Escuela Politécnica Superior de Jaén
Grado en Ingeniería Informática

Universidad de Jaén
Escuela Politécnica Superior
de Jaén

TRABAJO FIN DE GRADO

DESARROLLO DE VIDEOJUEGO PARA REALIDAD VIRTUAL CON INTERACCIÓN MULTIMODAL

Alumno

Raúl Fernández Cobo

Tutores

Juan Roberto Jiménez Pérez
(Departamento de Informática)

Juan Manuel Jurado Rodríguez
(Departamento de Informática)

Septiembre, 2022

(Página intencionalmente en blanco)

Universidad de Jaén

Departamento de Informática

Don Juan Roberto Jiménez Pérez y Don Juan Manuel Jurado Rodríguez, tutores del Trabajo Fin de Grado titulado: '**Desarrollo de videojuego para realidad virtual con interacción multimodal**', que presenta Don Raúl Fernández Cobo, otorgan el visto bueno para su entrega y defensa en la Escuela Politécnica Superior de Jaén.

Jaén, Septiembre de 2022

El alumno: Los tutores:

Raúl Fernández Cobo

Juan Roberto Jiménez Pérez

Juan Manuel Jurado Rodríguez

(Página intencionalmente en blanco)

Agradecimientos

Me gustaría dar las gracias a mi familia y amigos, siempre presentes con su apoyo incondicional durante todo este recorrido, sin ellos esto no hubiera sido posible.

(Página intencionalmente en blanco)

FICHA DEL TRABAJO FIN DE TÍTULO	
Titulación	Grado en Ingeniería Informática
Modalidad	Proyecto de Ingeniería
Especialidad (solo TFG)	Ingeniería del Software
Mención (solo TFG)	Sistemas Gráficos
Idioma	Español
Tipo	Específico
TFT en equipo	No
Autor/a	Raúl Fernández Cobo
Fecha de asignación	17/03/2022
Descripción corta	Este trabajo presenta el desarrollo del prototipo de un videojuego. En particular, se desarrolla un videojuego válido para realidad virtual y en el que el diseño de la interacción del jugador con el juego es un aspecto destacado, proporcionando una interacción multimodal.

NORMAS APLICADAS EN ESTE DOCUMENTO	
LOCALES	
TFT-UJA:2017	Normativa de Trabajos Fin de Grado, Fin de Máster y otros Trabajos Fin de Título de la Universidad de Jaén (Normativa marco UJA aprobada en Consejo de Gobierno)
TFT-EPSJ:2017	Normativa sobre Trabajos Fin de Grado y Fin de Máster en la Escuela Politécnica Superior de Jaén (Normativa EPSJ aprobada en Junta de Escuela)
TFT-EPSJ	Criterios de evaluación y normas de estilo para TFG y TFM de la Escuela Politécnica Superior de Jaén
NACIONALES E INTERNACIONALES	
ISO 2145:1978	Documentación - Numeración de divisiones y subdivisiones en documentos escritos
UNE 50132:1994	Traducción de la ISO 2145
APA 6ª edición	Estilo de referencias y citas de APA (American Psychological Association)

NORMAS UTILIZADAS COMO BASE O REFERENCIA	
NACIONALES	
UNE 157001:2014	Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico
UNE 157801:2007	Criterios generales para la elaboración de proyectos de sistemas de información
Estas normas se han utilizado como base o referencia para la inclusión de algunos contenidos y definiciones sobre elaboración de proyectos, entendiendo como proyecto la documentación consensuada entre una empresa y un cliente, que da lugar al perfeccionamiento de un contrato para la elaboración de una obra o la prestación de un servicio. Por consiguiente, no debe esperarse la aplicación de estas normas en cuanto a la completitud de los contenidos ni a la organización de los mismos.	

Contenido

Índice de ilustraciones

Índice de tablas

(Página intencionalmente en blanco)

1 INTRODUCCIÓN

(Página intencionalmente en blanco)

En este capítulo se presenta una introducción general al trabajo, aportando los conocimientos generales necesarios para la correcta comprensión del documento.

A lo largo de la presente memoria se utilizarán términos y acrónimos cuya descripción aparecen en el Capítulo 6 (Definiciones y abreviaturas).

1.1 Motivación

En la actualidad, la industria de los videojuegos experimenta un auge gracias a la amplia versatilidad de dispositivos que hacen posible una visualización e interacción cada vez más realista. Cualquier persona con una conexión a internet y un ordenador o dispositivo móvil tiene la posibilidad de jugar a un videojuego. En España en 2019, el sector de los videojuegos supuso 1479 millones de euros sumando la venta online y física [1]. En cuanto a la realidad virtual como industria, aunque todavía es un sector pequeño, está experimentando un mayor crecimiento estos últimos años. El mercado VR (Virtual Reality) generó 1.8 billones de dólares en todo el mundo durante 2020, lo cual supuso un aumento del 31,7% con respecto a 2019. De esos 1.8 billones de dólares 816 millones fueron generados por los videojuegos de realidad virtual. Sin embargo, el gasto en juegos VR sigue representando una pequeña parte (0,4%) de los ingresos generados por los fabricantes de videojuegos, esto se debe a que todavía son pocos los títulos capaces de obtener una calidad adecuada [2]. Además, el hardware necesario para jugar a estos juegos no está al alcance de todo el mundo siendo el precio medio de unas gafas VR de unos 600€.

Pese a las modestas cifras actuales de beneficio se prevé que la realidad virtual tenga un papel fundamental en la revolución tecnológica en la que estamos inmersos. Prueba de ello es el proyecto Metaverso liderado por Facebook, una de las empresas más relevante en el sector de las Tecnologías de la Información (TI). Se pretende ofrecer al usuario una novedosa interfaz de interacción y comunicación en entornos virtuales inmersivos. En este contexto, el desarrollo de aplicaciones multimodales es fundamental haciendo posible utilizar nuestros sentidos para percibir lo que sucede en el escenario y actuar de la manera más intuitiva posible.

El afán de conocimiento y la curiosidad sobre el funcionamiento de estas tecnologías han motivado el desarrollo de este proyecto, cuyo fin es el de crear una experiencia virtual abordando una interacción multimodal. “*La interacción multimodal es un proceso en el cual las personas y los dispositivos llevan a cabo una interacción*

conjunta combinando diversos formatos y estímulos (auditivos, visuales, táctiles y gestuales) que incrementan así el rango de la interacción y el contenido que se transmite" (Peraldo, 2010).

El desarrollo de un videojuego en realidad virtual con interacción multimodal plantea una serie de retos adicionales a los que plantea el desarrollo de un videojuego convencional. Es necesario identificar la interacción óptima para cada momento y asegurar la inmersión total del jugador en el entorno VR durante la experiencia.

Este documento expone la toda documentación relacionada con el desarrollo del proyecto.

1.2 Objetivos del trabajo

El principal objetivo de este proyecto es el **diseño e implementación de un prototipo de un videojuego para realidad virtual con interacción multimodal**. Previo al desarrollo se ha llevado a cabo un estudio sobre el estado actual del desarrollo de videojuegos en realidad virtual, así como un análisis de las posibles interacciones del jugador con el entorno 3D. A continuación, se definen los objetivos específicos:

- **Revisión de soluciones VR y aplicaciones multimodales**: Será necesario analizar las distintas opciones y escoger la más apropiada para el proyecto.
- **Diseño y desarrollo de un videojuego en 3D:** Se comenzará creando un proyecto 3D en Unity sobre el que más tarde se incorporará la VR.
- **Adaptación del videojuego a Realidad Virtual:** Una vez estén creados los aspectos básicos del videojuego (Entorno, Personajes, Historia, ...) se incorporará la Realidad Virtual.
- **Analizar y diseñar la interfaz e interacción con el usuario, así como identificar los mecanismos de interacción más adecuados al tipo de videojuego que se propone.**
- **Desarrollo de la interacción multimodal y análisis de la jugabilidad a partir de una valoración final de los jugadores:** Se realizará una encuesta para determinar la opinión de los jugadores con respecto a las interacciones que el videojuego ofrece.

Para el desarrollo de este proyecto serán necesarios conocimientos en Unity 3D, así como nociones básicas de informática gráfica relacionadas con el desarrollo de experiencias en realidad virtual. También será conveniente conocer y tener clara la metodología a usar y conocimientos sobre ingeniería del software para el correcto análisis, diseño e implementación del software.

1.3 Antecedentes y estado del arte

En esta sección se explican los orígenes de los aspectos relevantes para el proyecto y se exponen los últimos avances y trabajos relacionados que sirven de punto de partida para el desarrollo del presente proyecto.

1.3.1 Contexto histórico

Pese a la relativa novedad que puede suponer la realidad virtual, hay que remontarse al 1929 para encontrar el primer prototipo de un entorno en **VR** (*Virtual Reality*). En ese año, Edward Link creó el primer ***"Link Trainer"***, un simulador de vuelo mecánico usado para el ejército norteamericano, con el que se entrenaron más de 500.000 soldados.

Ilustración 1.1 – Simulador de vuelo "Link Trainer" [1]

Más tarde, en 1956, Morton Heilig sorprendió con su ***"Sensorama"***. Heilig, que se movía por la industria cinematográfica de Hollywood, quería averiguar cómo hacer

[1] https://www.britannica.com/technology/Link-Trainer

a la gente sentir como si estuvieran dentro de la película. El ***"Sensorama"*** simulaba una ciudad real por la que el usuario paseaba en una moto. Además, contaba con la estimulación de otros sentidos como el olfato o el tacto.

Ilustración 1.2 – "Sensorama" de Morton Heilig [2]

En 1965, otro inventor, Ivan Sutherland, creó **"*The Ultimate Display*"**, un dispositivo que consistía en un casco acoplado a un ordenador. El casco estaba sujeto al techo con un brazo mecánico que seguía la rotación de la cabeza.

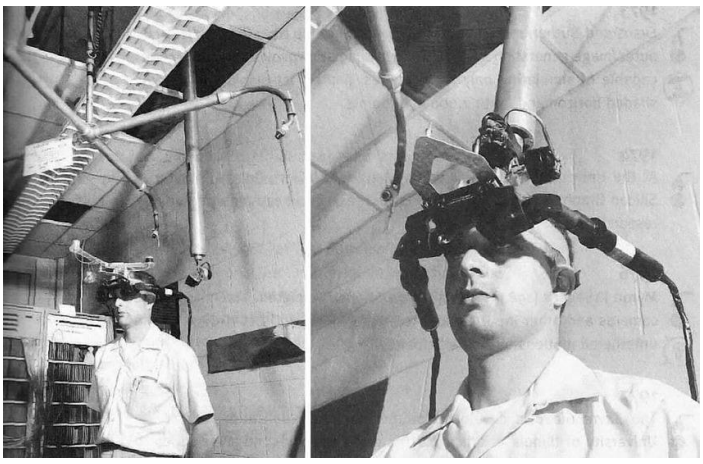


Ilustración 1.3 – Dispositivo "The Ultimate Display" de Ivan Sutherland [3]

[2] https://www.britannica.com/technology/Link-Trainer
[3] https://www.researchgate.net/figure/Ivan-Sutherlands-head-mounted-3D-display-c-1968-The-display-had-a-suspending_fig1_337438550

A principios de los 90, aparecieron los primeros videojuegos en VR de la mano de Sega y Nintendo con su ***"Sega VR"*** y ***"Virtual Boy"*** respectivamente, sin embargo, seguía siendo un sector a la sombra de los videojuegos convencionales.

Ya en 2012 Palmer Luckey presentó el primer prototipo de Oculus Rift, cuya versión para clientes no se lanzaría hasta 2015. Es a partir de este momento cuando las compañías líderes en el sector de los videojuegos comienzan a trabajar en sus propios modelos de VR y en 2016 Sony lanzaría su PlayStation VR, mientras que HTC y Valve lanzarían el HTC Vive.

Ilustración 1.4 – Gafas HTC Vive [4]

1.3.2 Últimos avances de la realidad virtual

Actualmente la realidad virtual no solo está presente en el sector de los videojuegos, sino que es relevante en áreas como la medicina, la arquitectura, la educación, o el turismo. A continuación, se exploran más a fondo algunos de los campos donde la VR se está haciendo hueco en estos últimos años y se muestran algunos ejemplos en la *ilustración 1.5*.

Uno de los ámbitos donde la VR se está comenzado a utilizar es el de la psicología [3]. Esta se está empleando en algunos centros para tratar enfermedades como fobias, trastornos de ansiedad o adicciones. De este modo se expone al paciente a un estímulo fóbico o traumático en un ambiente controlado, donde puede intervenir el profesional en cualquier momento. Ante esta exposición progresiva el

[4] https://www.xataka.com/realidad-virtual-aumentada/htc-vive-las-gafas-vr-mas-completas-las-podras-comprar-en-abril-por-799-dolares

paciente reacciona de modo que se reducen lentamente los umbrales de ansiedad. Otro de los usos de la VR es el turismo. Es posible viajar a los rincones más inhóspitos del mundo desde el sofá de casa. Y, aunque la experiencia no sea la misma, cuenta con la ventaja de conocer lugares intransitables como ruinas de antiguos lugares emblemáticos gracias a la recreación por ordenador. De igual manera, la VR puede aportar grandes beneficios a la educación. Es una herramienta más con la que los estudiantes pueden aprender conceptos y desarrollar sus habilidades. Existen entornos en VR como **Anatomyou,** que permite a los alumnos entender la anatomía humana a través de una navegación inmersiva por el cuerpo humano o **Titans of Space**, que ofrece la oportunidad de realizar una visita guiada por el sistema solar y conocer la estructura de los planetas [4]. Otra de las áreas donde actualmente se está aplicando la realidad virtual es en la medicina, por ejemplo, para el entrenamiento en cirugía. Es posible crear experiencias virtuales realistas para el entrenamiento de los cirujanos sin los riesgos que supone esta clase de entrenamientos en la vida real. Por último, en el ámbito de la arquitectura, la VR ofrece la posibilidad de contemplar un proyecto antes de su realización y de este modo presentarlo o evitar errores críticos de diseño. Existen para ello herramientas como **Edificius VR**, un software de arquitectura que permite desarrollar proyectos detallados en un entorno VR [5].

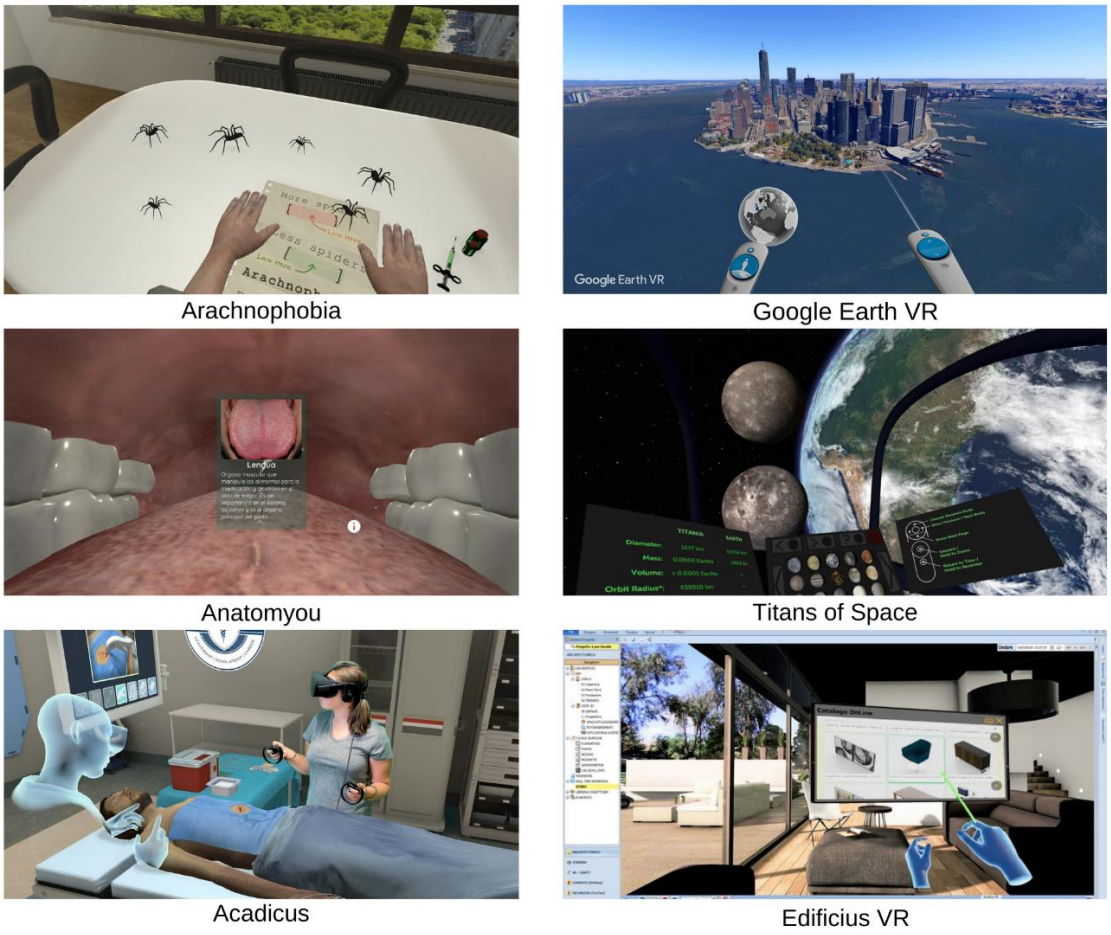


Ilustración 1.5 - Collage con imágenes de algunas aplicaciones VR

1.3.3 Interacción multimodal

Cuando se habla de interacción multimodal, se hace referencia a aquella interacción entre una persona y un dispositivo en la que intervienen diversos formatos y estímulos. Actualmente son muy pocos los avances de VR con interacción multimodal.

Sus aplicaciones suelen estar enfocados a fines educativos, por ejemplo, en 2019, desde el departamento de Inteligencia Artificial e Ingeniería del Software de la Universidad Complutense de Madrid se diseñó un entorno de VR para entrenar a las personas a hablar en público [6]. Este entorno, mostrado en la *ilustración 1.6,* posicionaba al jugador en diversas situaciones en las que tenía que hablar públicamente, como una exposición en clase, o una entrevista de trabajo. El público, representado con esferas, reaccionaba en tiempo real a través del análisis de diferentes parámetros del orador, como su tono de voz, el contenido del discurso o la dirección de su mirada.

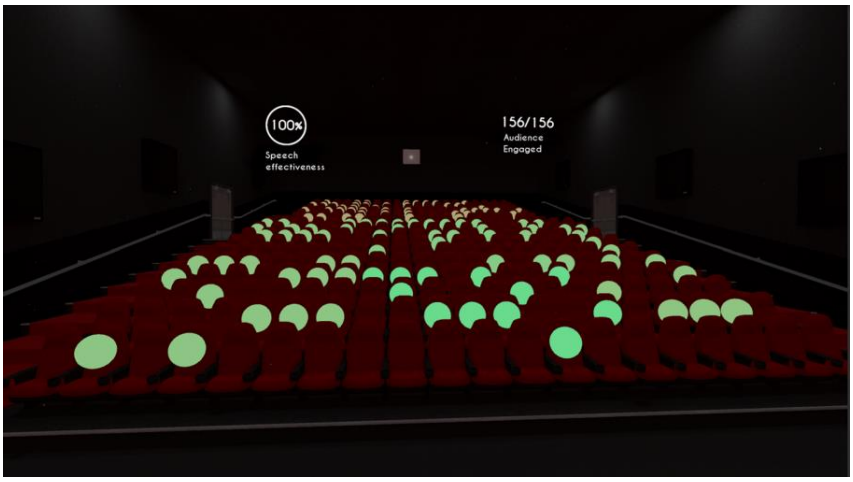


Ilustración 1.6 - Entorno virtual de entrenamiento del discurso [6]

Otro ejemplo de interacción multimodal reciente es el uso del "*Eye Tracking"*. Esta tecnología analiza los movimientos oculares del usuario y extrae información relevante a partir de esos datos. Se puede saber qué está mirando el usuario o deducir si algo le está impresionando, por ejemplo. En una conferencia de prensa de HTC en 2019 donde presentaron su HTC Vive Pro VR (modelo que incorpora esta tecnología), presentaron algunos de los usos del *"Eye Tracking"*. Se presentó la técnica "*Foveated rendering*" que incrementa la resolución de una imagen decrementando la resolución de las imágenes que se encuentran fuera del campo de visión de los ojos. Esto permite que la imagen sea mucho más nítida.

En un estudio realizado en la Universidad de Zaragoza se explora la importancia de la multimodalidad en entornos VR [7]. En él se resumen los últimos avances de investigación sobre la multimodalidad en VR y se resaltan aquellas ideas más relevantes de cara al futuro. También se mencionan las consecuencias que puede tener una experiencia multimodal si esta no está diseñada correctamente, pudiendo distraer al usuario y empeorar la experiencia final en lugar de mejorar la inmersión.

En otro estudio desarrollado en la Universidad de Oporto se realiza un estudio del uso de una interacción *Hands-free* (manos libres) en entornos de realidad virtual [8]. Para ello se identifican distintos tipos de interfaces *hands-free* utilizadas en los entornos VR, se determinan para qué tareas de interacción se utilizan esas interfaces y se identifican qué métricas se usan para evaluarlas. La ventaja de estas técnicas

hands-free es que permiten realizar tareas a los usuarios mientras tienen sus manos disponibles para otras funciones. El método de interacción más estudiado es el uso de la voz, tanto a través del reconocimiento del discurso como de comandos de voz. Se utiliza la voz para acciones como navegar por menús, confirmar la selección de un objeto, o para escribir texto directamente trascrito de la voz, por ejemplo. Otros métodos muy estudiados son el uso del seguimiento de los ojos o del seguimiento de la cabeza, que se usan también para tareas de selección e incluso se combinan con el uso de la voz (seleccionar con la mirada y confirmar con la voz). De hecho, estos dos últimos métodos resultan ser más satisfactorios para los usuarios en tareas de selección que los controladores comunes.

En conclusión, no hay duda de que estas tecnologías cuentan con un increíble potencial y unas capacidades asombrosas y que, a pesar de tratarse de unos avances muy recientes y estar todavía en vías de desarrollo, seguro supondrán un antes y un después en la historia de la Realidad Virtual.

1.4 Alcance

El alcance de este proyecto abarca la implementación de un videojuego en realidad virtual junto con las funcionalidades necesarias para que el usuario pueda interactuar con el juego por medio de los controladores y por medio de la voz. Es decir, la interacción será multimodal, permitiendo al usuario elegir el tipo de interacción con el sistema en ciertas mecánicas. Además, se llevará a cabo una propuesta preliminar de una herramienta de procesamiento de lenguaje natural con la que poder evaluar la satisfacción del jugador a partir de palabras clave recogidas en un corpus.

1.5 Tecnologías utilizadas

A continuación, se describen brevemente las tecnologías y herramientas utilizadas en el proyecto:

- **Unity 3D**: Es un motor gráfico de videojuegos multiplataforma creado por Unity Technologies. Este permite el diseño, creación y funcionamiento de un entorno interactivo. Permite crear videojuegos en 2D, 3D e incluso para Realidad Virtual. Cuenta con una gran comunidad y una documentación actualizada y

excelente. Este motor ha sido seleccionado por su numerosa documentación y por la experiencia previa en su uso en el grado.

- **Visual Studio Code**: Editor de código utilizado para la creación y edición de los distintos scripts empleados en el proyecto. Es el editor de código normalmente utilizado con Unity.
- **Plastic SCM**: Herramienta de control de versiones y gestión de código empleada durante el desarrollo del trabajo. Esta herramienta está integrada perfectamente con el motor Unity, lo cual facilita su uso y es el motivo principal de su elección.
- **SteamVR:** Herramienta para experimentar contenido en Realidad Virtual en diferentes dispositivos hardware. Se ha escogido esta herramienta por tener soporte para las gafas de Realidad Virtual empleadas en el proyecto.
- **C#:** Es el lenguaje de programación empleado en el proyecto. Es el utilizado en Unity y además se cuenta con experiencia previa.
- **Visual Paradigm:** Herramienta para la creación de todo tipo de diagramas. Se ha escogido por su uso y previa experiencia en el grado.
- **Paint3D:** Herramienta para la edición de imágenes 2D y 3D. Se ha escogido por su facilidad de uso.
- **Audacity:** Herramienta para la edición de audio. Se ha escogido por ser un software libre y de fácil uso.
- **Windows Speech API**: es una API creada por Microsoft que ofrece herramientas para el reconocimiento de voz y el procesamiento del discurso.
- **SteamVR Plugin**: Plugin mantenido por Valve que permite conectar sin problemas la herramienta SteamVR con Unity.

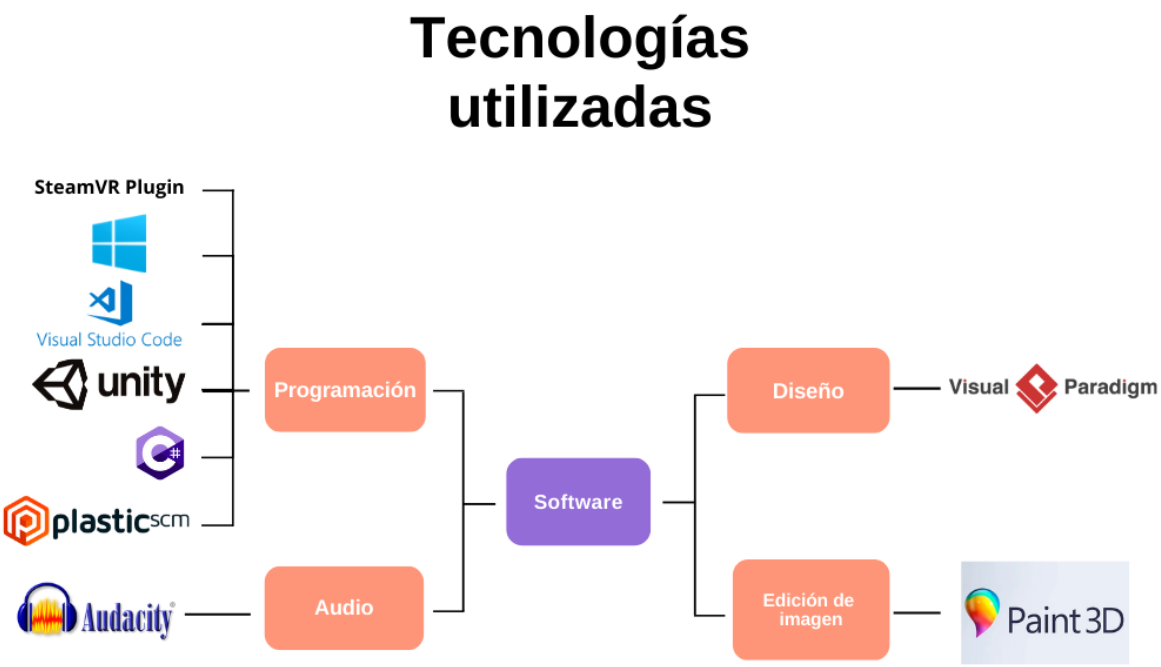


Ilustración 1.7 - Esquema gráfico de las tecnologías utilizadas

1.5.1 Resumen de las deficiencias y carencias identificadas

La principal deficiencia identificada es la innovación que supone un sistema de interacción multimodal y, por tanto, la escasa documentación, así como de herramientas profesionales para su implementación.

Otra deficiencia encontrada es la ineficiencia que presentan algunas herramientas de reconocimiento del lenguaje, al tratarse de tecnologías en desarrollo. Estas no logran reconocer las palabras del jugador cuando existe ruido de fondo o cuando la pronunciación no es perfecta. Esto obliga al usuario a jugar en una zona sin ruido y a forzar la pronunciación para que las herramientas funcionen correctamente.

Otra carencia identificada es la poca documentación del plugin SteamVR, utilizado para la creación del entorno VR.

1.6 Metodología de desarrollo de software

En este apartado se describe la metodología software empleada, así como la justificación de las razones por las que se ha seleccionado.

Para el desarrollo de este proyecto se ha seleccionado la metodología **Kanban**, debido principalmente a su flexibilidad y versatilidad, características muy útiles ya que el proyecto es desarrollado por una sola persona. La idea principal consiste en definir los requisitos al principio del proyecto, permitiendo cambios durante el desarrollo y estructurar el proyecto en incrementos que generen cada uno una versión funcional.

1.6.1 ¿Qué es Kanban?

Kanban es una metodología ágil de desarrollo de software centrada en la entrega "*just-in-time*" de la funcionalidad y la gestión de "*Work In Progress*" [9]. Esta metodología se basa en la mentalidad **Lean** en la que se basan también otras metodologías como Scrum o XP. El principal objetivo de **Lean** es encontrar problemas en el método de trabajo y resolverlos, es decir, reducir el esfuerzo y/o recursos hacia objetivos que no están produciendo valor para el cliente.

En esta metodología se emplean los llamados tableros Kanban de los cuales se hablará más adelante. Kanban no proporciona un desarrollo iterativo, sino incremental. Para cada incremento se representan las diferentes etapas del ciclo de vida del desarrollo de software. El objetivo es limitar la cantidad de trabajo realizado en cada una de estas etapas para controlar y propiciar un desarrollo fluido, y que de este modo no se produzcan cuellos de botella. Estos límites se llaman "*Work In Progress* (WIP)" y son la medida de la capacidad del equipo para trabajar en varias tareas al mismo tiempo. Solo cuando se completan las tareas, las nuevas se incorporan al ciclo.

1.6.2 Ventajas de Kanban

Algunas las ventajas que ofrece Kanban frente a otras metodologías y por las cuales se ha seleccionado esta para el proyecto son las siguientes:

- **Roles**: Kanban no define roles, lo cual facilita el desarrollo en proyectos con un solo miembro.

- **Cambios durante incrementos**: En Kanban pueden ocurrir cambios en cualquier momento sin que esto suponga un problema para el desarrollo
- **Eventos**: Kanban tampoco define eventos ni reuniones lo cual he considerado apropiado debido a que solo hay un miembro en el proyecto.
- **Duración de etapas**: En Kanban no se define una cadencia de entrega, de este modo un incremento puede durar 3 días y otro 3 semanas, por ejemplo.

Estas características proporcionan una flexibilidad que se han considerado convenientes para el desarrollo de este proyecto.

1.6.3 Tablero Kanban

Kanban es una combinación de dos palabras japonesas 看 (Kàn), que significa "signo" o "señal visual", y 板 (Bǎn), que significa "tablero" [10]. Esta metodología se implementa usando los llamados tableros Kanban como ya se ha mencionado anteriormente. Estos tableros se organizan por columnas donde cada una representa una etapa del trabajo. En la *ilustración 1.8* se muestra un ejemplo de tablero Kanban creado en Trello.

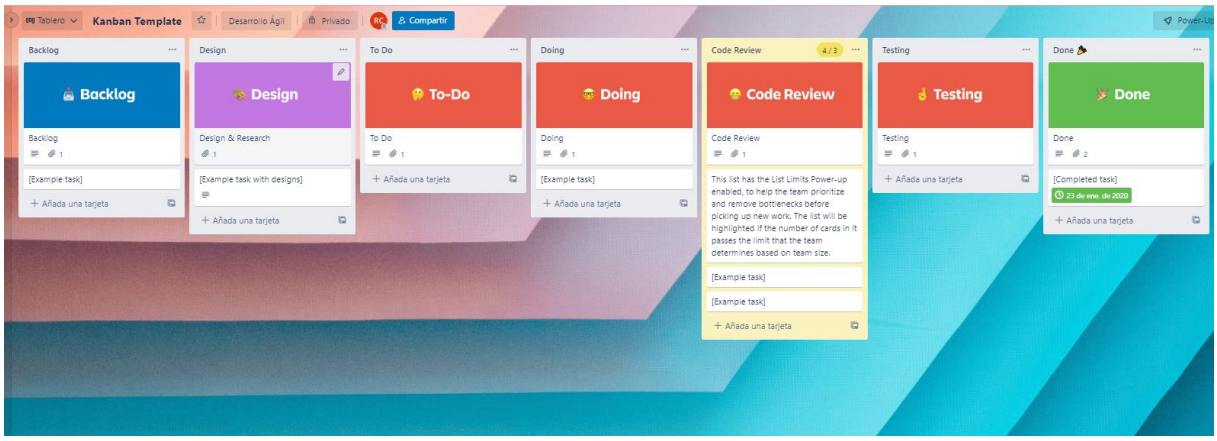


Ilustración 1.8 - Tablero Kanban

1.6.4 Kanban en el proyecto

Para el desarrollo de este proyecto se ha tomado la decisión de establecer la mayor parte de los requisitos al comienzo del proyecto pero con la implementación por incrementos, que aportarían cada uno una versión funcional del sistema. Una vez redactados los requisitos se les asignará a cada uno una puntuación no lineal de entre los siguientes valores numéricos: **{1,2,3,5,8,13,20}**, siendo **20** el valor máximo y **1** el

valor mínimo. De este modo se asociará a cada requisito una puntuación para su **coste** y otra para su **valor.** Por cada uno de los incrementos se seleccionarán un subconjunto de requisitos del conjunto de requisitos iniciales y se ordenarán según su valor y coste. Estos incrementos tendrán duraciones diferentes y sin ninguna limitación.

1.7 Visión general del videojuego

En esta sección se menciona brevemente la intención general del videojuego, así como obras de un estilo parecido.

El videojuego se titula "**Mystery Island VR**". La aventura se desarrolla en una isla con cuatro zonas bien diferenciadas. El objetivo principal es entrar en un templo que se encuentra en el centro de la isla. Para ello es necesario conseguir tres objetos, que se encuentran repartidos por la isla. Para conseguir estos objetos será necesario eliminar enemigos o realizar algunas misiones. El entorno VR garantiza una experiencia inmersiva al jugador y permite la interacción con distintos elementos del juego. Además, se ofrece una interacción multimodal con la que el jugador podrá hacer uso de la voz, además de los controles convencionales, para realizar acciones durante el transcurso de la experiencia.

El estilo del juego es de aventura en primera persona. Está diseñado para todos los públicos. Algunos títulos VR que han inspirado el videojuego son "**No Man's Sky VR"** *(ilustración 1.9)* y **"The Elder Scrolls V: Skyrim VR"** *(ilustración 1.10)*

Ilustración 1.9 - Imágenes del videojuego "No Man's Sky VR" [5]

[5] https://blog.latam.playstation.com/2019/04/02/asi-es-como-funciona-no-mans-sky-en-playstation-vr/

Ilustración 1.12 – Imagen del videojuego "Skyrim VR" [6]

1.8 Estructura de la memoria

Para facilitar la comprensión y lectura del documento se comentará a continuación la **estructura** del mismo:

- **Capítulo 1. Introducción**: En este capítulo se presenta la especificación del trabajo, así como todos los conocimientos necesarios para su correcta comprensión.
- **Capítulo 2. Análisis, planificación y presupuesto**: En este apartado se presentarán los aspectos relacionados con el análisis del sistema a desarrollar y la obtención de los requisitos, que servirán de base para establecer una planificación y elaborar un presupuesto.
- **Capítulo 3. Ejecución del proyecto**: En este punto se especifica todo lo relativo al desarrollo del proyecto, incluyendo información sobre cada uno de los incrementos realizados a lo largo del proyecto, así como las pruebas de validación y verificación asociadas a estos.
- **Capítulo 4. Conclusiones y trabajos futuros**: En esta sección se expone una conclusión general del proyecto, así como una estimación de los contenidos que han quedado pendientes por hacer en un futuro.

[6] https://elderscrolls.bethesda.net/en/article/4YSRrlXtpuKeWIK6EMiG84/skyrim-vr-comes-to-steamvr

- **Capítulo 5. Apéndices**: En esta sección se compone de información adicional que puede resultar útil para la correcta comprensión del trabajo.
- **Capítulo 6. Definiciones y abreviaturas**: En este capítulo se presenta una relación de la terminología específica, definiciones, abreviaturas, etc., que se han utilizado a lo largo del trabajo, así como su significado.
- **Capítulo 7. Bibliografía**: En este punto se expone la bibliografía utilizada para la realización del proyecto.

2 ANÁLISIS, PLANIFICACIÓN Y PRESUPUESTO

(Página intencionalmente en blanco)

En este capítulo se presentan los aspectos relacionados con el análisis del sistema a desarrollar y la obtención de los requisitos, que sirven de base para establecer una planificación y elaborar un presupuesto.

2.1 Especificaciones del sistema

2.1.1 Requisitos funcionales y no funcionales

Los requisitos son aquellos aspectos que el sistema debe cumplir al final de la realización del trabajo, y que sirven de referencia para su validación final, así como para la estimación de un presupuesto. A continuación, se presentan dos tipos de requisitos:

- Requisitos funcionales: Son aquellos requisitos que describen cómo debe comportarse el sistema.
- Requisitos no funcionales: No describen una funcionalidad, sino la calidad que se espera de los servicios que va a ofrecer el sistema

Los requisitos elaborados para el sistema se describen a continuación. La plantilla utilizada para documentar los requisitos es la siguiente:

ID	FRQ-XXX
Descripción	
Versión	
Información adicional	
Dependencias	
Prioridad	

Tabla 2.1 - Plantilla para la documentación de requisitos

ID	FRQ-001 Control de la cámara
Descripción	El jugador podrá controlar la cámara
Versión	V1
Información adicional	Con el movimiento del visor de VR se moverá la cámara
Dependencias	
Prioridad	Muy alta

Tabla 2.2 - FRQ-001

ID	FRQ-002 Movimiento del personaje
Descripción	El jugador podrá mover su personaje en todas direcciones
Versión	V1
Información adicional	A través de un touchpad el jugador podrá mover su personaje en cualquier dirección
Dependencias	
Prioridad	Muy alta

Tabla 2.3 - FRQ-002

ID	FRQ-003 Sprint
Descripción	El jugador podrá esprintar pulsando una tecla específica
Versión	V1
Información adicional	Al pulsar la tecla aumentará la velocidad del personaje.
Dependencias	FRQ-002
Prioridad	Muy alta

Tabla 2.4 - FRQ-003

ID	FRQ-004 Interacción con objetos
Descripción	El jugador podrá interactuar con algunos objetos
Versión	V1
Información adicional	En el mapa se definirán algunos objetos con los que el jugador podrá interactuar
Dependencias	
Prioridad	Muy alta

Tabla 2.5 - FRQ-004

ID	FRQ-005 Arco
Descripción	El jugador podrá disparar flechas usando un arco
Versión	V1
Información adicional	Una de las armas principales es un arco, que permitirá al jugador disparar flechas infinitas
Dependencias	FRQ-004
Prioridad	Muy alta

Tabla 2.6 - FRQ-005

ID	FRQ-006 Espada
Descripción	El jugador podrá empuñar una espada para atacar a los enemigos
Versión	V1
Información adicional	Una de las armas principales es una espada, que permitirá al jugador dañar a sus enemigos
Dependencias	FRQ-004
Prioridad	Muy alta

Tabla 2.7 - FRQ-006

ID	FRQ-007 Interacción por voz
Descripción	El sistema permitirá al jugador interactuar con el juego por medio de la voz
Versión	V1
Información adicional	Durante la partida, el jugador podrá usar comandos de voz para interactuar con el juego
Dependencias	
Prioridad	Alta

Tabla 2.8 - FRQ-007

ID	FRQ-008 Menú de pausa
Descripción	El jugador podrá acceder a un menú de pausa pulsando una tecla específica o usando la voz
Versión	V1
Información adicional	El sistema definirá una función que permita mostrar el menú de pausa tanto pulsando un botón como diciendo la palabra “menú”
Dependencias	FRQ-007
Prioridad	Alta

Tabla 2.9 - FRQ-008

ID	FRQ-009 Información útil
Descripción	El jugador podrá visualizar información útil pulsando una tecla específica
Versión	V1

Información adicional	El sistema definirá una función que permita mostrar información útil tanto pulsando un botón
Dependencias	
Prioridad	Media

Tabla 2.10 - FRQ-009

ID	FRQ-010 Menú radial
Descripción	El jugador podrá acceder a un menú radial y cambiar de arma pulsando una tecla específica
Versión	V1
Información adicional	Se diseñará un menú radial a través del cual el jugador podrá visualizar las armas disponibles y cambiar entre ellas
Dependencias	
Prioridad	Media

Tabla 2.11 - FRQ-010

ID	FRQ-011 Cambio de arma por voz
Descripción	El jugador podrá cambiar de arma usando la voz
Versión	V1
Información adicional	Se definirán comandos de voz para que el jugador pueda cambiar de arma utilizando estos.
Dependencias	FRQ-007
Prioridad	Media

Tabla 2.12 - FRQ-011

ID	FRQ-012 Ataque
Descripción	El jugador podrá atacar y eliminar enemigos usando sus armas
Versión	V1
Información adicional	
Dependencias	
Prioridad	Alta

Tabla 2.13 - FRQ-012

ID	FRQ-013 Tutorial
Descripción	El sistema deberá ofrecer al jugador la posibilidad de realizar un tutorial al inicio del juego
Versión	V1
Información adicional	
Dependencias	
Prioridad	Alta

Tabla 2.14 - FRQ-013

ID	FRQ-014 Muerte
Descripción	El jugador podrá morir si se dan las condiciones para ello
Versión	V1
Información adicional	Si el jugador recibe daño hasta perder todos sus puntos de vida este morirá y tendrá la opción de reiniciar el juego
Dependencias	
Prioridad	Alta

Tabla 2.15 - FRQ-014

ID	FRQ-015 Diálogo
Descripción	El sistema generará paneles con el diálogo de los NPC cuando el jugador esté cerca de ellos
Versión	V1
Información adicional	
Dependencias	
Prioridad	Alta

Tabla 2.16 - FRQ-015

ID	FRQ-016 Salir juego
Descripción	El sistema permitirá al jugador salir del juego a través del menú de pausa
Versión	V1
Información adicional	
Dependencias	FRQ-008
Prioridad	Alta

Tabla 2.17 - FRQ-016

ID	FRQ-017 Contador de piedras
Descripción	El sistema almacenará en todo momento el número de piedras sagradas que el jugador ha recolectado
Versión	V1
Información adicional	El jugador debe conseguir las 3 piedras sagradas que hay en el mapa para acceder al templo
Dependencias	FRQ-004
Prioridad	Alta

Tabla 2.18 - FRQ-017

ID	FRQ-018 Reanudar partida
Descripción	El sistema permitirá al jugador reanudar la partida una vez está en el menú de pausa pulsando un botón o usando la voz
Versión	V1
Información adicional	Una vez en el menú de pausa, el jugador puede pulsar el botón de reanudar o decir la palabra “reanudar” para reanudar el juego
Dependencias	FRQ-007
Prioridad	Media

Tabla 2.19 - FRQ-018

ID	FRQ-019 Teletransporte
Descripción	El sistema permitirá al jugador teletransportarse a las zonas habilitadas cuando este pulse una tecla específica
Versión	V1
Información adicional	Por el mapa habrá puntos de teletransporte a los que el jugador podrá teletransportarse pulsando un botón y apuntando con el controlador
Dependencias	
Prioridad	Alta

Tabla 2.20 - FRQ-019

ID	FRQ-020 Sonido
Descripción	El sistema permitirá al controlar el volumen de la música del juego
Versión	V1

Información adicional	En el menú se habilitará una opción para ajustes de sonido
Dependencias	
Prioridad	Baja

Tabla 2.21 - FRQ-020

ID	FRQ-021 Partículas de daño
Descripción	El sistema deberá generar partículas cuando se dañe a los enemigos
Versión	V1
Información adicional	
Dependencias	FRQ-012
Prioridad	Baja

Tabla 2.22 - FRQ-021

ID	FRQ-022 Escenario abierto
Descripción	El sistema deberá proporcionar un escenario abierto explorable por el jugador
Versión	V1
Información adicional	
Dependencias	FRQ-001
Prioridad	Alta

Tabla 2.23 - FRQ-022

ID	FRQ-023 Enemigos
Descripción	El sistema deberá generar enemigos que persigan al jugador
Versión	V1
Información adicional	
Dependencias	
Prioridad	Alta

Tabla 2.24 - FRQ-023

ID	FRQ-024 Misiones
Descripción	El sistema deberá hacer al jugador resolver misiones para avanzar en el juego

Versión	V1
Información adicional	
Dependencias	
Prioridad	Alta

Tabla 2.25 - FRQ-024

ID	FRQ-025 Personajes animados
Descripción	El sistema deberá animar los personajes según su situación
Versión	V1
Información adicional	
Dependencias	
Prioridad	Alta

Tabla 2.26 - FRQ-025

ID	NFRQ-001 Rendimiento
Descripción	El sistema deberá tener un rendimiento adecuado para asegurar la correcta inmersión del jugador en el entorno virtual
Versión	V1
Información adicional	
Dependencias	
Prioridad	Alta

Tabla 2.27 - NFRQ-001

ID	NFRQ-002 Portabilidad
Descripción	El sistema deberá ser ejecutable en cualquier PC que cumpla con los requerimientos mínimos
Versión	V1
Información adicional	Entre los requerimientos mínimos están la necesidad de tener instalado un software de RV o la potencia de la tarjeta gráfica
Dependencias	
Prioridad	Alta

Tabla 2.28 - NFRQ-002

2.1.2 Casos de uso

En este apartado se describe de forma general el comportamiento del sistema y del actor que hará uso de este, que en este caso solo será uno, el propio jugador. En la *ilustración 2.1* se puede ver un diagrama de casos de uso del sistema.

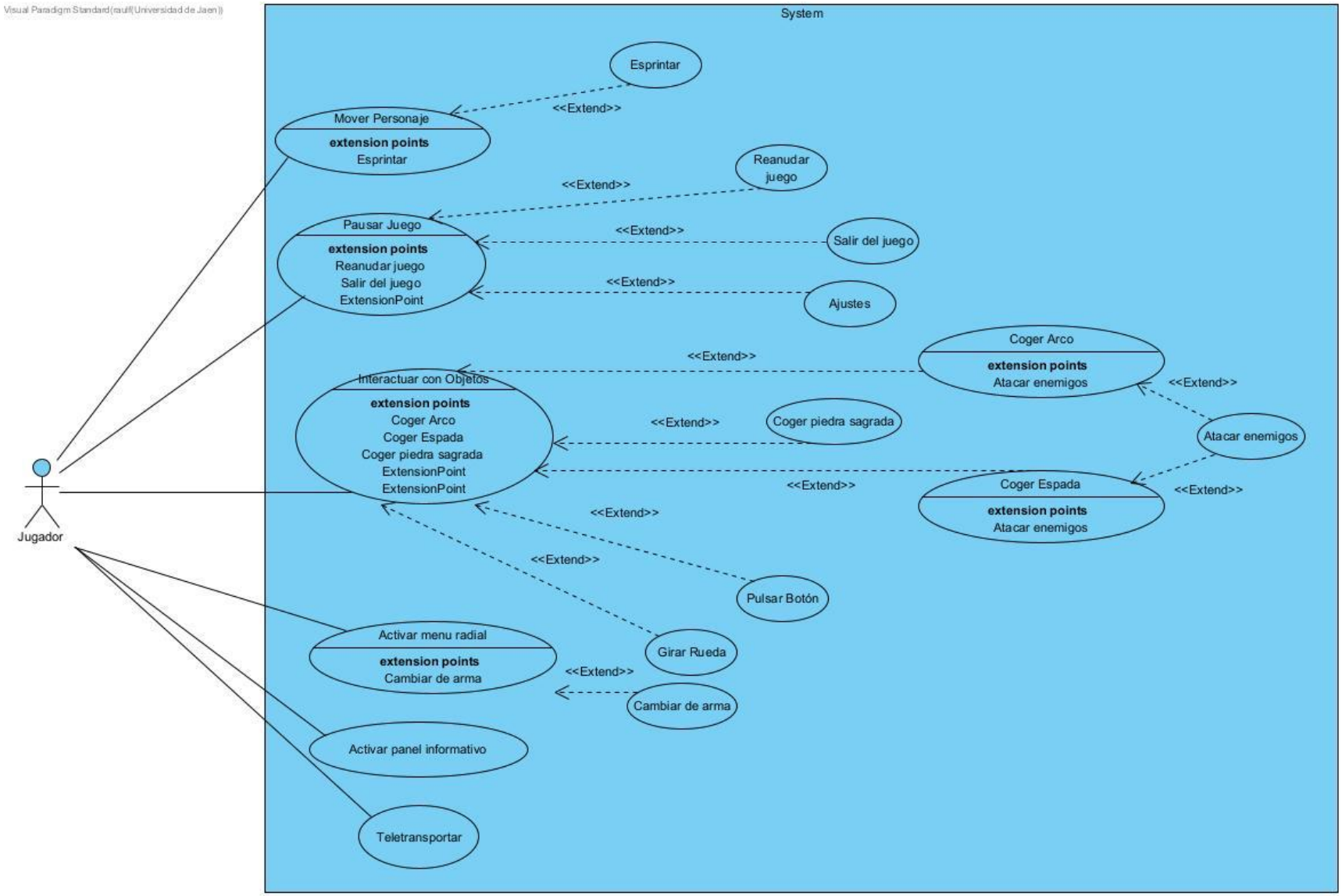


Ilustración 2.1 - Diagrama de casos de uso

2.1.3 Especificación de casos de uso

En este apartado se especifican cada uno de los casos de uso identificados en el sistema. Se utilizará la letra J para referirse al jugador y la letra S para referirse al sistema.

ID y Nombre:	CU-001 Mover Personaje
Actor	Jugador
Descripción	Mueve el personaje principal en una dirección
Disparador	El jugador desliza su dedo por el touchpad
Dependencias	
Precondiciones	Se debe estar en el juego
Escenario principal	1. El J toca el touchpad 2. El S mueve el personaje según la posición del dedo del J en el touchpad
Escenario Alternativo	
Postcondiciones	El personaje se mueve
Excepciones	
Prioridad	Alta
Otra información	

Tabla 2.29 - CU-001 Mover personaje

ID y Nombre:	CU-002 Esprintar
Actor	Jugador
Descripción	Incrementa la velocidad del personaje
Disparador	El jugador pulsa el touchpad izquierdo
Dependencias	
Precondiciones	El personaje debe estar moviéndose
Escenario principal	1. El J toca el touchpad 2. El S incrementa la velocidad del personaje
Escenario Alternativo	
Postcondiciones	El personaje esprinta
Excepciones	
Prioridad	Media
Otra información	

Tabla 2.30 - CU-002 Esprintar

ID y Nombre:	CU-003 Pausar Juego
Actor	Jugador
Descripción	Muestra el menú de pausa
Disparador	El jugador pulsa el botón menú del controlador izquierdo o dice la palabra “Menú”
Dependencias	
Precondiciones	Se debe estar en el juego
Escenario principal	1. El J pulsa el botón 2. El S activa el panel de menú 3. El S activa rayos láser provenientes de las manos del personaje para permitir al J interactuar con el UI
Escenario Alternativo	
Postcondiciones	El menú de pausa de activa
Excepciones	
Prioridad	Alta
Otra información	

Tabla 2.31 - CU-003 Pausar juego

ID y Nombre:	CU-004 Reanudar juego
Actor	Jugador
Descripción	Reanuda la partida
Disparador	El jugador selecciona la opción de Reanudar o dice la palabra “Reanudar”
Dependencias	CU-003
Precondiciones	Debe estar activado el menú de pausa
Escenario principal	1. El J selecciona Reanudar o dice la palabra “Reanudar” 2. El S desactiva el panel de menú de pausa 3. El S desactiva los rayos láser
Escenario Alternativo	
Postcondiciones	
Excepciones	
Prioridad	Alta
Otra información	

Tabla 2.32 - CU-004 Reanudar juego

ID y Nombre:	CU-005 Salir del juego
Actor	Jugador
Descripción	Cierra el juego
Disparador	El jugador selecciona la opción de Salir o dice la palabra “Salir”
Dependencias	CU-003
Precondiciones	Debe estar activado el menú de pausa
Escenario principal	1. El J selecciona Salir o dice la palabra “Salir” 2. El S cierra la aplicación
Escenario Alternativo	
Postcondiciones	Se cierra el juego
Excepciones	
Prioridad	Alta
Otra información	

Tabla 2.33 - CU-005 Salir del juego

ID y Nombre:	CU-006 Ajustes
Actor	Jugador
Descripción	Muestra el menú de ajustes
Disparador	El jugador selecciona la opción de Ajustes o dice la palabra “Ajustes”
Dependencias	CU-003
Precondiciones	Debe estar activado el menú de pausa
Escenario principal	1. El J selecciona Ajustes o dice la palabra “Ajustes” 2. El S muestra las opciones ajustables
Escenario Alternativo	
Postcondiciones	
Excepciones	
Prioridad	Alta
Otra información	

Tabla 2.34 - CU-006 Ajustes

ID y Nombre:	CU-007 Interactuar con objeto
Actor	Jugador

Descripción	Activa algún evento
Disparador	El jugador acerca su mano a un objeto interactivo y pulsa el gatillo
Dependencias	
Precondiciones	Se debe estar en el juego
Escenario principal	1. El J se acerca a un objeto y pulsa el gatillo 2. El S acciona un evento relacionado con el objeto
Escenario Alternativo	
Postcondiciones	
Excepciones	
Prioridad	Alta
Otra información	

Tabla 2.35 - CU-007 Interactuar con objeto

ID y Nombre:	**CU-008 Coger Arco**
Actor	Jugador
Descripción	Equipa al jugador con un arco y flechas
Disparador	El jugador acerca su mano al arco y pulsa el gatillo
Dependencias	CU-007
Precondiciones	Se debe estar en el juego
Escenario principal	1. El J acerca su mano al arco y pulsa el gatillo 2. El S enlaza las manos del personaje con el arco y la flecha
Escenario Alternativo	
Postcondiciones	Se enlazan las manos con el arco y las flechas
Excepciones	
Prioridad	Alta
Otra información	

Tabla 2.36 - CU-008 Coger arco

ID y Nombre:	**CU-009 Coger Espada**
Actor	Jugador
Descripción	Equipa al jugador con una espada
Disparador	El jugador acerca su mano a la espada y pulsa el gatillo

Dependencias	CU-007
Precondiciones	Se debe estar en el juego
Escenario principal	1. El J acerca su mano a la espada y pulsa el gatillo 2. El S enlaza la mano del personaje con la espada
Escenario Alternativo	
Postcondiciones	Se enlaza la mano con la espada
Excepciones	
Prioridad	Alta
Otra información	

Tabla 2.37 - CU-009 Coger espada

ID y Nombre:	CU-011 Atacar enemigo
Actor	Jugador
Descripción	Realiza daño a un enemigo
Disparador	El jugador dispara una flecha contra el enemigo o acerca su espada a él
Dependencias	CU-007
Precondiciones	Se debe estar en el juego y tener equipada un arma
Escenario principal	1. El J dispara una flecha hacia el enemigo o lo golpea con la espada 2. El S reduce la vida restante del enemigo 3. El S genera un sonido que simula un golpe 4. El S genera unas partículas que simulan daño
Escenario Alternativo	
Postcondiciones	El enemigo pierde vida
Excepciones	
Prioridad	Alta
Otra información	

Tabla 2.38 - CU-011 Atacar enemigo

ID y Nombre:	CU-012 Coger piedra sagrada
Actor	Jugador
Descripción	Coge una piedra sagrada
Disparador	El jugador acerca su mano a este objeto
Dependencias	CU-007

Precondiciones	Se debe estar en el juego y completar alguna misión
Escenario principal	1. El J acerca su mano al objeto 2. El S aumenta el contador de piedras mágicas 3. El S activa la imagen de la piedra en el panel de información 4. El S genera un sonido 5. El S genera unas partículas 6. El S destruye el objeto
Escenario Alternativo	
Postcondiciones	La piedra es recogida
Excepciones	
Prioridad	Alta
Otra información	

Tabla 2.39 - CU-012 Coger piedra sagrada

ID y Nombre:	**CU-013 Pulsar botón**
Actor	Jugador
Descripción	Pulsa un botón
Disparador	El jugador acerca su mano a este objeto
Dependencias	CU-007
Precondiciones	Se debe estar en el juego
Escenario principal	1. El J acerca su mano al objeto 2. El S activa una animación que abre una compuerta
Escenario Alternativo	
Postcondiciones	Se abre la compuerta final
Excepciones	
Prioridad	Alta
Otra información	

Tabla 2.40 - CU-013 Pulsar botón

ID y Nombre:	**CU-014 Girar rueda**
Actor	Jugador
Descripción	Gira una rueda que abre un portón
Disparador	El jugador acerca su mano a este objeto y pulsa el gatillo

Dependencias	CU-007
Precondiciones	Se debe estar en el juego
Escenario principal	1. El J acerca su mano al objeto y pulsa el gatillo 2. El J gira la rueda en el sentido correcto 3. El S realiza la animación de abrir la compuerta
Escenario Alternativo	
Postcondiciones	Se abre la compuerta
Excepciones	
Prioridad	Alta
Otra información	

Tabla 2.41 - CU-014 Girar rueda

ID y Nombre:	CU-014 Activar menú radial
Actor	Jugador
Descripción	Muestra el menú radial
Disparador	El J mantiene pulsado el touchpad del controlador derecho
Dependencias	
Precondiciones	Se debe estar en el juego
Escenario principal	1. El J pulsa el touchpad derecho 2. El S activa el menú radial
Escenario Alternativo	
Postcondiciones	
Excepciones	
Prioridad	Alta
Otra información	

Tabla 2.42 - CU-014 Activar menú radial

ID y Nombre:	CU-015 Cambiar de arma
Actor	Jugador
Descripción	Cambia el arma del jugador
Disparador	El J desliza el dedo por el touchpad derecho y deja de presionar el botón
Dependencias	CU-014

Precondiciones	Se debe estar en el juego y con el menú radial activo
Escenario principal	1. El J desliza su dedo para seleccionar el arma 2. El S equipa al J con el arma seleccionada
Escenario Alternativo	
Postcondiciones	El jugador es equipado con el arma seleccionada
Excepciones	
Prioridad	Alta
Otra información	

Tabla 2.43 - CU-015 Cambiar de arma

ID y Nombre:	CU-016 Activar panel informativo
Actor	Jugador
Descripción	Muestra un panel con información útil (vida, tiempo de juego, contador de piedras mágicas y un mapa)
Disparador	El J pulsa el botón menú del controlador derecho
Dependencias	
Precondiciones	Se debe estar en el juego
Escenario principal	1. El J pulsa el botón menú del controlador derecho 2. El S activa el panel informativo 3. El J deja de pulsar el botón 4. El S desactiva el panel informativo
Escenario Alternativo	
Postcondiciones	
Excepciones	
Prioridad	Alta
Otra información	

Tabla 2.44 - CU-016 Activar panel informativo

ID y Nombre:	CU-016 Teletransportar
Actor	Jugador
Descripción	Teletransporta al jugador a una zona habilitada
Disparador	El J pulsa el botón empuñadura y apunta hacia una zona habilitada de teletransporte

Dependencias	
Precondiciones	Se debe estar en el juego
Escenario principal	1. El J mantiene pulsado el botón empuñadura 2. El S muestra un rayo de teletransporte 3. El J apunta hacia un punto de teletransporte 4. El J suelta el botón 5. El S teletransporta al personaje a la zona
Escenario Alternativo	
Postcondiciones	El personaje es teletransportado a la zona elegida
Excepciones	
Prioridad	Alta
Otra información	

Tabla 2.45 - CU-016 Teletransportar

2.2 Historia

Una vez presentada la especificación del sistema, es fundamental introducir al lector el contexto donde se desarrollará el videojuego. La historia se desarrolla en una misteriosa isla situada en un lugar desconocido. Esta isla está habitada por seres místicos y en ella se diferencian 4 biomas. Una zona rocosa habitada por duendes, un bosque repleto de vegetación en el que se encuentra el templo sagrado, una zona de costa habitada por pájaros y una zona volcánica habitada por criaturas de lava. El personaje principal se encuentra en el corazón de la isla y su misión es la de conseguir entrar en el templo sagrado. Deberá explorar la isla en busca de las tres piedras sagradas, necesarias para entrar en el templo.

2.3 Personajes

En este apartado se describen las características principales de los personajes del videojuego.

2.3.1 NPC

Los **Non-playable-Character o NPC** son aquellos personajes que son controlados por el juego y no por el jugador. Estos personajes pueden ser bastante importantes ya que en ocasiones ayudan al jugador a avanzar en la historia. En este

apartado se presentan de forma breve los personajes no jugables que pueden encontrarse a lo largo de la experiencia.

ID	Características
NPC-01	**Nombre**: Fenris el elfo **Descripción**: Es un elfo nativo de la isla. Nunca ha visitado otro lugar. Es uno de los personajes que ayudará al protagonista. Este explicará que hay que hacer para entrar en el templo.
NPC-02	**Nombre:** Mono **Descripción:** Es un mono que habita en la isla. Tiene en su poder una de las piedras sagradas. Encargará al protagonista una misión cuya recompensa será la piedra sagrada azul.
NPC-03	**Nombre:** Murciélago **Descripción**: Es otro de los personajes que ayudarán al jugador a avanzar en el juego. Este advertirá al jugador sobre los peligros de la zona de la lava.
NPC-04	**Nombre:** Murciélago ciego Descripción: Este ser es el dueño de otra de las piedras sagradas. Se encuentra al final de la zona de lava y entregará al jugador la piedra sagrada de la zona si este cumple una misión.
NPC-05	**Nombre:** Bob el minero **Descripción:** Bob es un tranquilo minero de pocas ambiciones que habita en su mina. Este trata de buscar todo el mineral existente para venderlo fuera de la isla. Encargará al jugador la misión de ayudarle y si la completa le entregará otra de las piedras sagradas.
NPC-06	**Nombre:** Guardián de la segunda puerta **Descripción:** Este personaje se encuentra dentro del templo custodiando una de las compuertas finales. Propondrá un acertijo al jugador que tendrá que resolver para poder continuar

Tabla 2.46 - NPCs

2.3.2 Enemigos

Los enemigos son aquellos personajes controlados por el juego cuyo objetivo es el de acabar con el personaje principal, en este caso con el jugador. Son muy importantes ya que determinan la dificultad de la experiencia de juego. En este

apartado se presentan de forma breve los enemigos que pueden encontrarse a lo largo de la experiencia.

ID	Características
EN-01	**Nombre**: Stone Monster **Puntos de Vida**: 1 **Rango de detección**: 20 **Descripción**: Monstruo volador que habita en la zona volcánica. Lanzará bolas de fuego al jugador cuando este esté a menos de cierta distancia
EN-02	**Nombre**: Duende **Puntos de Vida**: 3 **Rango de detección**: 15 **Descripción**: Duende que habita por la zona rocosa. Este atacará al jugador con su porra cuando esté a menos de cierta distancia
EN-03	**Nombre**: Eagle **Puntos de Vida**: 1 **Descripción**: Pájaro que se encuentra en la zona de la playa. No atacará al jugador.

Tabla 2.47 - Enemigos

2.3.3 Objetos

Estos son aquellos objetos con los que el jugador puede interactuar en el juego.

ID	Características
OBJ-01	**Nombre**: Piedra sagrada **Descripción**: Existen tres piedras en toda la isla. Cada una de un color. El jugador puede interactuar con ellas y son necesarias para entrar en el templo.
OBJ-02	**Nombre**: Mineral **Descripción**: Mineral de color púrpura que abunda en la mina.
OBJ-03	**Nombre**: Arco

	Descripción: Arma principal del juego. Permitirá al jugador disparar flechas. Se encuentra en el camino hacia la playa.
OBJ-04	**Nombre:** Espada **Descripción:** Arma secundaria del juego. Permitirá al jugador realizar ataques cuerpo a cuerpo. Se encuentra en el camino hacia la playa.
OBJ-05	**Nombre:** Ojo **Descripción**: Es el ojo perdido de uno de los murciélagos. El jugador puede interactuar con este objeto.
OBJ-06	**Nombre:** Rueda **Descripción:** Es una rueda cuyo giro abre la primera puerta del templo. El jugador deberá girarla para pasar a la siguiente sala.
OBJ-07	**Nombre:** Esferas mágicas **Descripción:** Esferas situadas en la última zona del templo. Estas desbloquearan algo al ser activadas.
OBJ-08	**Nombre:** Botón **Descripción:** Botón situado en la última zona del templo. Se desbloqueará al activar las tres esferas. El jugador deberá pulsarlo para desbloquear la puerta final.
OBJ-09	**Nombre:** Pico **Descripción:** Pico situado en la mina. El jugador lo puede utilizar para minar los cristales.
OBJ-10	**Nombre:** Lámpara **Descripción:** Se sitúa en la mina y sirve para alumbrar.
OBJ-11	**Nombre:** Piedra **Descripción:** Hay muchas a lo largo del mapa y el jugador podrá interactuar con ellas

Tabla 2.48 - Objetos

2.4 Escenarios

Los escenarios son parte fundamental de un videojuego. Estos definen la estética y ambientación de este. A continuación, se describen los **escenarios principales** y se presenta un primer esbozo de estos, que posteriormente servirá para su creación en la parte de implementación.

2.4.1 Isla

La isla es el **escenario principal** del videojuego. Es donde sucede la mayor parte de la historia. El jugador tendrá libertad para moverse por ella y descubrirla mientras juega.

En la isla se diferencian claramente **cuatro zonas**. Cada una de estas zonas presentará características diferentes a través de las cuales el jugador podrá diferenciarlas con facilidad. La *ilustración 2.2* muestra un primer boceto de la forma de la isla.

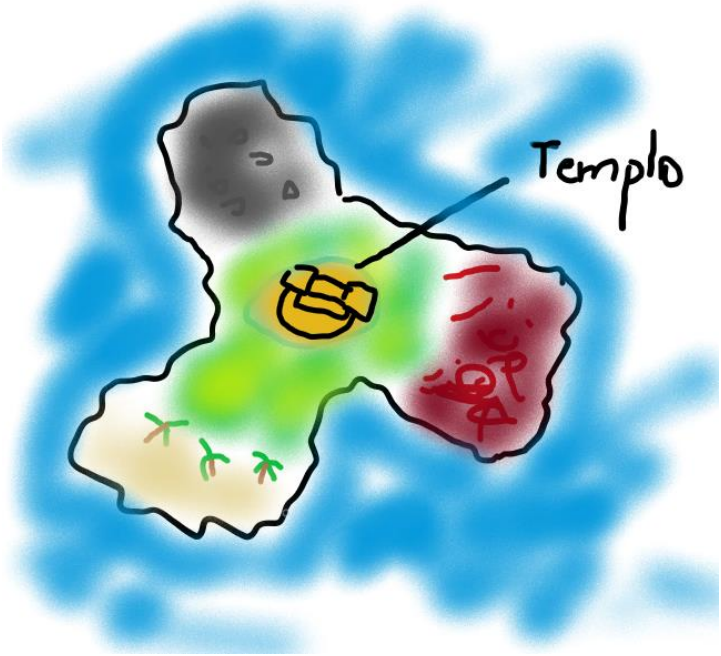


Ilustración 2.2 - Boceto inicial de la isla

La **zona del bosque** es la zona inicial del juego. Aquí se encuentra el elfo, el primer NPC con el que interactuará el jugador. También en esta zona se realizará un tutorial interactivo. El bosque actúa de nexo entre todas las zonas por lo que el jugador tendrá que atravesarlo en varias ocasiones. Se trata de un bosque con diferentes tipos de árboles y arbustos que alberga un templo sagrado en su corazón. El templo está incrustado en una gran montaña y rodeado de ruinas con inscripciones rúnicas. De este solo se ve de este su entrada principal. La entrada posee un conjuro que no

dejará pasar a nadie que no posea las tres piedras sagradas. En la *ilustración 2.3* se presenta un boceto de la entrada del templo

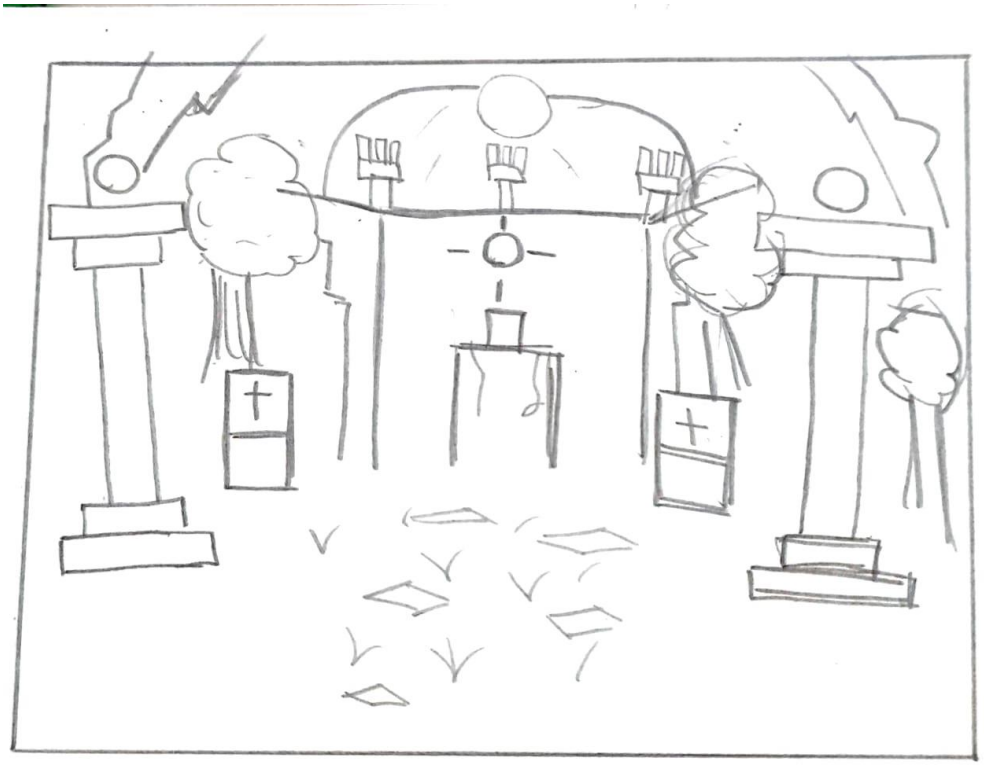


Ilustración 2.3 - Boceto de la entrada del templo

La **zona de lava** es una zona volcánica con un gran lago de lava en su interior. El paisaje se caracteriza por su poca flora y abundancia de estructuras rocosas y volcánicas. Esta zona está repleta de enemigos que el jugador podrá eliminar o evitar. En la *ilustración 2.4* se muestra un boceto de esta zona

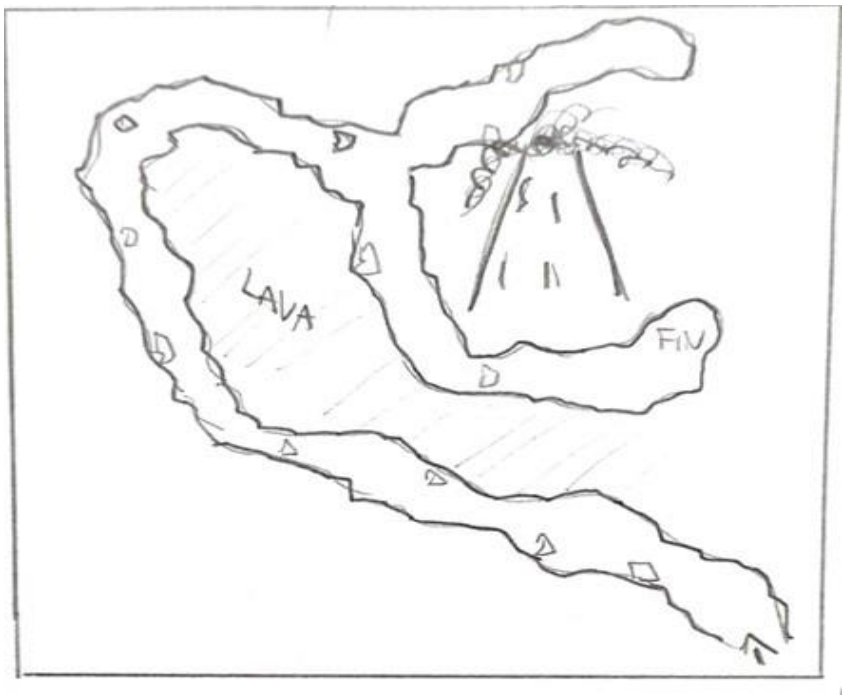


Ilustración 2.4 – Boceto de la zona de lava

La **zona de roca** es una zona montañosa y el único camino para acceder a la mina. Cuenta con un paso de madera que permitirá al jugador cruzar una estructura rocosa. En esta zona se encuentran algunos enemigos que atacarán al jugador. El paisaje se compone principalmente de rocas. Al final del camino se encuentra el acceso una mina como se puede observar en la *ilustración 2.5*.

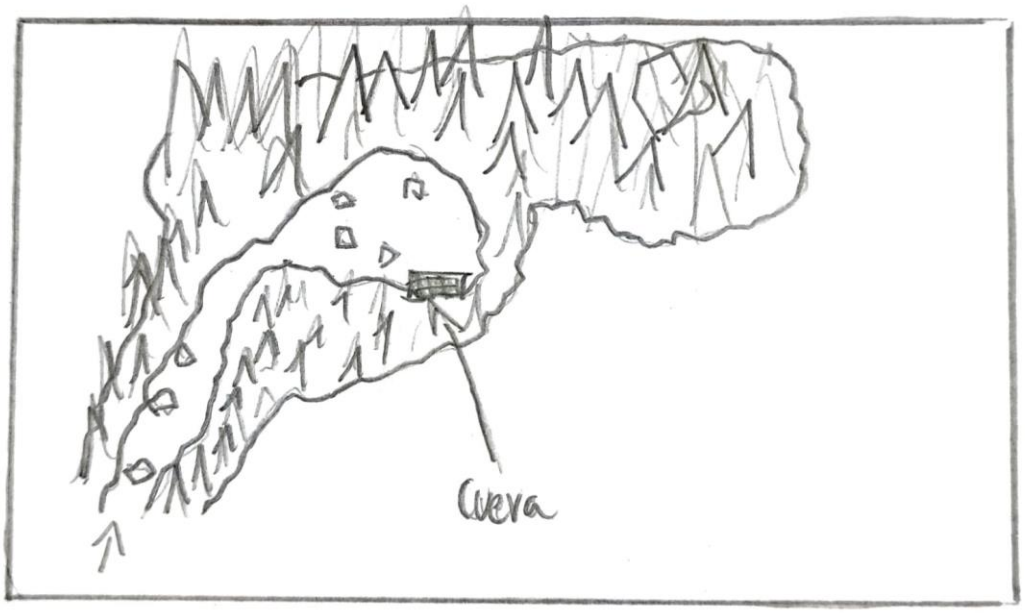


Ilustración 2.5 – Boceto de la zona rocosa

La **zona de playa** presenta un paisaje compuesto por una playa repleta de palmeras y pequeñas piedras. Allí se encuentra uno de los NPC, que asignará una misión al jugador. En la *ilustración 2.6* se puede observar un boceto de esta zona.

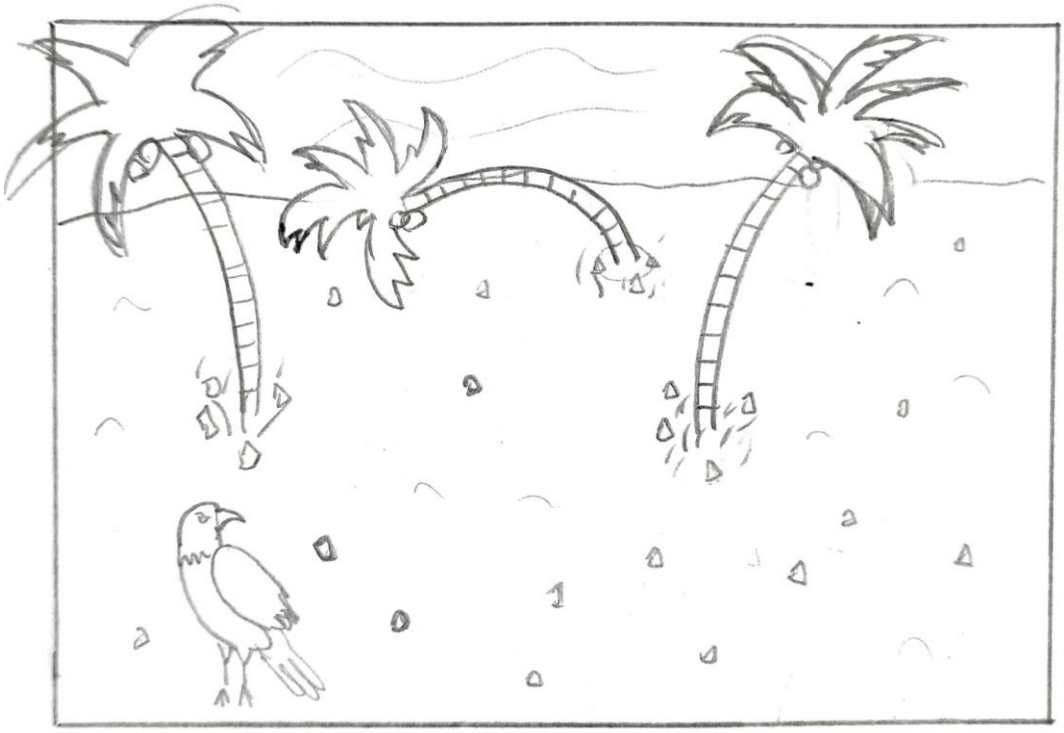


Ilustración 2.6 - Boceto de la zona de la playa

2.4.2 Mina

Al final de la zona rocosa se encuentra la entrada a la mina. Esta mina presenta un claustrofóbico camino soportado por vigas de madera que desemboca en una cueva repleta de minerales. Dentro se encuentra un NPC, que encomendará una misión al jugador (*ilustraciones 2.7 y 2.8*).

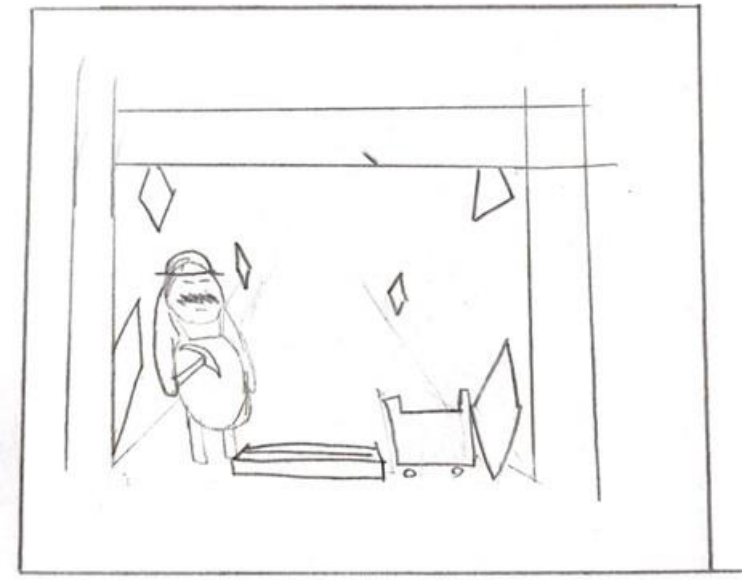


Ilustración 2.7 – Boceto de la entrada de la mina

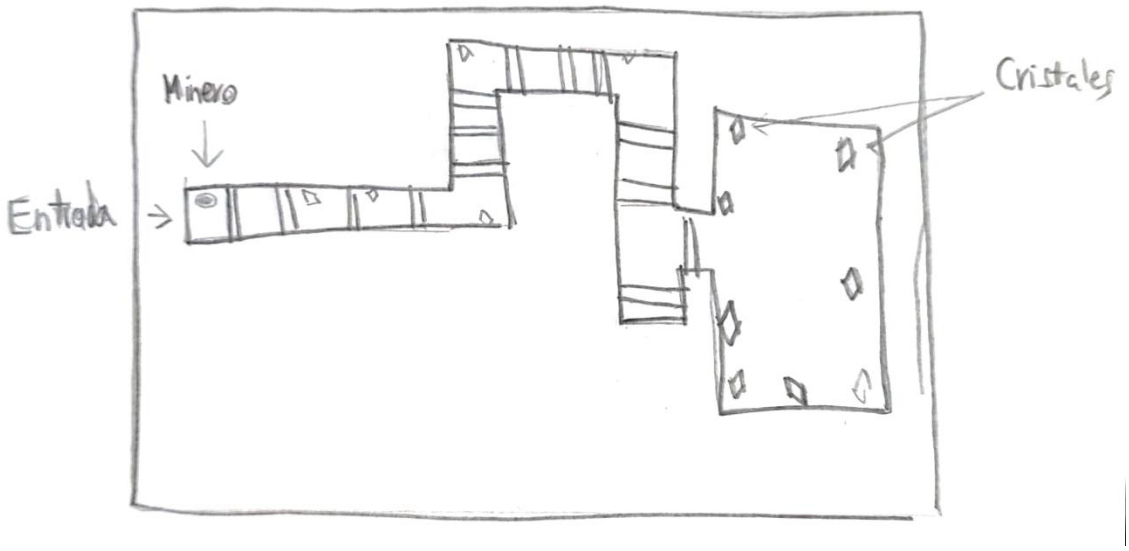


Ilustración 2.8 - Boceto de la mina

2.4.3 Templo

El interior del templo presenta una serie de puzles que el jugador tiene que resolver para avanzar. Este se divide en tres salas principales. La primera es la sala inicial a la que el jugador llega a parar cuando entra. En la segunda sala el jugador deberá girar una rueda para levantar el portón que impide el paso. Entre la segunda y tercera sala hay un pasillo en el que se encuentra un NPC, este propondrá un acertijo que el jugador deberá adivinar para pasar a la tercera sala. En la tercera sala el jugador tiene

que utilizar el arco y accionar un mecanismo para abrir la puerta final. En la *ilustración 2.9* se puede observar un plano del interior del templo.

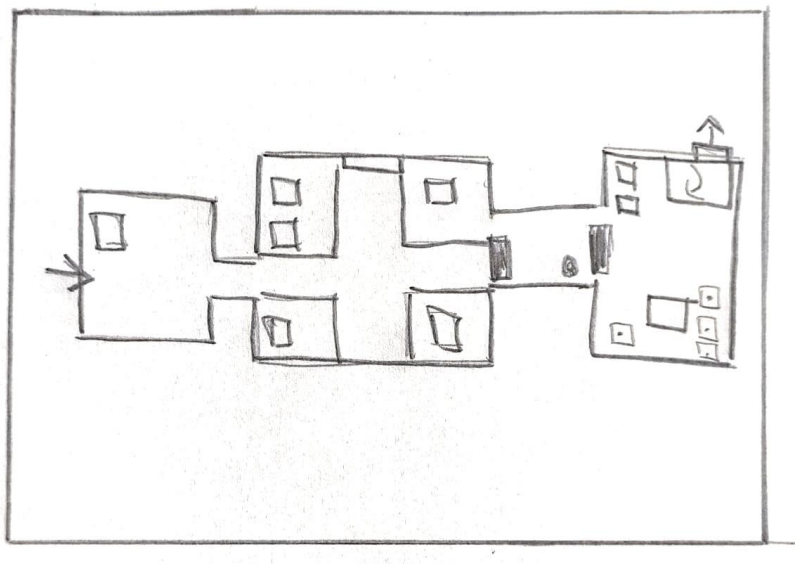


Ilustración 2.9 – Boceto del interior del templo

.

2.5 Interfaz de usuario

El diseño de la interfaz de usuario (UI) tiene especial importancia al tratarse de un entorno en VR. Esta además de ser intuitiva y fácil de utilizar, no tiene que generar molestias en la vista del usuario simplificando al máximo la interacción con botones. Es conveniente utilizar paneles interactivos, no muy intrusivos en el campo de visión del usuario, con los se pueda interactuar a lo largo de la partida. A continuación, se presentan los principales elementos de la interfaz diseñados como *wireframes*.

2.5.1 Menú de pausa

Este panel debe incluir las opciones de reanudar la partida, ir a la ventana de ajustes o salir del juego.

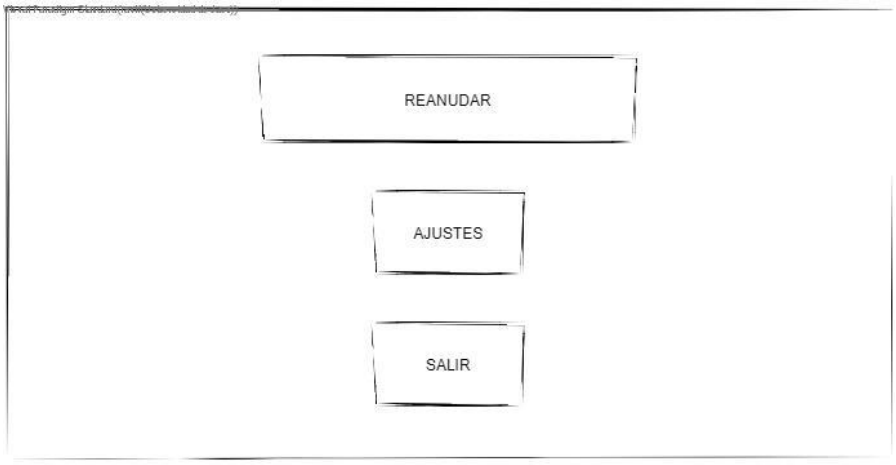


Ilustración 2.10 - Wireframe del panel de pausa

2.5.2 Tutorial

Este panel aparecerá al comenzar el juego y dará la posibilidad al jugador de realizar un tutorial para conocer los controles básicos. El jugador podrá hacerlo o no.

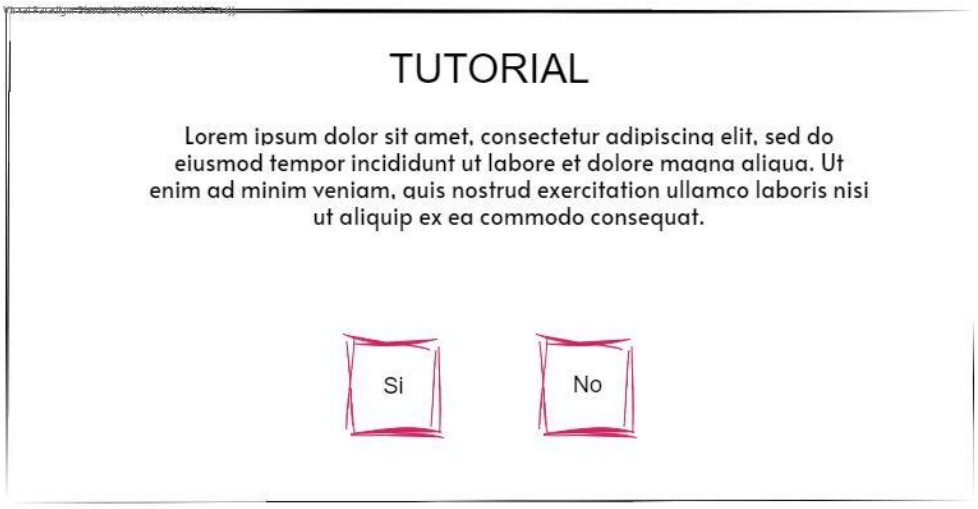


Ilustración 2.11 - Wireframe del comienzo del tutorial

Una vez en el tutorial se utilizará el siguiente panel como plantilla.

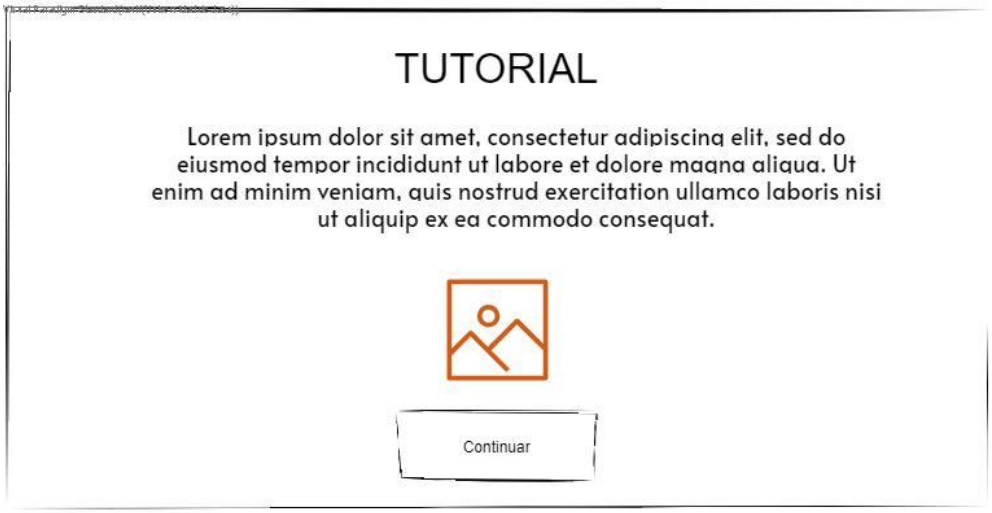


Ilustración 2.12 - Wireframe del panel tutorial

2.5.3 Menú radial

El jugador podrá cambiar de arma accionando un menú radial con algún botón. El menú radial deberá tener una forma parecida a la del siguiente boceto.

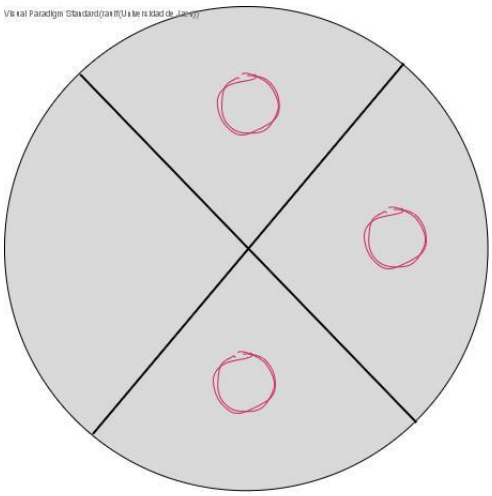


Ilustración 2.13 - Wireframe del menú radial para cambiar de arma

2.5.4 Panel informativo

El jugador tendrá la posibilidad de visualizar información útil durante la partida, como el número de piedras conseguidas, el nivel de vida restante o el tiempo de juego. Además, contará con un mini mapa que le ayudará a ubicarse en la isla.

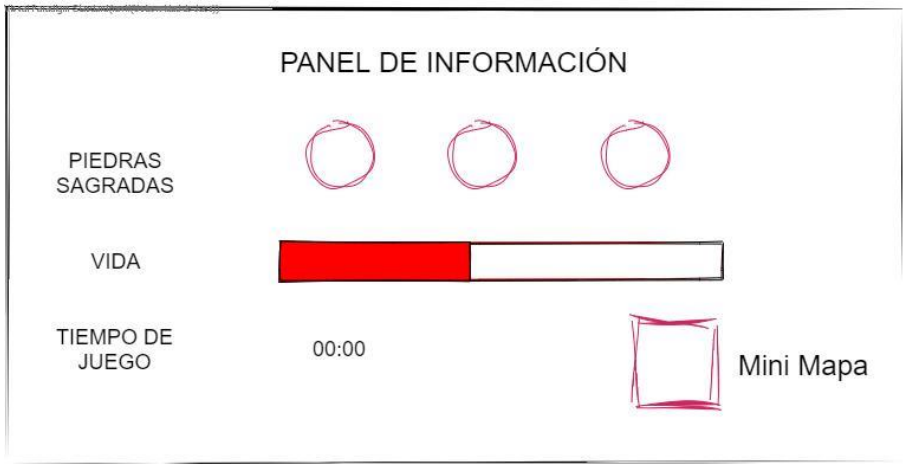


Ilustración 2.14 - Wireframe del panel informativo

2.5.5 Panel de muerte

Tras morir el jugador visualizará algunos datos y tendrá la posibilidad de pulsar un botón de reinicio.

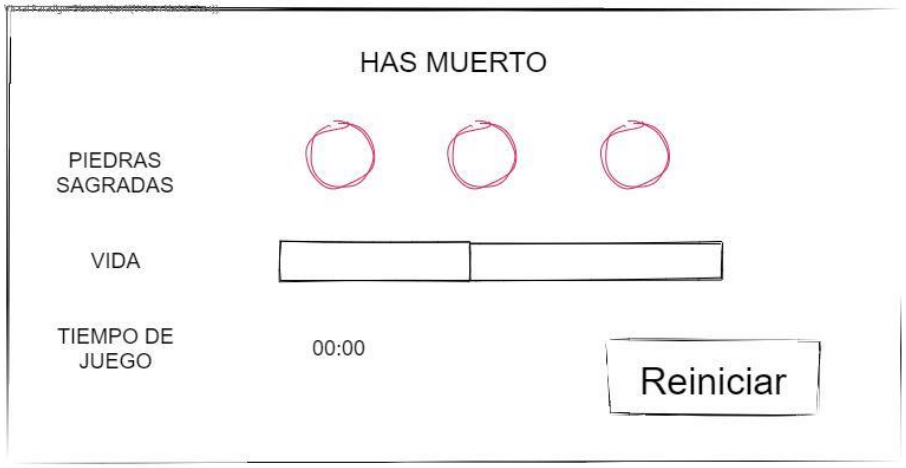


Ilustración 2.15 - Wireframe del panel de muerte

2.5.6 Panel de ajustes

El panel de los ajustes permitirá al jugador subir o bajar el volumen del juego

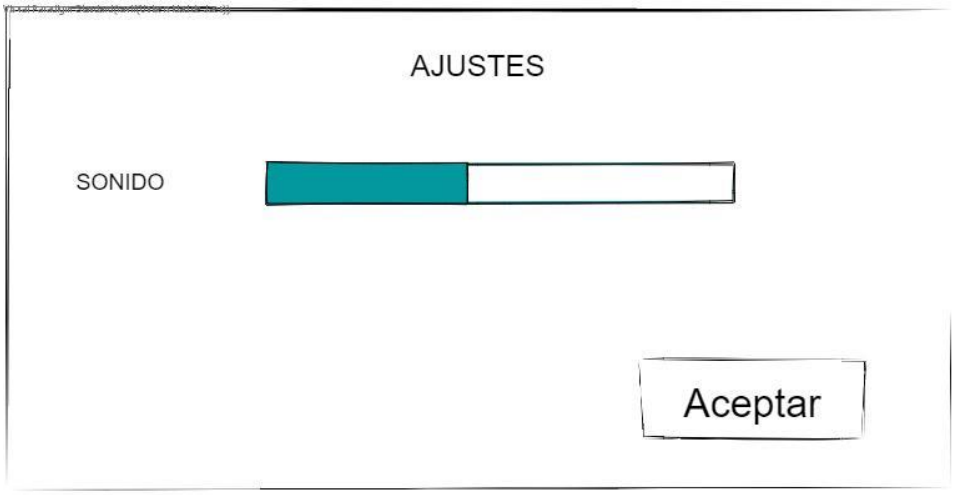


Ilustración 2.16 - Boceto del panel de ajustes

2.5.7 Panel de carga

El panel de carga se introducirá cuando se produzca un cambio de escena.

Ilustración 2.17 - Wireframe del panel de carga

2.6 Estudio de alternativas y viabilidad

Un proyecto de estas características requiere un estudio fundamentado para elegir adecuadamente las tecnologías más apropiadas y con las que obtener los mejores resultados. En esta sección se presentan las diferentes alternativas tecnológicas que se han tenido en cuenta para el desarrollo del proyecto y la justificación de aquellas que han sido seleccionadas.

Con el objetivo de la creación del videojuego se ha realizado una comparativa entre **distintos motores gráficos que permiten el desarrollo de videojuegos en RV**.

Motor gráfico	Características
Unity 3D	- Lenguaje C# - Multiplataforma - Curva de aprendizaje baja - Mucha documentación - Gratis
Unreal Engine 4	- Lenguaje C++ o Blueprint - Mayor nivel de calidad - Curva de aprendizaje alta - Gratis

Tabla 2.49 - Comparación motores gráficos

Tras la comparación de estos dos motores se decidió escoger **Unity** ya que había sido utilizado anteriormente y el hecho de que cuente con una comunidad tan amplia es una gran ventaja. Además, Unreal Engine ofrece un mayor nivel de calidad y por tanto requiere un equipo potente, lo cual era un inconveniente dado que el equipo utilizado para el desarrollo del proyecto no lo era.

Para la selección de las **gafas de VR** se ha realizado una **comparativa de las especificaciones de las distintas gafas del mercado**. Se han descartado directamente aquellas gafas que no son compatibles con un PC como por ejemplo las gafas de Sony PlayStation VR.

Gafas RV	Características
HTC Vive	- Frecuencia de refresco de 90Hz - Resolución de 1080x1200 por ojo - Campo de visión de 100º - Mucha precisión - Cámara frontal de seguridad - No autónomas - 899€

Oculus Rift CV1	- Resolución de 1080 x 1200 píxeles por ojo - Frecuencia de refresco de 90Hz - Campo de visión de 110º - 478€
Oculus Quest 2	- Resolución de 1920x1832 píxeles por ojo - Frecuencia de refresco de 90Hz - Autónomas - Campo de visión de 90º - 449€
HP Reverb G2	- Resolución de 2160 x 2160 píxeles por ojo - Frecuencia de refresco de 90Hz - Campo de visión de 114º - No autónomas - No aptas para equipos pequeños - 696,17€

Tabla 2.50 - Comparación gafas VR

Tras esta comparación se decidió que la mejor opción serían las gafas **HTC Vive**, el motivo principal es que la Universidad de Jaén contaba con un ejemplar y por lo tanto no sería necesario un gasto económico. Además, al tratarse de unas gafas de hace unos años, las limitaciones técnicas eran menores y estas eran compatibles con el equipo utilizado para el desarrollo del proyecto. El resto de las gafas ofrecen una mejor calidad, pero requieren ordenadores muy potentes.

Para la incorporación de la realidad virtual en Unity se ha realizado la **comparativa entre dos SDKs que proporcionan herramientas para la creación de contenido en VR**.

SDK	Características
SteamVR Unity Plugin	- Documentación escasa - Muy buena compatibilidad con HTC Vive - Gran cantidad de recursos incluidos
Unity XR toolkit	- Buena documentación - Compatibilidad con varias gafas de VR - Escasez de recursos incluidos

Tabla 2.51 - Comparación de SDKs

Para implementar la realidad virtual se decidió utilizar el **SDK SteamVR** ya que está desarrollado exclusivamente para las gafas HTC Vive, que son las que se iban a usar en el proyecto, además de ser compatible con otros dispositivos. También contaba con un gran número de recursos de gran utilidad ya implementados.

2.7 Descripción de la solución propuesta

En esta sección se describirán con más detalle las tecnologías seleccionadas para el desarrollo del proyecto.

2.7.1 Unity

Es un motor de videojuegos multiplataforma creado por *Unity Technologies* que permite el diseño, la creación y el funcionamiento de entornos interactivos, es decir, videojuegos.

Ilustración 2.18 - Logo de Unity[7]

Unity es conocido por su baja curva de aprendizaje. Otros motores ofrecen una mayor calidad, pero en cambio su uso es mucho más complejo. Unity es muy fácil e intuitivo, lo que lo convierte en uno de los motores más usados en todo el mundo. Además, cuenta con infinidad de documentación y tutoriales en la web, así como con una gran comunidad gracias a la cual es fácil encontrar la solución a casi cualquier problema.

Unity, además de ser un motor, es un IDE, que proporciona una interfaz que da acceso al usuario a todas las herramientas para el desarrollo. Unity tiene un editor

[7] https://es.m.wikiversity.org/wiki/Archivo:Unity_Technologies_logo.svg

visual que permite a los creadores simplemente arrastrar y soltar elementos en escenas y luego manipular sus propiedades.

El lenguaje de programación usado en Unity es C#, que es un lenguaje orientado a objetos y proveniente de la familia C similar a C++, Java y JavaScript. Es intuitivo y fácil de aprender.

Este motor apenas tiene características para el modelado, más allá de algunas primitivas. Sin embargo, es compatible con las principales aplicaciones de modelado 3D como Blender o Maya por lo que los desarrolladores pueden apoyarse en estas otras herramientas. Además, Unity cuenta con una tienda de recursos (*Unity Asset Store*), una tienda virtual desde la que se puede descargar una gran cantidad de recursos tanto gratuitos como de pago (*ilustración 2.19*).

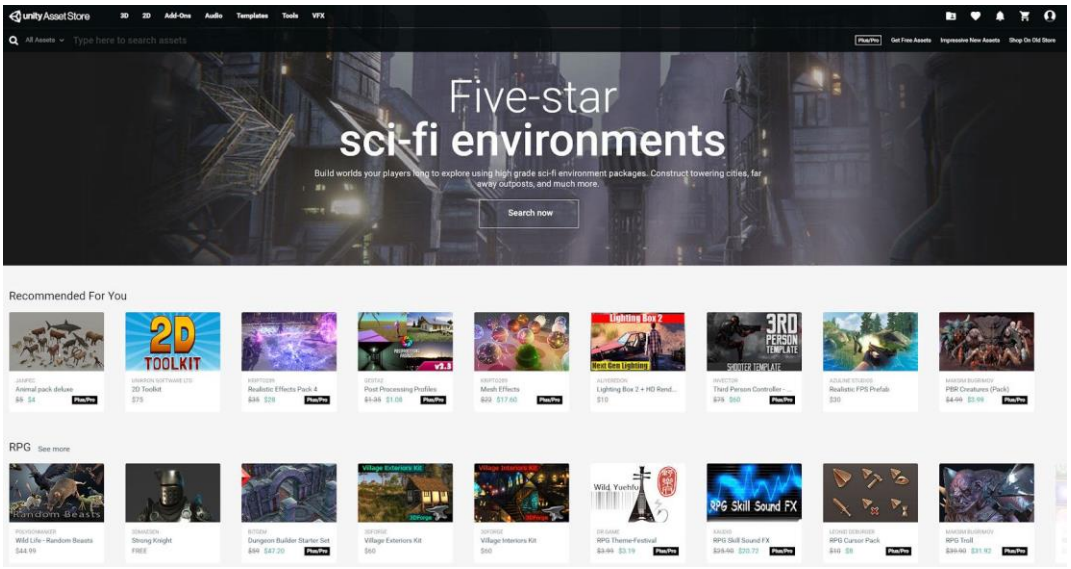


Ilustración 2.19 - Asset Store de Unity [8]

Unity se compone principalmente de escenas. A su vez, cada escena se compone de GameObjects que básicamente representan cada una de las entidades que hay en una escena de Unity y a los cuales se le pueden asignar una serie de componentes y scripts, donde se programa la lógica. En cada script se deberá programar el código que determine el comportamiento del GameObject al que está asignado. Una de las características más comunes de los scripts es que la mayoría deben heredar la clase MonoBehaviour. Esta clase es la clase principal de Unity, y se

[8] https://unity3d.com/es/quick-guide-to-unity-asset-store

encarga de que los scripts que la heredan se ejecuten según el ciclo de vida mostrado en la *ilustración 1.8.*

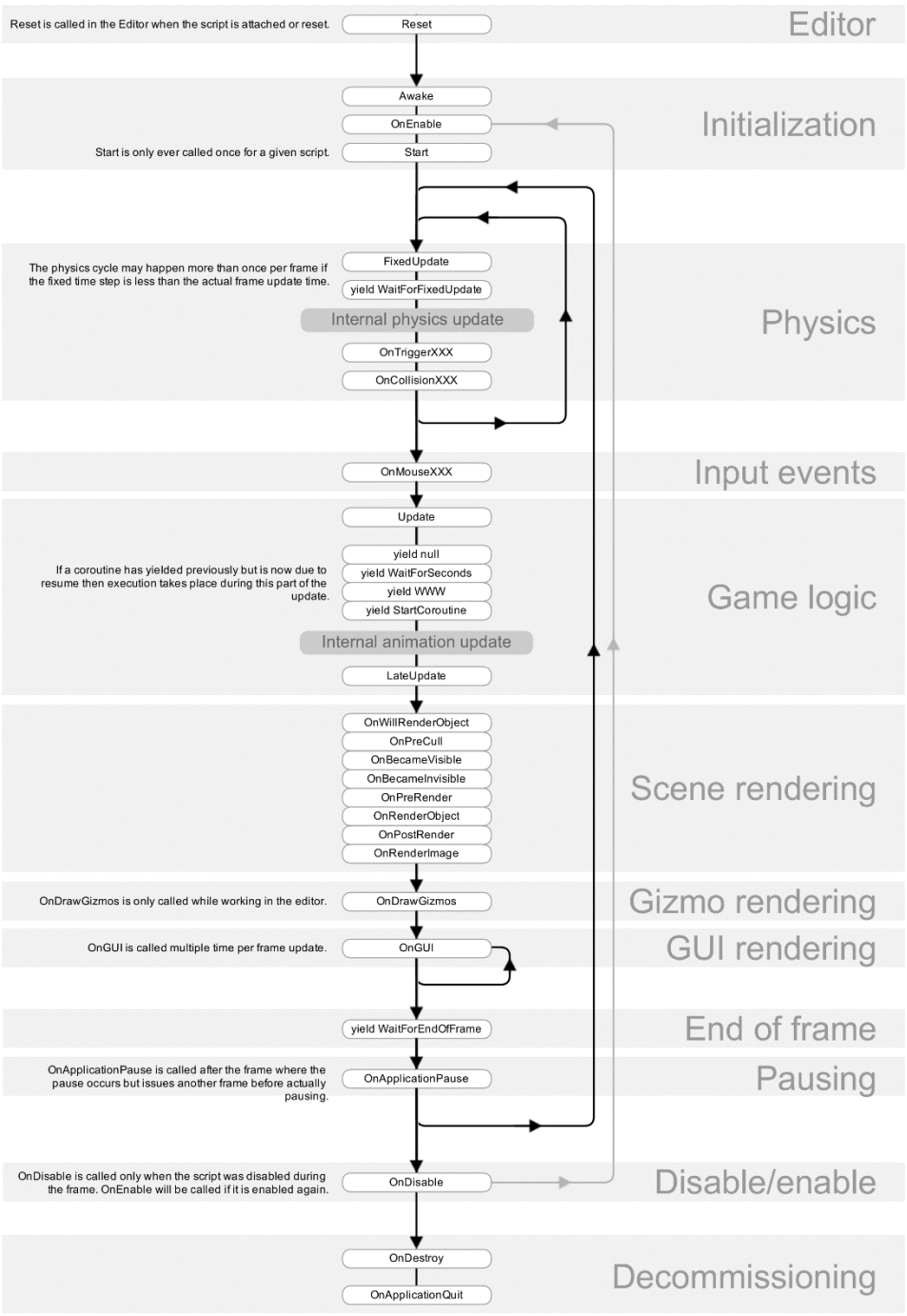


Ilustración 2.20 - Ciclo de vida de la clase MonoBehaviour [9]

[9] https://docs.unity3d.com/uploads/Main/monobehaviour_flowchart.svg

En conclusión, Unity es una de las mejores herramientas a día de hoy para desarrollar videojuegos y no solo eso, sino que también ofrece herramientas para crear otro tipo de proyectos como visualizaciones arquitectónicas, películas de animación o simulaciones.

2.7.2 SteamVR

SteamVR es una herramienta creada por Valve que permite conectar tanto HTC Vive como otros visores a un PC y que este los reconozca adecuadamente. Además, proporciona un entorno interactivo al usuario donde este puede acceder a un menú y realizar diferentes acciones como ejecutar un juego o configurar el visor VR (*ilustraciones 2.21 y 2.22*). A día de hoy SteamVR soporta los siguientes visores: HTC Vive, Valve Index, Oculus Rift, Windows Mixed Reality headsets y otros.

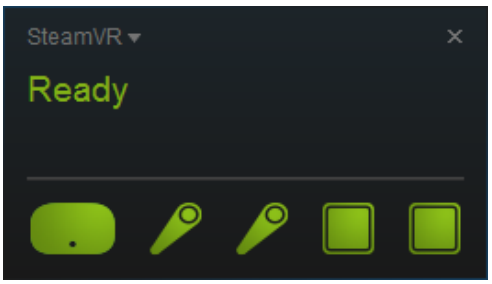


Ilustración 2.21 - Ventana para controlar SteamVR [10]

Ilustración 2.22 - Interfaz de usuario de SteamVR [11]

[10] https://hazlolinux.com/realidad-virtual/que-es-steamvr/
[11] https://venturebeat.com/arvr/steamvr-beta-adds-new-dashboard-oculus-quest-icon/

2.7.3 SteamVR Unity plugin

Este plugin mantenido por Valve permite conectar sin problemas SteamVR con Unity. De este modo los desarrolladores pueden utilizar una API a la que se pueden conectar la mayoría de visores VR. Este plugin gestiona tres aspectos importantes: Carga los modelos 3D para los controladores, maneja la entrada de los controladores, y estima la posición y el estado de las manos mientras el usuario usa los controladores. Además de hacer todo eso, incluye ejemplos de interacción para ayudar a los desarrolladores a entender cómo funciona como se observa en la *ilustración 2.23*.

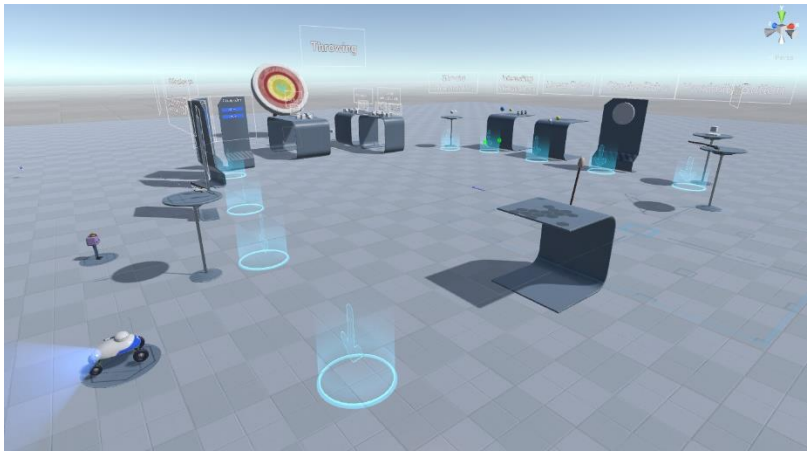


Ilustración 2.23 - Escena de ejemplo de interacciones de SteamVR

2.8 Arquitectura del sistema

2.8.1 Análisis de los requisitos

A partir de los requisitos recogidos en el apartado anterior, y de los casos de uso asociados, se ha extraído una lista provisional de clases conceptuales y atributos necesarios para el correcto funcionamiento del sistema.

Clase	Atributos	Relaciones
Jugador	• vida • input • manoderecha • manoizquierda	• Jugador-Arco • Jugador-Espada • JugadorPiedraSagrada • Jugador-Teletransporte
GameManager	• contadorPiedras	• GameManager-MenuRadial • GameManager-Jugador

		• GameManager-UIManger
Enemigo	• vida • particulasDanio • sonidoDanio • animaciones[]	
Arco	• rangoDisparo • flecha • manoAsignada • sonidoFlecha	
Espada	• manoAsignada • sonidoGolpe	
PiedraSagrada	• particulasPiedra • sonidoPiedra • animacionPiedra	
Teletransporte	• rayoTeletransporte	• Teletransporte-PuntoTeletrasnporte
PuntoTeletransporte	• particulasTeletransporte	
UIManager	• panelPausa • panelInformacion • rayoUI	
Menu Radial	• imágenes[]	
NPC	• dialogoNPC • sonidoNPC • animacionesNPC[]	

Tabla 2.52 - Clases, atributos y relaciones obtenidas del análisis

A partir de estas clases, atributos y relaciones se obtuvo el siguiente modelo del dominio:

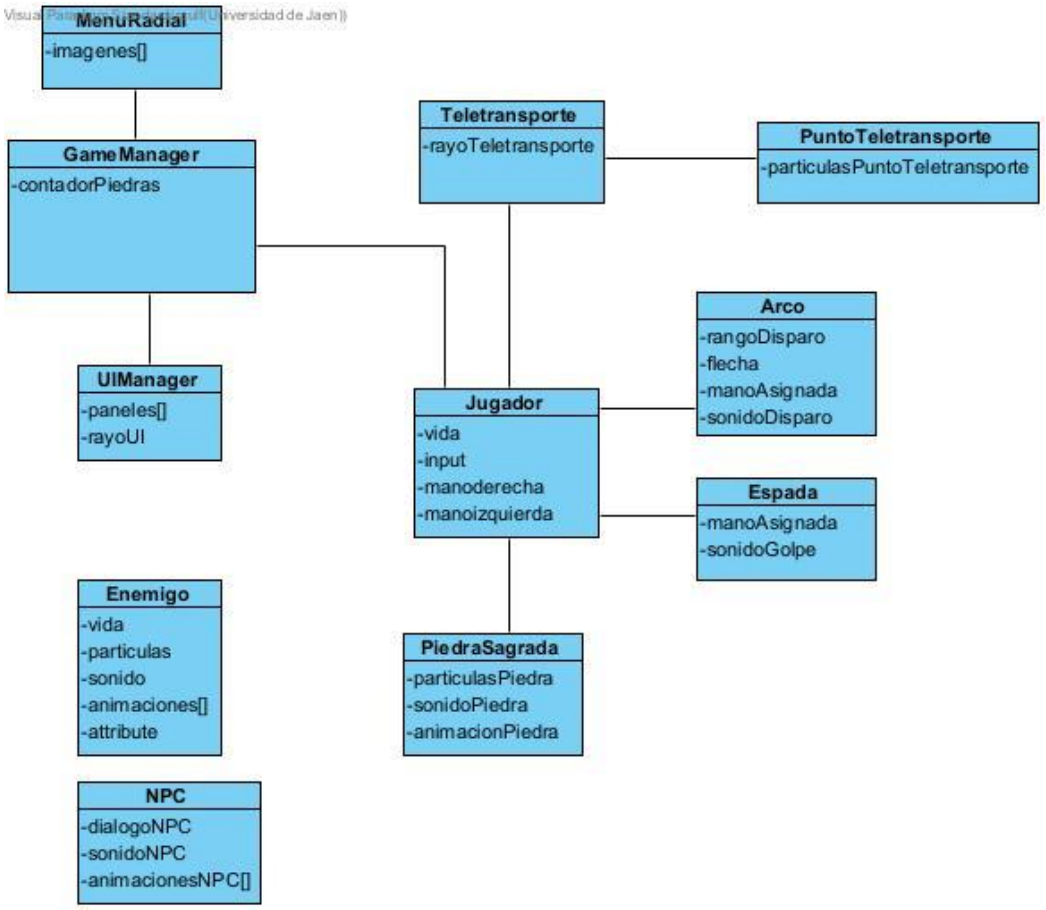


Ilustración 2.24 - Modelo del dominio desde el análisis

2.8.2 Diagrama de modos de juego (Flowboard)

Un diagrama de modos de juego (*Flowboard)* es un gráfico incluido en el libro "Fundamentals of game design 3rd ed de Ernest Adams" que mezcla dos tipos de diagramas: el **diagrama de flujo** (*flow-chart*) y **la historia de usuario** (*storyboard*). Este diagrama organiza los distintos modos de experiencia de juego, proporcionando una estructura al juego. A continuación, se muestra un *Flowboard (ilustración 2.25)* en el que se clarifica el funcionamiento general del sistema.

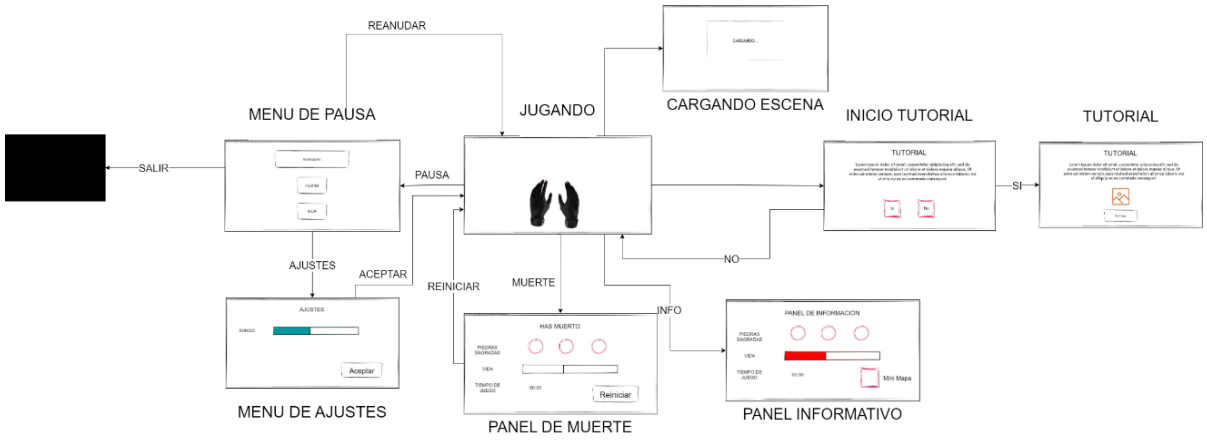


Ilustración 2.25 – Diagrama de modos de juego (Flowboard) del videojuego

2.9 Estimación del tamaño y esfuerzo

El modelo escogido para la estimación del tamaño y el esfuerzo ha sido **COCOMO** debido a su previo uso en el grado. COCOMO es una jerarquía de modelos de estimación de costes software que incluye los submodelos básico, intermedio y detallado. Se basa en la estimación previa del tamaño del software en líneas de código (LDC) utilizando datos históricos y factores de ajuste. Fue propuesto por Boehm y basado en el análisis de 63 proyectos.

COCOMO dispone de dos ecuaciones básicas que serán explicadas a continuación:

- $E = a \cdot (KDLC)^b \cdot M(x)$
- $TD = c \cdot E^d$

Donde E es la estimación, TD el tiempo dedicado, M(x) el factor de ajuste y a, b y c parámetros según el tipo de desarrollo del proyecto.

COCOMO clasifica el tipo de desarrollo según las siguientes características:

Tipo de desarrollo	Características
Orgánico	• Entorno de desarrollo estable • Poca innovación técnica • Tamaño pequeño(<50KLDC) • Tiempo relativamente corto
Empotrado	• Software con requerimientos muy restrictivos • Gran volatilidad(inestable) • Complejo

	• Entorno de gran innovación técnica
Semi-Libre	• Entre los dos anteriores

Tabla 2.52 - Clasificación según el tipo de desarrollo

El factor de ajuste se obtiene de la multiplicación de los valores de los atributos del proyecto según la tabla que se muestra a continuación:

Conductores de coste	Valoración					
	Muy bajo	Bajo	Nominal	Alto	Muy alto	Extra alto
Atributos del software						
Fiabilidad requerida del software	0,75	0,88	1,00	1,15	1,40	-
Tamaño de la base de datos	-	0,94	1,00	1,08	1,16	-
Complejidad del producto	0,70	0,85	1,00	1,15	1,30	1,65
Atributos de hardware						
Restricciones del tiempo de ejecución	-	-	1,00	1,11	1,30	1,66
Restricciones del almacenamiento principal	-	-	1,00	1,06	1,21	1,56
Volatilidad de la máquina virtual	-	0,87	1,00	1,15	1,30	-
Tiempo de respuesta del ordenador	-	0,87	1,00	1,07	1,15	-
Atributos de personal						
Capacidad del analista	1,46	1,19	1,00	0,86	0,71	-
Experiencia en la aplicación	1,29	1,13	1,00	0,91	0,82	-
Capacidad de los programadores	1,42	1,17	1,00	0,86	0,70	-
Experiencia en S.O utilizado	1,21	1,10	1,00	0,90	-	-
Experiencia en el lenguaje de programación	1,14	1,07	1,00	0,95	-	-
Atributos del proyecto						

Prácticas de programación modernas	1,24	1,10	1,00	0,91	0,82	-
Utilización de herramientas software	1,24	1,10	1,00	0,91	0,83	-
Limitaciones de planificación del proyecto	1,23	1,08	1,00	1,04	1,10	-

Tabla 2.53 - Factores de ajuste

Lo primero que se ha de hacer es clasificar el proyecto según su tipo de desarrollo. Dado el tamaño pequeño del proyecto, su innovación técnica y su tiempo relativamente corto se clasificará como **Semi-Libre**. Una vez clasificado el proyecto se pueden realizar tres versiones de COCOMO: básico, intermedio y detallado. En este caso se va a realizar la **versión intermedia**. Los valores de los parámetros a, b y c de la ecuación vienen dados por la siguiente tabla:

Tipo	a	b	c	d
Orgánico	3.20	1.05	2.50	0.38
Semi-Libre	3.00	1.12	2.50	0.35
Empotrado	2.80	1.20	2.50	0.32

Tabla 2.54 - Valores según el tipo de proyecto

Por último, es necesario calcular el factor de ajuste, para ello se han tenido en cuenta los siguientes factores:

- Fiabilidad del software nominal: 1,00
- Tiempo de respuesta bajo: 0,87
- Capacidad de los programadores nominal: 1,00
- Experiencia en la aplicación baja: 1,13
- Experiencia con el lenguaje de programación baja: 1,07
- Utilización de herramientas software alta: 0,91
- Limitación de planificación del proyecto alta: 1,04

Por tanto $M(x) = 1{,}00 \cdot 0{,}87 \cdot 1{,}00 \cdot 1{,}13 \cdot 1{,}07 \cdot 0{,}91 \cdot 1{,}04 = 0{,}99$

Aplicando los demás valores y estimando que el proyecto contará con aproximadamente 3 KLDC obtenemos los siguientes resultados:

- $E = 3 \cdot 3^{1,12} \cdot 0,99 = 10,16 \frac{persona}{mes}$
- $TD = 2,50 \cdot 10,16^{0,35} = 5,62\ meses$
- $P = \frac{10,16}{5,62} \cong 2\ personas$
- $LDCM = \frac{3000}{5,62} \cong 534$ líneas de código / mes

Tras esto se concluye que el número estimado de líneas de código al mes es de 534 y que el proyecto tendrá una duración de 5,62 meses aproximadamente.

2.10 Planificación temporal

A continuación, se muestra información sobre los distintos incrementos del proyecto.

Incremento	Fecha de Inicio	Fecha de finalización	Tareas	Subtareas
Incremento 1	04/04/2022	15/05/2022	1	4
Incremento 2	06/06/2022	31/07/2022	2	7
Incremento 3	01/08/2022	01/09/2022	2	0

Tabla 2.55 - Incrementos

En la *ilustración 2.26* se muestra el diagrama de Gantt para la planificación del proyecto. En él se puede observar que la planificación se ha organizado en tres incrementos. Durante el primer incremento se llevará a cabo el desarrollo de los aspectos básicos del videojuego, sin llegar a implementar la realidad virtual. A continuación, habrá un periodo en el que no se trabajará debido al periodo de exámenes de la Universidad de Jaén. Seguidamente se incorporará la realidad virtual al videojuego, para ello se definirán las interacciones del usuario con el entorno virtual y se adaptará el entorno creado previamente para asegurar la inmersión completa del jugador. Este incremento será el que más tiempo conlleve. Por último, se desarrollarán

la interacción por voz y la interfaz de usuario, aspectos de vital importancia para la consecución de la interacción multimodal en el videojuego.

Podemos concluir que la duración del proyecto será de **22 semanas (5 meses)**, de principios de abril hasta finales de agosto. Cabe destacar que el desarrollo de la documentación no se ha incluido en la planificación porque esta se ha ido realizando a lo largo del proyecto.

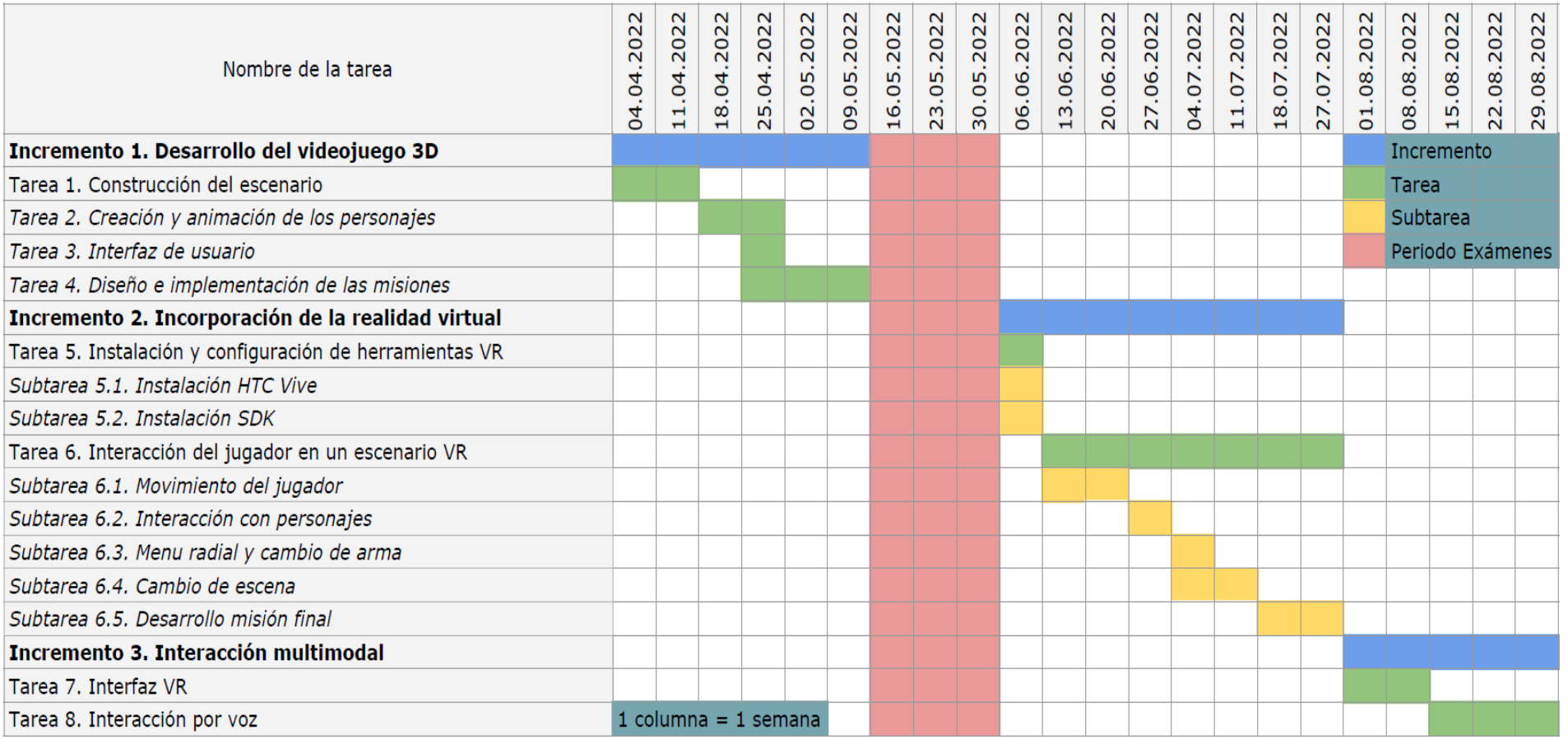


Ilustración 2.26 - Diagrama de Gantt de la planificación

2.11 Análisis de riesgos

Para la realización de este apartado se ha recurrido a la creación de una **tabla de gestión de riesgos** para clasificar los distintos posibles riesgos que pueden afectar tanto al planteamiento como al desarrollo del trabajo.

ID	Riesgo	Ámbito	Plan de actuación	Probabilidad	Impacto
R-01	Fallo en la instalación de las gafas de VR en el equipo debido a las bajas prestaciones de este.	Proyecto	Contingencia	0.3	0.8
R-02	Lento rendimiento del videojuego en el equipo de desarrollo debido a la alta complejidad de este.	Producto	Contingencia	0.5	0.5
R-03	Retraso en la entrega del proyecto	Proyecto	Contingencia	0.3	0.8
R-04	La herramienta de reconocimiento de voz no actúa como se esperaba	Producto	Contingencia	0.3	0.4
R-05	La familiarización con el desarrollo en VR lleva más tiempo del esperado	Proyecto	Mitigación	0.1	0.2

Tabla 2.56 - Gestión de riesgos

Para el riesgo **R-01** se ha diseñado un plan de contingencia que consistirá en pedir prestado un equipo que cuente con los requisitos hardware mínimos para el correcto funcionamiento de las gafas de VR.

Para el riesgo **R-02** se ha diseñado un plan de contingencia que propone crear proyectos simples individuales para probar el funcionamiento de las distintas funcionalidades por separado.

Para el riesgo **R-03** el plan de contingencia consistirá en desechar funcionalidades secundarias y realizar horas extra.

Para el riesgo **R-04** se ha diseñado un plan de contingencia que propone buscar herramientas alternativas que actúen de la forma adecuada.

Por último, para el riesgo **R-05** se ha optado por la mitigación, tratando de evitar que esta situación ocurra realizando un estudio previo al comienzo del proyecto.

2.12 Presupuesto

En este apartado se tomarán como referencia los datos obtenidos en el apartado 1.15 sobre el esfuerzo y el tiempo. Además, se desglosará el precio de los componentes hardware y software utilizados a lo largo del desarrollo del proyecto.

En la *tabla 2.58* se muestran los distintos **componentes hardware** utilizados, así como su coste y vida útil estimada, necesaria para calcular la amortización.

Componentes Hardware	Coste	Duración (Meses)	Amortización
Ordenador Asus Tuf Gaming FX504GD-EN561(Intel Core i7-8750H, 16GB RAM, 1TB HDD, 128GB SSD, GeForce GTX1050)	800€	60	13,33€
Kit HTC Vive Gafas de Realidad Virtual	638,99€	72	8,87€
Monitor Xiaomi 23.8”	100€	48	2,08€
Ratón Corsair RGP0030	32,79€	48	0,68€
Teclado Krom Kernel TKL	49,90€	48	1,03€
Total			**25,99€**

Tabla 2.57 - Tabla de costes hardware

Teniendo en cuenta la amortización de los componentes hardware y que la duración del proyecto es de 5 meses, se estima un **gasto total en hardware de 129,95€**

En la *tabla 2.59* se muestra el desglose del **software** utilizado. La mayoría de los programas son gratuitos, lo cual reducirá los gastos.

Componentes Software	Coste
Microsoft Word	149€
Windows 10 Home	149.99€
Unity 3D	0€
Visual Paradigm	0€
Visual Studio Code	0€
Steam	0€
SteamVR	0€
Plastic SCM	0€
Audacity	0€
Windows Speech API	0€
SteamVR Plugin	0€
TOTAL	**298,99€**

Tabla 2.59 - Tabla de costes software

Para calcular el **presupuesto relacionado con el personal**, se han tenido en cuenta los principales roles presentes en el ciclo de vida de desarrollo de un videojuego. Se ha escogido un salario promedio para cada uno de los roles. Para la duración de cada rol se ha tenido en cuenta el diagrama de Gantt para la planificación del proyecto.

- Programador
- Diseñador de videojuegos
- Artista3D
- Guionista de videojuegos
- Productor audiovisual

Personal	Objetivo	Sueldo semanal (bruto)	Duración	Sueldo Total
Programador	Programar la lógica del videojuego	624,45€	14 semanas	8.742,41€
Diseñador del videojuego	Definir la experiencia de usuario y escoger las	647€	3 semanas	1.941€

	herramientas para el desarrollo			
Artista 3D	Diseñar los aspectos visuales y animar los personajes del videojuego	449,75€	1 semana	449,75€
Guionista del videojuego	Escribir la historia y narrativa del videojuego	395€	1 semana	395€
Productor del videojuego	Gestionar el marketing, así como la toma de decisiones según el feedback de los usuarios	471€	2 semanas	942€
Total				**12.470,16€**

Tabla 2.60 - Tabla de costes de personal

Por otro lado, para calcular los **costes indirectos** se ha establecido una cuantía en función del presupuesto total del proyecto. En este caso estos gastos van a suponer un coste extra del 10% del coste total. De este modo podemos concluir lo siguiente:

Tipo de coste	Coste
Costes hardware	129,95€
Costes software	298,99€
Costes de personal	12.470,16€
Costes indirectos	1.289,91€
Coste total del proyecto	**14.189,01€**

Tabla 2.61 - Coste total del proyecto

Por lo tanto, como se observa en la *tabla 2.61,* el presupuesto estimado para el proyecto es finalmente de **14.189,01€**

(Página intencionalmente en blanco)

3 EJECUCIÓN DEL PROYECTO

(Página intencionalmente en blanco)

En este capítulo se procede a explicar en detalle cada uno de los incrementos realizados durante el desarrollo del proyecto. Para ello se dividirá en los siguientes subapartados:

- **Incremento 1**. Desarrollo de videojuego 3D.
- **Incremento 2**. Incorporación de la Realidad Virtual.
- **Incremento 3**. Interacción multimodal.

En cada uno de los incrementos se seguirá una estructura similar para lo cual se definirán una serie de secciones comunes:

- **Lista de requisitos**: En esta sección se realizará una selección y se enumerarán los requisitos que se van a implementar durante el desarrollo del incremento.
- **Desarrollo**: En esta sección se detallarán los procesos y técnicas utilizados para llevar a cabo el diseño e implementación del incremento.
- **Resultados**: En esta sección se dará una valoración general sobre los avances que se han obtenido con el incremento, así como una conclusión de los resultados obtenidos.
- **Pruebas**: En esta sección se mencionarán las pruebas llevadas a cabo para la validación de las funcionalidades o componentes desarrollados en el incremento.

3.1 Incremento 1. Desarrollo del videojuego 3D

Este primer incremento tuvo lugar desde el **04/04/2022** hasta el **15/05/2022**, es decir, tuvo una duración de **6 semanas**, este supuso la implementación del juego base al que más tarde se le añadiría la realidad virtual.

3.1.1 Requisitos seleccionados/ Requisitos a implementar

Los **requisitos** que se van a implementar en este incremento son los siguientes:

- FRQ-022. Escenario abierto
- FRQ-023. Enemigos

- FRQ-024. Misiones
- FRQ-025. Personajes animados
- FRQ-017. Contador de piedras

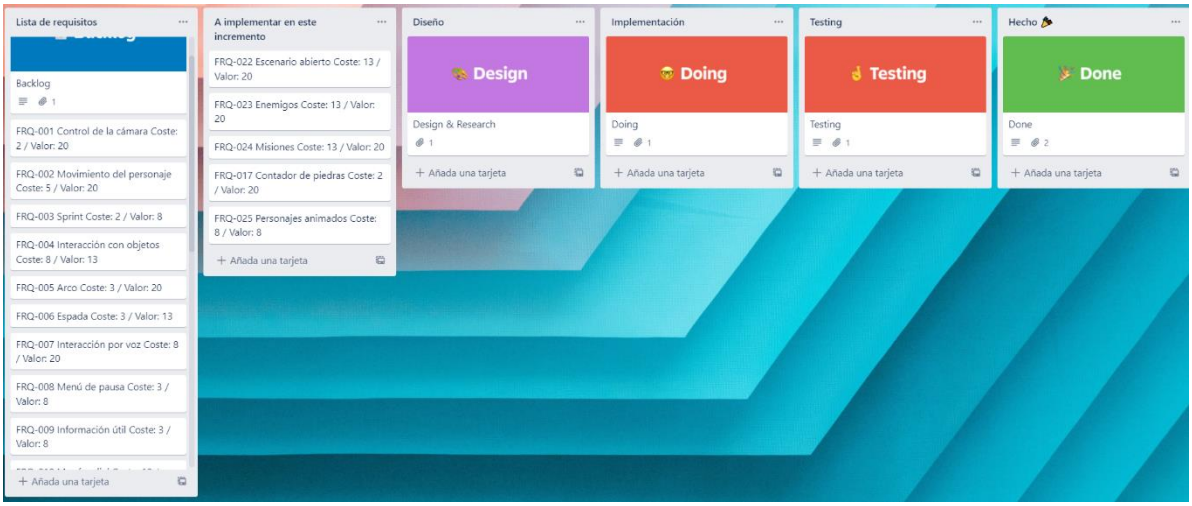


Ilustración 3.1 - Tablero Kanban al comienzo del incremento

Además, para satisfacer los requisitos se tratarán de plasmar, mediante las herramientas que ofrece Unity, las ideas mostradas en el apartado 2.4 sobre los escenarios del videojuego.

3.1.2 Desarrollo

Se comenzó por crear un nuevo proyecto en Unity, para ello se escogió la opción de proyecto 3D que ofrece Unity y la versión 2020.3.36f1 como se puede observar en la *ilustración 3.2.*

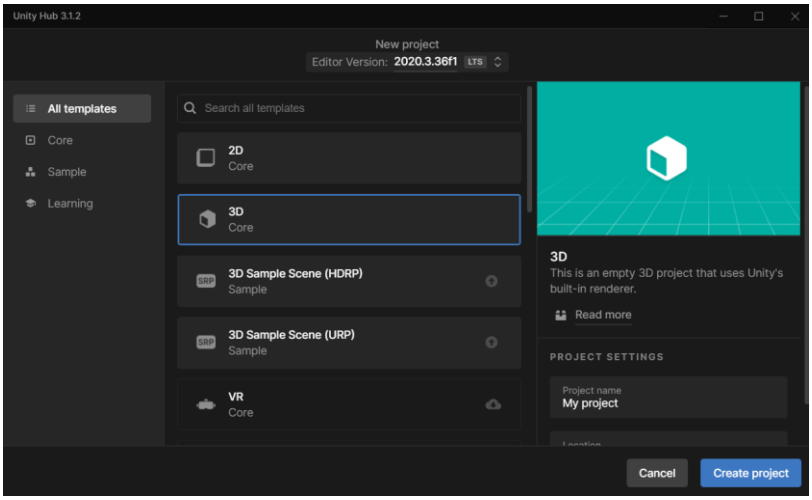


Ilustración 3.2 - Captura de la pantalla de creación de un nuevo proyecto en Unity

A continuación, se describen las distintas subtareas realizadas a lo largo del incremento.

3.1.2.1 Construcción del escenario

En primer lugar, se comenzó por crear el escenario principal del videojuego, la isla. Ya se tenía una ligera idea de cómo iba a ser la forma de esta, así que intento plasmar la idea inicial utilizando las herramientas de modelado de terreno que ofrece Unity. La isla iba a estar rodeada de agua, por lo que el terreno debía ser bastante grande. Se utilizaron **9 terrenos** de Unity. En la *ilustración 3.3* se puede observar el mapa una vez modelado y pintado. En la isla se pueden diferenciar claramente 4 zonas. Una zona central más verdosa que es la zona del bosque. Una zona rocosa en la parte inferior de la imagen, una zona volcánica en la parte superior y por último una pequeña zona de costa en la parte derecha de la imagen.

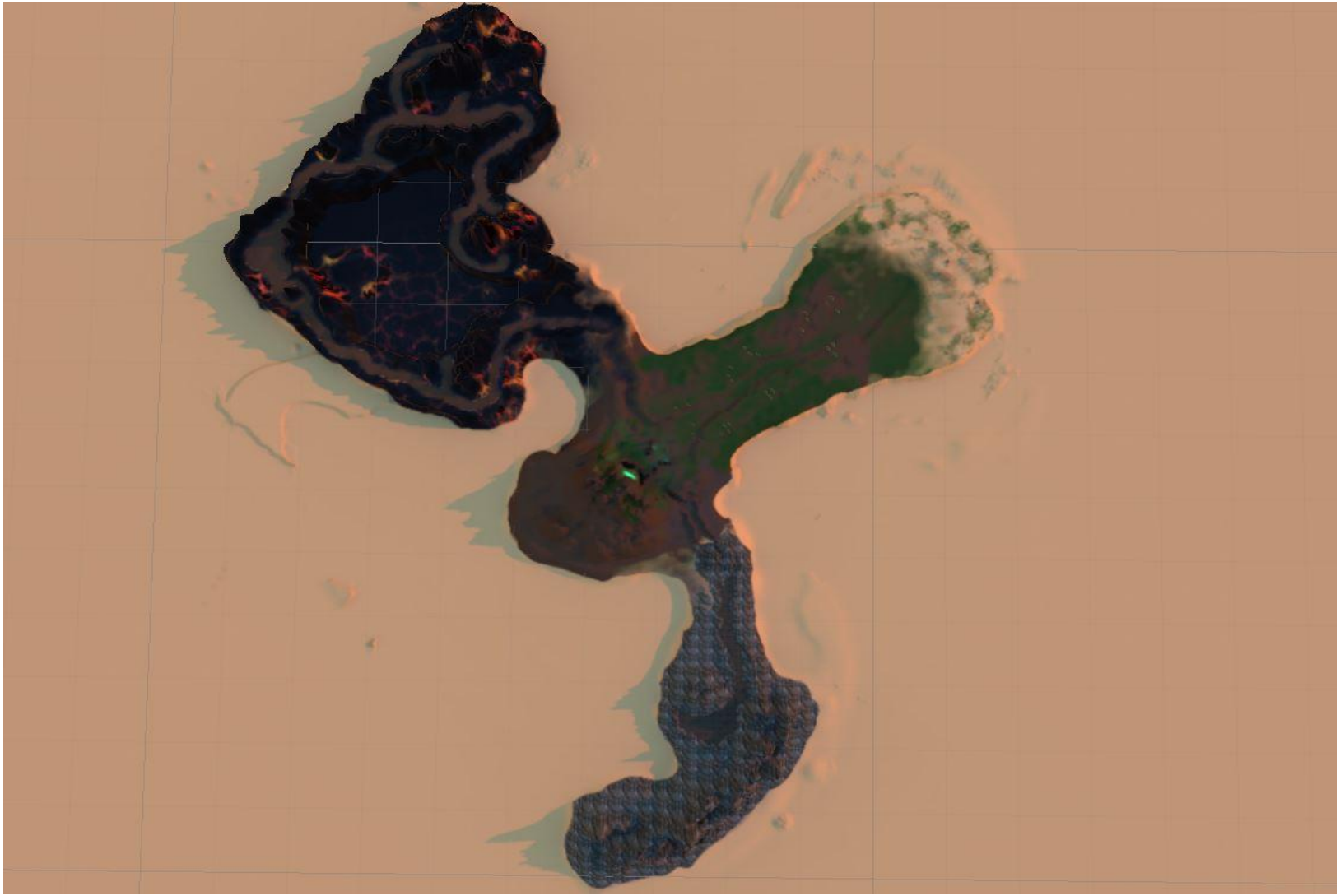


Ilustración 3.3 - Imagen de la isla

A continuación, una vez modelado y pintado el terreno, se procedió a añadir elementos decorativos que mejorasen la calidad del escenario, como árboles, flores y rocas.

Para la **zona de la playa** se decidió usar un modelo de palmera encontrado en la *Unity Asset Store*, así como unas piedras para decorar el terreno. En las *ilustraciones 3.4 y 3.5*, y se puede observar la evolución de la zona.

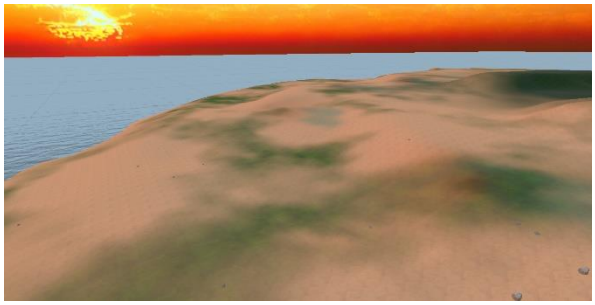


Ilustración 3.5 - Imagen de la zona de la playa antes

Ilustración 3.4 - Imagen de la zona de la playa después

Para la **zona del bosque** se decidió utilizar varios modelos de árboles que se encontraron en la *Unity Asset Store*. Además, también se usaron modelos de arbustos y flores y hierba para el terreno.

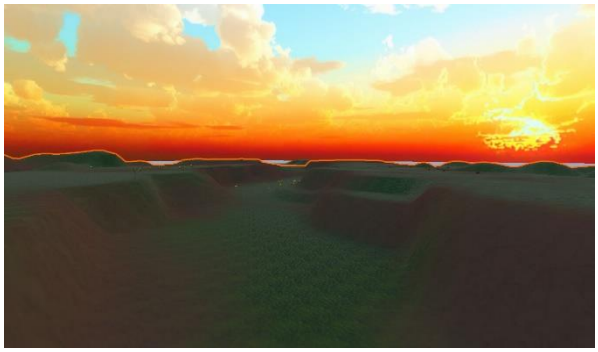


Ilustración 3.6 - Imagen de la zona del bosque antes

Ilustración 3.7 - Imagen de la zona del bosque después

Para crear el **templo sagrado** se recurrió también a la tienda de Unity. Se encontró un paquete idóneo para su creación. Se trata de un paquete con infinidad de *prefabs* (suelos, paredes, columnas, elementos decorativos, …) para que los usuarios

puedan crear sus propios templos. De este modo se construyó el templo y su decoración. Además, para que el jugador no pudiera ver el interior, se colocó un sistema de partículas simulando niebla en la entrada del templo.

Ilustración 3.8 - Templo

En la **zona de lava** o **zona volcánica** se añadió un lago de lava. Para esto se usó un plano de Unity al que se le añadió un material que simulaba la lava y su movimiento. Además, se añadieron algunos elementos decorativos como rocas y efectos de partículas simulando fuego y humo.

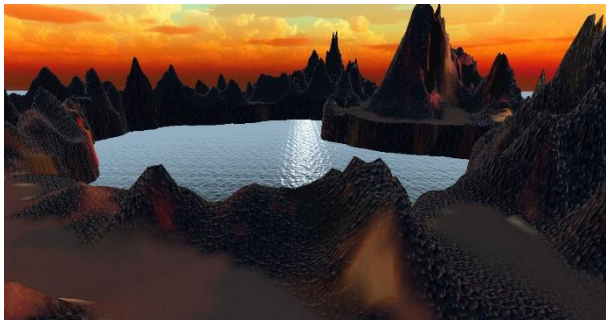


Ilustración 3.9 - Imagen de la zona de lava antes

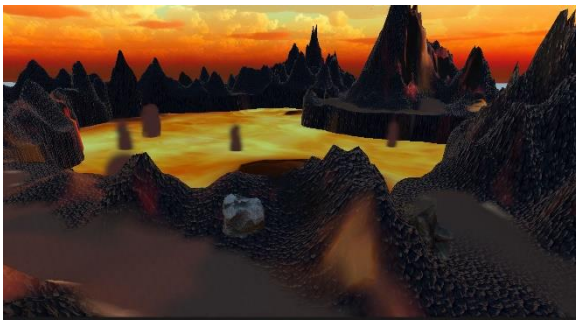


Ilustración 3.10 - Imagen de la zona de lava después

Para la **zona rocosa** se añadió un puente de madera que habilita el paso por las montañas. Además, se añadieron rocas como elemento decorativo.

Ilustración 3.11 - Zona rocosa antes

Ilustración 3.12 - Zona rocosa después

Además, en la parte final de esta zona se colocó la entrada a la mina, que consistía en unas rocas en forma de entrada y un sistema de partículas simulando una niebla para que el jugador no viera el interior.

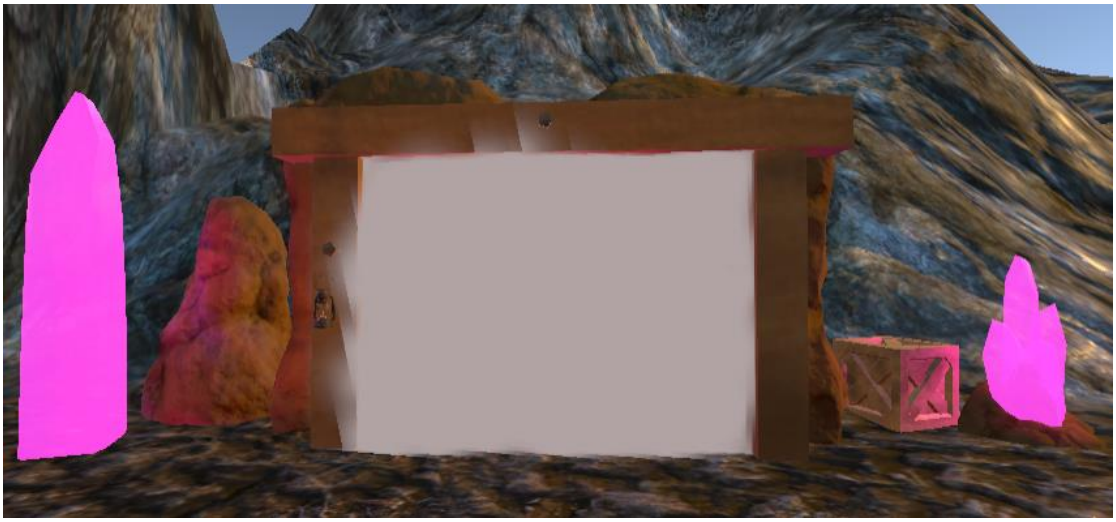
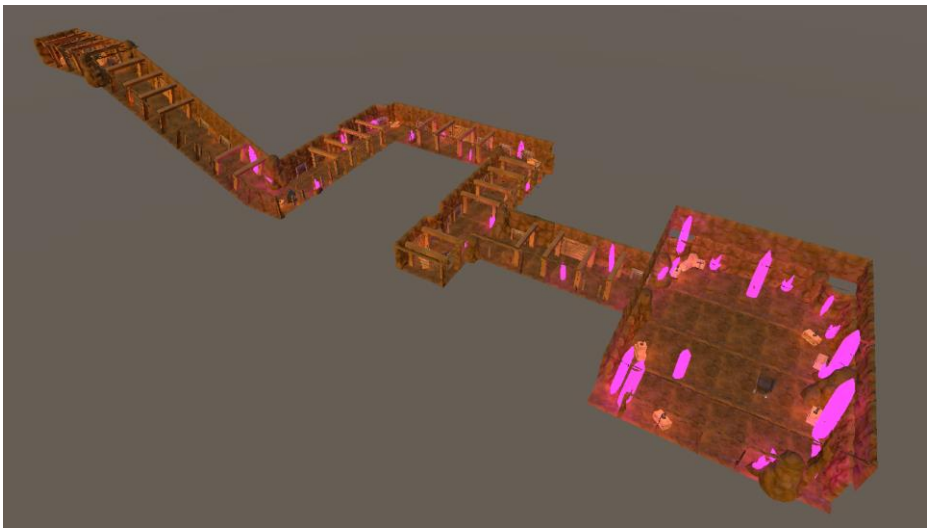


Ilustración 3.13 - Entrada de la cueva

Para el **interior de la cueva** se utilizaría un modelo 3D de una cueva obtenido de la tienda de *assets* de Unity y se instanciarían en ella algunos elementos decorativos.



También se añadió un skybox que simulaba un cielo despejado.

3.1.2.2 Creación y animación de personajes

Para la creación de los personajes del juego se recurrió una vez más a la *Unity Asset Store* y a la página de *Sketchfab*, una web para la descarga de modelos 3D. Los personajes del videojuego ya se han descrito anteriormente en el apartado de personajes, por lo que esta sección se centrará más en la parte de implementación y de las funciones que realizan.

En primer lugar, se vio conveniente crear a los **NPC** (*Non-Playable-Character*), es decir, los personajes con los que el jugador podría interactuar y que lo guiarían en su aventura.

El primero que se creó fue el **Elfo.** El modelo 3D fue descargado de la tienda de Unity, sin embargo, este venía sin aminaciones incorporadas. Para crear las animaciones se utilizó la herramienta **Mixamo,** que además también se utilizaría posteriormente para otros personajes**.** Esta es una página web que permite la animación de modelos 3D con aspecto humanoide a través de un generoso catálogo de animaciones entre las que se puede escoger de forma gratuita.

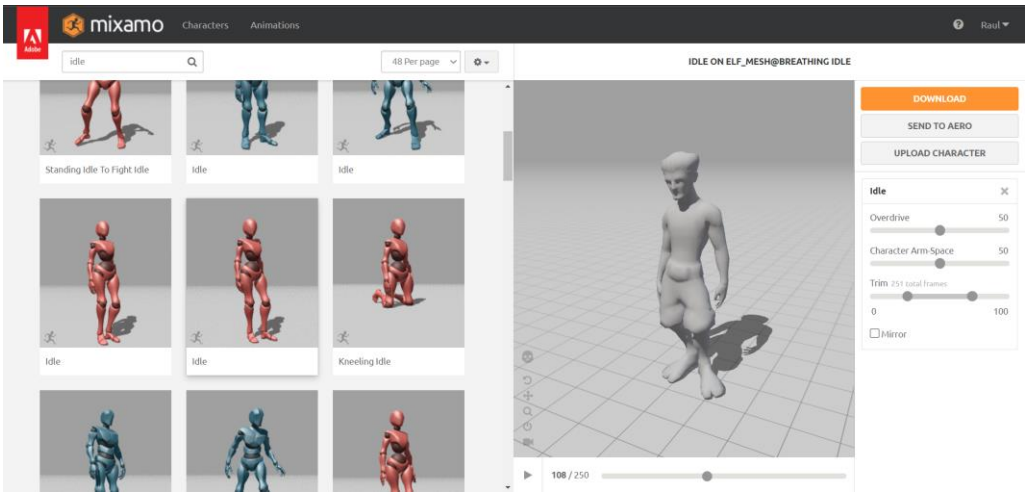


Ilustración 3.14 - Herramienta Mixamo para la animación de personajes

En este caso era necesaria una animación de **IDLE** que es la animación básica que hace que el personaje cuando no está realizando ninguna acción. Tras arrastrar el personaje a la escena, se le añadió un collider y un script en el que se programaría su comportamiento. En la *ilustración 3.15* se muestra el comportamiento básico del personaje.

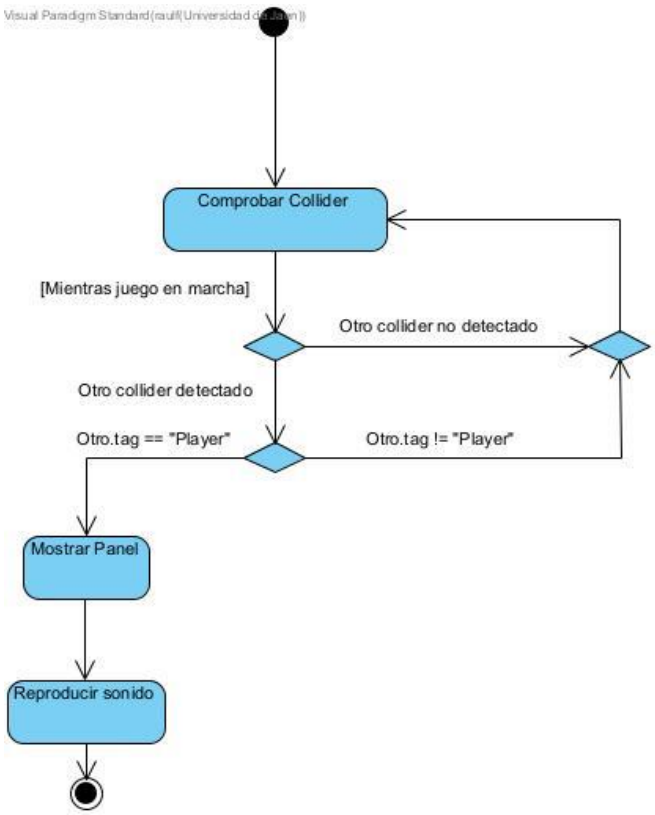


Ilustración 3.15 - Diagrama del comportamiento general de un NPC

Para los demás NPC se siguió el mismo procedimiento, se descargaban los modelos 3D, se animaban si era necesario (algunos de ellos ya incluían animaciones y para los que no se volvió a recurrir a Mixamo), se añadían a la escena y se programaba su lógica. Todos se programaron de forma parecida. Si el jugador entra dentro de su rango de detección se activa el panel de diálogo y se reproduce algún sonido.

En segundo lugar, se creó a los **enemigos**, en este caso hay tres, dos de ellos son agresivos y el otro es neutral. La idea era la de buscar un tipo de enemigo para cada zona de la isla. Todos los enemigos fueron descargados de la tienda de *assets* de Unity.

El primer enemigo que se creó fue el **enemigo de la zona de la playa**. Este consiste en un pájaro con una animación en bucle que simula el vuelo y que se encuentra encima de algunas palmeras de esta zona. Estos enemigos están directamente instanciados al inicio del juego en posiciones fijas. Su programación es bastante simple. Si detectan la entrada de un *collider* con una etiqueta determinada emiten un sonido y unas partículas que simulan daño.

Ilustración 3.16 - Pájaro

El siguiente enemigo que se creó fue el **enemigo de la zona rocosa**. Este es un duende cuyo comportamiento tiene algo más de complejidad. En resumen, lo que hace es comprobar la distancia con el jugador constantemente y actuar en base a esa distancia. Si está lejos alternará aleatoriamente entre las animaciones de idle y de caminar. Si está cerca correrá hacia el jugador y atacará. En la *ilustración 3.17* se muestra con más detalle el funcionamiento del comportamiento del enemigo.

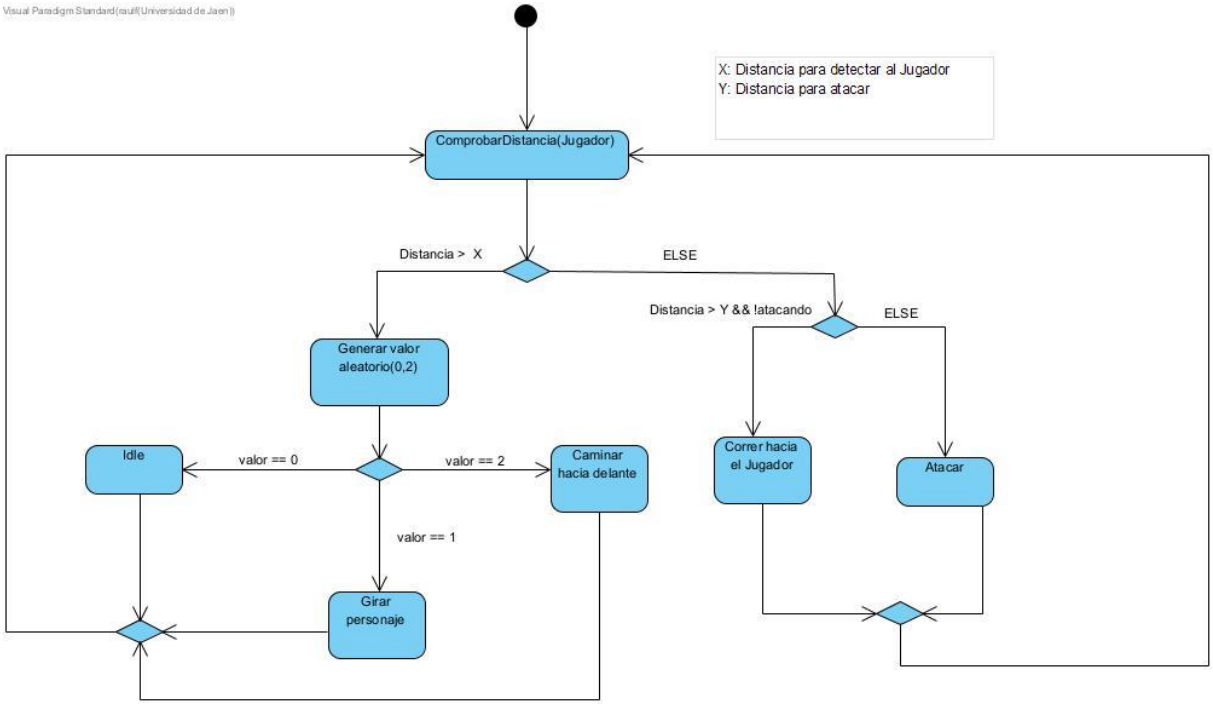


Ilustración 3.17 - Diagrama de estados del comportamiento del enemigo Duende

Ilustración 3.18 - Duende

Para eliminar a este enemigo, el jugador deberá usar alguna de sus armas. Cuando el enemigo muere realiza una animación que simula su muerte.

El último enemigo fue el de la zona del volcán. Este es una criatura de forma esférica que flota y dispara bolas de fuego. El comportamiento es bastante similar al del enemigo anterior, con la salvedad de que el duende ataca cuerpo a cuerpo mientras que este lo hace a distancia. Para ello, cuando este está en posición de atacar, se utiliza la función **Instantiate()** de Unity que permite instanciar copias de un objeto. En este caso se instancian copias del objeto "bola de fuego". Desde otro *script* se controla el movimiento de la bola que simula estar flotando. Estas copias son destruidas al colisionar con el jugador o en su defecto, después de un tiempo predefinido.

Ilustración 3.20 - StoneMonster

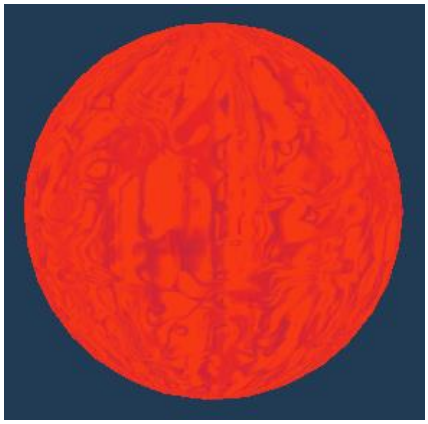


Ilustración 3.19 - Bola de fuego

3.1.2.3 Interfaz de usuario

Para la interfaz de usuario se diseñaron algunos componentes que se utilizarían más adelante, pero no se indagó en profundidad porque luego se desarrollaría más a fondo cuando se implantase la VR.

En primer lugar, se pensó que sería conveniente dar información al usuario sobre qué objetos tiene. Por eso se utilizó la herramienta Paint3D para crear un **icono** para cada piedra sagrada. Estos iconos aparecerían transparentes al principio del juego y ganarían opacidad cuando el jugador recogiera los objetos correspondientes.

Ilustración 3.21 - Icono piedra sagrada púrpura (transparente)

Ilustración 3.22 - Icono piedra sagrada roja

Ilustración 3.23 - Icono piedra sagrada azul

También se diseñó una **barra de vida** para que el jugador supiera en todo momento el nivel de vida restante.

Ilustración 3.24 - Barra de vida

Para los **paneles de diálogo** se utilizaría el siguiente fondo y la fuente para el texto “INMORTAL”

Ilustración 3.25 - Fondo del panel de diálogo

Ilustración 3.26 - Fuente del panel de diálogo

Por último, se vio conveniente que el jugador contase con una pequeña ayuda para moverse por la isla. Por eso se pensó en la idea de incluir un **mini mapa** para que el jugador pudiera consultar su posición en todo momento. Para ello se creó una

cámara que estaría a varios metros sobre el jugador y que apuntaría hacia abajo. También se añadió una capsula que estaría encima del jugador y ayudaría a localizarlo en el mini mapa.

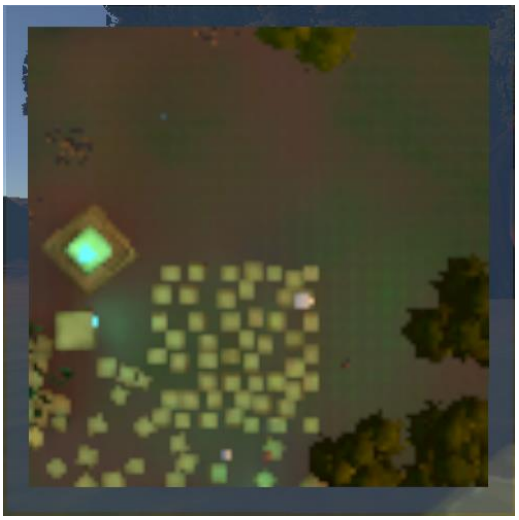


Ilustración 3.27 – Mini mapa

3.1.2.4 Diseño e implementación de las misiones

Para obtener las tres piedras sagradas el jugador deber superar tres misiones que le son asignadas en cada zona de la isla (sin contar la zona inicial). El arma principal del juego, como se ha mencionado anteriormente, es un arco, por lo tanto, las misiones preferiblemente debían obligar al jugador a usar el arco o en su defecto, alguna de sus otras armas.

Para la **zona de la playa** la misión la ofrece el NPC del mono. Este se encuentra entre el final de la zona del bosque y el comienzo de la playa. Al acercarse el jugador a él, este muestra un panel con el diálogo. Además, se genera otro panel para dar la opción al jugador de aceptar la misión. Esta misión consiste en acabar con todos los pájaros de la playa. Una vez en la misión se establece un contador de pájaros utilizando la función **FindObjectsWithTag("Pajaro")**. Esta función busca en la escena todos los objetos con la etiqueta dada. De este modo se obtiene el número total de objetos con la etiqueta pájaro en la escena. Cuando el jugador elimina un pájaro, se decrementa el contador.

Cuando el contador llega a cero se da por completada la misión y se lanza una función que materializa la piedra junto a unos efectos de partículas al lado del NPC.

Para la **zona rocosa** el jugador debe introducirse en la mina que se encuentra a final del camino. Una vez dentro, el NPC minero le ofrece la misión de esta zona. La misión consiste en destruir todos los minerales que hay en la cueva, se puede usar el arco o un pico (*ilustración 3.28*) situado junto al minero. También el jugador puede interactuar con una serie de lámparas situadas por el recorrido (*ilustración 3.29*). El "modus operandi" del código es muy similar al anterior. Se utiliza una función para obtener el número de cristales de la escena y se va decrementando conforme el jugador los va destruyendo. Para detectar la colisión de las armas con los cristales se creó un script asignado a cada cristal que al detectar la colisión generara partículas, sonido y destruyera el propio cristal. Cuando el contador de cristales llega a 0 se activa la piedra sagrada de la zona y el jugador puede recogerla. Una vez recogida aparece un objeto que actúa de teletransportador y permitirá al jugador volver a la zona inicial.

Ilustración 3.28 - Pico

Ilustración 3.29 - Lámpara

Para la **zona del volcán** la misión es distinta. Esta consiste en llevar un objeto desde la mitad del camino hasta el final, donde se encuentra el murciélago ciego. De hecho, el objeto que el jugador debe entregarle es el ojo que este está buscando. Lo que dificulta la misión son los enemigos que se encuentran por la zona. Para mover el ojo el jugador puede optar por dispararle flechas o cogerlo con la mano. Para detectar la colisión del ojo con el personaje del murciélago se definió un *collider*

alrededor del murciélago. Cuando el jugador se aproxima y entrega el ojo aparece la piedra mágica roja junto con unas partículas y un sonido. También se realizan una serie de acciones que hacen desaparecer el ojo que hemos cogido y lo sitúan en la zona ocular del murciélago, dando la sensación de que el jugador se lo ha colocado.

Una vez el jugador ha reunido las tres piedras sagradas puede entrar en el templo. Para ello existe una variable estática en la clase **Jugador**, que se incrementa cuando el jugador recoge una piedra.

Las misiones de la zona del templo se pospusieron a cuando se introdujera la realidad virtual.

3.1.3 Resultados

Para el final de este incremento ya se había construido la base del videojuego, sobre la que se integraría la realidad virtual, que incluiría modificaciones en algunos aspectos tras descubrirse las posibilidades que esta ofrece. En la *ilustración 3.30* se muestra un diagrama de clases del videojuego hasta el momento. En él se pueden observar cinco paquetes que representan los elementos principales del videojuego y en los que se agrupan las diferentes clases.

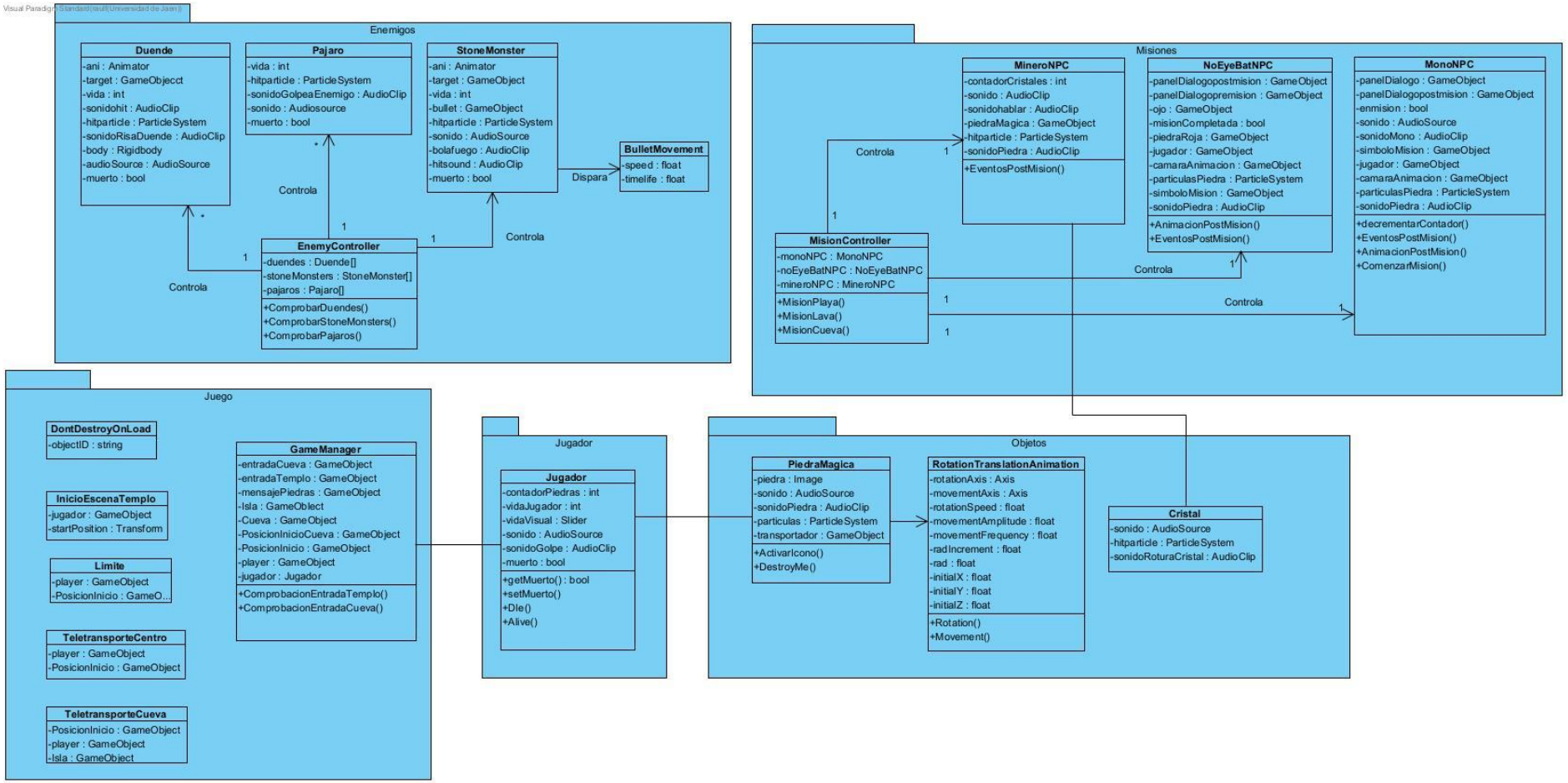


Ilustración 3.30 - Diagrama de clases del primer incremento

3.1.4 Pruebas

Para realizar las pruebas era necesario instanciar al jugador. Para ello se utilizó un recurso del paquete de recursos estándar de Unity, el **FPSController**, que permite controlar un personaje en primera persona y se le asignó la etiqueta “Player”. Aun con este recurso hay muchas cosas que no se podían probar, como todo aquello en lo que intervengan las armas, ya que aún no se habían implementado por completo porque estas sufrirían cambios llegada la VR. Las pruebas realizadas se recogen en las siguientes tablas:

ID y Nombre	PR-01 Test de escenario abierto
Resultado esperado	El usuario puede caminar por el escenario abiertamente.
Resultado obtenido	El usuario puede caminar por el escenario abiertamente.
Observaciones	Es necesario establecer unos límites del mapa para que el jugador no salga de la isla.

ID y Nombre	PR-02 Test de interacción con enemigos
Resultado esperado	Los enemigos reaccionan ante la presencia de un GameObject con etiqueta “Player”.
Resultado obtenido	Los enemigos reaccionan correctamente ante la presencia del jugador.
Observaciones	Es necesario establecer límites para que los enemigos no se salgan del mapa.

ID y Nombre	PR-03 Test de recompensas de las misiones
Resultado esperado	Al finalizar las misiones se genera un evento que hace aparecer una piedra sagrada
Resultado obtenido	Las piedras sagradas aparecen correctamente junto a sus partículas y sonidos.
Observaciones	

ID y Nombre	PR-04 Test de animación de los personajes
Resultado esperado	Todos los personajes están correctamente animados
Resultado obtenido	Tanto los NPCs como los enemigos presentan una correcta animación.
Observaciones	Podrían añadirse animaciones a los NPC a parte de la de Idle.

3.2 Incremento 2. Incorporación de la Realidad Virtual.

Este segundo incremento tuvo lugar desde el 06/06/2022 hasta el 31/07/2022, es decir, tuvo una duración de **8 semanas**. En él se procedió a integrar en el proyecto la realidad virtual, para lo cual fue necesario la instalación y una previa familiarización con el entorno.

3.2.1 Requisitos seleccionados/ Requisitos a implementar

Los **requisitos** que se van a implementar en este incremento son los siguientes:

- FRQ-001. Control de la cámara
- FRQ-002. Movimiento del personaje
- FRQ-003. Sprint
- FRQ-004. Interacción con objetos
- FRQ-005. Arco
- FRQ-006. Espada
- FRQ-012. Ataque
- FRQ-014. Muerte
- FRQ-015. Diálogos
- FRQ-019. Teletransporte
- FRQ-010. Menú radial
- FRQ-021. Partículas de daño

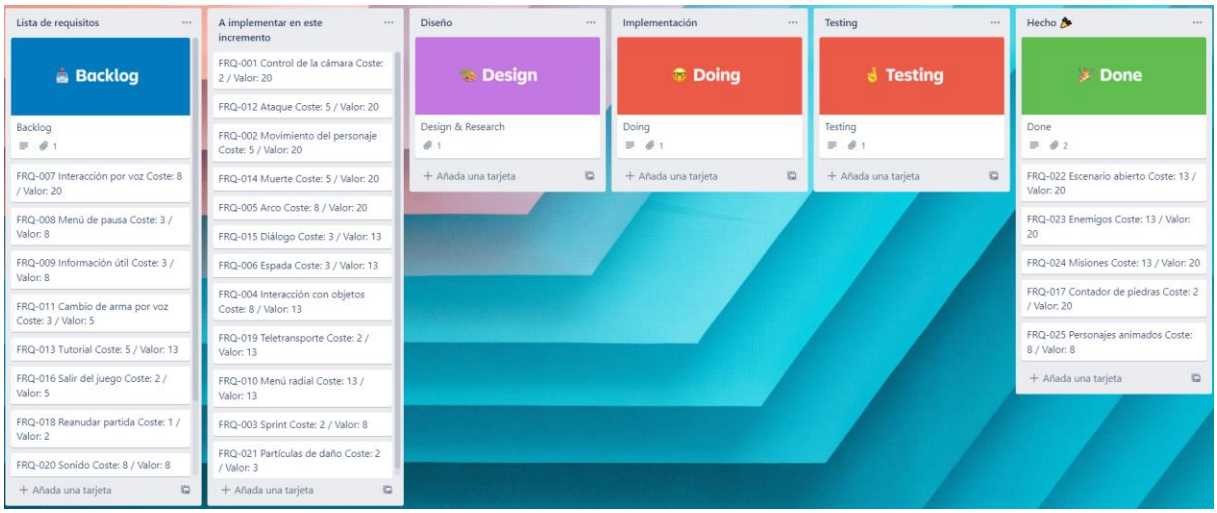


Ilustración 3.31 - Tablero Kanban al comienzo del incremento

3.2.2 Desarrollo

3.2.2.1 Instalación y configuración de herramientas VR

En primer lugar, se comenzó por la instalación del software necesario para el funcionamiento de las gafas HTC Vive. Una vez instalado se realizó el tutorial que ofrece SteamVR para la instalación física del visor. La caja contiene muchos elementos los cuales se muestran en la *ilustración 3.32*.

Ilustración 3.32 - Elementos HTC Vive [12]

[12] https://elchapuzasinformatico.com/2016/03/htc-vive-unboxing-accesorios/)

Para el correcto funcionamiento de las gafas, es necesario colocar dos sensores que vienen en la caja a unos dos metros de altura y orientarlos de modo que apunten el uno hacia el otro, pero con una inclinación de unos 30º- 45º hacia abajo. El espacio de juego debe ser de mínimo de 2 x 1.5 metros, mientras que el tamaño máximo es de 4 x 3 metros.

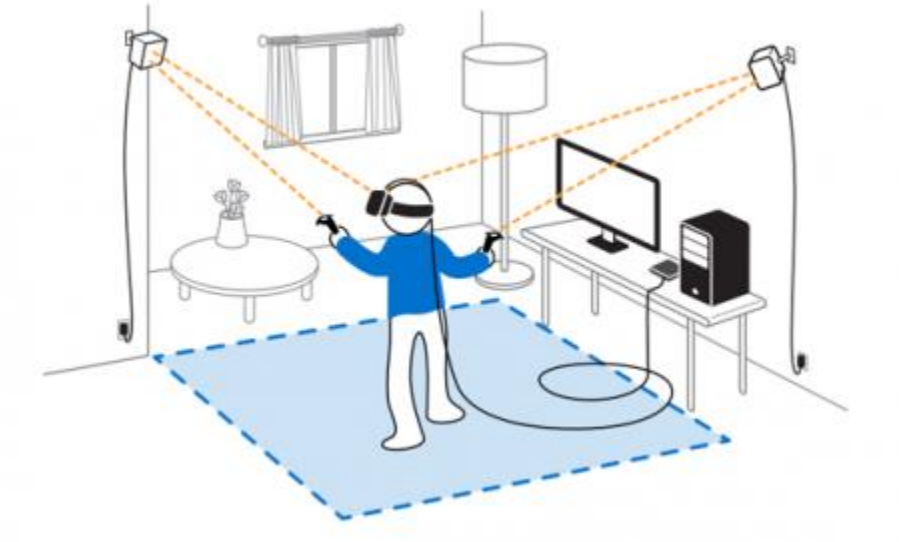


Ilustración 3.33 – Espacio de juego de HTC Vive [13]

En primer lugar, se hizo una instalación rápida y poco precisa para comprobar que el equipo en el que se iba a trabajar cumplía con los requisitos mínimos para hacer funcionar las gafas adecuadamente. Una vez se comprobó que funcionaban correctamente se realizó una mejor instalación de los sensores.

Una vez estaba todo instalado correctamente se decidió crear un nuevo proyecto en Unity para hacer pruebas antes de integrarlo en el proyecto final. Una vez creado el proyecto se instaló el **plugin SteamVR**, mencionado en el apartado 2.6.7. Este plugin se encuentra disponible para descargar en la *Unity Asset Store.* Una vez descargado se importó a través del *Package Manager* de Unity. Al importarlo, este se encarga automáticamente de realizar todos los cambios necesarios en la configuración del proyecto para integrar la VR. Además, este plugin incluye una serie

[13] https://www.researchgate.net/figure/A-representation-of-player-with-Vive-inside-the-play-area-21_fig3_326434680

de *prefabs* y *scripts* que facilitan el desarrollo en VR. Algunos de los más importantes se describen brevemente a continuación:

- **Player [Prefab]**: Este prefab permite crear una instancia funcional de jugador simplemente arrastrándolo a la escena.
- **Interactuable [Script]**: Este script sirve para dotar al objeto al que se asigna de una serie de propiedades que permiten al jugador interactuar con el objeto. También permite customizar la interacción añadiendo acciones a eventos o diferenciando por ejemplo entre objeto arrojable o no arrojable.
- **SteamVR_LaserPointer [Script]**: Este script permite activar un puntero laser que da la posibilidad al jugador de interactuar con la interfaz de usuario.
- **Teleporting [Prefab]**: Permite al jugador activar la teletransportación pulsando la tecla definida para ello. Cuando el jugador pulsa el botón se genera un rayo desde la mano del jugador para apuntar hacia un punto de teletransporte habilitado.
- **TeleportPoint [Prefab]**: Permite arrastrar a la escena puntos de teletransporte a los que el jugador podrá teletransportarse.

Se utilizó una de las escenas de prueba que incluye el *plugin* para probar el funcionamiento. Esta escena se trata de un pequeño plano en el que hay una instancia del jugador rodeada de elementos interactivos que permiten probar los distintos tipos de interacción que ofrece SteamVR.

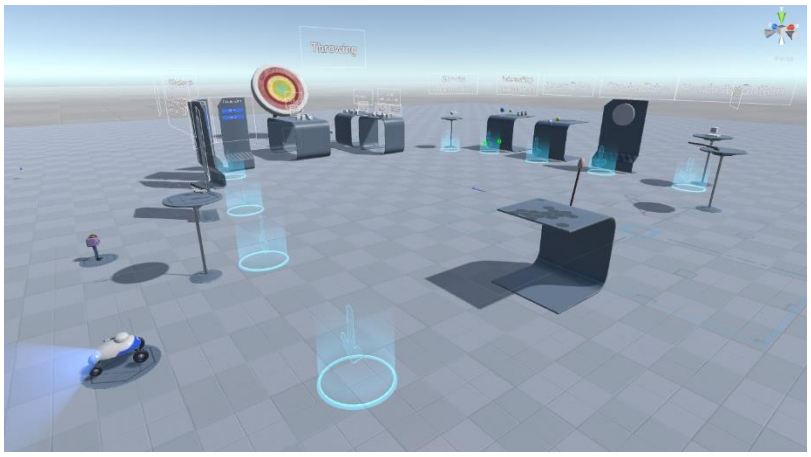


Ilustración 3.34 - Escena de prueba de interacción de SteamVR

Tras realizar algunas pruebas se tomó la decisión de integrar el *plugin* en el proyecto principal.

3.2.2.2 Interacción del jugador en un escenario VR

Antes de realizar la integración de la realidad virtual en el proyecto principal se realizó una copia de seguridad del proyecto.

Para la incorporación de la VR en el proyecto principal se realizaron los mismos pasos que en el proyecto de prueba. Una vez importado el *plugin* el primer paso fue arrastrar el prefab **Player** a la escena. Después se eliminaron algunas plantas y efectos para que el mapa tuviera menos nivel de detalle, ya que el rendimiento no era el óptimo. Lo siguiente fue añadir a la escena el prefab **Teleporting** y añadir algunos **TeleportPoint** para que el jugador tuviera la opción de teletransportarse.

Ilustración 3.35 - Punto de teletransporte (TeleportPoint)

Tras hacer alguna prueba se vio conveniente aumentar el tamaño del prefab **Player** porque el mapa se quedaba demasiado grande.

A continuación, se decidió implementar el movimiento del jugador. Antes de entrar en detalle es necesario hacer una breve presentación de lo que ofrecen los controladores de HTC Vive. Se usará la numeración de la *ilustración 3.36* para facilitar la explicación. Cada mando tiene 5 botones accionables.

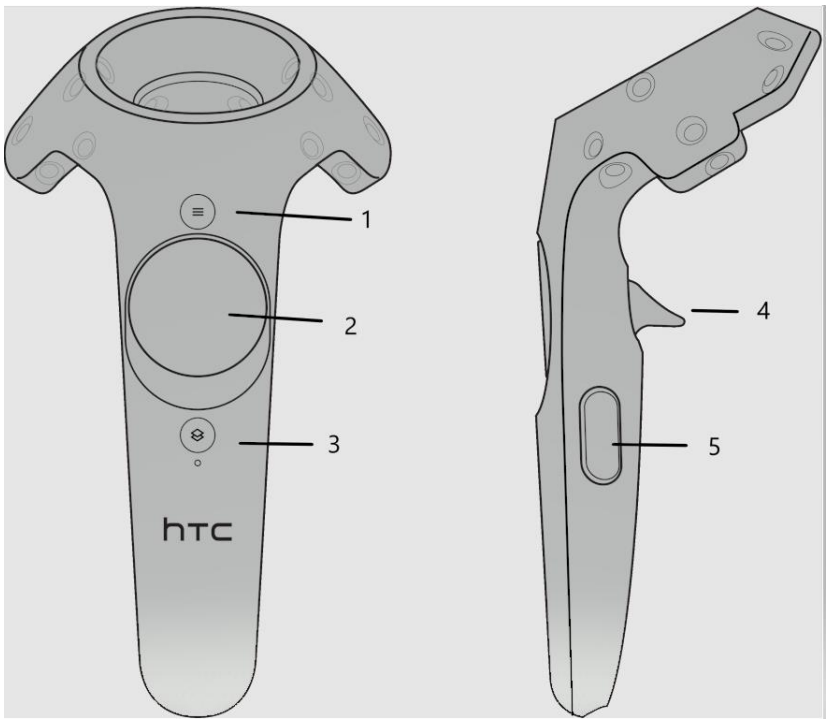


Ilustración 3.36 - Botones HTC Vive controller

1. Botón Menú: Se suele asignar al menú propio de cada juego.
2. Touchpad: Actúa como botón y como panel táctil. Detecta el movimiento del dedo del usuario por él.
3. System: Activa la interfaz de SteamVR.
4. Gatillo: Se suele utilizar para coger objetos o disparar.
5. Empuñadura: Se puede utilizar para coger objetos también.

Una vez explicados los controles configurables se puede proceder con la explicación.

El objetivo principal era conseguir que el jugador pudiera moverse usando el touchpad del HTC Vive Controller ya que esto no está implementado por defecto. Para modificar el input es necesario acceder a una ventana que aparece al instalar el plugin llamada SteamVR Input. Esta ventana permite crear nuevas acciones y modificar la asignación de los controles en el juego.

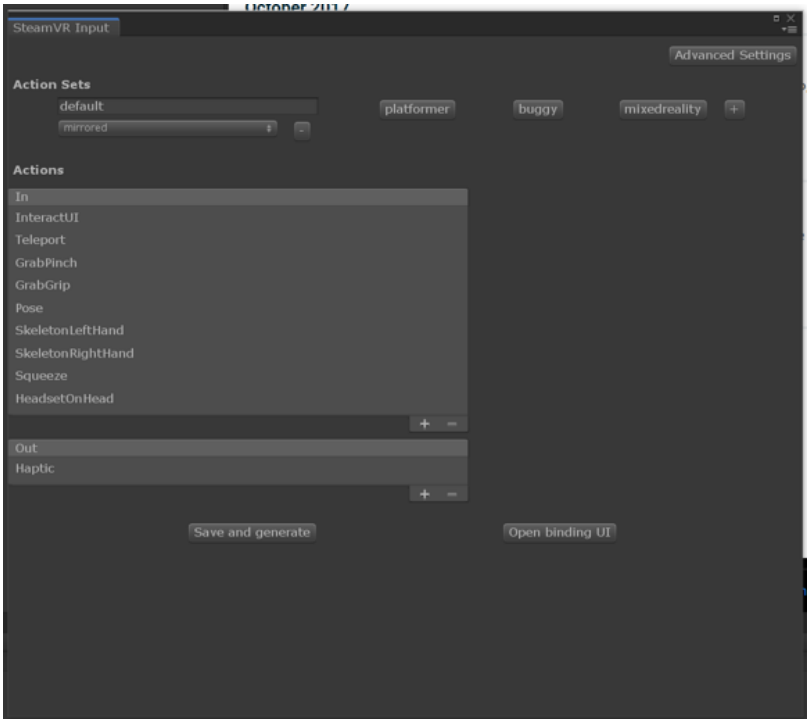


Ilustración 3.37 - Ventana de SteamVR Input

La secuencia de pasos fue la siguiente:

1. Abrir la pestaña de SteamVR Input y crear una nueva acción.

2. Introducir el nombre y tipo de la nueva acción. Entre los tipos disponibles se encuentran (Boolean, Vector2, Single, Vector3, Pose, Skeleton) en este caso el tipo es vector2 ya que lo que se necesita es saber la posición del dedo del jugador en el touchpad, es decir dos valores, x e y.
3. Guardar y generar la acción
4. Abrir la pestaña “Open binding UI”. Una vez se abre esta pestaña, es posible modificar la asignación de controles por defecto o crear una nueva. En este caso se modificó la asignación por defecto.
5. Asignar la nueva acción al botón correspondiente, en este caso al touchpad izquierdo.
6. Crear un nuevo script llamado **PlayerController** en el que se defina el comportamiento de la acción creada anteriormente.

Tras esto el jugador ya estaba instanciado en el juego y con la posibilidad de moverse libremente por el mapa.

El próximo paso fue **introducir en la escena el arma principal** del juego, el arco. El *plugin* SteamVR incluye un *prefab* de un arco, se decidió utilizar este debido a su buen funcionamiento. Se pensó en la posibilidad de instanciar directamente el arco en las manos del jugador, pero debido a la complejidad que esto suponía y a la poca libertad que esto daría al jugador se decidió instanciar el objeto en una zona al comienzo del juego para que fuese el jugador el que cogiese el arco. Cuando el jugador coge el arco, automáticamente se instancia una flecha en la mano contraria a la del arco. Cuando se dispara se instancia otra flecha en la mano del jugador, es decir, hay flechas infinitas.

El jugador podía entonces moverse o teletransportarse, pero aun así el mapa a veces se hacía grande. Por eso se decidió implementar una nueva acción que permitiera **correr** al jugador. Esta simplemente aumentaría la velocidad de movimiento del jugador mientras este pulsara un botón. Se realizó la secuencia de pasos anterior para agregar una nueva acción y se asignó a la pulsación del touchpad izquierdo.

A continuación, se decidió indagar en **la interacción con los NPC**. Tras pensarlo detenidamente se llegó a la conclusión de que lo mejor era crear paneles

con los diálogos de los NPC y que estos se activaran cuando el jugador entrase en un rango. De este modo cuando el jugador entrase en el rango se aparecería el diálogo y se reproduciría algún sonido.

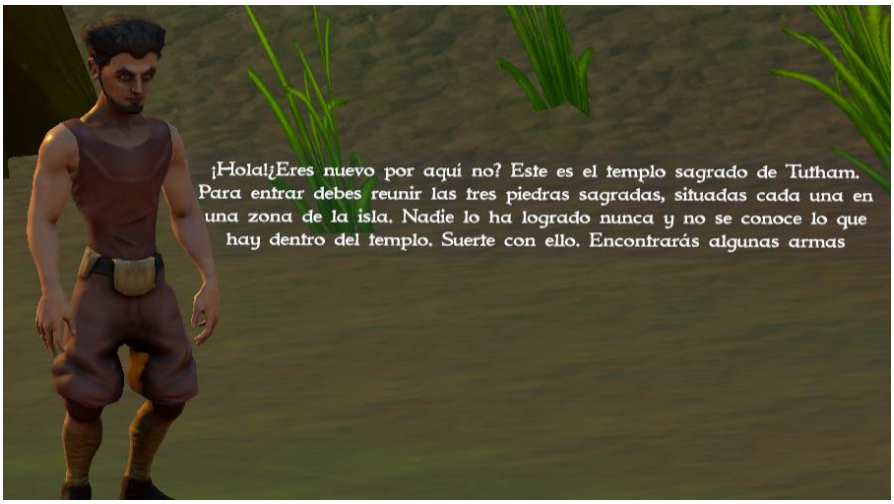


Ilustración 3.38 - Ejemplo de NPC con panel de diálogo

La **interacción con los enemigos** no cambió mucho con respecto al juego base. Se les asigno una etiqueta concreta a armas que sería la que se comprobaría al entrar en contacto con el collider de los enemigos. De este modo si se dispara a un enemigo o se le golpea con la espada este pierde puntos de vida, se instancian unas partículas y se reproduce un sonido que simula que el enemigo ha sido dañado.

Lo próximo que se realizó fue la **implementación de un menú radial** (*ilustración 3.40*) para la selección de arma, ya que la idea era introducir al menos otra arma en el juego. Este se trataría de un círculo dividido en cuatro secciones. Primero se implementó una versión en el proyecto de prueba para comprobar su funcionamiento. El principal problema que surgió no era el de crear el menú en sí, sino el de cambiar de arma.

Por cómo funciona las manos en SteamVR, el *player* tiene asociado por defecto las dos manos, derecha e izquierda incluidas en el prefab de **Player**. Cuando el jugador interactúa con algún objeto, como por ejemplo el arco, se enlaza la mano con el objeto y no es fácil deshacer ese enlace desde el código, además, el

comportamiento del arco es en sí bastante complejo ya que incluye también que la otra mano se enlace con una flecha. Se intentaron las siguientes soluciones:

1. Replicar el estado del Player con el arco de modo que cuando se seleccionara el arco en el menú radial el estado del Player pasase a ser "Player con arco"
2. Crear más instancias de manos asociadas al jugador que se activan y desactivan. El jugador, una vez en el juego, cogerá manualmente un arma y cambiará a un set de manos diferente para coger la siguiente arma. De este modo una vez haya cogido todas las armas podrá intercambiar entre los distintos sets de manos y de este modo cambiar de arma sin problema.
3. Instanciar el arco en la mano del jugador nada más iniciar la partida.

Finalmente se optó por la segunda solución, ya que, tras la primera, que no funcionó, se consideró la más lógica.

Para la implementación del menú radial se siguió una secuencia parecida a la del movimiento del jugador. Fue necesario crear una nueva acción y asociarla en este caso al touchpad derecho. El menú funciona como un *canvas* que está asociado a la cámara, es decir, se mueve con la vista del jugador, que se activa cuando el jugador pulsa el *touchpad* derecho. Manteniendo el botón pulsado el jugador puede deslizar su dedo para elegir el arma. Al soltar se escogerá el arma seleccionada. En la *ilustración 3.39* se muestra un diagrama de secuencia de lo que ocurre cuando el jugador activa el menú radial.

Cuando el jugador pulsa el *touchpad* del controlador derecho se ejecuta la función **Touch()** a través de la cual se invoca la función **Show()** que muestra el menú radial cuando el jugador pulsa el botón y lo oculta cuando el jugador lo suelta. Mientras el menú esta activo se ejecuta la función **Position()** que se encarga de llamar a la función **SetTouchPosition()** pasándole la posición del dedo del jugador en el *touchpad* en todo momento. A continuación, se realizan una serie de operaciones en la clase **RadialMenu** que se encargan de transformar esa posición en un valor numérico entre 0 y 360 grados, debido a que el *touchpad* tiene una forma circular. De este modo se obtiene la sección, de las cuatro que tiene el menú radial, que está seleccionando el jugador. Una vez el jugador suelta el botón se ejecuta la función **ActivateHighlightedSection()** que invoca la función asociada a la sección

seleccionada, que en este caso es cambiar al arma seleccionada, y se oculta el menú radial.

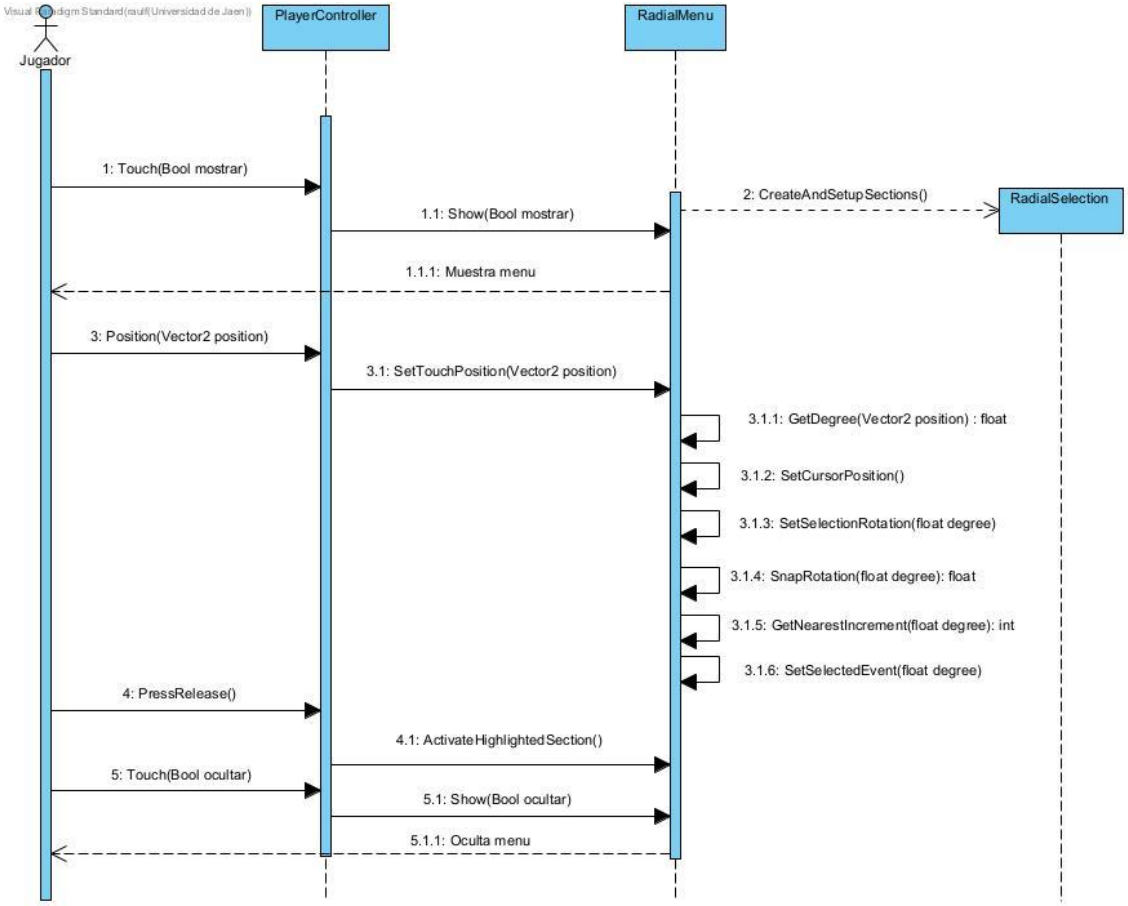


Ilustración 3.39 - Diagrama de secuencia de cambiar arma

Ilustración 3.40 - Menú radial

Finalmente se introdujo un modelo de espada descargado de la *Unity Asset Store* y se colocó en la escena. Para que el juego lo detectara como un objecto interactivo se le asigno el *script* correspondiente. A continuación, fue necesario ajustar la posición de la *mesh* de la espada para que cuando el jugador interactuara con ella esta se ajustara a la posición de la mano y quedara cogida de una forma realista.

Ilustración 3.41 - Imagen de la espada en la mano del jugador

Seguidamente se continuó con la **implementación del cambio de escena**. En un principio, tanto la parte de la cueva como el templo estaban en escenas diferentes, sin embargo, finalmente esto no fue así.

Se creó un script llamado **DontDestroyOnLoad** cuyo objetivo era el de no destruir los objetos a los que se asignara. En este caso se asignó a los objetos que controlan la lógica del juego. El prefab **Player** ya viene con algo parecido incorporado. Uno de los *scripts* que incorpora da la posibilidad de activar o desactivar la opción de que el personaje se destruya al cargar una escena. Esta opción por defecto está desactivada, es decir, al cargar una escena nueva el personaje no se destruye. Esto generaba un problema y es que cuando el jugador muere y se reinicia la escena principal, o este vuelve de otra escena, hay dos instancias del jugador en la escena.

Esto se intentó solucionar eliminando desde un *script* una de las dos instancias cuando se encontraban más de una, pero no funcionaba correctamente. Por lo tanto, se tomaron dos decisiones para solucionar este problema:

- Los elementos de la escena de la cueva pasarían a estar en la escena principal. De este modo cuando el jugador entrara en la cueva se modificaría su posición a la posición inicial de la cueva. Cuando el jugador estuviera en la cueva se desactivarían todos los elementos de la isla y viceversa, con el objetivo de mejorar el rendimiento.
- Cuando el jugador muriera no se recargaría la escena. Simplemente se teletransportaría al jugador a la zona inicial y se le repondría la vida. De este modo tampoco perdería progreso.

Una vez solucionado el cambio de escena se comenzó **a trabajar en la escena del interior del templo**. Para el interior del templo la idea era que el jugador tuviera que resolver una serie de puzles para poder avanzar. Para el diseño se utilizó como base una escena de ejemplo que incluía el propio *asset*. Después se añadieron algunos elementos decorativos y luces. El templo se divide en tres salas.

En la **primera sala** se creó una gran puerta para bloquear el paso del jugador. La idea era crear algún mecanismo con el que el jugador pudiera abrir la puerta. Para ello se usó uno de los elementos que incluye el plugin SteamVR. A través de un *script*, permite enlazar el movimiento de un objeto con algún evento. En este caso se utilizó una rueda que estaría en una pared como el objeto. Entonces se asociaría el movimiento de la rueda sobre el eje z con una animación que mueve el objeto que hace de puerta dando la sensación de que esta se abre hacia arriba.

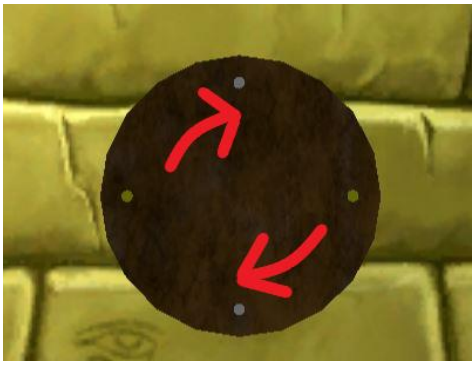


Ilustración 3.43 - Rueda para abrir el portón

Ilustración 3.42 - Portón de la primera sala

Tras esa sala el jugador atraviesa **un pasillo** con otro portón al final. Allí se encuentra un NPC. Este actúa como los demás NPC y muestra un diálogo cuando el jugador entra dentro de un rango. En este caso el diálogo muestra un acertijo que el jugador debe acertar para poder abrir la puerta. Esto se desarrollará en la parte de **interacción por voz**.

En la **última sala** el jugador tiene que realizar una serie de acciones para desbloquear la puerta final. El plugin SteamVR incluye un *script* que permite asociar a un objeto, en este caso un cubo, una animación que simula ser un botón. Además, se pueden realizar eventos que se llaman al pulsar el botón. En este caso se decidió asociar el botón con una animación que mueve una estructura. Esta estructura, al moverse, desbloquea la puerta final. El jugador pulsaría el botón y se desbloquearía la puerta. Se decidió complicar algo más el nivel ya que esto era muy sencillo. Por ello se decidió que el jugador debía hacer algo antes de poder pulsar el botón. Se añadieron tres esferas en la sala a las que el jugador debía disparar (*ilustración 3.44*). Estas esferas cambiarían de color y emitirían un sonido al ser golpeadas por una flecha. Una vez el jugador hubiera disparado a las tres esferas aparecería el botón (*ilustración 3.45*).

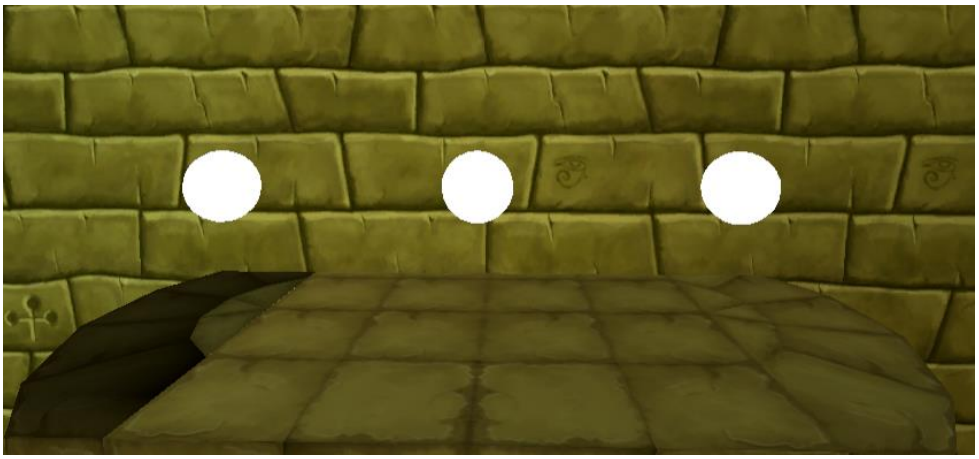


Ilustración 3.44 - Esferas de la tercera sala

Ilustración 3.45 - Botón de la tercera sala

Para ello se estableció un contador que incrementaría al disparar a cada esfera y cuando fuera igual a tres activaría el *GameObject* del botón. Finalmente se creó un *collider* en la puerta final del templo que al entrar en contacto con el jugador cargaría la escena con los créditos finales.

A continuación, se describen otros cambios que se introdujeron en la escena principal.

En la **zona inicial**, en el camino hacia la playa se introdujeron una serie de paneles tutoriales con los que el jugador podría aprender a realizar acciones básicas como coger el arco y disparar o cambiar de arma. Para ello se utilizaron diferentes *canvas*.

Ilustración 3.47 - Panel tutorial sobre los transportadores

Ilustración 3.46 - Panel tutorial sobre el cambio de arma

En la **zona del volcán** se creyó conveniente realizar otro camino que acortase la distancia a recorrer por el jugador, para ello se utilizó la herramienta de modelado de terreno de Unity y se hizo un camino que atraviesa el lago de lava. Además, se incluyeron puntos de teletransporte para ofrecer al jugador la opción de desplazarse usando el teletransporte.

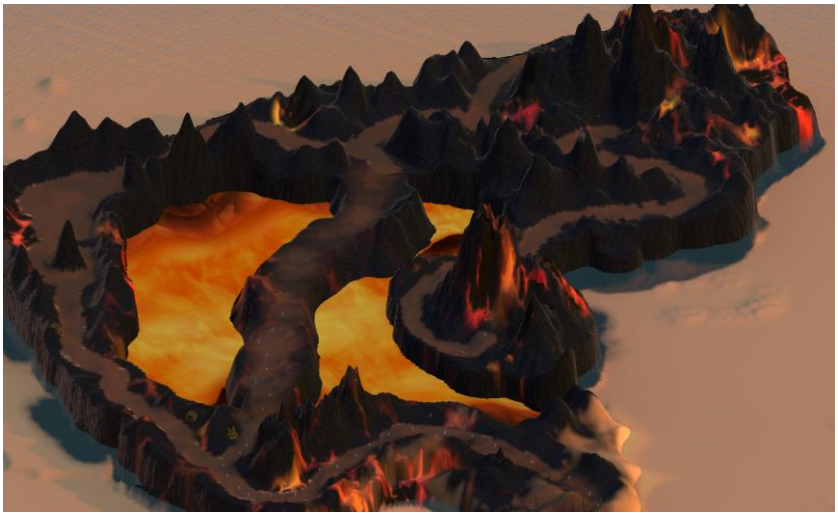


Ilustración 3.48 - Zona del volcán con el nuevo camino

Además, se le asignó al ojo del murciélago un *script* para que el jugador pudiera interactuar con él cogiéndolo con la mano.

Ilustración 3.49 - Ojo del murciélago

Por último, se colocaron una serie de *colliders* a lo largo del escenario para definir los límites del mapa tanto del jugador como de los enemigos.

3.2.3 Resultados

Tras este incremento se obtuvo una versión jugable en VR, pero sin una interfaz de usuario apropiada y sin interacción multimodal. No obstante, el jugador podría completar el juego de principio a fin salvo por la zona del acertijo en el templo. En la *ilustración 3.50* se observa un diagrama de clases del sistema al terminar el incremento. En este se puede observar cómo se han añadido nuevos elementos con respecto al diagrama obtenido del primer incremento. Se ha añadido una clase **PlayerController** encargada de gestionar el input del jugador y de llevar a cabo los eventos relacionados con el movimiento del jugador y su muerte. Esta clase se relaciona también con la clase **RadialMenu** que se encarga de gestionar los eventos relacionados con el menú radial. También se ha añadido el paquete **SteamVR** que agrupa algunas de las clases que proporciona el plugin SteamVR y que gestionan la VR. Así mismo, se han añadido clases como **CaballeroNPC**, **EsferaTemplo** y **ButtonAction** que gestionan los eventos relacionados con los puzles del interior del templo. La clase **WeaponChanger** define las funciones encargadas de cambiar de arma.

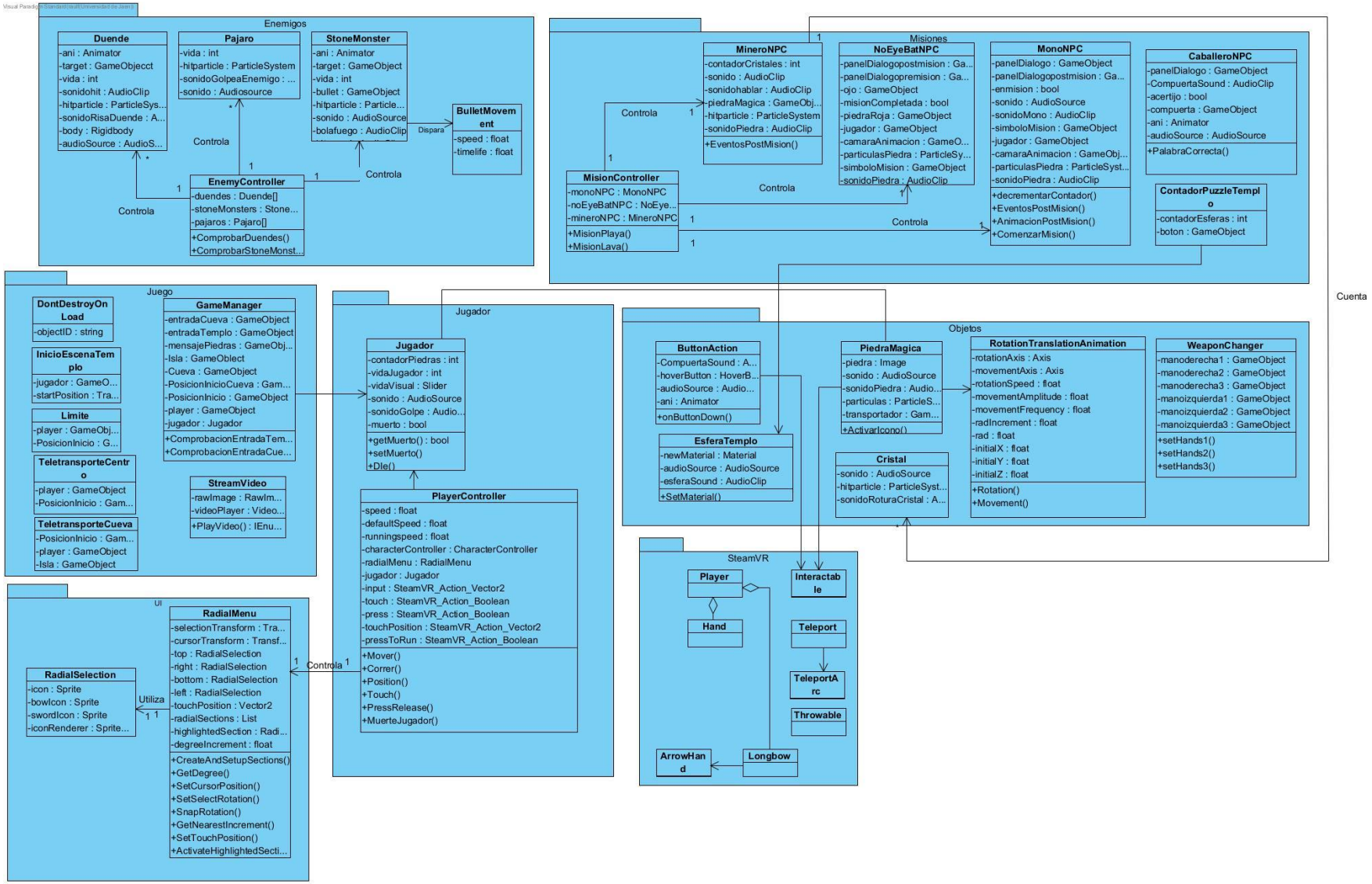


Ilustración 3.50 - Diagrama de clases del segundo incremento

3.2.4 Pruebas

ID y Nombre	PR-05 Test de control de la cámara
Resultado esperado	El usuario podrá controlar la cámara con el movimiento de su cabeza
Resultado obtenido	El usuario puede mover correctamente la cámara con el movimiento de su cabeza
Observaciones	

ID y Nombre	PR-06 Test de movimiento del personaje
Resultado esperado	El usuario podrá controlar su personaje mediante los mandos
Resultado obtenido	El usuario puede mover correctamente a su personaje utilizando los controles asignados
Observaciones	El movimiento es demasiado suave, no da sensación real de estar caminando

ID y Nombre	PR-07 Test de Sprint
Resultado esperado	El usuario podrá aumentar la velocidad de movimiento de su personaje accionando un botón
Resultado obtenido	El usuario puede hacer esprintar al personaje correctamente
Observaciones	

ID y Nombre	PR-08 Test de interacción con objetos
Resultado esperado	El usuario podrá interactuar con unos objetos específicos
Resultado obtenido	El usuario puede interactuar con los objetos especificados
Observaciones	

ID y Nombre	PR-09 Test de interacción con el arco
Resultado esperado	El usuario podrá interactuar con el arco y disparar
Resultado obtenido	El usuario puede interactuar el arco y disparar correctamente
Observaciones	

ID y Nombre	PR-10 Test de interacción con la espada
Resultado esperado	El usuario podrá interactuar con la espada y esta se asociará al lugar correcto de la mano del jugador
Resultado obtenido	El usuario puede interactuar la espada y esta se sitúa en un lugar correcto
Observaciones	

ID y Nombre	PR-11 Test de ataque
Resultado esperado	El usuario podrá atacar a los enemigos y estos reaccionarán de forma natural de acuerdo a las físicas
Resultado obtenido	El usuario puede atacar a los enemigos, además los enemigos emiten unas partículas y un sonido.
Observaciones	La reacción de los enemigos no es natural de acuerdo con las físicas.

ID y Nombre	PR-12 Test de muerte
Resultado esperado	El usuario podrá morir en la partida
Resultado obtenido	El usuario puede morir en la partida
Observaciones	

ID y Nombre	PR-13 Test de diálogos
Resultado esperado	Los diálogos aparecerán cuando el jugador entre dentro del rango del NPC
Resultado obtenido	Los diálogos aparecen cuando el jugador entra dentro del rango del NPC correspondiente
Observaciones	

ID y Nombre	PR-14 Test de teletransporte
Resultado esperado	El jugador podrá teletransportarse usando los mandos
Resultado obtenido	El jugador puede teletransportarse correctamente usando los mandos
Observaciones	En ocasiones tarda un tiempo excesivo en reaccionar

ID y Nombre	PR-15 Test del menú radial
Resultado esperado	El jugador podrá cambiar de arma usando el menú radial

Resultado obtenido	El jugador puede cambiar de arma usando el menú radial
Observaciones	A veces al cambiar de arma vibran los mandos excesivamente

3.3 Incremento 3. Interacción multimodal.

Este incremento se realizó desde el 01/08/2022 hasta el 01/09/2022 es decir, tuvo una duración de **un mes**. En él se llevó a cabo la búsqueda de una herramienta para el reconocimiento de voz y se implementaron la interfaz de usuario en VR y la interacción por voz.

3.3.1 Requisitos seleccionados/ Requisitos a implementar

Los **requisitos** seleccionados en este incremento son los siguientes:

- FRQ-007. Interacción por voz
- FRQ-008. Menú de pausa
- FRQ-009. Información útil
- FRQ-013. Tutorial
- FRQ-016. Salir juego
- FRQ-018. Reanudar partida
- FRQ-020. Sonido
- FRQ-011. Cambio de arma por voz
- NFRQ-001. Rendimiento
- NFRQ-002. Portabilidad

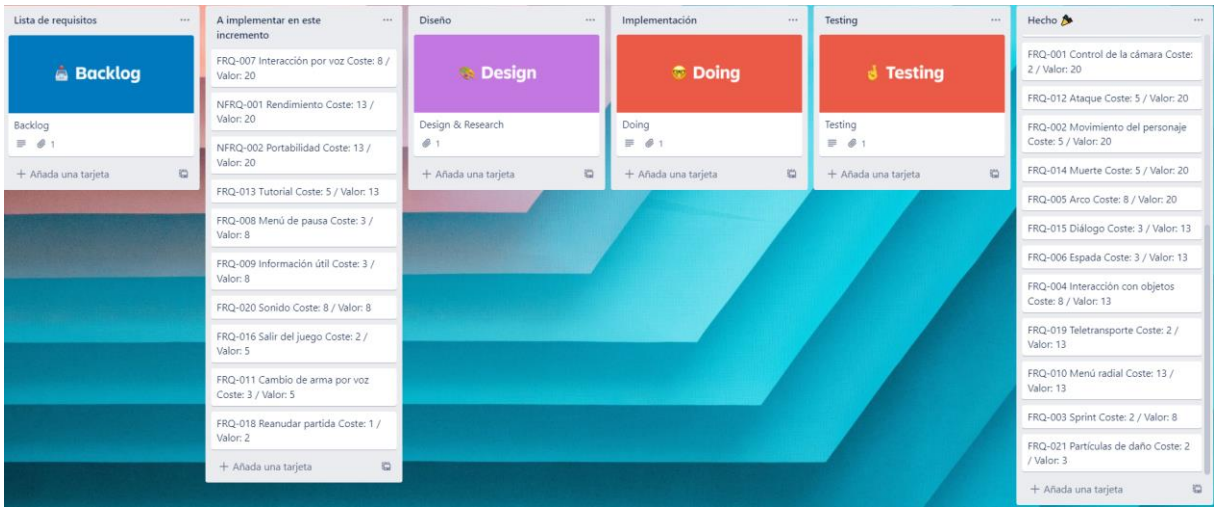


Ilustración 3.51 - Tablero Kanban al comienzo del incremento

3.3.2 Desarrollo

3.3.2.1 Interfaz VR

La realidad virtual plantea un problema adicional y es la interacción del usuario con la interfaz. Para ello se diseñaron unos paneles mostrados en el apartado 2.5 , que se pensaron para que el usuario los pudiera activar de alguna manera e interactuar con ellos usando la voz o los controladores.

En primer lugar, se vio conveniente la creación de un **menú de pausa** para que el jugador pudiera parar el juego y salir de él. El menú consistiría en un *canvas* situado a una distancia adecuada de la vista del jugador con las opciones de reanudar, ajustes y salir. Para ello se instanció un *canvas* en la escena, se colocó en una posición correcta y se situó en la jerarquía de Unity como hijo de la cámara de VR. De este modo este *canvas* se movería con el movimiento de la cabeza del usuario. Al *canvas* se le añadió un panel y a este tres botones, para cada una de las tres opciones. Solo faltaba gestionar la interacción del usuario con el panel, para lo que se hizo lo siguiente:

1. Se creó una nueva acción en la ventana de SteamVR Input y se siguió la secuencia de pasos mostrada en la sección anterior para asignar la activación del menú a un botón de uno de los HTC Vive *controllers*.
2. El script **PlayerController** sería el encargado de llamar a la clase **PlayerUI** para activar el panel de menú cuando el usuario pulsara el botón asignado.

3. Se añadió el *script* **SteamVRLaserPointer** a cada una de las manos del Player. Este *script* se encarga de generar un rayo desde la mano del personaje que permite la interacción con la interfaz.
4. Para poder interactuar con los botones fue necesario asignarles *colliders* para que el puntero laser pudiera detectarlos.
5. En un script llamado **LaserPointerController** se añadió código que se encargaría de detectar las manos del usuario y activar los láseres cuando se activara el menú. También se encargaría de desactivarlos cuando el menú se cerrase.
6. Se asociaron las acciones correspondientes a los botones del panel, en este caso el botón “Reanudar” cerraría el panel, el botón “Salir” cerraría la aplicación y el botón “Ajustes” abriría el panel de ajustes.

El panel de ajustes daría la opción al usuario de bajar o subir el volumen de la música principal del juego. Para ello se creó un deslizador cuyo valor se asociaría al volumen de la música.

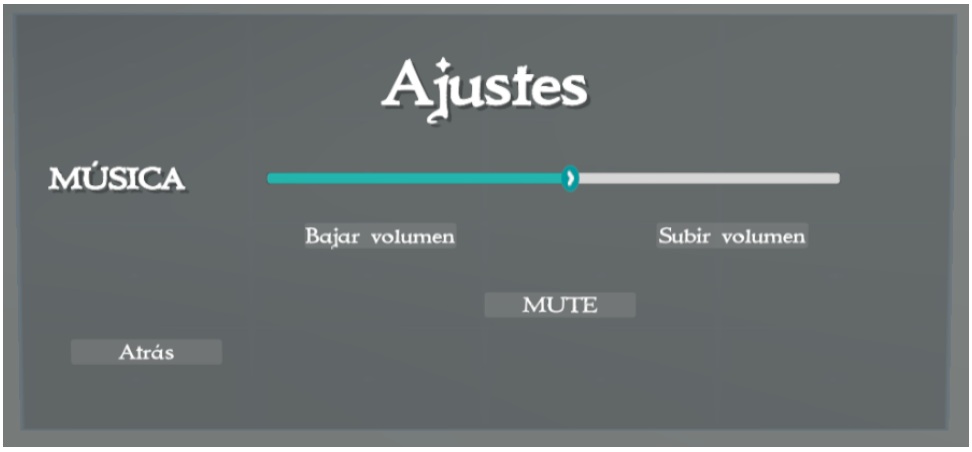


Ilustración 3.52 - Panel de ajustes

También se creyó necesario crear una serie de **paneles al comienzo del juego** que sirvieran a modo de **tutorial** para enseñar al usuario los controles básicos. Se siguió un procedimiento similar al del menú principal, solo que estos paneles aparecerían directamente al inicio del juego.

El primer panel da la opción al usuario de hacer o no el tutorial. Si el usuario pulsa en el botón “No” directamente se desactiva el panel. Si pulsa el botón “Si” este

panel se desactiva y se activa el siguiente panel. A partir de este panel solo hay un botón de “Continuar”. Una vez pasados todos los paneles se ha completado el tutorial. La gestión de activación y desactivación de estos paneles se ha controlado desde el mismo *script* **TutorialUIManager**.

Ilustración 3.53 – Ejemplo de panel tutorial 1

Ilustración 3.54 - Ejemplo de panel tutorial 2

Se crearon otros paneles de menor importancia pero que facilitarían la experiencia de juego del jugador.

- Un panel informativo (*ilustración 3.55*) inspirado en el esbozado en el apartado de diseño que el jugador podría activar pulsando un botón y que mostraría información del juego como la vida, las piedras conseguidas, el tiempo de juego y un mini mapa.
- Un panel que se activaría durante la misión de la playa e informaría al jugador sobre el número de enemigos restantes.

Ilustración 3.55 - Panel de información de juego

3.3.2.2 Interacción por voz

En un primer lugar, la idea era buscar una herramienta compatible con Unity que permitiera el reconocimiento de voz y el procesamiento del lenguaje natural para tratar de analizar el discurso del jugador y sacar conclusiones sobre la jugabilidad del videojuego. Por tanto, se procedió a la búsqueda de una herramienta que proporcionara ambas cosas. Se trata de una tecnología todavía en vías de desarrollo lo que se tradujo en una búsqueda poco fructífera. Se tuvieron en cuenta las siguientes herramientas:

- **Unity NLP**: es una colección de herramientas de procesamiento de lenguaje natural escritas en C# que está dirigida al motor Unity.
- **Windows Speech API**: es una API creada por Microsoft que ofrece herramientas para el reconocimiento de voz y el procesamiento del discurso.

Tras probar ambas herramientas en proyectos de prueba se decidió escoger la API de Windows por los siguientes motivos:

- Cuenta con una buena documentación en la propia página de Unity
- Permite el reconocimiento de voz
- La herramienta Unity NLP dio problemas en sus pruebas
- No se necesita importar ningún paquete, solo usar una librería en los scripts.

Además, esta API permitía tanto el reconocimiento del discurso, es decir de todo lo que el usuario dice, como el reconocimiento de palabras clave, para la interacción por comandos de voz, por ejemplo. Para probar la herramienta se creó un

proyecto de prueba en el que se instanció un *canvas* con un texto que transcribiría en tiempo real las palabras del usuario. Básicamente se trataba de un script que se encargaba de tomar como entrada el audio del micrófono y transcribir a un texto que se mostraba en pantalla lo que entendía la máquina. De hecho, el *script* mostraba dos textos:

- Hipótesis: Mostraba todo lo reconocido por la máquina.
- Resultado: Mostraba solo aquello se consideraba correcto y coherente.

El funcionamiento era bastante bueno, pero fallaba cuando no se hablaba con claridad.

Una vez se probó lo suficiente se introdujo en el proyecto, en realidad solo era necesario crear un script que importara la librería **UnityEngine.Windows.Speech.** Tras esto se realizaron diferentes pruebas. Una de ellas consistía en que cuando se reconociera la palabra “Hola” y el jugador estuviera cerca del NPC del inicio del juego este se acercaría. Sin embargo, comenzaron a **surgir problemas con el reconocedor del discurso**.

- Este tenía asignado un tiempo límite, tras el cual dejaba de escuchar hasta que se reiniciara el juego.
- Se detenía cuando la ejecución del juego en el ordenador pasaba en algún momento a estar en segundo plano.
- No se permitía ejecutar al mismo tiempo el reconocedor del discurso y el reconocedor de palabras clave

Por ese motivo y tras consultarlo con los tutores se tomó la decisión de **utilizar solo el reconocedor de palabras clave**. Surgió la idea de usar comandos de voz durante el juego que permitieran al jugador realizar diversas acciones y además almacenar estas palabras con el objetivo de en un futuro poder evaluar la satisfacción del jugador a partir de palabras clave recogidas en un corpus.

Se implementaron una serie de comandos de voz que el jugador podría usar durante el juego. Algunos comandos realizarían una acción u otra dependiendo del estado del juego en el momento en el que el jugador dice la palabra.

Comando	Precondición	Acción
“Menú”	No tener activado el menú de pausa	Activa el panel de menú de pausa
“Salir”	Tener activado el menú de pausa	Cierra la aplicación
“Reanudar”	Tener activado el menú de pausa	Desactiva el panel de menú de pausa
“Si”	Tener activado el primer panel del tutorial inicial	Desactiva el panel y activa el siguiente (Comienza el tutorial)
“No”	Tener activado el primer panel del tutorial inicial	Desactiva el panel del tutorial
“Si”	Tener activado el panel de aceptar misión	Comienza la misión
“Siguiente”	Tener activado alguno de los paneles del tutorial	Desactiva el panel actual y activa el siguiente
“Vela”	Estar en el templo, en la zona donde se encuentra el Caballero.	Abre la puerta a la segunda sala del templo
“Arco”	Tener el arco en el inventario	Equipa al jugador con el arco
“Espada”	Tener la espada en el inventario	Equipa al jugador con la espada
“Puños”	Ninguna	Equipa al jugador con los puños
“Subir volumen”	Ninguna	Sube el volumen de la música
“Bajar volumen”	Ninguna	Baja el volumen de la música del juego
“Ajustes”	Ninguna	Abre el panel de ajustes
“Silenciar”	Ninguna	Silencia el volumen de la música del juego
“Reiniciar”	Estar muerto	Reinicia la partida

Tabla 3.1 - Comandos de voz

Para hacerlo funcionar se creó un diccionario que almacenara los pares (*string*, *Action*). De este modo se asignaba a cada *string* (cada palabra clave) una acción.

```
private Dictionary<string, Action> actions = new Dictionary<string, Action>();
```

Ilustración 3.56 - Fragmento de código de la creación del diccionario

```
void Start()
{
    actions.Add("Arco",cambiadorArma.setHands1);
    actions.Add("Espada",cambiadorArma.setHands2);
    actions.Add("Manos",cambiadorArma.setHands3);
    actions.Add("Menú",menu.Pause);
    actions.Add("Reanudar", menu.Resume);
    actions.Add("Salir",menu.QuitGame);
    actions.Add("Atrás",menu.Resume);
    actions.Add("no",tutorial.CancelarTutorial);
    actions.Add("Vela",CaballeroNPC.PalabraCorrecta);
    actions.Add("Continuar",tutorial.SiguientePanel);

    m_Recognizer = new KeywordRecognizer(m_Keywords);
    m_Recognizer.OnPhraseRecognized += OnPhraseRecognized;
    m_Recognizer.Start();
}
```

Ilustración 3.57 - Fragmento de código de la asignación de las palabras y acciones

Después se invocaba la acción correspondiente con el *string* de entre los almacenados que se reconociera durante la partida.

```
private void OnPhraseRecognized(PhraseRecognizedEventArgs args)
{
    actions[args.text].Invoke();
}
```

Ilustración 3.58 - Fragmento de código de la invocación de las acciones

Para que el jugador pudiera saber cuándo se ha reconocido una palabra correctamente, se decidió implementar **un sistema de partículas que materializase los comandos de voz** en forma de partículas cuando el jugador los pronunciase. De este modo cuando el jugador pronuncia un comando y este es reconocido correctamente se generan unas partículas con la palabra indicada.

3.3.3 Resultados

Al final de este incremento se obtuvo como entregable una versión jugable del videojuego con interacción multimodal. El jugador podría interactuar con el juego y con la interfaz de usuario por medio de los controladores y de la voz. En la *ilustración 3.59* se muestra el diagrama de clases final. Se pueden observar cambios notables con respecto al anterior diagrama como por ejemplo la clase **KeyWordRecognition,** encargada de gestionar todo lo relacionado con la interacción por voz. También se han añadido varias clases al paquete **UI** que controlan los eventos relacionados con la interfaz de usuario. En el paquete **Misiones** se han incorporado las clases responsables de la misión del interior del templo que a su vez están relacionadas con nuevas clases del paquete **Objetos** que representan objetos situados en esta misión final. Otras clases como **SteamVRLaserPointer** o **LaserPointerController** han sido elaboradas para crear y controlar el puntero laser que permite la interacción del jugador con la interfaz.

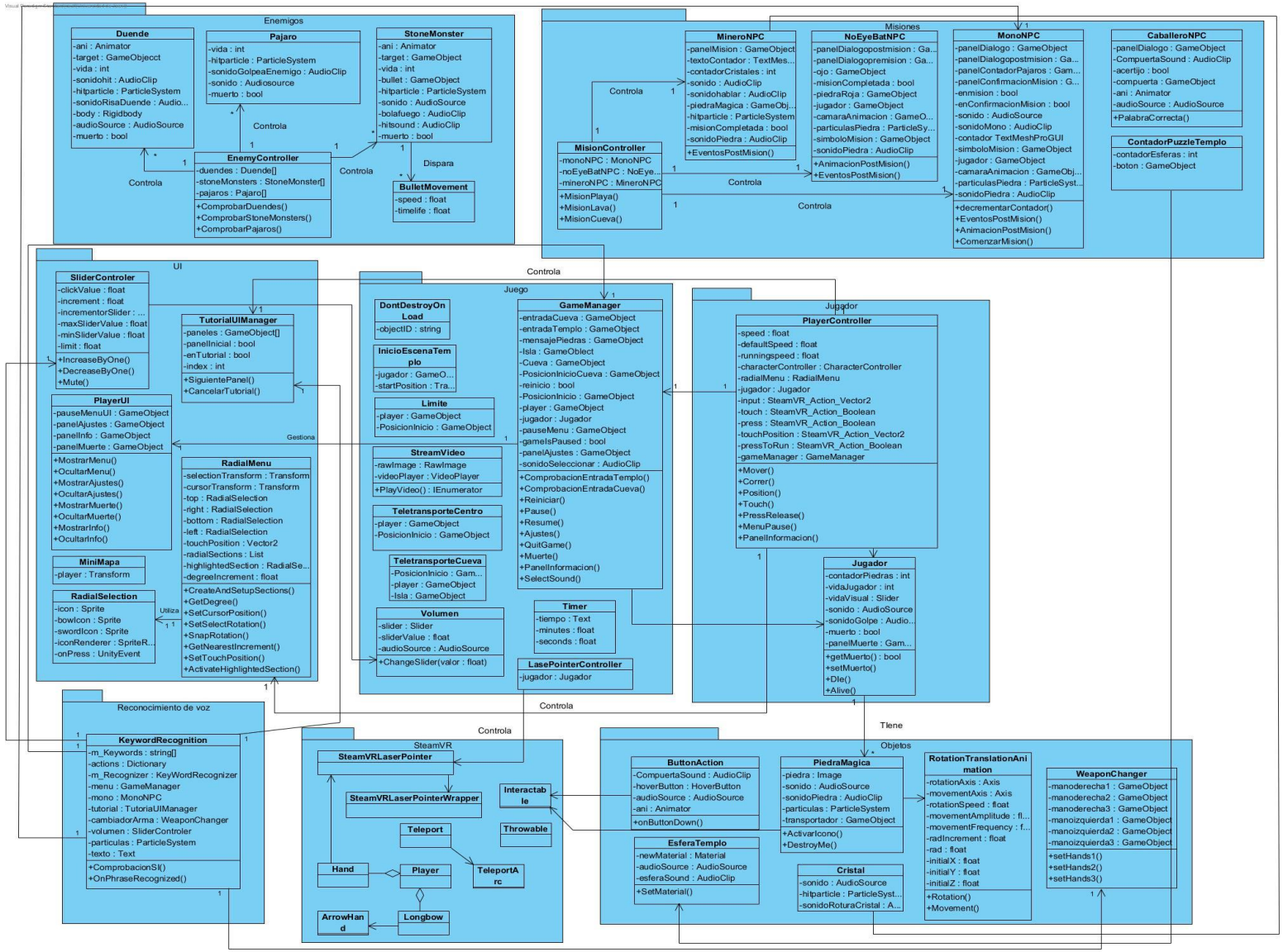


Ilustración 3.59 - Diagrama de clases final

3.3.4 Pruebas

ID y Nombre	PR-16 Test de interacción por voz
Resultado esperado	El usuario podrá activar eventos con comandos de voz
Resultado obtenido	El usuario puede activar eventos con comandos de voz
Observaciones	Las palabras deben pronunciarse con claridad

ID y Nombre	PR-17 Test de menú de pausa
Resultado esperado	El usuario podrá activar un menú de pausa con diferentes opciones e interactuar con él por voz y manualmente
Resultado obtenido	El usuario puede activar un menú de pausa con varias opciones e interactuar con él por voz y manualmente
Observaciones	

ID y Nombre	PR-18 Test de panel de información
Resultado esperado	El usuario podrá acceder a un panel informativo
Resultado obtenido	El usuario puede acceder al panel informativo
Observaciones	

ID y Nombre	PR-19 Test de tutorial
Resultado esperado	El usuario tendrá la posibilidad de hacer un tutorial al comienzo del juego
Resultado obtenido	El usuario puede hacer un tutorial al comienzo del juego
Observaciones	

ID y Nombre	PR-20 Test de salir del juego
Resultado esperado	El usuario podrá salir del juego pulsando el botón correspondiente en el menú de pausa o por comando de voz
Resultado obtenido	El usuario puede salir del juego pulsando el botón de salir del menú de pausa o diciendo salir.
Observaciones	

ID y Nombre	PR-21 Test de reanudar partida

Resultado esperado	El usuario tendrá la posibilidad de reanudar la partida pulsando el botón de reanudar o por comando de voz
Resultado obtenido	El usuario puede reanudar el juego pulsando el botón del menú de pausa o diciendo “reanudar”
Observaciones	

ID y Nombre	PR-22 Test de sonido
Resultado esperado	El usuario tendrá la posibilidad de ajustar el sonido del juego
Resultado obtenido	El usuario puede ajustar el volumen de la música
Observaciones	Solo se podrá cambiar el volumen de la música del juego

3.4 Patrones de diseño utilizados

Un patrón de diseño es una solución general y reutilizable que puede ser aplicada para resolver un problema concreto. En este apartado se expondrán los patrones de diseño software aplicados en el proyecto, así como el lugar donde han sido aplicados.

- **Patrón controlador**: Consiste en crear una clase que de algún modo detecte un evento al que está asociada y realice la acción asociada a dicho evento. En este caso ha sido utilizado por ejemplo en las clases ***EnemyController*, *MissionController* o *PlayerController*.** Estas clases son las responsables de lanzar eventos de acuerdo con el estado de aquellas clases que controlan. Por ejemplo, la clase ***EnemyController*** se encarga de destruir los enemigos que cuando se da cierta condición.
- **Patrón alta cohesión y bajo acoplamiento:** Se ha tratado de realizar un sistema cohesivo, haciendo que cada clase esté totalmente relacionada con los datos u operaciones que controla. Así mismo, se ha mantenido un bajo acoplamiento evitando cargar a una clase con toda la responsabilidad y distribuyendo esta entre distintas clases siendo mínimas las dependencias entre estas.

- **Patrón prototipo:** Este patrón consiste en crear nuevos objetos clonándolos de una instancia ya existente. Ha sido utilizado en las bolas de fuego que dispara el enemigo **StoneMonster.** Se crean múltiples copias de este objeto a partir de una instancia obtenida de un prefab.
- **Patrón peso ligero (*Flyweight*):** La idea de este patrón es la de optimizar el uso de memoria haciendo que varios objetos parecidos compartan datos. De este modo en lugar de tener que guardar una gran cantidad de datos para cada objeto, se guardan solo aquellos datos que son únicos y el resto se comparten. Ha sido utilizado para los árboles y otros elementos decorativos del videojuego. Todos los árboles del mismo tipo comparten textura y malla, lo único que es necesario almacenar es su posición y rotación. Unity se encarga de hacer esto mediante la herramienta de edición de terreno.

3.5 Pruebas en usuarios

Una vez obtenido el último incremento y versión jugable, se tomó la decisión de dar a probar a distintos usuarios, para posteriormente realizar dos encuestas y evaluar aspectos vitales del juego como son la interacción multimodal o la inmersión en el entorno VR.

Para la primera encuesta se estableció una escala del 1 al 5 siendo 1 “muy mal” y 5 “muy bien”.

ID	Pregunta	J1	J2	J3	J4
PRG-01	¿Cómo valorarías la interacción por voz?	5	3	4	5
PRG-02	¿Cómo valorarías la interacción con los enemigos?	2	3	3	3
PRG-03	¿Cómo valorarías la interacción con la interfaz?	4	5	5	5
PRG-04	¿Cómo valorarías la inmersión en el entorno virtual?	3	5	5	5
PRG-05	¿Cómo valorarías el movimiento del personaje?	3	4	4	3
PRG-06	¿Cómo valorarías el apartado sonoro del videojuego?	3	4	5	4
PRG-07	¿Cómo valorarías los controles del videojuego?	4	5	5	4

PRG-08	¿Cómo valorarías el apartado gráfico del juego?	3	4	3	3

Tabla 3.2 - Encuesta 1

La segunda encuesta consistió en cuatro preguntas de respuesta corta:

ID	Pregunta	J1	J2	J3	J4
PRG-07	¿Qué zona del juego has disfrutado más?	Templo	Playa	Mina	Templo
PRG-08	¿Qué zona del juego has disfrutado menos?	Playa	Lava	Lava	Lava
PRG-09	¿Qué arma has disfrutado más?	Arco	Arco	Arco	Arco
PRG-10	¿Has tenido sensación de mareo/desequilibrio durante la experiencia?	No	Si	No	Si

Tabla 3.3 - Encuesta 2

Tras observar detenidamente los resultados se obtuvieron las siguientes conclusiones:

- La interacción por voz funciona adecuadamente y proporciona al usuario un modo atractivo de interactuar con el juego.
- La interacción con los enemigos es simple y rígida, por lo que es un aspecto a mejorar.
- La interacción con la interfaz de usuario funciona correctamente.
- Para la mayoría de usuarios se logra una buena inmersión en el entorno virtual.
- El movimiento del personaje puede ser mejorado.
- El aspecto sonoro del videojuego puede llegar a ser repetitivo.
- Los controles están distribuidos de forma que los jugadores se adecúan fácilmente a su uso.
- En el apartado gráfico se han obtenido puntuaciones bajas denotando que la calidad gráfica es mejorable.
- La zona del videojuego menos disfrutada en general es la “zona de la lava” mientras que para la zona más disfrutada la opinión es más dispersa.
- El arma que más han disfrutado los jugadores es el arco.

- La mitad de los jugadores han sufrido sensación de desequilibrio/mareo durante la experiencia.

4 CONCLUSIONES Y TRABAJOS FUTUROS

(Página intencionalmente en blanco)

En esta sección se presentan las conclusiones obtenidas tras terminar el proyecto, así como las propuestas de mejoras y líneas de trabajo futuras.

Este proyecto ha tenido como objetivo el desarrollo de una experiencia en VR con el extra de proporcionar una interacción multimodal. Tras su ejecución se han obtenido las siguientes conclusiones generales:

1. El desarrollo en VR supone una serie de cambios con respecto al desarrollo de videojuegos convencional para los que se debe estar preparado y bien informado.
2. El plugin empleado para el proyecto cuenta con menos documentación que otros existentes y esto ha podido dificultar el desarrollo.
3. La experiencia en VR debe garantizar un rendimiento óptimo o puede perderse la sensación de inmersión.
4. En el desarrollo de un videojuego son tan importantes las labores del equipo de programación como las del equipo de arte y diseño.
5. La interacción por voz supone una nueva forma de crear videojuegos y experiencias y ayuda a la inmersión del jugador en el mundo virtual. Se le puede sacar mucho partido y sin duda supondrá un antes y un después en el mundo de los videojuegos.

Por otro lado, aquí se listan una serie de funcionalidades cuya implementación ha quedado pendiente de hacer en un futuro:

- La utilización de palabras clave para la estimación, a través del uso de herramientas del procesamiento del lenguaje natural, del nivel de satisfacción del usuario sobre el videojuego. Se plantea utilizar un corpus de sentimientos para clasificar las palabras que dice el jugador mientras juega y determinar a partir de estas su opinión sobre el juego o hacer un análisis de la jugabilidad.
- Mejorar la interacción del jugador con los enemigos, haciendo la interacción más realista de acuerdo con físicas reales.
- Añadir más enemigos y zonas. Desarrollar enemigos con ataques más complejos y otras zonas que explorar.

- Mejorar la calidad de algunas texturas que se ven extrañas en VR y pueden molestar a la vista del usuario.
- Mejorar el diseño de los paneles y del menú. Añadiendo para ello más detalles y creando una interfaz de usuario más atractiva para el usuario.

5 APÉNDICES

(Página intencionalmente en blanco)

En este apartado se incluirá la documentación adicional que pueda resultar de utilidad para comprender correctamente el proyecto.

5.1 Instalación y configuración del sistema

Para ejecutar el videojuego simplemente hay que descomprimir el archivo descargado y ejecutar el archivo con la extensión **.exe** *(ilustración 5.1)*. Es necesario contar con la herramienta SteamVR.

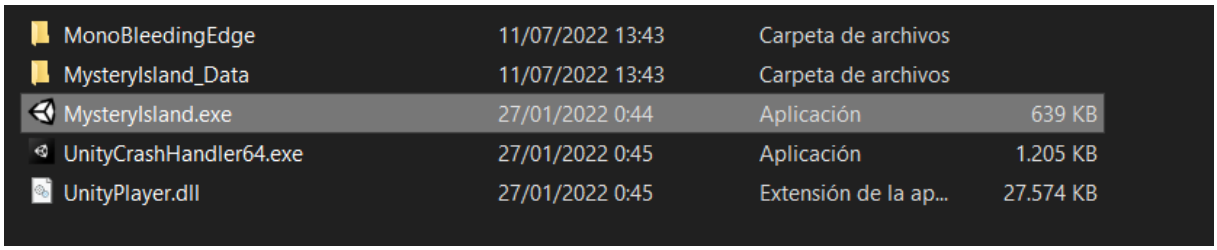


Ilustración 5.1 - Carpeta con el ejecutable del juego

5.2 Manuales de usuario

El videojuego ha sido principalmente desarrollado para ser jugador utilizando los controladores de HTC Vive. A continuación, se muestran los controles por defecto.

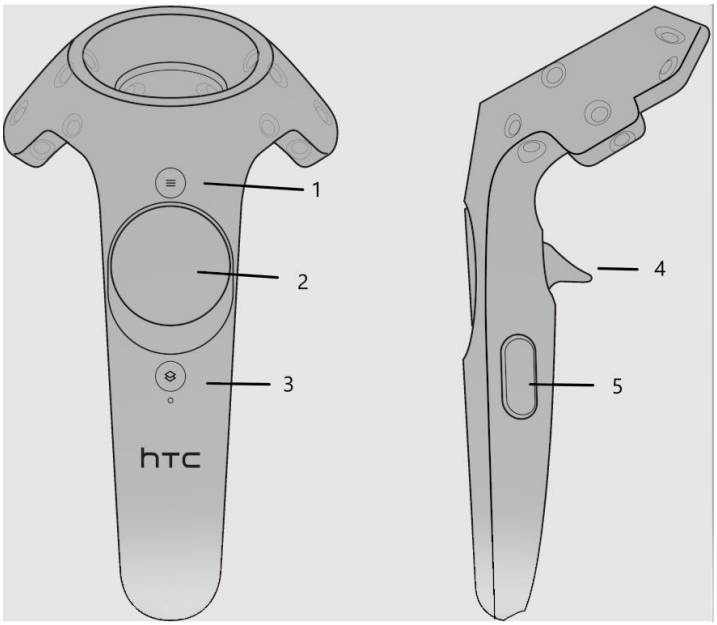


Ilustración 5.2 - Controles HTC Vive

CONTROLADOR IZQUIERDO

1. Menú pause
2. Mover/correr
3. Menú SteamVR
4. Agarrar
5. Teletransportar

CONTROLADOR DERECHO

1. Panel de información
2. Menú radial
3. Menú SteamVR
4. Agarrar
5. Teletransportar

6 DEFINICIONES Y ABREVIATURAS

(Página intencionalmente en blanco)

En esta sección se definirán los términos específicos y abreviaturas necesarios para la correcta comprensión del documento.

6.1 Definiciones

- **Eye tracking**: Es el proceso de evaluar, bien el punto donde se fija la mirada (donde estamos mirando), o el movimiento del ojo en relación con la cabeza.
- **Software development kit (SDK):** Es una colección de herramientas software en un único paquete instalable.
- **Plugin:** Es un software que permite extender las posibilidades de otro software sin modificar el código.
- **Aplication programming interface (API)**: Es un conjunto de funciones, subrutinas y procedimientos que ofrece una biblioteca para ser usada por otro software como una capa de abstracción.
- **Plan de contingencia**: Consiste en estar preparado para lo peor y tener un plan de acción preparado si se materializa un riesgo.
- **Plan de mitigación**: Estrategia orientada a reducir la probabilidad o el impacto del riesgo.
- **Prefab**: Elemento de Unity que permite almacenar un GameObject con todas sus propiedades y componentes como un recurso utilizable.
- **Canvas**: Es un GameObject de Unity que engloba todos los elementos de la UI.
- **GameObject**: Es la unidad básica para construir cualquier cosa en Unity. Puede representar un personaje, un escenario o cualquier elemento que pertenece al juego. Funciona como contenedor de componentes que implementan su funcionalidad.
- **Skybox**: Es un cubo de seis caras dibujado detrás de los gráficos del juego que funciona como el fondo y hace parecer el escenario más profundo de lo que es.

- **Collider**: Componente de Unity que define la forma de un objeto para los propósitos de colisiones físicas.
- **Asset**: Es un ítem que puedes usar en tu proyecto. Puede ser un script, una textura, un modelo 3D, etc.
- **Corrutina**: Es una función que tiene la capacidad de pausarse y reiniciarse exactamente donde se quedó en el frame anterior.
- **Wireframe**: Guía visual que representa el esqueleto o estructura visual de un contenido.
- **Shader**: Los Shaders son Assets que contienen código e instrucciones para que la tarjeta gráfica ejecute.

6.2 Abreviaturas

- **WIP**: Work In Progess.
- **UI**: User Interface.
- **VR**: Virtual Reality.
- **API:** Aplication programming interface.
- **SDK:** Software development kit.
- **FRQ:** Requisito funcional
- **CU:** Caso de uso
- **NFRQ:** Requisito no funcional

7 BIBLIOGRAFÍA

(Página intencionalmente en blanco)

[1] «El videojuego en España», *Asociación Española de Videojuegos*, 11 de diciembre de 2015. http://www.aevi.org.es/la-industria-del-videojuego/en-espana/ (accedido 11 de julio de 2022).
[2] «EL MERCADO DE LOS VIDEOJUEGOS DE REALIDAD VIRTUAL | I AM VR», 29 de agosto de 2021. https://i-amvr.com/el-mercado-de-los-videojuegos-de-realidad-virtual/ (accedido 30 de agosto de 2022).
[3] «Todo lo que tenes que saber sobre la Realidad virtual y Psicología», 22 de julio de 2020. https://www.psicologoscordoba.org/realidad-virtual-y-psicologia/ (accedido 31 de agosto de 2022).
[4] «Las top 7 aplicaciones de realidad virtual para la educación | VeeR VR Blog». https://veer.tv/blog/es/las-top-7-aplicaciones-de-realidad-virtual-para-la-educacion/ (accedido 31 de agosto de 2022).
[5] «iVR, la inmersión VR en arquitectura», *BibLus*, 12 de noviembre de 2019. https://biblus.accasoftware.com/es/ivr-la-inmersion-vr-en-arquitectura/ (accedido 31 de agosto de 2022).
[6] M. El-Yamri, A. Romero-Hernandez, M. Gonzalez-Riojo, y B. Manero, «Designing a VR game for public speaking based on speakers features: a case study», *Smart Learn. Environ.*, vol. 6, n.º 1, p. 12, nov. 2019, doi: 10.1186/s40561-019-0094-1.
[7] D. Martin, S. Malpica, D. Gutierrez, B. Masia, y A. Serrano, «Multimodality in VR: A survey», *ACM Comput. Surv.*, p. 3508361, ene. 2022, doi: 10.1145/3508361.
[8] P. Monteiro, G. Gonçalves, H. Coelho, M. Melo, y M. Bessa, «Hands-free interaction in immersive virtual reality: A systematic review», *IEEE Trans. Vis. Comput. Graph.*, vol. 27, n.º 5, pp. 2702-2713, may 2021, doi: 10.1109/TVCG.2021.3067687.
[9] «¿Qué es la metodología Kanban? | Deloitte España», *Deloitte Spain*. https://www2.deloitte.com/es/es/pages/technology/articles/que-es-metodologia-kanban.html (accedido 7 de julio de 2022).
[10] Asana, «¿Qué es la metodología Kanban y cómo funciona? • Asana», *Asana*. https://asana.com/es/resources/what-is-kanban (accedido 13 de julio de 2022).
[11] «Kanban vs. Scrum», *Deloitte Spain*. https://www2.deloitte.com/es/es/pages/technology/articles/kanban-vs-scrum.html (accedido 13 de julio de 2022).
[12] «Historia de la Realidad Virtual – Xperimenta Cultura». https://xperimentacultura.com/historia-de-la-realidad-virtual/ (accedido 13 de julio de 2022).
[13] «SteamVR Unity Plugin | SteamVR Unity Plugin». https://valvesoftware.github.io/steamvr_unity_plugin/index.html (accedido 19 de julio de 2022).
[14] «El Modelo COCOMO». http://www.sc.ehu.es/jiwdocoj/mmis/cocomo.htm (accedido 30 de agosto de 2022).
[15] thetuvix, «Voice input in Unity - Mixed Reality». https://docs.microsoft.com/en-us/windows/mixed-reality/develop/unity/voice-input-in-unity (accedido 31 de agosto de 2022).
[16] «Game programming patterns in Unity with C# - Command Pattern | Habrador». https://www.habrador.com/tutorials/programming-patterns/1-command-pattern/ (accedido 31 de agosto de 2022).
[17] «Unity Asset Store - The Best Assets for Game Making». https://assetstore.unity.com/ (accedido 31 de agosto de 2022).
[18] U. Technologies, «Unity - Scripting API»: https://docs.unity3d.com/ScriptReference/ (accedido 31 de agosto de 2022).

[19] «Newsfeed», *Sketchfab*. https://sketchfab.com/feed (accedido 31 de agosto de 2022).
[20] «Mixamo». https://www.mixamo.com/#/ (accedido 31 de agosto de 2022).
[21] «Freesound - Freesound». https://freesound.org/ (accedido 31 de agosto de 2022).
[22] «FakeYou, Your Deep Fake Text to Speech Website.» https://fakeyou.com (accedido 31 de agosto de 2022).
[23] «(950) YouTube». https://www.youtube.com/ (accedido 31 de agosto de 2022).
[24] «Sonidos mp3 Gratis. Busqueda y Descargas de ringtones gratis.» http://www.sonidosmp3gratis.com/ (accedido 31 de agosto de 2022).
[25] «Interacción multimodal. Qué es y su aplicación en el ámbito de la enseñanza», *Interacción multimodal. Qué es y su aplicación en el ámbito de la enseñanza*. https://laschicasdelcavle.blogspot.com/2019/02/interaccion-multimodal-que-es-y-su.html (accedido 9 de septiembre de 2022).