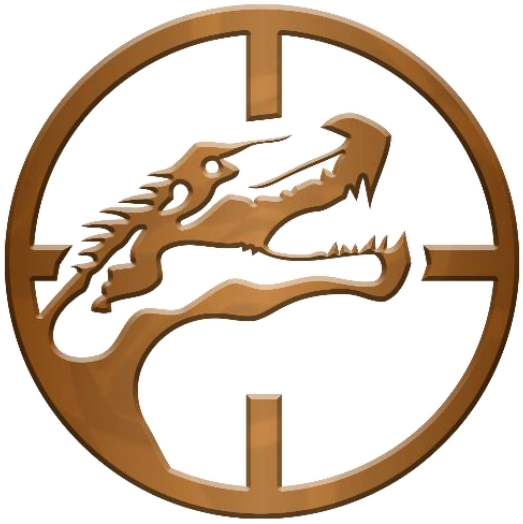


# Jurassic Quest VR

**Autor: Carlos E. Lira Naya**

Consultor: Rafel Perez Vidal
Profesor: Joan Arnedo Moreno

## TFG - Videojuegos

Grado de Ingeniería Informática

7 de junio de 2020

**FICHA DEL TRABAJO FINAL**

| | |
|---|---|
| **Título del trabajo:** | *JurassicQuestVR* |
| **Nombre del autor:** | *Carlos Eduardo Lira Naya* |
| **Nombre del consultor/a:** | *Rafel Perez Vidal* |
| **Nombre del PRA:** | *Joan Arnedo Moreno* |
| **Fecha de entrega:** | 06/2020 |
| **Titulación:** | *Grado en Ingeniería Informática* |
| **Área del Trabajo Final:** | *Videojuegos* |
| **Idioma del trabajo:** | *Español* |
| **Palabras clave** | *Videojuegos, Realidad Virtual (VR), FPS* |

**Resumen del Trabajo:**

JurassicQuestVR es un videojuego que he decidido desarrollar como proyecto de final de grado. La confección de un videojuego involucra todas las fases del proceso de desarrollo del software, haciendo de este un candidato ideal para finalizar la especialización de ingeniería del software dentro del ámbito de la ingeniería informática.

Con este juego se espera adquirir nuevos conocimientos relacionados directamente con el desarrollo de videojuegos y especialmente con entornos de realidad virtual, que como se verá en posteriores apartados de este documento, tienen un nivel de complejidad adicional respecto a los entornos 3D convencionales.

Este TFG está pensado para una duración de un semestre y, por lo tanto, se han planteado unos objetivos realistas y acordes al tiempo del que se despone. Tras culminar el semestre, se espera tener JurassicQuestVR sea un juego que tenga la funcionalidad completa e implemente los primeros tres niveles de juego.

En actualizaciones posteriores a este trabajo de fin de grado se espera añadir contenido de forma progresiva para que el jugador pueda alcanzar el desenlace la JurassicQuestVR y poder así cerrar la historia que se inicia con este proyecto.

**Abstract:**

JurassicQuestVR is a video game I developed as my final project to conclude my bachelors degree. The creation of a video game involves all of the different phases of software development, which makes it an ideal candidate for incorporating all fundamental competencies of a Computer Science degree specialized in Software Engineering.

By developing this project, I expect to acquire new skills directly related with video game development and, more specifically, every aspect that involves virtual reality (VR) environments. As we will see more in depth in further chapters of this document, developing a game in VR adds an additional layer of complexity when compared to standard 3D environments.

The total length of this project encompassed four months; therefore, all the objectives have been selected taking this time constraint into consideration. After finishing the semester, the expectation is to have a finalized version of JurassicQuestVR, which contains the first three functional levels.

After completing this project, the goal is to further expand the game by increasing the number of quests and functional content in order to complete the entire story.

# Índice

# Índice de tablas

# Índice de figuras

# 1. Introducción

## 1.1. Contexto y justificación del trabajo

La idea tras JurassicQuestVR viene de dos aficiones que tengo desde muy pequeño, los dinosaurios y los videojuegos. Este proyecto me pareció la ocasión perfecta para cerrar mis estudios de Ingeniería Informática, ya que los conocimientos adquiridos durante la carrera me van a permitir integrar estos dos mundos dentro de un entorno virtual.

El desarrollo de un videojuego nos lleva por todos los procesos de desarrollo del software, desde la selección de *engine* hasta la esquematización, desarrollo e implementación de sus componentes. Esto hace que este tipo de tecnología sea un candidato ideal como proyecto de final de grado.

## 1.2. Objetivos del trabajo

El objetivo principal de este trabajo es diseñar un videojuego en realidad virtual desde cero. Durante este proceso, se pretenden emplear los conocimientos adquiridos durante la carrera, tanto de diseño, como de programación o usabilidad para obtener un producto final lo más completo posible.

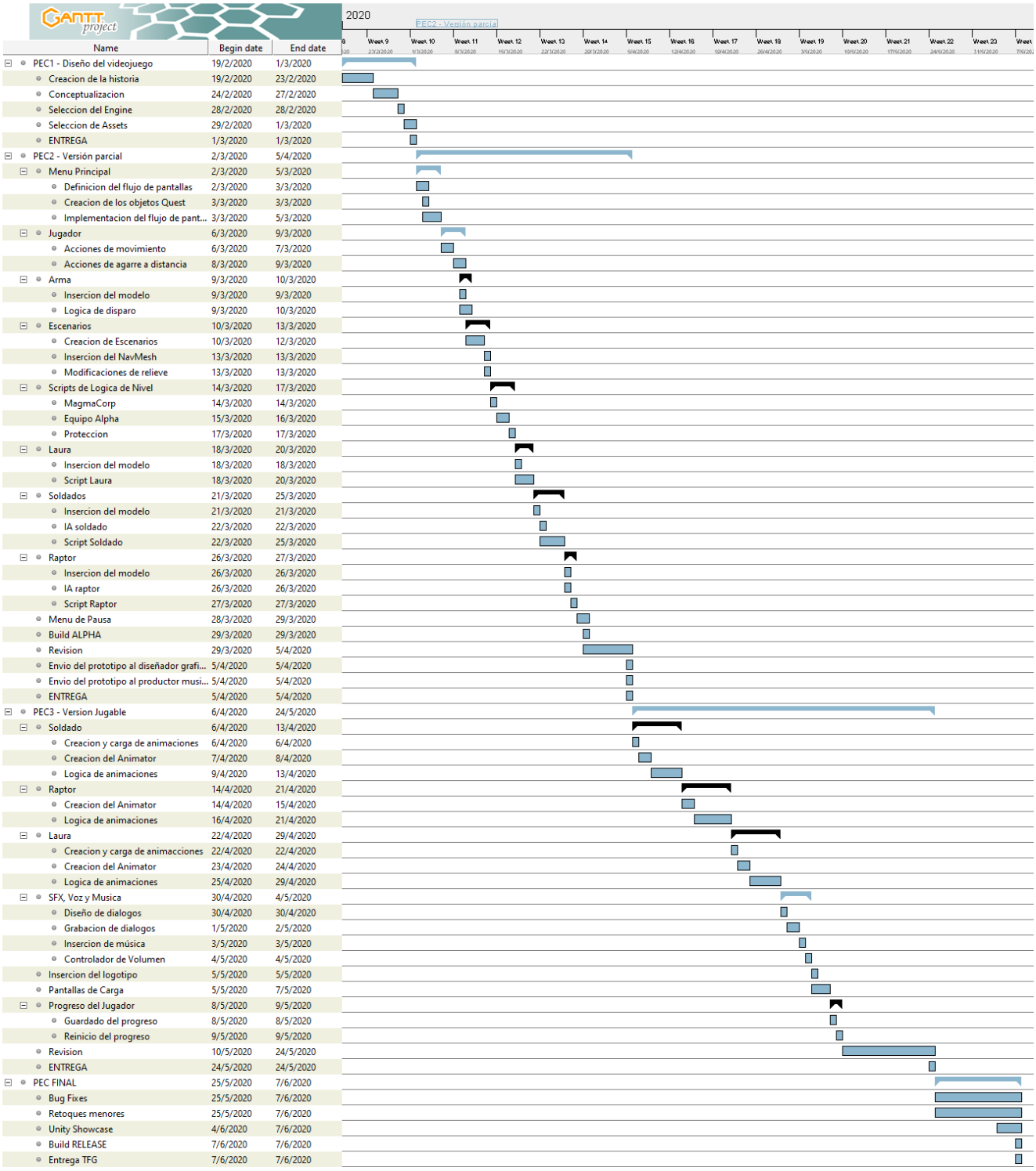
Además, este proyecto servirá para adquirir conocimientos exclusivos relacionados con los *game engines* y otros aspectos de desarrollo específico de videojuegos que no necesariamente son tratados durante la ingeniería informática.

## 1.3. Enfoque y método a seguir

Para llevar a cabo este proyecto el enfoque que se va a realizar se divide en tres partes. Una primera parte que comprenderá el proceso creativo del desarrollo de la historia. Posteriormente se buscarán las herramientas necesarias para poder convertir el proyecto en realidad (selección de *engines*, librerías y *assets* necesarios). Finalmente, se repartirán temporalmente los componentes a realizar durante la duración del semestre y se iniciará la fase de programación.

## 1.4. Planificación del trabajo

Para la planificar los objetivos del desarrollo de JurassicQuestVR se ha realizado el siguiente Gantt que divide las tareas de implementación y desarrollo en a lo largo del semestre, adaptándose a las fechas de las entregas de las pruebas de evaluación continua. Nótese que los últimos días de cada PEC quedan reservados para revisar lo realizado y optimizar o corregir funcionalidades que no estén trabajando como se espera.




| Name | Begin date | End date |
|---|---|---|
| PEC1 - Diseño del videojuego | 19/2/2020 | 1/3/2020 |
| Creacion de la historia | 19/2/2020 | 23/2/2020 |
| Conceptualizacion | 24/2/2020 | 27/2/2020 |
| Seleccion del Engine | 28/2/2020 | 28/2/2020 |
| Seleccion de Assets | 29/2/2020 | 1/3/2020 |
| ENTREGA | 1/3/2020 | 1/3/2020 |
| PEC2 - Versión parcial | 2/3/2020 | 5/4/2020 |
| Menu Principal | 2/3/2020 | 5/3/2020 |
| Definicion del flujo de pantallas | 2/3/2020 | 3/3/2020 |
| Creacion de los objetos Quest | 3/3/2020 | 3/3/2020 |
| Implementacion del flujo de pant... | 3/3/2020 | 5/3/2020 |
| Jugador | 6/3/2020 | 9/3/2020 |
| Acciones de movimiento | 6/3/2020 | 7/3/2020 |
| Acciones de agarre a distancia | 8/3/2020 | 9/3/2020 |
| Arma | 9/3/2020 | 10/3/2020 |
| Insercion del modelo | 9/3/2020 | 9/3/2020 |
| Logica de disparo | 9/3/2020 | 10/3/2020 |
| Escenarios | 10/3/2020 | 13/3/2020 |
| Creacion de Escenarios | 10/3/2020 | 12/3/2020 |
| Insercion del NavMesh | 13/3/2020 | 13/3/2020 |
| Modificaciones de relieve | 13/3/2020 | 13/3/2020 |
| Scripts de Logica de Nivel | 14/3/2020 | 17/3/2020 |
| MagmaCorp | 14/3/2020 | 14/3/2020 |
| Equipo Alpha | 15/3/2020 | 16/3/2020 |
| Proteccion | 17/3/2020 | 17/3/2020 |
| Laura | 18/3/2020 | 20/3/2020 |
| Insercion del modelo | 18/3/2020 | 18/3/2020 |
| Script Laura | 18/3/2020 | 20/3/2020 |
| Soldados | 21/3/2020 | 25/3/2020 |
| Insercion del modelo | 21/3/2020 | 21/3/2020 |
| IA soldado | 22/3/2020 | 22/3/2020 |
| Script Soldado | 22/3/2020 | 25/3/2020 |
| Raptor | 26/3/2020 | 27/3/2020 |
| Insercion del modelo | 26/3/2020 | 26/3/2020 |
| IA raptor | 26/3/2020 | 26/3/2020 |
| Script Raptor | 27/3/2020 | 27/3/2020 |
| Menu de Pausa | 28/3/2020 | 29/3/2020 |
| Build ALPHA | 29/3/2020 | 29/3/2020 |
| Revision | 29/3/2020 | 5/4/2020 |
| Envio del prototipo al diseñador grafi... | 5/4/2020 | 5/4/2020 |
| Envio del prototipo al productor musi... | 5/4/2020 | 5/4/2020 |
| ENTREGA | 5/4/2020 | 5/4/2020 |
| PEC3 - Version Jugable | 6/4/2020 | 24/5/2020 |
| Soldado | 6/4/2020 | 13/4/2020 |
| Creacion y carga de animaciones | 6/4/2020 | 6/4/2020 |
| Creacion del Animator | 7/4/2020 | 8/4/2020 |
| Logica de animaciones | 9/4/2020 | 13/4/2020 |
| Raptor | 14/4/2020 | 21/4/2020 |
| Creacion del Animator | 14/4/2020 | 15/4/2020 |
| Logica de animaciones | 16/4/2020 | 21/4/2020 |
| Laura | 22/4/2020 | 29/4/2020 |
| Creacion y carga de animacciones | 22/4/2020 | 22/4/2020 |
| Creacion del Animator | 23/4/2020 | 24/4/2020 |
| Logica de animaciones | 25/4/2020 | 29/4/2020 |
| SFX, Voz y Musica | 30/4/2020 | 4/5/2020 |
| Diseño de dialogos | 30/4/2020 | 30/4/2020 |
| Grabacion de dialogos | 1/5/2020 | 2/5/2020 |
| Insercion de música | 3/5/2020 | 3/5/2020 |
| Controlador de Volumen | 4/5/2020 | 4/5/2020 |
| Insercion del logotipo | 5/5/2020 | 5/5/2020 |
| Pantallas de Carga | 5/5/2020 | 7/5/2020 |
| Progreso del Jugador | 8/5/2020 | 9/5/2020 |
| Guardado del progreso | 8/5/2020 | 8/5/2020 |
| Reinicio del progreso | 9/5/2020 | 9/5/2020 |
| Revision | 10/5/2020 | 24/5/2020 |
| ENTREGA | 24/5/2020 | 24/5/2020 |
| PEC FINAL | 25/5/2020 | 7/6/2020 |
| Bug Fixes | 25/5/2020 | 7/6/2020 |
| Retoques menores | 25/5/2020 | 7/6/2020 |
| Unity Showcase | 4/6/2020 | 7/6/2020 |
| Build RELEASE | 7/6/2020 | 7/6/2020 |
| Entrega TFG | 7/6/2020 | 7/6/2020 |

*Figura 1: Planificación temporal*

## 1.5. Breve resumen de los productos obtenidos

Una vez finalizado el trabajo, se espera tener los siguientes productos:

- APK de JurassicQuestVR con el juego completo
- Un perfil de Unity Connect
- Un video explicativo del proyecto
- Un repositorio virtual en GitHub con el código fuente

## 1.6. Breve descripción de capítulos de la memoria

El presente documento tiene 8 capítulos adicionales que tratan los siguientes temas:

*Capítulo 1: Introducción*

Capítulo introductorio que justifica la elección del proyecto y lo divide en fases.

*Capítulo 2: Estado del arte*

En este capítulo se realizará una revisión del género del videojuego y un análisis de varios entornos de desarrollo.

*Capítulo 3: Definición del Juego*

Este capítulo explicará aspectos del diseño creativo del juego, inspiraciones y referencias a juegos anteriores, conceptualización y un breve resumen de los costes asociados al desarrollo.

*Capítulo 4: Diseño técnico*

El capítulo del diseño técnico es el más extenso y el que explica, en mayor detalle, los diferentes componentes que componen JurassicQuestVR. Se detallarán aspectos como los recursos utilizados, tanto externos como propios y se explicará detenidamente los diferentes elementos que componen la arquitectura del proyecto.

*Capítulo 5: Diseño de niveles*

Este capítulo es una breve guía explicativa de los niveles de JurassicQuestVR, incluyendo aspectos de diseño y de jugabilidad que indican como superar los niveles de forma exitosa.

*Capítulo 6: Manual del usuario*

El manual del usuario incluye los requisitos mínimos para poder jugar, así como una explicación de los controles y de los enemigos de JurassicQuestVR.

*Capítulo 7: Usabilidad e integración social*

Este capítulo nos proporciona una perspectiva distinta a la del programador e indica otros aspectos que se han tenido en cuenta en el desarrollo de este proyecto.

*Capítulo 8: Conclusiones*

Las conclusiones cierran el documento con una reflexión del proyecto realizado y las lecciones aprendidas.

# 2. Estado del arte

## 2.1. Revisión del género

JurassicQuestVR es un juego de acción y aventura en realidad virtual y primera persona. Como su nombre indica, el género acción-aventura incluye elementos de juegos de acción, como una dinámica de juego rápida, con elementos de juegos de aventura que envuelven partes más de lógica. Este tipo de género, se dice que nació en 1979 con el juego Adventure de la Atari 2600.

Actualmente, existen infinidad de juegos que entran dentro de esta categoría, ejemplos de ellos serían The Legend of Zelda, la saga Halo, Half-Life 2, si nos referimos a entornos 2D y 3D tradicionales; y, si nos referimos a entornos de realidad virtual podemos apreciar juegos como Moss, la saga Vader Immortal, etc.

En general, este subgénero es muy amplio y tiene una definición relativamente laxa, por lo que considerar un juego dentro de la categoría acción-aventura, en vez de acción o aventura es sujeto de interpretación.

## 2.2. Revisión de la tecnología

Para poder desarrollar el juego, es necesario un *game engine*. Un *game engine* es el entorno de desarrollo sobre el que se construye el juego. En lo que a este punto se refiere, se han valorado dos *engines*: Unity y Unreal Engine, ya que son los dos *gameframes* más utilizados actualmente. Cada uno tiene sus pros y sus contras y a continuación se expondrá un resumen de los mismos.

### *Unity*

Unity es un *engine* muy completo y tiene una curva de aprendizaje relativamente baja en comparación a otros *engines*. Otra de las ventajas de Unity, es que como se ha popularizado mucho entre profesionales y gente que quiere adentrarse en la industria del videojuego, dispone de muchísima información y guías gratuitas tanto en YouTube como en el propio portal de Unity. Adicionalmente, otro de los pros de Unity es que dispone de un “asset store” muy completo y dispone de recursos gratuitos y de pago.

Respecto a los contras, Unity no tiene un motor gráfico tan complejo como el de Unreal, por lo que, si se buscan diseños espectaculares, van a ser más difíciles de lograr. También, sobre el papel, uno de los inconvenientes de Unity es que la versión gratuita solo es gratuita hasta cierto límite, que es obtener ingresos de más de 100.000$ en los últimos 12 meses.

Juegos notables: Hearthstone, Cuphead y Beat Saber (List of Unity games, 2020)

### *Unreal Engine 4*

Por otro lado, encontramos Unreal Engine. Unreal es un *engine* conocido por su calidad gráfica y su optimización en la utilización de la iluminación, permitiendo obtener resultados fotorrealistas. Adicionalmente, este engine es el que realiza una mejor optimización del entorno y la implementación.

Sin embargo, los contras de Unreal Engine son que tiene una curva de aprendizaje mucho más elevada, y derivada de esta complejidad es muy poco recomendable desarrollar juegos con equipos muy pequeños sin experiencia previa. Además, similar a Unity uno puede utilizar la versión de Unreal de forma gratuita, pero, a partir de ingresos superiores a 1.000.000$, empieza a cobrar un 5% sobre el beneficio.

Juegos notables: Gears V, Borderlands 3 y Tropico 6 (List of Unreal Engine games, 2020)

# 3. Definición del juego

## 3.1. Idea del Juego

### 3.1.1. Breve descripción del juego

JurassicQuestVR es un juego en realidad virtual en el que el jugador encarna el papel de Emery, un agente de la agencia de inteligencia europea: TITAN. El jugador, en el trascurso de varios niveles deberá infiltrarse en el cuartel general de la organización terrorista MagmaCorp y viajar al pasado para obtener una cura de un virus mortal.

### 3.1.2. Subgénero y referencias a videojuegos existentes

JurassicQuestVR es un juego que mezcla componentes de FPS con infiltración y está dividido en misiones. Según el jugador va superando las diferentes misiones que se le proponen, este va desbloqueando nuevo contenido que le permite continuar avanzando por la historia del juego. Este concepto se asemeja al de los juegos de 007 donde uno tiene una historia subdividida en misiones que se van desbloqueando al progresar.

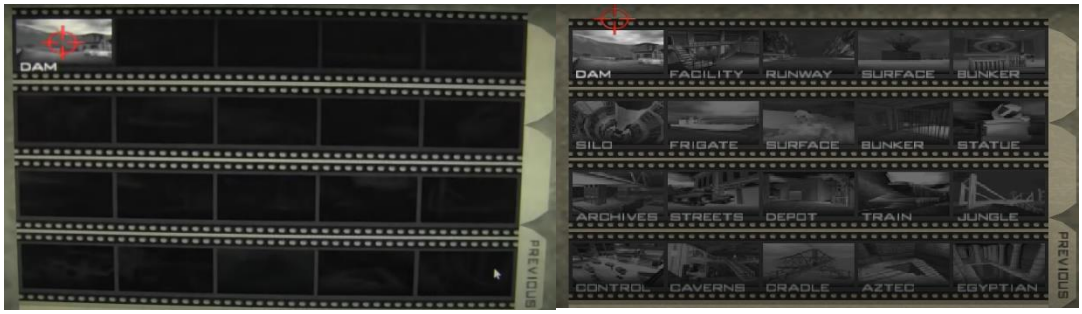



*Figura 2: GoldenEye - N64 (The Mr. X Podcast, 2015)*

Por lo que respecta a los aspectos de infiltración y combate, se busca obtener un estilo de juego similar a Metal Gear Solid donde el jugador, por norma general, no entra en combate hasta que este es detectado o lo inicia de forma activa.


*Figura 3: Metal Gear Solid (Metal Gear Solid Review, 2018)*

Finalmente, uno de los principales atractivos del juego es el combate contra dinosaurios. Por eso se pretende recrear la tensión que generan juegos como Carnivores (1998) cuando uno está siendo atacado por dinosaurios y tiene la necesidad de eliminar a su presa antes de que sea esta la que lo elimine a uno. Adicionalmente, gracias al factor de realidad virtual, se pretende intensificar esta sensación notoriamente.

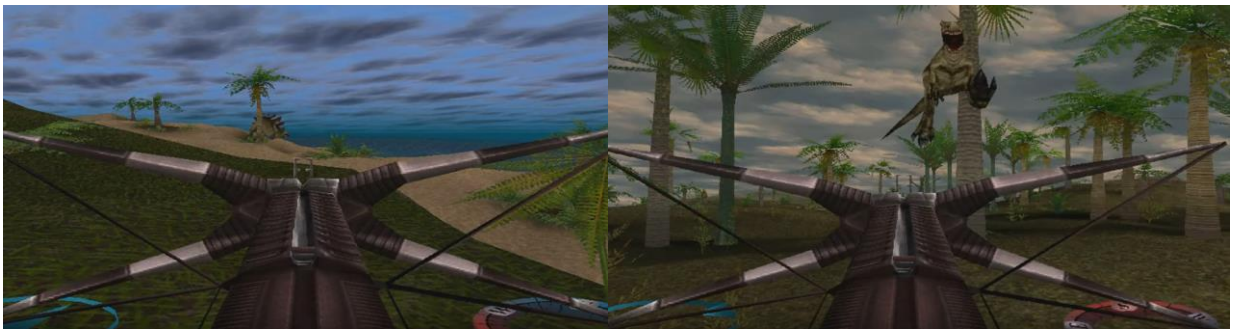


*Figura 4: Carnivores (1998) (Carnivores 1998 PC, 2018)*

### 3.1.3. Tipo de interacción juego-jugador

Como se ha mencionado anteriormente JurassicQuestVR es un juego en realidad virtual cuyo principal objetivo es la inmersión del jugador en la historia. Por ello, se utilizarán elementos de interfaz mínimos para simular un entorno real. Es decir, no se representarán elementos como barras de vida, sino que los únicos elementos de interfaz que se utilizarán serán indicadores para los objetos con los que se pueda interactuar y, excepcionalmente, para opciones de omitir diálogos.

El juego será en primera persona y para interactuar con él se utilizarán los mandos de Oculus Quest. El jugador podrá desplazarse físicamente en su entorno, no obstante, debido a temas de usabilidad y practicidad, se espera que el jugador se mueva principalmente utilizando los joysticks de los controles.

Respecto al funcionamiento de los controles, se intentará simular al máximo el comportamiento de la mano para interactuar con los objetos del entorno. Por ejemplo, acercarse a un objeto y cerrar el puño apuntando hacia el agarrará el objeto, o, flexionar el dedo índice al estar empuñando un arma será el equivalente a apretar el gatillo.

Adicionalmente, por temas de usabilidad, para compensar las posibles restricciones de movilidad del jugador (ya sean corporales o de su entorno), JurassicQuestVR contará con una mecánica de agarrar a distancia, evitando así que los jugadores tengan que hacer movimientos incómodos como por ejemplo agacharse.

Por último, el principal objetivo que pretende alcanzar JurassicQuestVR es la total inmersión del jugador en la realidad del juego. Por ello, el jugador encarnará al "Agente 01", el agente carecerá de modelo 3D y de voz. El objetivo de esto es independizar temas raciales y de identidades de género de la historia de JurassicQuestVR y pretender que un mayor número de público pueda sumergirse en la trama.

### 3.1.4. Plataforma de destino

La plataforma de destino del juego serán los dispositivos Oculus de realidad virtual. El juego está pensado para ser ejecutado en Oculus Quest, ya que esta será la principal plataforma contra la que se desarrolle y se testee, pero en un futuro, fuera del alcance de este trabajo, se pretende ampliar este abanico a los dispositivos Rift y Go.

## 3.2. Conceptualización

### 3.2.1. Historia, ambientación y trama

#### *Historia*

Nos encontramos en el año 2025. MagmaCorp, una organización terrorista formada por militares renegados de los ejércitos de numerosos países, ha desarrollado ha desarrollado el KelsoR, un patógeno extremadamente contagioso y letal para el ser humano. MagmaCorp ha lanzado una amenaza a escala mundial advirtiendo de futuros ataques en las principales capitales mundiales.

Para evitar esta potencial tragedia, la agencia de espionaje e inteligencia europea (TITAN) envió al equipo Alpha, una división de agentes de élite, para infiltrarse en MagmaCorp y obtener información sobre como detener el KelsoR. Sin embargo, se perdió el contacto por radio con los integrantes del grupo.

Por ello, ahora TITAN se ha llamar del retiro al que ha sido su mejor agente, el Agente 01. El Agente 01 deberá infiltrarse en MagmaCorp y averiguar cualquier información que pueda ayudar a detener el KelsoR. El destino de la humanidad está sus manos.

#### *Ambientación*

El juego tiene dos ambientaciones principales, una actual y otra que nos transporta a 75M A.C. La etapa actual será dentro de las instalaciones de MagmaCorp, donde el protagonista deberá infiltrarse en un almacén custodiado por los soldados de la organización. Ahí descubrirá un portal que le permitirá transportarse al pasado.

La etapa del pasado está ambientada en una isla de aspecto tropical, donde los únicos elementos generados por el hombre son los que MagmaCorp ha transportado previamente. Esta isla está habitada por dinosaurios y criaturas prehistóricas.

#### *Trama*

La historia se dividirá en múltiples misiones, no obstante, para este proyecto solo se implementarán las tres primeras:

##### *MagmaCorp*

El Agente 01 aparece en las instalaciones de MagmaCorp. Deberá de infiltrarse sin ser detectado y adentrarse en el portal que lo llevará al pasado. Para ello deberá ayudarse de la baja iluminación del entorno guiándose por la iluminación de las linternas de los soldados.

#### *Equipo Alpha*

El Agente 01 acaba de atravesar el portal y se encuentra en una isla 75M de años A.C. Allá encontrará a Laura, la única superviviente del equipo Alpha, que ha sido capturada por MagmaCorp. Tras liberar a Laura de sus captores, ella le al protagonista sus descubrimientos sobre el KelsoR: el patógeno ha sido desarrollado con ADN de raptor. Finalmente, ambos protagonistas se dirigirán a la zona donde se encuentra el "Sujeto 0", el raptor a partir del que han creado el virus.

#### *Protección*

Tras acabar con los soldados que custodiaban al "Sujeto 0", Laura se dispone a obtener una muestra de sangre de este. No obstante, el tiroteo y el olor a sangre han atraído a los raptores de los alrededores. El Agente 01 deberá proteger a Laura y sobrevivir hasta que finalice la extracción de sangre.

### 3.2.2. Definición de los personajes y elementos

#### *TITAN*

TITAN es una agencia de inteligencia europea que vela por la seguridad mundial.

#### *Agente 01*

El jugador encarna el papel del Agente 01. El mejor agente que jamás ha tenido TITAN.

#### *Laura*

Laura es una soldado de élite de TITAN, pertenece a la división Alpha. Se sabe que Laura es extremadamente habilidosa con las armas a distancia y es una excelente conductora.

#### *MagmaCorp*

MagmaCorp es una organización terrorista formada por militares renegados. Su principal motivo es el caos y control mundial.

#### *KelsoR*

El KelsoR es un patógeno desarrollado por MagmaCorp a partir de ADN de raptor. El virus es mortal para los humanos.

*Sujeto 0*

El Sujeto 0 es el raptor cuya sangre ha sido utilizada para generar el KelsoR. En su sangre está la clave para descubrir el anticuerpo que sea capaz de combatir al virus.

*Raptores*

Los raptores son extremadamente agiles y agresivos. Ante cualquier amenaza u objetivo, no dudarán en atacar.

*Soldado de MagmaCorp*

Los soldados de MagmaCorp son los encargados de defender las instalaciones y toda la información relacionada con el KelsoR. Se desconocen sus habilidades o a quien responden.

### 3.2.3. Interacción entre los actores del juego

Además del protagonista, distinguimos tres actores o tipos de actores principales en la historia. Por un lado, tenemos a Laura que será quien guie al protagonista a través de la historia. Su papel consiste en comunicarle al protagonista, a través de voz, cuáles son los siguientes pasos a seguir.

Por otro lado, tenemos los actores enemigos. En primer lugar, encontramos a los soldados. Estos tienen tres estados de juego: patrulla, alerta y combate. Cuando un soldado entra en modo combate, este empieza a perseguir al jugador y a dispararle. La persecución no cesará hasta que el soldado o el jugador mueran.

Por último, encontramos a los raptores. Los raptores son criaturas que se habitan en el pasado y son extremadamente agresivas. Ellos perseguirán y atacarán al jugador hasta que uno de los dos muera. El raptor atacará saltando sobre el jugador o mordiéndole.

### 3.2.4. Objetivos plantados al jugador

Los objetivos que se le plantean al jugador se especifican en el menú de selección de misión. Estos varían en función del nivel pudiendo contener elementos de infiltración, rescate o supervivencia. A continuación, se explican los objetivos por nivel.

*MagmaCorp*

Llegar al portal sin ser detectado por los solados de MagmaCorp.

*Equipo Alpha*

Obtener un arma, rescatar a Laura y seguir sus instrucciones.

*Protección*

Sobrevivir al ataque de los raptores mientras Laura extrae la muestra de ADN.

### 3.2.5. Bocetos y concept art

A continuación, se presentan unos bocetos a mano alzada de los tres niveles para que el lector los visualice con mayor facilidad.

*MagmaCorp*

Se presenta una vista cenital del edificio de MagmaCorp. El jugador (Player) deberá llegar hasta el portal sin ser detectado por los enemigos (en rojo). La iluminación del nivel será baja y la principal guía que podrá utilizar el jugador para saber cuándo moverse es la iluminación de las linternas de los soldados.

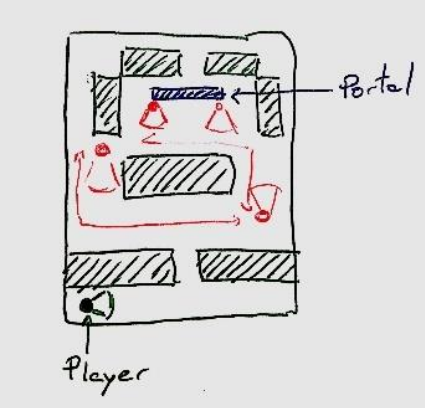



*Figura 5: MagmaCorp (Concept)*

*Equipo Alpha*

Cuando el jugador salga del portal, deberá ir a buscar un arma para poder deshacerse de los soldados (en rojo) que tienen capturada a Laura. Una vez Laura haya sido liberada, le guiará hasta donde se encuentra el “Sujeto 0”. A continuación, vemos un bosquejo de la zona inicial del nivel.

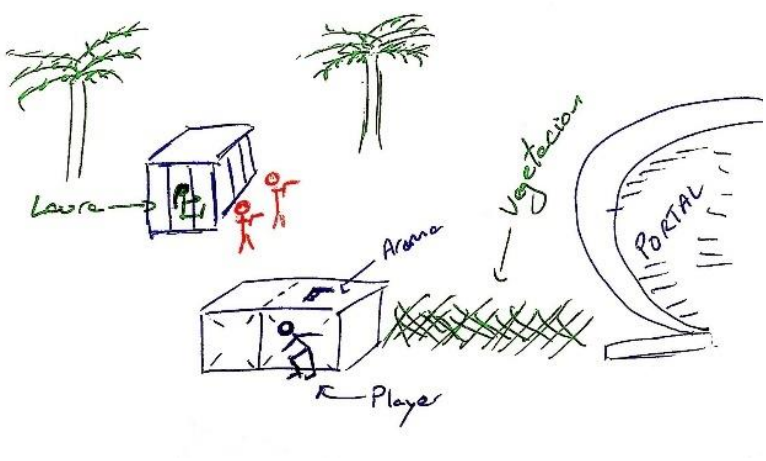



*Figura 6: Equipo Alpha (Concept)*

### *Protección*

Laura y el Agente 01 se encuentran ante el "Sujeto 0" (raptor sedado). Mientras Laura realiza la extracción de sangre, el jugador deberá defenderlos del ataque de otros raptores.

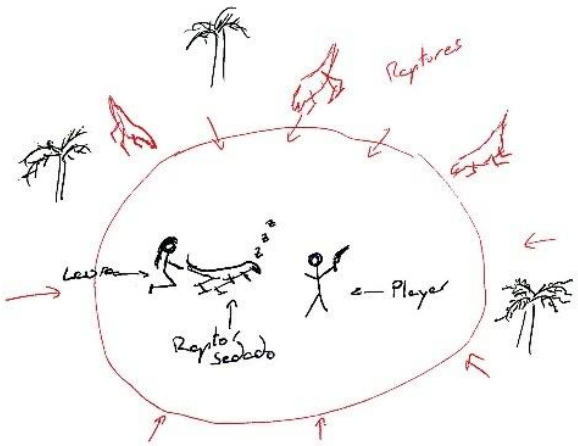



*Figura 7: Protección (Concept)*

## 3.3. Desarrollo y Roadmap

### 3.3.1. Elección del *engine*

En el apartado 2.2 de este documento, se ha presentado una comparativa entre Unreal Engine y Unity. Finalmente, para desarrollar JurassicQuestVR se ha optado por utilizar Unity. Los principales factores que me han llevado a tomar esta decisión son, la gran disponibilidad de guías y tutoriales gratuitos, su sencilla integración con Oculus y que, anteriormente, he realizado proyectos muy sencillos en este *engine.*

### 3.3.2. Planificación de objetivos

Recordamos que el objetivo principal de este proyecto será tener finalizados los primeros 3 niveles del juego: MagmaCorp, Equipo Alpha y Protección. No obstante, para tener una funcionalidad completa, deben incluirse elementos como menús, etc. Por este motivo se va a realizar una planificación de objetivos que nos asegure que ningún elemento quede descuidado.

Los objetivos a tener en cuenta, pueden dividirse en dos grandes categorías: menús y jugabilidad. Hay 2 elementos de interfaz que tendremos que implementar en el juego. Un menú principal que nos permitirá acceder a los diferentes niveles del juego y un menú de pausa que nos permita, durante una partida, reiniciarla o salir. Puede observarse que en un principio queda excluido realizar una interfaz de usuario dentro de la partida, para dar un aspecto más realista.

En cuanto a los aspectos de jugabilidad encontramos varios: el jugador (Agente 01), Laura, los enemigos y las armas. Cada uno tendrá sus componentes de sonido y animaciones y en el caso de los enemigos tendrán que disponer de una inteligencia artificial.

El *roadmap* del proyecto puede verse en el diagrama de Gantt representado en el apartado 1.4 de este documento.

## 3.4. Costes asociados al proyecto

Finalmente, se va a realizar la siguiente distribución de costes por recurso a nivel de proyecto. Nótese, que algunos elementos ya estaban a mi disposición antes de empezar el desarrollo, no obstante, el proyecto se va a tratar como si se empezase desde 0 y, por lo tanto, se contabilizarán dichos elementos.

### *Hardware*

A continuación, se presenta una lista de los elementos de hardware que se utilizarán en el desarrollo.

| ITEM | DESCRIPCIÓN | PRECIO |
|---|---|---|
| **Acer Predator Helios 300** | Ordenador portátil compatible con realidad virtual. | 1200.00$ |
| **Oculus Quest 64GB** | Equipo de realidad virtual | 400.00$ |
| **Oculus Link** | Cable USB-C de fibra óptica que permite integrar Quest con Unity en tiempo real. | 80.00$ |
| **TOTAL HARDWARE** | | 1680.00$ |

*Tabla 1: Coste de Hardware*

### *Software*

A continuación, se presenta una lista de los elementos de hardware que se utilizarán en el desarrollo.

| ITEM | DESCRIPCIÓN | PRECIO |
|---|---|---|
| **Unity** | Engine sobre el que se desarrollara el juego. | 0.00$ |
| **Visual Studio** | IDE sobre el que se codificará | 0.00$ |
| **Mixamo.com** | Portal de Adobe que realiza animaciones sobre modelos 3D. | 0.00$ |
| **Audacity** | Software para edición de Audio | 0.00$ |
| **TOTAL SOFTWARE** | | 0.00$ |

*Tabla 2: Coste del Software*

### *Modelos*

A continuación, se presenta una lista de los elementos de hardware que se utilizarán en el desarrollo. Nótese que debido a la escasez de tiempo para realizar el proyecto y mi falta de práctica en modelado 3D, se va a destinar un presupuesto para comprar modelos si es necesario.

| ITEM | DESCRIPCIÓN | PRECIO |
|---|---|---|
| **Asset Jurassic Pack** | Pack del Unity Asset store que incluye modelos de dinosaurios y animaciones. | 119.00$ |
| **Modelo Laura** | Modelo utilizado para Laura | 35.00$ |
| **Asset Portal** | Asset del Unity Asset store con modelos y animaciones de diversos portales. | 10.00$ |
| **TOTAL MODELOS** | | 164.00$ |

*Tabla 3: Coste de los modelos 3D*

### *Música y SFX*

A continuación, se presenta una lista de los elementos de sonido que se utilizarán en el desarrollo. Todos los sonidos y músicas utilizadas serán *royalty free* o propietarias. Para las voces, mis amigos y familiares se han ofrecido a realizarlas gratuitamente.

| ITEM | DESCRIPCIÓN | PRECIO |
|---|---|---|
| **SFX** | Efectos de sonido | 0.00$ |
| **Música** | Música | 0.00$ |
| **Voces** | Voces | 0.00$ |
| **TOTAL SONIDO** | | 0.00$ |

*Tabla 4: Coste del sonido*

### *Programación*

Para el coste de la programación, se partirá de una tarifa básica de desarrollador de videojuegos junior en Estados Unidos, que es de unos 25$ la hora. Adicionalmente, se estima que el proyecto tendrá unas 400 horas de duración. Respecto al modelaje, como se ha comentado anteriormente, los *assets* serán adquiridos y no diseñados.

| ITEM | DESCRIPCIÓN | PRECIO |
|---|---|---|
| **Programación** | Coste estimado de la mano de la programación y diseño del proyecto teniendo en cuenta un salario de 25$ la hora y una duración estimada de 400 horas. | 10.000$ |
| **Modelaje 3D** | Coste del modelaje 3D | 0.00$ |
| **TOTAL MANO DE OBRA** | | 10.000$ |

*Tabla 5: Coste de la mano de obra*

### *Coste Total*

Tras agregar los costes de desarrollo mencionados anteriormente de programación, sonido, modelaje, software y hardware, obtenemos un coste total de desarrollo de 11844$.

| ITEM | DESCRIPCIÓN | PRECIO |
|---|---|---|
| **Hardware** | Coste total del hardware | 1680.00$ |
| **Software** | Coste total del software | 0.00$ |
| **Assets y Modelos** | Coste total de assets y modelos | 164.00$ |
| **Musica y SFX** | Coste total de la música y SFX | 0.00$ |
| **Mano de obra** | Coste total de la mano de obra | 10.000$ |
| **TOTAL DESARROLLO** | | 11844.00$ |

*Tabla 6: Coste total de desarrollo*

# 4. Diseño técnico

## 4.1. Entorno y requisitos

### 4.1.1. Entorno

Tal y como se ha especificado en el apartado anterior, para desarrollar JurassicQuestVR se ha optado por utilizar Unity como engine de desarrollo. Principalmente, los motivos por los que se ha escogido Unity, son la relativamente baja curva de aprendizaje que tiene, la abundancia de recursos y tutoriales disponibles, la versatilidad de su "Asset Store" y su sencilla integración con el SDK de Oculus.

Adicionalmente, como IDE complementario a Unity se va a utilizar Visual Studio 2019, ya que este es gratuito y sus capacidades de IntelliSense facilitan mucho la tarea del programador. Además, Visual Studio es una herramienta con la que trabajo a diario y por lo tanto confío en que me proporcionará los recursos necesarios para llevar a cabo el proyecto de forma cómoda.

### 4.1.2. Requisitos

Respecto a los requisitos técnicos del entorno de desarrollo, vamos a necesitar tres elementos principales:

*Ordenador compatible con VR*

Puesto que el proyecto es en realidad virtual, el equipo con el que se desarrolla ha de ser capaz de soportar la carga adicional de renderización que implica la VR.

| | Requisitos mínimos | Equipo utilizado |
|---|---|---|
| **Procesador** | Intel I5-4590 (o equivalente) | Intel I7-9750H |
| **Gráfica** | NVIDIA GTX 970 (o equivalente) | NVIDIA GeForce RTX 2060 |
| **RAM** | 8GB | 16GB |
| **Puertos** | USB 3.0 | USB 3.0 |
| **S.O.** | Win7 x64 | Win10 x64 |

*Tabla 7: Requisitos de desarrollo*

### *Oculus Quest*

El sistema objetivo para el que se está desarrollando el juego es Oculus Quest, por lo tanto, disponer de un dispositivo de este tipo es un requisito indispensable. Tanto la version de 64GB como la de 128GB funcionarán.

### *Oculus Link*

Oculus Quest está diseñado para ser un equipo “standalone”, por lo que no necesita de una conexión a un PC para reproducir contenido en realidad virtual. Esto es un pro muy grande, pero para desarrollar aplicaciones es incómodo puesto que para testearlas se debe hacer el *build* e instalar en el *headset*.

Oculus Link, por suerte, nos permite sobrepasar esta limitación. Este cable de fibra óptica se conecta al puerto USB 3.0 del equipo y permite renderizar las imágenes del equipo en el *headset* en tiempo real a través del propio software de Oculus.

## 4.2. Herramientas de software utilizadas

A continuación, se presenta un listado de las herramientas que se han empleado en el desarrollo de JurassicQuestVR.

### *Unity*

Como se ha explicado en apartados anteriores, Unity es el *engine* sobre el que se ha construido el juego.

### *Visual Studio 2019*

Visual Studio es el *Integrated Development Environment (IDE)* que se ha utilizado para crear los *scripts* y clases (en C#) que implementan la funcionalidad del juego.

### *Mixamo.com*

Mixamo es una plataforma gratuita de Adobe que permite subir modelos 3D que tengan un esqueleto y genera animaciones para ellos.

### *Audacity*

Audacity es un software de edición y grabación de sonido que ha sido utilizado para modificar y limpiar los sonidos de las voces.

### *Oculus App*

La aplicación de escritorio de Oculus permite visualizar en tiempo real en el *headset*, a través del cable Oculus Link, el juego en modo reproducción desde Unity.

### *adbLink*

Por último, adbLink es un software que permite instalar y desinstalar de forma simplificada APKs en el Oculus Quest.

## 4.3. Assets y Recursos Externos

A continuación, se presenta un listado de los assets y recursos externos que se han utilizado para el desarrollo de JurassicQuestVR.

### *Oculus Integration*

Este asset es el que permite integrar en el proyecto el SDK de Oculus.

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

### *Jurassic Pack Vol. I Dinosaurs*

Jurassic Pack es uno de los assets principales del juego. Este contiene el modelo de los raptores y la base del mapa utilizado en los niveles “Equipo Alpha” y “Protección”. Este asset viene con su propio set de sonidos ambientales, además de animaciones y sonidos de dinosaurios.


*Figura 8: Jurassic Pack Asset*

Link: https://assetstore.unity.com/packages/3d/characters/animals/jurassic-pack-vol-i-dinosaurs-35660

### *Snaps Prototype | Sci-Fi / Industrial*

Asset que ha proporcionado las herramientas necesarias para crear el primer nivel “MagmaCorp”.

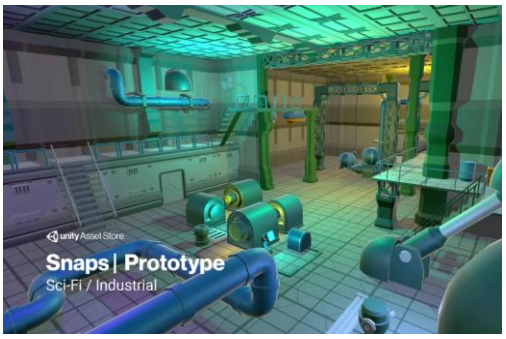



*Figura 9: Snaps Prototype*

Link: https://assetstore.unity.com/packages/3d/environments/sci-fi/snaps-prototype-sci-fi-industrial-136759

### *Shipping Container*

Asset que proporciona los contenedores metálicos utilizados en “MagmaCorp”.

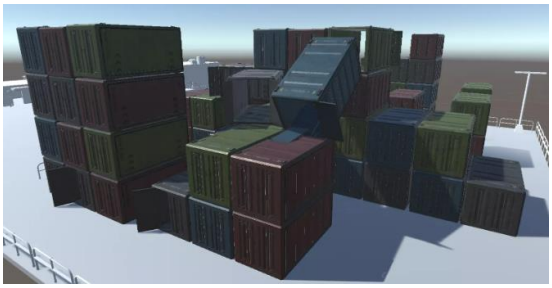


*Figura 10: Shipping Container*

Link: https://assetstore.unity.com/packages/3d/environments/shippingcontainer-147902

### *Wooden Crates*

Cajas de madera utilizadas en TeamAlpha. Sobre estas, el jugador encuentra la pistola.

*Figura 11: Wooden Crates*

Link: https://assetstore.unity.com/packages/3d/props/wooden-crates-16599

### *Low Poly Soldiers Demo*

Modelo base utilizado para los soldados. Se ha escogido un modelo *low poly* para disminuir el número de polígonos a renderizar por el headset.

*Figura 12: Low Poly Soldiers*

Link: https://assetstore.unity.com/packages/3d/characters/low-poly-soldiers-demo-73611

### *Modern Guns: Handgun*

Modelo de la pistola que utiliza el jugador. La pistola viene con sus propias animaciones y script de disparo que controla su *Animator Controller.*

*Figura 13: Modern Guns: Handgun*

Link: https://assetstore.unity.com/packages/3d/props/guns/modern-guns-handgun-129821

### *Desert Eagle .50 AE Close Single Gunshot A Sound Effect*

Efecto de sonido utilizado para la pistola.

Link: https://www.fesliyanstudios.com/play-mp3/7148

### *Pack of magic portals*

Este pack de portales es el que proporciona el modelo 3D del portal de “MagmaCorp” y “Equipo Alpha”.

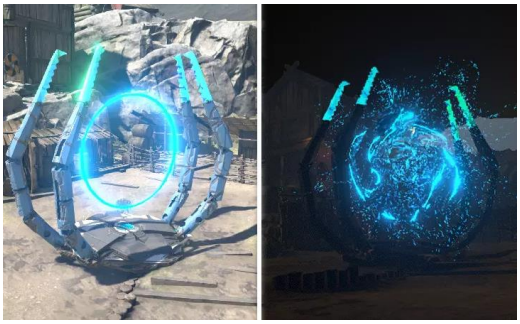


*Figura 14: Pack of magic portals*

Link: https://assetstore.unity.com/packages/vfx/particles/pack-of-magic-portals-123516

### *Air portal (Sound Effect)*

Efecto de sonido utilizado para simular la estática del portal. Ese ha sido modificado con Audacity para lograr una secuencia de audio con un *loop* perfecto.

Link: https://freesound.org/people/humanoide9000/sounds/329028/

### *Unity Samples UI*

Este asset ha sido empleado para obtener un diseño de menú de carácter futurista.


*Figura 15: Unity Samples UI*

Link: https://assetstore.unity.com/packages/essentials/unity-samples-ui-25468

### *Warrior Scarlet v2 Girl*

Modelo 3D utilizado para encarnar a Laura.

*Figura 16: Warrior Scarlet v2*

Link: https://www.turbosquid.com/3d-models/warrior-girl-scarlet-v2-model-1284326

## 4.4. Recursos Propios

Para la creación de JurassicQuestVR, además de los recursos gráficos y auditivos mencionados anteriormente, se han generado otros tantos para completar el contenido del juego y que son propietarios de JurassicQuestVR.

### 4.4.1. Música

JurassicQuestVR dispone de su propia banda sonora original. Esta ha sido creada y diseñada por Manuel Alejandro Ramirez Diaz, bachiller de música en "Performance & Contemporary Writing and Production" por la universidad de Berklee (Boston, MA).

Manuel Alejandro ha creado 4 piezas musicales:

JurassicQuest: Para el menú principal

MagmaCorp: Para el primer nivel

Team Alpha: Para el segundo nivel

Protection: Para el tercer nivel

Además, también ha generado los efectos de sonido que se reproducen al hacer click, retroceder o *hoverear* un botón de la interfaz.

### 4.4.2. Logotipo

El logo de JurassicQuestVR ha sido creado por Marc Grau Garcías, diseñador gráfico.


*Figura 17: Logo JurassicQuestVR*

### 4.4.3. Voz

La voz de Laura, la agente de TITAN, ha sido puesta por Laura Jeannette Ramírez Diaz. Por descontado, el personaje tiene el nombre de Laura en agradecimiento a ella.

## 4.5. Arquitectura de Componentes

La forma como se ha organizado el juego es a través de una división en escenas. A partir de ahí, cada escena tiene sus propios objetos de juego (*GameObjects*) los cuales, a su vez, tienen *scripts* añadidos que permiten añadir lógica a dichos objetos.

A grosso modo, JurassicQuestVR cuenta de 6 escenas: una escena persistente accesible desde todas las escenas de juego, cuatro escenas jugables y una escena de testeo. Cada una de estas tienen una serie de elementos y componentes asociados que trabajan en conjunto para implementar la lógica del juego. A continuación, se muestra un diagrama de los componentes principales de las escenas jugables.

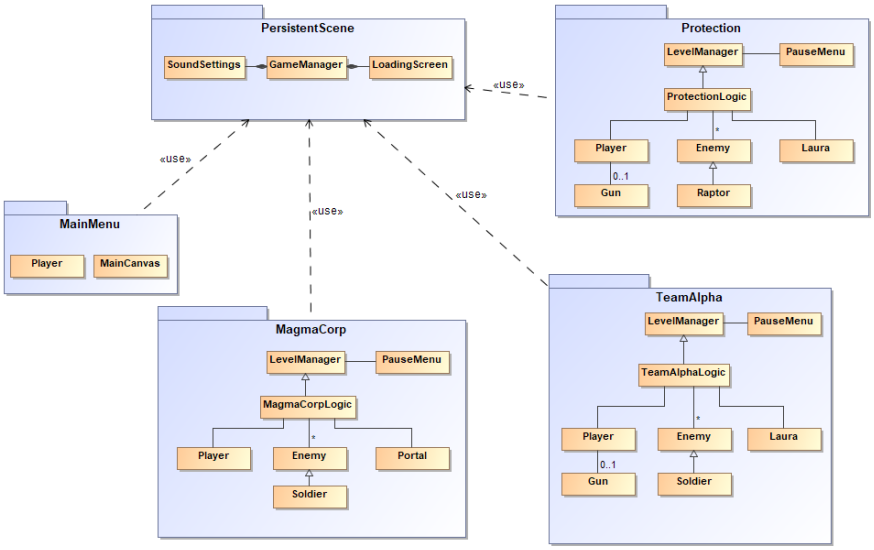



*Figura 18: Diagrama de componentes I*

Además de estas escenas jugables, el juego dispone de una escena de testeo que funciona de forma totalmente autónoma que se utiliza para probar y desarrollar nuevas funcionalidades.

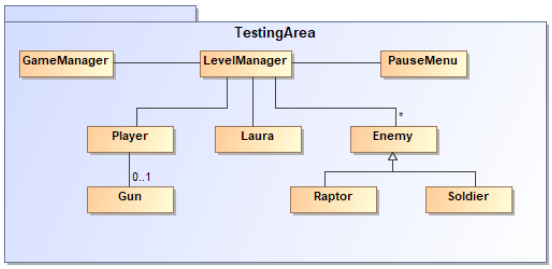



*Figura 19: Diagrama de componentes II*

En los apartados 4.6, 4.7, 4.8 y 4.9 de este documento, se hace una explicación en profundidad de todos los elementos que componen el juego.

## 4.6. Escenas

Este proyecto ha utilizado 6 escenas principales, cuatro de ellas forman parte de la *build* final y la quinta se utiliza como entorno de testeo, que se utiliza para probar modelos, animaciones y comportamientos en un entorno simplificado antes de agregarlos a las escenas de producción.

A continuación, se muestran las escenas utilizadas:

### 4.6.1. PersistentScene

Esta escena, como su nombre indica persiste durante la ejecución del juego y únicamente se destruye al salir de la aplicación. La "PersistentScene" se encarga de llevar los componentes que son utilizados en todas las escenas del juego para hacerlos accesibles desde todas partes.

Hay tres componentes principales que se arrastran entre escenas:

#### *GameManager*

Este se encarga de monitorizar el progreso del jugador, guardar y trasladar las opciones de sonido además de transmitir los sonidos básicos que aparecen en todas o casi todas las escenas (canción de victoria, canción de derrota y los sonidos de los botones).

Adicionalmente, el GameManager es el encargado de cargar las escenas y activar la pantalla de carga durante procesos de carga y descarga.

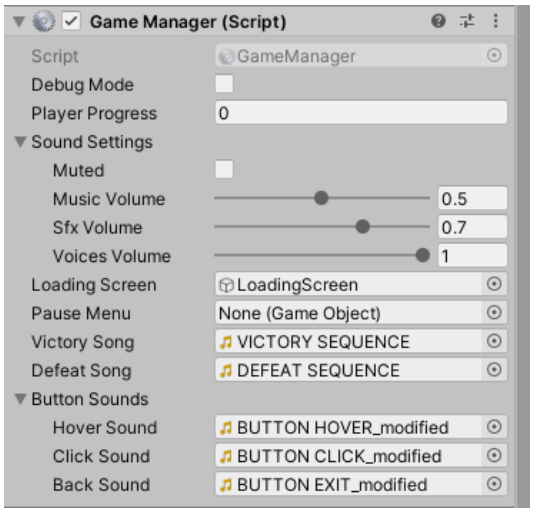



*Figura 20: GameManager*

#### *OVRManager*

El OVRManager es el encargado de trasladar las interacciones del juego a través del SDK de Oculus al headset de realidad virtual. Es muy importante que solo haya un único objeto que tenga un OVRManager. En caso de existir más de uno, estas entran en conflicto entre ellas por el control del headset y resulta en comportamientos no deseados.

Esto era especialmente problemático en los transcursos de escena ya que entre el proceso de descarga de una escena y de carga de la otra, había un momento en el que dos objetos Jugador llegan a coexistir en el tiempo (nótese que cada cámara de un jugador, por defecto, tiene un OVRManager). Por ese motivo, se decidió trasladar el OVRManager a la escena persistente, de modo que todos los objetos Jugador referencien a la misma instancia del manager.

#### *Pantalla de carga*

La pantalla de carga se compone de dos objetos, una cámara y un OVROverlay. El OVROverlay es una aplicación del SDK de Oculus que nos permite poner una imagen delante del lente del headset. Se ha utilizado esta combinación de modo que cuando la cámara de la pantalla de carga se active, esta tenga precargado un OVROverlay compuesto por un *skybox* y el logotipo del juego.


*Figura 21: Pantalla de carga*

### 4.6.2. MainMenu

La escena “MainMenu” corresponde a la escena del menú principal. Esta es la primera pantalla donde el jugador aparece al ejecutar el juego y desde la que puede seleccionar el nivel a jugar, modificar las opciones de sonido, consultar los controles, borrar su progreso y consultar los créditos.

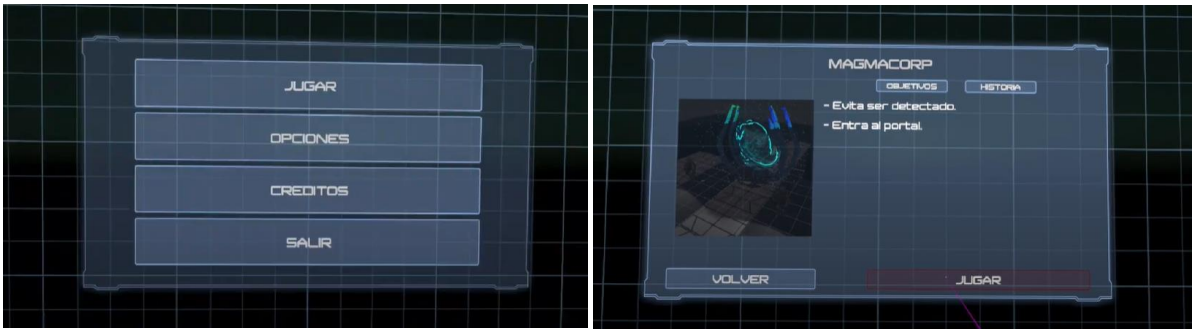



*Figura 22: Menu principal*

El panel de muestra de misiones carga dinámicamente un objeto “Quest” en función de la misión que oprime el jugador. Los objetos “Quest” están compuestos por 5 elementos: un QuestId que hace referencia a la escena que representan, un título, una historia, una imagen y una lista de objetivos.

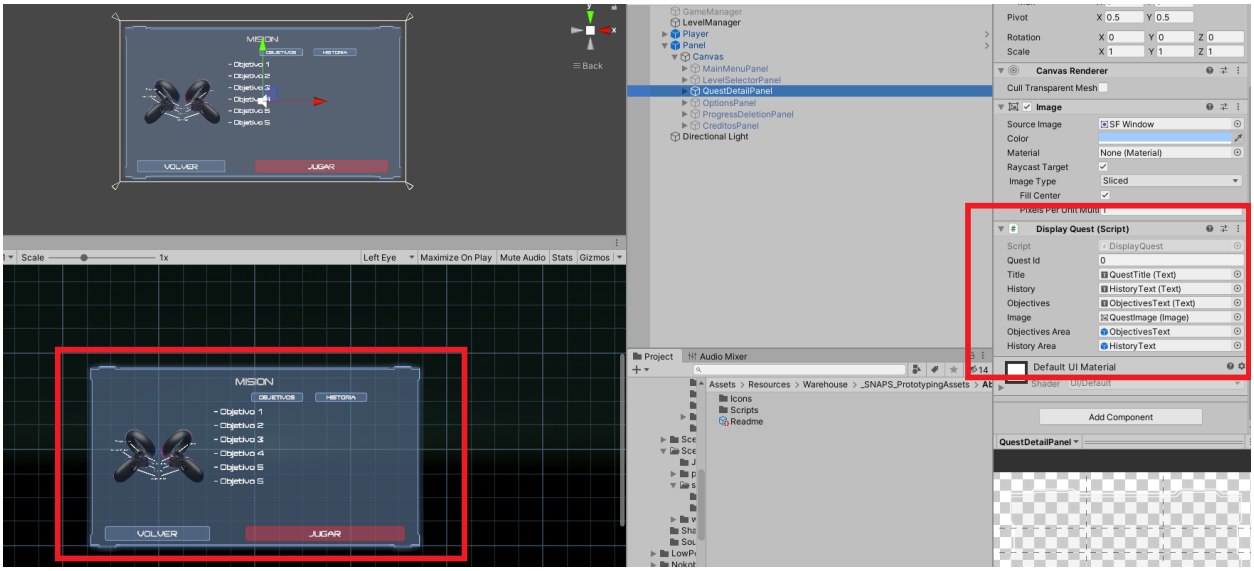



*Figura 23: Objetos Quest*

### 4.6.3. MagmaCorp

MagmaCorp se corresponde al primer nivel de juego y corresponde a la base militar de MagmaCorp. Esta escena incorpora las primeras mecánicas de sigilo, como por ejemplo las soldados patrullando. Ser detectado en este nivel implica perder. Es un nivel considerablemente oscuro y se busca que el jugador interactúe con la oscuridad del entorno para poder llegar al portal. Los soldados son fácilmente identificables por los haces de luz que proyectan.

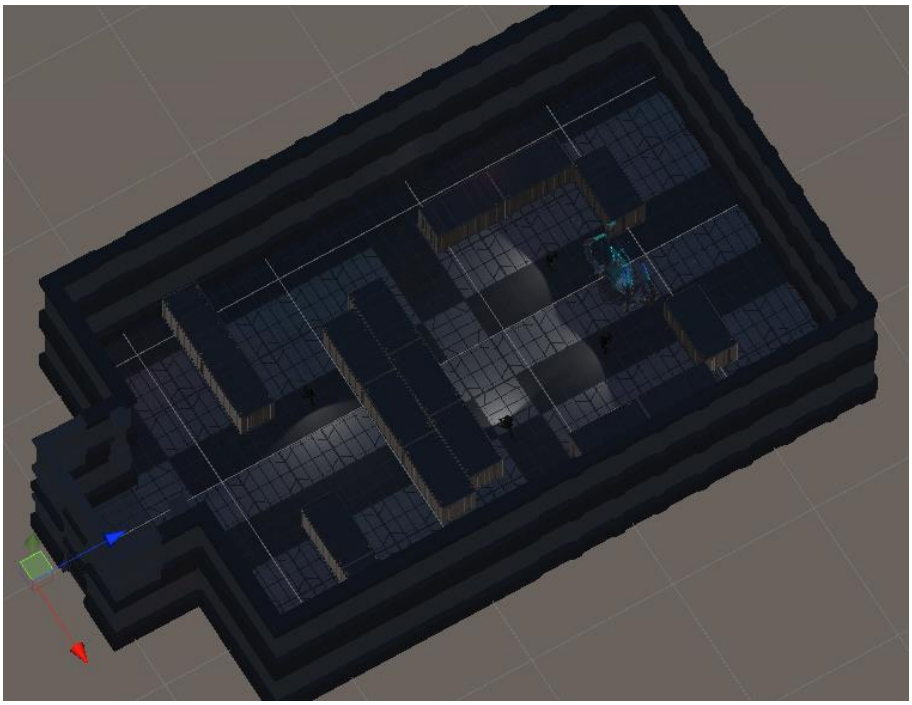


*Figura 24: MagmaCorp (Vista aérea)*

Los soldados tienen tres modalidades: patrulla, alerta y combate. Estas son fácilmente identificables en función del color de sus linternas:

- Blanco: No alerta
- Amarillo: Modo alerta, posible jugador avistado (el jugador tiene 1s para salir del rango de visión)
- Rojo: Modo combate, jugador detectado

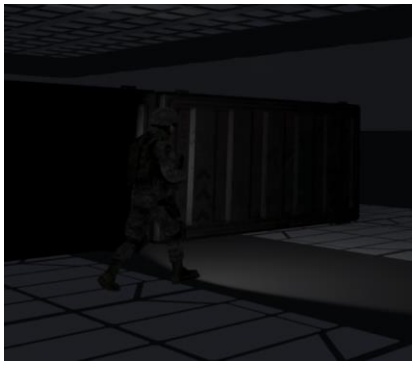


*Figura 25. Modo Patrulla*

*Figura 26. Modo Alerta*

*Figura 27. Modo Combate*

### 4.6.4. TeamAlpha

TeamAlpha es el segundo nivel de juego e incorpora las primeras mecánicas de combate, aquí el jugador deberá hacerse con un arma para acabar con los soldados. Las armas tienen el único factor de interfaz que se utiliza en JurassicQuestVR. En estas se utiliza un *crosshair* que tiene dos propisitos, indicarle al jugador que se trata de un objeto interactuable y, que, depende del color está en rango de agarre o no.

*Figura 28. Arma no agarrable*

*Figura 29. Arma agarrable*

En esta escena, además se introducen los conceptos de *boundaries* o paredes invisibles. El mapa utilizado es muy grande y el objetivo de estas paredes es contener al jugador dentro de una zona de juego determinada. Con esto se pretende que el jugador tenga cierta libertad de movimiento y exploración, pero dentro de unos límites razonables del área de juego.

*Figura 30: Pared invisible*

### 4.6.5. Protection

Protection es el último nivel jugable de JurassicQuestVR este empieza donde finaliza el nivel anterior. Se ha escogido esta localización en específico para este nivel debido a su estética (incluye una cascada, un lago y montañas). Esta escena introduce tres elementos principales, raptores, un sistema oleadas y un sistema de *spawn*.

Los raptores, a diferencia de los soldados, siempre están en modo combate, por lo que desde el momento que aparecen, persiguen y atacan al jugador. Los raptores tienen 3 disparos de vida, y una precisión del 100% y para poder atacar al jugador deben estar a menos de 6 metros del mismo.

*Figura 31: Raptor I*

Respecto a las oleadas, estas funcionan de la siguiente manera. Cada 30 segundos se empezará a generar una nueva oleada. Los raptores aparecerán en los primeros 20 segundos de cada oleada separados temporalmente de forma uniforme. La primera oleada tendrá 3 raptores y a partir de ahí, el número de raptores por olea se incrementará en 1 cada dos oleadas.

A modo de ejemplo…

| | **Oleada 1** | **Oleada 7** |
|---|---|---|
| **Minuto de aparición** | 0:30m | 3:30m |
| **Raptores** | 3 | 6 |
| **Tiempo de respawn entre raptores** | 20/3 = 6.66s | 20/6 = 3.33s |

*Tabla 8. Lógica de oleadas*

Adicionalmente, este nivel también limita el área de movimiento con un *boundary* obligando al jugador a permanecer cerca del área inicial e impidiéndole que se posicione en zonas ventajosas que obligarían a los raptores a venir en una única dirección.

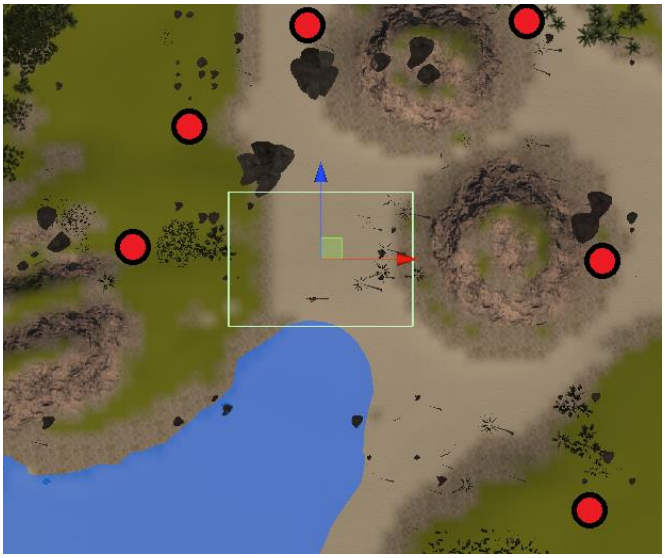


*Figura 32: Puntos de aparición*

### 4.6.6. _TestingArea

_TestingArea corresponde a la escena de pruebas que se utiliza para añadir o modificar funcionalidades en el entorno. Esta actúa a modo de *sandbox* ya que proporciona un entorno donde se puede probar libremente cualquier script, modelo o animación, de forma aislada, antes de introducirlo en las escenas de producción.

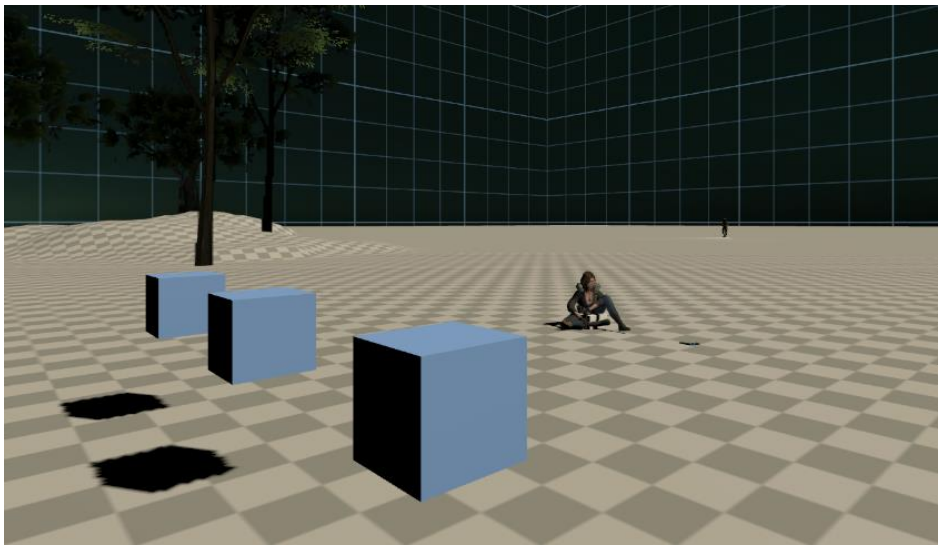


*Figura 33: Escena de pruebas*

## 4.7. Scripts

La programación del juego se lleva a cabo a través de diversas piezas de código que han sido generadas como scripts que se han agregado a los diferentes componentes de los que se compone JurassicQuestVR. Seguidamente, vemos un listado de los scripts, dividido por categorías, junto con un breve resumen de su funcionalidad.

### 4.7.1. Jugabilidad

#### *GameManager*

El GameManager es el script que controla los aspectos que son transversales en todas las escenas del juego, como por ejemplo el guardado del progreso o las opciones de sonido. Este es un *singleton* al cual acceden todas las escenas a través de su LevelManager.

#### *LevelManager*

El LevelManager es un script base que se encarga de comunicarse con el GameManager para invocar las funciones de guardado, activación de pantallas de carga, misión superada y misión fallida. Este script, adicionalmente, almacena variables que comparten todos los niveles, como sería el caso de los enemigos asesinados. El LevelManager es el único elemento de las escenas que se comunica con el GameManager.

#### *MagmaCorpLogic*

Este script es el encargado de implementar la lógica del primer nivel. Es una extensión de LevelManager y se encarga de monitorizar de forma activa si el jugador llega al portal o no.

#### *EndTrigger*

Este script está asociado al portal del primer nivel. Cuando el jugador entra en el *trigger* del portal (ha superado la partida) este script se comunica con MagmaCorpLogic para indicarle que el jugador ha llegado al portal.

#### *TeamAlphaLogic*

TeamAlphaLogic extiende a LevelManager y es el encargado de monitorizar la lógica del segundo nivel. Va monitorizando el cumplimiento de condiciones, como un determinado número de enemigos asesinados o que el jugador se encuentra a cierta distancia de Laura, y según se van cumpliendo va desbloqueándole al jugador los siguientes objetivos del nivel. Cuando se cumplen todos los objetivos, comunica al GameManager la superación del nivel.

*ProtectionLogic*

Finalmente, ProtectionLogic nuevamente extiende a LevelManager y es el encargado de implementar toda la lógica del tercer nivel, incluyendo el *spawn* de las oleadas. El script recibe como parámetros factores como el número de enemigos de la primera oleada, el tiempo entre oleadas y el número de oleadas. A partir de ahí se realizan los cálculos y se generan los enemigos. El código también se encarga de reproducir los diálogos de forma secuencial según va avanzando el tiempo en el nivel.

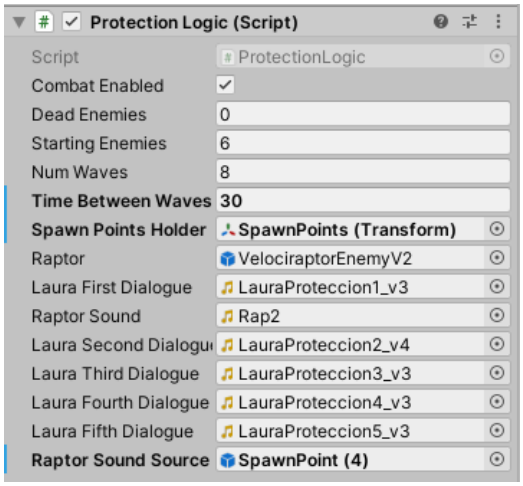



*Figura 34: ProtectionLogic (Script)*

### 4.7.2. Interfaz

*UpdateMainCanvas*

Este script es el encargado de actualizar el canvas principal permitiéndole al jugador acceder a nuevas misiones según progresa en el juego. Cada vez que se carga la escena del menú principal, el script accede a la variable que almacena el progreso del GameManager y activa o desactiva las misiones.

*MainCanvasPanelDisplay*

Esta clase es la encargada de mostrar un panel u otro en el menú principal. Cuando se abre el juego por primera vez, el jugador verá el panel principal de opciones. No obstante, cuando el jugador vuelve al menú principal desde uno de los niveles jugables, se le muestra el panel de selección de nivel.

*Quest*

Quest es un objeto scripteable que permite generar misiones desde la UI de Unity de forma sencilla. Cada Quest tiene un título, unos objetivos, una historia, una imagen y un ID de misión.

### *DisplayQuest*

Este script actualiza la misión a mostrar en el panel de misión en función del ID del subpanel seleccionado. DisplayQuest carga de forma dinámica en la interfaz los parámetros de la misión seleccionada.

### *SceneIndexes*

Enumerador que asocia el nombre de las escenas con su ID de escena correspondiente. Esto permite seleccionar las escenas más fácilmente desde otros scripts ya que no tenemos que ir recordando el ID de las mismas para acceder a ellas.

### *InitializeObjects*

Script utilizado en los paneles de interfaz donde hay multiples opciones que utilizan el mismo canvas. Este script hace que cada vez que se acceda a una interfaz de este estilo, siempre aparezca la misma opción activada. Un ejemplo de esto es en el menú de pausa, donde siempre que se pause el juego aparecerá el menú de controles y el menú de sonido estará desactivado.

### *GameManagerLoader*

Este script simplemente se utiliza como referenciador del GameManager desde los objetos de interfaz. Esto es especialmente útil con los objetos de interfaz que están desactivados por defecto (como el canvas del menú de pausa). El GameManagerLoader nunca está desactivado, por lo que al principio del nivel inicializa su referencia con el GameManager, luego todos los elementos de la interfaz que necesiten comunicarse con el GameManager, lo harán a través del Loader.

### *DefeatReason*

Pequeño fragmento permite customizar el mensaje que se le muestra al jugador cuando este fracasa en la superación de un nivel.

### *PauseMenu*

Esta clase se encarga de manejar la lógica del menú de pausa. Cuando se detiene el juego, ya sea por una pausa manual del jugador, una derrota o una victoria, este script es el responsable de mostrar la pantalla correspondiente. Adicionalmente, también se encarga de pausar los AudioSource correspondientes a los diálogos, ya que, recordamos que los AudioSource son independientes de la escala temporal.

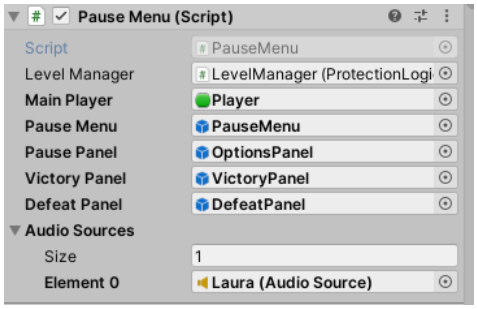



*Figura 35: PauseMenu (Script)*

*FadeOnCollision*

Script que pone el negro el visor del jugador cuando este colisiona contra un objeto. Esto se utiliza para cuando el jugador camina físicamente en su entorno real y entra, por ejemplo, a una pared dentro del juego.

*QuitOnClick*

Código que cierra la aplicación al oprimir “Salir”.

### 4.7.3. Sonido

*buttonFX*

Clase añadida a las interfaces que en función de la acción realizada (*hover*, *click* o retroceso) reproduce un sonido u otro.

*SoundSettings*

Clase serializable que permite alterar las opciones de sonido gráficamente desde el GameManager. Adicionalmente, los valores de las opciones de sonido se guardan en disco para que el jugador no tenga que modificarlas cada vez que entra al juego.

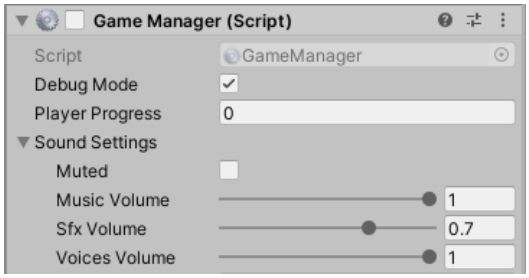



*Figura 36: Opciones de sonido*

*UpdateAudio*

Script que actualiza en tiempo real el sonido del juego cuando el jugador lo modifica desde las opciones. Además, cuando se cierra el menú de audio, este script graba los valores las opciones de sonido seleccionadas por el jugador en el disco.

*GameMusic*

Clase encargada de reproducir los sonidos de victoria o derrota.

### 4.7.4. Entidades

*Player*

Script asociado al objeto jugador y monitoriza su vida. Este contiene una función "PlayerHit" que recibe una cantidad de daño y resta vida al jugador en consecuencia. Cuando la vida llega a 0, se le manda un mensaje al LevelManager para indicar que el jugador ha perdido.

*Laura*

El personaje de Laura actúa como una marioneta manejada por el LevelManager. El script Laura, permite controlar el modelo y el audio source de Laura de forma remota. Este implementa funciones de habla, recorrer un camino, rotar, levantarse, además de funciones Get para saber si el modelo está realizando una acción en un momento determinado.

*Enemy*

Similar al caso del script Player, Enemy es el script que controla la vida de los enemigos. Además, esta clase permite determinar las opciones de combate del jugador (daño, precisión, tiempo de enfriamiento entre ataques, …) e implementa funciones para determinar si el jugador está en línea de visión o detectar si el jugador ha entrado en contacto con el enemigo.

*Soldier*

Soldier es el script encargado de implementar la lógica de los soldados. Este hereda del script Enemy e implementa un modo patrulla en el que el soldado constantemente chequea si el jugador está en su campo visual, un sistema de detección y alerta, y la lógica del modo combate. Adicionalmente, este script interactúa con el *animator* del soldado para animar el modelo acorde a las acciones que realiza.

#### *Raptor*

Equivalente al script Soldier, el script Raptor se encarga de implementar la lógica de combate de los raptores. Maneja factores como el aviso del raptor cuando está a 100 metros del jugador, la mecánica de ataque en salto y ataque cuerpo a cuerpo, además de controlar el daño al jugador se realice una vez la animación de ataque haya finalizado.

### 4.7.5. Animación

#### *AnimatorActivateBoolAtEntrance*

Script que realiza un set a "verdadero" de una variable booleana del *AnimatorController* al empezar una animación. Este se utiliza para poner la variable "AttackReady" a verdadero cuando un raptor se encuentra en estado *idle* o en carrera.

#### *AnimatorResetBoolAtEnd*

Script que realiza un set a "falso" de una variable booleana del *AnimatorController* al finalizar una animación. Este se utiliza para monitorizar cuando ha finalizado una animación de ataque, cuando ha terminado un giro, etc.

#### *AnimatorResetFloatAtEnd*

Script que realiza un set a "0" de una variable de tipo *float* del *AnimatorController* al finalizar una animación. Esto se utiliza al animar los giros de los soldados, ya que en función del ángulo que se pase como parámetro al *animator*, este realizará una animación u otra.

### 4.7.6. Armas

#### *ShootIfGrabbed*

Esta clase implementa la función de disparo, el aviso de enemigos cercanos, la posición del arma cuando el jugador la agarra y, además, hace que los controles vibren cuando se produce oprime el gatillo.

*ProjectileDestroy*

Este script se añade a los objetos correspondientes a la bala y al casquillo. Cuando uno de estos objetos tiene un tiempo de vida mayor a 20 segundos o colisiona contra otro objeto, se activa este script haciendo que la bala o casquillo se destruyan a los pocos segundos.

*EnemyImpact*

Por último, este script se ha añadido a las balas de modo que, si colisionan con un *trigger* de un enemigo, invoquen a la función de restar vida del enemigo contra el que ha colisionado.

## 4.8. Inteligencia Artificial

### 4.8.1. Soldados

Para la inteligencia artificial de los enemigos, se ha empleado dos mecanismos. En primer lugar, para el modo patrulla de los soldados, se ha codificado un sistema de co-rutinas que permiten a un GameObject seguir un recorrido determinado. Cuando el soldado llega a uno de los puntos de control del recorrido, espera por un tiempo determinado (5 segundos por defecto) y procede a rotar y a desplazarse hasta el siguiente punto de su recorrido.

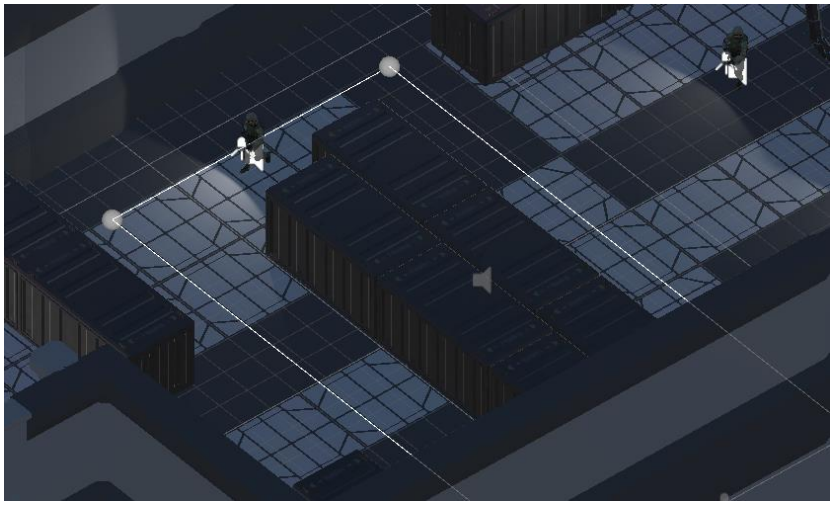


*Figura 37: Rutas de patrulla*

En segundo lugar, cuando los soldados abandonan el modo patrulla y entran en modo de alerta, estos se detienen y se ponen a mirar al lugar donde han detectado al jugador. Desde este momento, el jugador tiene 1 segundo para salirse del rango de visión del soldado. Si este se sale a tiempo, el soldado vuelve al modo patrulla, si no, entra en modo combate.

*Figura 38: Soldado en alerta*

Por último, tenemos el modo de combate. Un enemigo puede entrar en modo de combate por diversos motivos. En el caso de los soldados, puede ser por tener al jugador más de 1 segundo en rango de visión, por entrar en contacto con el jugador (aunque el soldado este de espaldas), tras recibir un disparo o tras escuchar un disparo a menos de 200 metros de distancia.

Un soldado en modo combate tiene la linterna en rojo y perseguirá y disparará al jugador. Para que el soldado pueda efectuar un disparo tiene que encontrarse a 50 metros o menos del jugador y tiene que tenerlo en rango de visión. Para calcular si el jugador está en línea de visión del soldado, el soldado lanza un *raycast* contra el jugador, si este impacta, está en visión. En caso de que el jugador no esté en visión o se encuentre a más de 50 metros, el soldado se desplazará hacia el jugador hasta que se cumplan ambas condiciones.

Para desplazar a los enemigos por el mapa, se ha utilizado un NavMesh. Este realiza un *baking* previo sobre el escenario en cuestión delimitando las zonas por las que los enemigos (NavMeshAgents) pueden pasar o no, según unos parámetros indicados. En la siguiente captura, podemos ver en color azul las zonas por las que pueden moverse los soldados tras localizar al jugador.

*Figura 39: NavMesh I*

Adicionalmente, tras efectuarse un ataque, entran en juego factores como la precisión y el daño. Cuando el soldado dispara, calcula un numero aleatorio entre el 1 y el 100, si este es menor al porcentaje de precisión del soldado, se considera un impacto y se le procede a restar al jugador tanta vida como valor de daño tenga el soldado.

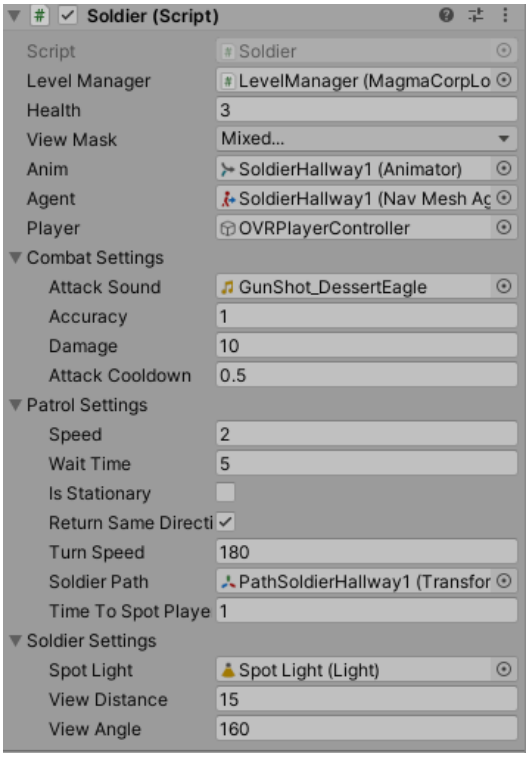



*Figura 40: Soldier (Script)*

### 4.8.2. Raptores

Los raptores siguen el mismo principio que los soldados cuando están en modo combate. Un raptor siempre irá a atacar al jugador, no importa si lo ha visto o no. Los raptores también se guían utilizando un NavMesh.

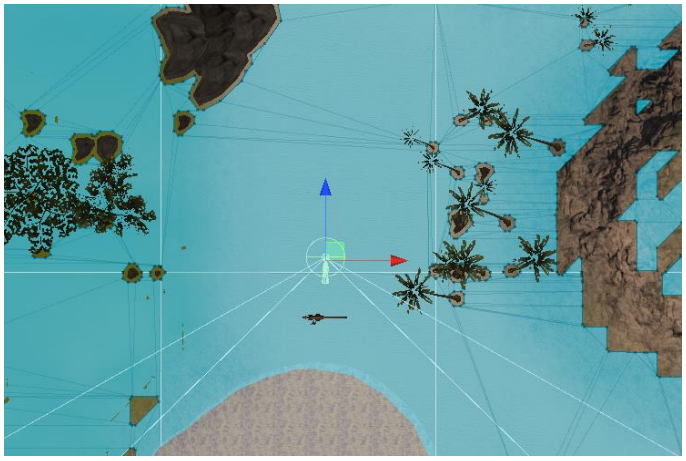


*Figura 41: NavMesh II*

A diferencia de los soldados, los raptores tienen una precisión del 100% y hacen tanto daño como vida tiene el jugador. Cuando un raptor se acerca a menos de 50 metros de distancia del jugador, este emite un sonido de aviso. Esto se ha hecho para que el jugador tenga un tiempo mínimo de reacción cuando le atacan por la espalda.

Adicionalmente, para poder atacar al jugador, los raptores tienen que encontrarse a menos de 6 metros de él. La animación de ataque que utilicen dependerá del estado del raptor dentro de su *animator*. Cuando un raptor está corriendo realizará un ataque con salto, sin embargo, si este está quieto en el suelo alternará entre dos tipos de ataque cuerpo a cuerpo de forma aleatoria.

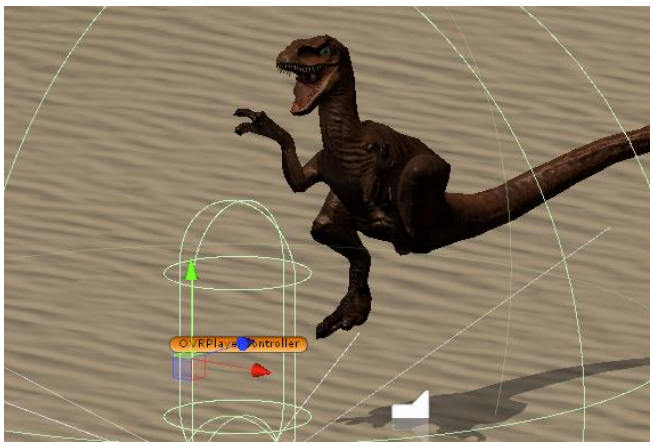


*Figura 42: Ataque en salto*

## 4.9. Animator Controllers

Para las animaciones de Laura, los raptores y los soldados, se han utilizado *Animator Controllers*. Estos son un asset nativo de Unity que permite manipular una serie o secuencia de animaciones que se asocian a un modelo 3D.

### 4.9.1. Soldado

Los soldados cuentan con un *animator controller* que se divide en tres máquinas de estados. Como vemos en la siguiente imagen, un soldado se encuentra por defecto dentro de la máquina de estados "PatrolStateMachine". Está implementa las animaciones de caminar y girar.

Cuando el jugador es visto por el soldado, sale al *Animator Controller* principal donde espera apuntando hacia el jugador. Si no se confirma la visualización del jugador, el controlador saltará de nuevo al modo patrulla. Pero si se confirma, accederá a la máquina de estados de combate ("CombatStateMachine"). Esta máquina implementa las animaciones de disparo, carrera, recibir disparo, etc.

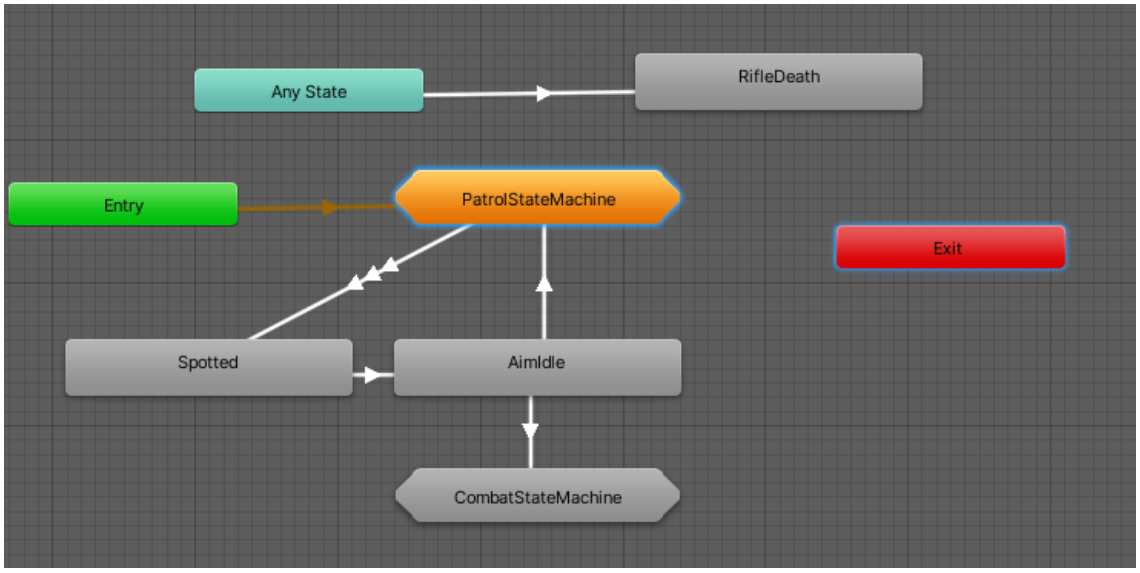



*Figura 43: Soldado (AnimatorConstroller)*

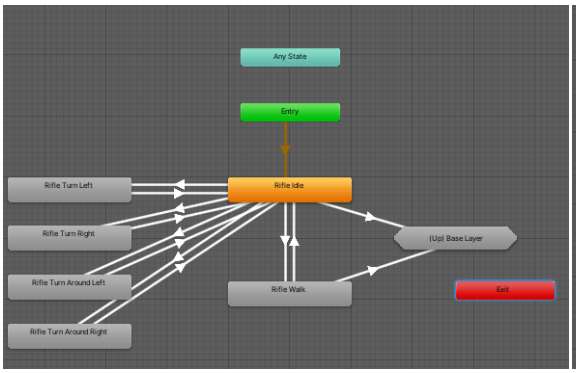



*Figura 44: PatrolStateMachine*

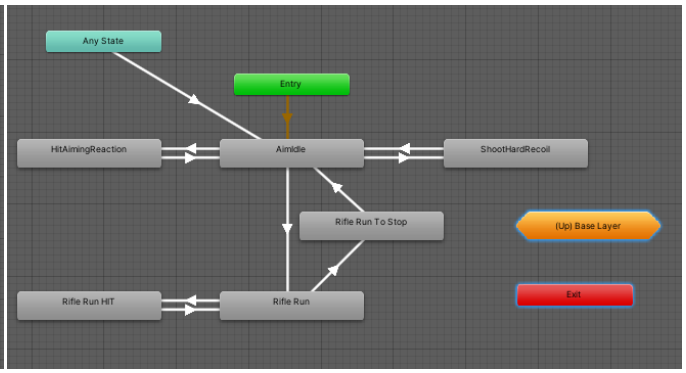



*Figura 45: CombatStateMachine*

### 4.9.2. Raptores

El *Animator Controller* de los raptores utiliza una única máquina de estados que permite al raptor realizar animaciones de gruñir, correr, ataque en salto y ataque cuerpo a cuerpo.

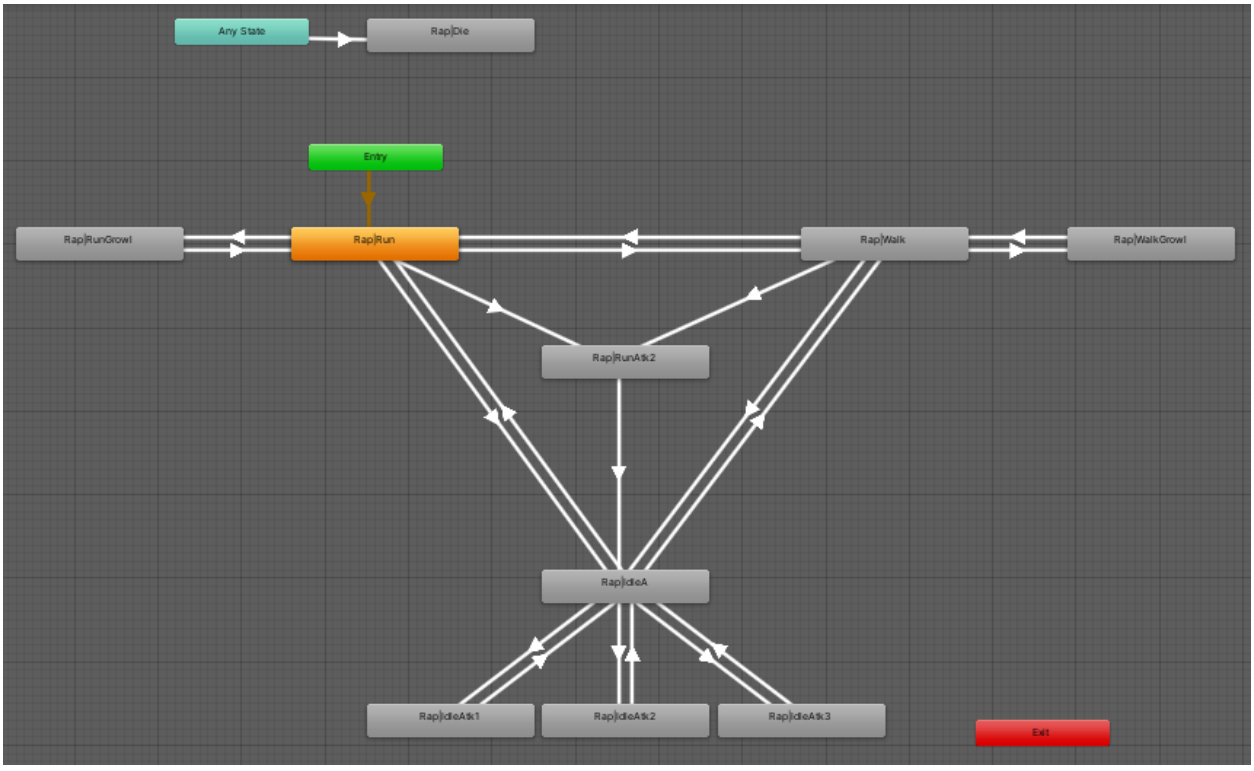



*Figura 46: Raptor (AnimatorController)*

### 4.9.3. Laura

Finalmente, el *AnimatorController* de Laura, es el más sencillo de todos. Este implementa las animaciones de estar sentado, levantarse, *idle*, correr y hablar.

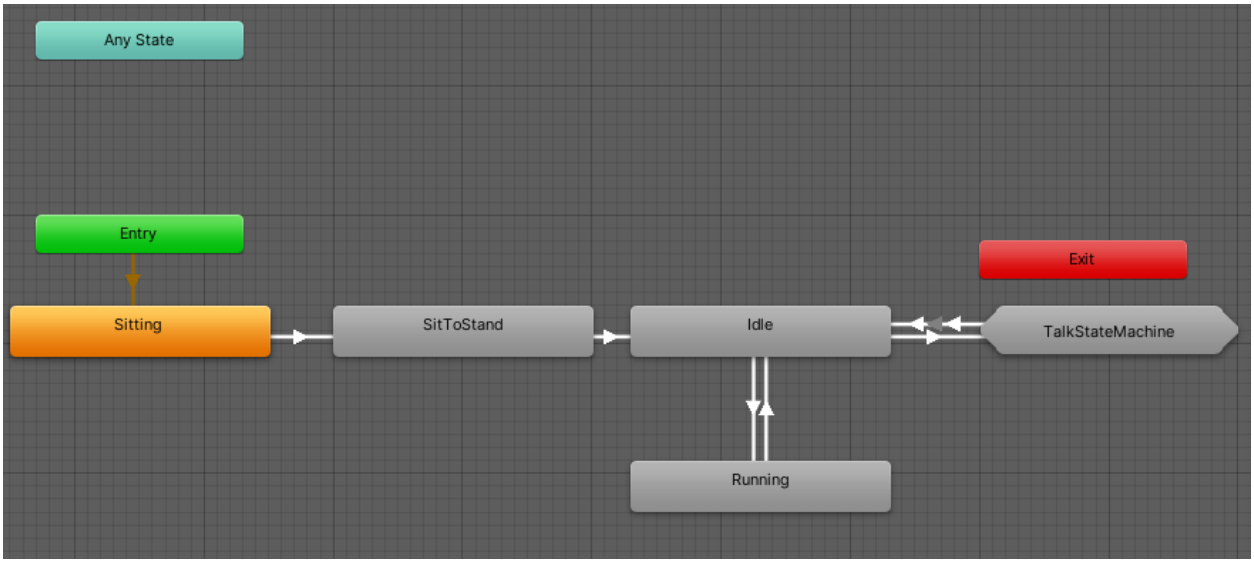



*Figura 47: Laura (AnimatorController)*

Como hemos visto en la anterior captura la animación de hablar está compuesta por una máquina de estados "TalkStateMachine". Esta va alternando diferentes animaciones de habla y además intercala estados *idle* entre ellas para no estar repitiendo siempre el mismo movimiento de brazos, ya que esto se vería poco realista.

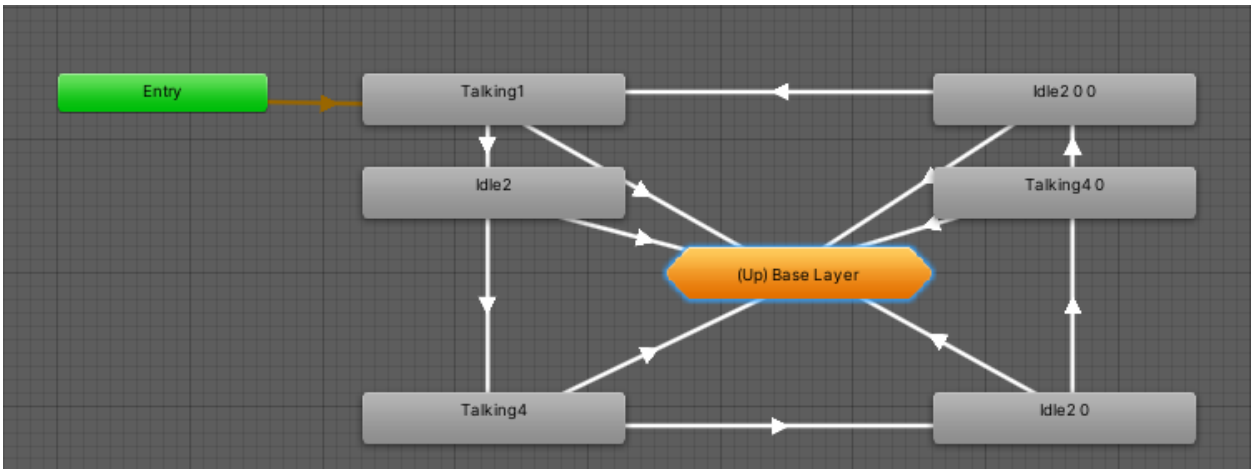



*Figura 48: TalkStateMachine*

# 5. Diseño de niveles

## 5.1. MagmaCorp

MagmaCorp es un nivel pequeño que requiere alcanzar un portal de forma sigilosa. Para superarlo se deberán aprovechar tiempos y posicionamientos específicos que nos permitan llegar hasta el final sin ser detectados.

En la siguiente captura, podemos ver en verde la posición inicial del jugador; representados con números podemos ver a los soldados; y, el recuadro rojo final es el objetivo: el portal.

Hay dos factores cruciales para superar el nivel, en primer lugar, hay que dejar que el soldado número 1 pase por delante del pasillo inicial para colocarnos en el primer círculo naranja. Ahí deberemos esperar a que el soldado #2 finalice su primer recorrido, se dé la vuelta y vuelva a su posición inicial. El saliente formado por los contenedores actúa de cobertura y evita que el soldado #2 vea al jugador.

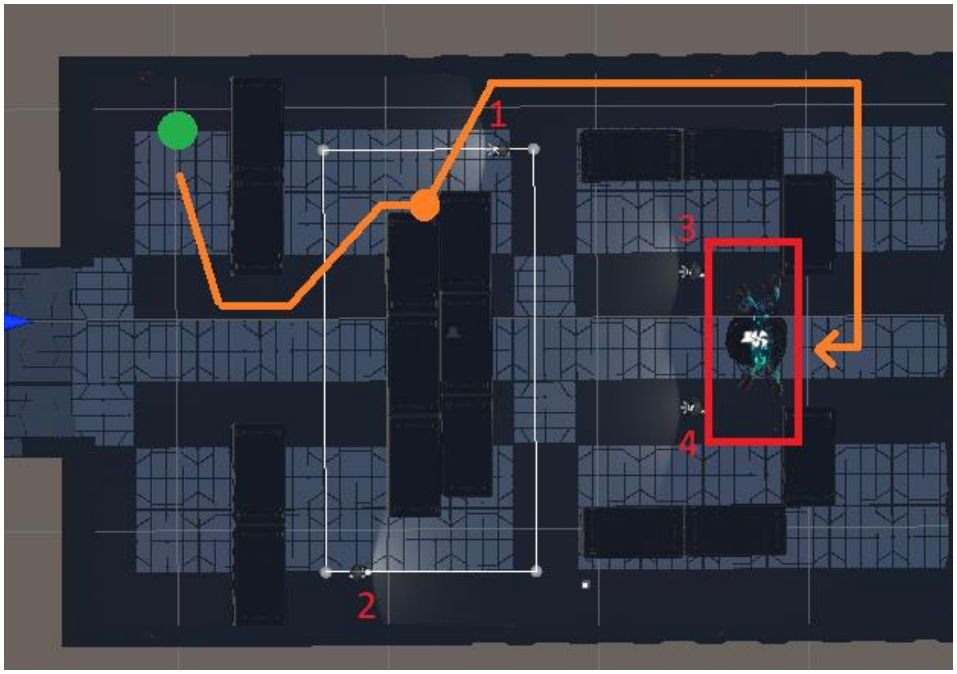



*Figura 49: MagmaCorp (Guía)*

Una vez el soldado #2 se mueva deberemos movernos rápidamente, ya que dispondremos de tan solo unos pocos segundos hasta que entremos en visión del soldado #1. El jugador deberá avanzar siguiendo el camino naranja, lo más pegado posible a la pared, ya que eso evitará que se entre en rango de visión del soldado #3.

La única forma de llegar al portal sin ser detectado es entrar desde atrás.

## 5.2. Equipo Alpha

Equipo Alpha es el nivel más extenso del juego y está dividido en 3 fases. En la primera de ella el jugador deberá liberar a Laura de los soldados de MagmaCorp que la tienen retenida. Para ello, el jugador deberá coger el arma que se encuentra en el primer punto azul y eliminar a los soldados #1, #2 y #3. Es importante disparar al soldado #1 estando el jugador posicionado antes de la línea roja, ya que los disparos realizados más adelante alertarán a los soldados #2 y #3. Una vez los soldados han sido abatidos, el jugador deberá acercarse a Laura (segundo punto azul).

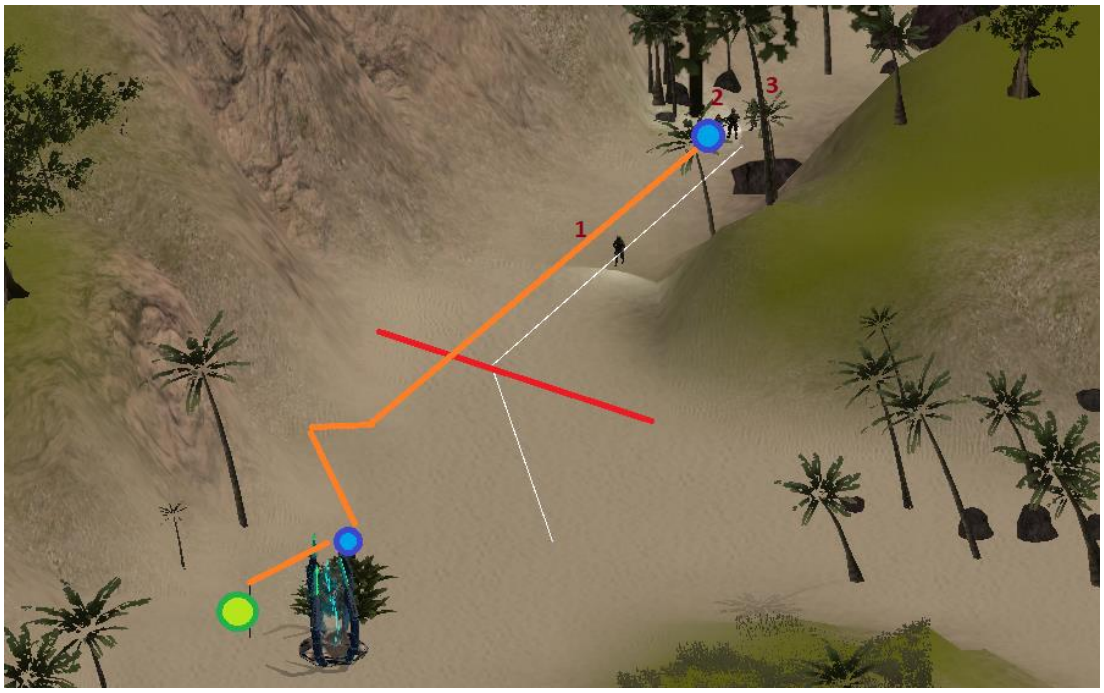



*Figura 50: Equipo Alpha (Guía I)*

La segunda fase es un mero trámite en el que Laura explica sus descubrimientos acerca de cómo detener el KelsoR. Cuando ella termina de hablar, guiará al jugador hasta la zona donde se encuentra el "Sujeto 0". Es muy importante hacer caso a Laura y no dejar la pistola, ya que en la siguiente zona la vamos a necesitar.

*Figura 51: Equipo Alpha (Guía II)*

Finalmente, cuando llegamos donde se encuentran las rocas, deberemos abatir a los soldados #4 y #5. Para eliminarlos podremos utilizar las rocas como cobertura y así impedir que nos disparen. Una vez eliminados Laura se acercará al raptor y se terminará el nivel.

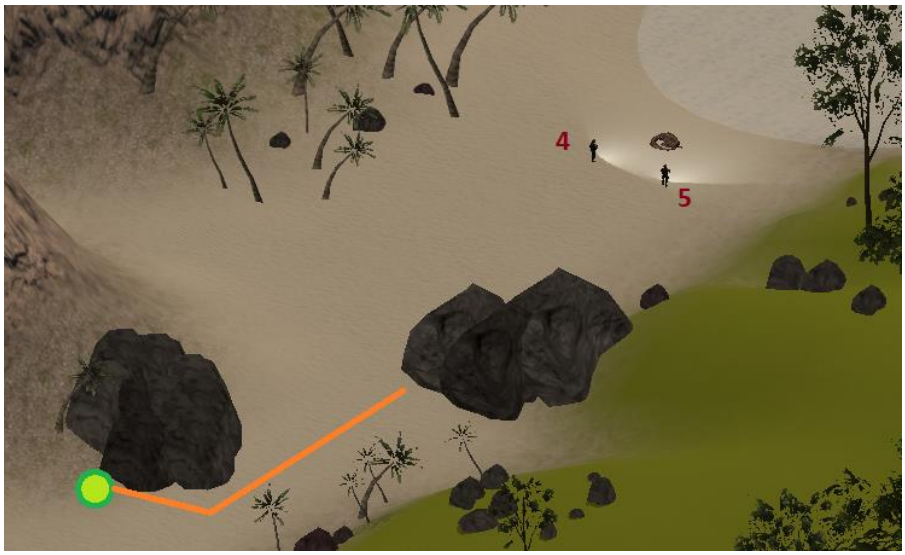



*Figura 52: Equipo Alpha (Guía III)*

## 5.3. Protección

El último nivel de JurassicQuestVR es: Protección. En esta pantalla, el jugador se encuentra donde el "Sujeto 0" y debe proteger a Laura mientras ella realiza la extracción. Durante este proceso, irán apareciendo oleadas de raptores desde puntos de aparición aleatorios. Para sobrevivir al nivel, el jugador deberá coger el arma y evitar que los raptores le lleguen a atacar.

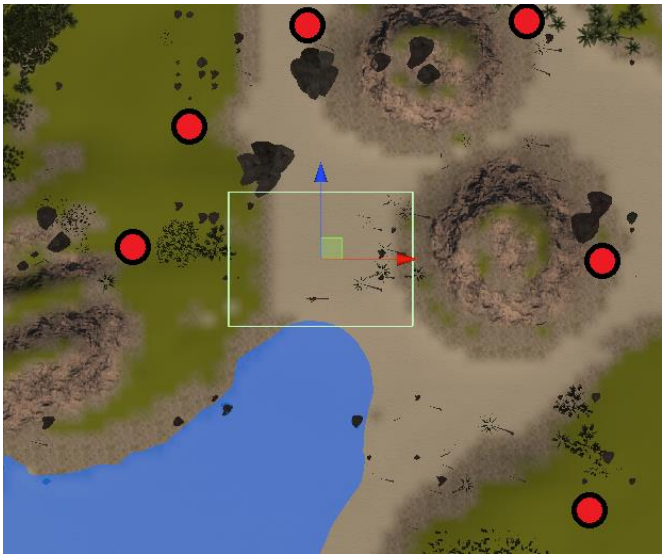


*Figura 53: Protección (Guía)*

El jugador deberá sobrevivir durante los 5 minutos de duración del nivel. Este concluirá cuando el jugador haya matado al último raptor.

# 6. Manual de usuario

## 6.1. Requisitos mínimos

- Oculus Quest
- 4GB Disponibles

## 6.2. Controles

Para desplazarse e interactuar con el entorno de JurassicQuestVR se utilizarán los controles de Oculus Quest. Las acciones disponibles son las siguientes:

- Botón Start: Pausar/Reanudar
- Joystick izquierdo: Desplazarse
- Joystick derecho: Rotar
- Gatillo de mano (izda y derecha): Agarrar objeto
- Botón X/A: Soltar objeto
- Gatillo de índice (izda y derecha): Disparar

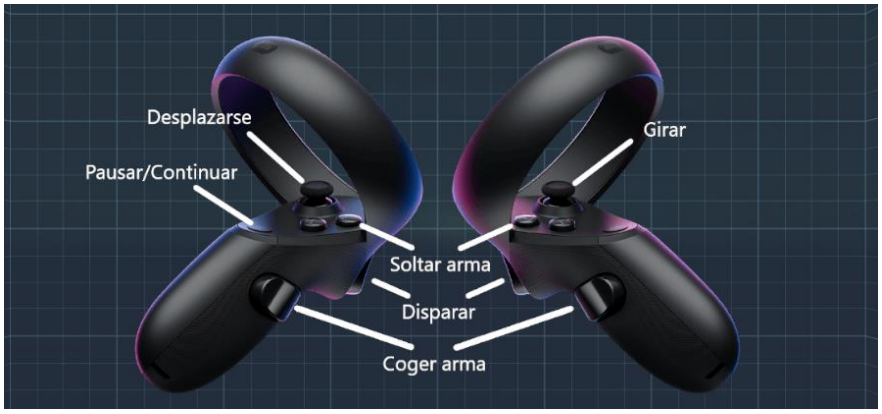



*Figura 54: Controles de OculusQuest*

## 6.3. Enemigos

En JurassicQuestVR encontraremos dos tipos de enemigos principales. Por un lado, tenemos los soldados de MagmaCorp y por otro los raptores. A continuación, se presentan sus fichas técnicas.

### 6.3.1. Soldados

| Ficha técnica | |
|---|---|
|  | **Salud: 5HP**<br>**Daño: 20%**<br>**Precisión: 50%**<br>**Distancia de ataque: 50m** |

*Tabla 9: Ficha técnica del soldado*

### 6.3.2. Raptores

| Ficha técnica | |
|---|---|
|  | **Salud: 3HP**<br>**Daño: 100%**<br>**Precisión: 100%**<br>**Distancia de ataque: 6m** |

*Tabla 10: Ficha técnica del raptor*

# 7. Usabilidad e integración social

El objetivo de JurassicQuestVR es que el jugador se sumerja por completo en la historia del Agente 01. No obstante, para crear una experiencia lo más cercana posible a la realidad, el juego cuenta con unos detalles que buscan compaginar comodidad y practicidad.

## 7.1. Agarre a distancia

Como se ha explicado en anteriores apartados de este documento, el jugador dispone de un mecanismo que le permite agarrar objetos que se encuentren a 3 metros de distancia de él. Esta funcionalidad se ha añadido por motivos de accesibilidad y comodidad.

Pese a que esta funcionalidad representa un comportamiento no realista, se ha considerado un punto necesario ya que se presupone que algunas personas que jueguen no van a poder (o querer), agacharse, ya sea por limitaciones fisiológicas o de su entorno de juego.

## 7.2. Tiempo bala

Otro detalle que se aleja un poco de la realidad es que al disparar una pistola en el juego uno puede ver brevemente la trayectoria de la bala. Este comportamiento se ha implementado para suplir la carencia de una mirilla digital. El hecho de ver hacia donde sale la bala es de gran ayuda para el jugador a la hora de apuntar el arma.

## 7.3. Munición ilimitada

Continuando con temas del arma, otra funcionalidad introducida es la de la munición ilimitada. Esto se ha hecho principalmente por tres motivos.

El primero es que se cuándo se acabase la munición, el jugador debería de coger un cargador e introducirlo en el arma, y dado que el jugador no tiene modelo 3D esos cargadores tendrían que aparecer de la nada.

El segundo es relacionado al movimiento del jugador, ya que, para recargar la pistola, el jugador debería de ayudarse de ambas manos para realizar el procedimiento. Al probar esta funcionalidad, el movimiento de recarga terminaba siendo repetitivo y se hacía más pesado que divertido.

Por último, el tercer motivo es la propia diversión del juego. El hecho de contar con munición ilimitada permite que el jugador tenga más libertad, ya que no tiene que estar constantemente pendiente de recargar, lo que facilita la creación de niveles más intensos y con más enemigos, como es el caso de las oleadas de Protección.

## 7.4. Combatiendo la cinetosis

Uno de los aspectos más tediosos de la realidad virtual es que esta puede llegar a producir disconformidad e incluso nauseas en el jugador. No obstante, hay algunas técnicas que el programador puede utilizar para mitigar o atenuar este efecto.

### 7.4.1. Relación entre aceleración y espacio

La principal causa de cinetosis en ambientes virtuales viene relacionada con la velocidad a la que se mueven los elementos del entorno. Esto pude experimentarlo de primera mano al determinar la velocidad a la que se le permite mover al jugador. Pantallas más pequeñas o ambientes más cerrados, son más propensas a generar mareos ya que los objetos entran y salen con más facilidad de la cámara.

Un paralelismo con la vida real sería cuando uno mira por la ventana en un coche. Si uno se intenta enfocar en los elementos más cercanos a la ventana y que pasan a mayor velocidad, uno se marea mucho antes que si por el contrario observa elementos en la lejanía.

Por este motivo, al jugador se le ha dado velocidades distintas en los mapas de interior y en los mapas de exterior. En MagmaCorp la aceleración del jugador es de 0.075, mientras que en Equipo Alpha y Protección es de 0.35. Pese a ser una diferencia muy grande, debido a la diferencia en el tamaño de los mapas y al tipo de misiones que entablan (infiltración versus combate), el jugador no percibe que camine mucho más rápido o lento y la sensación de disconformidad disminuye drásticamente.

### 7.4.2. Cortina en negro

Otro de los factores que generan cinetosis es la disparidad de movimiento entre el cuerpo real y el virtual. Esto no suele ser problema ya que el movimiento en el entorno real se traduce a un movimiento en el virtual, no obstante, ¿Qué pasa cuando en un entorno virtual tenemos una pared delante y no la tenemos en el real?

Normalmente, en un juego 3D convencional, se generaría una colisión entre el jugador y la pared virtual y no podría avanzar. No obstante, esto no es nada recomendable realizarlo en un ambiente virtual, ya que el cuerpo del jugador real seguiría avanzando, pero su campo visual seguiría siendo el mismo.

Para suplir esta función, se ha añadido un *trigger* en la cámara del jugador, de modo que cuando esta detecte lo que sería una colisión, este llame al OVRScreenFade del APK de Oculus. Esta clase nos permite poner la visión del usuario en negro, de modo que mientras este dentro de la pared virtual, este no vea nada. Entonces, instintivamente, el jugador caminará hacia atrás y volverá a colocarse en la zona de juego.

### 7.4.3. Giros laterales

Por último, otro aspecto que generaba malestar en el jugador era al realizar giros con el joystick. Este caso es precisamente el contrario al anterior, ya que aquí el cuerpo está quieto y es la visión la que se mueve. Rotar con el joystick de una forma lineal genera muchísima cinetosis, ya que es un movimiento que el cerebro lo interpreta como natural, no obstante, el cuerpo no lo está realizando y se produce una discordancia entre la imagen procesada y la sensación percibida.

Para corregir este aspecto, se ha implementado una mecánica de rotación en bloque. Cuando el jugador utiliza los *joysticks* para rotar de izquierda a derecha, o viceversa, cada vez que lo oprime se realiza un giro de 45 grados. Este movimiento el cerebro no lo interpreta como un movimiento natural, sino que simplemente lo interpreta como una imagen nueva totalmente distinta a la anterior, lo cual elimina prácticamente toda sensación de malestar.

## 7.5. Diversidad e integración social

Como se ha repetido numerosas veces en este documento, el principal objetivo de JurassicQuestVR es proporcionarle al jugador una experiencia inmersiva que le haga sentir que está viviendo la historia y que sienta que él es Agente 01.

Precisamente, este es el motivo por el que en la historia nunca se especifica una descripción física del Agente 01, en el juego siempre se le reconoce como “agente” y además carece de un modelo 3D. El objetivo tras este detalle es que cualquier persona que juegue a JurassicQuestVR pueda empatizar y sentirse parte del juego independientemente de su edad, raza o identidad de género.

# 8. Conclusiones

En términos generales, este trabajo ha sido una experiencia muy positiva. La naturaleza compleja de un proyecto como JurassicQuestVR ha hecho que tenga que, además de tener que enfrentarme al proceso de desarrollo de un videojuego (selección de *engine*, conceptualización, desarrollo, etc.), tenga que controlar aspectos adicionales específicos de la realidad virtual.

Un juego en realidad virtual tiene una serie de elementos que ni siquiera se contemplan en un juego 3D convencional. Un ejemplo de esto es que la figura del jugador siempre debe de estar presente, tanto en menús de selección, como en pantallas de juego. Durante la ejecución de un nivel, un jugador puede agarrar un objeto y pausar el juego. Entonces, el programador debe tomar una decisión con qué hacer con el objeto agarrado. ¿Se deja caer?, ¿Se lleva al menú de pausa?, … Esta serie de cuestiones no se plantean en un entorno 2D o 3D convencionales, puesto que un menú de pausa no requiere de la presencia de las manos del jugador.

Otro factor que se ha tenido que tener en cuenta, específico al desarrollo de Oculus Quest, es la capacidad de procesamiento del hardware. Cuando se programa sobre Unity, el editor utiliza los recursos de RAM, procesador y tarjeta gráfica del ordenador, que es quien realiza toda la tarea de renderización y la envía al *headset*. No obstante, después de realizar el *build* y cargar el proyecto en el *headset*, este no tiene la misma capacidad que el ordenador, por lo que hay que minimizar aspectos como el número de polígonos o las partículas.

Factores como los anteriormente mencionados hacen que la programación de un juego en VR sea un proyecto considerablemente más complejo que el videojuego tradicional. Hay muchísimas variables que uno no se plantea que puedan ser un problema hasta que toca implementarlas.

Por suerte, Unity y sus *assets* disponibles han facilitado mucho la tarea a la hora de crear el juego. Herramientas como el APK de Oculus facilitan mucho la tarea de integración entre Unity y el *headset*, de modo que simplemente, viendo algunos tutoriales junto con la documentación de Oculus, uno pueda poner en marcha las bases de un proyecto en cuestión de minutos.

Continuando con los *assets*, el hecho de poder disponer de *assets* previamente construidos (modelos 3D, mapas, …) ha acelerado muchísimo el desarrollo del proyecto. El hecho de disponer de estas herramientas ha permitido que haya podido llevar a cabo este proyecto satisfactoriamente, ya que, si se hubiera tenido que generarlos desde cero, habría hecho falta un equipo de dos o tres personas más.

Como cierre del documento, comentar que, tras haber finalizado este proyecto, estoy muy contento con los resultados obtenidos, ya que considero que el producto final obtenido está a la altura de mis expectativas. Han sido cuatro meses que han requerido de muchísimas horas de trabajo. No obstante, esto me ha permitido mejorar drásticamente mis conocimientos de programación de videojuegos y, sobre todo, en aspectos relacionados con entornos de realidad virtual.

En cuanto al futuro de JurassicQuestVR se han dejado una serie de líneas abiertas pendientes de incluir. En primer lugar, se añadirán nuevos modos de juego como un modo historia, un modo *survival* y un modo arcade. En segundo lugar, se añadirá nuevo contenido de modo que el jugador pueda descubrir el desenlace de JurassicQuestVR. Por último, otro de los objetivos que se incluirán en futuras actualizaciones es la traducción del juego a diferentes idiomas.

# Glosario

2D

Acrónimo utilizado para referirse a entornos de 2 dimensiones., 12, 64

3D

Acrónimo utilizado para referirse a entornos de 3 dimensiones., 4, 12, 16, 23, 24, 25, 28, 31, 32, 52, 61, 62, 63, 64

*Animator Controller*

Asset nativo de Unity que permite establecer secuencias de animaciones., 31, 52, 53

APK

Acrónimo para Android Application Package, 10, 62, 64

asset

Item que puede ser utilizado dentro de un proyecto., 12, 29, 32, 52

*baking*

Compilacion previa de un tipo de datos., 49

*build*

Acción de compilar un programa para obtener su ejecutable., 27, 35, 64

cinetosis

Trastornos producidos en el organismo a causa del movimiento., 3, 62, 63

*engine*

Tambien llamado "game engine" o "gameframe", es el entorno sobre el que se construye un videojuego., 1, 8, 12, 13, 22, 26, 28, 64

*idle*

Tipo de animación utilizada para representar periodos de no actividad., 47, 54

joystick

Palanca que permite realizar acciones de movimiento o rotación., 63

NavMesh

Tambien conocida como malla de navegación, es una estructura de datos abstracta utilizada por la inteligencia artificial para encontrar un camino que enlace dos puntos., 6, 49, 50, 51

script

Fragmento de código que implementa una funcionalidad., 31, 41, 42, 43, 44, 46, 47, 48

UI

Acrónimo utilizado para referirse a la interfaz de usuario., 5, 32, 43

VR

Acrónimo utilizado para referirse a la realidad virtual., 26, 64

# Referencias

Andersen, K. (17 de noviembre de 2018). *Metal Gear Solid Review*. Obtenido de PushSquare: http://www.pushsquare.com/reviews/psone/metal_gear_solid

Brackeys. (27 de marzo de 2017). *How to make a HIGH SCORE in Unity*. Obtenido de YouTube: https://www.youtube.com/watch?v=vZU51tbgMXk

Brackeys. (3 de enero de 2018). *SCRIPTABLE OBJECTS in Unity*. Obtenido de YouTube: https://www.youtube.com/watch?v=aPXvoWVabPY

EpicGames. (s.f.). *UnrealEngine4 - FAQ*. Obtenido de UnrealEngine: https://www.unrealengine.com/en-US/faq

Fattie. (9 de marzo de 2016). *Sin nombre*. Obtenido de StackOverflow: https://stackoverflow.com/a/35891919

Game Dev Guide. (23 de diciembre de 2019). *Designing a Loading Screen in Unity*. Obtenido de YouTube: https://www.youtube.com/watch?v=iXWFTgFNRdM

HOFPG Hall of First Person Games. (27 de abril de 2018). *Carnivores 1998 PC*. Obtenido de YouTube: https://www.youtube.com/watch?v=hht0Abiw8j4

Multiples Autores. (27 de mayo de 2020). *List of Unity games*. Obtenido de Wikipedia: https://en.wikipedia.org/w/index.php?title=List_of_Unity_games

Multiples Autores. (31 de mayo de 2020). *List of Unreal Engine games*. Obtenido de Wikipedia: https://en.wikipedia.org/wiki/List_of_Unreal_Engine_games

Oculus. (s.f.). *Oculus Developer Forums*. Obtenido de Forums Oculus VR: https://forums.oculusvr.com/developer/

Oculus. (s.f.). *Oculus Documentation for Unity*. Obtenido de Developer Oculus: https://developer.oculus.com/documentation/unity/

The Mr. X Podcast. (21 de agosto de 2015). *GoldenEye 007 - 00 Agent Longplay*. Obtenido de YouTube: https://www.youtube.com/watch?v=Yqf6MNydn-U

Unity. (6 de septiembre de 2016). *Live Training 5th September 2016 - Creating a Main Menu*. Obtenido de YouTube: https://www.youtube.com/watch?v=OWtQnZsSdEU

Unity. (s.f.). *Unity FAQ*. Obtenido de Unity3d: https://unity3d.com/unity/faq

Valem. (19 de enero de 2019). *How to make a VR game - OCULUS (Series)*. Obtenido de YouTube: https://www.youtube.com/watch?v=sKQOlqNe_WY&list=PLrk7hDwk64-Y7ELKfkw8ox8TaT9y3gNpS

# Enlaces de interés

1. Repositorio de GitHub

https://github.com/carlos-lira/JurassicQuestVR

2. Google Drive con el ejecutable (requiere cuenta de la UOC)

https://drive.google.com/open?id=1c7AhRH7U8ThwF0_50bzf5-2EQ-f3Xx3a

3. Video (Gameplay completo)

https://www.youtube.com/watch?v=w1oXg8rJfqw