Universitat Oberta de Catalunya
uoc.edu

OLD GODS: LAVA

Trabajo final de Máster

Eduardo Borges Fernández
24-5-2020

Plan de Estudios del Estudiante

Área del trabajo final

Helio Tejedor Navarro

Joan Arnedo Moreno

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Old Gods: Lava*
Nombre del autor:	*Eduardo Borges Fernández*
Nombre del consultor/a:	*Helio Tejedor Navarro*
Nombre del PRA:	*Joan Arnedo Moreno*
Fecha de entrega (mm/AAAA):	05/2020
Titulación:	*Máster en diseño y desarrollo de videojuegos*
Área del Trabajo Final:	*Trabajo Final de Máster*
Idioma del trabajo:	*Castellano*
Palabras clave	*Videojuego, RPG, TFM*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El trabajo consta de la realización de un videojuego RPG de acción inspirada mayormente en World Of Warcraft, su objetivo principal dentro del juego es derrotar a un gran enemigo con una serie de mecánicas usando habilidades variadas. El objetivo del videojuego en sí es ser un producto, cuyo fin es entretener y generar emociones en el usuario final. La plataforma de uso es PC y se considera apto para todos los públicos. El resultado ha sido una versión jugable, aunque con detalles por mejorar y la conclusión ha sido que la planificación de un proyecto de esta índole es bastante más compleja y requiere de mucha experiencia y conocimientos previos.

Abstract (in English, 250 words or less):

This project is about the realization of an action RPG videogame mostly inspired by World Of Warcraft, its main objective within the game is to defeat a great enemy with a series of mechanics using various abilities. The objective of the video game itself is to be a product, the purpose of which is to entertain and generate emotions in the end user. The platform of use is PC and is considered suitable for all audiences. The result has been a playable version, although with details to improve and the conclusion has been that planning such a project is much more complex and requires a lot of previous experience and knowledge.

Índice de contenido

ÍNDICE DE FIGURAS

1. Introducción

1.1. Contexto y justificación del Trabajo

Los juegos han sido desde siempre una fuente de entretenimiento para las personas, generan en estas sensaciones de felicidad, que les permiten alejarse de sus problemas y permite que la creatividad fluya. Algunos juegos son individuales, otros en grupo, unos más inmersivos y otros más casuales, y otros tantos son improvisados por cada persona.

La implantación de estos juegos en medios informáticas son los denominados videojuegos, que han ido cada vez cogiendo más visibilidad en las últimas décadas y permiten una más fácil distribución y mayor accesibilidad para jugar entre usuarios de distintas culturas. Los videojuegos mueven grandes cantidades de dinero, en algunos países más que el cine y la música juntos, y en periodos excepciones como en el del Covid-19 en 2020, se convierte en uno de los productos más consumidos ya que es una actividad elegida por muchos para el confinamiento.

Este trabajo intenta conseguir un videojuego RPG de acción, siendo representado por un combate contra un boss con mecánicas balanceadas para crear una sensación de reto y diversión al jugador, priorizando la comodidad usando los controles.

1.2. Objetivos del Trabajo

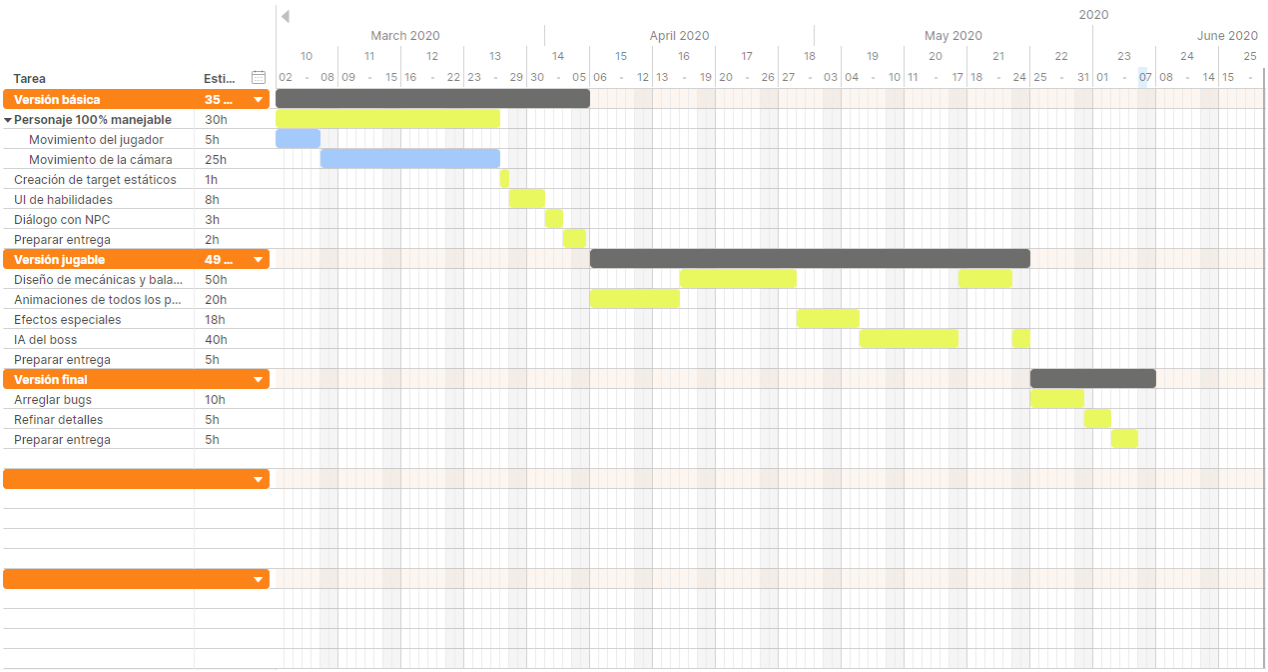
- **Balanceo en la dificultad de las mecánicas**: encontrar el equilibrio entre daño, duración y efecto de las habilidades tanto del boss como del jugador.
- **Eficiencia, Reutilización y Flexibilidad**: conseguir realizar una estructura de juego que no utilice muchos ciclos de procesamiento innecesarios, que sea reutilizable y fácilmente modificable (para facilitar el punto anterior de balanceo).
- **Fluidez de control**: aportar al usuario una capacidad de observación del entorno cómoda a través de un buen diseño e implementación de movimiento de jugador y cámara.
- **Variedad de habilidades y personalización**: permitir personalizar el jugador del usuario a través de una selección de aspecto y dotarlo de varias habilidades con distintos efectos.

1.3. Enfoque y método seguido

El desarrollo del software ha seguido la metodología en espiral, para así asumir el menor riesgo sobre el proyecto, valorando los riesgos en cada iteración y poder así replanificar el siguiente bloque de trabajo. Dado que el plazo es reducido y los recursos y el tiempo de dedicación se ven acotados por otros factores, se considera que la capacidad de replanificación de gran importancia, por lo tanto, la metodología descrita se considera la más apropiada.

Para el desarrollo del videojuego se ha optado por desarrollar desde cero los scripts del juego dado que proporcionarán el mayor ajuste para satisfacer los objetivos del apartado anterior. Mientras que para otros casos (animaciones, modelos…) por motivos de tiempo y conocimiento se ha optado por adaptarlos hasta que visualmente sean lo suficientemente satisfactorios.

1.4. Planificación del Trabajo



1.5. Breve sumario de productos obtenidos

- Sistemas generales que permiten generar nuevo contenido de manera ágil. (nuevas habilidades, gestión de sonidos, gestión de diálogos)
- Escenario visualmente correcto.
- Interfaz clara y concisa.
- Controles aceptables con mejoras pendientes.
- Efectos especiales
- Lógica eficiente, y en su mayoría robusta y flexible.
- Balanceo de juego aceptable.
- Selector de “skins”

1.6. Breve descripción de los otros capítulos de la memoria

- *Estado del arte*: se revisarán las características generales del género del juego, se hará una breve vista atrás para comprender más la procedencia de estas y finalmente se procederá a analizar títulos más actuales que tengan relación de forma significativa con el género y en los que se ha podido buscar inspiración en la fase de planificación.

- *Definición del juego:* se explicará más en profundidad la definición del juego, su contexto de historia, sus personajes y su ambientación.

- *Diseño técnico:* se abordarán los elementos de desarrollo como el entorno, los scripts desarrollados, los assets propios, modificados y aprovechados, estrategias y la arquitectura que todo esto toma en su conjunto.
- *Diseño de niveles:* se mostrará y detallarán las razones de cada elemento del entorno.
- *Manual de usuario:* se explicarán todos los controles y los pasos a seguir para no perder detalle de las acciones realizables.
- *Conclusiones:* se recopilarán todas las sensaciones, valoraciones y observaciones obtenidas durante los meses en los que se ha realizado este Trabajo Fin de Máster.

2. Estado del Arte

2.1. Revisión sobre el género de su juego

El juego objeto de este proyecto se ha desarrollado siguiendo la base de concepto para un juego RPG (*Role-Playing Game*), en el que el usuario es capaz de tomar decisiones que afectan al transcurso y evolución del mismo. En un juego de tipo RPG se suele tener características que evolucionan a medida que el usuario consigue diferentes hitos o cuando consigue determinada experiencia a partir de un sistema de niveles o escalas similares. Gracias a este sistema de progreso, el juego desbloquea características a medida que el usuario va avanzando en él, como habilidades exclusivas de su personaje, jefes más difíciles, mundos diferentes, etc.

Actualmente, establecer qué define a un videojuego RPG es una tarea difícil, puesto que la multitud de géneros híbridos que se han formado alrededor de este hacen al género mucho más amplio y diverso. En una aproximación, tal y como comentábamos en el apartado anterior, las características esenciales de un RPG son:

- Capacidad de mejorar al personaje principal en el transcurso del juego incrementando su nivel y/o características.
- Un sistema de combate basado en la selección de hechizos o habilidades que tengan diferentes efectos, apoyados en un sistema de inventario con equipamiento (armas, armaduras, pociones, etc.).
- Una misión u objetivo principal que centre el juego en una línea temporal de historia y progreso clarificado.
- Capacidad de interactuar con los diferentes elementos del entorno: personajes que controla el propio juego (NPC: *Non-Player Character*), elementos de recolección y otras criaturas (aliados o enemigos).

- Posibilidad de elegir entre diferentes clases de personaje que definan habilidades, hechizos y características distintas entre clases pero comunes entre jugadores de la misma clase (por ejemplo: mago, guerrero, paladín, etc.)

Cabe destacar que éstas son las características más usuales que se duelen dar en juegos del género, pero un RPG no tiene porque necesariamente tener todas y cada una de estas características. Los modernos híbridos RPG normalmente suelen tener algunas de las características anteriores combinadas con elementos de otros géneros.

2.1.1. RPG: del tablero al juego

Como sabemos, los juegos de rol vienen de juegos al estilo más tradicional de papel y bolígrafo, tales como Dragones y Mazmorras entre los más conocidos del género. En este tipo de juegos de mesa los jugadores dan vida a personajes de ficción que de manera continuada narrar sus acciones y toman decisiones para evolucionar en el transcurso de la partida.

FIGURA 1: Juego de tablero de Dragones y Mazmorras

En dichos juegos, el objetivo es ir tomando decisiones sobre determinadas situaciones que son planteadas e introducidas por el "*game master*", que actúa de narrador y de moderador sobre la partida. En relación a los videojuegos, la parte del *game master* está totalmente automatizada y la Inteligencia Artificial desarrollada decidirá las acciones de los diferentes personajes no controlador por jugadores (NPCs) y los enemigos (así como los diferentes elementos dinámicos del entorno).

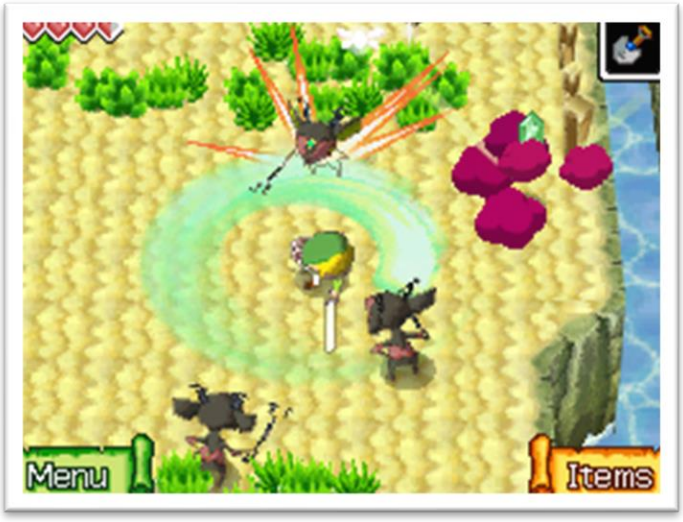


FIGURA 2: The Legend of Zelda: Phantom Hourglass en pleno combate contra la IA

En los juegos de tipo RPG más tradicionales, seguir una historia principal base bien desarrollada suele ser el aspecto más fundamental a conseguir, sin embargo este aspecto ha ido perdiendo importancia en las diferentes vertientes del género, tales como los MMORPG (*Massively Multiplayer Online RPG*). Incluso otros combinan lo mejor de una historia principal con el combate de acción en tiempo real (ARPG: *Action RPG*).

Ya conociendo el desarrollo del género, revisaremos el aspecto del arte y desarrollo y echaremos un vistazo a los juegos más tradicionales RPG, títulos tales como *Betrayal at Krondor*, lanzado en 1993 para PC; una visión general en el género MMORPG en *WoW: World of Warcraft* entre otros; y finalmente a títulos más actuales y desarrollados en motores más potentes, como *Dark Souls*.

2.1.2. Arte en tradición: Betrayal at Krondor

Vamos a introducir un título que, aunque no tuviese tanto éxito como otros, aplica conceptos muy interesantes en un RPG además de combinar mucho la narrativa con la jugabilidad, así como el combate por turnos, sistema de equipamiento y exploración de mundo abierto.

"*Betrayal at Krondor"* está basado en una serie de novelas de fantasía escritas por Raymond E. Feist desarrollado por Dynamix y publicado por Sierra On-line en el verano de 1993. La obra en la que se basa contiene alrededor de 30 libros que tratan sobre la lucha interdimensional entre criaturas mágicas y humanos hechiceros poderosos en un planeta similar a la Tierra, conocido como *Midkemia*.

La jugabilidad de este título está basada en la exploración del mundo abierto a través de una vista en primera persona del entorno mientras se viaja, combinándolo con una vista en tercera persona mientras se combate. La interfaz del usuario está controlada por ratón principalmente y por determinadas teclas del teclado.

FIGURA 3. Vista en primera persona mientras se viaja.

FIGURA 4. Vista del mapa desde perspectiva, también se usaba para moverse.

Como se puede ver en las figuras anteriores, la interfaz destaca por tener un menú siempre visible de acción y pausa, así como la vista activa de los personajes que conforman el grupo activo (el que va a combatir). El juego se desarrolla en torno a 9 capítulos y presta mucha atención en el desarrollo narrativo. Es por ello por lo que muchas de las escenas representen diferentes elementos del entorno (castillos, casas, bosques) y con distintos NPCs que combinan atuendos y variedad de diseño.

FIGURA 5. Conversación con un NPC

El objetivo del juego es ir atravesando los distintos obstáculos que se plantean, ya sean combates tácticos o resolviendo puzles y acertijos para ir avanzando en la historia principal. Consta de un sistema de inventario y armamento íntegro, cada pieza con su descripción y aspecto de historia. Incluye características interesantes como la de pociones, encantamientos, armas a distancia, cuerpo a cuerpo y reparación de equipo (un ítem se puede romper si no se supervisa), características que se pueden ver en títulos más modernos como *Dark Souls*.

Una vez dentro de una batalla, el jugador entrará en un combate por turnos, donde el propio jugador decidirá la acción de sus unidades a través del menú visual que se muestra siempre visible. Cada elemento seleccionado mostrará sus estadísticas actuales clave (vida, daño, etc.) en el remarco visible de la zona del menú, a modo de no colapsar la principal pantalla con barras de vida o similares.

FIGURA 6. Combate en activo seleccionando una unidad enemiga.

En el sistema de exploración se usan fondos dinámicos de diferentes *assets* de elementos que se van acercando, alejando y reposicionándose a medida que el personaje se mueve. La vista superior de mapa 2D simplemente es una vista de las rutas que se mueve la flecha del personaje a la par pero muy simplificado y poco detallado.

FIGURA 7. Elementos del entorno de exploración.

2.1.3. Final Fantasy: historia, combate y personalidad

Cuando hablamos de videojuegos RPG, uno de los principales títulos que se nos tienen que venir a la cabeza es Final Fantasy. La primera entrega de lo que será la saga sale por primera vez en Japón en 1987 a mano de la desarrolladora Square Co. (actualmente conocida como Square Enix) en la plataforma de NES: *Nintendo Entertainment System*.

FINAL FANTASY

FIGURA 8. Logo de la saga Final Fantasy

Entre las entregas más aclamadas encontramos a Final Fantasy X. Esta nueva entrega sale en Europa en 2002, y mantiene todo lo esencial de un clásico RPG de combate por turnos sencillo, aportando un hilo argumental completamente original y sorprendente. Además, adhiere muchas características al juego secundario, es decir, añade misiones, logros y objetivos ocultos y secundarios que no desarrollan el hilo principal de la historia pero que complementan al juego a medida que se avanza.

Este título fue tan bien recibido que ha sido uno de los varios en los que la desarrolladora se ha centrado en sacar una entrega remasterizada para pulir y mejorar sobre todo aspectos estéticos y agregar efectos visuales que, en la época y por los medios, no se pudieron desarrollar, en definitiva, una mejora gráfica.

FIGURA 9. Poster combinado de Final Fantasy X y la secuela X-2 remasterizadas.

Los personajes de la entrega están muy bien definidos, tanto en aspecto visual como en características y rasgos de personalidad que los hacen únicos. Esto favorece a que el usuario intime más con ellos y se cree un vínculo para la jugabilidad. A esto hay que añadirle que esta entrega de Final Fantasy es la primera en añadir voces de audio al os personajes. Esta característica fue muy bien recibida por la comunidad y le daba al juego el toque de esencia tanto a los personajes como a la historia.

FIGURA 10. Diseño de los personajes principales del título.

Pese a no ser el primero en introducir un árbol de habilidades completo para las diferentes clases de personajes jugables, dicho árbol ofrece una amplia personalización de las mismas, puesto que todas las clases se comunican entre sí. Con esto nos referimos a que si, por ejemplo, tu personaje es en principio una maga, en el trascurso de la historia podrás avanzar a la parte del árbol que ofrece habilidades específicas de guerrero y podrás usarlas aun siendo una maga y viceversa.

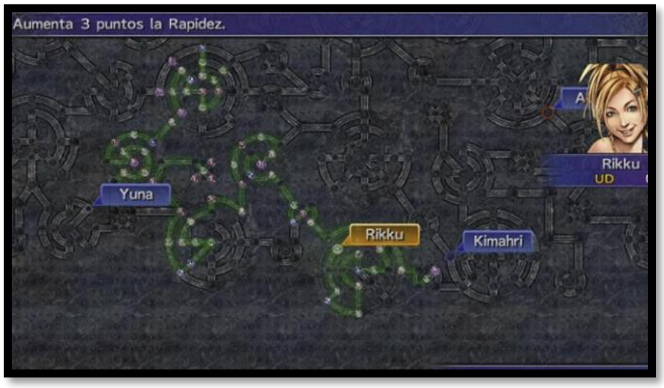


FIGURA 11. Árbol de habilidades del Final Fantasy X

En cuanto al sistema de combate, viene a ser un sencillo y tradicional combate por turnos, pero se podría decir que añade una cualidad de tiempo real en la que eres capaz de ver el estado completo de tus aliados y los enemigos, así como sus posibles intenciones. Esto ayuda a realizar estrategias más complejas o diferentes en cuanto al combate.

FIGURA 12. Sistema de combate.

2.1.4. MMORPG: World of Warcraft

Quizás los videojuegos multijugador masivo online RPG son el género en más auge en la última década. Su éxito viene a partir de conseguir lo que es en esencia un RPG, combinándolo con la magia de internet y conseguir mezclar a una enorme cantidad de personas en un mismo espacio virtual, capaces de llevar a cabo las acciones que se esperan de un juego de este estilo: combate orientado a enemigos con mecánicas por IA, recolección de elementos inusuales difíciles de conseguir, armamento poderoso que mejora enormemente el estado del personaje, etc.

Entre los títulos más famosos de este subgénero encontramos el *World of Warcraft*, desarrollado y producido por Blizzard Entretaiment, lanzado en 2004 y que todavía permanece en activo en la actualidad, sacando nuevo contenido para sus usuarios activos. En WoW no hay un objetivo en concreto, si bien es cierto se desarrollan unos hilos argumentales principales en cada nueva actualización grande de juego, el juego tiene multitud de características que lo hacen variable y con infinitas posibilidades.

FIGURA 13. Logo de World of Warcraft.

La jugabilidad de este juego consta de varias cosas. En primer lugar, nosotros como jugador deberemos escoger un bando por el que luchar, así como la raza y sus clases, que varían en función de dicho bando escogido. Una vez dentro del juego, el movimiento se basa en combinar teclas claves de teclado con movimiento de ratón, así como con combinación de clicks. Veremos que la cámara de nuestro proyecto se basa en gran medida en la jugabilidad que ofrece ésta.

FIGURA 14. Pantalla de creación de personaje: elegir bando, raza y clase.

En cuanto al sistema de combate, se desarrolla bajo otra premisa que deja de lado los turnos: el tiempo real. Los diferentes jefes finales, a los que nos enfrentaremos en los desafíos que se nos plantean en el transcurso de las misiones (normalmente cuando el jugador decide realizar mazmorras o bandas con otros jugadores), tendrán mecánicas únicas de ataque en tiempo real que supondrán un reto de estrategia para los jugadores más allá de lanzar habilidades y hacer daño.

FIGURA 15. Combate en una banda o "raid" con múltiples jugadores contra un jefe principal

En la imagen superior, podemos observar como el jefe ha dejado en el suelo elementos incandescentes que permanecen en el escenario, haciendo daño al jugador si pasa o se mantiene encima de ellos. La colocación del grupo de jugadores que intenta vencer al jefe es clave para que no se creen zonas inaccesibles o llenas de obstáculos perjudiciales que supongan la derrota del equipo.

2.1.5. Otros subgéneros

Como mencionamos en el apartado introductorio, dentro de la categoría de los RPG encontramos multitud de subgéneros que combinan aspectos fundamentales de éste, como puede ser la progresión del personaje y sus niveles, con otros géneros. Sin ir más lejos, en la actualidad uno de los subgéneros más populares son los denominados *Action RPG*.

El término de acción se añade al implantar un estilo de combate distinto a lo que ha venido concebido en otros RPG más clásicos por turnos. Ahora el estilo de tiempo real es una característica que le da vida y presencia al juego. Si echamos un vistazo a entregas punteras y exitosas como *Dark Souls*, el sistema de combate se basa en enemigos NPC que atacan con distintas técnicas y habilidades al jugador, capaces de identificar momentos claves como la defensa o tomar pociones y reaccionar ante ello.

FIGURA 16. Combate contra un jefe en Dark Souls III.

En este tipo de juegos, seleccionar el momento de realizar cada acción es crucial para que el combate salga satisfactorio, de lo contrario el mínimo error será penalizado con el fracaso y la vuelta a empezar. Los combates a tiempo real son una característica bastante aclamada por los jugadores y establece la nueva tendencia de los RPG en los títulos más modernos.

2.2. Revisión sobre la tecnología: plataformas y características

2.2.1. Final Fantasy X (Remastered) – Phyre Engine.

Las versiones remasterizadas de estos títulos, lanzadas en 2014, se desarrollaron utilizando Phyre Engine como motor de juego. Éste es un *Game Engine* multiplataforma que ofrece un gran rendimiento a la hora de trabajar tanto como en fase rápida de prototipado como en fase completa de producción. Este Engine surge de la necesidad de generar juegos de la manera más personalizada posible y "*free to use source*".

Entre las principales plataformas para las que este motor ofrece desarrollo, encontramos PlayStation 4, PlayStation 3, PlayStation VR, PlayStation Vita y Portable, Microsoft Windows (para OpenGL y DirectX 11), Android y iOS entre otras. Pese a estar desarrollado por Sony, este motor de juego permite ser utilizado en muchas otras plataformas, aunque es cierto que su rendimiento de procesamiento es mucho mayor en dispositivos que utilicen los procesadores de PlayStation.

Phyre Engine se distribuye como un paquete que incluye el código fuente y las herramientas de Windows necesarias para el uso flexible del motor. Como mencionamos, éste soporta OpenGL y Direct3D, además de librerías a más bajo nivel para sistemas PS3. Se incluyen "*plantillas jugables*" en el propio código como ejemplos e incorpora tecnologías como Havok, NVIDIA PhysX y Bullet para las físicas de los escenarios.

2.2.2. Game Engine en la actualidad.

Actualmente, entre los motores más reconocidos en la industria y que más auge están teniendo encontramos a Unreal Engine y CryEngine entre otros. La plataforma de CryEngine es de uso gratuito, adquieres todos los contenidos del motor sin necesidad de adquirir licencias de pago o características ocultas. Es un motor multiplataforma, es decir, permite desarrollar para más de un sistema operativo (PS4, Xbox, Windows PC, etc.). Un punto a favor es que como tal, CryEngine dispone de una gran cantidad de recursos formativos y complementarios con los que desarrollar, así como muchos assets gratuitos para juegos.

Otro de los motores más famosos es sin duda Unreal Engine, con el que se han desarrollado títulos como Fornite, Spyro: Reignited Trilogy o el último remake de Final Fantasy VII. Es uno de los motores más usados y actualmente extendido, distribuido por la compañía Epic Games. Permite la manipulación total de entornos 3D en tiempo real con unos gráficos hiperrealistas, así como físicas fiables e inmersivas. Ya están en el mercado con versiones del motor para plataformas futuras como PS5 y XBOX X, y con los buenos resultados que proporciona, veremos más de un título exitoso bajo este motor.

3. Definición del Juego

El juego está inspirado en su mayoría en el título World Of Warcraft, dado que por las características de tiempo y personal de desarrollo en este juego es obviamente mucho inferior, el juego podría ser considerado un fragmento de *WoW* en el que se centra en un combate contra un *boss*. [FIGURA. 15]

3.1. Género

El género del juego es RPG de acción porque los jugadores representarán un rol en un mundo ficticio de fantasía donde van a tener que combatir contra un boss en tiempo real utilizando una serie de habilidades de las que han sido dotados.

3.2. Fases de interacción del jugador

En menú: el jugador podrá elegir la apariencia de su personaje.

En el momento previo al combate: se podrá explorar la arena, con una ambientación muy oscura para darle más intriga al conjunto del escenario hasta que se empiece el combate, pero permitiendo explorar la UI y los controles al usuario.

En el momento del combate: se podrá visualizar el escenario en todo su esplendor y empezará un combate con mecánicas, está será la fase de mayor interacción del usuario.

3.3. Historia y ambientación

Dado que no se podrá ir descubriendo la historia avanzando por distintas zonas, ya que todo el juego ocurre dentro de una arena, el contexto lo da un NPC. El contexto de historia que dará será el siguiente.

*"Hola Lubie Piwo, sé que no sabrás como has despertado aquí, yo tampoco lo sé muy bien realmente... El caso es que estás en Meeseks, un pueblo al sur del continente, el que es mi pueblo... bueno ... el que era, ahora solo es un montón de lava y escombros *snif*, por favor yo estoy herida, tengo fuerza para dos últimos hechizos... sí… tengo poderes mágicos… veo que ves mucha oscuridad, puedo usar uno de mis hechizos para ayudarte, pero necesito que castigues al causante de esta destrucción... El otro hechizo lo voy a usar para irme y dejarte solo JAJAJAJAJAJAAJ. Avísame cuando estés preparado."*

Ya que el diálogo resalta que estamos en un pueblo que ahora solo es un montón de lava y escombros, ese es el estilo de la ambientación. Un cielo rojo, con una niebla anaranjada, lava como principal elemento del terreno y piedras volcánicas.

3.4. Definición de personajes

- **Kuarx**: es el NPC que aporta el contexto del por qué el jugador se encuentra ahí y que además inicia el combate. Es una mujer de la nobleza con poderes mágicos.

- **Derrolicto**: es el *boss* a derrotar para conseguir la victoria. Es un gólem azteca que se ha hundido en el suelo por su peso y suelta frases en inglés.
- **Lubie Piwo**: es el nombre del campeón que manejará el jugador. Tiene una personalidad desconocida y un aspecto cambiante.

3.5. Objetivo

Derrotar a Derrolicto en el menor tiempo posible y buscar la mejor rotación de habilidades en cada situación.

3.6. Concept Art

El escenario, el boss y las mecánicas están inspiradas en WoW: Cataclysm.

FIGURA 17. Mecánicas de combate en WoW

FIGURA 18. Escenarios de World of Warcraft

FIGURA 19. Diseño de jefes de Cataclysm.

4. Diseño técnico

4.1. Entorno de desarrollo

- Game Engine: Unity
- Sistema Operativo: Windows 10

4.2. Requerimientos técnicos del entorno

Minimum requirements	Windows	macOS	Linux (Support in Preview)
Operating system version	Windows 7 (SP1+) and Windows 10, 64-bit versions only.	Sierra 10.12.6+	Ubuntu 16.04, Ubuntu 18.04, and CentOS 7
CPU	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support
Graphics API	DX10, DX11, and DX12-capable GPUs	Metal-capable Intel and AMD GPUs	OpenGL 3.2+ or Vulkan-capable, Nvidia and AMD GPUs.
Additional requirements	Hardware vendor officially supported drivers	Apple officially supported drivers	Gnome desktop environment running on top of X11 windowing system, Nvidia official proprietary graphics driver or AMD Mesa graphics driver. Other configuration and user environment as provided stock with the supported distribution (Kernel, Compositor, etc.)
	For all operating systems, the Unity Editor is supported on workstations or laptop form factors, running without emulation, container or compatibility layer.		

4.3. Herramientas empleadas en el desarrollo

- **Unity:** el *game engine* (motor del juego) en el que se ha desarrollado.
- **Mixamo**: herramienta web (mixamo.com) gratuita de Adobe. Utilizada para dotar de animaciones a los modelos con esqueletos humanoides a través de un amplio catálogo de éstas ya creadas. De aquí proceden todas las animaciones del juego.

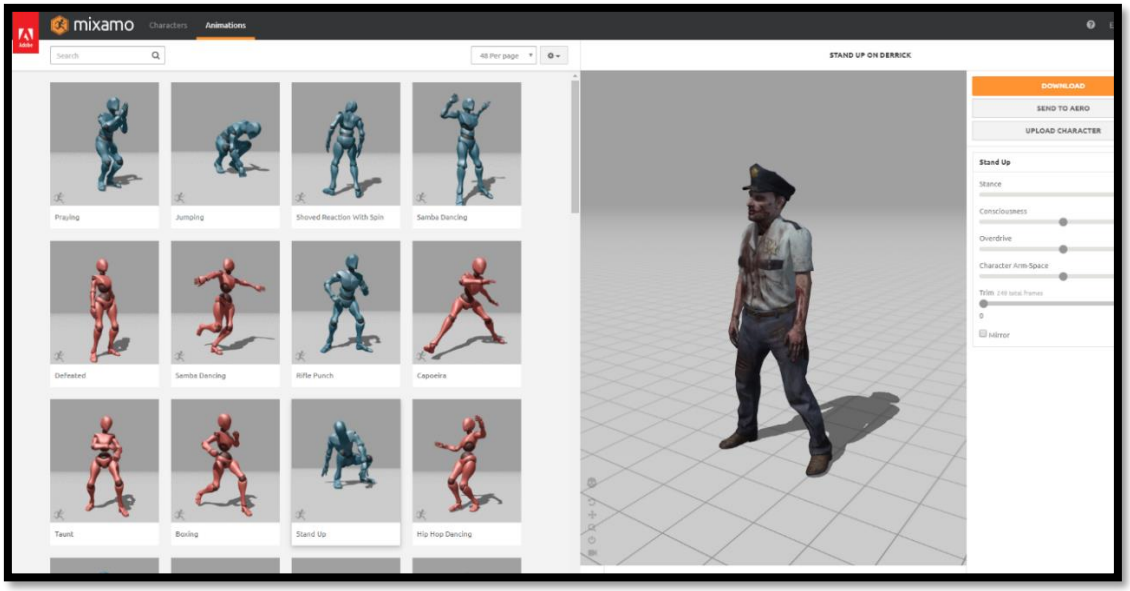


FIGURA 20. Herramienta Web Mixamo

- **Audacity**: herramienta *open source* de edición de audio. Utilizada para editar (recortar/modular/modificar) el audio de los efectos sonoros del juego.

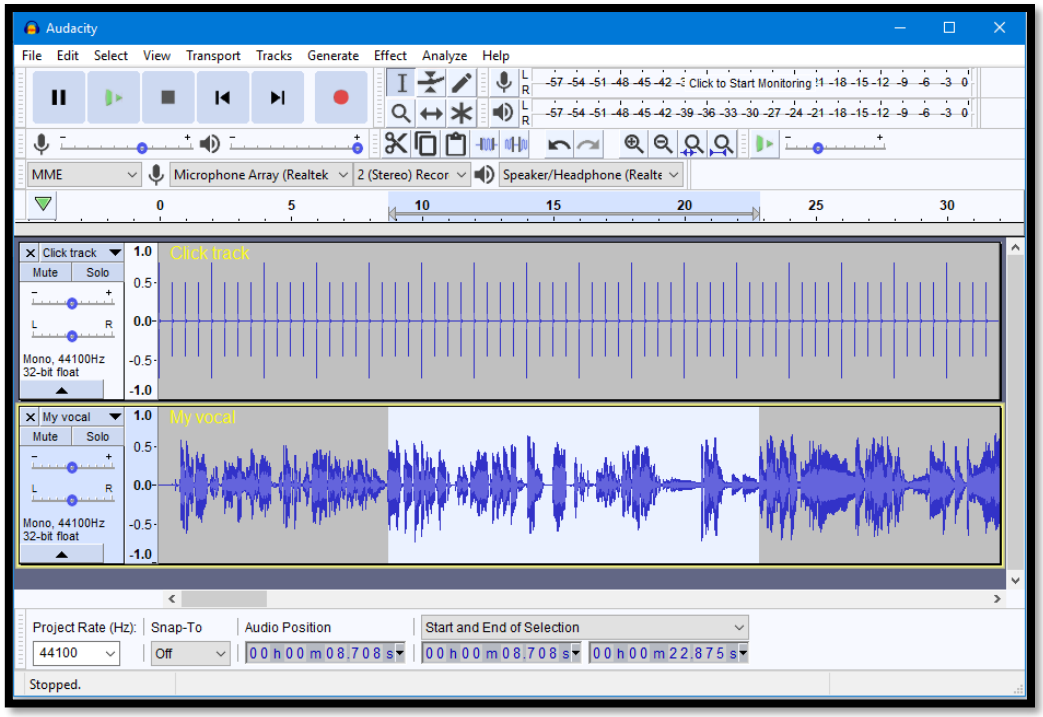


FIGURA 21. Entorno de trabajo de Audacity.

- **Paint.net**: editor de imágenes para Windows.

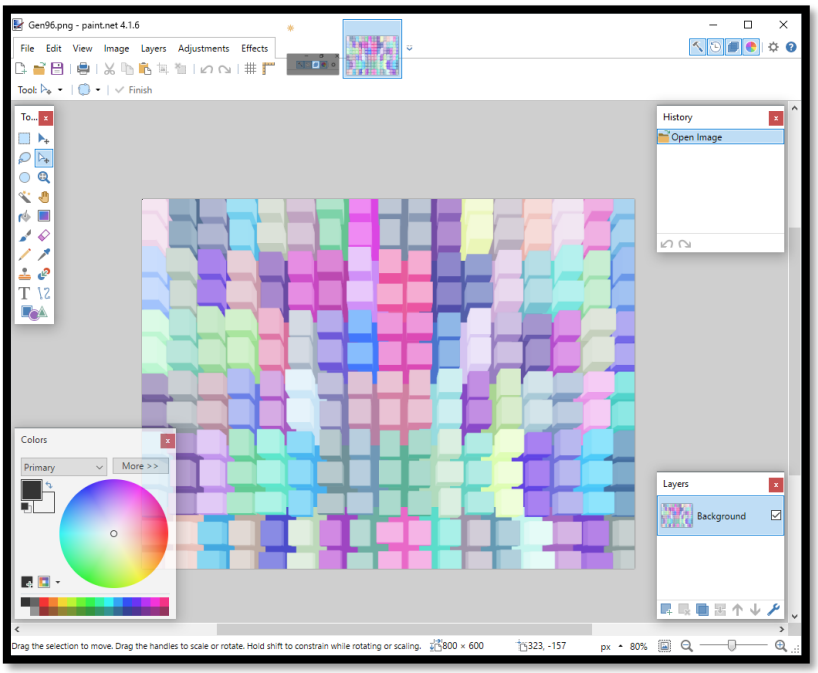


FIGURA 22. Entorno de trabajo de Paint.net

- **Shader Graph**: plugin de Unity obtenido a partir de su Package Manager, necesarios los paquetes '*Lightweight Render Pipeline*' y '*HD RenderPipeline*'. Utilizado para la creación de shaders de forma ágil y sin necesidad de introducirse en la notación de estos en Unity.

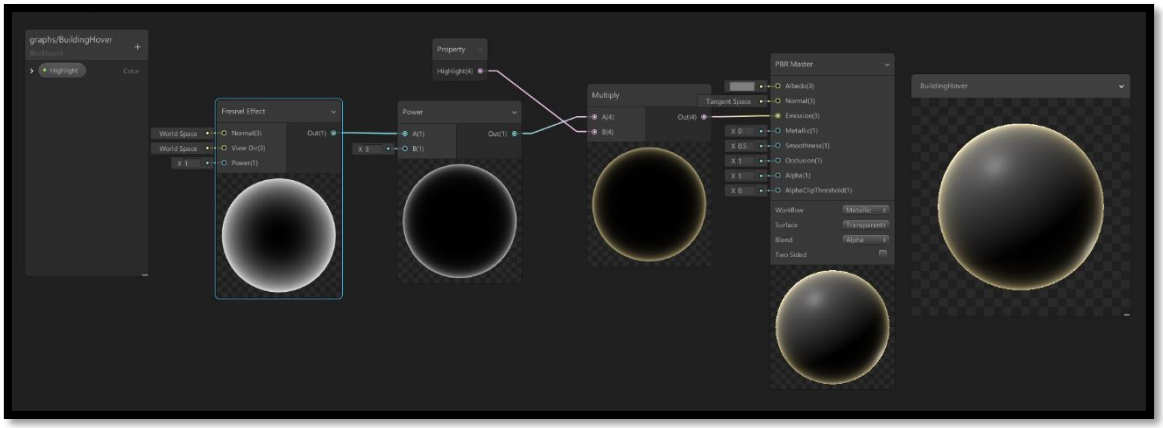


FIGURA 23. Ventana de edición de Shader Graph: creación de un Shader Fresnel

- **HDRP:** HD Render Pipeline obtenido a través del Package Manager de Unity para conseguir efectos visuales más reconfortantes y para poder usar otras herramientas como la anterior.

4.4. Assets y recursos del juego

4.4.1. Recursos de origen propio

[1] SHADERS:

1) Lava Shader:

Creado para conseguir el material de lava que aparece por todo el escenario.

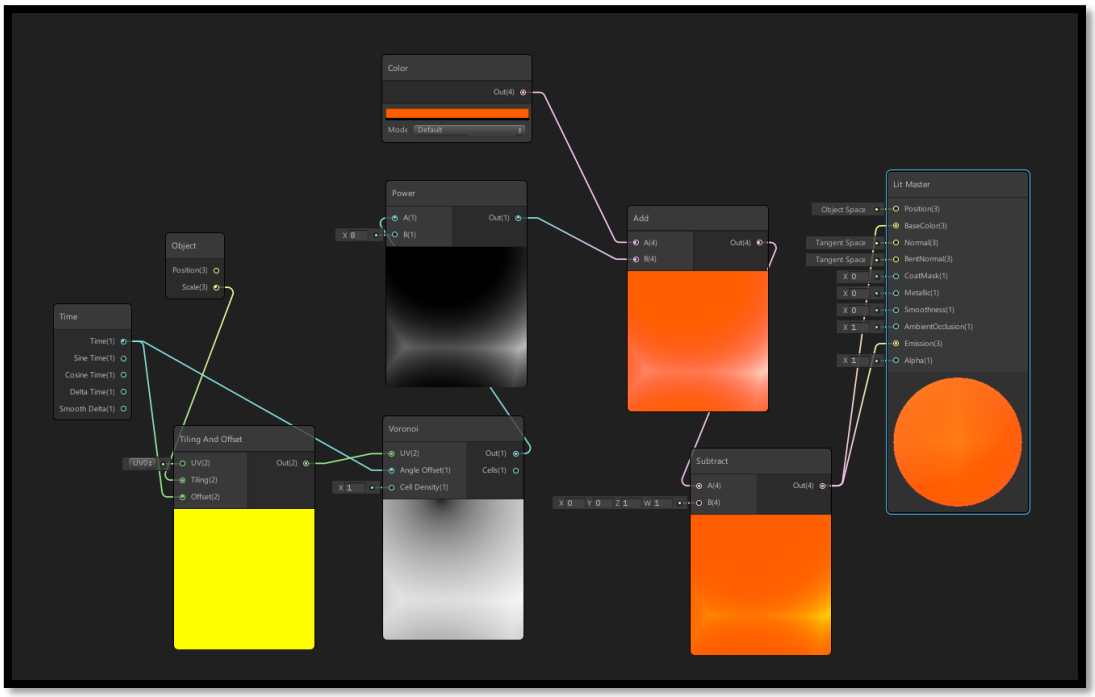


FIGURA 24. Esquema de creación del Lava Shader

Se basa en generar puntos *voronoi*, los cuales serán desplazados en el tiempo, aplicando un Tiling igual a la escala del objeto para que siempre mantenga su aspecto y no se deforme. Los puntos *voronoi* son potenciados para que el negro #000000 ocupe el mayor espacio y posteriormente poder sumarle el color predominante para que desaparezca el negro y el blanco quede blanco #FFFFFF. Por último, al blanco se le resta el canal BLUE para que la parte minoritaria sea la combinación entre el canal RED y el GREEN #FFFF00 (el amarillo). El resultado se aplica a al color base del material y a la emisión para que sea luminoso como la lava.

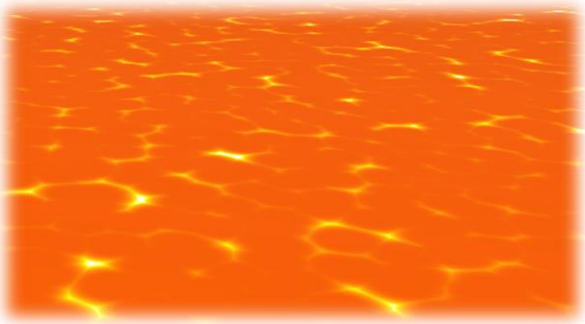


FIGURA 25. Resultado del Lava Shader

2) Filler Shader:

Creado para conseguir un material que represente un fill de progreso para usar con las barras de vida y que se renderice siempre delante de todos los demás.

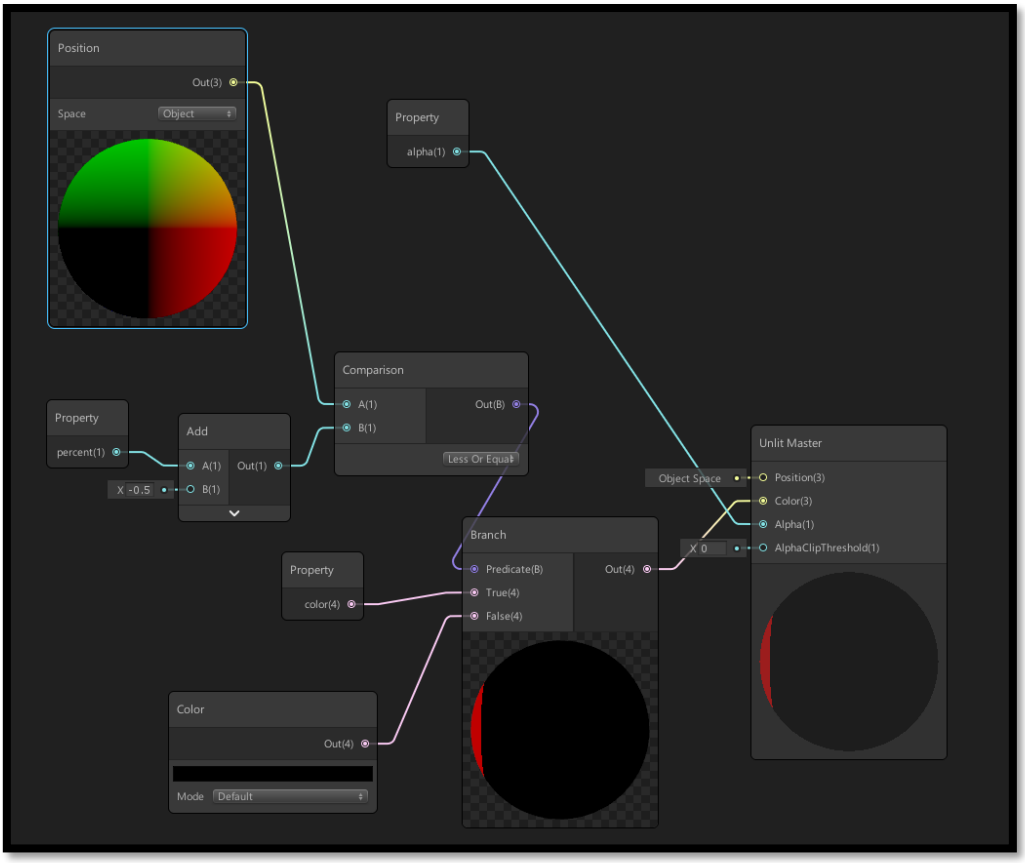


FIGURA 26. Esquema de creación del Filler Shader

Se basa en definir una constante expuesta entre 0.0 y 1.0 (el espacio de los vértices del objeto normalizados), y comparar si esa variable es mayor o menor que cada fragmento del objeto; los que caigan por debajo pertenecerán a la "zona fill" y el resto al background de la geometría que los contiene. En este caso el background es negro y el color del fill es definido por una variable de color expuesta (un uniform). O uniform expuesto es el Alpha del material. Todo esto se encuentra en el fichero X-Filler-SG, pero con solo lo explicado el Shader no permite renderizar delante de todo lo demás, ya que Shader Graph no tiene control sobre opciones como el ZWrite, el DepthTest o el RenderType. Por lo tanto, la solución ha sido coger el código de Shader generado por Shader Graph y crear un nuevo Shader donde el ZWrite esté a Off y los tags sean los siguientes:

```
Tags {
    "Queue" = "Transparent"
    "IgnoreProjector" = "True"
    "RenderType" = "Transparent"
    "PreviewType" = "Plane"
}
```

CÓDIGO 1. Tags del Filler Shader

Y así con este Shader se consigue el material para las barras de vida en escena 3D (MeshBars).

FIGURA 27. Meshbar sin clipear con otros meshes

3) <u>Laser Shader</u>:

Creado para conseguir un efecto de láser para una habilidad del *boss*.

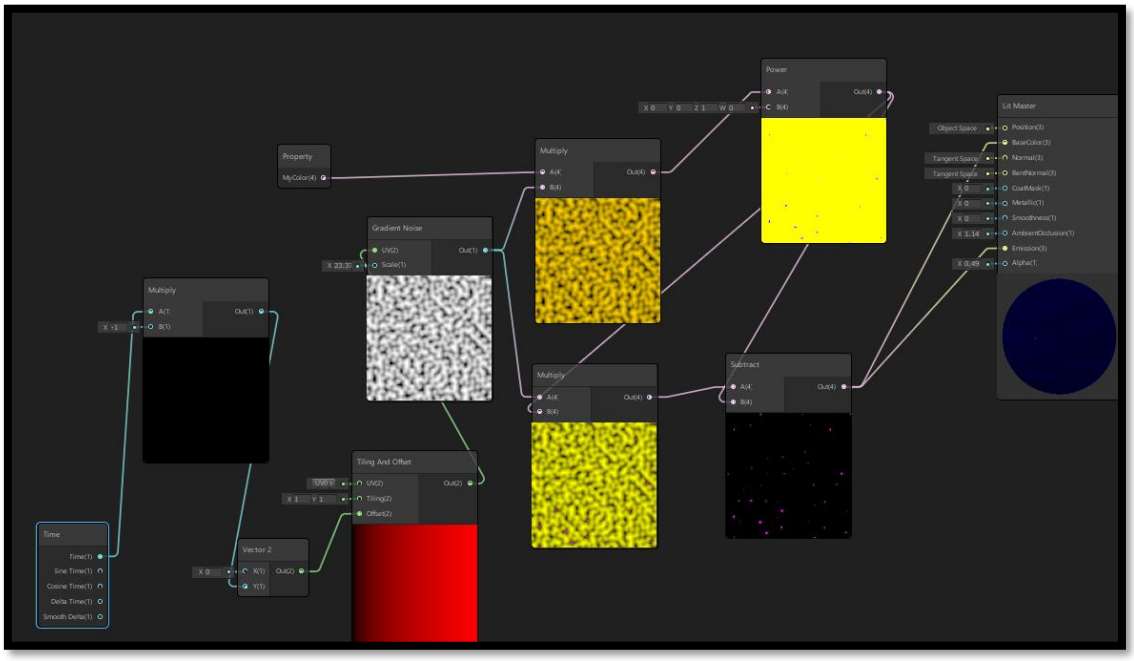


FIGURA 28. Esquema de creación de Laser Shader

Su basa en definir ruido y que se mueva en una dirección, (el ruido y dirección que tendría un láser al ser lanzado), se aplican multiplicaciones y restas para generar caos en los colores y al final se obtiene una textura negra con punteados (el ruido del láser). La textura negra será la final, pero se verá afectada por la iluminación y color del fondo y pasará en el caso de juego a ser rojiza.

FIGURA 29. Resultado Laser

4) Fresnel Shader:

Se ha creado para iluminar los hacer el VFX de la habilidad *Divine Shield*, la cuál es una esfera a modo de escudo con partículas luminosas.

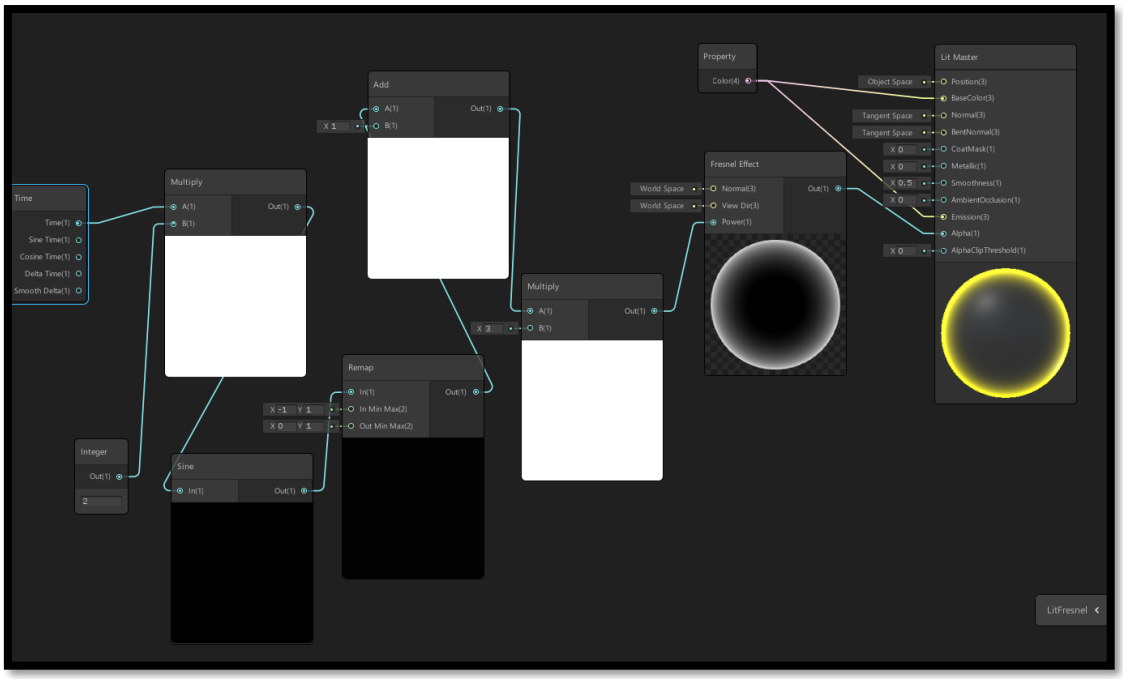


FIGURA 30. Esquema de creación del Fresnel Shader.

Dado que el efecto Fresnel es muy común, Shader Graph contiene un nodo que lo genera, pero se ha añadido que la fuerza del Fresnel varíe en el tiempo para conseguir un pequeño parpadeo (para que se vea un poco más dinámico). Para esto se hace uso de la función seno (0 – 1) y del control de la velocidad del tiempo (velocidad de parpadeo), el resultado de esta operación define el Alpha del material, el cual deberá ser transparente. Además, se puede controlar el color del Fresnel con el *Uniform* de color.

FIGURA 31. Resultado efecto Divine Shield

[2] Materiales

Algunos materiales son de otros orígenes, pero han sido modificados, ya sea su transparencia su emisión o su color base. El resto son propios empleando los shaders anteriores.

[3] Partículas

Todas las partículas proceden de un paquete externo, pero algunas han sido modificadas para obtener el resultado deseado.

- Partículas del *Divine Shield* (FIGURA 31. Resultado efecto Divine Shield)
- Orbe curativo (Figura 20)

[4] Escena y escenario

Todas las escenas y escenarios creados son propios.

FIGURA 32. Escena de selección de aspecto

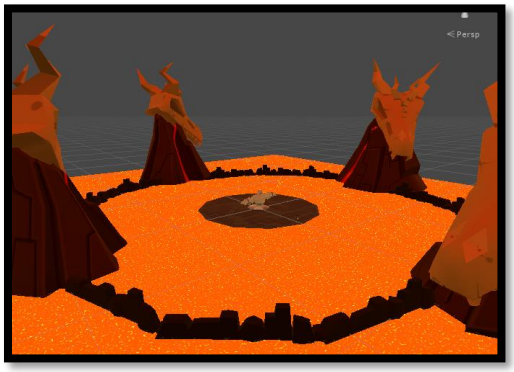


FIGURA 33. Escenario principal del juego

[5] UI

Todos los elementos de UI han sido diseñados a partir de los objetos básicos de Unity.

[6] Scripts

Todos los scripts son de origen propio, excepto uno que es de otro origen y ha sido modificado.

➔ Scriptables Objects:

1) *Dialog*:

Se ha definido con el fin de crear cuadros de diálogo con los NPC de manera ágil. Los campos que lo componen son los siguientes:

```
public string trashTalk;
public DialogChoice[] choices;
```

CÓDIGO 2. Atributos de la clase Dialog

El primer atributo corresponde a todo el diálogo que va a reflejar el NPC en el panel al hablar con él. El segundo atributo corresponde al conjunto de réplicas con las que podemos interactuar.

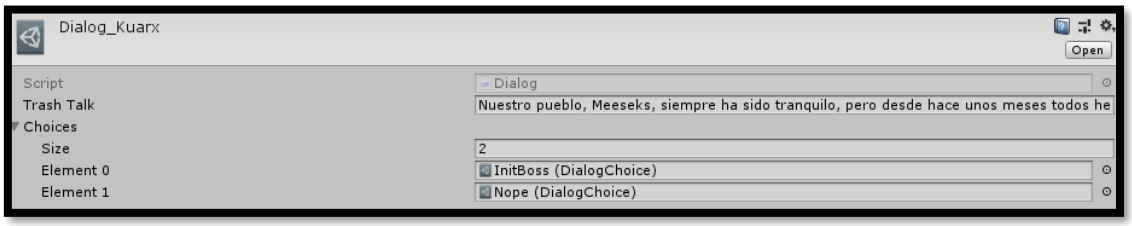


FIGURA 34. Ejemplo Dialog Scriptable (Kuarx NPC)

2) *DialogChoices*:

Esas réplicas también son Scriptables ya que pueden variar de muchas formas. Sus atributos son los siguientes:

```
public string text;
public GlobalAction action;
```

CÓDIGO 3. Atributos de la clase DialogChoices

El primero se refiere a un texto de respuesta y el segundo a la acción que va a generar en el NPC, estas acciones son definidas como un *Enum* para posteriormente identificarlas en código. En este caso *InitBoss* y *NOPE* son las acciones definidas, la primera inicia el combate y la segundo no hace nada.

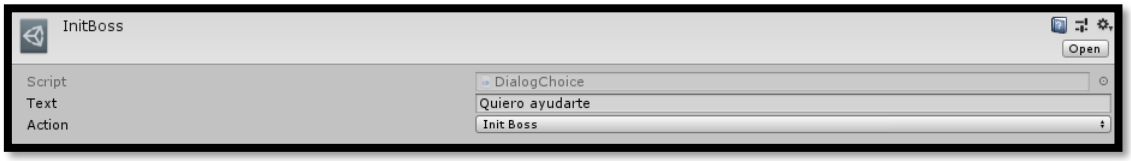


FIGURA 35. Ejemplo InitBoss DialogChoice

Ligando el *Dialog* generado al *NPCController* correspondiente, se podrá obtener el siguiente resultado:

FIGURA 36. Interacción con NPC

3) Entity:

Todo aquel elemento del juego que sea seleccionable o esté dotado de nombre y vida representa una entidad. Se ha creado para definir de forma ágil nuevos personajes u objetivos que puedan ser seleccionados. Sus atributos son los siguientes:

```
public Material baseMaterial;
public Material mouseOverMaterial;
public Texture2D cursorTexture;

public string entityName;
public int initHealth;
public int maxHealth;

public Relation entityRelation;
```

CÓDIGO 4. Atributos de la clase Entity

- **baseMaterial***:* material del mesh de la entidad mientras no es seleccionado.
- **mouseOverMaterial***:* material del mesh cuando el cursor pasa por encima de esa entidad.
- **cursorTexture***:* aspecto que obtendrá el cursor cuando este pasa por encima de la entidad.
- **entityName**: nombre que se mostrará en la placa de interfaz cuando se selecciona esa entidad.
- **initHealth**: vida inicial de la entidad
- **maxHealth**: vida máxima de la entidad
- **entityRelation**: define cual es la relación del jugador con esa entidad, puede ser: Enemy, Ally o NPC, lo que según su definición alteraría el color de la vida en la placa de interfaz siendo rojo, verde o amarillo respectivamente.

FIGURA 37. Entidad Derrolicto (Boss)

FIGURA 38. Selección de un elemento de clase Entity

4) *OverlapActionMap*:

Abreviado como OAMap, define que acción realizar sobre objetos intersectados, procedentes del script Overlaper, descrito en epígrafes posteriores, cuyo tag está incluido en cada entrada del OAMap.

```
[System.Serializable]
public class OAEntry
{
    public string[] inclusionTags;
    public OverlapAction action;
}

public OAEntry[] entries;
```

CÓDIGO 5. Definición de los OAMap

El OAMap es un conjunto de entradas, donde cada una define sobre que tags afecta y que acción produce. Las acciones están definidas como *Enum* y actualmente existen dos: *Hurt* (que causa daño en la vida del objeto interceptado) y *Clear* (que destruye el objeto interceptado).

- Ejemplo A:

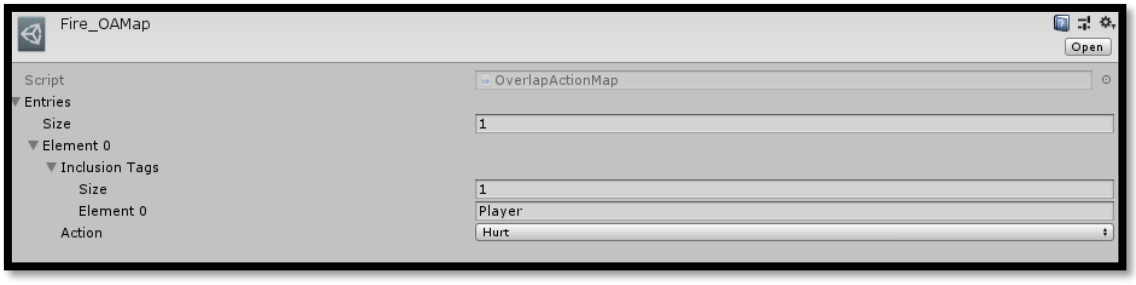


FIGURA 39. Configuración OAMap del fuego

Este OAMap define como afecta el overlap del fuego en el juego. Realiza sobre los objetos Player la acción de *Hurt*, por lo tanto, el fuego solo hará daño a los jugadores.

- Ejemplo B:

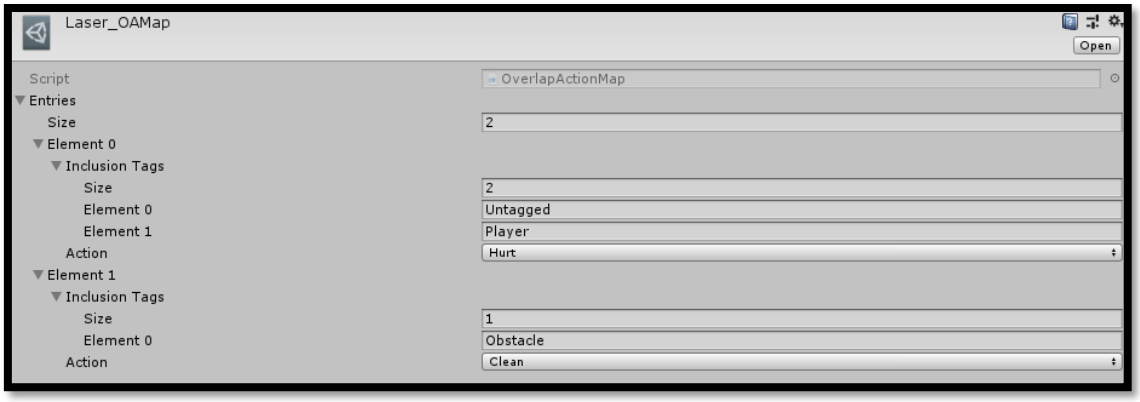


FIGURA 40. Configuración OAMap del laser

Este OAMap define como afecta el overlap del rayo láser en el juego. Realiza daño a Player y objetos Untagged, y además destruye todos los obstáculos que toque (por ejemplo, el fuego es un obstáculo, por lo que el láser apaga el fuego al entrar en contacto con este).

5) *Skill*:

Creado para poder definir habilidades y poder balancearlas de forma ágil.

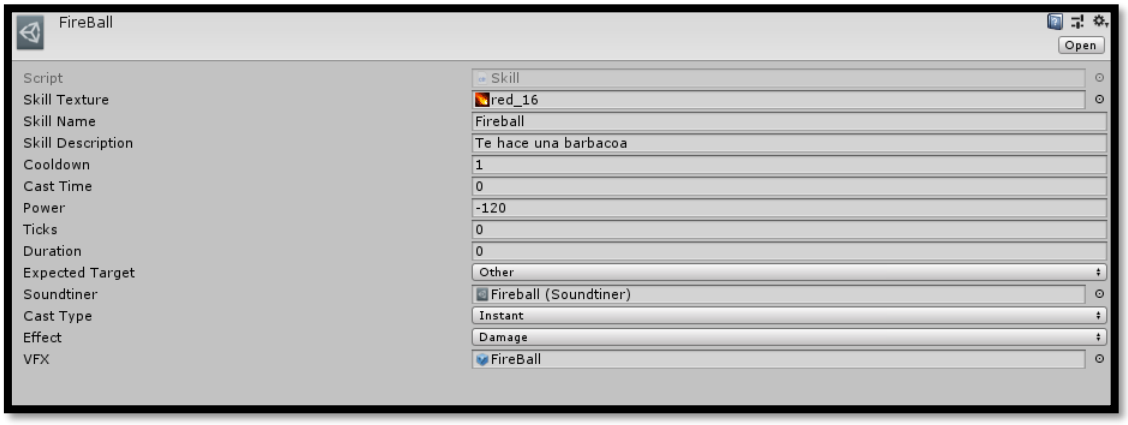


FIGURA 41. Definición de Fireball

- **skillTexture**: representación gráfica 2D de la habilidad
- **skillName**: nombre de la habilidad
- **skillDescription**: descripción de qué hace la habilidad
- **cooldown**: segundos que han de pasar para poder reutilizar la habilidad
- **castTime**: segundos que tarda en ser lanzada
- **power**: poder de la habilidad
- **ticks**: en caso de ser una canalización, cuantos ticks de daño va a producir.
- **duration**: lo que dura la habilidad si representa un buff o debuff
- **expectedTarget**: define que entidad es el receptor del efecto de la habilidad, existen 3 tipos: Self (solo la puede recibir el mismo que la lanza), Other (solo la puede recibir otro que no sea el que la lanza), Both (todos pueden recibirla)
- **soundtiner**: sonidos y configuración de estos cuando la habilidad es lanzada

- **castType**: tipo de lanzamiento, existen tres: Instant (instantáneo), Casted (lanzado después de un tiempo), Canalized (mientras se lanza ocurre el efecto).
- **VFX**: efecto que produce la habilidad en el objetivo

6) *Soundtiner*:

Creado para poder manejar fácilmente los AudioSource cuando existen varios en una misma entidad. Consiste en definir la configuración que va a adoptar el AudioSource, el conjunto de clips de sonidos, el orden de reproducción y el canal por el que se reproducen (Los canales se definen para un objeto incluyendo el script SoundController).

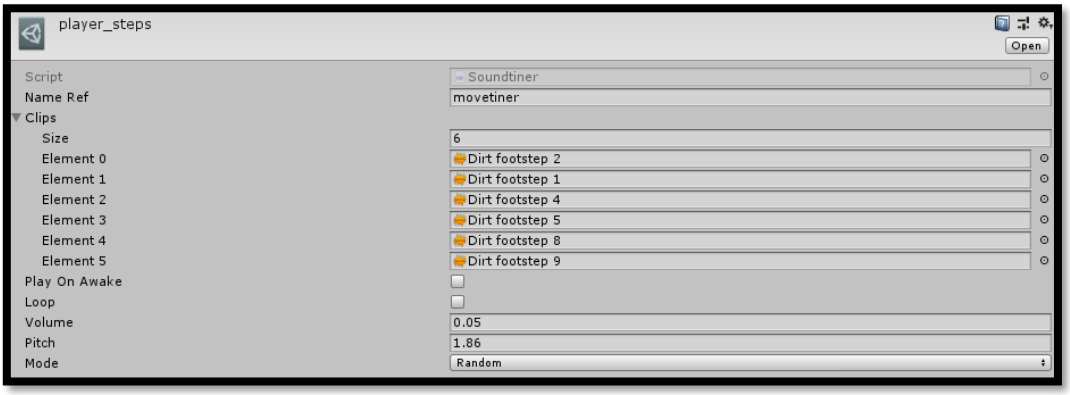


FIGURA 42. Soundtiner de pasos

- **NameRef**: nombre del canal para identificar el AudioSource
- **Clips**: colección de clips de audio.
- **Mode**: modo en el que moverse por la colección, existen tres: First (solo reproduce el primero de la colección), Random (reproduce de forma aleatoria uno de los clips), Seq (reproduce en secuencia los clips, solo una reproducción por llamada a Play).

Por ejemplo, los pasos son una colección de distintos tipos de pasos y en cada evento de animación al dar uno reproducirá un clip aleatorio de la colección del soundtiner.

Interfaces

1) *State*: Interfaz que define las funciones que ha de implementar cada estado de la máquina de estados del boss.

```
void UpdateState();
void OnEnter();
void OnExit();
void ToState(IState s);
void Event(string eventName);
```

- **UpdateState**: el update que adoptará el boss al transitar a ese estado

- **OnEnter**: lo primero que ocurre al entrar a un estado (cambios de animación, reproducción de algo, etc)
- **OnExit**: lo que se ejecuta antes de salir del estado (reset de parámetros)
- **ToState**: transita al estado indicado
- **Event**: definir cómo actuar en cada estado a eventos que reciba el boss

---

Static Classes

1) *GameConstants*:
Constantes del juego a modo de configuración fácil de acceder.
- *Colores dependiendo de la relación con el jugador*.
- Gravedad base
- Índice de skin de jugador seleccionada

MonoBehaviour

1) *InfoAgent*:

Agente que se encarga de visualizar por pantalla la mayoría de causas por las que el jugador no podría realizar determinada acción.

FIGURA 43 - InfoAgent informando de incumplir un requisito

2) *PrefabsAgent*: agente que contiene los prefabs que van a ser usados por distintas entidades, con el fin de tenerlos centralizados en un punto, sean más fácil y seguros de modificar y acceder.

3) *RaycastAgent*: agente que detecta los elementos seleccionados que heredan de RayComponent. Además, tiene capacidades para bloquear la utilización de la cámara cuando detecta que se interactúa con la UI.

4) *SceneAgent*: agente que se encarga de llamar a los cambios de escenas.

5) *SpawnerAgent*: agente que obtiene los transform de sus hijos y los almacena como spawners donde generar cualquier GameObject. Usados para spawnear a los Anubiseth por parte del boss.

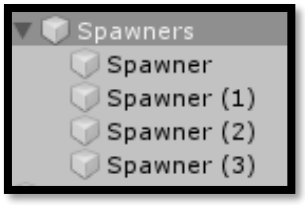


FIGURA 44. Jerarquía de Spawner Agent

6) *TargetAgent*: agente que se encarga de mantener la selección actual del jugador y mostrar su placa correspondiente en la UI.

7) *EnvAgent*: agente que guarda etiquetadas con un nombre los distintos scene settings que actúan sobre la iluminación, la niebla y el color del cielo. Permite cambiar entre ellos solo pasando la etiqueta.

---

8) *BossController*: controlador del boss que instancia y contiene los estados por los que transitar, siendo también un contenedor de información común a todos estos, por ejemplo, una referencia al jugador, a las frases del boss, a objetos necesarios para sus habilidades, a su Animator y a algunos métodos como mirar jugador o generar objeto. Se podría definir como el cerebro del boss.

9) *HealerController*: controlador que otorga a un GameObject la capacidad de curar, al asignarle un target a este script lo curará una determinada cantidad en una determinada frecuencia, ambas configurables desde Inspector.

10) *LaserController*: controlador del prefab *Laser* que permite que se pueda adherir a un punto uno de sus extremos y a pesar de ser escalado siga conservando el mismo punto de origen.

11) *MeteorController*: controlador del prefab *Meteor* que simula el comportamiento de un meteorito una vez se asigna un punto de origen y destino. Sigue la trayectoria definida por los dos puntos (con aceleración) y cuando impacta contra un sólido, se destruye, y spawnea un fuego, si no impacta contra nada desaparece cuando cae por debajo de una Y definida. Antes de inicia su trayectoria genera una marca semiesférica de donde va a impactar.

12) *StateDefinitions*: clases que representan los distintos estados en los que puede encontrarse el boss. Los estados son los siguientes de manera resumida:

- **LaserClock**: posteriormente a un breve casteo y una frase, el boss activa su laser y rota en sentido de las agujas del reloj. **(**FIGURA 29**)**

- **FireRain**: el boss pone sus manos en el cielo y provoca que caigan meteoritos en puntos aleatorios dentro de un círculo donde el radio se puede configurar y su punto central es el jugador.

FIGURA 45. FireRain State

- **Support**: invoca tantos Anubiseth (healers) como spawners indicados tenga el BossController, el boss cambiará de fase cuando estos hayan muerto o se supere un tiempo máximo definido.

FIGURA 46. Support State

- **Quake**: ráfaga de palmadas que provocan daño al jugador dependiendo de la distancia a la que se encuentre del boss.

FIGURA 47. Quake State

- **Idle**: simplemente mira al jugador.

FIGURA 48. Idle State

---

13) *Overlaper*: script encargado de detectar todos los objetos con determinados layers dentro de un área. Las áreas de solapamiento serán geometrías básicas, la geometría de una caja y la de una cápsula. Está diseñado de tal forma que tienen en cuenta el lossyScale. El tipo de geometría básica se puede seleccionar con un enum desde el inspector. Utilizado por el láser (cápsula), la lava (caja) y el fuego (caja). Cada vez que obtiene los resultados del solapamiento genera un nuevo timestamp para identificarlo. Los resultados máximos son ocho.

14) *Actioner*: requiere de un componente Overlaper para poder funcionar, su funcionamiento se basa en leer los resultados del Overlaper cuando detecta un nuevo timestamp y aplicar a esos resultados las acciones del OAMap asociado.

15) *AutoDestroy*: destruye el gameObject pasado el tiempo indicado.

16) *Billboard*: orienta hacia la cámara el gameObject constantemente

17) *BossCountdown*: inicia una cuenta atrás y cuando esta termina activa el cerebro del boss linkeado a través del Inspector.

18) *EffectSyncronizer*: utilizado para sincronizar la destrucción de efectos que contienen partículas y sonido. Espera hasta que los dos hayan terminado y luego destruye el objeto o lo desactiva.

19) *MeshBarController*: se encarga de actualizar las barras de vida 3D. Debido a que las MeshBar se rellenan modificando uniforms del shader del material *Fill*, ha sido necesario utilizar MaterialPropertyBlock [PONER URL EXTERNO] para generar distintas instancias del material, ya que las barras tendrán distintos porcentajes de vida.

20) *SkinController*: en todos los personajes, las skins están contenidas como hijos, por lo que este script detectará todos los hijos que tienen como tag “Skin”, para guardar el número de skins (el index máximo) y permitirá modificar skin indicando el index. Nota: las skins deben ser contiguas y empezar como hijo 0.

21) *SoundController*: a través del inspector define una lista de canales de audio pertenecientes al gameObject al que está ligado, cada canal generará un AudioSource y guardará su referencia ligada con su nombre y cada vez que recibe una petición de reproducción reconfigurará el canal como indique el Sountinder.

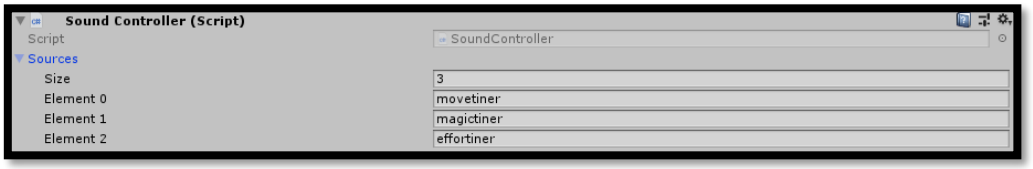


FIGURA 49. Definición de canales de audio en gameObject

---

22) *CameraController*: controla totalmente el movimiento y posición de la cámara. Se basa en enfocar la cámara constantemente a un punto definido y en dejar ángulos de libertad en los que rotar la cámara en torno a ese punto (la libertad de Y se limita para que no dé lugar a comportamientos extraños). Además, puede con la rueda acercar la cámara (reduciendo la magnitud del vector dirección),

reorientar el vector forward del jugador al de la cámara (pulsando los dos botones del ratón), realinear la cámara cuando está descolocada en su punto (0,0,Z) siendo Z la distancia del zoom, y no permitir que la cámara atraviese objetos para poder ver siempre al jugador. La táctica de esta última característica se basa en lanzar un rayo entre el punto al que mira la cámara y la posición de esta, si el rayo intersecta con algo entre medias, la cámara reduce su zoom para adelantar esa posición de colisión del raycast.

23) *MovementController*: controla el movimiento del character controller, el salto y envía interrupciones a la barra de lanzamiento de hechizos cuando detecta algún input de movimiento por parte del usuario.

24) *PlayerCore*: es el nexo de todos los componentes que conforman al jugador (que son bastantes) y determina en todo momento cual es el punto central del jugador (para que la cámara pueda seguirlo). Los dos scripts anteriores son dependientes de este y se inicializarán cuando esté los invoque.

---

25) *EPController*: controla la visualización de la entidad como placa en la UI. La placa contiene la barra de vida y el nombre.

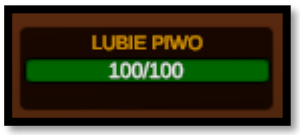


FIGURA 50. Interfaz de la vida del jugador

26) *NPCController*: controla los NPC y la interacción de diálogo con ellos usando los Scriptables de Dialog y el DialogController.

27) *PopupController*: controla los números que aparecen cuando se daña/cura a un target. Tienen un atributo de ruido que permite que no todos aparezcan en el mismo punto.

28) *RayComponent*: clase abstracta de la que hereda todo aquel objeto que va a ser raycasteado en el espacio 3D.

29) *SkillController*: escucha el input del usuario para lanzar la habilidad definida y controla el cooldown para que no se pueda volver a lanzar.

30) *TargetController*: toda entidad tendrá este script. Permite que un gameObject pueda ser seleccionado ya que modifica el layer del gameObject al que está adherido. Además, se encarga de gestionar los efectos producidos por las habilidades y de informar a los EPController cuando deben ser actualizados. Los buffs los controla a través de callbacks.

31) *CastStage*: controla que solo se pueda lanzar una habilidad al mismo tiempo, captura las interrupciones a causas del movimiento, gestiona la barra de lanzamiento de la UI e informa a cada SkillController cuando ha de entrar en cooldown.

32) *ChoiceContainer*: contenedor para acceder más rápido a los componentes del prefab choice.

33) *DialogController*: se encarga de actualizar el diálogo del NPC y las respuestas. Borra las antiguas respuestas e instancia nuevas y abre/cierra el panel de diálogo.

34) *OptiosController*: escucha continuamente la tecla Escape para hacer un toggle del menú de pausa y monitoriza al GameObject boss y al player, dependiendo de cuál sea NULL primero determinará si ha ocurrido una victoria o una derrota del jugador y mostrará el mensaje correspondiente (bloqueando también los botones que no procedan tener activados).

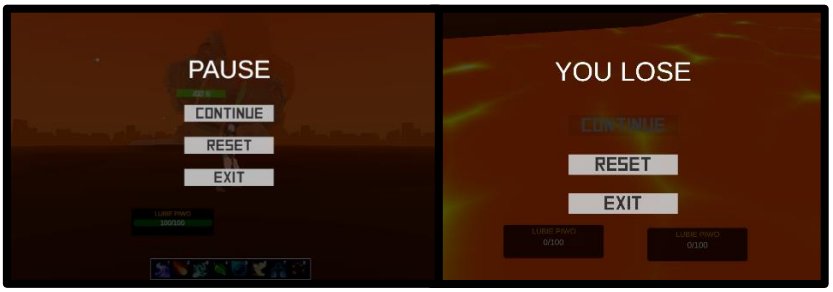


FIGURA 51. Menú de opciones (en mitad de juego izq. y cuando el jugador pierde dcha.)

35) *SkillBar*: instancia las habilidades del jugador a lo largo de la barra, le asigna una tecla a cada una, chequea que los requisitos para poder lanzar una habilidad se cumplen, posteriormente lo pasa a CastStage y este responde si puede ser lanzada o no (true/false), en caso afirmativo, el proceso de lanzamiento se inicia y todas las demás habilidades entran en global cooldown.

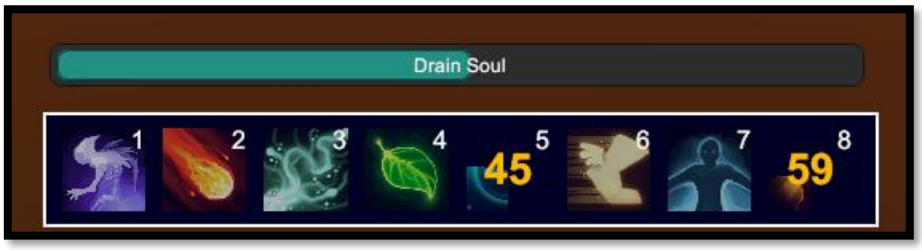


FIGURA 52. Barra de habilidades y estado

36) *SkillDescriptor*: extrae la información de los Scriptables de skills capturadas por el raycaster y la muestra de tal manera que sea fácil de consumir para el usuario.

FIGURA 53. Descripción de las habilidades

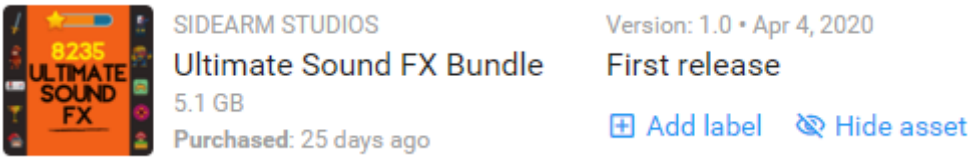
37) *GuidePanel*: muestra un conjunto de paneles paginados en los que explica las habilidades del boss definidas en los Scriptables *Guide Entry*.

FIGURA 54. Guía de las mecánicas del jefe

4.4.2. Recursos aprovechados de otros orígenes

Asset store de Unity:

- **Ultimate Sound FX Bundle**: todos los sonidos del juego proceden de este bundle.



https://soundcloud.com/sidearmstudios/sets/ultimate-sfx

- **Volcano Pack**: la decoración lejana del escenario procede de este Asset.

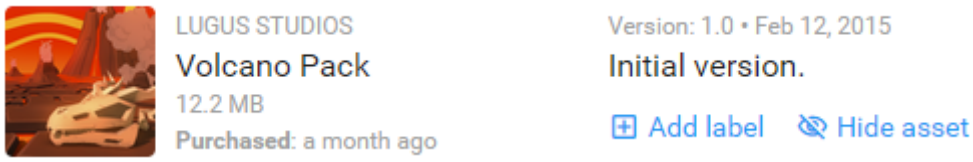


FIGURA 55. Decoración lejana del escenario.

- **POLYGON - Fantasy Rivals**: algunos modelos proceden de este Asset.

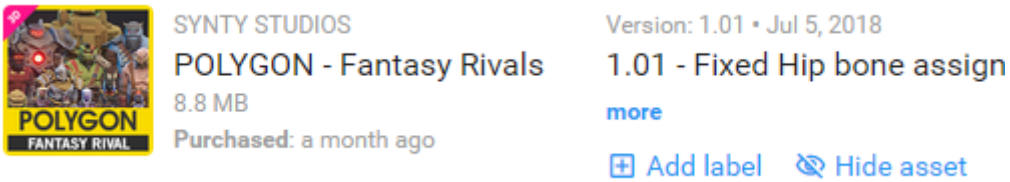


FIGURA 56. Anubiseth

FIGURA 57. Jefe: Derrolicto.

- **POLYGON – Fantasy Characters**: algunos modelos proceden de este Asset.

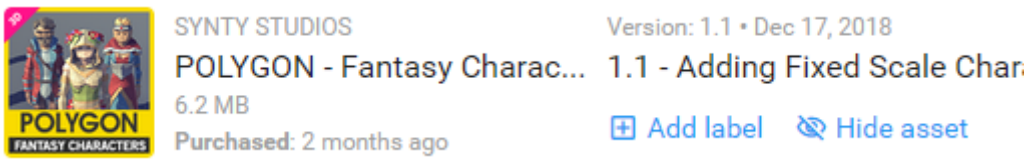


FIGURA 58. Mesh de los aspectos de los personajes

- **Skill Icon Pack**: los sprites 2D que representan las skills proceden de este Asset.

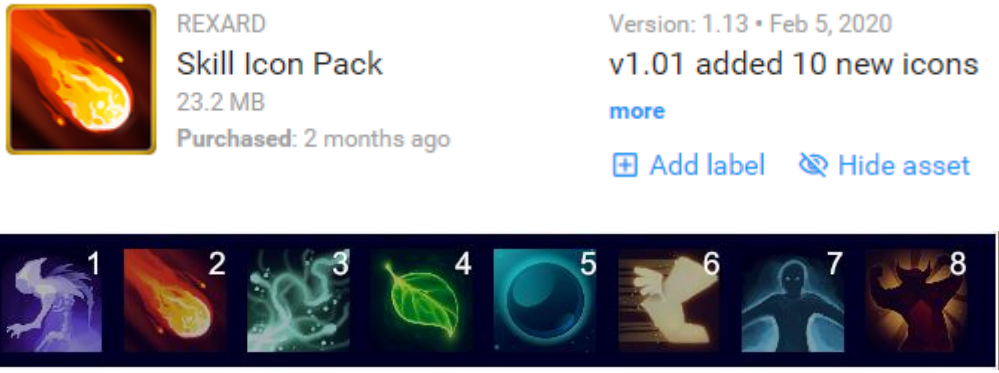
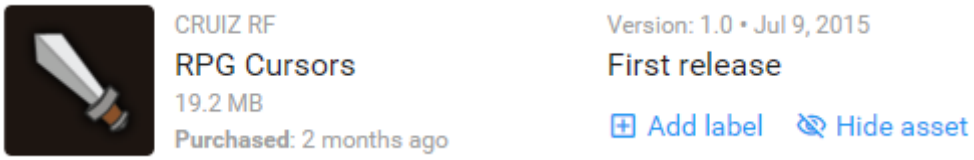


FIGURA 59. Iconos en la barra de habilidades

- **RPG Cursors**: las texturas que va a adoptar el cursor proceden de este Asset.



- **Cartoon FX Free**: algunas partículas proceden de este Asset.

http://jeanmoreno.com/unity/cartoonfxfree/

- **Mixamo:** todas las animaciones proceden de esta herramienta y repositorio.

4.5. Esquema de la arquitectura del juego

4.5.1. Arquitectura de Software

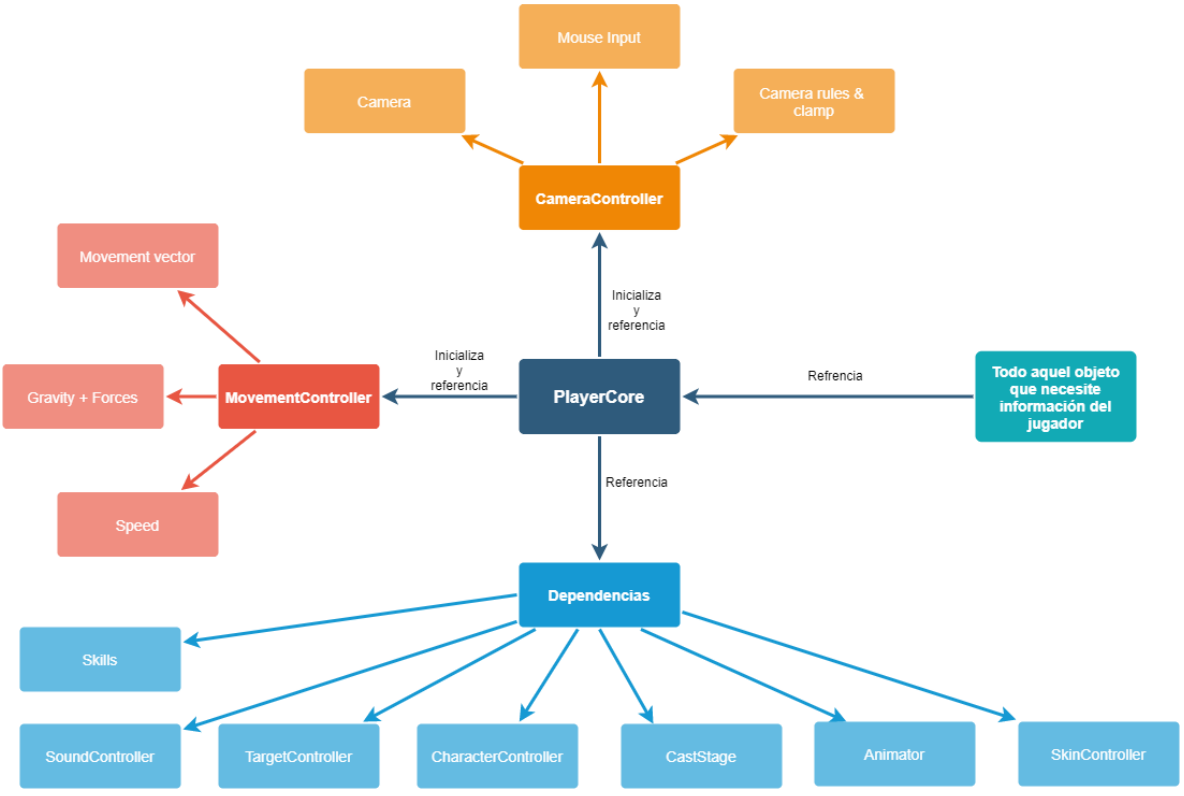


FIGURA 60. Arquitectura que conforma al jugador

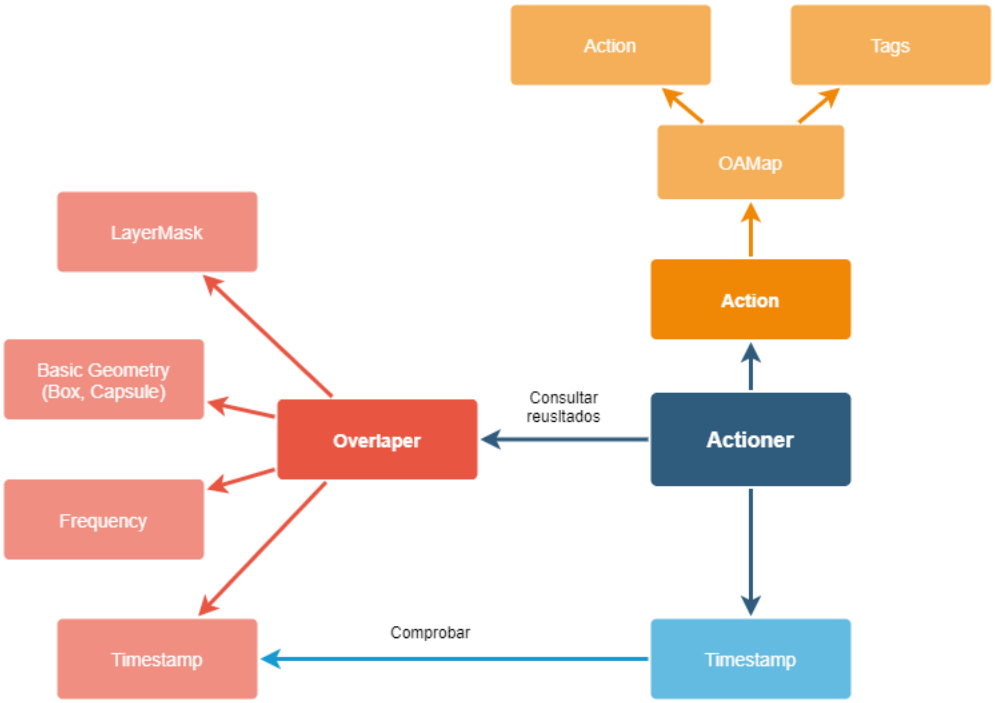


FIGURA 61. Arquitectura que siguen los efectos de área (Lava, fuego, laser…)

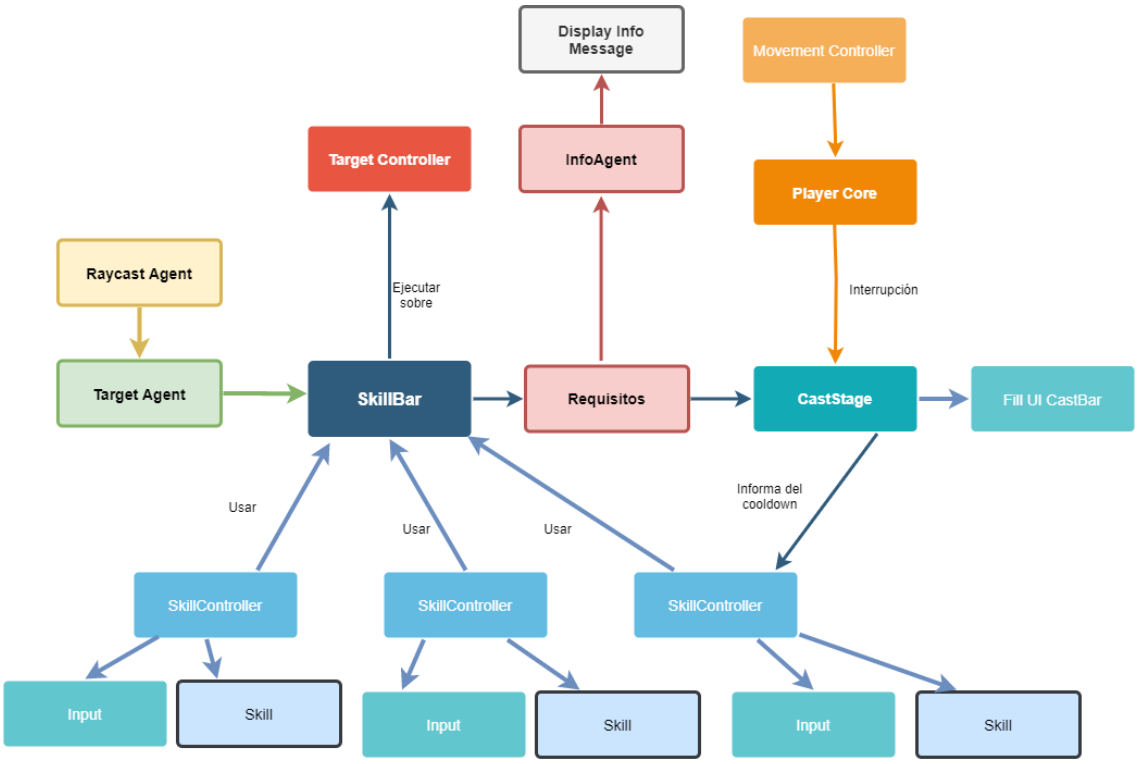


FIGURA 62. Arquitectura del sistema de habilidades del jugador

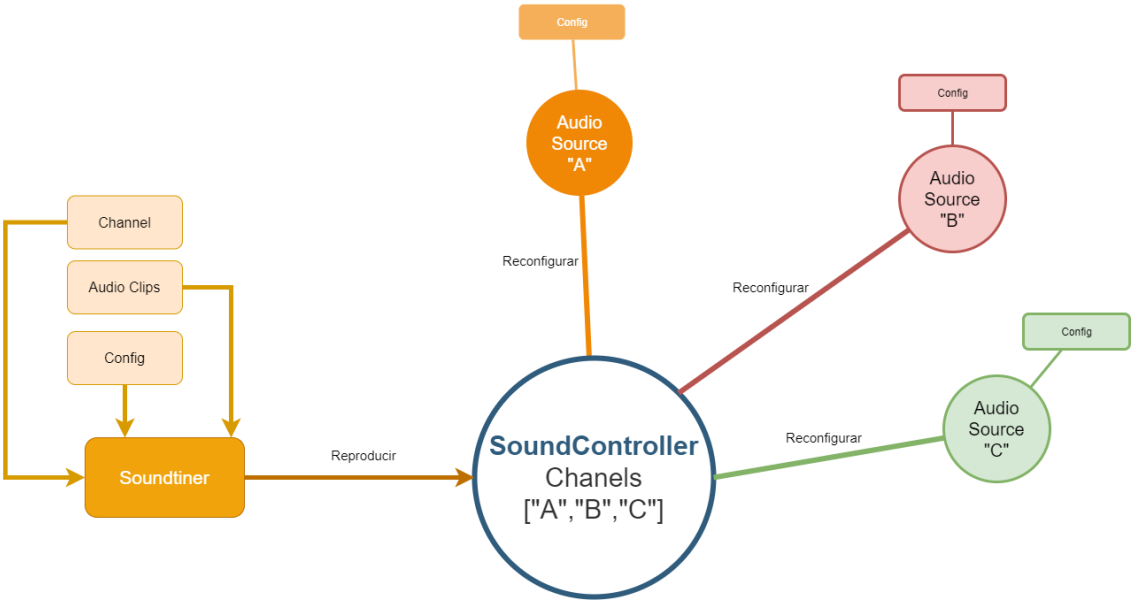


FIGURA 63. Arquitectura del sistema de sonido de las entidades

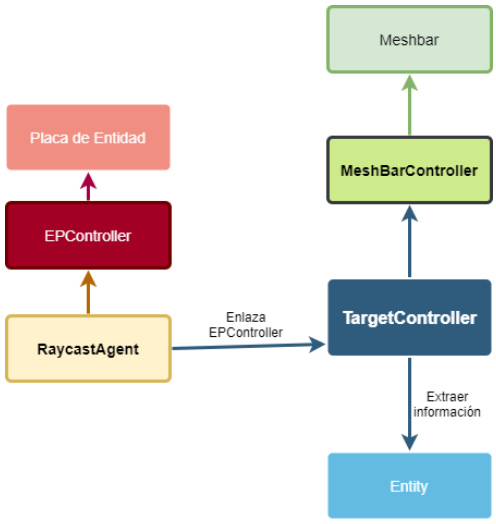


FIGURA 64. Arquitectura básica de cualquier entidad

4.5.2. Diseño y funcionamiento de la IA de los enemigos

El comportamiento del boss está definido a través de una máquina de estados, la cual es controlada por el script BossController.

Los estados tienen las siguientes características:

```
public interface IState
{
    void UpdateState();
    void OnEnter();
    void OnExit();
    void ToState(IState s);
    void Event(string eventName);
}
```

CÓDIGO 6. Definición base de un estado

- ***UpdateState***: definirá que debe estar comprobando continuamente en cada estado el boss.
- ***OnEnter***: acciones que deban pasar solo al entrar en el estado. Normalmente es activar animaciones o hacer algún cálculo que va a necesitar.
- ***OnExit***: acciones que deban pasar al salir del estado. Normalmente consiste en reiniciar las variables del estado.
- ***ToState***: método que permite transitar al siguiente estado.
- ***Event***: responde a eventos externos, por ejemplo, eventos de animación que llegarán etiquetados con un nombre. Un caso de uso es cuando la animación del rayo láser lanza el evento de que es el momento de activarlo y empezar a girar. Otro caso es en el momento que se da la palmada en la habilidad Quake, en ese momento comprueba la distancia del jugador e inflige daño.

Cada estado es responsable a que otro estado transitar, y el controlador del boss cuando detecta un cambio de estado, antes de hacer el cambio ejecuta *OnExit* y una vez hecho el cambio ejecuta *OnEnter*.

```
public IState CurrentState
{
    get
    {
        return currentState;
    }

    set
    {
        if(currentState != null) currentState.OnExit();
        currentState = value;
        if (currentState != null) currentState.OnEnter();
    }
}
```

CÓDIGO 7. Property de cambio de estado

A causa de que no debería influir el RNG y el jugador tendría que poder prever cual es la siguiente habilidad para decidir sus acciones, las habilidades se reproducen de forma cíclica y lo único que varía es la duración, por ejemplo, en el caso del estado Support, si se matan a todos los Anubiseth se cambia de estado o en su defecto si se tarda demasiado en matarlos también se cambia.

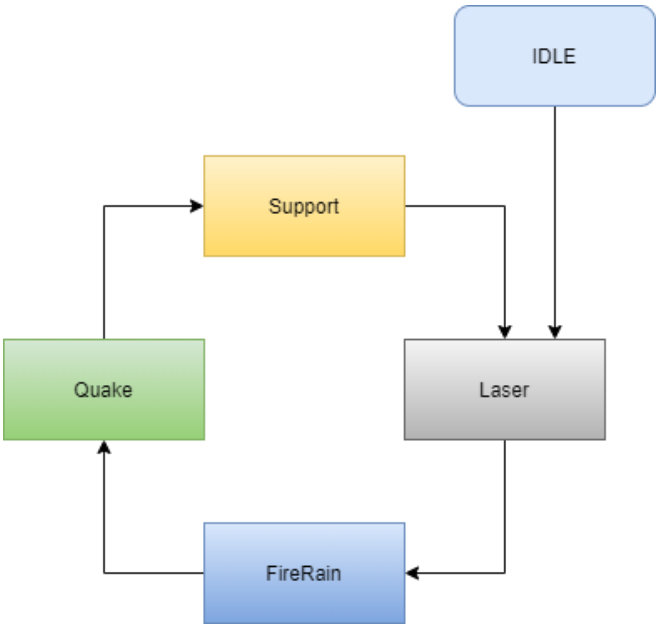


FIGURA 65. Diagrama de decisión de las habilidades del boss.

4.5.3. Jerarquía de escenas

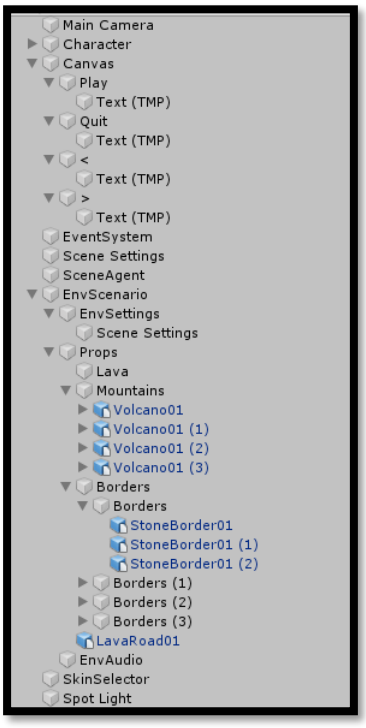


FIGURA 66. Jerarquía Game Objets menú

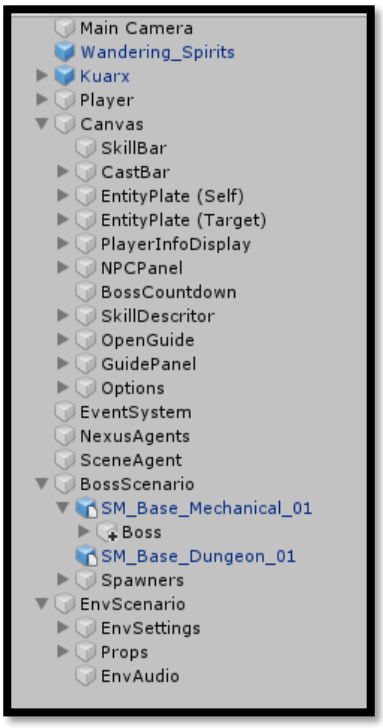


FIGURA 67. Jerarquía Game Objets Main Scene

5. Diseño de niveles

5.1. Mapa del nivel

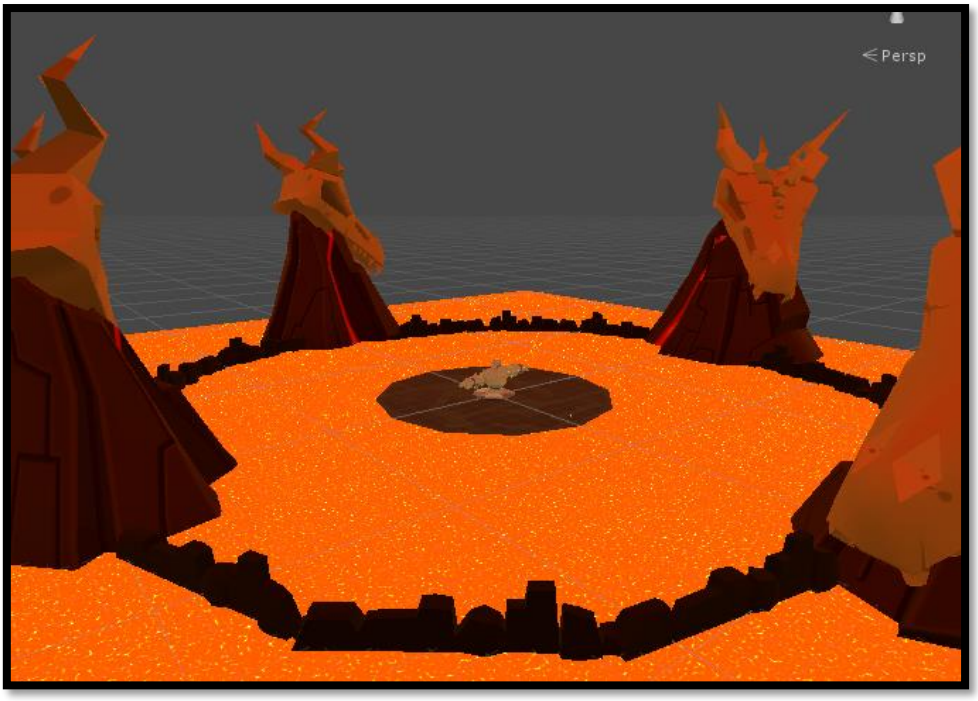


FIGURA 68. Vista general del escenario del juego desde arriba.

FIGURA 69. Escenario principal con la vista del jugador

5.2. Criterios de su diseño

- **Limitar el espacio de movimiento del jugador,** y así dificultar lo suficiente el combate. Por ejemplo, caen meteoritos que dejan fuego en el suelo, ese fuego va a permanecer durante 1 minuto y se irán acumulando, lo que reducirá más aún el espacio del jugador. Si no estuviera rodeado el campo de juego por lava, el jugador podría alejarse de tal manera que sería demasiado fácil y poco proporcionado.
- **Estaticidad del boss,** está decisión del nivel se debe a que cuadrar las animaciones de moverse y caminar (si no se refina bastante) puede dejar acciones extrañas y clipping de meshes. El boss es muy grande para la plataforma en la que se encuentra y si estuviera de pie sería complicado ver su parte superior sin hacerle un escalado. Por lo tanto, la decisión ha sido causada para facilitar y acortar el desarrollo al no tener que preocuparse por las animaciones del tren inferior y por motivos estéticos.
- **Oscuridad inicial,** el motivo de que antes de empezar el combate esté todo oscuro es para impresionar más visualmente cuando se inicia, para que el jugador tenga una sensación de intriga y también para que solo se fije inicialmente en el NPC que inicia el combate.

FIGURA 70. Oscuridad Inicial que oculta todos los elementos que no son el NPC

- **Marcadores de impacto,** ocurre antes de que caiga un meteorito, por ejemplo, la finalidad es de alertar el jugador para que tenga un tiempo de reacción para esquivar impactos.

6. Manual de Usuario

6.1. Requerimientos técnicos del hardware

Minimum requirements	Windows	macOS	Linux (Support in Preview)
Operating system version	Windows 7 (SP1+) and Windows 10, 64-bit versions only.	Sierra 10.12.6+	Ubuntu 16.04, Ubuntu 18.04, and CentOS 7
CPU	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support
Graphics API	DX10, DX11, and DX12-capable GPUs	Metal-capable Intel and AMD GPUs	OpenGL 3.2+ or Vulkan-capable, Nvidia and AMD GPUs.
Additional requirements	Hardware vendor officially supported drivers	Apple officially supported drivers	Gnome desktop environment running on top of X11 windowing system, Nvidia official proprietary graphics driver or AMD Mesa graphics driver. Other configuration and user environment as provided stock with the supported distribution (Kernel, Compositor, etc.)
	For all operating systems, the Unity Editor is supported on workstations or laptop form factors, running without emulation, container or compatibility layer.		

FIGURA 71.Tabla de requerimientos de hardware.

6.2. Instrucciones del juego

6.2.1. Primeros pasos

1. Ejecutar el ejecutable contenido en la carpeta Build/.

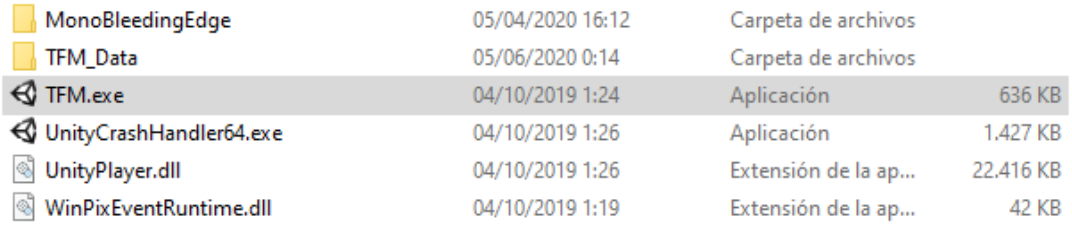


FIGURA 72. Contenido de la carpeta Build

2. Aparecerá una pantalla donde se podrá seleccionar el aspecto del jugador con las flechas laterales.

3. Una vez se tenga el aspecto deseado se clicka el botón PLAY.

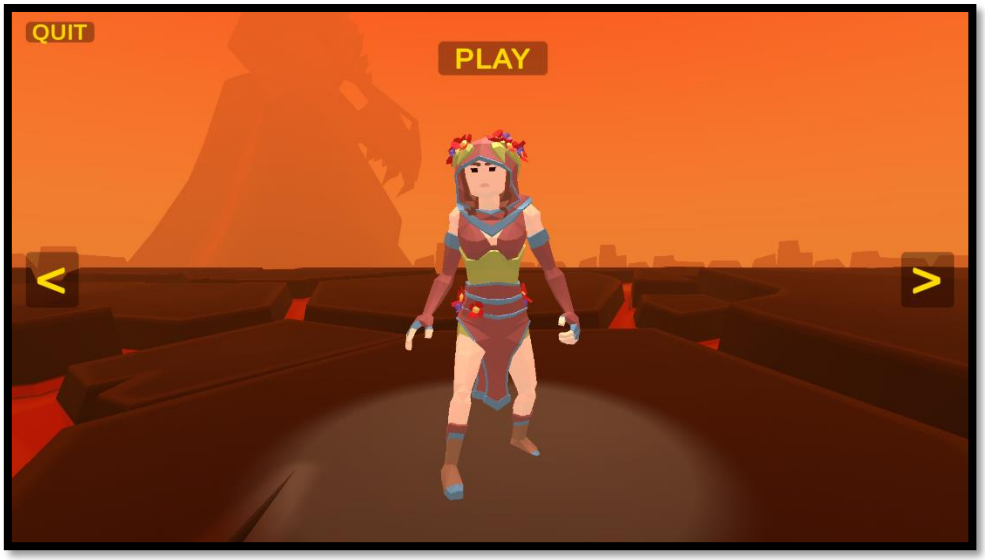


FIGURA 73. Pantalla de selector de aspecto.

4. Ya se está dentro del juego, para comenzar el combate se habla con el NPC llamado Kuarx y se selecciona la primera respuesta.

6.2.2. Controles

ÍNDICE DE CONTROLES	TECLAS
Seleccionar	Click Izquierdo
Hablar/Interactuar	Click Derecho
Caminar hacia adelante	W
Caminar hacia atrás	S
Rotar Izquierda	A
Rotar Derecha	D
Caminar al frente (dirección de la cámara)	Click Izquierdo + Click Derecho
Mover Cámara	Click Izq ó Dcho + Movimiento del Ratón
Saltar	Espacio
Menú Pausa	Escape
Habilidades	[1] - [9]

7. Conclusiones

En estos meses de trabajo he aprendido que la planificación no es tan ideal como se piensa en un primer momento, que Unity es un motor algo incómodo cuando un proyecto empieza a crecer (dado que la gestión de los archivos y las dependencias es menos que ideal), que a veces no debo obsesionarme tanto con el aspecto del código cuando se traduce en perder tiempo y que coger un papel y planificar la mayor parte de los componentes del juego y su arquitectura es esencial para no tener que dar pasos hacia atrás, aunque en algún momento tenga que dar alguno.

No se han logrado todos los objetivos planteados en un primer momento, ya que se propusieron dos tipos de conjuntos de habilidades y solo hay uno, faltan mejoras en las animaciones, algunos efectos de sonidos y visuales, no tiene online, falta por pulir más la UI en general y no me siento satisfecho totalmente con la arquitectura del código ya que entidades como el boss tienen muchos parámetros hardcodeados.

El motivo principal es la falta de tiempo, ya que compaginar un trabajo de 8 horas con un TFM es complicado y cansado, y el otro motivo es haber sido muy optimista

pensando que no iban a ocurrir baches técnicos para lograr algunos comportamientos. Verlo de manera global lo hace parecer más sencillo de lo que realmente es.

La planificación se ha tenido como una hoja de ruta, pero no como una "biblia" ya que es importante llegar a tiempo, pero no por ello se van a dejar pasos anteriores bug/rotos/incompletos, por lo que se ha priorizado pulir los primeros y recortar los objetivos de los siguientes pasos para que no suba la cantidad de fallos de implementación.

A pesar de surgir estos puntos negativos, satisfecho por haber podido terminar una versión jugable.

"*I try to avoid long-range plans and visions - that way I can more easily deal with anything new that comes up.*" Linus Torvalds

Las líneas de trabajo a futuro se deben centrar en mejorar el sistema de inteligencia del boss, en dar más información al jugador y mejorar todos los efectos visuales/sonoros y añadir aquellos que faltan. A partir de ahí y refactorizando un poco el código podría plantearse el salto a nuevos personajes y modo online.

8. Glosario

[1] **Open Source**: de fuente o código abierto.

[2] **Shaders**: programas del render pipeline que ejecutan en la GPU.

[3] ***Voronoi***: estrategia matemática para dividir un plano en tantas regiones como puntos tengamos de tal forma que a cada punto le asignemos la región formada por todo lo que está más cerca de él que de ningún otro.

[4] **Tiling**: embaldosar con una textura un material.

[5] **Uniform**: constantes que exponen los shaders al material.

[6] **Shader Graph**: herramienta de edición de Shaders de manera gráfica en Unity.

[7] **Background**: fondo.

[8] **MeshBars**: Barras 3D, renderizadas como un Mesh.

[9] **Mesh**: geometría junto con un material.

[10] **Boss**: jefe en inglés. Así es como se denomina a las entidades poderosas a batir en los juegos.

[11] **WoW**: World of Warcraft. Videojuego de Blizzard Entertainment.

9. Bibliografía

[1] Role-Playing Game (RPG), 23 de Abril de 2020 [Online]. https://www.techopedia.com/definition/27052/role-playing-game-rpg

[2] What is ARPG? [Online] https://honeysanime.com/what-is-arpg-definition-meaning/

[3] Top 100 RPG of All Time, IGN [Online] https://www.ign.com/lists/top-100-rpgs/

[4] Betrayal at Krondor (BaK) Gameplay, 22 de marzo de 2019 [Online] https://www.youtube.com/watch?v=HJGcKhPQAzs

[5] BaK Overview [Online] https://www.youtube.com/watch?v=a8COnpsL-H8

[6] BaK Review, HonestGamers [Online] http://honestgamers.com/2563/pc/betrayal-at-krondor/review.html

[7] Final Fantasy Saga [Online] https://es.wikipedia.org/wiki/Final_Fantasy_(franquicia)

[8] Final Fantasy X Overview [Online]: https://www.youtube.com/watch?v=L3KsizibaWA

[9] World of Warcraft - Wrathion Raid Guide [Online] https://www.youtube.com/watch?v=joKnHlp5QoM

9.1. Enlace de referencias de figuras:

1. Wrath of Ashardalon Board Game by Dungeons & Dragons. https://m.media-amazon.com/images/I/61ntkL0NpqL.jpg
2. The Legend of Zelda: Phantom Hourglass. Rol de acción y aventura. https://vignette.wikia.nocookie.net/zelda/images/4/4b/Jugabilidad_PH.png/revision/latest?cb=20130128104855&path-prefix=es
3. Árbol de habilidades Final Fantasy X https://cdn.hobbyconsolas.com/sites/navi.axelspringer.es/public/styles/1200/public/media/image/2015/01/433424-guia-final-fantasy-x.jpg?itok=7CTY1m15
4. Carátula Final Fantasy X https://i.ebayimg.com/images/g/r4IAAOSwyaNeQHaU/s-l640.jpg
5. Poster combinado FFX & FFX-2 https://i0.wp.com/nerfeados.com/wp-content/uploads/2019/05/FFX_Portada.jpg?fit=1920%2C1080

10. Anexos