Escuela Politécnica Superior

Diseñador de mazmorras

Grado en Ingeniería Multimedia

Trabajo Fin de Grado

Autor:
Beatriz Sabater Serna

Tutor/es:
Mireia Luisa Sempere Tortosa

Septiembre 2017

Universitat d'Alacant
Universidad de Alicante

Motivación

Desde que empecé a jugar a rol de manual, en el cual podía tanto inventar como vivir historias fantásticas en las que un grupo de aventureros emprendía un viaje y superaba distintas situaciones, me había planteado el cómo sería si algún día pudiese ver todas esas mazmorras y malvadas criaturas delante de mí en vez de tener que describirlas con mi voz y unos cuantos gestos delante de una hoja de papel y unos dados.

Cuando tuve la oportunidad de probar la primera versión de las Oculus Rift, supe por primera vez que todo aquello podría hacerse realidad algún día mediante la realidad virtual y una versión adaptada del rol de manual en forma de videojuego. En éste lo importante no sólo sería que se pudiese encarnar a un personaje y recrear con él una aventura; si no que los directores de juegos pudiesen diseñar infinitas mazmorras y presentarlas tanto a sus jugadores como al resto del mundo.

A medida que avanzaba en el grado de Ingeniería Multimedia y veía cómo mis capacidades de desarrollar videojuegos iban incrementándose, pensé que el trabajo de fin de grado sería una buena oportunidad para poder crear un prototipo de esta idea, la cual ya no me parecía tan lejana. Sabía que era un proyecto grande, ambicioso y que no podría implementar todas las funcionalidades que me había imaginado que tendría, ya que lo desarrollaría yo sola y dispondría de recursos y tiempo limitados.

Paralelamente estuve planteándome diversas preguntas sobre cómo quería desarrollarlo, qué quería conseguir con su elaboración, los objetivos a alcanzar… Al final llegué a la conclusión de que quería que fuese una demostración de mi aptitud actual en lo que respecta a la creación de videojuegos con el fin de que alguna empresa pudiese interesarse por mis habilidades o el proyecto. También quería mostrar originalidad, pasión y personalidad; por lo que al final me decanté por este proyecto.

Aun con todo esto quise plantearme además un reto: aprender algo nuevo que me sirviese para el mundo laboral. Es por esto que al final me decanté por usar Unreal Engine 4, ya que este motor es uno de los más populares actualmente y muchas empresas empiezan a requerir que tengas conocimientos sobre el motor para poder trabajar con ellas. Pensé que el trabajo de fin de grado sería una buena oportunidad para poder familiarizarme con él.

Al final la lista de objetivos se fue engrosando, pero tenía claro aquellos que eran prioritarios: avanzar todo lo posible en esta primera versión del *Diseñador de mazmorras*, aprender a utilizar el motor de videojuegos Unreal Engine 4 y sentirme orgullosa cuando muestre el resultado final ante el tribunal.

Dedicatoria

Este trabajo de fin de grado no es un simple proyecto más, es la culminación de varios años de estudio y la exposición de lo que he aprendido durante ellos. Nada de esto hubiese sido posible si mis padres no hubiesen confiado en que era capaz de enfrentarme a este reto; gracias por apoyarme y darme las facilidades para poder estudiar aquello que me gusta.

A mi tutora Mireia y a mis compañeros de carrera, en especial Albert por haberme ayudado durante este proyecto respondiéndome las dudas de Unreal Engine 4.

A mi hermano, a mi yaya y a mi perro Mordisquitos por hacerme compañía durante un gran periodo de mi vida.

Y en especial agradezco a mi pareja Cristina el acompañarme durante este largo recorrido, darme apoyo cuando más lo necesitaba y aguantar mis quejas y enfados; gracias por estar ahí pasase lo que pasase.

Índice de contenido

Tabla de contenido

Índice de figuras

Índice de tablas

Terminología

- **API:** siglas de la expresión inglesa *application programming interface* (interfaz de programación de aplicaciones). Conjunto de subrutinas, protocolos y herramientas para desarrollar software que otorga al programador facilidades.
- **Blueprint:** lenguaje de programación visual propio de Unreal Engine 4.
- **CPU:** siglas de la expresión inglesa *central processing unit* (unidad central de proceso).
- **Direct X o DX:** colección de API desarrolladas para facilitar tareas complejas relacionadas con los archivos multimedia en Windows.
- **Director de juego:** figura que controla la narrativa en una aventura de rol de manual.
- **DLL:** siglas de la expresión inglesa *dynamic-link library* (biblioteca de enlace dinámico). Ficheros que contienen código ejecutable que se cargan bajo demanda.
- **GPU:** siglas de la expresión inglesa *graphics processor unit* (unidad de procesamiento gráfico).
- **Mod:** extensión de software que modifica un videojuego original añadiéndole nuevas características y funciones.
- **Motion blur:** rastro dejado por los objetos en movimiento.
- **Renderizado:** proceso de generar una imagen o vídeo mediante el cálculo de iluminación partiendo de un modelo 3D.
- **Rol de manual:** un juego de mesa en el que los participantes encarnan personajes ficticios que actúan como sus alter egos en una aventura o historia.
- **SDK:** siglas de la expresión inglesa *software development kit* (kit de desarrollo software). Conjunto de herramientas que permite al desarrollador crear una aplicación informática.
- **Shader:** programa que se emplea para los cálculos de iluminación y percepción 3D.
- **Soft shadows**: sombras difuminadas que se producen por una fuente de luz lejana.
- **Unreal Engine 4:** motor de videojuegos empleado en este proyecto.

1. Introducción

1.1. Planteamiento inicial

El rol de manual es un juego de mesa en el que los participantes encarnan personajes ficticios que actúan como sus alter egos en una aventura o historia contada por un director de juego. Esta persona es la encargada de dirigir la narrativa, describir las situaciones que envuelve a los protagonistas y crear las mazmorras y aventuras.

El objetivo de este Trabajo de Fin de Grado es adaptar este tipo de juegos a un videojuego en el que se pueda interactuar mediante un equipo de realidad virtual. Para conseguir esto, se crearán dos aplicaciones que estarán integradas en el mismo programa: una interfaz de creación de mazmorras para los directores de juego y un juego de aventuras estilo *Skyrim* en el que se jugará la mazmorra que previamente ha sido creada.

El creador de mazmorras será un programa en el que se podrá diseñar una mazmorra utilizando una dinámica parecida a la creación de casas de *The Sims*, que consta de distintos menús en los que se pueden seleccionar distintos elementos como paredes, suelo, objetos de decoración o muebles.

La parte jugable constará de una cámara en primera persona mediante la que se podrá ver tanto en realidad virtual como sin ella la mazmorra creada sin ninguna mecánica implementada, ya que no es el objetivo de este proyecto. Aun así se creará la estructura necesaria para que estas mecánicas puedan desarrollarse en un futuro.

1.2. Objetivos

Aunque el objetivo inicial del proyecto era el descrito en el anterior apartado, surgieron nuevos propósitos a medida que fui avanzando en el diseño del proyecto. Comencé a plantearme qué herramientas emplearía para poder desarrollarlo y decidí analizarlas para poder tomar una buena decisión.

Expuesto lo anterior, a continuación listo los objetivos concretos finales:

- Realizar un estudio sobre los distintos motores gráficos y entornos de desarrollo actuales, su compatibilidad con distintos hardware de Realidad virtual, su historia y elegir de entre ellos el más adecuado para usar en este proyecto.

- Analizar las distintas tecnologías actuales de Realidad Virtual, su compatibilidad con los motores gráficos estudiados, su historia y elegir de entre ellas la más adecuada para este proyecto.
- Aprender a utilizar una nueva herramienta o ampliar mis conocimientos sobre ella.
- Desarrollo del creador de mazmorras.
- Desarrollo de la parte jugable en la que el jugador podrá utilizar un hardware de realidad virtual y moverse por la mazmorra diseñada.
- Desarrollo de un sistema de cargador de niveles que permita jugar las distintas mazmorras creadas.

1.3. Estructura del documento

- **Introducción**: incluye la descripción del proyecto, los objetivos a alcanzar, la estructura del documento y los referentes del diseño del videojuego.
- **Metodologías y herramientas**: explicación del flujo de trabajo empleado durante el proyecto y resumen de las herramientas y estación de trabajo utilizados.
- **Diseño del proyecto:** apartado con bocetos explicativos del diseño inicial de la aplicación.
- **Análisis y especificaciones del sistema:** diversos documentos que se crearon para identificar los requisitos del proyecto, los posibles problemas que podían surgir durante su realización y un desglose de tareas específicas con las horas presupuestadas para cada una de ellas.
- **Análisis de software y hardware:** documentos en los que se plantean varios motores de videojuegos y dispositivos de realidad virtual con el objetivo de elegir el que mejor se adapte al proyecto.
- **Desarrollo del proyecto:** documento acompañado de un gran número de capturas en el que se muestra el resultado del proyecto y los requerimientos que se han implementado.
- **Conclusiones:** apartado final del documento en el que se hace un balance de los objetivos y requisitos del proyecto. También se plantean futuras líneas de trabajo y posibles comercializaciones. Por último, la memoria concluye con la opinión personal de la autora de este documento.

1.4. Referentes

The Sims 2 es un videojuego de simulación social desarrollada por Maxis y distribuida por Electronic Arts. Salió a la venta en septiembre de 2004 para Windows y lleva ya cuatro entregas, siendo esta la segunda; y es un éxito a nivel mundial habiendo vendido la franquicia cerca de las 200 millones de copias, convirtiéndolo en uno de las sagas de videojuegos más vendidas de todos los tiempos.

The Sims 2 tiene una interfaz y un modo de construcción sencillo e intuitivo, el cual se puede apreciar en la Figura 1.

Figura 1: Captura de pantalla del modo construcción de The Sims 2

The Elder Scrolls V: Skyrim es un videojuego de rol de acción del tipo mundo abierto desarrollado por Bethesda Game Studios. Salió a la venta el 11 de noviembre de 2011 para Windows, Xbox360 y PlayStation 3; y fue un éxito que actualmente supera los 30 millones de copias vendidas.

Skyrim permite elegir entre una cámara en tercera persona que se sitúa detrás del personaje o una cámara en primera persona como la que se ve en la Figura 2.

Figura 2: Captura de pantalla de The Elder Scrolls V: Skyrim en primera persona

2. Metodologías y herramientas

2.1. Metodología scrum

Para este proyecto se ha empleado una adaptación de la metodología ágil scrum. En este proceso se realizan entregas parciales y regulares del proyecto final, priorizando las tareas que tienen mayor impacto. Cada iteración tiene que proporcionar un resultado completo y ser un incremento de la anterior.

Con el objetivo de implementar esta metodología se ha elegido Trello, una herramienta online gratuita de gestión de proyectos en el que existe un tablero en el que se pueden crear diversas listas en las que se colocan tarjetas que describen las tareas. El trabajo de fin de grado se ha dividido en cuatro hitos: el "hito 0" en el que se incluye la planificación inicial del proyecto y la elección y manejo del motor gráfico y el dispositivo de realidad virtual; el primer hito donde se crea la base del proyecto implementando las interfaces, la situación inicial de los niveles y las acciones básicas que puede realizar el usuario; y los hitos 2 y 3 que son mejoras sobre este primer hito.

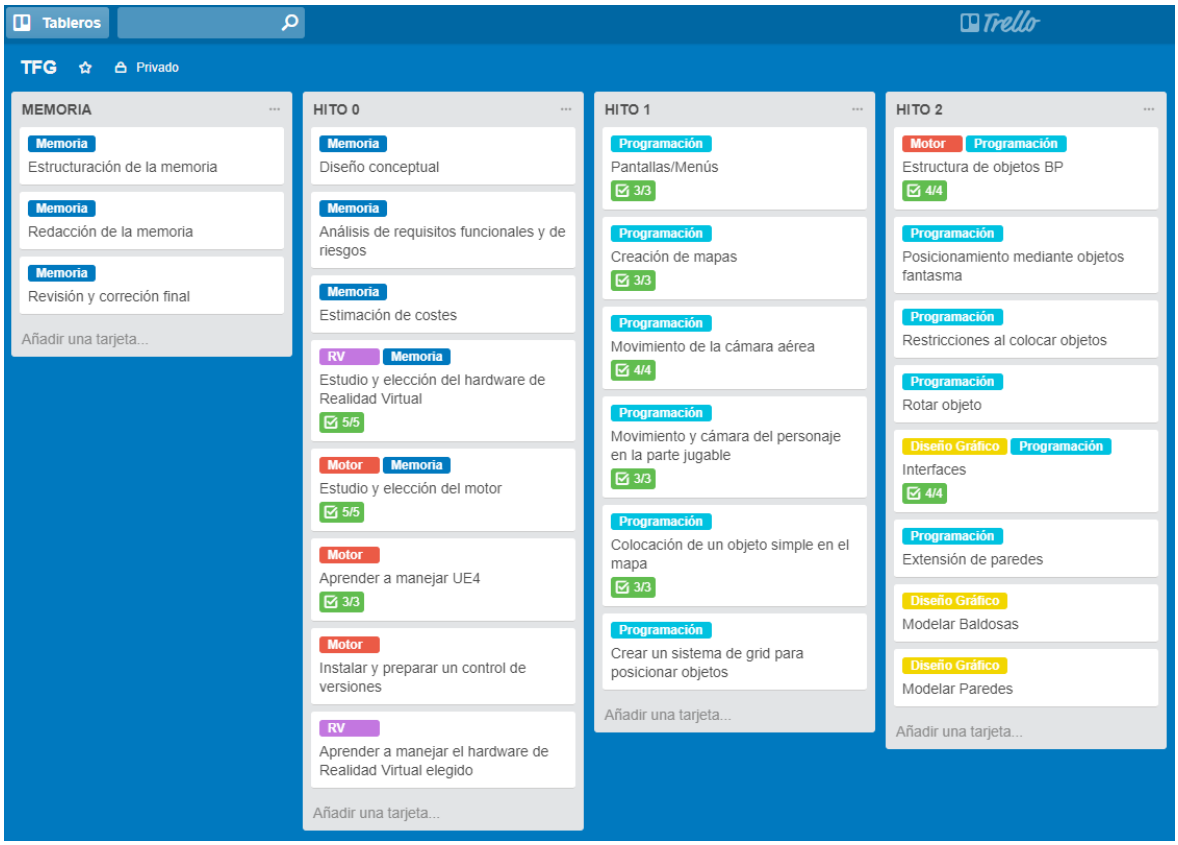


Figura 3: Distribución de tareas de los tres primeros hitos en Trello

Trello no permite poner fechas para las tareas de forma eficaz, por lo que se complementó con una pizarra dividida en diez días en los que se fueron asignando tareas, cumpliendo así con el objetivo de la metodología scrum de hacer una actualización diaria que depende de los avances. Las tareas iban cambiando de día dependiendo del estado del proyecto y de si superaban las horas estimadas. Las tareas siguen un código de colores, siendo el amarillo la memoria, el verde la programación, el rosa salmón el diseño gráfico y el rosa revisiones de código.

Figura 4: Pizarra en la que se asignaban las tareas a días específicos

2.2. Estación de trabajo

- **Sistema operativo:** Windows 10
- **CPU:** Intel Core i5 4690K 3’5 GHz
- **GPU:** NVIDIA GeForce GTX 960
- **Memoria RAM:** 8 GB
- **Almacenamiento:** HDD 1TB y SSD 128GB

2.3. Herramientas empleadas

- **Unreal Engine 4**: motor de videojuegos con el que se ha hecho la aplicación.
- **Oculus Rift**: hardware de realidad virtual empleado para el proyecto.
- **Photoshop**: software de edición de gráficos empleado para creación de los botones y los gráficos de este documento.
- **Maya**: programa de modelado en 3D con el que se han elaborado alguno de los modelos del proyecto.

3. Diseño del juego

3.1. Menús/Pantallas

3.1.1. Menú principal

Figura 5: Boceto del menú principal

Pantalla de inicio en el que el jugador puede elegir o bien crear un mapa o bien jugar algún mapa ya creado. También puede acceder al menú de opciones y salir del juego.

3.1.2. Modo construcción

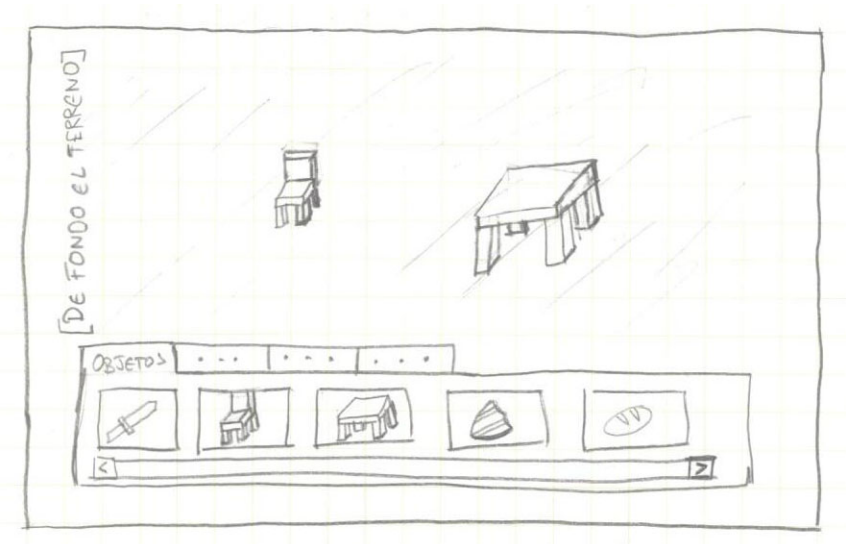


Figura 6: Boceto del modo construcción

Pantalla en la que el jugador crea una mazmorra mediante una interfaz en la que salen listados distintos objetos que pueden posicionarse en cualquier parte del terreno que no esté ocupada ya por otro elemento.

3.1.3. Cargar mazmorra

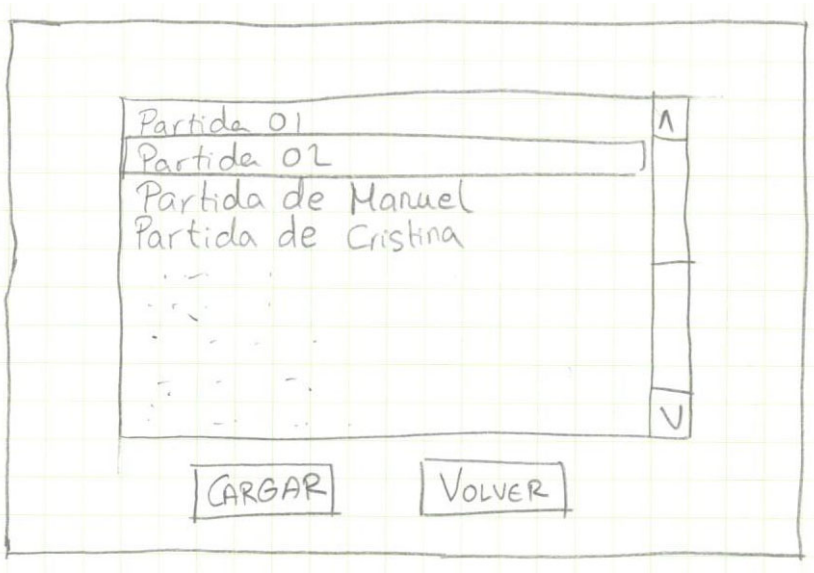


Figura 7: Boceto de la pantalla de cargar mazmorra

Lista de todas las mazmorras que han sido creadas y guardadas. El jugador podrá seleccionar una y darle a cargar para poder jugar a la mazmorra.

3.1.4. Mazmorra creada

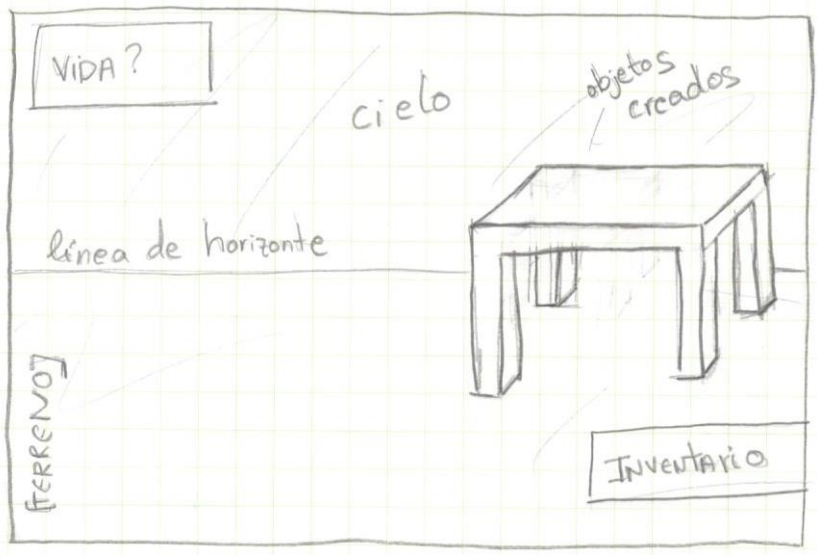


Figura 8: Boceto de una mazmorra ya creada

Mazmorra que ha sido previamente creada en el modo construcción. Se considera la posibilidad de que haya un inventario y se muestren diversos atributos del personaje como la vida o la energía.

3.1.5. Opciones

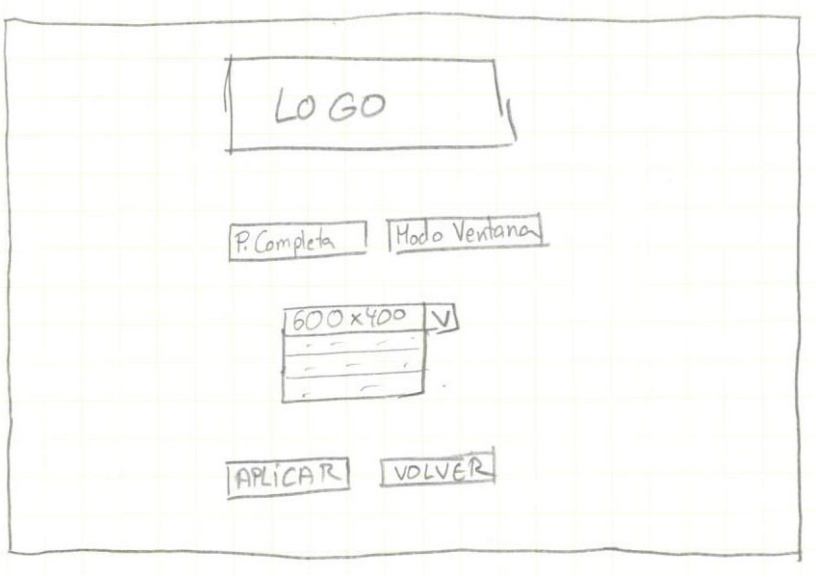


Figura 9: Boceto de la pantalla de opciones

El usuario podrá elegir a qué resolución desea ver el juego o si prefiere el modo ventana o pantalla completa.

3.2 HUD

3.2.1. Interfaz del menú de construcción

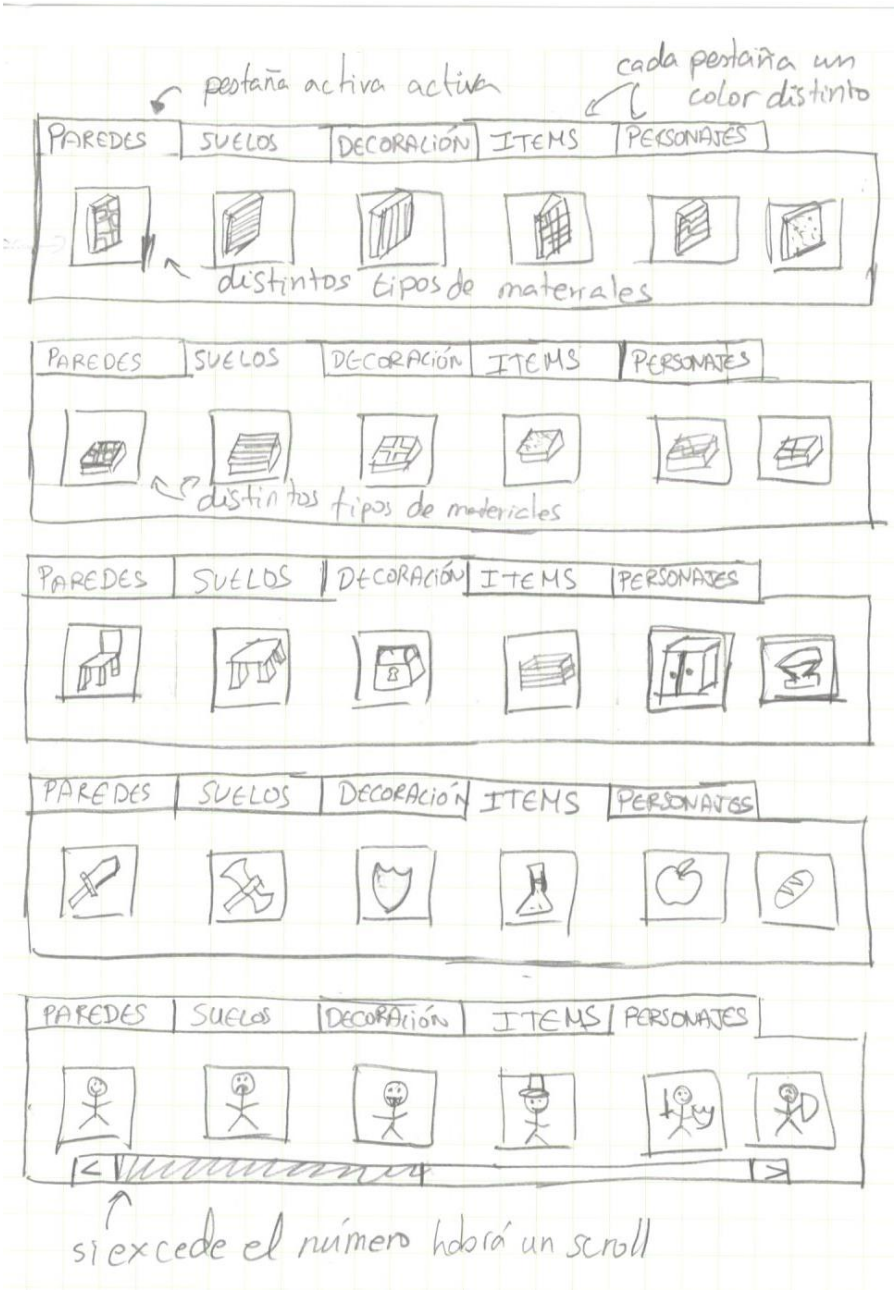


Figura 10: Boceto de las distintas pestañas de la interfaz del modo construcción

La interfaz principal del modo principal se encuentra en la parte inferior central; consta de diversas pestañas que se alternan para mostrar al usuario todos los elementos disponibles para usar en la construcción de la mazmorra.

3.2.2. Información del estado del personaje

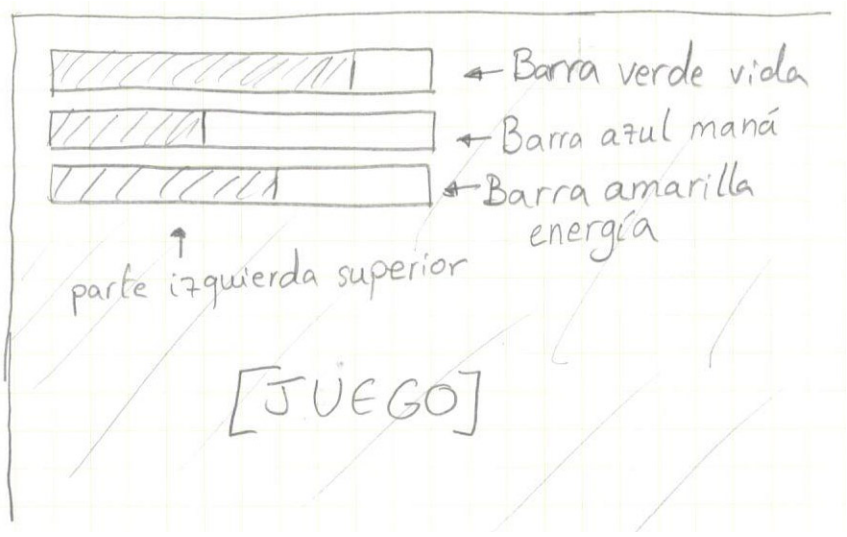


Figura 11: Boceto de la información del estado del personaje

En la parte superior izquierda el jugador podrá ver tres barras reflejarán la vida, el maná y la energía restante del personaje.

3.2.3. Inventario

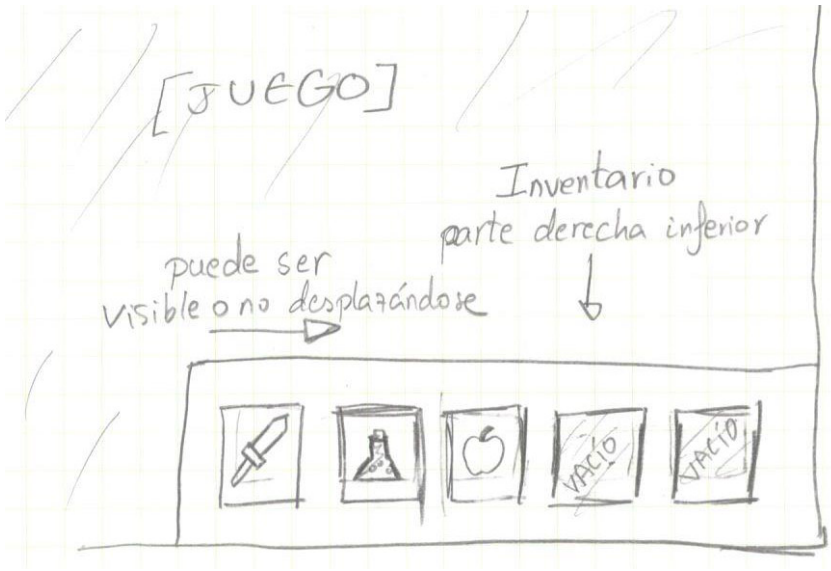


Figura 12: Boceto del inventario

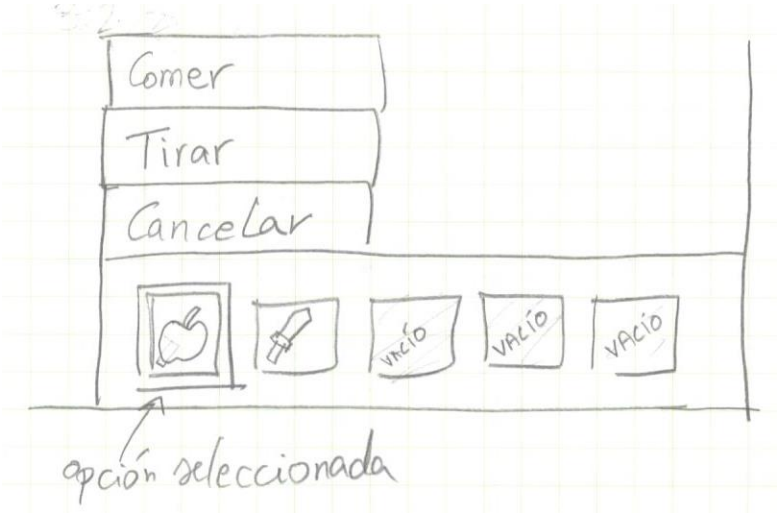


Figura 13: Boceto de las acciones de los objetos del inventario

El inventario se mostrará en la parte inferior derecha de la pantalla al pulsar un botón. Cuando se selecciones uno de los objetos con el ratón, aparecerán tres opciones; la primera estará relacionado con la acción del objeto, la segunda con dejar el objeto de nuevo en el terreno y la tercera cerrar estas tres opciones.

3.4. Escenario

Habrá cinco tipos de elementos que podrán ser puestos en el escenario, el cual inicialmente es un terreno plano y con un material de césped, los cuales serán:

- **Paredes:** su única función es separar instancias. Para facilitar su construcción podrá extenderse a lo largo.
- **Suelos:** su única función es crear distintas alturas. Para facilitar su construcción podrá extenderse a lo ancho y a lo largo.
- **Decoración:** no tiene ninguna función específica, puede ser desde un cofre hasta una estatua.
- **Ítems:** pueden recogerse, siendo añadidos al inventario y pudiéndolos usar.
- **Personajes:** tienen animaciones.

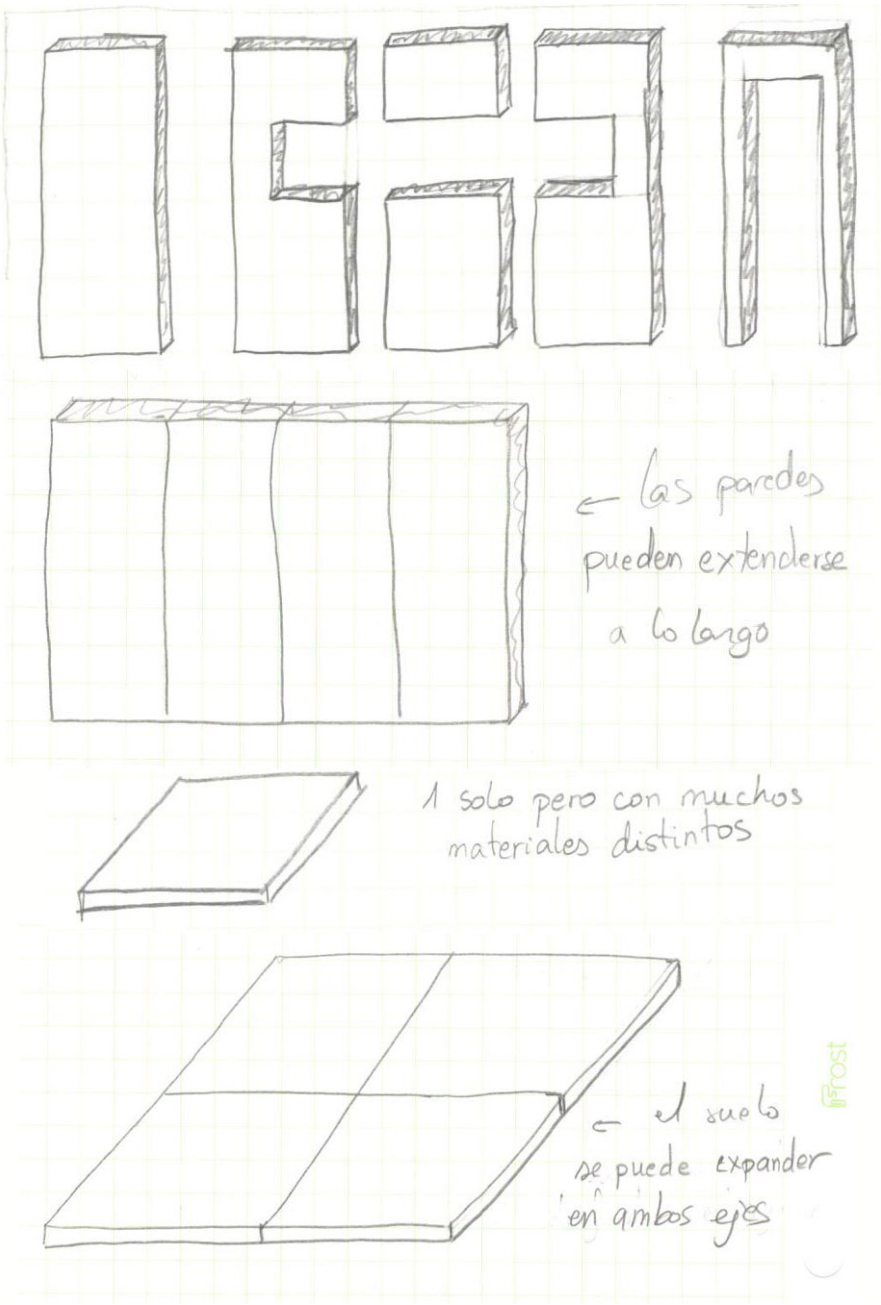


Figura 14: Elementos del modo construcción - Suelos y Paredes

3.5. Cámaras

El programa contará con dos tipos de cámara: una aérea para el modo construcción y otra en primera persona para jugar la mazmorra creada con posibilidad de usar un equipo de realidad virtual.

4. Análisis y especificaciones del sistema

4.1. Análisis de requisitos funcionales

	Requisito funcional	Descripción
F01	Movimiento de cámara en el modo construcción	Cámara aérea con la que el usuario podrá ver el mundo que está construyendo. Incluye movimiento, rotación, alejamiento y acercamiento.
F02	Sistema de *grid*	Sistema que permitirá colocar objetos siguiendo una cuadrícula. No es visible para el usuario.
F03	Creación de objetos fantasma	Al pulsar uno de los botones de crear objeto en el creador de mazmorras, aparecerá éste de color semitransparente en el terreno, lo que indicará que aún no está colocado definitivamente. Sólo podrá haber un objeto fantasma activo a excepción del suelo y las paredes.
F04	Movimiento de objetos fantasma	Al mover el ratón los objetos se moverán en el terreno siguiendo al cursor en pantalla.
F05	Rotación de objetos fantasma	Se podrán girar los objetos fantasma hacia ambos sentidos.
F06	Posicionamiento de paredes extensibles	Los objetos que sean de tipo pared no se pondrán de forma individual en el mapa, en su lugar se facilitará su creación permitiendo que el jugador la alargue hasta que él desee.
F07	Posicionamiento de suelo extensible	Los objetos que sean de tipo suelo no se pondrán de forma individual en el mapa, en su lugar se facilitará su creación de forma que pueda cubrirse una superficie rectangular.
F08	Posicionamiento del techo encima de las paredes	Podrá crearse techo encima de las paredes para crear así estancias cerradas.
F09	Restricciones de posicionamiento	Los objetos no podrán ser puestos en el lugar elegido si hay otro objeto ocupando ese mismo espacio. Si la posición no es correcta el fantasma pasará a ser de color rojo transparente.
F10	Deseleccionar el objeto fantasma actual	Dejará de aparecer en el mapa el objeto fantasma que teníamos seleccionado.

F11	Creación del elemento en el mapa	Si hay un objeto fantasma en el mapa se eliminará y se creará el objeto ya definitivo con el material correspondiente no transparente si tiene un correcto posicionamiento.
F12	Iluminación de los elementos del mapa si el cursor está encima	Si no hay ningún objeto fantasma seleccionado los objetos ya colocados brillarán cuando se pase el ratón por encima de ellos.
F13	Borrado de elementos del mapa	Se podrán borrar los objetos que hayan sido puestos en el mapa.
F14	Guardar mapa	Se almacenará el mapa en un archivo con todos los objetos que el usuario ha puesto en él. Habrá múltiples ranuras de guardado.
F15	Listar mapas	Todos los mapas serán mostrados en una lista para que el usuario elija cuál cargar.
F16	Cargar mapa	Los mapas almacenados podrán ser cargados con todos los objetos que el usuario puso en él.
F17	Cámara en primera persona	Cuando se juega el mapa creado el jugador lo verá con una cámara en primera persona.
F18	Vistas en realidad virtual	El jugador tendrá la opción de jugar con las gafas de realidad virtual Oculus Rift mediante la cámara en primera persona.
F19	Movimiento del personaje	El personaje puede moverse por el terreno.
F20	Colisión del personaje con objetos del entorno	Los objetos en el mapa serán obstáculos para el jugador y no podrá atravesarlos.
F21	Saltar	El personaje podrá elevarse y subir encima de objetos con colisiones.
F22	Animación de personajes y enemigos	Los personajes y enemigos que estén en el mapa tendrán una animación principal que en algunos casos podrá alternar entre variaciones.
F23	Estado del personaje	El personaje tendrá vida, maná y energía que podrá perder al recibir daño, al usar hechizo o al correr; y que podrá recuperar mediante algunos ítems.
F24	Sistema de inventario	Los ítems son un tipo de objetos que pueden almacenarse en el inventario y gestionarse desde éste.

F25	Coger ítem	Los ítems estarán por el terreno y podrán ser recogidos para añadirlos al inventario
F26	Soltar ítem	Los ítems pueden ser devueltos al terreno desde el inventario.
F27	Usar ítem	Cada ítem tendrá una forma de uso: algunos se podrán equipar como las armas y otros usar para recuperar vida.
F28	Cambio de resolución	El jugador en el menú de opciones podrá elegir la resolución de pantalla y entre los modos ventana o pantalla completa.

Tabla 1: Listado de todos los requisitos funcionales que debe tener el videojuego

4.2. Gestión de riesgos

Un documento de gestión de riesgos es un informe en el que se determinan aquellas circunstancias que pueden llegar a influir en la realización de un proyecto. Estos factores pueden ser determinantes en su resultado ya que pueden derivar al retraso o a la cancelación de éste.

Al crear un documento de este tipo se tienen que tener en cuenta las siguientes pautas:

1. Identificación del riesgo: Se determinan cuáles van a ser aquellos factores que aparecerán en el documento y a qué tipo pertenecen (tecnológico, personal, hardware, software…).

2. Análisis de probabilidad y efecto: Se determina la probabilidad con la que podría suceder el evento (baja, media, alta…) y su efecto (bajo, medio, catastrófico…).

3. Planificación de riesgos: Se crea un plan de actuación para prevenir un riesgo o minimizar el impacto del efecto de éste.

4. Monitorización del riesgo: Se estudia cuáles son los identificadores potenciales de una circunstancia que pueda afectar al proyecto.

	R01 – Enfermedad
Descripción	El único integrante del equipo de este proyecto cae enferma.
Tipo	Personal
Probabilidad	Baja
Efecto	Muy serio

Identificadores	Indicios de una enfermedad como el resfriado en el que puede haber dolores de cabeza o fiebre.
Estrategia	Evitar cambios de temperatura y climas extremos, tomar medicación en cuanto aparezcan los primeros síntomas e ir al médico.

Tabla 2: Riesgo R01 - Enfermedad

	R02 – Avería en la estación de trabajo
Descripción	El ordenador en el que se trabaja, o alguno de sus periféricos como la pantalla, se estropea e impide continuar trabajando.
Tipo	Hardware y software
Probabilidad	Baja
Efecto	Muy serio
Identificadores	Pantallazos azules, errores de memoria, humo, altas temperaturas de la CPU o GPU…
Estrategia	Monitorizar la temperatura del ordenador. En caso de avería traspasar los datos y programas a otro ordenador para poder continuar trabajando.

Tabla 3: Riesgo R02 - Avería de la estación de trabajo

	R03 - Problemas con las herramientas escogidas
Descripción	Unreal Engine 4 u Oculus Rift tienen problemas de rendimiento en la estación de trabajo o surgen incompatibilidades entre ellos.
Tipo	Hardware y software
Probabilidad	Baja
Efecto	Moderado
Identificadores	Altas temperaturas en CPU o GPU, alto uso de memoria, cierres repentinos o errores en los programas.
Estrategia	Hacer diversas pruebas antes de empezar el proyecto con ambas herramientas y probar sus rendimientos. Si aparecen incompatibilidades cambiar las Oculus Rift por otra de las opciones estudiadas.

Tabla 4: Riesgo R03 - Problemas con las herramientas escogidas

	R04 –Estimación de costes errónea
Descripción	Las horas estimadas para llevar a cabo diversas tareas son escasas requiriendo más tiempo. Esto podría derivar a que no pudiesen entregarse todos los objetivos que se plantearon.
Tipo	Estimación
Probabilidad	Media
Efecto	Serio
Identificadores	Tareas que van tomando más tiempo del estimado o jornadas más largas para poder acabar los objetivos en el tiempo marcado.
Estrategia	Estudiar la estimación de costes y hacer reajustes si fuese necesario. Analizar si hay tareas que puedan ejecutarse en un menor número de horas. Eliminar alguna de las tareas que no sea indispensable en caso extremo.

Tabla 5: Riesgo R04 - Estimación de costes errónea

	R05 – Sobreestimación de capacidades
Descripción	El proyecto abarca demasiados requisitos y se vuelve demasiado grande para una persona; o el aprendizaje y el desarrollo en Unreal Engine 4 exige un nivel de habilidad superior al que se posee.
Tipo	Personal y estimación
Probabilidad	Baja
Efecto	Muy serio
Identificadores	Dificultades a la hora de implementar el código, los requisitos no son completados, las tareas de programación llevan demasiado tiempo…
Estrategia	Pedir ayuda a algún compañero que haya trabajado con Unreal Engine 4 o a la tutora del proyecto. Plantearse disminuir los requisitos.

Tabla 6: Riesgo R05 - Sobreestimación de capacidades

	R06 – Aparición de errores
Descripción	Aparecen errores de implementación debido a una mala estructuración u optimización del código.
Tipo	Software

Probabilidad	Muy alta
Efecto	Bajo
Identificadores	El programa no funciona correctamente.
Estrategia	Crear un código limpio y bien estructurado para poder identificar el problema rápidamente y solucionarlo.

Tabla 7: Riesgo R06 –Aparición de errores

	R07 – Pérdida de datos
Descripción	Por un error de hardware o humano se pierde la memoria o los datos de la aplicación.
Tipo	Hardware o personal
Probabilidad	Baja
Efecto	Moderado
Identificadores	Problemas con el disco duro.
Estrategia	Subir copias diarias a la nube y creación de un repositorio en otro disco duro donde se tengan los datos duplicados.

Tabla 8: Riesgo R07 – Pérdida de datos

4.3. Estimación de costes

Para no repetir información en este documento he decidido incluir en esta tabla también la cantidad de horas empleadas para cada actividad, siendo usual que esta información se introduzca al final del documento en una nueva tabla.

Al final de esta tabla y en el apartado 6. Desarrollo del proyecto daré detalles del por qué hay algunas tareas que tienen una gran diferencia entre las horas estimadas y las horas realizadas.

Pienso que los tutoriales deben estar en esta tabla debido a que han sido un gran número de horas invertidas al ser una herramienta y un lenguaje de programación completamente nuevo para mí. Esta tarea no consistía sólo en ver el vídeo, sino que a su vez iba ejecutando lo que explicaba en éste y si al finalizar el vídeo no funcionaba lo que había hecho tenía que volver a revisarlo entero.

Nombre de la tarea	**Horas estimadas**	**Horas reales**

Unreal Engine 4		
Instalar y preparar un control de versiones	2	4
Creación y estructuración del proyecto	5	5
Tutoriales	35	-
"Introduction to Blueprints" por Unreal Engine	-	5
"UMG UI Inventory" por Unreal Engine	-	8
"Unreal Engine RTS Tutorial" por 3 Prong Gaming	-	12
"WTF Is?" y "HTF do I?" series por Mathew Wadstein	-	5
Varios tutoriales del canal de Youtube Virtus Learning Hub	-	5
Otros tutoriales	-	10
Limpieza del proyecto para la entrega	4	4
Oculus Rift		
Preparar el equipo de trabajo y configurar las gafas	2	2
Probar distintas demos	4	4
Instalar el SDK en Unreal Engine 4	2	0
Hacer distintas pruebas en Unreal Engine 4	4	4
Programación		
Estructura de clases/Blueprints	5	5
Funcionalidades del menú principal	5	5
Funcionalidades del menú de opciones	5	15
Funcionalidades del menú de cargar mazmorra	4	3
Funcionalidades del menú de construcción	8	4
Sistema de posicionamiento de cuadrícula	2	3
Crear objeto fantasma y que se mueva por terreno	2	2
Paredes extensibles	4	6
Suelo extensible	3	5
Controlar el solapamiento de objetos	3	4
Cambiar el material del objeto fantasma a rojo	1	4
Crear el objeto definitivo en el mapa	2	2
Iluminación del objeto al pasar el ratón por encima	3	3
Destrucción del objeto iluminado	1	1
Controles del personaje	2	1

Cámara del diseñador de mazmorras	2	2
Movimiento de la cámara del diseñador de mazmorras	4	6
Guardar/cargar mazmorras	16	16
Creación de todos los objetos del modo construcción	8	8
Configuración de la realidad virtual para el proyecto	4	8
Correcciones de errores y bugs	20	15
Diseño gráfico		
Modelos de suelo y paredes	3	3
Botones del menú principal y del menú de opciones	3	3
Iconos del modo construcción	4	4
Búsqueda de modelos, animaciones y texturas	6	6
Creación de otros modelos	30	36
Creación y diseño de las interfaces de los menús	6	6
Documentos		
Estructuración de la memoria	4	4
Análisis de los motores gráficos	20	24
Análisis de hardware de realidad virtual	20	17
Análisis y especificaciones del sistema		
Análisis de requisitos funcionales	4	5
Análisis de riesgos	4	4
Estimación de costes	7	7
Redacción de la memoria	40	40
Revisión y corrección de textos de la memoria	8	8
Preparación de la presentación	8	8
Total	**329**	**361**

Tabla 9: Listado de tareas y horas asignadas de la estimación de costes

5. Análisis de hardware y software

5.1. Análisis y estudio de los motores de videojuegos

5.1.1. Conceptos Iniciales

Un motor de videojuegos es un sistema software concebido para el diseño, la creación y la representación de videojuegos. Estos motores proporcionan una *API* y un *SDK* en los que suele incluirse un conjunto de herramientas de edición visual. Los elementos de un motor de videojuegos abarcan diversos apartados como la representación gráfica, scripting, físicas, audio, inteligencia artificial o red-

A nivel de desarrollo podemos distinguir entre los motores propios, que son aquellos que el equipo de trabajo deberá crear para poder diseñar e implementar el videojuego más adelante; y los motores ya implementados, los cuales han sido desarrollados previamente por una empresa, con los cuales el equipo de trabajo sólo tendría que hacer uso de ellos pudiendo dedicarse exclusivamente a la creación del videojuego.

Usar un motor ya implementado ofrece ciertas ventajas frente a tener que desarrollarlo:

- **Mejora del flujo de trabajo:** los diversos componentes del equipo trabajan sobre sus ficheros de datos pudiendo establecerse una separación entre programadores, artistas, músicos, desarrolladores y diseñadores.
- **Reusabilidad:** permite utilizar el mismo software, partes de código o implementaciones para varios videojuegos.
- **Portabilidad:** debido a la abstracción del hardware un juego puede ser desplegado en múltiples plataformas.
- **Menor coste de desarrollo:** la creación de un nuevo motor supone una elevada inversión de tiempo y dinero que se ahorraría si se hiciese uso de uno ya implementado. Además de que la mayoría ofrece entornos y herramientas que facilitan el trabajo.
- **Evolución del hardware:** mientras que un motor propio suele quedarse obsoleto con el tiempo, los motores ya implementados ofrecen soporte y actualizaciones para seguir la actualidad tecnológica y adaptarse al nuevo hardware. El coste de desarrollar un motor propio aumenta con el avance de la tecnología.

- **Documentación y foros:** las empresas que desarrollan motores de videojuegos ponen a disposición del usuario documentación de la *API* para facilitar su uso. Aquellos más populares también ofrecen tutoriales grabados por la propia empresa y foros en los que una amplia comunidad de desarrolladores comparten sus creaciones y atienden las dudas que plantean otros usuarios.

En resumen, un motor ya implementado permite al equipo centrarse en el diseño y la creación de un videojuego; y enfocarse en aquello que tiene más importancia en la actualidad: ofrecer al usuario un juego único y diferente.

Debido a las características, especificaciones del proyecto y tiempo disponible para el desarrollo de este proyecto se trabajará con un motor de videojuegos ya implementado.

5.1.2. Origen de los motores de videojuegos

En los inicios todo el juego estaba contenido en el ejecutable, por lo que la complejidad de los juegos de aquel momento era mínima ya que el código se volvía intratable y las especificaciones del hardware eran escasas. El término motor de videojuegos no aparece hasta mediados de los 90.

Con el surgimiento de los sistemas de ficheros aparecen nuevas posibilidades como separar los ejecutables de los datos o cambiar el contenido sin verse obligado a cambiar la mecánica del juego.

A principios del 1989, Origin Systems desarrolló un motor de videojuegos para el género ciencia ficción llamado Ultima Underworld. Este motor poseía un algoritmo para el mapeado de texturas que podía ser aplicado a distintos elementos de la escena como suelos o paredes. Los requerimientos de este sistema eran utilizar un ordenador basado en un Intel 386.

En 1993 el sector de los motores sufre una revolución cuando id Software lanza su popular videojuego *Doom*, para el cual se había desarrollado un motor de videojuegos que era capaz de crear una sensación de perspectiva 3D mediante ilusiones ópticas aunque todos sus gráficos fuesen en 2D. Su renderizado era rápido y requería un ordenador basado en un Intel 386 con entrada y soporte para VGA. Doom Engine es considerado el primer motor de videojuegos al asentar parte de las bases de motores y juegos posteriores como Unreal Engine o idTech.

XnGine (Bethesda Softworks 1995) fue el primer motor de videojuegos en 3D que se desarrolló basado en DOS. Posteriormente usaría gráficos de alta definición y sería compatible con tarjetas de vídeo 3dfx.

Quake engine (id Software 1996) fue el primer motor de videojuegos que empleaba 3D real. Poseía una capacidad única de procesamiento para renderizar mapas omitiendo aquellas áreas que el jugador no era capaz de visualizar. Para conseguir esto hacía uso de ciertas técnicas de Z-buffering, un método capaz de determinar aquellos objetos que deben ser renderizados.

Renderware (Criterion Software 1996) llegó a ser uno de los motores más populares para juegos multiplataforma, tanto que en la actualidad todavía puede emplearse para desarrollar videojuegos para Windows, Apple Mac OS-X, PlayStation 2, PlayStation 3, PlayStation Portable (PSP), Wii, GameCube, Xbox y Xbox360. Su principal uso fue proveer una solución a las dificultades de programación de gráficos de PS2. *SimCity* (EA 2013) fue el último juego conocido que se desarrolló con este motor.

Quake II/idTech 2 (id Software 1997) ofrecía soporte nativo para en lenguaje de programación C, OpenGL y efectos de coloreado de luces. Utilizaba DLL, lo que incrementó la capacidad de creación de mods.

GoldSRC, nombre empleado internamente por la empresa Valve Corporation para referirse a la gran cantidad de modificaciones hechas al motor de juego Quake para desarrollar su videojuego *Half-Life* (1998), ofrecía soporte para OpenGL y Direct3D; así como la posibilidad de crear mods, llevando a los videojuegos a una nueva era. Valve a su vez modificó *Half-Life* para crear el famoso videojuego *Counter Strike* (1999).

Uno de los motores gráficos más populares fue y sigue siendo Unreal Engine (Epic Games 1998), el cual fue utilizado para la creación del videojuego *Unreal Tournament* desarrollado por la misma compañía en 1999. Poseía un lenguaje propio para crear scripts llamado *UnrealScript* y un editor de mapas llamado *UnrealEd*.

En 1999 salió la nueva versión del motor de id Software: Quake III. Ofrecía soporte para la utilización de colores de 32-bits, shaders y funciones de red avanzadas.

Torque (GarageGames 2001) fue desarrollado para modificar el videojuego de disparos en primera persona *Tribes 2* (Dynamix 2001). Incluía un editor de mapas y la opción de renderizar con una reducción del número de polígonos.

Serious Engine (Croteam 2001) se diseñó para poder albergar grandes espacios y gran número de personajes en pantalla en cualquier momento. Este motor dio lugar a la popular *saga Serious Sam* de la misma compañía.

Más tarde en *Doom 3* (id Software 2004), desarrollado con id Tech 4 (id Software 2004), la mayoría de superficies con luz eran renderizadas en tiempo real, creando sombras más realistas pero con la desventaja de no poder mostrar sombras suaves (*soft shadows*). La solución para este problema fue emplear luces proyectadas que lograban crear una ilusión que simulaba este tipo de sombras.

En el 2004 salieron al mercado los videojuegos *Counter-Strike: Source* y *Half-Life 2*, ambos desarrollados por Valve Corporation, revelando el motor llamado Source en el que había estado trabajando la compañía como sucesor de GoldSrc. En *Half-Life 2* se implementó tecnología avanzada de shaders, luces dinámicas, físicas de sombras, *motion blur* en tiempo real y varios efectos realistas como superficies de agua reflectante.

CryEngine (Crytek 2004) empleó *pixel shaders* para crear agua realista en *FarCry* (Crytek 2004) y un *shader* de DirectX 10 en la segunda versión del motor para la creación del videojuego *Crysis* (Crytek 2007).

Rockstar Advanced Game Engine (Rockstar San Diego 2006) combina motor de físicas, de audio y de animación; librería de redes, un entorno de trabajo de renderizado y un lenguaje de scritping. La culminación de este motor fue *Grand Theft Auto IV*.

En 2006 Epic Games lanza al mercado Unreal Engine 3 el cual fue diseñado para sacar ventaja de la programación de shaders. Todos los cálculos de iluminación se hacían por píxel en vez de por vértice como su antecesor. Con este motor llegaron a desarrollarse videojuegos como *Gears of War* (Epic Games 2006), *Mass Effect* (Bioware 2007) o *Bioshock* (2K Boston, 2K Australia 2007).

En 2008 salió *Battlefield: Bad Company*, un videojuego de EA en el que destacó el que pudiese destruirse aproximadamente el 92% del entorno con increíbles efectos de explosiones. Incluía un motor de sonido que permitía recibir distintos sonidos según la proximidad a los objetos.

CryEngine 3 de Crytek fue lanzado en 14 de octubre de 2009 incluyendo soporte de desarrollo para diversas plataformas y para Direct X 9, 10 y 11; y los lenguajes de programación C y C++.

Su sucesor debía haber sido CryEngine 4, no obstante Crytek decidió no anunciarlo con un nuevo número de versión.

En marzo de 2014 Epic Games lanzó al mercado Unreal Engine 4 con un nuevo modelo de suscripción, pero no fue hasta marzo de 2015 cuando se anunció el modelo que utiliza actualmente, en el que el motor es completamente gratuito de usar. Modificaron el código del motor para que estuviese escrito completamente en C++.

En marzo de 2015 se lanzó oficialmente Unity 5, del cual cabe destacar su aumento de potencia que permitía crear gráficos y renderizados mejores; pudiendo crear así escenas más complejas y detalladas. Añadieron el editor de WebGL y mejoraron la iluminación global.

En la *Game Developers Conference* de 2015 Valve anunció Source 2 como sucesor del motor Source. El objetivo era reemplazar todas las herramientas obsoletas para permitir crear contenido de forma rápida y eficiente. Anunció que recibiría soporte para la Vulkan API y que utilizaría un nuevo motor de físicas llamado Ribkon, reemplazamiento de Havok, con el cual se podrían visualizar las físicas en el propio editor del motor. La empresa comunicó que el motor sería gratuito siempre y cuando el juego fuese publicado en su plataforma de juegos Steam.

En marzo de 2016 Crytek anunció su nueva versión CryEngine V; su novedad más importante era el modelo de licencia llamado “paga lo que quieras”, con el cual CryEngine se volvía gratuito e incluía acceso al código. Comunicaron que en un futuro tendría soporte para Direct X 12 nativo, Vulkan y realidad virtual.

A finales de agosto de 2017 Unity cambió de nomenclatura y lanzó su nueva versión Unity 2017. Con esta versión eliminó uno de los lenguajes a los que había ofrecido soporte desde Unity 1.0: UnityScript, la alternativa de C# similar a Javascript.

5.1.3. Motores de videojuegos

En la elección de un motor de videojuegos hay que tener en cuenta las necesidades del proyecto y las características del equipo de trabajo. Si el motor elegido no ha sido utilizado previamente puede requerir de un aprendizaje y una familiarización con sus herramientas que se traduce en tiempo; de ahí la importancia de tomar una buena decisión.

Algunos factores que debemos tener en cuenta son: la complejidad, la disponibilidad de una buena documentación, el lenguaje de programación, el acceso al código fuente, los tipos de licencia o con qué representaciones gráficas cuenta (2D o 3D).

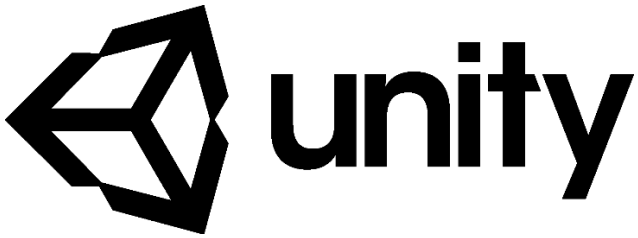


Figura 15: Logotipo de Unity

Creado por Unity Technologies y lanzado al mercado en 2005, Unity 5 es un motor de videojuegos disponible para Windows y MAC que emplea el lenguaje de programación C# y permite trabajar con gráficos 2D y 3D.

Está preparado para el desarrollo de videojuegos en distintas plataformas, empleando OpenGL en Windows, macOS y Linux; Direct3D en Windows y Xbox One; OpenGL ES en Android e iOS; WebGL en la web; y las API correspondientes de las consolas PlayStation 4, PsVita, WiiU, Nintendo 3DS y Nintendo Switch.

Unity es uno de los motores más utilizados para la creación de videojuegos para *smartphone*. Actualmente es el SDK que usa por defecto la Nintendo WiiU.

Algunos videojuegos desarrollado con Unity son: *Ori and the Blind Forest* (Moon Studios 2015), *Hollow Knight* (Team Cherry 2017), *Syberia 3* (Microïds 2017), Pokémon GO (Niantic 2016), *Super Mario Run* (Nintendo EPD 2016)…

Licencia

Unity ofrece 4 tipos de licencias; todas incluyen las funciones básicas del motor, la posibilidad de exportar para todas las plataformas de desarrollo y actualizaciones de la última versión del motor.

- Unity Personal
  Versión gratuita siempre y cuando la empresa genere menos de 100.000 dólares al año.

- Unity Plus
  Esta licencia está pensada para individuos y equipos que necesitan una mayor optimización para sus juegos comerciales. Sigue teniendo la misma restricción de ingresos que *Unity Personal* de 100.000 dólares al año pero incluye más características de soporte y personalización de interfaz. Su precio es de 35 dólares al mes en un plan de 12 meses o 49 dólares al mes en un plan mensual.

- Unity Pro

  Pensada para profesionales o equipos que necesitan completa flexibilidad para crear juegos comerciales. Viene incluido todo lo que ofrece *Unity Plus* pero sin límite de ingresos anuales. Su precio es de 125$ al mes con un año de compromiso.

- Unity Enterprise:

  Para organizaciones con equipos grandes que requieren el código fuente y un mayor soporte para sus proyectos. Esta licencia puede obtenerse mediante un representante de Unity, dependiendo el precio de adquisición de diversos factores como el tamaño de la empresa y el número de integrantes del equipo de desarrollo.

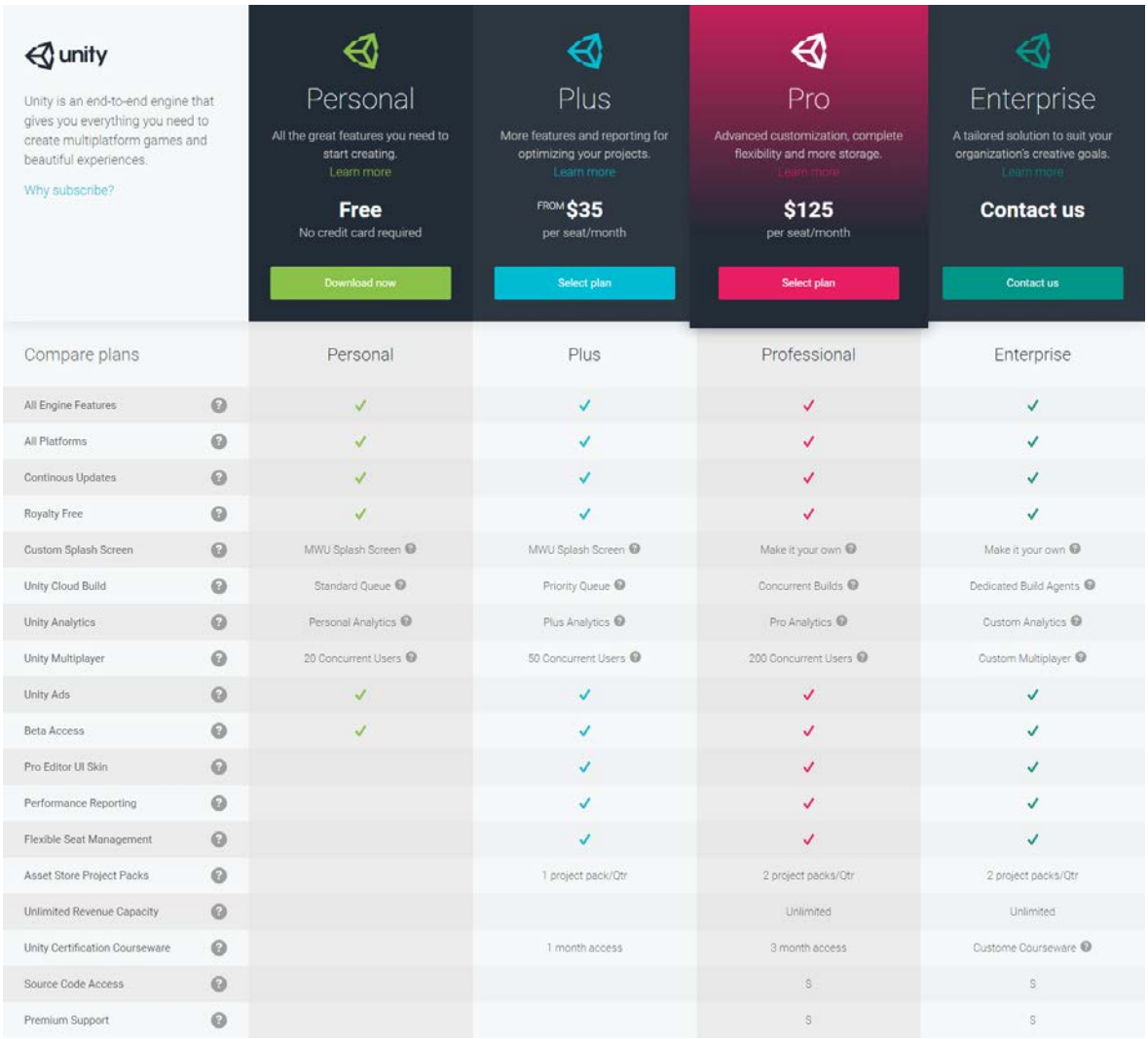


Figura 16: Tabla-comparación oficial de las licencias que ofrece Unity

Ventajas

- Es compatible con diversas herramientas de creación de gráficos como Blender, 3Ds Max, Maya, ZBrush, etc.

- Ofrece soporte para varios modelos de hardware de realidad virtual: Google Cardboard, Draydream, Oculus Rift, HTC Vive, Samsung VR y Playstation VR.
- Permite la creación de prototipos de manera rápida gracias a la existencia de shaders, scripts y animaciones ya definidas.
- Pone a disposición del desarrollador una tienda con diversos elementos como scripts, modelos, animaciones…
- Debido a que es muy popular existe una gran cantidad de contenido como tutoriales publicado por otros desarrolladores.
- En la página oficial de Unity hay publicados diversos tutoriales que son de gran ayuda.

Desventajas

- Tiene menor potencial gráfico que otros motores gráficos como Unreal Engine 4 o CryEngine.
- Es gratuito sólo hasta los 100.000$ de ingresos en la empresa.
- La versión gratuita no incluye todas las herramientas de personalización y soporte.
- No incluye plantillas de creación de proyecto.

Requisitos del sistema

Recomendados para el desarrollo: Ordenador con el sistema operativo Windows 7 SP+1, 8, 10 o Mac con Mac OS X 10.9+; y una tarjeta gráfica con DX9 (modelo de shader 3.0) o DX11 con capacidades de funciones de nivel 9.3.

Para ejecutar juegos: Ordenador con el sistema operativo Windows XP SP2+, Mac OS X 10.9+, Ubuntu 12.04+ o SteamOS+; tarjeta gráfica con DX9 (modelo de shader 3.0) o DX11 con capacidades de funciones de nivel 9.3; y una CPU compatible con el conjunto de instrucciones SSE2.

Figura 17: Logotipo de Unreal Engine 4

La primera vez que se implementó Unreal Engine fue en el videojuego de disparos en primera persona *Unreal*, desarrollado por la compañía Epic Games en 1998. Aunque su función inicial fuese la creación de juegos del género mencionado, ha evolucionado convirtiéndose en un motor que puede emplearse para desarrollar videojuegos y aplicaciones variadas.

El motor está escrito en C++, pudiendo modificarse ya que el código fuente está a disposición del desarrollador. Existen dos lenguajes de programación para crear la lógica del juego: C++ y un lenguaje visual propio llamado *Blueprint*. Con este último no es necesario escribir líneas de código, ya que mediante cajas y conexiones entre ellas se pueden crear mecánicas, interfaces y materiales, entre otros. Los dos lenguajes pueden coexistir en un mismo proyecto, otorgando una mayor flexibilidad al equipo de trabajo.

La versión actual permite exportar para Windows, macOS, Linux, SteamOS, HTML5, iOS, Android, Nintendo Switch, Playstation 4, Xbox One y diversos dispositivos de realidad virtual. Las herramientas específicas de las videoconsolas mencionadas están disponibles al registrarse en sus perspectivas plataformas sin coste adicional.

Algunos de los videojuegos creados con Unreal Engine 4 son: *Gears of War 4* (The Coalition 2016), *Kingdom Hearts III* (Square Enix), *Final Fantasy VII Remake* (Square Enix)…

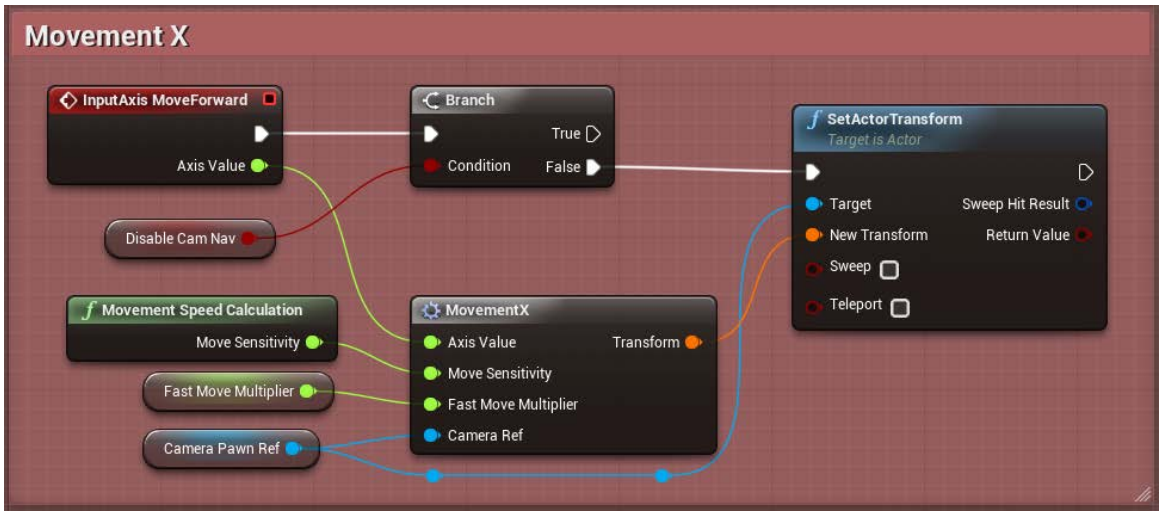


Figura 18: Ejemplo de Blueprint

Licencia

Unreal Engine 4 es totalmente gratuito y posee una licencia basada en regalías del 5% que es aplicable a partir de los 3000$ de ingresos por juego. Esto significa que el 5% de los ingresos brutos del videojuego o aplicación desarrollada hay que ingresárselo a Epic Games cada trimestre.

Existen otro tipo de licencias personalizadas tanto para usuarios como para empresas. Sus términos tienen que negociarse con un representante de la empresa Epic Games.

Ventajas

- Es compatible con diversas herramientas de creación de gráficos como Blender, 3Ds Max, Maya, ZBrush, etc.
- Ofrece soporte para varios modelos de hardware de realidad virtual: Google Cardboard, Draydream, Oculus Rift, HTC Vive, Samsung VR y Playstation VR. Sus *SDK* vienen instalados por defecto.
- El programa es completamente gratuito, no es necesario pagar por mejoras.
- Es uno de los motores de videojuegos más potentes en la actualidad.
- El editor de materiales es muy avanzado y ofrece muchas facilidades. Esto permite crear todo tipo de materiales, por muy complejos que sean.
- Los *blueprints* permiten crear prototipos rápidamente y sin escribir código.
- Rapidez en la compilación de código.

- Dispone de un bazar en el que se pueden comprar o vender diversos elementos como modelos, animaciones, scripts, etc. Epic Games pone a disposición del usuario de forma gratuita varios de estos componentes.

- La página web de Epic Games tiene una gran cantidad de tutoriales oficiales.

Desventajas

- Algunos desarrolladores tienen dificultades para adaptarse a la programación visual.

- El programa ofrece gran cantidad de opciones, haciendo que al usuario pueda llevarle mucho tiempo aprender a manejarlas todas.

Requisitos del sistema

Recomendados para el desarrollo: Ordenador de sobremesa con Windows 7 64-bit o un Mac con Mac OS X 10.9.2, 8 GB RAM, un procesador Intel o AMD *quad-core* de 2'5GHz, y una tarjeta gráfica compatible con DX11 como la NVIDIA GeForce 470X o la AMD Radeon 6870HD.

Para ejecutar juegos: Depende del juego desarrollado.

Figura 19: Logotipo de CryEngine

Creado originalmente una demostración para la empresa Nvidia, CryEngine fue presentado en 2002 por la empresa Crytek. Debido a su gran potencial decidió implementarse por primera vez en el videojuego *Far Cry* (Crytek, 2004).

Su versión actual, CryEngine V lanzada en 2016, es la única que permite desarrollar para realidad virtual y ofrece soporte para DirectX 12 y Vulkan. Además, se creó un nuevo modelo de licencia llamado *Pay what you want* (para por lo que quieres) y se facilitó el acceso al código fuente.

El motor está escrito en C++, Lua y C#; lenguajes que también pueden usarse para la creación del código del videojuego. Incluye el famoso editor *Sandbox*, el cual también podría encontrarse en sus predecesores. Este editor permite un control total sobre las creaciones en

tiempo real y posee un gráfico de flujo que utiliza lenguaje visual para que pueda ser modificado sin necesidad de editar el código.

El editor puede instalado en Windows y tiene soporte multiplataforma para: Oculus Rift, HTC Vive, Windows PC, Linux PC, Xbox One y Playstation 4.

Los videojuegos creados con CryEngine V son escasos ya que a día de hoy su versión más utilizada sigue siendo CryEngine 3. Juegos creados con CryEngine 3: *Monter Hunter Online* (Tecent Games 2013), *Crysis 3* (Crytek 2013), *Archeage* (XL Games 2014), *Ryse: Son of Rome* (Crytek, 2013)…

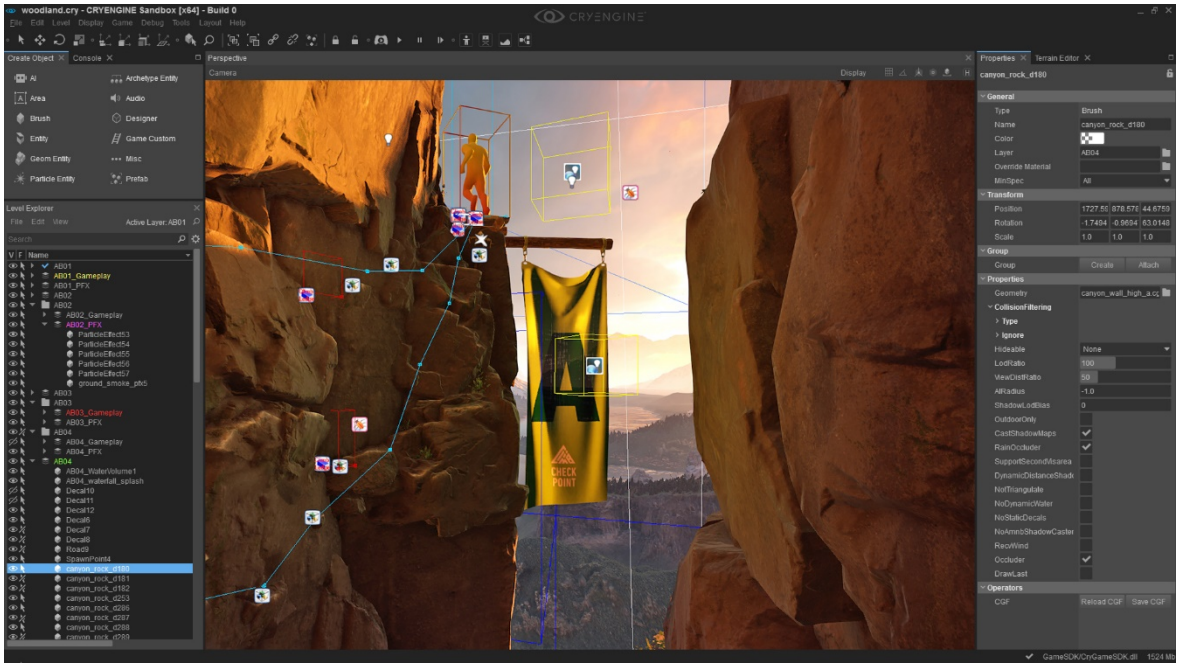


Figura 20: Editor Sandbox del motor CryEngine

Licencia

CryEngine es completamente gratuito; libre de regalías independientemente de los ingresos brutos de la empresa. La versión gratuita permite el uso todas las funciones del motor, el acceso su código fuente, la publicación de videojuegos en todas las plataformas a las que ofrece soporte, un bazar y el acceso a la documentación y tutoriales.

Además, Crytek ofrece dos tipos de cuota mensuales llamadas *Base Membership* y *Premium Membership* para aquellos desarrolladores o empresas que necesitan entrenamiento o soporte adicional más allá del que puede ofrecer la comunidad.

Como el resto de motores, Crytek también ofrece membresías especiales para empresas que deben ser negociadas con alguno de sus representantes.

	BASE MEMBERSHIP €50 /mo	PREMIUM MEMBERSHIP €150 /mo
Access to Insider an exclusive CRYENGINE Answers section	✓	✓
Regular webinars by CRYENGINE experts	✓	✓
User revenue share from Marketplace sales	80%	80%
Ability to purchase Support Packages Sold separately	✓	✓
Response times on Support Requests	3 working days	2 working days
Training Sold separately	Fixed	Fixed + Customizable
Discount on Support Packages	✗	10%
Project Kickoff	✗	✓
Dedicated Account Manager	✗	✓
Consulting services Sold separately	✗	✓

Figura 21: Tipos de cuotas que ofrece CryEngine

Ventajas

- Puede ser totalmente gratuito.
- Es el motor más potente analizado en este documento.
- Gran nivel de renderizado y acabado de nubes, vegetación y elementos de paisajes.
- Gran atractivo debido a la calidad de sus gráficos y sus efectos especiales meteorológicos.
- Su editor *Sandbox* es de gran utilidad a la par que flexible.
- Posee una tienda en la que pueden obtenerse objetos 3D, sonidos, escenas y otros elementos creados por la comunidad.

Desventajas

- Según su página web sólo ofrece soporte para Oculus Rift y HTC Vive en lo que respecta a realidad virtual.
- Está enfocado a los videojuegos de disparos en primera persona, por lo que encontraremos facilidades si queremos desarrollar otro género.
- La curva de aprendizaje es muy elevada.

Requisitos del sistema

Recomendado tanto para el editor como para los juegos de CryEngine V: sistema operativo Windows 7, 8.1 o 10 de 64-bits; procesador Intel Quad-Core (i5 2300) o AMD Octo-Core (FX 8150), 8GB de memoria RAM, tarjeta gráfica NVIDIA GeForce 660Ti o AMD Radeon HD 7950 (mínimo 2GB dedicados VRAM GDDR5), DX11, 8GB disponibles de espacio en el disco duro y una tarjeta de sonido compatible con DirectX.

5.1.4. Tabla comparación

	Unity 5	Unreal Engine 4	CryEngine V
Plataformas de desarrollo	Windows 7, 8, 10 Mac OS X 10.9+	Windows 7/8 64-bit Mac OS X 10.10.5 + Ubuntu 15.04	Windows 7, 8.1, 10 64-bit
Licencia	Gratuita hasta los 100000$ de ingreso en la empresa	Gratuita y a partir de los primeros 3000$ de ingreso por el juego 5% en regalías	"Paga lo que quieras", gratuito sin regalías ni límites de ingresos
Lenguaje	C#	C++, Blueprints	C++, C#, Lua
Realidad Virtual	Cardboard, Daydream, Oculus Rift, Gear VR, HTC Vive, Sony VR	Cardboard, Daydream, Oculus Rift, Gear VR, HTC Vive, Sony VR	Oculus Rift, HTC Vive
Complejidad	Media	Alta	Muy Alta
Potencia	Media	Alta	Muy Alta
Multiplataforma	Windows, macOS, Linux, SteamOS, Xbox One, Android, iOS, WebGL, PlayStation 4, PsVita, Nintendo 3DS, WiiU, Nintendo Switch y Xbox One	Windows, macOS, Linux, SteamOS, HTML5, iOS, Android, Nintendo Switch, PlayStation 4 y Xbox One	Windows, Linux, Xbox One, PlayStation 4
Documentación	Sí	Sí	Sí
Tutoriales	Sí	Sí	Sí
Tienda	Sí	Sí	Sí

Tabla 10: Tabla-comparación de las características de los motores analizados

5.1.5. Conclusión

CryEngine V es un motor de videojuegos muy potente que ofrece muchas posibilidades, no obstante su curva de aprendizaje es muy superior a Unreal Engine 4 y Unity 5. Esto podría suponer una mayor cantidad de horas dedicadas a aprender el funcionamiento del programa teniendo que eliminar algún requisito funcional.

Unity 5 es al contrario que CryEngine V, posee un escaso de potencial y su calidad de renderizado no puede compararse a los otros motores. En anteriores proyectos ya he trabajado con este software, si bien mi experiencia no fue mala, no ofrece tantas posibilidades a la hora de desarrollar como Unreal Engine 4.

En mi opinión, Unreal Engine 4 es la elección idónea para este proyecto debido a su potencia, su calidad de renderizado y a su lenguaje visual *Blueprint*, el cual permite programar de forma rápida

5.2. Análisis y estudio del hardware de Realidad Virtual

5.2.1. Conceptos iniciales

La realidad virtual es un término relativamente recientemente que corresponde a la representación de escenas o imágenes de objetos producidos por un sistema informático y que da la sensación de su existencia real. No debe confundirse con el término de realidad aumentada, el cual es un sistema que complementa el mundo real con objetos virtuales generados por ordenador que dan la sensación de coexistir en el mismo espacio.

La finalidad de la realidad virtual es engañar a los sentidos, de forma que el usuario crea que lo que está percibiendo es la realidad. Esto se consigue mediante entornos generados por computador, principalmente compuestos de gráficos de computador interactivos y diseñados para una inmersión física o psicológica para una o más personas en una realidad alternativa a través de tecnología e interfaces que permite a un usuario interactuar de forma intuitiva con estos entornos.

Los gráficos por ordenador, la animación por ordenador o cualquier otra visualización que no sea interactiva y en tiempo real; no son realidad virtual.

5.2.2. Historia de la realidad virtual

La historia de la realidad virtual es extensa, por lo que en este apartado sólo se van a exponer los hechos más importantes y aquellos dispositivos que estén relacionados de una u otra forma con la realidad virtual.

1956 – **Sensorama:** Morton Heilig crea una cabina parecida a una recreativa en la cual se proyectaba una película en color, con sonido estéreo, olores, viento y vibraciones. Esta máquina fue creada con la intención de que el usuario estuviese totalmente inmerso en la película. Aun así este aparato no era interactivo ya que simplemente se veía la película.

1965 – **The Ultimate Display:** Ivan Sutherland describió el término de “Ultime Display” como un sistema que podía simular la realidad hasta el punto en el que uno no se podría distinguir la simulación de la realidad actual.

1968 – **Espada de Damocles:** Ivan Sutherland y su estudiante Bob Sproull crean la primera montura capaz de reproducir realidad virtual que estaba conectada a un ordenador y no a una cámara. El mecanismo era demasiado grande y pesado como para que una persona pudiese

usarlo de forma cómoda y por ello estaba sostenido del techo. Los gráficos generados eran primitivas.

1972 – Primer simulador de vuelo creado para la armada de EEUU por General Electric.

1977 - **The Sayre Glove:** Daniel J. Sandin, Thomas A. DeFant y Rich Sayre diseñan el primer guante capaz de transmitir datos mediante la flexión de la mano. El guante tenía unos tubos flexibles con unos sensores basados en la luz en un extremo y un emisor de luz en el otro, por lo que al flexionar los dedos, la cantidad de luz que pasaba por el tubo disminuía, registrando estos datos el sensor y sabiendo cuánto estaban flexionados.

1982 – Simulador de vuelo contenido en un casco por Furness.

1985 – HMD con un LCD monocromo de tamaño bolsillo desarrollado por McGreevy y Humphries.

1990 – **Sense 8**: Primera compañía comercial de software de VR que ofrece las primeras herramientas para sistemas SUN.

1991 – SUN presenta una demostración de su Portal Virtual, ambiente VR de mayor resolución hasta el momento.

1992 – Debuta el primer sistema CAVE. Desarrollado por Carolina Cruz-Neira, Daniel J. Sandin y Thomas A. DeFanti en la Universidad de Illinois; el CAVE es una habitación cúbica donde varios proyectores proyectan las imágenes en las paredes (entre 3 y 6 pareces). El usuario lleva gafas 3D puede moverse por toda la habitación explorando y viendo los objetos como si estuviesen en la realidad, aunque obviamente no tocándolos. Suele haber varios altavoces para proveer un sonido 3D.

1993 – SGI anuncia un motor de Realidad Virtual.

1993 – Gafas de realidad virtual de SEGA.

1994 – InmersaDesk, versión reducida y portable de un sistema CAVE.

1995 – **Virtual Boy:** Nintendo sacó la primera videoconsola portátil de realidad virtual, de 32 bits con gráficos en 3D de verdad en rojo y negro. Fue un fracaso comercial debido a lo voluminosa que era y su fragilidad. Además no disponía de gran catálogo. Japón y Norte América al precio de 180$.

1999 – **Matrix:** película escrita y dirigida por los hermanos Wachowski en la que se presenta un concepto de realidad virtual. No fue la primera cinta que representaba la realidad virtual, pero sí fue una de las que popularizó este tipo de tecnología al usuario.

2001 – **iSmell Personal Scent Synthesizer**: DigiScents desarrolla un dispositivo capaz de emitir olores a partir de 128 cartuchos que contienen esencias que pueden ser combinadas entre sí. La máquina se controla desde un ordenador mediante una conexión USB.

2008 – Primeros televisiones con visión 3D sin necesidad de gafas.

2010 – **Kinect**: periférico desarrollado por Microsoft para la Xbox360. Permite controlar e interactuar con la consola mediante una cámara que detecta los gestos del usuario y un micrófono que reconoce comandos de voz.

2012 – **Kickstarter Oculus Rift:** comienzo de una nueva etapa de la realidad virtual, a partir de este punto varias empresas empiezan a desarrollar sus propias monturas para competir en el mercado.

5.2.3. Monturas de realidad virtual con dependencia móvil

Figura 22: Logotipo de Google Cardboard

Google Cardboard es una montura abatible de cartón combinada con unas lentes, un imán, un trozo de velcro y una cinta elástica. Para funcionar requiere de un *smartphone* conectado al sistema. Las lentes permiten a la persona percibir las imágenes de los ojos izquierdo y derecho como una única imagen tridimensional.

Aunque el sistema ha sido diseñado por Google no hay un vendedor o un fabricante oficial, no obstante en su página web pone a disposición del usuario una serie de esquemas e instrucciones de montaje para que pueda crear la montura por sí mismo. La empresa también tiene en su página web una lista de fabricantes que se dedican a vender estas monturas y a las

cuales se les ha sido otorgado un certificado que indica que son compatibles con Cardboard. Google no garantiza la seguridad, calidad o funcionalidad de estos visores.

Las gafas se lanzaron el 24 de junio de 2014 y en mayo de 2017 se estima que se habían vendido alrededor de 10 millones de monturas y se había descargado más de 160 millones de veces su aplicación. Debido a este éxito Google anunció que estaban trabajando en otra plataforma de realidad virtual llamada Google Daydream.

La única forma de interacción con estas monturas es con el movimiento de la cabeza. El imán en el lateral se trata de un anillo de neodimio magnético el cual se mantiene a su sitio gracias a un imán cerámico que se encuentra en el interior de las gafas, ambos influyen sobre la brújula del *smartphone*. Gracias a esto se puede saber hacia dónde está mirando el usuario y el sistema reacciona cuando éste observa un punto fijo durante cierto tiempo; permitiendo navegar entre los menús.

Figura 23: Imagen de la montura Google Cardboard

Ventajas

- Precio muy inferior frente a la competencia.
- Puede fabricarse de forma casera a medida del móvil empleado.
- La inmensa mayoría de *smartphone* son compatibles.

Desventajas

- La calidad de imagen, la latencia y la detección de movimiento dependen del hardware del *smartphone* que estemos utilizando, por lo que puede que la experiencia no sea confortable.

- Las aplicaciones/juegos que se desarrollen tienen que requerir poca potencia al estar creado para ejecutarse en un dispositivo móvil.
- Dependiendo el material con el que se fabrica puede deteriorarse rápidamente.

Figura 24: Logotipo de Samsung Gear VR

Las Samsung Gear VR es una montura de realidad virtual desarrollada por Samsung Electronics en colaboración de Oculus y manufacturadas por Samsung. Salieron al mercado el 27 de noviembre de 2015 aunque anteriormente había producido ya dos versiones de desarrollo.

El *smartphone* actúa como procesador y monitor mientras que las Gear RV actúan como el controlador, el cual ofrece un campo de visión de 101° y diversos sensores que mejoran el control de su rotación.

Para mejorar la interacción con el usuario posee un *touchpad* y una serie de controles en la parte derecha de la montura que permite movernos por las distintas aplicaciones y juego con facilidad. También incluye un mando que permite controlar el movimiento.

Se conecta al Smartphone vía micro-USB y sólo los siguientes dispositivos Samsung Galaxy son compatibles con el dispositivo: Galaxy Note 5, Galaxy S6/S6 Edge, S6 Edge +, Galaxy S7/S7Edge, Galaxy A5 y Galaxy S8/S8+.

Figura 25: Dispositivo Samsung Gear VR

Ventajas

- Su diseño ofrece robustez.
- La experiencia mejora gracias al uso de sensores dentro de la propia montura, permitiendo una detección de movimiento de cabeza más exacto.

Desventajas

- Sólo es compatible con móviles de gama alta muy específicos por lo que incrementaría el precio final si no disponemos de uno.

5.2.4. Monturas de realidad virtual sin dependencia móvil

Figura 26: Logotipo Playstation VR

Presentado inicialmente como "*Project Morpheus*" en la *Games Developer Conference* en el 2014, las PlayStation VR son unas monturas de realidad virtual diseñadas por Sony para su videoconsola de sobremesa PlayStation 4. Finalmente salieron a la venta el 13 de octubre de 2016 a un precio de 399€ sin cámara incluida. En junio de 2017 llevaba un millón de unidades vendidas.

Las gafas incluyen la pantalla de 5’7 pulgadas con una resolución de 1080p y un procesador que permite proyectar la imagen a la televisión a la vez que procesa el audio en caso de tener conectado unos auriculares. Posee nueve LEDs de posicionamiento para que la “Playstation Camera” pueda detectar el movimiento del casco; por lo que sin esta cámara no puede vivirse la experiencia completa. El precio de la cámara ronda los 55€.

Existen dos modos para jugar con PlayStation VR en funcionamiento: en el primero se proyecta en el televisor lo mismo que el usuario está viendo en la montura; mientras que en el segundo caso se proyectas imágenes distintas para poder jugar cooperativo o competitivo. Los usuarios interactúan con el mundo virtual a través del mando de la consola DualShock 4 o PS Move.

Figura 27: Montura PlayStation VR

Ventajas

- Detección de movimiento avanzada.
- Es compatible con varios jugadores.

Desventajas

- Aunque Sony mencionó la posibilidad de ofrecer soporte en ordenador, de momento sólo puede ser utilizado en la PlayStation 4.
- Montura pesada y algo incómoda.
- El gasto se dispara si tienes que comprar la consola, la cámara y los mandos.

Figura 28: Logotipo de Oculus Rift

Oculus Rift nació como un kickstarter en 2012 por Palmer Luckey el cual alcanzó los 2’5 millones de dólares. En marzo de 2014 Facebook compró la empresa que crearon y con ella las Oculus por 2 billones de dólares.

Oculus ha tenido varios prototipos y versiones de desarrollo: el DK1 a mediados de 2013, con una resolución de 640 × 800 por cada ojo y una pantalla LCD; y el DK2 a mediados de 2014, con una resolución de 960 × 1080 por ojo y pantalla OLED. Estas versiones fueron lanzada al mercado para que aquellas personas interesadas en el desarrollo de aplicaciones o videojuegos en realidad virtual pudiesen ir creando, aunque al final gran parte de las compras fueron por entusiastas de la tecnología.

Actualmente Oculus Rift está incentivando a los desarrolladores para que utilicen Unreal Engine 4 como plataforma para crear juegos de realidad virtual para ser colgados en la tienda de Oculus. Este incentivo se basa en Oculus cubre los ingresos que deberían hacerse a Epic Games del 5% de las ganancias del juego hasta los 5 millones de dólares.

Figura 29: Monturas Oculus Rift

Ventajas

- Más cómodas y menos pesadas que las HTC Vive.
- Menor precio que HTC Vive.
- Tiene cascos de sonido incluidos.

Desventajas

- No dispone de tecnología *room scale*.

Figura 30: Logotipo de HTC Vive

HTC Vive es una montura de realidad virtual desarrollada por HTC y Valve Corporation que tuvo dos versiones de desarrollo en agosto y septiembre de 2015 y que salió al mercado la versión definitiva en abril de 2016.

Usa un sistema de seguimiento llamado *room scale* que permite al usuario moverse por un espacio 3D e interactuar con el entorno mediante sus mandos. Esto lo consigue gracias a que el dispositivo tiene más de 70 sensores de infrarrojos y un giroscopio MEMS y un acelerómetro. Gracias a dos sensores *lighthouse* el sistema puede trabajar en un área 21 metros cuadrados. Estos sensores permiten rastrear al usuario con una precisión milimétrica y mediante sensores fotoeléctricos puede detectar cualquier objeto del entorno para evitar oclusiones.

También dispone de un sistema de seguridad llamado “Chaperone”, el cual guía a los usuarios para no tropezarse con ningún obstáculo incluso si éstos están en movimiento.

Figura 31: Dispositivo HTC Vive junto a sus mandos

Ventajas

- Dispone de tecnología *room scale*.
- Cámara interna que permite ver el entorno real para evitar colisiones.
- Las monturas permite acercar o alejar las lentes.

Desventajas

- Precio elevado.
- Los auriculares tienen que comprarse de forma independiente.

5.2.5. Tabla-comparación

	Oculus Rift	**HTC Vive**	**PlayStation VR**	**Gear VR**	**Cardboard**
Pantalla y resolución	Pentile Oled 2160 x 1200 píxeles	Pentile Oled 2160 x 1200 píxeles	5’7 pulgadas Oled 1920 x 1080 píxeles	Depende del smartphone	Depende del smartphone
Frecuencia de refresco	90 Hz	90Hz	90Hz-120Hz	Depende del smartphone	Depende del smartphone
Plataforma	Oculus Home	SteamVR	Playstation	Oculus Share	Google Play
Campo de visión	110°	110°	101°	101°	90°
Audio integrado	Sí	No, pero se pueden comprar cascos por separado	No, pero se pueden comprar cascos por separado	No	No
Micrófono integrado	Sí	Sí	No	No	No

Controles	Mando de Xbox One incluido, mandos inalámbricos Oculus Touch	Dos mandos inalámbricos incluidos. Opción mandos compatibles con PC	Mando DualShock 4 o PS Move	Touchpad o controller incluido	-
Conexiones	HDMI y dos puertos USB 3.0	HDMI y dos puertos USB 3.0	Procesador PSVR	MicroUSB	-
Requisitos mínimos del sistema	NVIDIA GTX 960 4 GB/AMD Radeon R9 290, Intel i3-6100/AMD Ryzen 3 1200 FX4350, 8GB RAM	NVIDIA GeForce GTX 1060/AMD Radeon RX 480, Intel Core i5-4590 o AMD FX 8350, 4GB RAM	PlayStation 4 con PlayStation Camera	Galaxy Note 5, S6, S6 Edge, S6 Edge+, S7, S7 EdgeA5, S8, S8+	Smartphone basado en Android o IOS
Precio sin incluir gastos de envío	449 euros con los mandos	699 euros con los mandos	399 euros sin PS Camera (59'99 euros)	129 dólares con controller	De 5 a 30 euros
Lanzamiento última versión	Abril 2016	Abril 2016	Octubre 2016	Marzo 2017	-

Figura 32: Tabla-comparación de las características del hardware de realidad virtual analizados

5.2.6. Conclusiones

Habiendo analizado distintos sistemas cabe mencionar que la potencia y la interacción de las gafas con dependencia móvil no serían compatible con las funcionalidades del proyecto diseñado. Por lo tanto las Google Cardboard y Gear VR no son ninguna una opción viable.

Debido a que el juego no ha sido diseñado para consolas sino para versiones de escritorio, las Sony VR tampoco son compatibles con el diseño inicial del proyecto.

Las diferencias entre Oculus Rift y HTC Vive son escasas, por lo que podría escogerse cualquiera de las dos sin que afectase ello a los requisitos funcionales del proyecto.

Finalmente he escogido Oculus Rift debido a que la tutora de este proyecto me ha facilitado su obtención y las características de mi estación de trabajo se ajustan mejor a los requisitos recomendados.

6. Desarrollo del proyecto

6.1. Desarrollo general del proyecto

El desarrollo del proyecto comenzó con diversos bocetos que dieron lugar al 3. Diseño del juego. A partir de este documento se crearon los 4. Análisis y especificaciones del sistema: 4.1. Análisis de requisitos funcionales, 4.2. Gestión de riesgos y 4.3. Estimación de costes.

Paralelamente a estos estudios se crearon los 5. Análisis de hardware y software: 5.1. Análisis y estudio de los motores de videojuegos y 5.2. Análisis y estudio del hardware de Realidad Virtual. Durante este proceso me reuní con mi tutora para ver qué posibilidades me ofrecía la universidad para trabajar con un dispositivo de realidad virtual en casa: ella misma disponía de unas Oculus Rift en el despacho las cuales me cedió para realizar las primeras pruebas.

Una vez tomada la decisión de qué motor iba a emplear, procedí a ver diversos tutoriales para Unreal Engine 4 que ofrecía su página oficial. El primero de ellos fue el tutorial "Introduction to Blueprints", en el que se enseñaba a iluminar una lámpara cuando el personaje pasaba cerca de ella. Iba implementando y modificando las funciones de los tutoriales a medida que avanzaban.

El siguiente tutorial que implementé fue "UMG UI Inventory", perteneciente también a la página oficial de Unreal Engine 4. En él se mostraba cómo crear un inventario básico y las barras de vida, maná y energía del personaje.

A partir de aquí comencé a desarrollar mi propio proyecto con ayuda de otros tutoriales y siguiendo los requisitos funcionales, la estimación de riegos y la metodología scrum.

6.2. Implementación y resultado final

6.2.1. Estructura del proyecto

Para asegurar la correcta lectura de este apartado a continuación se explicarán diversos conceptos relacionados con la estructura de un proyecto de Unreal Engine 4:

- **Level:** escenario que puede abrirse y modificarse desde el editor añadiendo actores u otros objetos. Cada nivel contiene un *Game Mode*.
- **Game Instance**: clase a la que se puede acceder y modificar desde todos los niveles, es útil a la hora de compartir información entre ellos.

- **Game Mode:** clase en la que se definen los blueprints *Game Session*, *Game State*, *Player Controller*, *Player State*, *HUD* y *Default Pawn*.
- **Game Session***:* clase encargada de gestionar el ingreso de jugadores y la interfaz del modo online.
- **Game State:** clase responsable de comunicar el estado del juego a los diversos clientes. Puede almacenar información como el número de jugadores conectados, la puntuación del equipo o las misiones que han sido completadas en el juego. No es recomendable para guardar estadísticas específicas de un jugador.
- **Player Controller:** actúa de interfaz entre el Pawn y el jugador, representa las acciones que hace el jugador. Es el encargado de manejar el input de teclado, ratón y mando.
- **Player State:** clase que se asigna a cada jugador de forma individual que contiene la misma estructura para todos. Es útil para guardar información como la vida, el maná o el inventario.
- **HUD**: clase encargada de dibujar los elementos que se superponen a los objetos del juego como la barra de vida o maná. Para su representación se utiliza la clase *widget*.
- **Widget**: canvas en el que se representan los gráficos de la *HUD* y las funciones que van a tener estos elementos.
- **Pawn:** representación física del personaje.
- **SaveGameObject:** crea un objeto .sav en el directorio *Saved* del ejecutable en el que se pueden almacenar datos. Estos perduran aunque el juego se cierre y poder leerlos cuando vuelva a iniciarse.

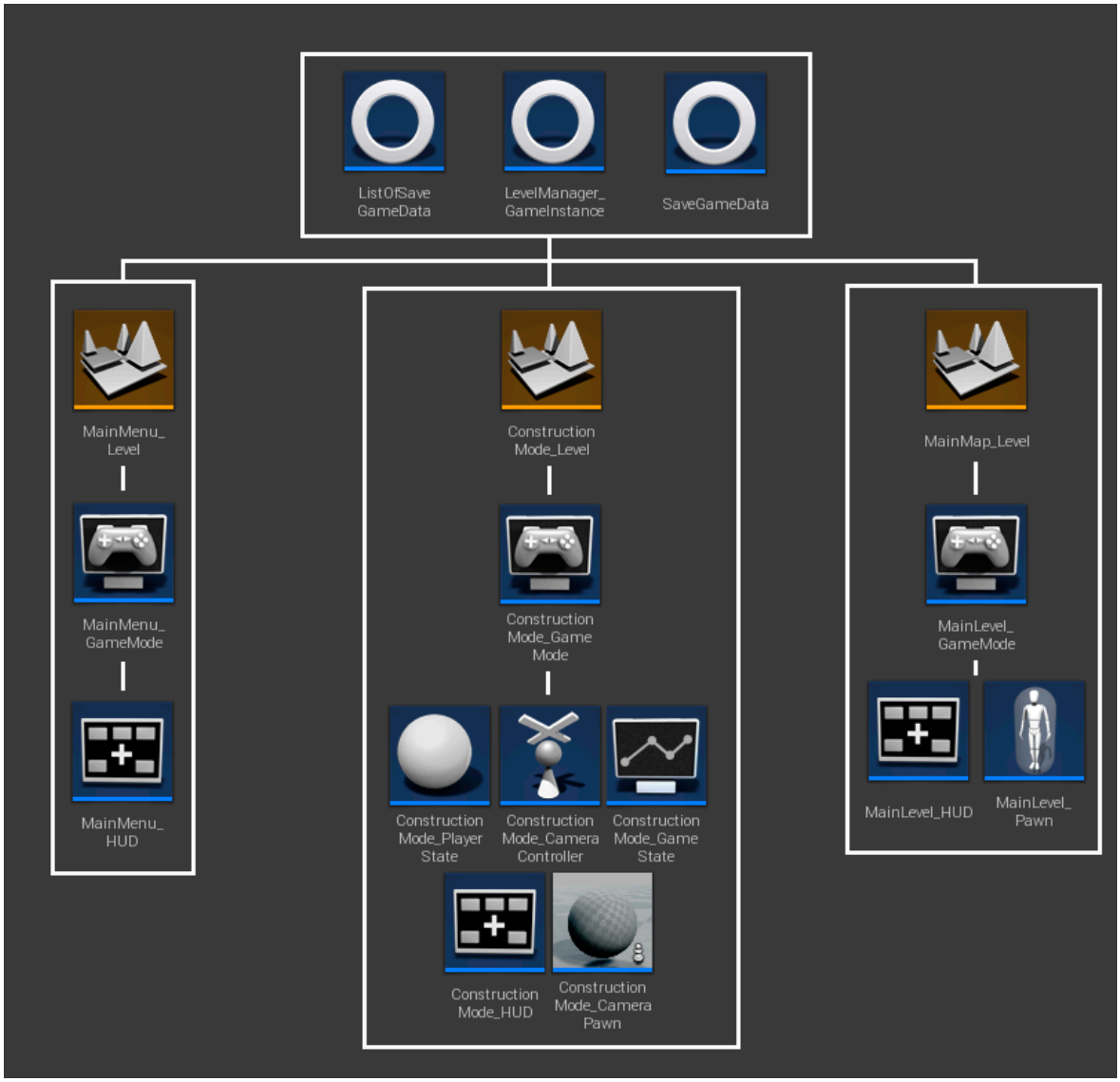


Figura 33: Estructura y componentes de los distintos niveles del Diseñador de mazmorras

6.2.2. Menú Principal

Lo primero que nos encontramos al iniciar el juego es el menú principal (*MainMenu_Level*), compuesto por un fondo (imagen por Epic Games) y cuatro botones (creación propia). Para interactuar con estos botones sólo es necesario pulsar el botón izquierdo del ratón.

Figura 34: Captura del menú principal del diseñador de mazmorras

- **Crear mazmorra:** nos lleva al modo construcción.
- **Cargar mazmorra:** abre una pestaña en la que aparece un listado de los datos guardados.
- **Opciones:** abre una pestaña en la que se muestran las opciones de resolución.
- **Salir:** cierra el juego.

Tanto "opciones" como "cargar mazmorra" pertenecen a este nivel y se encuentran en la misma interfaz. Para navegar entre ellas el programa se encarga de ocultar o mostrar una sección u otra dependiendo del botón que hayamos pulsado.

Cargar mazmorra

Interfaz con un campo desplegable en el que se nos muestran todas las partidas guardadas. Si no hay ningún archivo guardado muestra el mensaje "no hay datos guardados" y no permite desplegar el botón. Una vez hemos elegido el mapa y le hemos dado al botón "cargar", el nombre de éste se guarda en el *LevelManager_GameInstance* para que el *MainMap_Level* sepa que archivo debe leer.

El botón "regresar" nos permite volver al menú principal.

Figura 35: Recorte de pantalla del menú de cargar mazmorra

Opciones

En este menú el usuario puede elegir entre los modos ventana y pantalla completa; y la resolución a la que desea que se muestre el programa. Para que estas configuraciones tengan efecto debe pulsarse el botón “aplicar”. La aplicación guarda las últimas preferencias que se aplicaron y las carga cada vez que se abre el ejecutable.

Figura 36: Recorte de pantalla del menú opciones

6.2.3. Modo construcción

ConstructionMode_Level es un nivel compuesto por un terreno creado mediante la herramienta *landscape* que ofrece el programa y una cámara que nos permite ver todo el terreno.

Cuando este nivel se inicia se crean las clases *ConstructionManager_BP* y *GhostManager_BP*, encargadas de gestionar la colocación de objetos en el mapa.

Cámara aérea

ConstructionMode_CameraPawn contiene una cámara que está unida mediante un *spring arm* a una esfera no visible para el usuario. Esta esfera siempre se encuentra en el centro de la pantalla y es seguida por la cámara a la distancia que indica el *spring arm*.

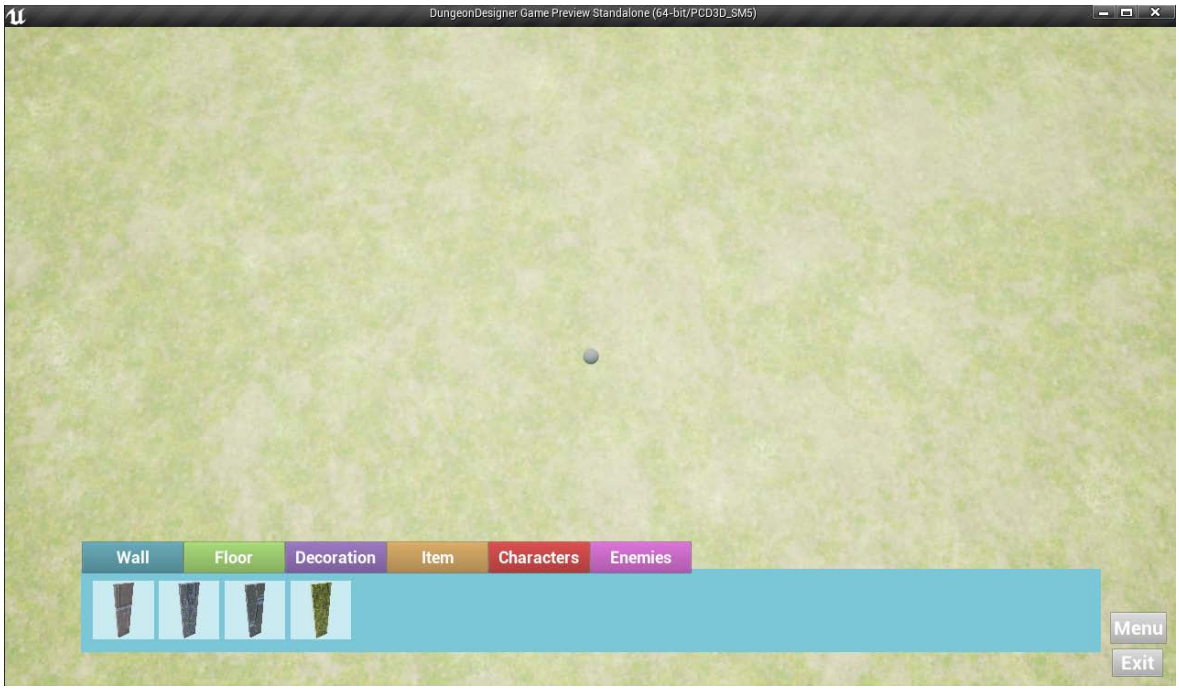


Figura 37: Captura de la cámara aérea y de la esfera que la controla

El controlador de esta cámara, *ConstructionMode_CameraController*, tiene implementado todas las funciones que permiten modificar la vista de la cámara.

Mover la cámara:

Para poder mover la vista y navegar por el mapa se emplean las teclas **WASD**, las **flechas de dirección** o el **ratón**. Éste último permite al usuario mover la cámara acercando el ratón a uno de los extremos de la pantalla.

Pulsando la tecla **mayúscula izquierda mientras nos movemos** aumentaremos la velocidad.

Rotar cámara:

Mover el ratón mientras pulsamos su **botón central** hace rotar la esfera y, por lo tanto, la cámara, permitiéndonos así observar distintas perspectivas.

Pulsando la tecla **control** junto al **botón central del ratón** la rotación volverá a su posición por defecto.

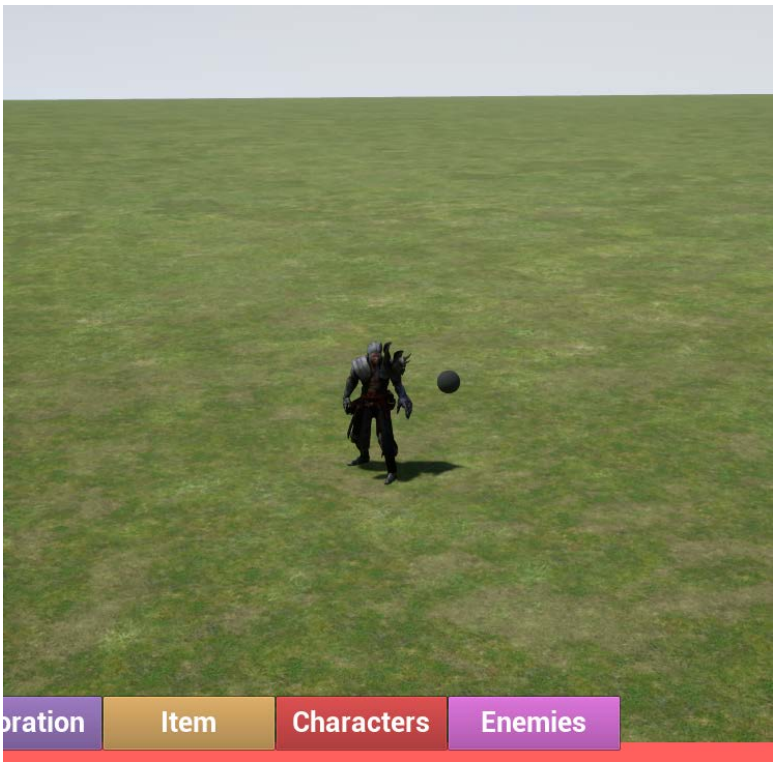


Figura 38: Recorte de pantalla en el que se puede apreciar la rotación de la cámara del modo construcción.

Zoom cámara:

Moviendo la **rueda del ratón** se modifica la distancia entre la cámara y la esfera, lo que permite alejarnos o acercarnos al terreno y sus elementos.

Pulsando la tecla **Alt** junto al botón central del ratón reiniciaremos el zoom a su valor por defecto.

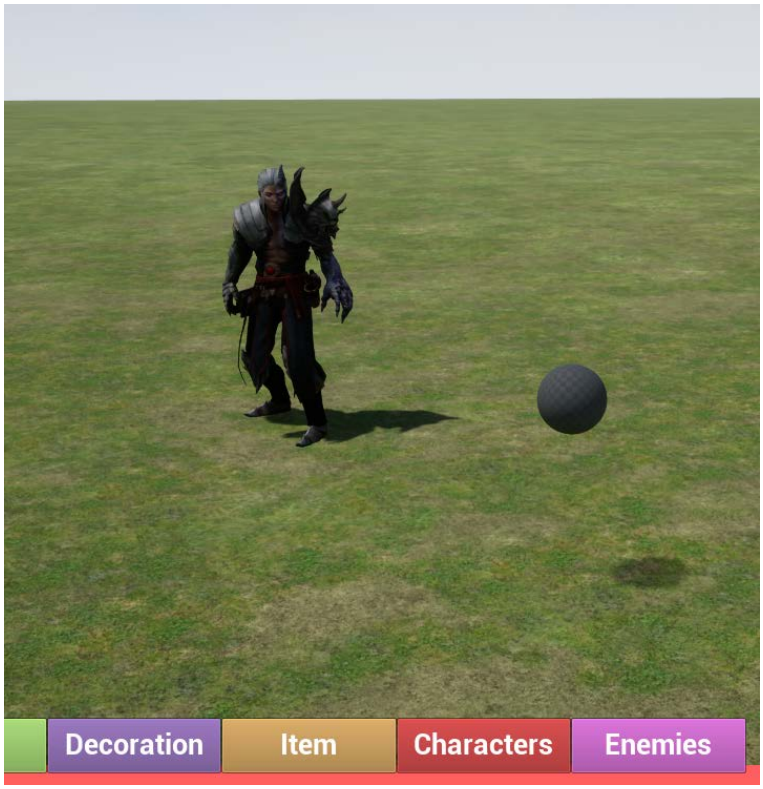


Figura 39: Recorte de pantalla en el que se puede apreciar el zoom de la cámara del modo construcción.

Interfaz de construcción

En la parte inferior de la pantalla encontramos una interfaz que consta de seis botones con texto y una caja que contiene diversos iconos de objetos. Si pulsamos los botones de la parte superior de la caja se cambiará entre pestañas y se mostrarán los iconos de objetos correspondientes a esa sección. La representación gráfica de los iconos ha sido creada a partir de los modelos que se encuentran dentro del juego.

Cuando uno de los iconos es pulsado, la interfaz se encarga de mandarle un mensaje al *GhostManager_BP* comunicándole qué tipo de objeto tiene que crear en el mapa.

Figura 40: Todas las pestañas de la interfaz del modo construcción

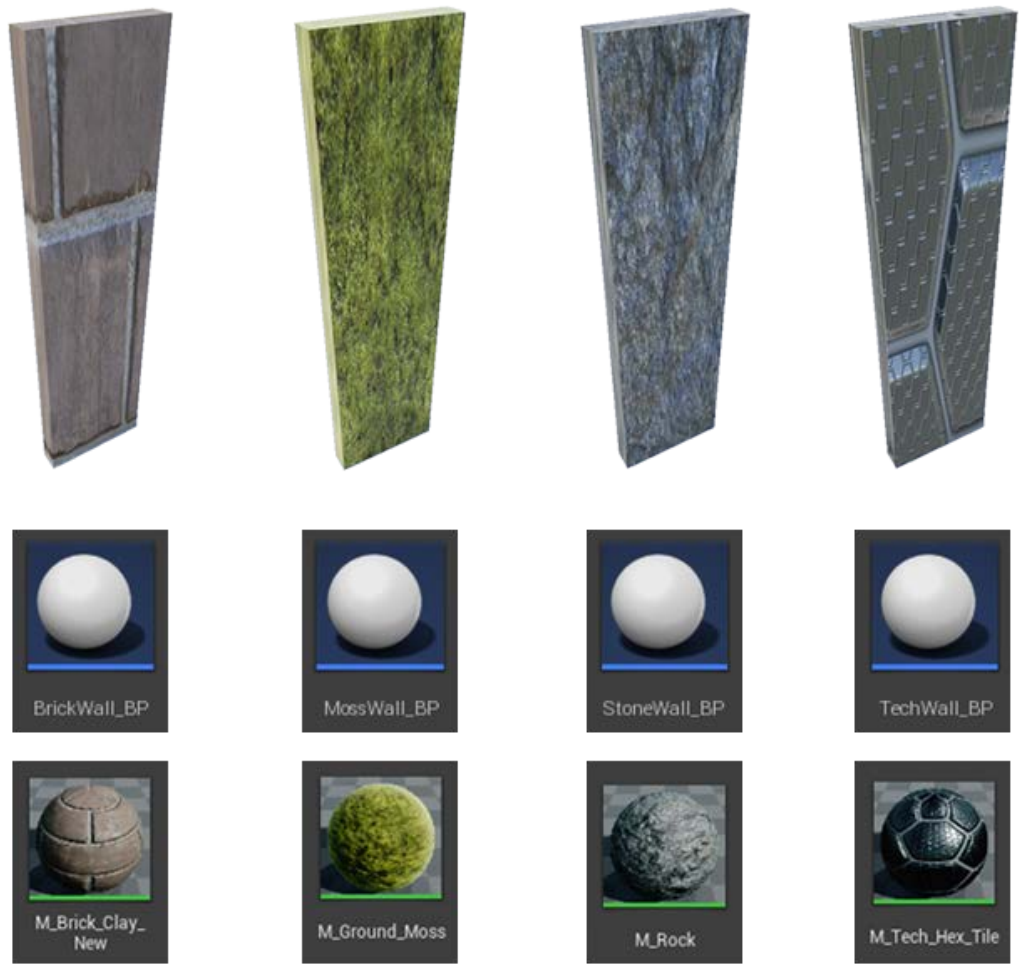
Clases de objetos

Todos los elementos que pueden ponerse en el terreno en el modo construcción y luego cargar en la mazmorra heredan de una misma clase llamada *ConstructionObjectMaster_BP*, la cual gestiona funciones comunes como la asignación o iluminación del modelo. También se encarga de facilitar la creación de actores en la lógica del juego.

Paredes

Las paredes miden 20 de ancho, 100 de largo y 400 de alto. Aunque en el boceto inicial apareciesen distintos modelos de paredes, en la práctica se observó que esto no era viable y por ello sólo encontramos un modelo actualmente.

Las paredes heredan de una clase padre llamada *WallMaster_BP* que permite que todos sus hijos sean extensibles a lo largo.

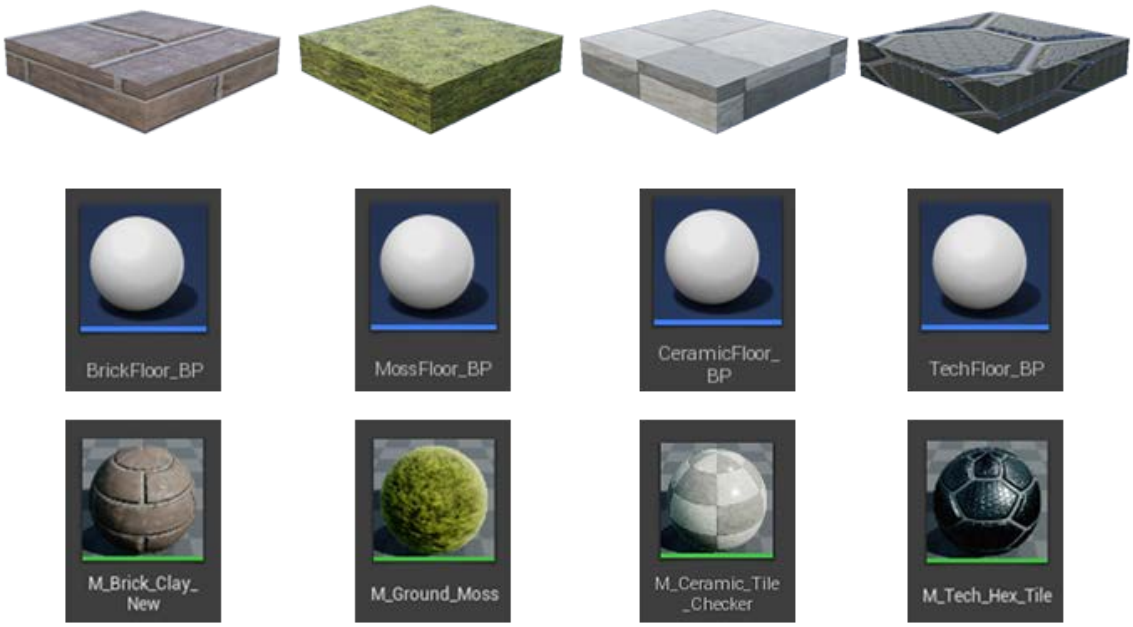


Modelo y blueprints: Beatriz Sabater Serna
Materiales: Epic Games modificado por Beatriz Sabater Serna

Figura 41: Modelos y materiales de las paredes

Suelos

Heredan de la clase *FloorMaster_BP*, la cual permite que sean extensibles en el eje X e Y, cubriendo una superficie cuadrada.

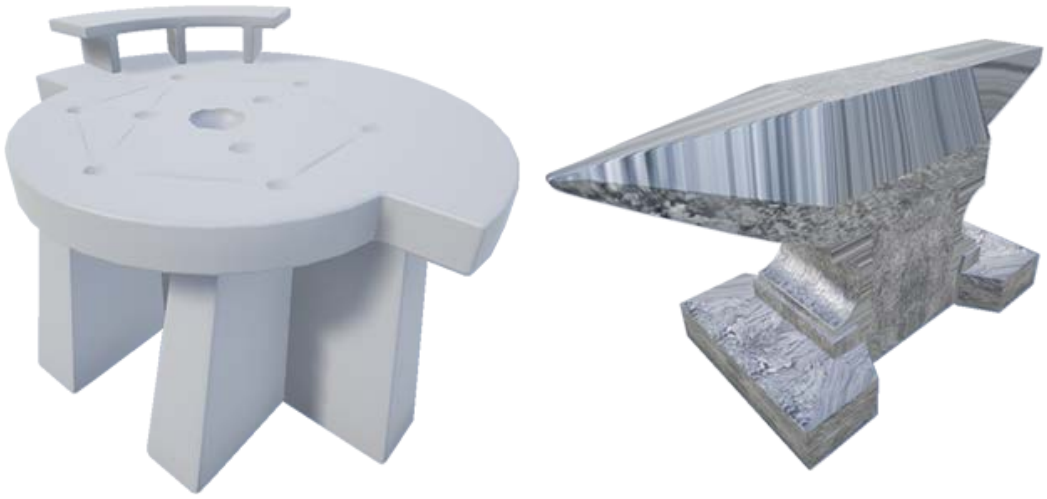


Modelo y blueprints: Beatriz Sabater Serna
Materiales: Epic Games modificado por Beatriz Sabater Serna

Figura 42: Modelos y materiales del suelo

Decoración

DecorationMaster_BP no tiene ninguna funcionalidad compartida implementada.



Modelos: Beatriz Sabater Serna
Material del yunque: Epic Games modificado por Beatriz Sabater Serna

Figura 43: Modelos de decoración yunque y mesa de alquimia

Modelos: Alexis Martín

Figura 44: Modelos de decoración caja y cofre por Alexis Martín

Ítems

Los objetos que heredan de la clase *ItemMaster_BP* pueden ser recogidos por el jugador y añadidos a su inventario. Esta clase a su vez se divide en: *ConsumableMaster_BP*, que son aquellos objetos que modifican el estado del personaje; y *WeaponMaster_BP*, clase no implementada pero que serviría para poder equipar armas al personaje.

Modelos: Beatriz Sabater Serna
Material de las gemas: Beatriz Sabater Serna

Figura 45: Modelos de ítems The Cracker (espada) y gemas

Modelo y material del martillo: Epic Games

Modelo y material del pan: CGTextures.com

Figura 46: Modelos de ítems Temperance (martillo) y pan

Personajes

Personajes no manejables que poseen animaciones dentro de la mazmorra creada. La clase de la que heredan, *CharacterMaster_BP*, tiene dos mayas: una *static mesh* para posicionarlo en el modo construcción, y una *skeletal mesh* para asignarle animaciones cuando se cargue la mazmorra.

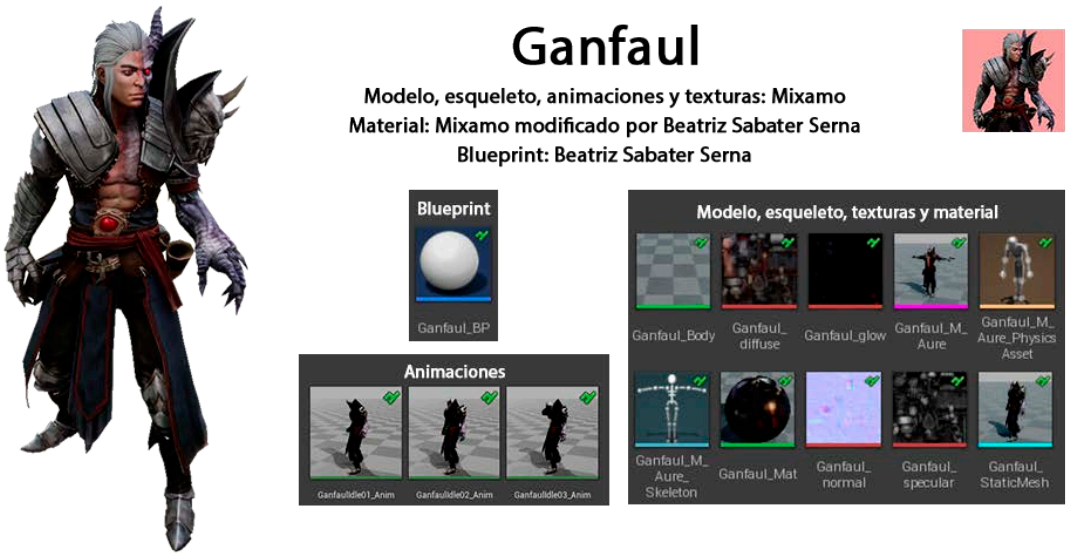


Figura 47: Componentes del personaje Ganfaul

Figura 48: Componentes del personaje Kachujin

Figura 49: Componentes del personaje Arissa

Figura 50: Componentes del personaje Paladin

Enemigos

Animales o monstruos que heredan de la clase *EnemyMaster_BP*. Actualmente no tiene ninguna función implementada, pero en un futuro podría integrarse la inteligencia artificial en este tipo de elemento. Tienen dos mayas como los personajes.

Modelo, esqueleto, animaciones y texturas: Epic Games
Material: Epic Games modificado por Beatriz Sabater Serna

Figura 51: Modelo de los enemigos lobo, oso, araña grande y gruntlin

Posicionar elemento: objetos fantasma

Cuando el *GhostManager_BP* recibe el mensaje desde la interfaz crea un *GhostConstructionObject_BP* con la misma representación gráfica que el objeto que le ha comunicado que debe crear. Cuando un objeto fantasma está en juego se actualiza de forma continua su posición dependiendo dónde esté el ratón en pantalla.

Figura 52: Objeto definitivo (izquierda) frente a un objeto fantasma (derecha)

Deseleccionar objeto fantasma

Si existe al menos un objeto fantasma en el mapa y se pulsa el **botón derecho del ratón**, el objeto dejará de estar seleccionado y desaparecerá.

Posición errónea del objeto fantasma

Cuando un objeto fantasma se encuentra en el mismo espacio que otro elemento del mapa, automáticamente cambia su color a rojo y no permite colocar el objeto en el terreno.

Figura 53: El color rojo en un objeto fantasma indica que su posición no es correcta

Colocar objeto

Si existe al menos un objeto fantasma en el terreno y su posición no es errónea, al pulsar el **botón izquierdo del ratón** la clase *GhostManager_BP* mandará a construir al *ConstructionManager_BP* todos los elementos fantasma a partir de la información inicial que mandó la interfaz.

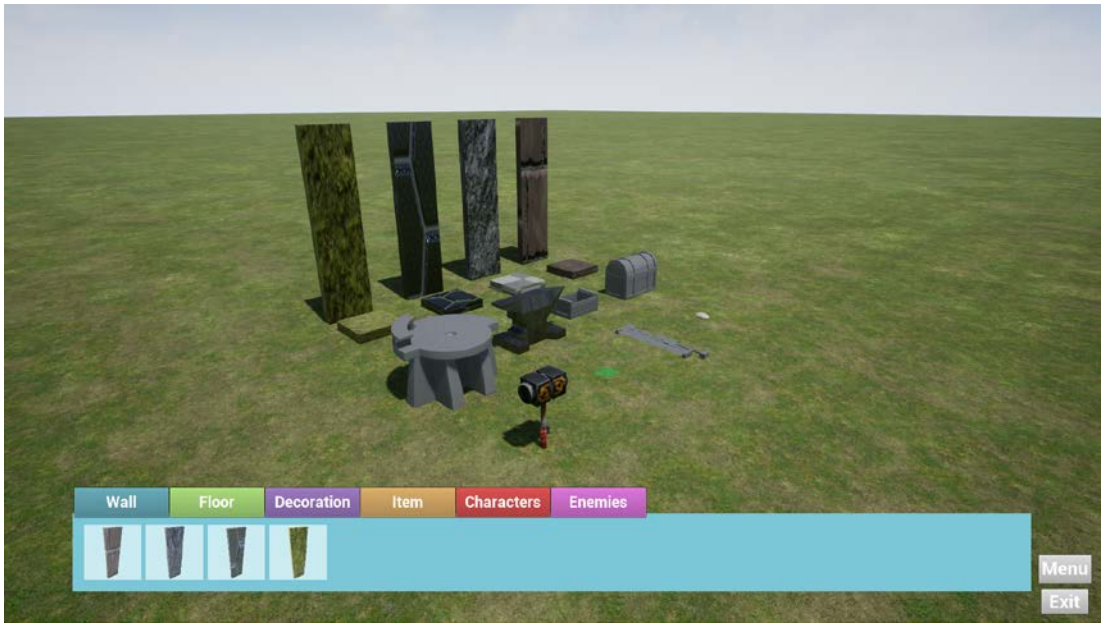


Figura 54: Varios objetos colocados en el mapa

Eliminar objeto del mapa

Si por equivocación se coloca un elemento en el terreno puede borrarse con el botón derecho del ratón siempre y cuando no haya ningún objeto fantasma en el mapa.

Para indicar cuál es el objeto que sería borrado éste parpadea con una luz amarilla cada vez que se pasa el ratón por encima.

Figura 55: Iluminación de un objeto al pasar el cursor por encima suya

6.2.4. Mapa cargado

El nivel *MainMap_Level* abre el archivo de guardado *SaveGameData* con el nombre que lee del *LevelManager_GameInstance* y se encarga de cargar todos los objetos. Si cuando se cambió de nivel había unas gafas de realidad virtual conectadas el juego, se cambiará automáticamente a la cámara en primera persona con realidad virtual y se ocultarán las interfaces; si no, el juego se cargará de forma normal.

Figura 56: Interfaz del mapa cargado que muestra la barra de salud, energía y maná del personaje

Figura 57: Interfaz del mapa cargado que muestra los objetos del inventario

Figura 58: Jugador disfrutando del mapa que ha creado con las Oculus Rift

6.2.4. Salvado/Guardado

La primera vez que se inicia el modo construcción el juego crea un archivo de guardado *ListOfSaveGameData* llamado "SlotMaster", el cual está preparado para contener todos los nombres de todos los archivos de guardado. Si la tecla **K** es pulsada el juego crea un nuevo archivo de guardado *SaveGameData* al que le asigna el nombre "Slot_X", donde X es el último número del archivo guardado más uno. Este archivo de guardado contiene un array con la referencia a las clases de los objetos que se han creado a la par que sus coordenadas.

Para listar los mapas en la pantalla de cargar mazmorra, se llama al "SlotMaster" que tiene todos los nombres de los mapas guardados.

6.2.5. Problemas encontrados

Oculus Rift

Debido a que compartí las monturas con otros compañeros del grado no pude realizar las pruebas definitivas hasta la semana anterior a la entrega. Al realizar estas comprobaciones el proyecto no funcionaba, por lo que tuve que dedicarme exclusivamente a solucionar este fallo durante un par de días.

Creación del ejecutable

Cuando intenté crear por primera vez el ejecutable del juego para Windows daba error. Después de investigar el fichero en el que se muestran los errores de compilación me di cuenta de que el problema era que algunos nombres de archivo tenían una ruta demasiada larga, por lo que tuve que cambiarlos de ubicación.

7. Conclusiones

7.1. Objetivos alcanzados

Todos los objetivos propuestos en el apartado 1.2. Objetivos fueron realizados: se crearon el análisis de motores de videojuegos y el de hardware de realidad virtual; aprendí a utilizar el motor de videojuegos Unreal Engine 4 y su lenguaje de programación; y el creador de mazmorras, la mazmorra creada y el sistema de guardado y carga de mapas son completamente funcionales.

No obstante un requisito del apartado 4.1. Análisis de requisitos funcionales no se pudo implementar debido a su complejidad y a la falta de tiempo:

- [F08] Posicionamiento del techo encima de las paredes.

7.2. Líneas futuras de trabajo

El *Diseñador de mazmorras* fue ideado para ser una base de algo mucho más grande, por lo que el número de mejoras que se podrían implementar son muchas.

La primera mejora que llevaría a cabo sería la sustitución de los modelos, texturas y animaciones actuales que no me pertenecen por unos propios; así como crear nuevos para añadirlos al modo construcción y que hubiese mayor variedad de elementos que pueden agregarse al mapa.

La siguiente modificación que haría en el proyecto sería implementar los objetos de tipo techo para poder crear habitaciones cerradas y que hubiese objetos que pudiesen posicionarse en las paredes como antorchas.

A continuación comenzaría a desarrollar las mecánicas de algunos elementos de la mazmorra empezando por lo más sencillo como abrir un cofre, hasta dinámicas más complicadas como la inteligencia artificial de los enemigos.

También habría que revisar el código para encontrar posibles fallos y arreglaros, así como para optimizarlos. Una sección en la que haría especial énfasis sería en revisar las colisiones de objetos fantasmas.

Por último agregaría un sistema de red para que los jugadores pudiesen hacer grupos de 4 personas y jugar de forma simultánea a una mazmorra para poderse ayudar entre ellos ya que el rol de manual suele ser una experiencia en equipo.

7.3. Posibles formas de comercialización

- **De pago:** los usuarios pagan una única vez para obtener la aplicación en alguna plataforma como Steam y se habilita algún espacio para que los jugadores puedan subir sus mazmorras.

- **Gratuito con opción de pago por contenido:** el programa es gratuito y tiene un pack de objetos base. El usuario puede comprar más packs de objetos para añadir a su juego y tener más posibilidades de crear mazmorras.

- De pago con compra de contenido: el programa es de pago y además se ofrecen packs que pueden ser comprados.

7.4. Opinión personal

A lo largo del proyecto me he planteado en varias ocasiones si éste era demasiado complejo o grande y que tal vez no sería capaz de cumplir todos los objetivos. También llegué a discurrir sobre si al final del proyecto sentiría insatisfacción debido a que su planteamiento era una pequeña parte de lo que algún día quería implementar.

Afortunadamente no se ha dado ninguno de los dos casos y estoy muy satisfecha con el resultado obtenido; incluyendo el hecho de haber podido experimentar con Unreal Engine 4 y su lenguaje propio *Blueprint*. Me han parecido dos herramientas muy potentes y me he sentido muy a gusto desarrollando con ellas, por lo que planeo emplearlas en futuros proyectos y ampliar mis conocimientos sobre ellas.

En conclusión, la experiencia del trabajo de fin de grado ha sido muy enriquecedora y estoy orgullosa del trabajo realizado.

8. Bibliografía y referencias

Brown, L. (2017). *A brief history of virtual reality*. [En línea] Wondershare. Disponible en: https://filmora.wondershare.com/virtual-reality/history-of-vr.html

Cryengine. (n.d.). *The complete solution for next generation game development by Crytek*. [En línea] Disponible en: https://www.cryengine.com/

Downie, C. (2016). *Nuevos tipos de suscripción y precios de Unity por lanzarse en Junio – Unity Blog*. [En línea] Unity Technologies Blog. Disponible en: https://blogs.unity3d.com/es/2016/05/31/new-products-and-prices/

Epic Games. (n.d.). *Game Engine Technology by Unreal*. [En línea] Unreal Engine Official Page. Disponible en: https://www.unrealengine.com/

Fine, R. (2017). *UnityScript's long ride off into the sunset – Unity Blog*. [En línea] Unity Technologies Blog. Disponible en: https://blogs.unity3d.com/es/2017/08/11/unityscripts-long-ride-off-into-the-sunset/

Google VR. (n.d.). *Daydream*. [En línea] Disponible en: https://vr.google.com/daydream/

Google VR. (n.d.). *Google Cardboard – Google VR*. [En línea] Disponible en: https://vr.google.com/intl/es_es/cardboard/

Graft, K. (2016). *Oculus, Epic make deal to give Unreal Engine devs a royalties break*. En línea] Gamasutra. Disponible en: https://www.gamasutra.com/view/news/282872/Oculus_Epic_make_deal_to_give_Unreal_Engine_devs_a_royalties_break.php

Lowood, H. (2014). *Game Engines and Game History | Kinephanos*. [En línea] Kinephanos.ca. Disponible en: http://www.kinephanos.ca/2014/game-engines-and-game-history/

Lozano Ortega, M. (2016). *Motores para videojuegos*.

Norman, K. (2016). *Game Engines: Past, Present and Future*. [En línea] Gamasutra.com. Disponible en: https://www.gamasutra.com/blogs/KevinNormann/20160412/270186/Game_Engines_Past_Present_and_Future.php

Oculus. (n.d.). *Oculus*. [En línea] Disponible en: https://www.oculus.com/

Pauly, L. (2009). *Doom to Dunia: A Visual History of Game Engines*.

Proyectos Ágiles. (n.d.). *Qué es SCRUM*. [En línea] Disponible en: https://proyectosagiles.org/que-es-scrum/

Riccitiello, J. (2015). *Unity 5 Launch – Unity Blog*. [En línea] Unity Technologies Blog. Disponible en: https://blogs.unity3d.com/es/2015/03/03/unity-5-launch/

Samsung. (n.d.). *Samsung Gear VR with Controller*. [En línea] The Official Samsung Galaxy Site. Disponible en: http://www.samsung.com/global/galaxy/gear-vr/

Sarathi Paul, P., Goon, S. and Bhattachary, A. (2012). *History and comparative study of modern game engines*. [En línea] Bipublication. Disponible en: https://bipublication.com/files/IJCMS-V3I2-2012-07.pdf

Schreier, J. (2016). *Cite a Website - Cite This For Me*. [En línea] Kotaku. Disponible en: http://kotaku.com/cryteks-video-game-engine-is-now-free-1765078659

Sempere Tortosa, M.L. (2016). Introducción a la Realidad Virtual.

Sony PlayStation. (n.d.). PlayStation Camera. [En línea] Disponible en: https://www.playstation.com/es-es/explore/accessories/playstation-camera/

Sony PlayStation. (n.d.). PlayStation®VR. [En línea] Disponible en: https://www.playstation.com/es-es/explore/playstation-vr/

Trends Staff, D. (2017). Oculus Rift vs. HTC Vive: Prices are lower, but our favorite remains the same. [En línea] Digital Trends. Disponible en: https://www.digitaltrends.com/virtual-reality/oculus-rift-vs-htc-vive/

Unity. (n.d.). *Unity - Game Engine*. [En línea] Unity Official SiteDisponible en: https://unity3d.com/es/

Virtual Reality Society. (n.d.). *History Of Virtual Reality - Virtual Reality Society*. [En línea] Virtual Reality Society. Disponible en: https://www.vrs.org.uk/virtual-reality/history.html

Vive, H. (2016). *HTC Vive on Steam*. [En línea] Steam Store. Disponible en: http://store.steampowered.com/app/358040/HTC_Vive/

Wikipedia. (n.d.). *CryEngine*. [En línea] Disponible en: https://en.wikipedia.org/wiki/CryEngine

Wikipedia. (n.d.). *Game engine*. [En línea] Disponible en: https://en.wikipedia.org/wiki/Game_engine

Wikipedia. (n.d.). *List of CryEngine games*. [En línea] Disponible en: https://en.wikipedia.org/wiki/List_of_CryEngine_games

Wikipedia. (n.d.). *List of Unity games*. [En línea] Disponible en: https://en.wikipedia.org/wiki/List_of_Unity_games#2017

Wikipedia. (n.d.). *List of Unreal Engine games*. [En línea] Disponible en: https://en.wikipedia.org/wiki/List_of_Unreal_Engine_games

Wikipedia. (n.d.). *The Elder Scrolls V: Skyrim*. [En línea] Wikipedia. Disponible en: https://en.wikipedia.org/wiki/The_Elder_Scrolls_V:_Skyrim

Wikipedia. (n.d.). *The Sims*. [En línea] Disponible en: https://en.wikipedia.org/wiki/The_Sims

Wikipedia. (n.d.). *Unity (game engine)*. [En línea] Wikipedia. Disponible en: https://en.wikipedia.org/wiki/Unity_(game_engine)

Wikipedia. (n.d.). *Unreal Engine*. [En línea] Wikipedia. Disponible en: https://es.wikipedia.org/wiki/Unreal_Engine

Wikipedia. (n.d.). *Google Cardboard*. [En línea] Disponible en: https://en.wikipedia.org/wiki/Google_Cardboard

Wikipedia. (n.d.). *Doom engine*. [En línea] Disponible en: https://en.wikipedia.org/wiki/Doom_engine

Wikipedia. (n.d.). *Frostbite (game engine)*. [En línea] Disponible en: https://en.wikipedia.org/wiki/Frostbite_(game_engine)

Wikipedia. (n.d.). *HTC Vive*. [En línea] Disponible en: https://en.wikipedia.org/wiki/HTC_Vive

Wikipedia. (n.d.). *Samsung Gear VR*. [En línea] Disponible en: https://en.wikipedia.org/wiki/Samsung_Gear_VR

Wikipedia. (n.d.). *Source 2*. [En línea] Disponible en: https://en.wikipedia.org/wiki/Source_2

Wikipedia. (n.d.). *Source (game engine)*. [En línea] Disponible en: https://en.wikipedia.org/wiki/Source_(game_engine)

Wikipedia. (n.d.). Cave automatic virtual environment. [En línea] Disponible en: https://en.wikipedia.org/wiki/Cave_automatic_virtual_environment

Wikipedia. (n.d.). Daniel J. Sandin. [En línea] Disponible en: https://en.wikipedia.org/wiki/Daniel_J._Sandin#The_Sayre_Glove

Wikipedia. (n.d.). Google Daydream. [En línea] Disponible en: https://en.wikipedia.org/wiki/Google_Daydream

Wikipedia. (n.d.). ISmell. [En línea] Disponible en: https://en.wikipedia.org/wiki/ISmell

Wikipedia. (n.d.). Oculus Rift. [En línea] Disponible en: https://es.wikipedia.org/wiki/Oculus_Rift

Wikipedia. (n.d.). PlayStation Camera. [En línea] Disponible en: https://en.wikipedia.org/wiki/PlayStation_Camera

Wikipedia. (n.d.). PlayStation VR. [En línea] Disponible en: https://en.wikipedia.org/wiki/PlayStation_VR