspiración para los distintos niveles en fobias muy conocidas, como son la aracnofobia, la claustrofobia y la acrofobia.

4.2. EDT

Este documento con la EDT pretende servir como guía inicial del desarrollo del proyecto. En él se reflejan las tareas identificadas inicialmente para poder planificar su ejecución de la mejor manera posible, aunque al estar utilizando una metodología ágil es normal que a lo largo del desarrollo se tengan que añadir algunas más.

Debido a su extensión, el EDT utilizado se presenta como archivo adjunto a este documento, en su versión digital en DVD.

4.3. Documento de diseño y biblia del juego

Reproducimos a continuación nuestro documento de diseño del juego que hemos titulado "Phobia". En él plasmamos todas las ideas iniciales intentando formalizarlas, por lo que servirá a modo de **especificación de requisitos** del proyecto.

<table>
<tr><th colspan="2">PHOBIA - Resumen</th></tr>
<tr><td>Géneros
● Supervivencia y Horror en 1ª persona</td><td>Modos
● Historia Monojugador, solamente</td></tr>
<tr><td>Público objetivo
● Adolescentes y adultos jóvenes de 13 a 35 años con cierta capacidad adquisitiva
● Fans de los juegos de terror y la RV
● Aficionados al terror y a la temática de las fobias y los problemas mentales</td><td>Plataformas
● PC Windows + HTC Vive (escala de habitación), también posible con Oculus Rift + Touch</td></tr>
<tr><td>Valores de producción
● Interior de una celda con elementos interactivos pensados para usar los controladores manuales
● Personaje con trasfondo que invita al jugador a querer conocer su historia
● Momentos impactantes en cada nivel</td><td>Hitos
● Idea original: 2016
● Dinámica y mecánica:
● Arachnophobia
● Claustrophobia
● Acrophobia
● Primera versión: junio 2017</td></tr>
</table>

4.3.1. Descripción

Phobia es un videojuego en RV a escala de habitación donde nos encarnamos en un misterioso individuo encerrado en una aterradora prisión. El protagonista, que parece sufrir intensas fobias y alucinaciones, es obligado por parte del equipo médico a realizar espeluznantes pruebas especialmente diseñadas para agredir su mente. El jugador tendrá un espacio reducido a su disposición que podrá explorar en busca de pistas que le ayuden

a comprender qué está pasando. Esta dinámica se alterna con la superación de pruebas rápidas e intensas donde está en peligro la vida del prisionero. Estas pruebas son casi siempre resultado de las alucinaciones del propio paciente, a quien los sanitarios a veces suministran objetos en función de la emoción que desean potenciar en ese momento y el tipo de acciones que quieren que este realice.

Para la producción se utilizarán recursos gratuitos (o muy económicos) como son el escenario 3D, los objetos interactuables, la música y los efectos sonoros del juego. Estos elementos se buscarán en bibliotecas de audio libres para el caso del audio o en las propias tiendas on-line de Unity y Unreal para el caso de los objetos y el escenario 3D.

La clave del juego es plantear tanto pruebas rápidas, puzles con algún elemento narrativo, que exploten el uso de la RV y los controladores de mano, como por ejemplo tener que librarse del acoso de una plaga de arañas al mismo tiempo que se elimina la fuente de donde están surgiendo. Los mismos mecanismos se podrían aprovechar en futuras ampliaciones para otras cosas, como permitir que el protagonista rompa algo en la pared de su celda y encuentre mensajes del anterior prisionero encerrado allí u objetos especiales. El juego irá intercalando pruebas de habilidad en entornos de tensión con otras que permitan al jugador pensar con calma. La idea es explorar el uso de los controladores manuales de VR tanto para tareas que requieran agudeza visual, sincronización y rapidez, como para otras más relajadas que permitan interactuar con el entorno de una manera más pausada.

4.3.2. Jugabilidad

Estética

La temática se presta a la utilización de una estética oscura de principio del siglo XX, en un mundo donde las enfermedades mentales no son tratadas con métodos científicos y los pacientes son vistos como meros sujetos disponibles para toda clase de pruebas. En este universo de juego se realizan experimentos de lesa humanidad bajo el pretexto de que no hay otra forma de curar a los enfermos. Ejemplos de estas prácticas, basadas en práctica reales de la medicina de la época, son la lobotomía y el electroshock.

Las **emociones básicas** que se intentan explotar en la experiencia son:

- **Desorientación**: El protagonista no sabe muy bien dónde está, sólo entiende que debe salir de ahí.
- **Miedo**
- **Excitación**: al tener que resolver algunos puzzles de manera atropellada, bajo mucha presión y en un tiempo límite
- **Tristeza**: al descubrir los asesinatos y la oscura historia del lugar.

Los **instintos** que entran en juego son:

- **Curiosidad**: en el tiempo pasado en la celda, el protagonista necesita saber por qué le están torturando y quién estuvo en su situación antes que él.
- **Supervivencia**: durante las pruebas el objetivo principal será sobrevivir.
- **Huida**: el protagonista estará en todo momento intentando escapar.

- **Equilibrio** y sincronización: habrá pruebas que requieran que realicemos acciones sincronizadas con ambas manos.

Dinámica

El paciente está encerrado en una celda de aislamiento del manicomio. Debido a las drogas, sufre alucinaciones por las que el espacio se transforma. Esto es, todos los niveles y sus respectivos puzles tendrán lugar en la celda, cuyo aspecto y funcionalidad irá modificándose según las necesidades del desafío a resolver.

Una vez resuelto el desafío, el paciente puede perder el conocimiento y despertar en la celda sin los efectos de las drogas. Diferenciamos así los momentos de estrés (bajo los efectos de las drogas) de los momentos de calma. Tras la resolución del puzzle, el jugador tendrá todo el tiempo que quiera para explorar la celda. Este tiempo finaliza cuando el personaje decide tomarse unas pastillas que se dejarán a su alcance y que le dan acceso al siguiente nivel. Así hasta completar el juego.

Mecánica

Entendemos por **mecánica** el conjunto de acciones que el jugador debe realizar para la consecución de sus objetivos. En el caso de nuestro juego, estas acciones variarán en cada nivel, pues buscamos explorar las diferentes posibilidades de los controladores.

4.3.3. Contenido

Historia

El paciente número 52 está interno en un oscuro manicomio de comienzos del siglo XX. Dentro se realizan perturbadores experimentos en la búsqueda del conocimiento de la mente humana, de lo que se deduce que 52 padece algún tipo de enfermedad mental. Las pruebas a las que es sometido son cada vez más duras, llegando incluso a poner en peligro su vida en más de una ocasión.

La verdad tras número 52 es que se trata de un conocido asesino de la época, a quien el juez ha declarado demente e internado en una institución de alta seguridad. Frustrados por no poder conectar con los sentimientos del psicópata, los sanitarios deciden enfrentarle a sus mayores miedos, esperando con esto que por fin entienda el sufrimiento que ha causado a sus víctimas y reacciones de una vez. Para ello realizan una puesta en escena de todas las torturas ideadas por 52, drogando al paciente en numerosas ocasiones para desorientarlo y potenciar un sentimiento de indefensión.

Objetos y lugares

Celda

Entendemos por celda el lugar en el que se encuentra el jugador en los momentos de calma. Esto es, la celda básica sin modificaciones para puzzles.

- Cama: Será el punto de inicio del estado de calma una vez finalizado cada puzzle.
- Retrete: Decoración y se utilizará como elemento fundamental en algunos niveles.
- Puerta: Diseñada con una especie de cajonera por la que los médicos facilitan las drogas al paciente.
- Drogas: Se facilitan a través de la puerta y pueden ser utilizadas como nexo de unión entre la zona de calma y la de estrés del próximo nivel.
- Mesa y cajones: Utilizada para proporcionar y colocar allí algunos objetos necesarios para la resolución de puzzles.

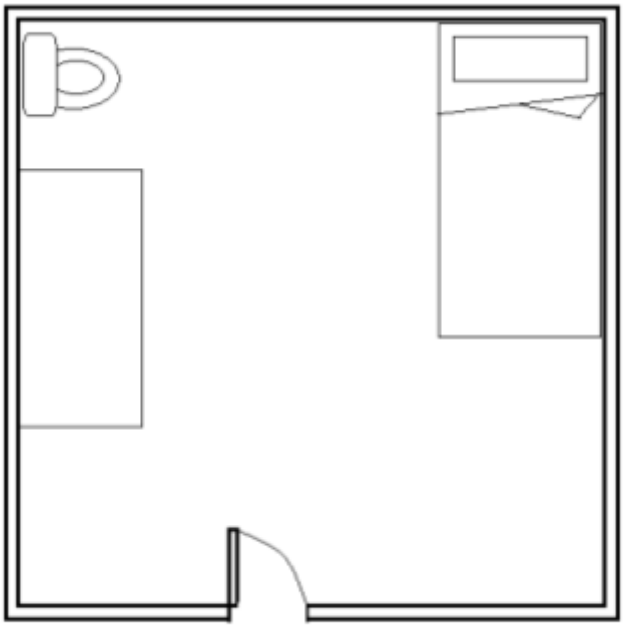

Figura 4.2: Diseño inicial de la celda

Nivel 1: Arachnophobia

En este nivel se experimenta con la aracnofobia, fobia a las arañas. Durante el primer nivel comenzarán a salir arañas del inodoro sin parar. El objetivo del puzzle es conseguir que dejen de salir rompiendo el mismo con una tubería metálica.

- Tubería: Utilizado para romper el retrete.

Nivel 2: Claustrophobia

En este nivel se experimenta con la claustrofobia, fobia a los espacios cerrados, y al poco de empezar veremos cómo las paredes comienzan a cerrarse. El jugador deberá encontrar la manera de pararlas antes de que le aplasten por completo. Esto se hará por medio de un elemento de la habitación robusto, el propio catre o cama donde duerme el prisionero.

- Cama: Elemento lo suficientemente grande como para bloquear el avance de las paredes haciendo de travesaño entre una y otra.

Nivel 3: Acrophobia

En este nivel se experimenta con la acrofobia, fobia a las alturas, y en general con la sensación de vértigo. La celda se muestra aparentemente normal hasta que el suelo se viene abajo, derrumbándose casi por completo, y el jugador debe buscar refugio antes de caerse. El objetivo de este puzzle es escalar por la pared hacia un lugar seguro desde donde poder escapar.

- Cuerda: Se puede usar para escalar y llegar hasta un punto de salida situado fuera de la celda.

Personajes

El diseño de personajes se presenta como algo secundario en este proyecto, pues el punto fuerte de la experiencia son los diferentes puzzles y la interacción con el entorno. No obstante, se han esbozado unos personajes básicos por si fuera posible lograr una iteración centrada únicamente en la narración.

El paciente número 52

Será el personaje interpretado por el jugador y su papel será el de exploración del entorno. Debido a las drogas, no recuerda nada de su vida antes de ingresar en el manicomio. No entiende muy bien por qué le están torturando constantemente.

Los médicos

Representados cuando sea estrictamente necesario mediante una voz en off que dará órdenes al paciente. En un futuro su papel en el juego será el de narradores, dando pistas al jugador acerca de la historia por medio de indirectas, aunque al principio simplemente llaman a la puerta para informar de que ya están listas las pastillas que debe tomarse el prisionero. Puede que le proporcionen al paciente recortes de periódico narrando sus asesinatos, para potenciar la conexión entre los asesinatos reales y la puesta en escena que está sufriendo, y de paso dándole pistas para poder superar los puzles.

El paciente número 50

En un futuro se quiere contar con la narrativa de que existió un paciente anterior al número 52, el número 50, que dejó mensajes en la celda. Durante la estancia en la misma, el jugador podría desentrañar estos mensajes mediante puzzles sencillos que se irían activando a medida que avance la trama. Una nota detrás de un azulejo levantado, un bote de pastillas escondido en una Biblia y cosas por el estilo… a implementarse en futuras versiones del juego.

Referencias

Para el diseño tanto de los puzzles como del argumento a contar nos hemos basado en varios videojuegos y películas:

Videojuegos

The Kitchen: Resident Evil: Todo el argumento sucede en un mismo sitio, al igual que nuestro juego.

A Chair in a Room: Historia similar. Un sujeto encerrado en una celda que debe obedecer las órdenes de sus captores.

Please, don't touch anything: Dinámica similar a la de la celda. Nivel de calma donde, a través de otros objetos, se accede a los diferentes niveles.

Películas

Saw: Saga reconocida del género de terror. En ella un asesino somete a sus víctimas a diferentes puzzles que deben resolver. Si lo consiguen, salen vivos de la prueba.

La habitación Fermat: Cuatro matemáticos, que no se conocen entre sí, son invitados por un misterioso anfitrión con el pretexto de resolver un gran enigma. Pronto descubren que se encuentran en una sala que empieza a menguar y que corren el riesgo de morir aplastados entre sus paredes.

Shutter Island: Un inspector de policía investiga un extraño crimen en un hospital psiquiátrico. Finalmente descubre que no todo es lo que parece y acaba recordando su propia historia olvidada.

4.3.4. Gestión de riesgos

En cualquier proyecto software pueden ocurrir cosas que escapan a nuestro control. Para evitar que estos sucesos pongan en riesgo el devenir del proyecto será necesario hacer una gestión de riesgos. Gracias a está gestión podremos tener una actitud proactiva frente a cualquier problema. A continuación podemos ver los campos dentro del proyecto que a priori, nos pueden causar mayores problemas.

Riesgos del proyecto:

- Planificación : se puede necesitar más tiempo
- Personal: posibles carencias en ciertos ámbitos durante el desarrollo
- Recursos: falta de algún tipo de recurso a lo largo del tiempo
- Requisitos: cambio de condiciones durante el desarrollo
- Implementación: se necesita más tiempo para implementar
- Verificación: se necesita más tiempo para realizar pruebas
- Incertidumbre técnica: se necesita mayores conocimientos técnicos
- Incertidumbre tecnológica: se necesita mayores conocimientos de la tecnología utilizada.

A continuación se adjunta una tabla de riesgos del proyecto junto con la probabilidad de que ocurra y el impacto que tendría sobre el mismo (Alto, medio o bajo).

Evento	**Nombre**	**Probabilidad**	**Impacto**
Planificación	Incumpliminento planificación inicial	Alta	Variable
	Incumpliminento planificación de sprints	Alta	Variable
Personal	Exámenes	Alta	Medio
	Falta miembro del grupo	Baja	Alto
	Falta de tiempo	Baja	Bajo
Recursos	Falta de ordenador	Media	Alto
	Falta de visor	Media	Alto
	Falta de controladores	Media	Alto
	Falta de espacio para desarrollar	Media	Medio
Requisitos	Cambio hitos de sprint	Alta	Bajo
Implementación	Falta tiempo implementar el total	Media	Medio
Verificación	No verificación de sprints	Baja	Alto
	No verificación producto final	Baja	Alto
Incertidumbre técnica	Proceso de desarrollo Software	Baja	Bajo
Incertidumbre tecnológica	Desconocimiento de herramientas	Baja	Alto
	Desconocimiento de visores VR	Baja	Alto
	Desconocimiento controladores.	Baja	Alto
	Desconocimiento artísticos	Media	Media

Tabla 4.1: Tabla de riesgos

Una vez conocidos los riesgos,la probabilidad y el impacto sobre el proyecto,es necesario tener un plan de contingencia y un plan de mitigación. En el siguiente apartado se explicará en qué consistirán dichos planes.

Plan de mitigación y contingencia

A continuación asignaremos a cada uno de los riesgos a tener en cuenta, la forma de mitigarlos y un posible plan de contingencia en aquellos casos que sean necesarios. Lo dividiremos para mayor sencillez en función de los eventos registrados en la tabla anterior.

1 Planificación (probabilidad)

- Incumplimiento planificación inicial : La probabilidad del incumplimiento de la planificación inicial es muy alta debido al desconocimiento tanto de las herramientas, objetivos y otros conceptos. Para mitigar este problema utilizaremos una metodología ágil debido a la versatilidad que nos ofrece este modelo de desarrollo Software.Gracias a esto podemos asegurarnos que el impacto recibido sea asumible.

- Incumpliminento planificación de sprints : Como con la planificación inicial la probabilidad de que ocurra es realmente elevada, pero gracias a los mecanismos propuestos dentro de metodología Scrum los posibles problemas se identificarán enseguida y se tomarán medidas para minimizar su impacto

2 Personal

- Exámenes : Debido a que no hemos acabado nuestra formación académica, habrá etapas durante el desarrollo que el proyecto estará parcial o totalmente parado. A día de hoy desconocemos la fecha de los exámenes en los que tendremos que participar, pero conociendo las asignaturas y las épocas de exámenes respecto de otros años intentaremos realizar una planificación en base a estos datos. En las reuniones de equipo se pactaran el desarrollo mínimo de actividades que cada uno de los miembros tiene que realizar durante estas fechas, haciendo un reparto equiparado de tareas entre exámenes y proyecto.

- Falta algún miembro del grupo : La probabilidad de que esto ocurra es prácticamente nula. En el hipotético caso que esto ocurriera deberíamos descartar objetivos secundarios y centrarnos solo en los primarios. El impacto de esta casuística dentro del equipo es muy alto.

3 Recursos

- Falta de ordenador: La posibilidad de que no tengamos un PC VR ready es un factor a tener en cuenta. En caso de no disponer de un PC en casa nuestro tutor nos ofreció el suyo que ya era totalmente funcional. Gracias a esto podemos mitigar el efecto. Pero en cualquiera de los casos no disponer del PC es un riesgo muy alto y a tener en cuenta.

- Falta de visor : No disponer de un visor nos impediría poder desarrollar el software. Para mitigar este efecto intentaremos hacer un software que funcione para dos dispositivos de realidad virtual (HTC VIVE y Oculus Rift) con esto nos

aseguraremos poder disponer de uno en todo momento. Como en el punto anterior el impacto que causaría no disponer de ello es muy alto.

- Falta controladores : Misma idea que en el punto anterior. Mitigar daño a la mitad pudiendo utilizar ambos tipos de controladores (HTC VIVE y Oculus).

- Falta de espacio para desarrollar : No disponer de un espacio lo suficientemente grande y equipado para desarrollar es un riesgo que nos preocupa. Actualmente se podría disponer de un pequeño hueco preparado para este propósito en un despacho. Para evitar este riesgo y si dispusieramos de un equipo de manera continua podríamos instalarlo en un lugar externo de libre acceso para mitigar totalmente el riesgo.

4 Requisitos

- Cambios hitos del sprint : La probabilidad de que esto ocurra es muy alta. Gracias a la metodología Scrum el riesgo es totalmente asumible.

5 Implementación

- Falta de tiempo implementación total: Falta de tiempo para implementar la idea total del documento de diseño. Es un factor que nos puede poner en problemas dividiremos los objetivos en principales y secundarias, siendo únicamente los principales los imprescindibles.

6 Verificación

- No verificación de sprint : La verificación de sprint es un paso fundamental por lo que no contemplamos no realizarlo. En caso de que no se pudiese realizar en un sprint, la parte no verificada pasaría a un nuevo sprint como tarea pendiente y será en este nuevo sprint donde se realizará. No se contempla entregar Software que no se valide previamente.

- No verificación del producto final : No se plantea entregar que no haya sido previamente probado. Para evitar esta situación haremos verificaciones constantes en cada sprint. Con esto nos aseguraremos que a pesar de tener menos software en la entrega total, será perfectamente funcional.

7 Incertidumbre técnica

- Proceso de desarrollo Software: Riesgo asumible ya que hemos sido formados en el desarrollo de software mediante metodologías ágil.

8 Incertidumbre tecnológica

- Desconocimiento de Herramientas : La herramienta que no hemos utilizado nunca y puede ser un riesgo no comprenderla es Unreal. En caso de que esto ocurriera sería un problema muy grave. Para mitigar esta situación podríamos cambiar de motor de desarrollo a Unity que ya conocemos de otras actividades. El

resto de herramientas no nos preocupan ya que las conocemos y hemos utilizado previamente.

- Desconocimiento de visores VR : Es una tecnología prácticamente nueva. A priori desconocemos pero nuestro tutor nos ha facilitado información que nos permitirá aprenderla. en caso de encontrar problemas existen cursos y compañeros que nos permitirán aprender de una manera rápida. Debido a esto la probabilidad de que ocurra es prácticamente imposible.

- Desconocimiento controladores: Misma política que con el punto anterior de visores.

- Desconocimiento de arte: Al no ser artista ni disponer de diseñadores gráfico es un riesgo a tener en cuenta. En caso de no encontrar ninguna figura de este tipo utilizaremos los materiales gratuitos que están a nuestra alcance.

5. Análisis, diseño e implementación

En este capítulo detallamos las fases de análisis, diseño e implementación del proceso de desarrollo que se ha seguido.

Se incluyen como anexo todas las actas de reuniones de Scrum, así como los documentos generados por el equipo de desarrollo durante cada una de los *sprints* (ANEXO VI: Diario de desarrollo) usando una misma plantilla acordada previamente por el equipo (ANEXO V). Esta sección es un relato ordenado de los aspectos más importantes del desarrollo según fueron ocurriendo, manteniendo la división en sprints. Al final de cada uno de estos sprints se comprobaba que el software desarrollado funcionaba de la manera esperada, por lo que pudimos llevar a cabo un progreso incremental desde el inicio del proyecto hasta la finalización del mismo. El resultado de estas pruebas internas se encuentra en el citado ANEXO VI.

5.1. Sprint 1: Mecanismos básicos

Durante la primera iteración del desarrollo se han creado los blueprints y las interacciones básicas para el uso de los visores y los controladores de RV durante la mayor parte del juego. Para ello, se ha creado un nivel como caja de pruebas (o sandbox) que nos ha servido para experimentar con la tecnología, con sus posibilidades y limitaciones.

5.1.1. Plugin utilizado

Actualmente existen dos plugins de uso generalizado para RV en Unreal Engine 4:

SteamVR

Es el plugin por defecto que proporciona Unreal Engine al utilizar la plantilla básica para moverse y realizar interacciones en RV. En un principio, el desarrollo de nuestro juego utilizaba este plugin como base, pero la dificultad de implementar colisiones en objetos agarrables contra objetos estáticos sin físicas era una limitación palpable, sobre todo teniendo en cuenta que dificultaba la realización de los niveles que ya teníamos diseñados.

Por el contrario, la creación de objetos agarrables era significativamente más rápida y sencilla.

VRExpansion Plugin

Plugin modificado por la comunidad de desarrolladores de UE4 centrando en su uso para OpenVR[22] y sobre todo de uso concreto para HTC Vive, que añade nuevas características y componentes y que soluciona nuestros problemas con objetos agarrables y sus colisiones.

[22] SDK desarrollado por Valve para facilitar la creación de software en RV por parte de desarrolladores independientes.

Cabe destacar como ventaja la implementación de un nuevo *PawnCharacter*[23] que añade realismo mediante colisiones con el entorno, nuevos tipos de movimiento además del teletransporte y un nuevo tipo de interacción, que es la escalada de cualquier objeto no agarrable.

Su principal desventaja es el ligero aumento de la complejidad y del tiempo de creación de objetos interactuables. A pesar de ello este será el plugin que utilizaremos en nuestro proyecto.

Vive_PawnCharacter

Vive_PawnCharacter será el pawn de nuestro juego e incluirá todos los elementos necesarios para obtener información e interactuar con el entorno.

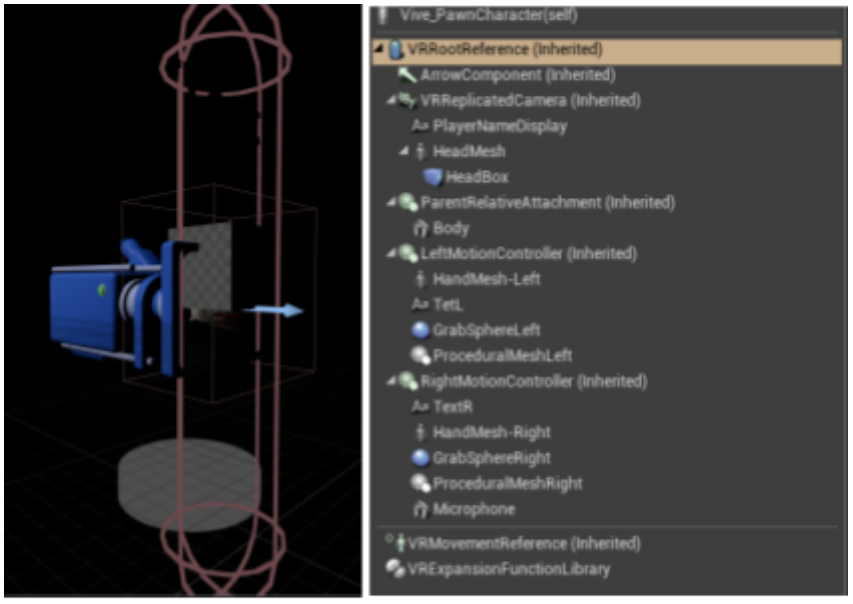

Figura 5.1: Diseño del *pawn character* que utilizaremos. La cápsula representa la propia presencia del jugador, mientras que el disco inferior representa la cintura del jugador. La cámara indica al motor dónde debe renderizar la imagen y el cubo sirve para representar la cabeza y simular las colisiones con la misma.

VRReplicatedCamera

Cámara cuyas imágenes recogidas se verán reflejadas en el visor del jugador. Lleva ligado un cubo con colisiones activadas que simula la cabeza del jugador, de tal manera que durante la partida éste no puede atravesar paredes u objetos con colisiones, impidiendo así la salida de la inmersión.

A la herencia inicial se ha añadido un *collider*[24] a la cabeza con el fin de analizar su superposición con otros actores.

[23] Se llama ***pawn character*** a la clase que representa a un jugador dentro del entorno del juego. Dentro de esta clase pueden incluirse varios elementos que permitirán la configuración del mismo y cómo debe reaccionar a la entrada recibida desde los controladores.

[24] Elemento encargado de detectar las colisiones de un objeto determinado con cualquier otro.

Body

Simula mediante colisiones el cuerpo del jugador con el mismo fin del *mesh*[25] utilizado en la cabeza.

LeftMotionController y RightMotionController

Contendrán los modelos y la lógica necesaria para implementar el uso de los controladores e incluye el modelo 3D de las manos y las esferas que determinarán la distancia a la que un objeto podrá ser agarrado.

5.1.2. Controladores: Movimiento e interacciones

Nuestro videojuego basa la experiencia de juego en el uso de los controladores y su interacción con el entorno. Así, la principal función de los controladores será la propia de unas manos, agarrar objetos y superficies. Para ello, el jugador, cuyas manos virtuales estarán abiertas en posición inicial, deberá apretar el gatillo si quiere realizar el gesto de prensado con toda la mano y soltarlo para volver abrir.

El movimiento del jugador en el espacio virtual se realiza mediante el movimiento físico del usuario en su estancia (para distancias pequeñas dentro del mundo virtual, menores en todo caso al espacio físico disponible) o bien mediante la teletransportación en el espacio virtual (si el usuario desea caminar en el mundo virtual una distancia mayor a la disponible en el espacio físico), apuntando con el controlador al lugar al que queremos movernos.

La variable a la que se accede para realizar las interacciones de agarre es del tipo referencia a Grip Motion Controller Component y permitirá tener acceso a la información del controlador, del objeto agarrado y ligar objetos al movimiento de la mano.

Agarrar un objeto: VRGrip Interface

Se trata de una interfaz desarrollada dentro del plugin VRExpansion que nos permite convertir cualquier objeto de la escena en agarrable por nuestros controladores. Basta con crear un Actor[26] que implemente dicha interfaz.

Esta interfaz nos permite controlar el comportamiento del objeto cuando intersecamos nuestras manos, sin colisiones, en un objeto agarrable y apretamos el gatillo del controlador, contrayendo nuestras manos y permitiéndonos agarrarlo. También podemos personalizar su comportamiento al movimiento, la respuesta al teletransporte cuando tenemos un objeto agarrado, la distancia de agarre cuando hay colisión, si permite un agarrar el objeto con las dos manos o sólo con una o el comportamiento del objeto cuando es soltado.

[25] Mesh: Cada uno de los objetos diseñados en Unreal Engine. Se diferencia de un Actor en que no contiene programación interna, mientras que el Actor puede definir un comportamiento programado.

[26] Actor: Cada uno de los elementos que componen una escena. Se diferencia de un Mesh en que tiene un comportamiento propio, siendo similar a una clase. Son la base de la Programación Orientada a Objetos utilizada en UE4.

El uso de los eventos de agarrado y soltado del objeto con el controlador es sencillo, aunque requiere del uso de una referencia al controlador en forma de variable dentro del Actor que representa a dicho objeto, así como una variable booleana de “agarrado”.

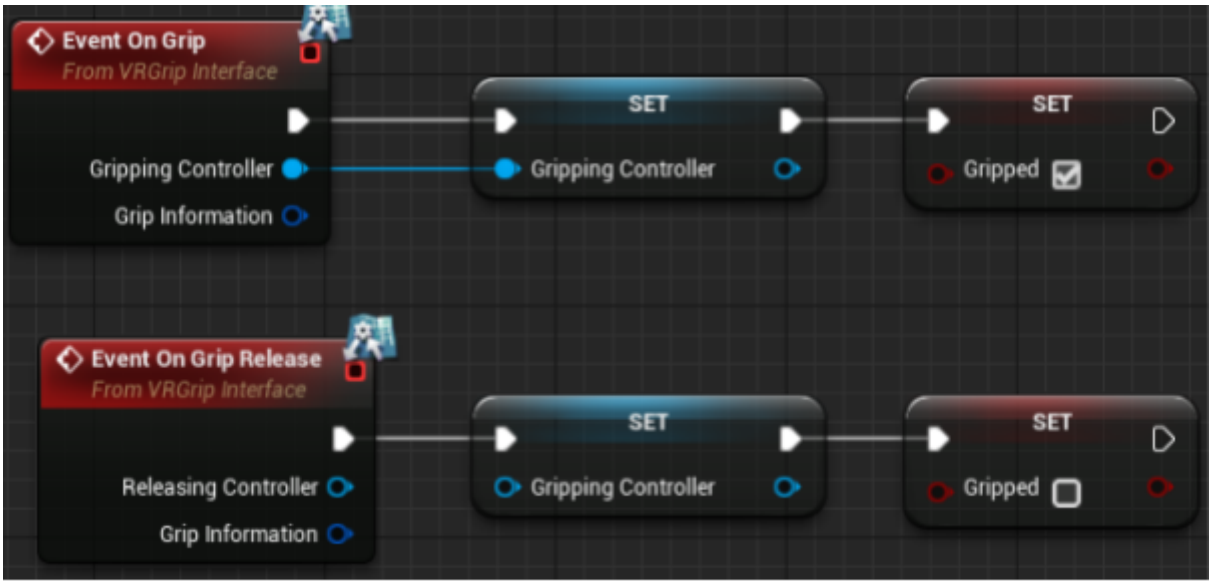

Figura 5.2: Blueprint de agarrado y soltado de objeto.

Movimiento del jugador por el entorno virtual

VRExpansion proporciona un amplio repertorio de modos de movimiento, desde el teletransporte, pasando por el movimiento natural de agitación de brazos al correr, o el desplazamiento mediante deslizamiento. Tras probar la experiencia con cada uno de ellos, optamos por implementar el teletransporte como movimiento predeterminado, ya que ha resultado ser el modo más cómodo y el que menos mareos y desorientación provocó en las pruebas realizadas. Este tipo de movimiento sacrifica en parte la inmersión, pero hace la navegación más agradable al usuario.

Para hacer posible el desplazamiento por el nivel ha de incluirse un objeto conocido como NavMeshBoundsVolume, clase estándar de Unreal Engine que proporciona los caminos permitidos para el usuario al colocarse sobre una superficie previamente indicada como navegable. Después, mediante el uso del controlador seleccionamos la superficie a la que nos queremos teletransportar al final de un arco que sale “disparado” de nuestras manos.

5.1.3. Nivel Sandbox o caja de arena

Para la implementación de las interacciones y el desarrollo de los assets necesarios hemos creado un espacio de prueba preliminar parecido a nuestro nivel principal y final. Este nivel de prueba servirá para desarrollar varias de los mecanismos que se incluirán finalmente en los diferentes niveles de nuestro juego.

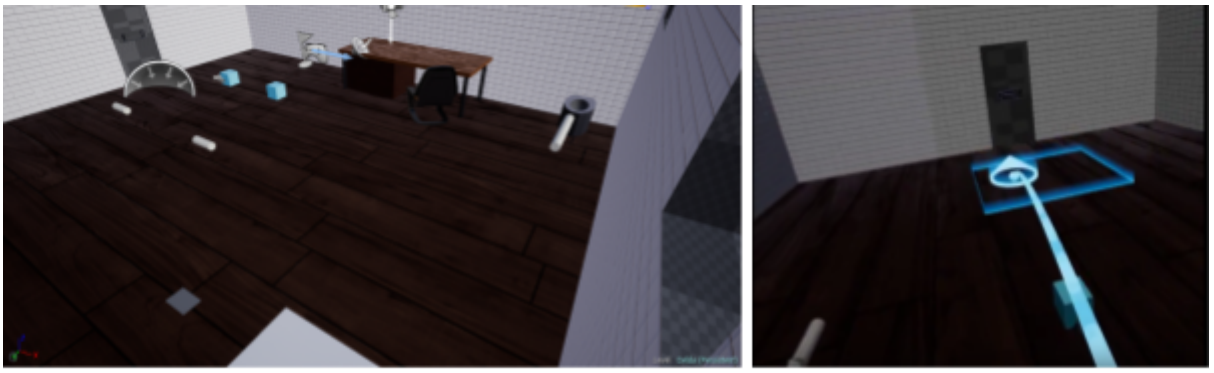

Figura 5.3: A la izquierda una vista general del nivel de desarrollo. A la derecha, prueba de teletransporte realizada dentro del mismo.

Golpeo y quiebra de elementos

Uno de los niveles, el denominado en fase de diseño como *Arachnophobia,* incluye un mecanismo muy característico consistente en romper un váter con un elemento que durante esta fase se encontraba todavía sin definir. Por ello optamos por diseñar un palo bastante rudimentario, así como una váter diseñado con cilindros.

El palo fue el primer objeto agarrable que diseñamos, teniendo que implementar la interfaz VRGrip que ya hemos citado (5.1.2.1. Agarrar un objeto: VRGrip Interface). La configuración de esta interfaz se harán extensible, a no ser que se necesite lo contrario, al resto de objetos agarrables del juego. Se reproduce esta configuración a continuación, a modo de guía:

TeleporterBehavior	Return Value: Teleport All Components.
SlotGripType	Return Value: Interactive Collision with Physics.
SimulateOnDrop	Return Value: true
ObjectType	Return Value: 0
IsInteractible	Return Value: true
isHeld	Input: Gripping Controller, Gripped Return Value: Teleport All Components
GripStiffness	Return Value: 1500
GripMovementReplicationType	Return Value: Force Client Side Movement
GripLateUpdateSetting	Return Value: Not when Colliding or Double Gripping
GripDamping	Return Value: 200
GripBreakDistance	Return Value: 100
GetInteractionSettings	N/A

FreeGripType	Return Value: Interactive Collision with Physics
DenyGripping	Return Value: false
ClosestSecondarySlotInRange	Return Value: Had Slot in range:false; Slot World Transform
ClosestPrimarySlotInRange	Return Value: Had Slot in range:false; Slot World Transform
CanHaveDoubleGrip	Return Value: false

Tabla 5.1: VRGrip Interface aplicada a nuestro proyecto.

En el caso del váter, debimos crear un Actor con un componente rompible (esto es, hereda de la clase *Destructible Mesh*, estándar de UE4). Mediante programación en *blueprint* conseguimos que este destructible reaccionara sólo a las colisiones con el palo implementado, y que recibiera daño en función de la velocidad con la que recibía el golpe.

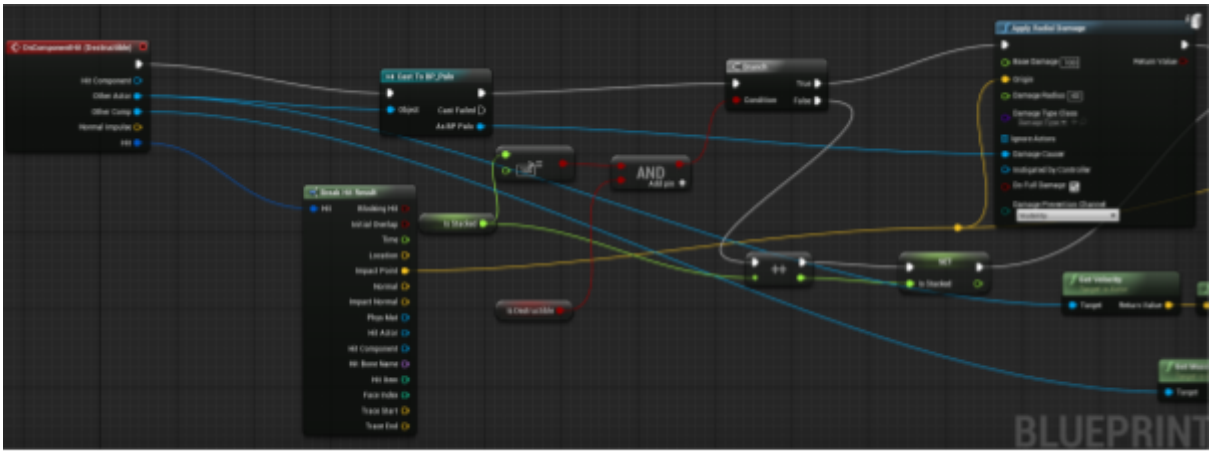

Figura 5.4: Comportamiento del elemento destructible del váter durante un evento de colisión.

Notas e interfaz de usuario

Por motivos narrativos, necesitamos desarrollar un sistema de notas adecuado para la nueva tecnología, contando con varias opciones comunes en este tipo de dispositivos, la implementación de un menú 3D es la más usada y recomendada, ya sea anclando el visionado de un actor en el centro de la visión del jugador o bien dándole entidad real dentro del universo del juego, como parte del propio mobiliario.

En nuestro caso se han desarrollado dos formas de imprimir una nota que podrían convenir a nuestros objetivos.

En la primera, el menú 3D aparece superpuesto a una nota anclada en la pared y se despliega si el usuario cierra la mano sobre ella. El resultado estético es adecuado, pero la interacción era poco intuitiva.

En la segunda y final solución al problema se consigue anclar el menú 3D a la mano que agarra la nota, en tiempo real. Así, cuando el usuario sujete la nota con las manos, sobre sus manos se desplegará un menú 3D con el texto necesario transcrito.

La interfaz hace uso de la herramienta de creación de widgets de UE4. Con ello crearemos nuestra interfaz de usuario y los botones necesarios para la interacción con el usuario.

Para la creación del menú 3D necesitamos añadir un elemento del tipo widget en un nuevo actor, referenciando a la clase del widget anteriormente creada.

Este actor será el que ligaremos al actor de la nota cogida por el usuario en tiempo de ejecución mediante el siguiente Blueprint.

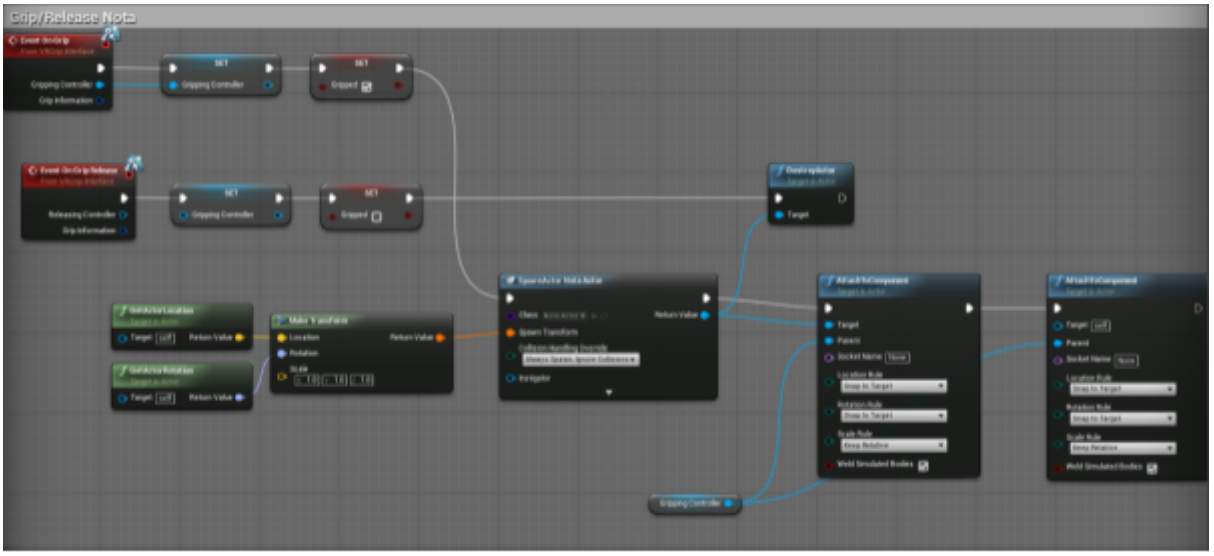

Figura 5.5: Blueprint para la aparición del widget una vez agarrada la nota.

En él, al agarrar la nota con la mano esta se vincula a la mano y generar el actor de la interfaz, debidamente rotado para la lectura cómoda de la nota por parte del usuario. Es posible que no siempre esté orientado de forma correcta y obligue al usuario a realizar una nueva sujeción de la nota. El actor correspondiente a la interfaz se destruye cuando se suelta la nota de la mano.

Es importante implementar la interface VRGripInterface en el actor de la interfaz y activar el flag en la función DenyGripping, ya que, si no, sería posible “escalar” la superficie de la nota transcrita, dando lugar a movimientos bruscos y antinaturales si se usa el gatillo del controlador sobre su superficie.

Puerta y cajón

Nuestro protagonista se encuentra cautivo en lo que parece una institución mental y su suministro periódico de pastillas ha de colocarse para el paciente en un lugar seguro para el personal y accesible para el paciente. Para ello, se ha creado una puerta metálica con una pequeña ventana de vidrio en la parte superior y una cavidad a media altura con un cajón que permite el paso de objetos desde el exterior.

Es necesario, para ello, la creación de un cajón interactuable con el usuario y los controladores mediante la creación de un objeto agarrable por el asa y deslizable sobre raíles virtuales con tope de apertura en el deslizamiento.

Para ello se ha creado un blueprint que contiene la puerta como raíz y del que heredan un cajón del tipo Grippable Static Mesh Component y un PhysicsConstraint cuya configuración nos permitirá realizar el deslizamiento del mismo.

El cajón incorpora implementada la interface VRGrip, a la que hay que modificar la activación de simulación de físicas cuando soltamos el cajón y asignamos límites en los ejes X, Y y Z en el espacio.

En el caso del PhysicsConstraint ligado al cajón, debemos bloquear su movimiento angular y limitar a 10 el movimiento lineal en X.

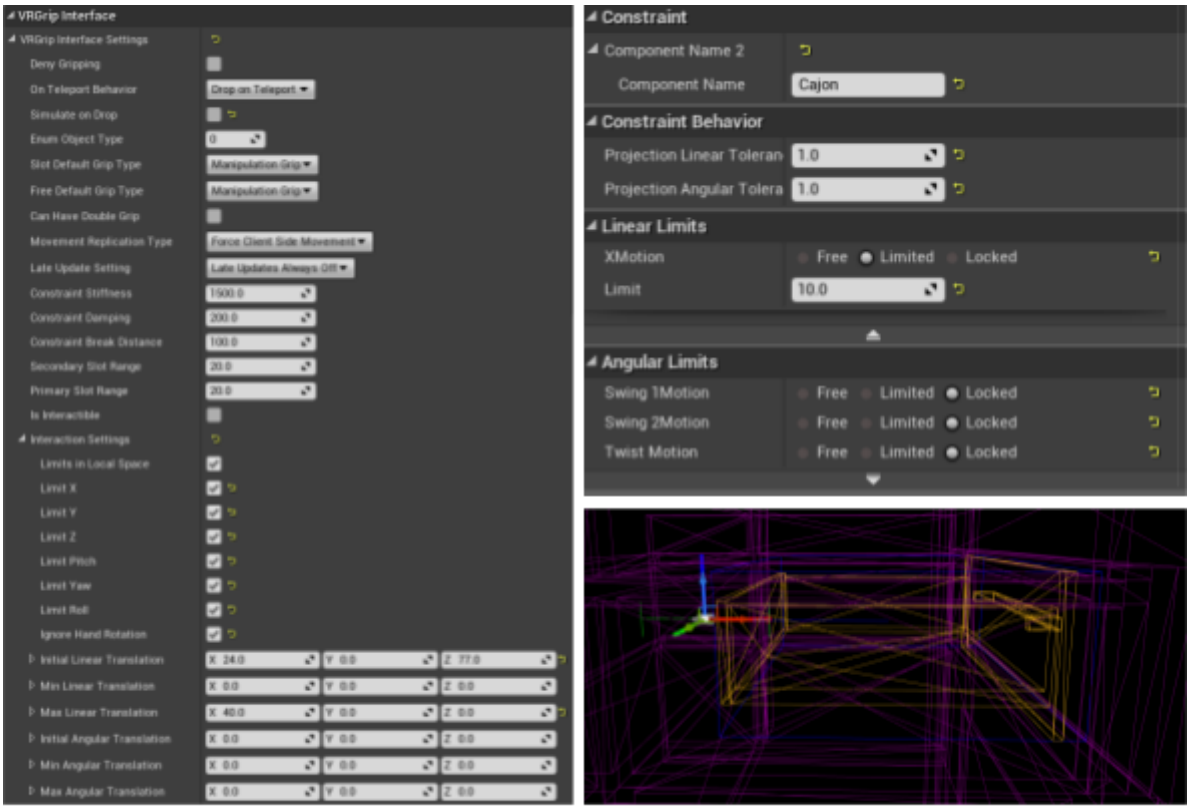

Figura 5.6: Configuración del elemento del cajón.

Cuerda de escalada

Buena parte del esfuerzo y el tiempo de investigación durante esta fase del proyecto fue absorbida por las pretensiones de crear un gancho enganchado a una cuerda del que se pudiese tirar. La complejidad técnica obligó a buscar alternativas, encontrando un componente por defecto de Unreal Engine 4 llamado Cable, que unido a dos elementos simula la física de forma rudimentaria en una cuerda.

A pesar de ello, el Cable no cuenta con colisiones, lo que merma la inmersión y la experiencia de usuario. Sí cuentan con colisiones los mangos, que dan coherencia a la física del cable al que son asignados en sus extremos. Sí hemos podido comprobar que es posible simular un mejor comportamiento físico mediante el uso de múltiples cables y conectores (como los mangos) y el uso de PhysicsConstraint. Bien es cierto que el comportamiento es más parecido al de una cadena con múltiples eslabones, que el de una

cuerda, además de que el aumento de cálculos físicos reduce significativamente el rendimiento.

Finalmente no quedamos satisfechos con el resultado, por lo que tuvimos que retirar este elemento del nivel Acrophobia en el que iba a requerirse el uso de esta cuerda.

Movimiento de escalada

Tras desechar la idea de la cuerda por dificultades en su funcionamiento, tuvimos que buscar alternativas a la resolución del nivel sobre la acrofobia.

Nos dimos cuenta de que VRExpansionPlugin proporciona la capacidad de escalar cualquier Static Mesh o Brush del nivel que no implemente VRGrip Interface con el flag DenyGripping a true. Para escalar, el jugador debe apretar el gatillo del controlador para agarrarse a la pared con al menos una de las manos, alternando entre las manos e impulsándose hacia arriba mientras busca dónde agarrarse con la otra mano. Es una interacción análoga a la real utilizada para escalar cualquier superficie en una situación real y nos pareció que podría ser una alternativa interesante para la experiencia de escapar del nivel.

Bote de pastillas consumibles

Nuestro protagonista tiene que tomar su medicación, que tendrá incidencia en cómo se sucederán los niveles y eventos del juego.

Para implementar estos mecanismos hemos creado dos actores dotados de comportamiento programado en Blueprints, uno correspondiente al bote de pastillas (y la tapa) y otro para las pastillas en sí.

El usuario recogerá un bote en cuyo interior se han hecho aparecer, en tiempo de ejecución, varias pastillas, con cálculo de físicas activado. El usuario podrá coger con una mano el bote y volcar el contenido del mismo sobre su boca (en realidad sobre cualquier parte de su cabeza), viendo así alteradas sus facultades y su visión.

El modelo 3D del bote se ha realizado de forma manual mediante formas básicas, lo que ha provocado que tuviésemos que hacer sus colisiones de forma rudimentaria para conseguir un hueco en su interior en el que introducir las pastillas.

Las pastillas, hasta que se abra el bote y se viertan en una superficie sólida o se tomen por el usuario, estarán dentro del bote, donde tendrán colisiones y físicas realistas y podrán colisionar entre ellas y con el interior del recipiente. A continuación se explicará más en detalle este asunto, pero cabe mencionar que se ha añadido a las pastillas sonido en su colisión, que se reproduce cuando colisionan con el bote.

Para poder crear un espacio hueco con colisiones en el interior del bote se ha recurrido a una solución provisional que soluciona el problema, consistente en la acumulación, en el perímetro del bote, de cajas de colisión combinadas, que restan realismo si se acude al

detalle y disminuyen el rendimiento, pero nos hacen posible hacer funcionar esta interacción.

Toma de pastillas

Las pastillas están formadas por un malla estática en forma de cápsula del que hereda un volumen de colisión en forma de caja. Esto nos permite simplificar las operaciones y el rendimiento cuando se dan múltiples colisiones, evitando así fallos propios de la acumulación de errores de físicas.

La función de las pastillas será la de ser consumidas por el usuario. Para ello, el mismo deberá coger el bote del cajón y volcar su contenido sobre su cabeza. Al realizar esta acción, la pastilla se precipitará sobre una esfera de colisión que rodea nuestra cabeza y que provocará la destrucción del Actor Pastilla, así como la aplicación de un estado alterado en la salida de vídeo del visor mediante la modificación de las opciones de postprocesado de imagen a las que se tiene acceso en el componente de la cámara de nuestro Pawn.

Las modificaciones en el post procesado consisten principalmente en la modificación del color de la imagen mostrada y en la adición de un ruido estático, entre otros.

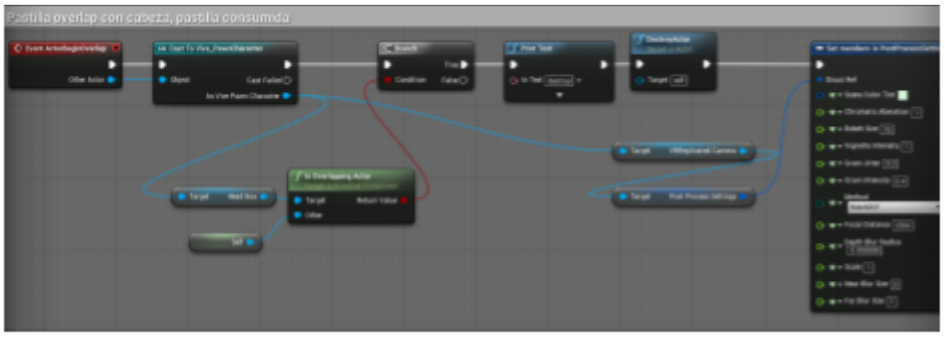

Figura 5.7: Blueprint para la destrucción de las pastillas y el procesado de imagen una vez tomadas las mismas.

5.2. Sprint 2: Funcionalidad de los niveles

Durante este sprint nos centramos en la implementación de niveles. El objetivo era tener tres niveles cuya funcionalidad pudiera probarse, sin atender demasiado a temas artísticos o de sonido. Para ello era necesario adaptar los mecanismos generados durante el primer sprint a un entorno productivo en el que tendría lugar el desarrollo de la experiencia. Este sprint se divide en dos partes bien diferenciadas: la creación del citado entorno de manera que todos pudiéramos trabajar con un escenario similar (recordemos que todos los puzzles tienen lugar dentro de la misma celda), y la implementación de los mecanismos propios de cada nivel, que realizamos de forma paralela cada uno de los miembros del equipo (cada uno un nivel).

5.2.1. Diseño de la celda y creación del nivel base

Se implementó en el entorno de desarrollo una celda siguiendo el diseño especificado durante la fase previa. El jugador no podía interactuar con nada por el momento, ya que se

creó en gran parte para comprobar si las medidas eran adecuadas (queríamos generar una sensación de agobio sin perjudicar demasiado la experiencia).

Se añadieron finalmente algunos elementos decorativos como tuberías y un lavabo. Estos elementos dan un aspecto más penoso a la celda, en el sentido de que indican que la persona no suele abandonar el espacio, pues tiene todo lo necesario para realizar sus funciones vitales dentro del mismo.

Figura 5.8: Diseño de la celda donde tiene lugar la experiencia.

Al tener tres niveles iniciales bien diferenciados, decidimos que cada miembro del equipo fuera el programador principal de uno de ellos, pudiendo requerir ayuda puntual del resto del equipo para solventar algún problema. Independientemente de los entornos de desarrollo que necesitará crear cada uno, se fijó como requisito indispensable que el resultado final se implementará en un entorno similar al que se observa en la Figura 5.8.

5.3.2. Implementación de nivel 1: Arachnophobia

Diseño

Durante el desarrollo de este nivel buscamos aprender a animar objetos diferentes al jugador, que se relacionen de alguna manera con el entorno y modifiquen la sensación del usuario. Se busca representar de alguna manera la aracnofobia, utilizando para ello elementos típicos de videojuego como son los generadores (*spawners*) automáticos de enemigos. Una vez destruido dicho generador, cesará la producción de enemigos.

En nuestro nivel, el váter de la habitación actuará de generador de arañas. Éste se activará cuando el jugador se acerque demasiado o cuando pase un tiempo predefinido. Una vez hecho esto, del váter comenzarán a salir arañas de manera indefinida hasta que el jugador lo destruya con ayuda de una herramienta diseñada a tal efecto. Si el jugador no lo destruye antes de que aparezca un determinado número de arañas, se considerará que no ha superado el nivel.

Implementación

Váter

Se trata de un actor diseñado expresamente para este nivel. Internamente contiene una variable booleana *spawnSpiders* y un sencillo bucle que se ejecuta mientras el valor de dicha variable sea verdadero. Dicho bucle llama constantemente al constructor del objeto araña que describiremos más adelante, pasándole como parámetros la situación actual del váter y una rotación fija para indicar en qué lugar debe colocar la araña. También indica un tamaño aleatorio para evitar la sensación de que todas las arañas son iguales.

Arañas

Las arañas no son más que numerosos ejemplares de una misma clase actor programada para que moverse constantemente hacia delante, cambiando su orientación en el momento en el que se encuentran con algún obstáculo, aplicando una rotación aleatoria para evitar que caminen en círculos constantemente. Estos dos movimientos clave se resolvieron utilizando una combinación formada por un evento personalizado llamado *MoveForward* que se invoca constantemente durante el bucle de ejecución (*Event Tick),* una función[27] llamada *ChangePosition,* que modifica la posición de la araña una vez detectado un obstáculo, y varios vectores de traza que se encargan de dicha detección. Pasamos a explicar en detalle cada uno de estos elementos.

MoveForward

Se trata de un evento simple que modifica la ubicación de la araña dentro del entorno, sumando un número fijo a la componente X de su vector de posición. Este evento se invoca a cada tick de tiempo (Event Tick).

[27] En Unreal, una función y un evento se diferencian por varios motivos: las funciones pueden tener parámetros de entrada y salida, mientras que los eventos no; los eventos ocurren dentro de la línea de tiempo, por lo que pueden invocar retardos y timelines; las funciones pueden usar variables locales, los eventos no.

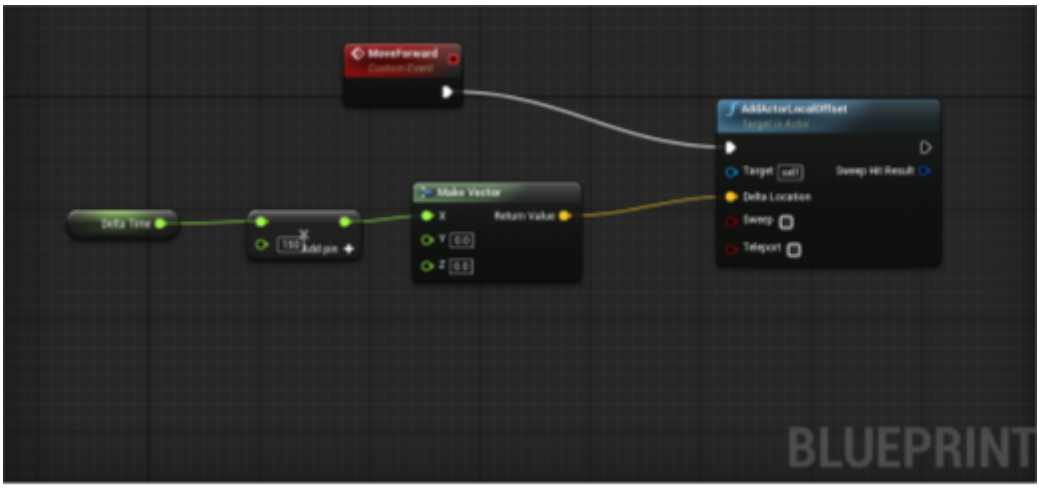

Figura 5.9: Evento MoveForward

ChangePosition

Diseñamos una función que recibe como parámetro de entrada las coordenadas de un punto. Una vez invocada, esta función cambia la posición del actor a ese punto y, además, comprueba su rotación actual para modificarla de manera que la araña se adapte al nuevo terreno, quedando sus patas tocando el mismo. En una primera versión nos centramos en que esta rotación sirviera simplemente para cambiar de terreno (de pared a suelo, por ejemplo). Esto provocaba que todas las arañas acabaran realizando el mismo camino todo el tiempo, por lo que tuvimos que modificar la parte de la rotación para que se orientara en una dirección aleatoria dentro de un arco de unos 90 grados (45 a la izquierda y 45 a la derecha) para que fuera más creíble.

Vectores de traza

Por último, para hacer uso de la función ChangePosition necesitábamos que la araña reconociese a cada paso el terreno que pisaba. Incluimos para ello cuatro vectores de traza (uno hacia adelante, dos hacia abajo y uno hacia atrás) que se proyectan una distancia fija cada vez que la araña avanza. El orden de ejecución es el siguiente:

- La araña avanza la distancia especificada en el evento MoveForward
- El vector de traza delantero comprueba si ha encontrado algún obstáculo (una pared, por ejemplo)
    - Si es así, se invoca ChangePosition con el punto de intersección entre el obstáculo y el vector de traza. Si no:
- Los vectores de traza hacia abajo comprueban si la araña tiene algo debajo
    - Si es así, no hace nada más

- Si no, el vector de traza trasero (que como se ve en la Figura 5.10 se encuentra un poco por debajo de la araña), busca el borde del lugar del que acaba de salir la araña y cambia su posición invocando a ChangePosition.

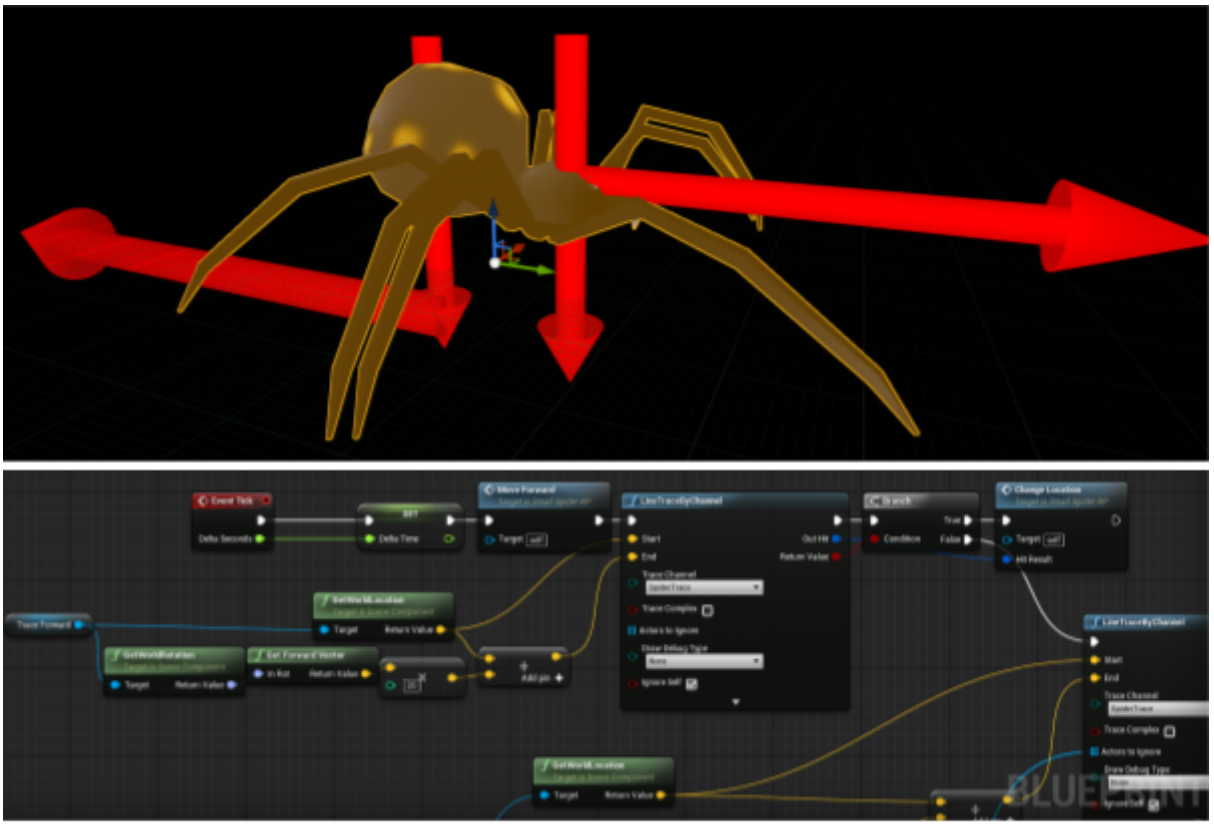

Figura 5.10: Arriba, el diseño de la araña con sus vectores de traza Abajo,.la primera comprobación que hace la araña tras moverse, siendo las siguientes similares pero usando distintas trazas.

5.3.3. Implementación de nivel 2: Claustrophobia

Diseño

El objetivo de este nivel es crear una sensación de claustrofobia. Para ello haremos que las paredes de la habitación se vayan acercando al jugador de manera progresiva. Para finalizar este nivel el usuario tendrá que parar las paredes durante un tiempo (utilizando los controladores o un elemento bloqueador que aún no hemos definido). Cuando esto ocurra, las paredes comenzarán a retroceder hasta que el jugador esté a salvo.

El diseño final es una celda que contiene diferentes objetos que en un determinado momento del nivel van a tener alguna funcionalidad. El nivel, como pudimos ver en el EDT, consistirá de una cama, una silla, una mesa, un grifo, un váter, dos ventanas una puerta con un cajón y las cuatro paredes de la estancia.

Implementación

Nivel de prueba y prototipo con cubos

Partiendo de un boceto, diseñamos el espacio y los objetos que iban a aparecer en la celda. Para esta primera aproximación, se utilizarán mallas estáticas sin funcionalidad para representar los objetos del interior de la celda. Las paredes son actores que se conforman utilizando varios static mesh, por el momento, no tienen ninguna funcionalidad.

En este punto, tanto los objetos como los actores, no disponen de ninguna textura definitiva ni ningún tipo de arte asociado. Son cubos y contenido que viene por defecto en el proyecto creado en Unreal.

Además, para hacer pruebas iniciales se desarrolla un nivel vacío para ir desarrollando y testeando las primeras versiones de software funcional antes de introducirlas al nivel final. La necesidad de este nivel surgió inicialmente por el desconocimiento de la tecnología. Es decir, se creó dicho nivel con el principal motivo de aprender.

Lo primero, es introducir cuatro cubos que simularán que son las cuatro paredes de nuestra celda. Con estos cubos diseñaremos el movimiento de las cuatro paredes del nivel.

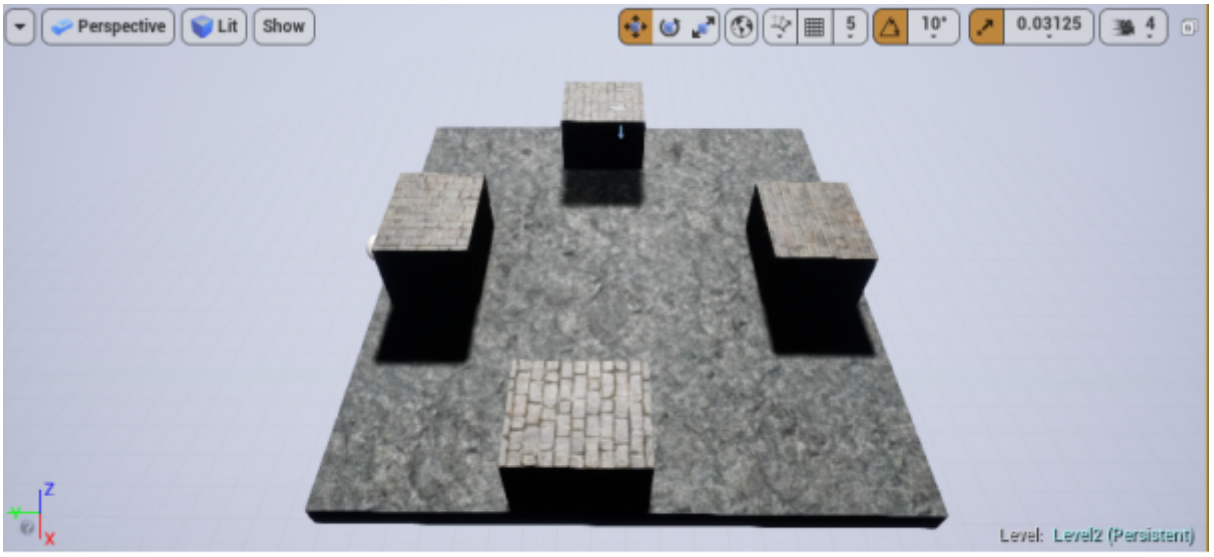

Figura 5.11: Nivel de prueba con cubos para el diseño de Claustrophobia

Como vemos en la Figura 5.11 son cuatro cubos enfrentados por pares. La idea es que cada uno de los cubos se mueva hacia al centro hasta que lleguen a cruzarse. También es importante destacar que cuando se enfrenten deben poder atravesarse. En caso contrario las paredes se bloquearán y no podrían acercarse más.

Ahora que ya tenemos todo dispuesto en nuestro mapa es hora de ponerse a desarrollar. Lo primero de todo es hacer que un cubo se moviese. Al principio apenas pudimos desarrollar porque desconocíamos la tecnología. Tras una investigación tanto en la documentación de Unreal como en sus foros, encontramos la función Timeline que a priori nos podría servir.

Función Timeline

En la Figura 5.12 podemos ver el panel que se carga nada más entrar dentro de la función Timeline.

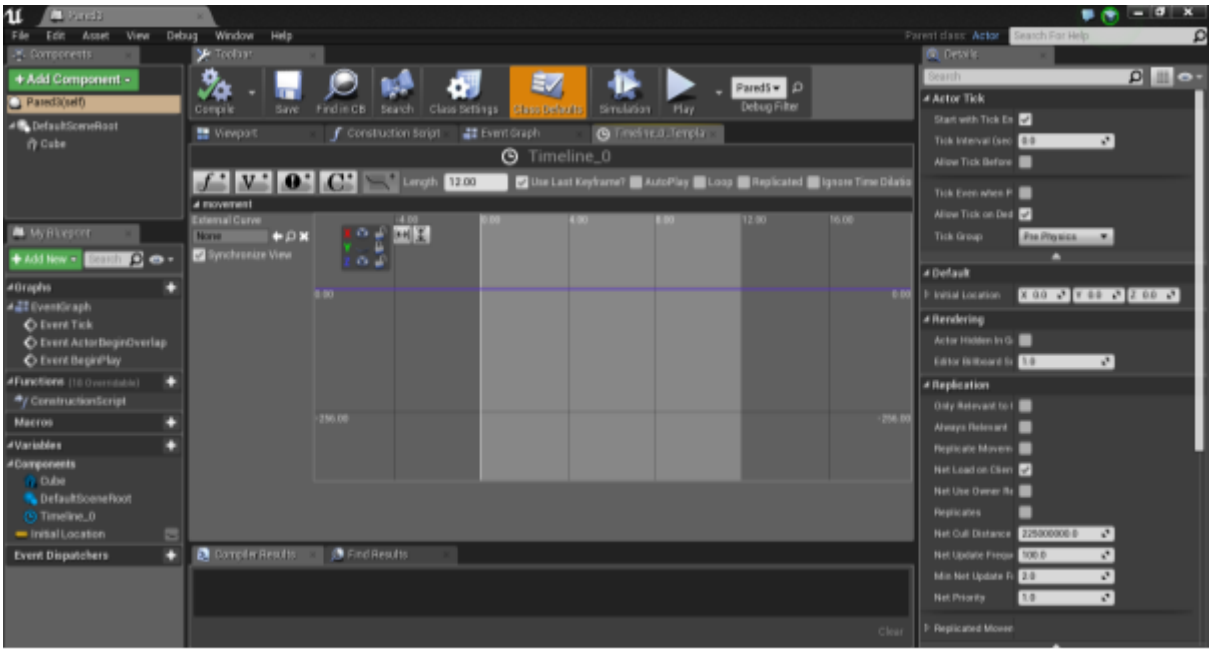

Figura 5.12: Panel inicial Timeline

En este panel podemos crear pistas de distintos tipos. Los tipos que nos permiten crear son de tipo float, vector, eventos o incluso pistas con colores. El funcionamiento de estas pistas es el mismo en cualquiera de los casos.

Se dispone de una gráfica en la que puedes editar cada uno de los valores del los ejes. Los valores que vamos a necesitar, y por lo tanto vamos a modificar, son el par tiempo-valor. Lo que va a indicar la gráfica es: “a lo largo de y segundos, el elemento se va a mover x unidades”. Gracias a esta gráfica podemos ajustar el tiempo que queremos que se mueva, la forma del movimiento (lineal, constante, etc.) y la cantidad de espacio recorrido en ese tiempo.

Disponemos de una variable de tipo vector (*initialLocation*), añadida en el Actor del cubo/muro e informada en tiempo de ejecución durante la construcción del mismo; por lo que crearemos una pista del mismo tipo. Gracias a esto, podemos hacer operaciones entre ambos, así que realizando una simple suma del espacio recorrido durante el tiempo asignado en cada iteración y la variable initialLocation, podemos conocer la posición del cubo en cada momento durante la ejecución del timeline.

Los ejes que vamos a utilizar son el X e Y. Cada par de cubos se moverá en el eje que le corresponde, uno en sentido positivo y otro en sentido negativo. La gráfica resultante sería de este tipo.

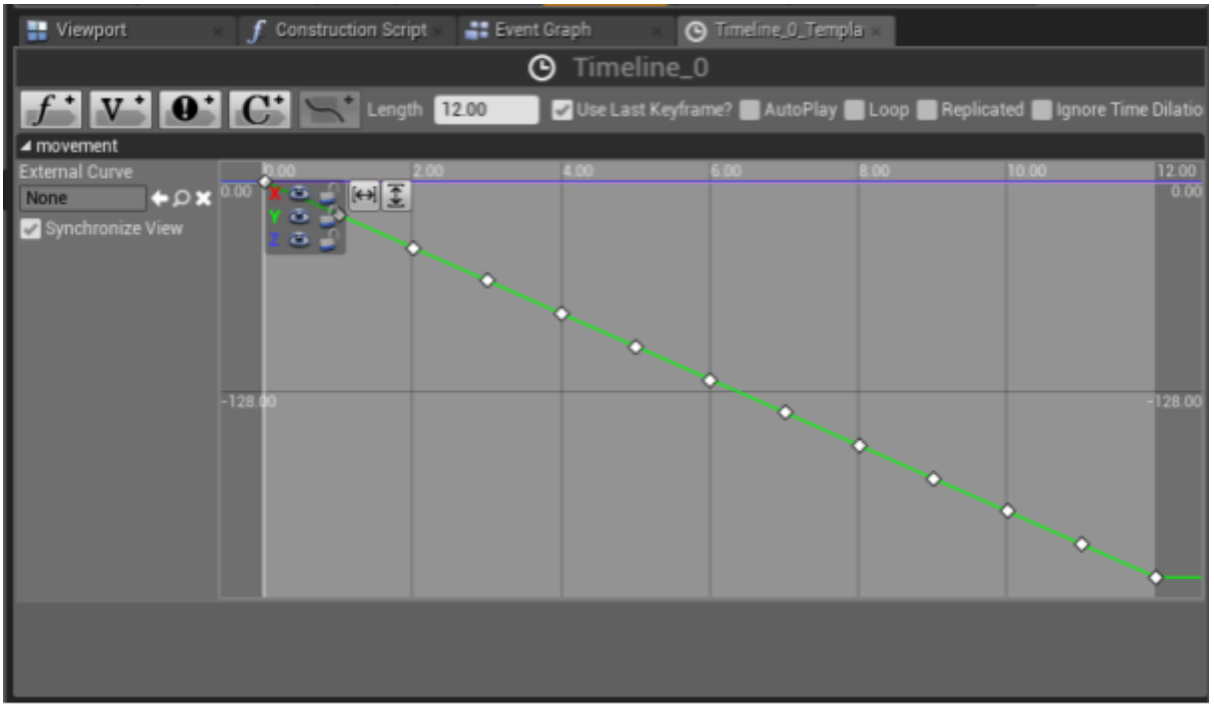

Figura 5.12: Timeline aplicado al Actor de la pared.

Con una función lineal tan sencilla como la de la gráfica anterior cubrimos el movimiento de los cubos que necesitamos. Los valores de la gráfica son totalmente ajustables simplemente añadiendo o quitando puntos a lo visto en la Figura 5.12 tanto para los cubos que se muevan en el eje y como en el eje x. Cuando queramos editar un único eje, ponemos el candado a los otros dos para evitar confusiones. El cuadro de texto en el que pone *length* marcará el tiempo máximo que queremos que dure el timeline. Cualquier acción fuera de este tiempo no se realizará.

Por último dentro de esta función, destacar los checkbox que sirven para definir las características de movimiento en el timeline. Por ejemplo, si marcamos loop, realizará el timeline de un lado a otro (origen-destino a destino-origen) constantemente en bucle. Nosotros utilizamos *Last Key Frame* para no ignorar el último frame activo durante la secuencia y que el movimiento se vea seguidos y no a saltos.

Destructible Mesh

Ante el avance de las paredes, los elementos de la habitación se irán rompiendo para crear una mayor sensación de agobio en el usuario. Para desarrollar esto en el nivel de prueba, utilizando cubos y contenido del Starter Content[28] aprenderemos a crear mallas destructibles (objetos que se pueden dividir en fragmentos durante el juego) y ajustar el grado de ruptura y de resistencia de cada uno de estos objetos.

Para crear un destructible mesh, simplemente debemos utilizar el static mesh y crear un destructible mesh a partir de él. Una vez generado los destructibles a tratar simplemente

[28] Pack de elementos incluidos por defecto en el entorno de Unreal.

tenemos que ajustar los valores de cada uno de ellos a nuestro gusto. En la Figura 5.13 podemos ver la características editadas en un destructible estándar de Unreal Engine .

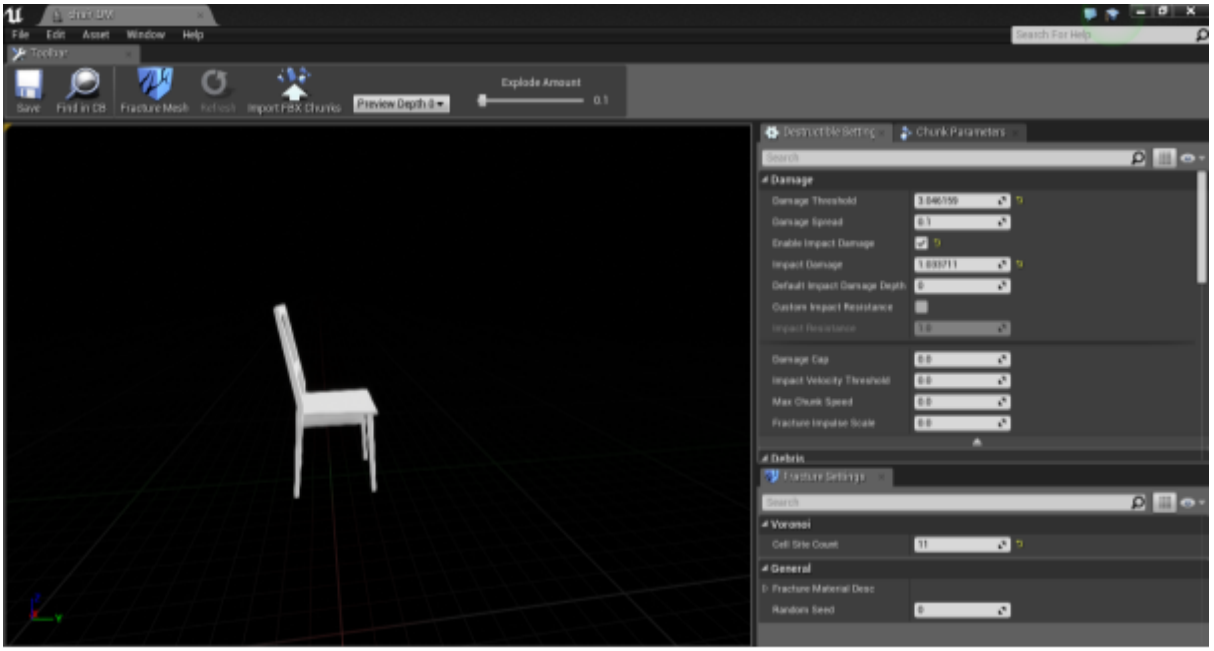

Figura 5.13: Panel de desarrollo de objetos destructibles.

A continuación explicaremos lo que significan cada uno de los parámetros de la imagen.

- Damage Threshold: La cantidad de daño que se le puede aplicar a un objeto o pedazo antes de fracturarlo.
- Damage Spread: Para controlar la distancia dentro del destructible al propagar el daño.
- Enabled Impact Damage: Permitir aplicar daño a un destructible cuando colisione con un objeto.
- Impact damage: cantidad de daño que recibe un objeto destructible al colisionar con otro objeto.
- Impact resistance: cantidad de resistencia al daño que tiene un objeto destructible. Esta cantidad se restará al daño recibido para calcular el daño final.
- Cell site count: la cantidad de pedazos que se generan en el proceso de fractura.
- Random seed: semilla aleatoria para generar pedazos en un destructible.

La configuración de cada uno de los destructibles varía en función de lo que queramos crear. En la Figura 5.13 podemos ver la configuración de uno de ellos como ejemplo. Lo importante de este punto es saber manejar estas características para crear los objetos destructibles a nuestro gusto.

Desarrollo funcional del movimiento de las paredes

A continuación se adjunta una captura (Figura 5.14) de la composición del blueprint que crea el movimiento de los cubos.

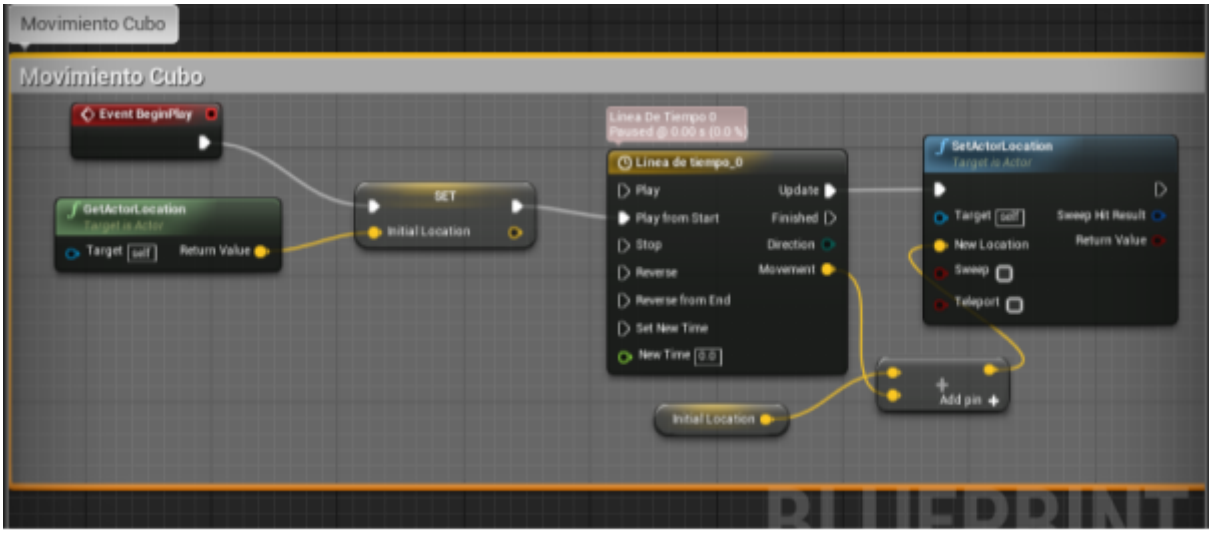

Figura 5.14: Función timeline incluida en el blueprint final

El inicio de la ejecución lo marca el *EventBeginPlay*. Invocado este evento ejecutamos el método set. Este método asigna un valor a una variable creada previamente llamada *initialLocation*. Esta variable nos servirá para conocer en qué punto exacto se encuentra el cubo en cada momento. Para asignar el valor inicial a la variable utilizamos el *getActorLocation* que junto con el método set, modifica la variable y le asigna la posición de partida del cubo en el nivel. Una vez que conocemos el punto de partida, invocamos a la función Timeline que hará que comience el movimiento del cubo.

Cuando acaba de ejecutarse el timeline, devuelve una actualización con la nueva posición del actor. Para calcular la posición correcta y pasarlo como argumento al new location de la función *setActorLocation*, hacemos la suma del movimiento recorrido por el timeline y al valor de la variable *initialLocation*. Para utilizar el valor de la variable, haremos un get sobre la misma. Haciendo esta operación durante el tiempo que se reproduce el timeline iremos viendo cómo se mueve el actor y por consiguiente conseguiremos la posición inicial en cada momento para la siguiente iteración.

Finalmente se incluyó toda esta funcionalidad en la celda base diseñada previamente y pasamos a implementar la condición de victoria: que las paredes parasen al cumplirse dicha condición y comenzasen un movimiento de retroceso.

Parada y retroceso

Llegados a este punto necesitamos agregar la funcionalidad de reverso y parada a nuestras paredes. Lo primero es comentar que hemos agregado una nueva variable booleana llamada *alive*. Con esta variable podemos controlar si al final del nivel el jugador lo ha superado o en caso contrario no lo ha conseguido. En este punto y para depurar simplemente haremos un print de una cadena en la pantalla con diferentes mensajes

dependiendo si supera el nivel o no, después del print se invoca la función *setGamepaused* para pausar el juego.

Ahora comenzaremos a analizar las novedades del blueprint. Primero analizaremos cómo gestionamos si se detiene la pared. Como vemos en la Figura 5.15, para gestionar la parada utilizaremos el método *Get input key down* con el botón izquierdo del ratón. La condición consiste en que si mantiene pulsado el botón quiere decir que está parando la pared por lo que el método *set* de la variable *alive* cambia su valor a *true* y en el método timeline le decimos que pare (Stop). En cuanto deje de pulsar el botón, la condición será falsa e indicaremos en el timeline que avance (Play). Para que pueda realizarse iteraciones constantes utilizamos la función event tick que se va ejecutando en cada frame reproducido. En la versión anterior no la utilizamos y el flujo era lineal desde el Event Begin Play, gracias a este evento podemos modificar el valor de las variables en cada frame.

Por otra parte, para gestionar el reverso de la pared el método es exactamente el mismo que para que pare, pero en vez de pulsar el botón izquierdo pulsaremos el derecho, modificaremos mediante el set la variable alive a true y en el timeline utilizaremos el input de *reverse*.

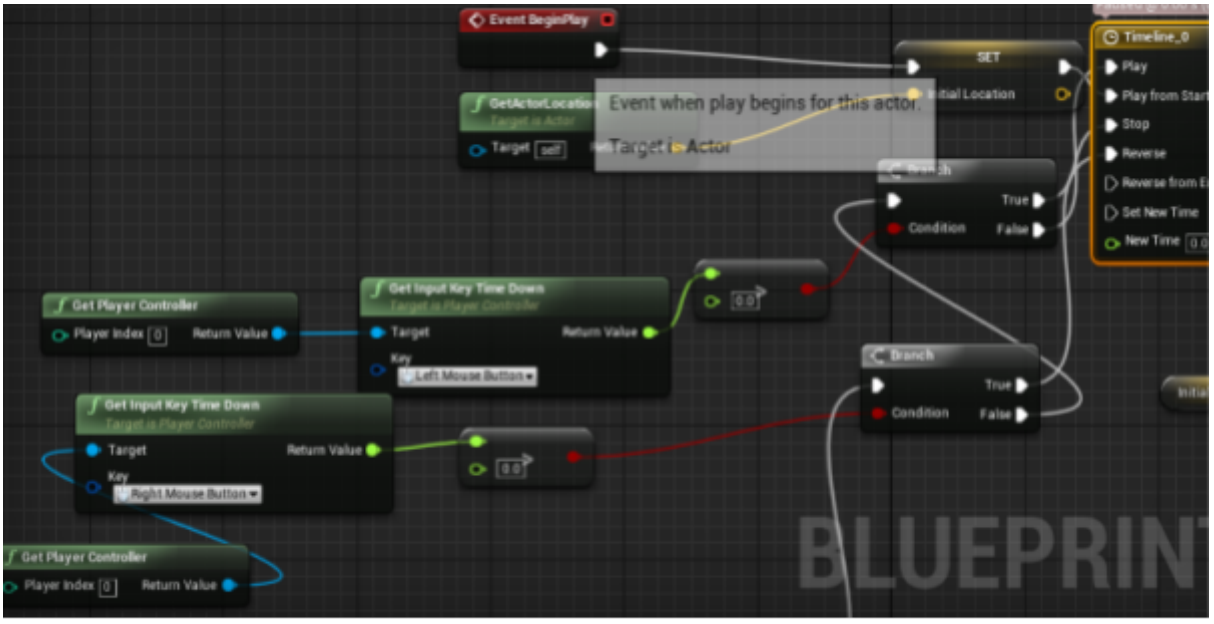

Figura 5.15: Parada y retroceso de las paredes.

Por último, queda comprobar que el nivel termina una vez las paredes se han cerrado del todo (el jugador pierde) o se han vuelto a abrir hasta su posición inicial (el jugador gana). Para ello, una vez acabado el timeline utilizamos su salida Finished y lo pasamos a una condición junto con la variable alive. Simplemente mirando si la variable está a true o a false vemos si el nivel está superado o no. En este caso utilizamos la función setGamePaused, que pausará el juego y mostrará por pantalla el mensaje que indiquemos en el método *printString*.

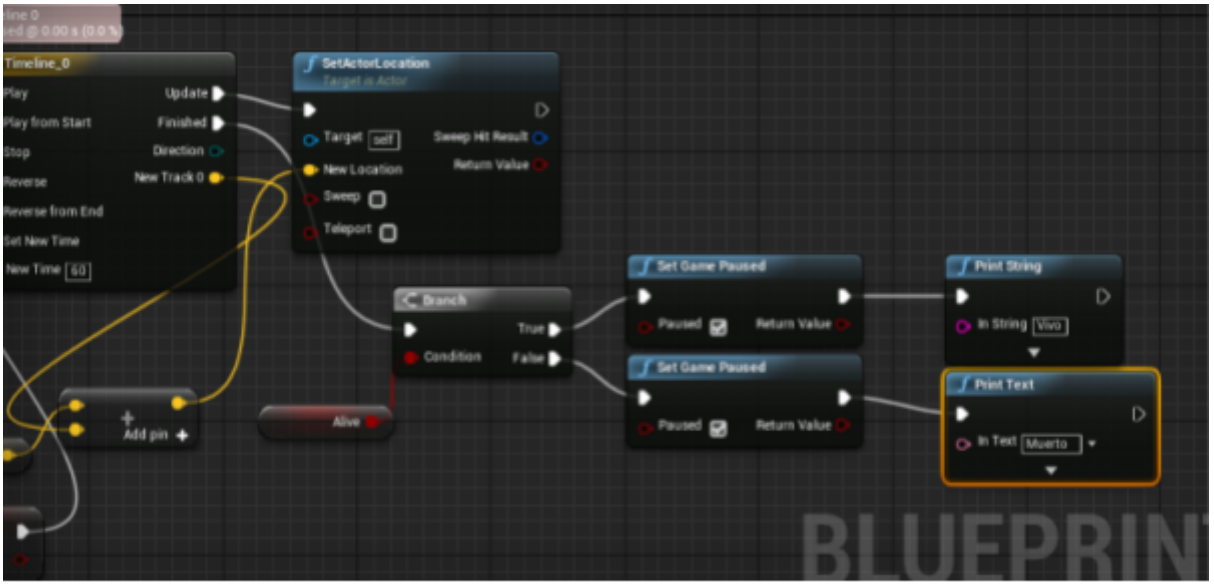

Figura 5.16: Condiciones de victoria y derrota en el nivel Claustrophobia

5.3.4. Implementación de nivel 3: Acrophobia

Diseño

Durante el desarrollo de este nivel pretendemos profundizar en el uso de escenarios destructibles mediante la creación de un entorno que se destruya de forma realista y que estimule la acrofobia, así como el uso de eventos personalizados que se adapten al guión del juego. Una acción disparará un evento que se ajuste al guión y se sucederán una serie de pequeñas roturascontroladas de baja intensidad en el suelo, quebrándolo parcialmente y permitiendo al usuario asomarse a un acantilado rocoso que generará esa sensación de vértigo. Una nueva acción permitirá al usuario retornar a la situación inicial, con el suelo y la habitación intacta.

Implementación

DestructibleFloor

Blueprint correspondiente al suelo que se destruirá durante el transcurso del nivel y que contiene un Destructible Mesh. Es un actor simple que incluye varios puntos de impacto donde se producirán pequeñas explosiones llegado el momento de derrumbar el suelo.

ExplodeFloor

Este evento, ubicado en el blueprint del nivel, dará lugar a la secuencia de operaciones que permitirá la rotura del suelo. Para ello y por este orden debemos:

- Añadir simulación de físicas a los objetos del escenario que no las tienen, tales como la silla, la mesa y el váter para que se precipiten al vacío cuando el suelo se desplome.

- Guardar en variables del nivel el lugar que ocupan en el espacio los diferentes elementos (Transform), que más tarde caerán al vacío. Guardamos así la posición de la silla, el escritorio, el váter y por supuesto, el suelo, ya que más tarde habrá que restaurarlo a su situación inicial.

- Destrucción del suelo mediante el uso de explosiones controladas de diferente intensidad. Dado que el mapa de rupturas del suelo hemos decidido que fuese aleatorio, se ha tenido que ajustar cada explosión para que el resultado fuese estético, ya que la disposición de los trozos y los objetos que se encuentran sobre el suelo modificaban el impacto de la explosión y el efecto de la onda expansiva en la superficie limítrofe del área de la explosión.

Las explosiones se realizan mediante la aplicación de un daño radial al suelo en los puntos colocados en el mapa como objetivo con un daño base suficiente como para vencer la resistencia del objeto (variando entre los 2000 y los 1100 puntos de daño) y un radio de acción de 50. Entre explosión y explosión se ha añadido un pequeño delay para permitir que diese la sensación de escalonamiento típica de una demolición controlada en lugar de provocar que la habitación entera estalle en mil pedazos.

Figura 5.17: Arriba, la preparación previa a la explosión. Abajo se aplica daño a los puntos del DestructibleFloor

Reconstrucción de la celda

Un nuevo evento indica el momento de reconstruir la habitación. Para ello se deberán destruir los actores que han caído al vacío y se encuentran en el fondo del acantilado, además del suelo quebrado. Inmediatamente después se deben recrear estos actores en el nivel justo en las posiciones que hemos almacenado previamente.

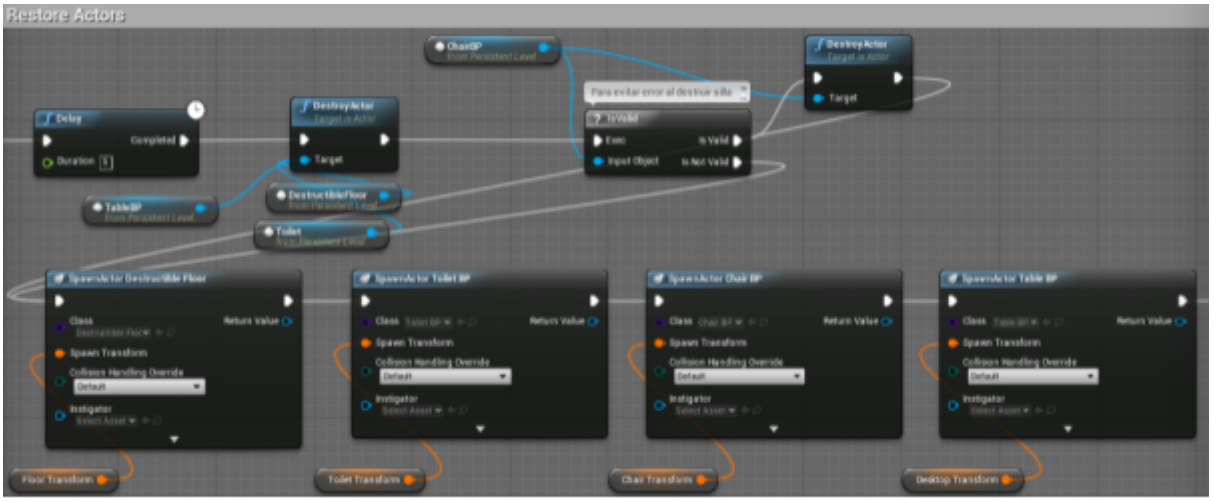

Figura 5.18: Reconstrucción de la habitación.

5.3. Sprint 3: Integración con el dispositivo de Realidad Virtual

Hasta ahora lo que hemos hecho ha sido un prototipo funcional de niveles que puede ser reproducido con cualquier ordenador, un videojuego que podríamos considerar "convencional". Organizamos el trabajo así por disponer sólo de un dispositivo de VR (el HTC Vive), por lo que era imposible trabajar de forma paralela en RV.

Sin embargo, ahora que ya tenemos completada la funcionalidad básica de los tres niveles que desarrollaremos, es necesario comprobar que nuestro diseño funciona con Realidad Virtual y hacer las modificaciones y ajustes que sean necesarios. Se incluye en este apartado cómo solucionamos los problemas encontrados al incluir esta funcionalidad. En algunos casos añadir esta nueva funcionalidad fue trivial, mientras que en otros se requirió algún pequeño cambio con respecto al diseño original que mencionaremos a continuación.

5.3.1. NavMesh y colisiones con el jugador

Lo primero que debemos conseguir es incluir el jugador correctamente en el entorno. Para ello debemos configurar las colisiones del mismo para evitar que pueda, por ejemplo, sacar la cabeza por una pared o atravesar los muebles con el cuerpo. Sin embargo, tras algunas pruebas sencillas comprobamos que la sensación al chocar con algún mueble es extraña, ya que en la realidad podemos continuar andando mientras que en el espacio virtual no nos movemos. Por esto decidimos dar cierta tolerancia a la colisión con algunos objetos.

Además, incluimos la posibilidad de poder teletransportarnos por la habitación gracias al uso del elemento *NavMesh* con el que ya habíamos experimentado durante el sprint 1.

Figura 5.19: Elemento NavMesh incluido en un nivel. La zona resaltada en verde es por la que podrá moverse el jugador.

Obsérvese en la Figura 5.19 que el jugador ahora puede, si quisiera, teletransportarse a una zona situada encima de la cama. Esto vino motivado por un pequeño cambio de diseño en uno de los niveles que comentaremos más adelante.

5.3.2. Adaptación a Realidad Virtual del nivel 1: Arachnophobia

Esta iteración fue la más sencilla dentro de este sprint, necesitando modificar poco o nada el diseño inicial. Simplemente nos limitamos a añadir la funcionalidad de destructible al Actor del váter ya generado. Pudimos utilizar todo lo diseñado a este efecto durante el primer sprint.

Tras las pruebas realizadas para este nivel nos dimos cuenta de que requerir al jugador que realizara alguna acción para comenzar el nivel era quizá muy poco intuitivo, por lo que tomamos la decisión de rediseñar la experiencia de manera que fuera un poco más guiada. Ahora las arañas empezarían a salir tras un lapso de tiempo prefijado. También decidimos fijar un tiempo aproximado de un minuto para cada nivel. En el caso de Arachnophobia, este tiempo es fácil de medir: el váter genera una araña cada 0,5 segundos, así que cuando se generen 120 arañas, el jugador pierde. Para esto añadimos al Actor del váter una variable *numSpiders* que se incrementará en uno cada vez que se genere una araña. Tras este incremento, comprueba que el número es menor de 120, parando el juego si no lo es.

5.3.3. Adaptación a Realidad Virtual del nivel 2: Claustrophobia

Quizá uno de los más costosos, por el movimiento continuo del entorno. Al cerrarse las paredes sobre el NavMesh colocado, se daba la posibilidad de teletransportarse fuera de la

habitación apuntando el mando a través de la puerta o alguna ventana. Para solucionar esto tuvimos que configurar el NavMesh como dinámico de manera que se fuera ajustando a medida que las paredes se iban cerrando. Esto se consiguió mediante una configuración incluida en el plugin VRExpansion.

Por otra parte, al cambiar el control de las paredes (para realidad parada y retroceso) desde el ratón a los controladores de HTC Vive nos dimos cuenta de que esta funcionalidad nos venía muy bien para depurar. Razón por la cual, aunque más tarde deberíamos quitarla, la dejamos incluida durante la fase de desarrollo.

Condición de victoria: elemento bloqueador

Tras analizar detenidamente nuestras opciones con los controladores comprobamos que la idea de parar las paredes con las manos, además de ser bastante complicada de implementar, no proporcionaba una experiencia inmersiva. Al perder de vista los controladores en pantalla no se tiene la sensación de estar sujetando nada pues, en realidad, el jugador no está apoyado contra una pared. De esta manera decidimos eliminar esta condición de victoria y centrarnos en el elemento bloqueador.

Decidimos que este elemento fuese la cama. En este punto del desarrollo ya habíamos elegido el recurso gráfico que se ve en las imágenes y nos parecía lógico que, al ser de hierro, pudiera permanecer intacta ante el avance de la pared. Por hacerlo un poco más complicado decidimos que el jugador debía volcar la cama de manera que se produjera un espacio libre entre esta y dos paredes para que pudiera colocarse “a salvo” en él.

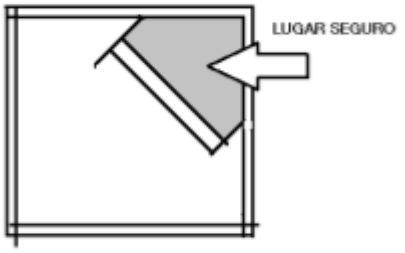

Figura 5.20: Diseño final de la condición de victoria en Claustrophobia

Para gestionar las colisiones de la cama con las paredes agregamos 4 cajas de colisiones nuevas repartidas entre la parte de alante de la cama y la parte de atrás, 4 booleanos llamados tocapata_XY (La X representa el muro que debe tocar y la Y la caja de colisiones asociada a esa pata.) los utilizamos para gestionar el contacto entre la cama y las paredes, una variable booleana victory para controlar si se supera el nivel o no y por último implementamos el interfaz de objetos agarrables VRGrip como ya se ha explicado anteriormente.

Para controlar que ambas patas tocan la pared correcta y en el momento adecuado, utilizamos la función *onBeginOverlap,* que hace un cast al objeto que debería estar tocando. Si es correcto, se modifica el valor de la variable *tocaPata* del muro correspondiente (Wall2 o Wall3) a *true*. En caso de que no esté tocando, la variable *tocaPata* obtendrá el valor false. Como vemos en la Figura 5.21, en caso de que estos dos booleanos sean true se modifica otra variable booleana llamada *tocandoPata* que previamente se ha incluido en el Actor del muro correspondiente. Esta variable está conectada directamente a la entrada Stop del timeline de ese muro, de manera que deje de avanzar cuando esto se cumple.

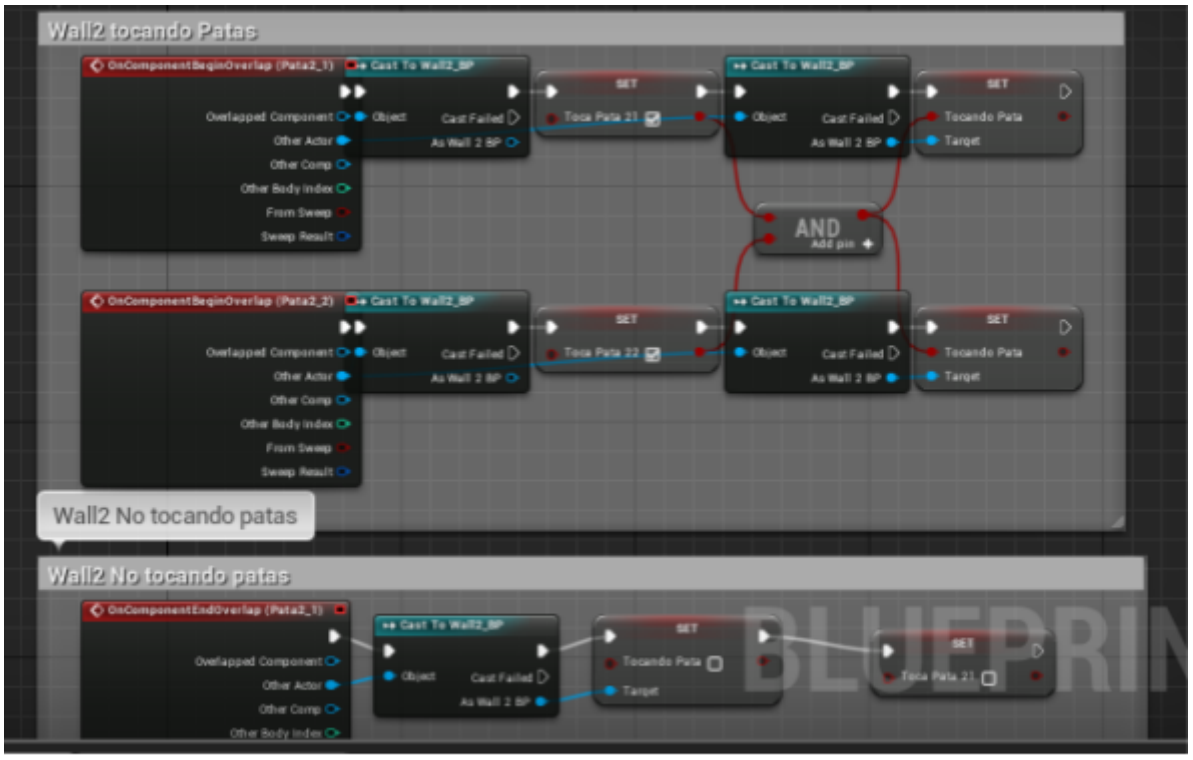

Figura 5.21: Eventos de overlap en las distintas patas.

Finalmente añadimos un *trigger*[29] debajo de la cama de manera que sólo es accesible una vez esta ha sido girada. Cuando el jugador se sitúa en este punto, el blueprint comprueba si la cama está colocada en la posición correcta (con cada pata tocando el muro correspondiente) y, de ser así, pone a true la variable *victoria* incluida en este mismo Actor. Será el blueprint del nivel el encargado de comprobar en cada frame si esta variable se cumple y, en caso de que así sea, pausar el juego indicando que se ha superado el nivel.

Condición de derrota: las paredes se cierran demasiado

Por último, comprobamos que con el nuevo diseño no nos valía medir el tiempo que se estaba ejecutando el timeline de cada muro, pues al parar sólo uno o dos de ellos y no todos como habíamos pensado inicialmente este tiempo era completamente diferente para cada muro. Solucionamos esto de manera sencilla colocando una traza similar a la colocada en las arañas del nivel de Arachnophobia. Esta traza, situada en la puerta de la celda en un ángulo de 45 grados, reacciona sólo al muro hacia el que apunta, poniendo el juego en pausa e indicando que el jugador ha perdido cuando éste se cierra demasiado.

5.3.3. Adaptación a Realidad Virtual del nivel 3: Acrophobia

De forma similar a lo decidido en Arachnophobia, eliminamos la necesidad de una acción por parte del usuario para iniciar el nivel. Una vez cargado este, se esperará unos segundos y se comenzará a destruir el suelo. Este nivel también fue bastante sencillo de adaptar, añadiendo simplemente el NavMesh para el movimiento del jugador. Tomamos la decisión

[29] Similar a una caja de colisiones, pero diseñado para disparar algún evento. En este caso, avaluar la condición de victoria del nivel.

de usar como lugar seguro la superficie sobre la cama, dado que es la única parte de la habitación que no se destruye.

Finalmente, añadimos dos triggers similares al ya explicado de la cama. Uno al fondo del acantilado, que abarca toda la escena como una especie de plano infinito e indicará que el jugador ha perdido si se dispara (porque ha caído al vacío); y otro en el lugar que debe alcanzar éste para salvarse, que disparará la condición de victoria.

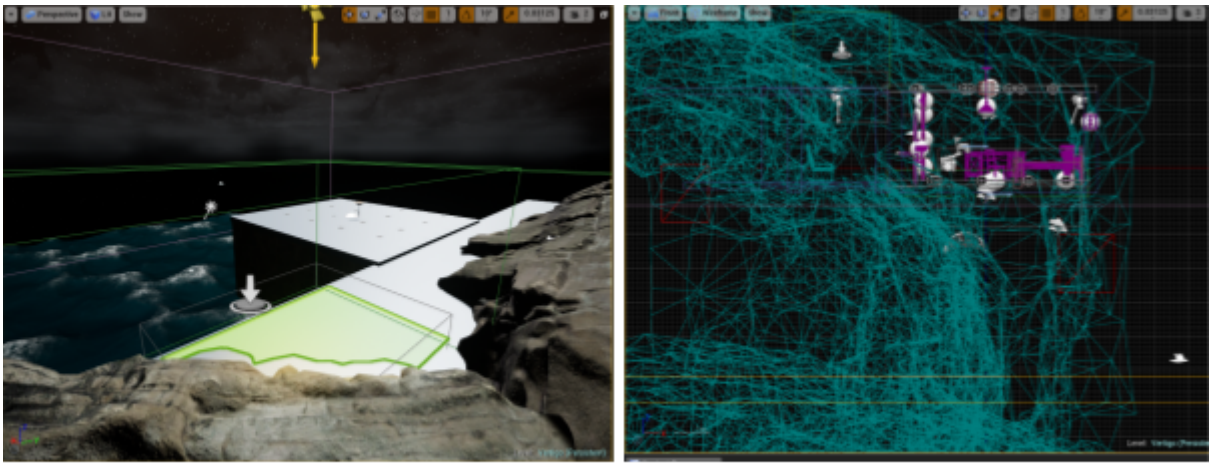

Figura 5.22: A la izquierda, el trigger para la condición de victoria del nivel Acrophobia. A la derecha, en la parte inferior de la imagen se puede observar el trigger de derrota.

5.4. Sprint 4: Integración de todos los niveles en la aplicación final

Con todos los niveles implementados y funcionando correctamente, llega el momento de unirlos en un juego completo que funcione de inicio a fin según lo diseñado. El requisito principal de este sprint es conseguir que desde el nivel de calma pueda cargarse cualquier otro nivel y, al completar este segundo, se vuelva al primero.

Además, como requisito adicional se presenta la creación de un menú principal desde el que iniciar la experiencia. Este menú se desarrollará como un nivel más en el que comenzará el jugador al ejecutar el software. Desde éste se podrá iniciar el juego desde un botón *Start* o acceder al nivel de sandbox, donde probar los diversos mecanismos implementados, haciendo uso del botón *Dev. Sandbox.*

5.4.1. La clase GameInstance

En cada ejecución del software desarrollado con Unreal se crea un ejemplar de una clase llamado GameInstance. Esta clase puede ser la predefinida de Unreal o una personalizada que cree el desarrollador. En nuestro caso, utilizamos la clase definida en el plugin VRExpansion, que hereda de la estándar de Unreal y se llama *VRGameInstance*.

En esta clase se puede almacenar cualquier tipo de información relativa al juego o incluso disparar eventos personalizados. Nosotros hemos añadido una variable de tipo Mapa en la que se indica el nombre de los niveles como clave y como valor un booleano que indica si el nivel se ha superado o no. De esta manera, consultando este mapa desde el nivel de calma podemos cargar un nivel u otro, o incluso modificar elementos de forma dinámica en el nivel. Utilizamos esto para crear un evento personalizado *LoadLevel* al que se puede llamar desde el nivel de calma cuando el jugador se toma las pastillas.

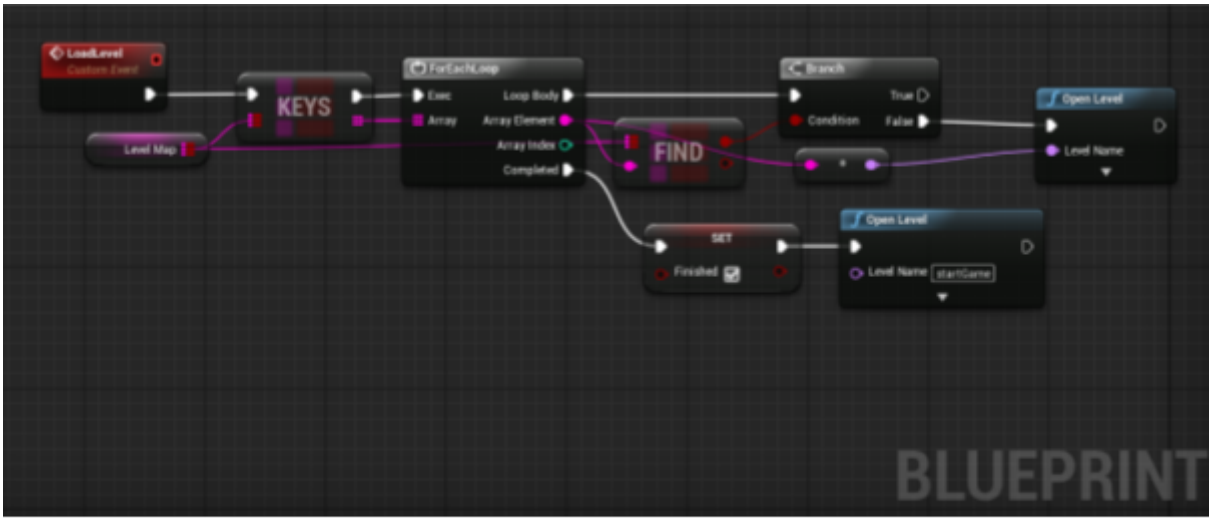

Figura 5.22: Llamando al evento LoadLevel, un bucle recorre el mapa con los niveles, cargando el primero que no se haya completado. Si se han completado todos, se indica en una variable booleana y se vuelve al menú inicial.

Este mapa nos permite una cierta escalabilidad en el proyecto, siendo realmente sencillo añadir más niveles si se necesitara. Bastaría con añadir el nombre del nivel como clave de dicho mapa y asegurarnos de que su valor es negativo al inicio del juego. Además, el hecho de que sea una variable pública nos permite consultarlo desde cualquier sitio que necesitemos para realizar modificaciones en la interfaz de acuerdo con el avance del juego. Usamos esto para cargar una nota diferente (véase 5.1.3. Nivel Sandbox o caja de arena, Notas e interfaz de usuario) en el nivel de calma dependiendo del nivel que se vaya a cargar a continuación, dando así pistas al jugador de cómo superarlo.

5.4.2. Menú de inicio del juego

Finalmente, debemos implementar un menú de inicio desde el que comenzar la experiencia. Se tratará de un menú sencillo con sólo dos opciones: *Start* y *Dev. Sandbox,* que nos llevarán respectivamente al nivel de calma para comenzar el juego o al nivel de desarrollo para echar un vistazo a todo lo desarrollado.

Para conseguir esto creamos un nivel completamente nuevo llamado *StartGame.* En este nivel el jugador no podrá teletransportarse ni interactuar con nada que no sea el menú. Para potenciar la inmersión se decidió colocar el menú en la puerta de la celda, como para indicar que se va a entrar en cuanto se pulse *Start.*

El menú se desarrolló con la ayuda de los widgets de Unreal, al igual que las notas explicadas en el punto 5.1.3. Además, para ayudar al usuario a tener una referencia con la que apuntar hacia los botones, se añadió al *PawnCharacter* una luz focal que sólo aparecerá durante el nivel StartGame. Esto se logra modificando durante la carga del nivel la variable *visibility,* incluida por defecto en todos los objetos de Unreal.

5.5. Sprint 5: Retoques finales y mejoras en iluminación y sonido

Con el proyecto cerrado a nivel funcional llevamos a cabo varias pruebas preliminares, de las que se realizan rutinariamente mientras se está desarrollando. En ellas identificamos la necesidad de incluir sonido en la experiencia para una mayor inmersión, así como algunos retoques en la iluminación que debíamos llevar a cabo para dirigir mejor al jugador hacia las soluciones de los distintos puzzles. También identificamos una serie de fallos menores cuyas soluciones detallamos en el capítulo 6 de esta memoria.

La inclusión de sonido fue más una tarea de exploración de recursos audiovisuales que de programación, pues debimos buscar sonidos que se adaptaran a lo que queríamos y, además, cuya licencia nos permitiera utilizarlos, a ser posible de forma gratuita. Una vez encontrados, se realizó un tratamiento sencillo con Audacity para adaptarlos al formato aceptado por Unreal (.wav), así como a la duración requerida en cada caso.

Una vez hecho esto, utilizamos la función *playSound* para reproducir sonido ambiental (por ejemplo, la lluvia que se oye en el nivel de calma) y *playSoundAtLocation* para sonidos localizados en algún punto (como la llamada a la puerta en el mismo nivel).

6. Pruebas con ejemplos de uso

En este capítulo expondremos las pruebas realizadas sobre nuestra aplicación y trataremos en detalle los diferentes errores y problemas que nos han ayudado a detectar los usuarios que han probado nuestro juego. Lo primero que hicimos, al no disponer de tiempo ni de recursos necesarios para realizar pruebas sistemáticas con un amplio número de usuarios, como nos hubiese gustado, fue crear dos perfiles en los que englobar a los participantes que deseamos tener en esta fase de pruebas.

Las pruebas consistieron en dejar al usuario en el menú de inicio y que por sus propios medios intentara poner en marcha el juego y superar todos los niveles, de forma autónoma. Durante la experiencia no debíamos dar pistas al usuario ni dirigirle, pero sí responderíamos a las preguntas que nos hiciera para evitar bloqueos. También se instó a los usuarios a compartir sus pensamientos durante la duración de la prueba para recopilar esa realimentación. Una vez acabada la experiencia le preguntábamos qué cosas cambiaría y qué cosas le costó identificar y por qué.

A continuación explicamos los dos perfiles diferentes que creamos. Para evitar utilizar nombres reales y violar así la Ley Orgánica de Protección de datos, estos perfiles tendrán una identidad que no coincidirá con su identidad real. Existirá un usuario **básico** de entre 50-60 años sin relación alguna con los videojuegos y un usuario **avanzado** de entre 20-30 años con relación constante con la tecnología y en particular con los videojuegos.

El perfil básico contó con una muestra de 6 usuarios, mientras que del perfil avanzado dispuso de una muestra mayor basada en 14 usuarios.

	**Nombre** Ángel **Apellido** Martín **Edad** 53 años **Sexo** Varón **Nacionalidad:** Español **Tipo Usuario:** Básico		**Nombre** Gonzalo **Apellido** Gonzalez **Edad** 24 años **Sexo** Varón **Nacionalidad:** Español **Tipo Usuario:** Avanzado
Descripción Trabajador a jornada completa con apenas conocimientos tecnológicos en el ámbito de los videojuegos. Interesado en RV por los avances que esta conlleva. Escéptico de la capacidad de inmersión que tendrá con un dispositivo de estas características. Nunca ha probado ningún videojuego ni de ordenador ni consola y tampoco ha tenido experiencia con visores de RV.		**Descripción** Estudiante apasionado por la tecnología. Se mueve en círculos muy técnicos y le despierta interés cualquier tipo de videojuego ya sea de PC, consola o cualquier otra plataforma. Aficionado a la RV, ha probado vídeos 360º con su móvil pero nunca un dispositivo como HTC Vive u Oculus Rift. Le gustaría saber si la RV sería una buena opción de inversión en ocio.	

Tabla 6.1: Perfiles de usuario de pruebas

Una vez definidos los perfiles realizamos las pruebas y registramos las dificultades que ellos han apreciado durante la experiencia. Para una mejor visualización de los resultados, hemos creado una tabla resumen con los fallos o dificultades que detectaron, cuál de los dos perfiles y el porcentaje de los mismos que detectó el fallo y las mejoras que realizamos en base a estas críticas.

Dificultad encontrada	**Mejora realizada**	**Usuario**	**Porcentaje (%)**
No veía objeto clave	Luz encima de estos objetos	Avanzado	60%
Falta Sonido	Agregamos sonido acorde a los niveles	Ambos	80%
No diferenciaba si perdía o no	Agregamos fade a rojo y un sonido cuando no superas el nivel	Ambos	70%
No hay diferencia entre nivel de calma y estrés	Bajamos intensidad de la luz y cambiamos el sonido	Básico	40%
Al caerse las pastillas no podía seguir jugando	Se autogeneran pastillas en caso de que se le caigan	Ambos	15%
No podía agarrar la nota ya que al intentarlo agarraba la mesa	Quitamos que la mesa fuese agarrable y dejamos agarrable únicamente la nota	Avanzado	30%
Al teletransportarse al lado del muro, lo atravesaba	Cambiamos las colisiones del actor y del mesh correspondiente al Wall1, que era el que fallaba	Ambos	15%
Objetos en el suelo imposibles de agarrar	Cambiamos altura de jugador y cambiamos de lugar todos los objetos que aparecían en el suelo	Básico	100%
No tenía feedback si hacía algo en el menú	Marcamos los bordes del botón del menú al pasar por encima	Avanzado	45%
No identificaba fin del nivel de Acrofobia	Bajamos la luz del nivel y agregamos un cartel para mayor visibilidad	Avanzado	25%
No sabía cómo colocar la cama nivel claustrofobia	Agregamos imágenes de cómo superar el nivel en las notas previas al mismo	Ambos	35%
Esfera blanca del cuerpo limitaba la visión	Quitamos la esfera del VivePawn Character	Avanzado	55%
Difícil agarrar objetos	Cambiamos el tamaño de la esfera de los controladores	Básico	30%

Limitación de espacio, chocaban con los objetos físicos de la habitación	Agregamos circunferencia de zona de seguridad azul en el suelo	Ambos	35%

Tabla 6.2: Tabla de dificultades

A continuación, podemos apreciar un gráfica basado en los defectos detectados el perfil de usuario que los detectó y el porcentaje de los mismo.

Tras el experimento podemos identificar que al usuario con experiencia en videojuegos le resultaron bastante intuitivos algunos niveles que en cambio al jugador sin experiencia no le resultaron tan fáciles. Sin embargo, ambos llegaban a conclusiones similares sobre el objetivo de cada nivel.

Por otra parte, el perfil de avanzado se adaptó rápidamente a los controles haciendo los movimientos de manera intuitiva, mientras que el perfil de usuario básico le resultó una tarea más tediosa al principio. También ambos usuarios destacaron la incomodidad del visor ya que les pesaba y les molestaban los cables.

Por último, la limitación de espacio del que disponíamos hacía que el jugador tuviese que estar siendo recolocado fisicamente en el centro de la habitación real de manera constante, cosa que le sacaba de la experiencia que estaba viviendo. Una solución habría sido distribuir la celda en menos espacio para tenerlo todo a mano y no tener necesidad de moverse, pero el planteamiento de puzles como el del nivel de claustrofobia o el de acrofobia no se habría podido utilizar.

A pesar de estas quejas puntuales, podemos decir que a ambos perfiles les pareció una idea novedosa y entretenida. Cabe destacar que muchos de los usuarios recalcaron el realismo del videojuego, algunos de ellos incluso llegó a tambalearse y sentir sensaciones de vértigo durante la experiencia . Por otra parte, una pequeña parte de los usuarios que realizaron las pruebas, nos comentaron que sufrieron mareos debidos a algunos movimientos bruscos de la cámara. Pese a las dificultades que encontraron, algunas de ellas fruto del desconocimiento de la tecnología, les pareció un juego bastante cerrado aunque destacaron aspectos artísticos y de pulido que se podrían mejorar.

7. Conclusiones

Como explicamos durante los primeros capítulos de esta memoria, el objetivo del proyecto era aplicar una metodología ágil durante la creación de un videojuego en Realidad Virtual. Buscábamos experimentar tanto con la tecnología como con la metodología, y aprender lo máximo posible sobre el proceso de desarrollo.

Al finalizar dicho proceso realizamos una reunión de equipo para revisar y confirmar si se han alcanzado estos objetivos. Esta revisión queda recogida en el primer apartado del capítulo. Después reflexionamos sobre la aplicación de esta metodología que hemos adaptado para nuestro proyecto. Finalmente terminamos esta memoria con un repaso a las líneas de trabajo futuro que se abren ante aquel que quiera continuar nuestro trabajo.

7.1. Objetivos específicos alcanzados

Podemos decir que el objetivo general de desarrollar un videojuego en RV con seguimiento de movimiento de visor y controladores manuales ha sido alcanzado de manera satisfactoria. Al final del proceso hemos logrado crear una aplicación software funcional que presenta un juego de terror, ambientado en el mundo de las fobias y las enfermedades mentales. Cuenta con menú inicial, diferentes niveles, una dinámica lineal pero eficaz para irlos superando y unos mecanismos de juego que aunque no estén pulidos del todo, son equiparables a los que usan ahora mismo los videojuegos en RV comerciales, y podrían servir como base para un producto de software de entretenimiento comercial.

Hemos tenido la oportunidad de experimentar con la identificación e integración en el juego de varios movimientos diferentes y posibilidades de interacción, a saber:

- **Observar el entorno en 360º** Tanto con HTC Vive y Oculus Rift, Unreal permite de manera muy sencilla que este mecanismo funcione.
- **Varias formas de navegación** por el entorno, ya que aunque nos hemos decantado por el teletransporte frente a otros métodos, los demás siguen pudiéndose probar en nuestro nivel de *sandbox* pensado para la experimentación de los desarrolladores. Comprobamos que esta interacción resta inmersión a la experiencia, pero las demás formas de navegar por un entorno amplio tienen mayor posibilidad de generar *simulator sickness*.
- **Agarrar objetos** de diferentes formas y tamaños, comprobando que aún se hace complicado diseñar una forma de agarrar objetos demasiado pequeños o finos (como la nota que hay que leer antes de jugar cada puzle, en el nivel de calma).
- **Golpear utilizando una herramienta**, durante el nivel Arachnophobia, comprobando que se puede medir la fuerza con la que se realiza el golpe, proporcionando una sensación bastante inmersiva de violencia y destrucción.
- **Colocación precisa de un objeto** en una posición, como la cama durante el nivel Claustrophobia. Comprobamos que la precisión no es algo exigible al jugador debido a que la sensación sobre el objeto agarrado no se corresponde con la realidad y eso dificulta manejar objetos como lo haríamos en el mundo real.

- **Escalada** durante el nivel Acrophobia. Comprobamos por nosotros mismos y durante las pruebas de usuario que la sensación de vértigo es real, llegando algún usuario incluso a tambalearse en el momento en el que caían al vacío. Este vértigo hace que, aunque no estemos escalando de verdad, la sensación sea realmente inmersiva.

Si bien es cierto que no logramos desarrollar algunas de las interacciones inicialmente planteadas como, por ejemplo la de escalar por una cuerda, nos sentimos satisfechos con el resultado final a tenor de las pruebas realizadas y del proceso exhaustivo de documentación que hemos seguido. Creemos que las interacciones implementadas pueden ser reutilizadas en proyectos futuros y servir de base a videojuegos con mecanismos aún más ambiciosos.

Por último, cabe destacar el conocimiento adquirido sobre RV, una tecnología y un mercado en alza, que complementa muy bien nuestra formación como ingenieros. Si bien es cierto que el uso de visores de RV resulta en una experiencia muy inmersiva, hemos comprobado que las dimensiones y las distancias pueden llegar a distorsionarse. Es recomendable tener esto en cuenta durante el desarrollo para crear una experiencia lo más satisfactoria posible. Llegamos a esta conclusión durante las primeras fases de desarrollo: al implementar la primera celda indicamos en Unreal unas medidas de 2,5x2,5 metros, una habitación que en la realidad resultaría bastante pequeña. Al colocarnos el casco de RV, sin embargo, la sensación era la de estar en una habitación bastante más grande, razón por la cual tuvimos que reducir el tamaño de la misma. Esto es sólo uno de los muchos detalles menores a los que nos fuimos enfrentando a lo largo del desarrollo y que, sin duda, nos han proporcionado una experiencia de la que carecíamos antes.

7.2. Valoración de la metodología

Nos sentimos especialmente contentos del resultado que ha dado la metodología Scrum adaptada a nuestro proyecto. Consideramos que sin haber seguido tan rigurosamente una metodología así, no habría sido posible obtener los resultados que hemos tenido dado todo el aprendizaje que teníamos por delante, los cambios de requisitos sobre la marcha que nos esperaban y el tiempo disponible que había para terminar el proyecto.

Como se puede comprobar en los múltiples anexos, hemos documentado todo el proceso de desarrollo de manera exhaustiva. Esto nos ha permitido revisar dicho proceso una vez finalizado para identificar fortalezas y las debilidades de esta metodología ágil aplicada al desarrollo de videojuegos en RV.

La forma de trabajar de manera iterativa e incremental nos ha permitido tener bastante seguridad a la hora de desarrollar nuestro juego, sobretodo la idea de trabajar en un juego modular, donde cada puzle se correspondiera con un nivel en cierto sentido “independiente”. Así, habiendo comprobado que una parte del desarrollo ya funciona, es más sencillo seguir trabajando y saber que cualquier fallo posterior que afecte al conjunto

de la aplicación estaría ocasionado por ese nuevo incremento en desarrollo y no vendría arrastrado por componentes o características desarrolladas con anterioridad.

Además, la identificación de las diferentes tareas en el documento de EDT, su inclusión en el *backlog* y su correspondiente reparto y asignación dentro de los distintos *sprints* nos ha permitido ir fijando hitos asumibles dentro de los plazos con que contábamos para desarrollar. Este misma reparto permitía también una fácil gestión del equipo de trabajo, ya que cada miembro conocía en todo momento lo que estaban haciendo sus compañeros y no sufría excesivas interrupciones por su parte. Las reuniones de Scrum que realizábamos dos veces por semana nos ayudaban a evitar bloqueos y a compartir ideas que pudimos ir reutilizando unos de otros a la hora de incorporarlas en cada nivel.

7.3. Trabajo futuro

Como se puede ver si se acude a la especificación de requisitos, no hemos llegado a tiempo para cumplir algunos de los requisitos secundarios que nos fijamos como "futuribles" del desarrollo. Como trabajo futuro faltaría por terminar la narración de la historia imaginada para el protagonista de la experiencia, el paciente número 52, así como añadir en el nivel de Calma algunos objetos que ayuden a narrar esta historia, como por ejemplo las cartas del paciente número 50. Nuestro objetivo principal de desarrollar una aplicación software completamente funcional ha quedado satisfecho, pero consideramos que podría completarse con una producción audiovisual completa que aporte más valor a la trama narrativa.

Por otra parte, durante las pruebas con usuarios alguno de ellos comentó que en el nivel de Claustrophobia deberían poder pararse las paredes siempre que la cama se atravesara entre dos de ellas, no única y exclusivamente de la manera concreta implementada, lo cual es una mejora en la especificación que sería interesante realizar. Cabe destacar que esto se contempló durante la fase de diseño, pero optamos por desarrollar un prototipo funcional que explicase el mecanismo del juego. Es posible replicar este prototipo para las múltiples posibilidades de bloqueo de paredes.

Muchas de estas mejoras y arreglos ya se contemplaron durante la fase de especificación con la definición de niveles, pero se optó por priorizar el cierre de los aspectos más fundamentales del software, para que este estuviese cerrado y funcional de cara a la fecha de finalización del proyecto. Quede la siguiente lista a modo de guía para futuros desarrollos:

- Profundizar en la narrativa y añadir nuevos elementos artísticos que completen la experiencia, tales como assets más personalizados o sonidos creados desde cero.
- Posibles mejoras en los distintos niveles.
    - Arachnophobia: Añadir más puntos de generación de arañas. Hacer que las arañas sigan al jugador, subiendo por su cuerpo hacia la cabeza.
    - Claustrophobia: la cama podría bloquear las paredes en más posiciones que la actual para hacer más intuitiva la resolución del puzzle.

    - Acrophobia: Podrían implementarse más puntos de salida que el punto único actualmente colocado.
- Animaciones entre los niveles que hagan más obvias las condiciones de derrota y victoria.
- Diseño e implementación de nuevos niveles con nuevos movimientos.
- Investigación, diseño e implementación de interfaces en Realidad Virtual mejoradas, tales como menús dentro del juego. Investigación necesaria, ya que los botones de los controladores se necesitan o bien para andar (botones centrales), agarrar objetos (gatillos traseros) o abrir el menú predeterminado de la plataforma SteamVR.
- Mejora del sistema de notas para que no se vean tan borrosas (necesario diseño de arte). Posible implantación de historial donde almacenar las notas, de manera que el usuario pueda revisarlas de una vez y hacerse una mejor idea de la historia que cuentan.
- Pruebas con usuarios más detalladas y avanzadas.
- Producción real y publicación de la experiencia.

APORTACIÓN INDIVIDUAL

Alejandro Blázquez

El desarrollo del trabajo de fin de grado consta de diferentes áreas, en cada una de las cuales mis compañeros y yo hemos trabajado y aportado para conseguir que el proyecto saliese adelante. A continuación, explicaré las partes del trabajo que he desarrollado y el cometido de las mismas por orden de realización.

Al principio del proyecto nos centramos en la documentación del mismo. Debido a la gran cantidad de puntos a tratar, decidimos separar la introducción y el estado de la técnica. De esta parte, mi cometido fue crear la introducción de esta memoria tanto en castellano como en inglés y buscar las posibles aplicaciones positivas y negativas de la Realidad Virtual en nuestro día a día.

En cuanto a la introducción, me decanté por tomar como referencia una de las mejores definiciones de Realidad Virtual que pude encontrar, e investigué sobre la historia de la tecnología desde sus inicios con los primitivos cascos de RV como la Espada de Damocles hasta la actualidad. A continuación, hablé del marco económico y social que englobaba a esta tecnología. Al tratarse el proyecto de realizar un videojuego narrativo basado en fobias, teníamos que explicar el porqué de realizar un videojuego de estas características y las motivaciones que nos llevaban a ello, esto es lo que podemos encontrar en los dos siguientes puntos.

Parar acabar con la introducción, debíamos tener en cuenta qué competencias de las estudiadas durante nuestra formación podríamos aplicar a nuestro trabajo. Para ello decidí investigar dichas competencias en la página oficial de la UCM y englobar algunas de ellas en el ámbito de nuestro Trabajo Fin de Grado.

Una vez finalizada la introducción me dispuse a buscar las aplicaciones positivas y negativas de la RV. Para ello, investigué acerca de los problemas o enfermedades que produce el uso prolongado de está tecnología y por otra parte, los beneficios que podría tener la RV en diferentes ámbitos de nuestra vida como por ejemplo la sanidad o la educación.

Una vez acabada la documentación acerca del estado de la técnica, comenzamos el desarrollo del software. Tras finalizar la lluvia de ideas general con el equipo y tener claro el objetivo final del software a desarrollar, lo primero que hice fue estudiar las diferentes metodologías de desarrollo que estaban a nuestro alcance. Tras la investigación decidimos utilizar la metodología Scrum.

Definida la metodología, mi cometido fue adaptar dicha metodología a nuestras necesidades generando una pequeña introducción explicando desde el manifiesto ágil hasta los diferentes pasos para llevar a cabo de forma correcta el proceso de desarrollo software. Toda la información teórica de Scrum adaptada a nuestro proyecto, la podremos encontrar en el ANEXO IV. Todo esto lo compaginé con la formación en Unreal Engine

necesaria para conocer el entorno de desarrollo antes de ponernos a crear el videojuego en sí.

A continuación, ya que ahora si conocíamos la teoría, era hora de empezar a aplicarla. En este punto me encargué de generar las plantillas de Scrum que utilizamos para los diferentes sprints y que podemos encontrar en el ANEXO V. En este punto, ya teníamos los documentos necesarios para generar sprints y poder empezar a desarrollar. Sin embargo, faltaba por crear la documentación previa como eran: los roles que desempeñamos cada uno de nosotros en el proyecto, tabla de riesgos y plan de contingencia y mitigación los cuales me encargue de generar.

Una vez creé toda esta documentación, me puse con el desarrollo del nivel Claustrophobia. Me encargué de todo lo relacionado con este nivel, desde agregar la funcionalidad inicial más básica hasta integrar el dispositivo de Realidad Virtual. A la vez que desarrollaba el nivel, iba desarrollando la documentación asociada al mismo.

Dicha documentación, consiste en el sprint de funcionalidad, el sprint de arte, el sprint de sonido y el sprint de integración de realidad virtua del nivel de Claustrophobia. Por otra parte también creé la documentación en la que se explicaba la programación de dicho nivel, desde el nivel de desarrollo hasta lo que a día de hoy podemos ejecutar y jugar.

Para acabar con los sprints, también documenté todo lo relacionado con el sprint de integración de RV y niveles, es decir la carga hilada de niveles que a día de hoy podemos ver en nuestro videojuego. Investigué acerca de los menús en RV y llevé a cabo el nivel en el que se integra el menú principal.

La documentación asociada al desarrollo de todos los sprints comentados hasta el momento podemos encontrarla en el ANEXO VI. En este anexo encontraremos todos los detalles de cada uno de los sprints explicados hasta este punto.

Por otra parte, en cuanto la programación final, me dediqué a agregar las últimas mejoras que nos comentó nuestro director en reuniones previas a la entrega del presente documento.

Por último, me encargué de ir registrando las actas de las diferentes reuniones periódicas que tuvimos con nuestro director y entre nosotros a lo largo del tiempo y generar las pruebas con usuarios, que podemos encontrar en el capítulo 6 de esta memoria.. Podemos ver todas las actas de reuniones en el ANEXO III.

Carlos Casado

Durante prácticamente la totalidad del proyecto uno de los principales cometidos de los integrantes ha sido la repartición justa y equitativa del trabajo, adecuándose a nuestro calendario, horario y objetivos del proyecto.

Cabría diferenciar varias partes del desarrollo, como una parte de lluvia de ideas y organización, que normalmente tenía lugar en reuniones del equipo de carácter más o menos informal durante el inicio del proyecto.

El desarrollo se ha visto limitado por la falta de material para cada uno de los integrantes, ya que en un principio sólo yo tenía un computador capaz de bregar con las exigencias de la RVI. De esta manera, empecé a trabajar en la toma de contacto con Unreal Engine. Para ello, realicé durante las semanas anteriores una formación intensiva de los cursos que el propio Epic ofrece en su web, además de algunas recomendaciones del director de nuestro proyecto y cuando creí que estaba preparado para trabajar con soltura con Unreal Engine empecé a sumergirme en la RVI, continuando con la investigación de una tecnología joven y vagamente documentada.

En este punto la investigación y formación constante en Unreal Engine y RV fue solapandose con el desarrollo de un primer prototipo de nivel de prueba y los mecanismos de juego, interacciones y principales objetos que previsiblemente podrían llegar a utilizarse en nuestro juego.

Así, pasaría a desarrollar el nivel de Sandbox incluido en el juego, con un arte muy básico y centrándome más en el desarrollo de las interacciones, el uso del primer plugin de RV (aunque posteriormente migraríamos a otro), el movimiento del personaje, la creación de objetos agarrables, la interacción con el Pawn y la escalada.

Algunos de los problemas que se resolvieron desarrollando blueprints o actores útiles fue la creación de objetos que pudiesen ser agarrados por el usuario, la interacción de esos objetos con otros estáticos para provocar su rotura parcial (para lo cual tuve que migrar a un plugin abierto creado por la comunidad), la creación de componentes agarrables incluídos en actores estáticos, la creación de cuerdas elásticas con físicas realistas con extremos agarrables, un sistema de notas de texto para leer dentro del juego y la alteración de la visión del usuario cuando se dispara un determinado evento.

Tras este trabajo, los componentes del grupo pasamos a trabajar en paralelo centrados, esta vez sí, en los niveles del videojuego propiamente dicho. Para ello nos organizamos asignando un nivel jugable a cada uno, siendo en mi caso el nivel Acrophobia, en el que el suelo debía romperse y ofrecer una vista impactante desde lo alto de una cala escarpada. Cabe destacar que en esta fase trabajamos sin HTC Vive, porque dejamos de tener acceso a él varias semanas, haciendo obligatorio tras el desarrollo de los niveles un sprint de integración de RV que se produciría bastante más tarde.

Para realizar este nivel, de nuevo tuve que realizar algunas investigaciones sobre la aplicación de fuerzas y daño en objetos rompibles y cómo conseguir una rotura de una capa parcial y gradual, así como la creación de un ambiente efectivo incluyendo assets

externos y la aplicación de brumas, pájaros o sonidos propio de un mar bravío, un grupo de gaviotas o la demolición de un edificio, así como ajustar los gráficos para mejorar el rendimiento, ya que el mar era una carga dinámica de trabajo especialmente pesada para el motor gráfico.

En la última parte del desarrollo, correspondiente a la integración en RV y de los niveles que inicialmente estaban separados, la ubicación del dispositivo HTC Vive (y en algún momento Oculus Rift) pasó a instalarse en casa de otro de los integrantes del equipo, por lo que la mayor parte de la carga de trabajo sobre RV recayó sobre él.

Mi contribución consistió en la aportación de ideas, el desarrollo puntual de ciertos apartados y algunas labores de diseño gráfico, como las fotos que llevan las notas de texto.

Antes de empezar el desarrollo del juego y durante la fase de reuniones e investigación documenté la parte del estado de la técnica correspondiente al hardware disponible en la realidad virtual, es decir, visores, controladores, sensores y otros elementos de hardware y productos relacionados con la tecnología, así como la documentación correspondiente a los entornos de desarrollo de videojuegos más populares.

También he realizado y documentado los sprints correspondientes a mi parte de análisis, diseño e implementación, es decir lo correspondiente al Sandbox y al nivel Acrophobia, junto con aportaciones puntuales en función de donde se necesitaba para completar el proyecto, como las palabras clave o la creación del diagrama EDT.

Juan Antonio Palacios

Durante el desarrollo de un proyecto de estas características es fundamental el trabajo en equipo, por lo que era necesario hacer un reparto de tareas justo y equitativo. Si bien es cierto que hay tareas de obligada realización por parte de todos los miembros del equipo, como puede ser la formación en unas tecnologías anteriormente desconocidas para nosotros, una parte muy importante del trabajo se ha podido repartir entre los distintos integrantes.

Para poder desarrollar algo innovador y también para conocer las buenas y malas prácticas asociadas a esta tecnología, era necesario hacer una revisión profunda del estado de la técnica. En este punto identificamos algunas subdivisiones importantes: Historia de la Realidad Virtual, actualidad de la misma como tecnología, y aplicaciones software desarrolladas hasta el momento. En mi caso, fui el encargado de investigar este último punto, dando como resultado una suerte de reportaje sobre los videojuegos relacionados con nuestro proyecto de alguna manera o de otra. Como se podrá observar leyendo esta memoria, nos encontramos con muchísima información, por lo que también me encargué de organizarla adecuadamente y de elaborar unas conclusiones por escrito en base al debate que mantuvimos mis compañeros y yo tras finalizar esta fase del estudio previo.

Durante la fase de especificación inmediatamente posterior a la lluvia de ideas en común con el grupo, me encargué de realizar la poda de requisitos necesaria para tener unos objetivos más claros. Confirmados estos requisitos con mis compañeros y con el director y el co-director del Trabajo Fin de Grado, pasé a diseñar las condiciones de victoria y derrota de los diferentes niveles mientras llevaba a cabo la formación necesaria en el entorno de desarrollo Unreal Engine. En concreto, estudié mucho acerca del manejo de luces y la animación de objetos, aparte de la formación imprescindible en programación mediante el uso de los blueprints. El resultado de esta fase de diseño quedó plasmado en el documento de diseño que llevamos a cabo con ayuda del director.

Tras esta fase de diseño, nos repartimos la implementación de los niveles de manera equitativa. Busqué distintas opciones para el control de versiones pero, al disponer sólo de un casco de RV decidimos que era más sencillo seguir usando Google Drive para guardar copias de seguridad de lo que hacíamos cada uno individualmente, al menos hasta la integración. A esta plataforma íbamos subiendo los diferentes desarrollos que más tarde integramos en un único proyecto de Unreal Engine.

Como se ve en la memoria, el diseño del juego se dividió en varios sprints según la metodología Scrum. Al tratarse de tres niveles bien diferenciados, nos encargamos cada uno de un nivel. Habiendo estudiado, como he dicho bastante, sobre iluminación, era lógico que, además del nivel Arachnophobia que se me asignó, me encargara también del diseño gráfico final de la celda. Por este motivo tuve que encargarme también de encontrar en las distintas tiendas online de recursos gráficos el mobiliario de la celda que se puede ver hoy en el proyecto terminado. Este escenario, con mobiliario incluido, fue el que se replicó después en todos los niveles. Me encargué también de documentar todo el proceso de desarrollo del nivel Arachnophobia, incidiendo en todos los detalles técnicos como las trazas y las funciones implementadas para el movimiento de las arañas.

Tras el sprint de funcionalidad, el director nos pudo proporcionar un dispositivo HTC Vive en exclusiva para nuestro proyecto, que finalmente se quedó en mi casa por motivos de espacio y de proximidad a las viviendas del resto de los integrantes del grupo. Acondicionamos un espacio para las pruebas que más tarde llevaríamos a cabo y pasamos a integrar la funcionalidad desarrollada en los niveles. Como no podía ser de otra manera, comencé añadiendo todo lo necesario al nivel Arachnophobia, del que era responsable, pero acabé aplicando varios cambios también al nivel Acrophobia.

Por último, investigué acerca de la carga de niveles, llegando a conocer varias opciones diferentes para lograr la carga secuenciada de los mismos. Finalmente me decidí por utilizar la clase VRGameInstance que comentamos en el punto 5.4.1.

Durante la fase de pruebas, por encontrarse el HTC Vive en mi casa, el peso de la adaptación del entorno recayó sobre mí, así como dirigir el desarrollo de las mismas. Llevé a cabo varias pruebas con usuarios, recopilando los datos sobre los fallos encontrados y las posibles mejores a llevar a cabo, aunque finalmente fue Alejandro quien llevó a cabo estas mejoras e implementó las correcciones que se pedían.

En cuanto a la documentación, fui el encargado de compilar y organizar la gran cantidad de información de la que disponíamos e incluirla en el presente documento, de manera que fuese lo más legible posible. Realicé la revisión ortográfica pertinente de todo lo que teníamos escrito para asegurar que el documento resultaba lo más profesional posible.

BIBLIOGRAFÍA Y REFERENCIAS UTILIZADAS

Abrash, M (2014). What VR Could, Should, and Almost Certainly Will Be Within Two Years. Retrieved June 10, 2016, from http://media.steampowered.com/apps/abrashblog/Abrash%20Dev%20Days%202014.pdf

Abubakra, Y., Herrero, M. A., Fernández, J., Sanz, D. M., & Zabala, A. (2015). Producción de una experiencia de horror inmersivo utilizando un entorno de desarrollo de videojuegos y un visor de realidad virtual . Retrieved June 10, 2017, from http://eprints.ucm.es/32965/1/TFG%20Producci%C3%B3n%20de%20una%20experiencia%20de%20horror%20inmersivo%20utilizando%20un%20entorno%20de%20desarrollo%20de%20videojuego.pdf

Berger, Andrea, Karpel, Patricia, Lejbowicz, Jacqueline, & Racki, Gabriel. (2013). Adicciones virtuales: efectos de conex-ficc-ión y retracción. Anuario de investigaciones, 20(2), 59-65. Recuperado en 09 de junio de 2017, de http://www.scielo.org.ar/scielo.php?script=sci_arttext&pid=S1851-16862013000200007&lng=es&tlng=es

Botella Arbona, C., García-Palacios, A., Baños Rivera, R.M. & Quero Castellano, S. (2007). Realidad Virtual y Tratamientos Psicológicos. Virtual Reality and Psychological Treatments.

Earnshaw, R. A. (Ed.). (2014). Virtual reality systems. Academic press

Fino, E. R., Martín-Gutiérrez, J., Fernández, M. D., & Davara, E. A. (2013). Interactive Tourist Guide: Connecting Web 2.0, Augmented Reality and QR Codes. Procedia Computer Science, 25, 338-344. doi:10.1016/j.procs.2013.11.040 ISSN 1877-0509

Haas, J. (n.d.). A History of the Unity Game Engine. Retrieved June 10, 2017, from https://web.wpi.edu/Pubs/E-project/Available/E-project-030614-143124/unrestricted/Haas_IQP_Final.pdf

Luckey, P. (2008, August 21). Oculus "Rift" : An open-source HMD for Kickstarter. Retrieved June 09, 2017, from http://www.mtbs3d.com/phpbb/viewtopic.php?f=140&t=14777

Olson, E. (2011). AprilTag: A robust and flexible visual fiducial system. Proceedings of the IEEE International Conference on Robotics and Automation. Retrieved June 10, 2017, from https://april.eecs.umich.edu/media/pdfs/olson2011tags.pdf

Patel, N., & Cober, D. (n.d.). Adjacent Reality. Retrieved June 09, 2017, from http://adjacentreality.org/

Rodríguez-García, J. I. (2006). Formación quirúrgica con simuladores en centros de entrenamiento. Surgical training with simulators in training centers.

Steuer, J. (1992) Defining Virtual Reality: Dimensions Determining Telepresence Steuer,

Sutherland, I. E. (1965). The ultimate display. Proceedings of the IFIPS, 2, 506–508

NOTICIAS E INFORMACIÓN VARIADA

Chaperone. (n.d.). Retrieved June 09, 2017, from https://xinreality.com/wiki/Chaperonehttps://www.playstation.com/es-es/explore/accessories/playstation-camera/

Constellation, Información para desarrolladores del sistema de sensores de Oculus Rift (n.d.). Retrieved June 9, 2017, from https://developer.oculus.com/documentation/pcsdk/latest/concepts/dg-sensor/#dg-sensor-position-tracking

Crystal Cove Debut, CES Recap, and Steam Dev Days. (n.d.). Retrieved June 10, 2017, from https://www3.oculus.com/en-us/blog/crystal-cove-debut-ces-recap-and-steam-dev-days/

Game Development Conference. (n.d.). Retrieved June 09, 2017, from http://www.gdconf.com/

Giokaris, P. (2014, January 16). Bringing VR to Life. Retrieved June 09, 2017, from http://www.gamasutra.com/blogs/PeterGiokaris/20140116/208779/Bringing_VR_to_Life.php?print=1

Google And IMAX Partner To Develop Groundbreaking Virtual Reality Camera. (2016, May 19). Retrieved June 09, 2017, from https://www.imax.com/content/google-and-imax-partner-develop-groundbreaking-virtual-reality-camera

Google Daydream View (n.d.). Retrieved June 9, 2017, from https://madeby.google.com/vr/

Google Spotlight Stories. (n.d.). Retrieved June 09, 2017, from https://atap.google.com/spotlight-stories/

HTC and Valve Partner to Make Virtual Reality Dream Come True. (2015, March 1). Retrieved June 09, 2017, from http://www.htc.com/us/about/newsroom/2015/2015-03-01-htc-valve-partner-to-make-virtual-reality-come-true/

HTC Vive Motion Controllers (n.d.). Retrieved June 9, 2017, from https://www.vive.com/us/support/category_howto/about-the-controllers.html

Información para desarrolladores de Oculus Touch (n.d.). Retrieved June 9, 2017, from https://developer.oculus.com/documentation/pcsdk/1.11/concepts/dg-input-touch-overview

Información sobre precios y condiciones de Unity. (n.d.). Retrieved June 09, 2017, from https://store.unity.com/es/

Informe The App Date fundación telefónica (n.d.). Retrieved June 09, 2017, from http://www.theappdate.es/static/media/uploads/realidadvirtualespana.pdf

Lighthouse, Manual de usuario de HTC Vive (n.d.). Retrieved June 9, 2017, from http://dl4.htc.com/web_materials/Manual/Vive/Vive_User_Guide.pdf?_ga=1.221562200.1506117373.1474511072

Mobile World Congress. (n.d.). Retrieved June 09, 2017, from https://www.mobileworldcongress.com/

Oculus at E3 2013. (2013, June 11). Retrieved June 10, 2017, from https://www3.oculus.com/en-us/blog/oculus-at-e3-2013/

Oculus Connect 2014. (2014, September 20). Retrieved June 10, 2017, from https://www3.oculus.com/en-us/blog/oculus-connect-2014/

Oculus Rift VR Motion Sickness - 11 Ways to Prevent It! (2015, December 25). Retrieved June 10, 2017, from http://riftinfo.com/oculus-rift-motion-sickness-11-techniques-to-prevent-it

WEBS OFICIALES

A Chair in a Room. (n.d.). Retrieved June 09, 2017, from http://www.achairinaroom.com/

Audacity. (n.d.). Retrieved June 09, 2017, from http://www.audacityteam.org/

Blender. (n.d.). Retrieved June 09, 2017, from https://www.blender.org/

Budget Cuts. (n.d.). Retrieved June 09, 2017, from http://www.neatcorporation.com/BudgetCuts/

CryEngine. (n.d.). Retrieved June 09, 2017, from https://www.cryengine.com/

Google I/O 2016. (n.d.). Retrieved June 09, 2017, from https://events.google.com/io2016/

Google CardBoard (n.d.). Retrieved June 9, 2017, from https://vr.google.com/cardboard/

PlayStation Move (n.d.). Retrieved June 9, 2017, from https://www.playstation.com/es-es/explore/accessories/playstation-move-motion-controller

PlayStation VR (n.d.). Retrieved June 9, 2017, from https://www.playstation.com/es-es/explore/playstation-vr/

Project Morpheus (n.d.). Retrieved June 9, 2017, from https://www.playstation.com/es-es/explore/ps4/features/projectmorpheus/

Proyecto SICEMAM. (n.d.). Retrieved June 09, 2017, from http://politecnicavila.usal.es/uploaded/SICEMAM.pdf

Sanitas en colaboración Psious. (n.d.). Retrieved June 09, 2017, from https://sanitas.psious.com/

Trello. (n.d.). Retrieved June 09, 2017, from https://trello.com/

DK2. (n.d.). Retrieved June 10, 2017, from https://www.oculus.com/ja/dk2/

DOOM. (n.d.). Retrieved June 09, 2017, from http://doom.com/es-es/

Emily Wants To Play. (n.d.). Retrieved June 09, 2017, from http://www.emilywantstoplay.com/

EVE: Online. (n.d.). Retrieved June 09, 2017, from https://www.eveonline.com

Fantastic Contraption. (n.d.). Retrieved June 09, 2017, from http://fantasticcontraption.com/

Fove VR (n.d.). Retrieved June 9, 2017, from https://www.getfove.com/

GooBall. (n.d.). Retrieved June 09, 2017, from http://www.ambrosiasw.com/games/gooball/

Google Drive. (n.d.). Retrieved June 09, 2017, from https://www.google.com/intl/es/drive/.

Google Glass (n.d.). Retrieved June 9, 2017, from https://developers.google.com/glass/distribute/glass-at-work

Holo Lens (n.d.). Retrieved June 9, 2017, from https://www.microsoft.com/en-us/hololens

I expect you to die. (n.d.). Retrieved June 09, 2017, from https://iexpectyoutodie.schellgames.com/

Oculus DK1. (n.d.). Retrieved from https://www1.oculus.com/order/dk1/

Oculus Dream Deck. (n.d.). Retrieved June 09, 2017, from https://www.oculus.com/experiences/rift/941682542593981/

Oculus Rollercoaster. (n.d.). Retrieved June 09, 2017, from http://www.oculusriftrollercoaster.com/wp-content/uploads/2014/05/castle.png

Open Source VR (n.d.). Retrieved June 9, 2017, from http://www.osvr.org/

Outlast 2. (n.d.). Retrieved June 09, 2017, from https://redbarrelsgames.com/games/outlast-2/

Photoshop. (n.d.). Retrieved June 09, 2017, from http://www.photoshop.com/

Please, don't touch anything. (n.d.). Retrieved June 09, 2017, from https://www.oculus.com/experiences/rift/1088797664494908/

Psious. (n.d.). Retrieved June 09, 2017, from https://psious.com/

Skype. (n.d.). Retrieved June 09, 2017, from https://www.skype.com/es/

Slack. (n.d.). Retrieved June 09, 2017, from https://slack.com/

SoftKinetic (n.d.). Retrieved June 9, 2017, from https://www.softkinetic.com/

Star VR (n.d.). Retrieved June 9, 2017, from http://www.starvr.com

Super Mario Bross. (n.d.). Retrieved June 09, 2017, from https://www.nintendo.es/Juegos/NES/Super-Mario-Bros--803853.html

Surgeon Simulator. (n.d.). Retrieved June 09, 2017, from http://www.surgeonsim.com/

Technolust. (n.d.). Retrieved June 09, 2017, from http://store.steampowered.com/app/379670/Technolust/

The Dark Inside. (n.d.). Retrieved June 09, 2017, from https://play.google.com/store/apps/details?id=com.tenprintgames.thedarkinside&hl=es_419

The Gallery: Call of the Starseed. (n.d.). Retrieved June 09, 2017, from https://www.oculus.com/experiences/rift/1041269642652957/

The Lab. (n.d.). Retrieved June 09, 2017, from http://store.steampowered.com/app/450390/The_Lab/?l=spanish

Tilt Brush. (n.d.). Retrieved June 09, 2017, from https://www.tiltbrush.com/

Unity. (n.d.). Retrieved June 09, 2017, from https://unity3d.com/es

Universe Sandbox 2. (n.d.). Retrieved June 09, 2017, from http://universesandbox.com/

Unreal Engine. (n.d.). Retrieved June 09, 2017, from https://www.unrealengine.com/what-is-unreal-engine-4

Voluntechies. (n.d.). Retrieved June 09, 2017, from https://www.voluntechies.org/

VR Cinema. (n.d.). Retrieved June 09, 2017, from https://thevrcinema.com/

WhatsApp. (n.d.). Retrieved June 09, 2017, from https://www.whatsapp.com/?l=es

Windows Office (Excel). (n.d.). Retrieved June 09, 2017, from https://products.office.com/es-es/excel

Wolfenstein. (n.d.). Retrieved June 09, 2017, from http://3d.wolfenstein.com/

Youtube. (n.d.). Retrieved June 09, 2017, from https://www.youtube.com/

Samsung Gear VR (n.d.). Retrieved June 9, 2017, from http://www.samsung.com/global/galaxy/gear-vr/

Rift Experiences | Oculus. (n.d.). Retrieved June 09, 2017, from https://www.oculus.com/experiences/rift/

Valve. (n.d.). Retrieved June 09, 2017, from http://www.valvesoftware.com/

Vive | Discover Virtual Reality Beyond Imagination. (n.d.). Retrieved June 09, 2017, from https://www.vive.com/eu/

Vuzix (n.d.). Retrieved June 9, 2017, from https://www.vuzix.com/