Moon's Shadow:
The Last Flame

Práctica Final

Informática Gráfica

Autores

Lucas Sabater Margarit

Antoni Navarro Moreno

Laura Rodríguez López

Juan Arturo Abaurrea Calafell

Hugo Valls Sabater

Introducción

Moon's Shadow: The Last Flame es un prototipo de videojuego de ambientación fantástica, desarrollado en Unity, y con modelos realizados en Blender. Esta práctica tiene como objetivo desarrollar una escena con personajes animados e interactivos, y una atmósfera interesante. Todo esto, utilizando métodos de modelaje en 3D, iluminación y animaciones.

Para ello, hemos trabajado tanto en el diseño artístico como en la implementación técnica: creación de modelos, integración en Unity, configuración de materiales, luces dinámicas y efectos visuales.

Esta memoria recoge todo el proceso de desarrollo de la escena: desde la idea inicial y el diseño conceptual, hasta la parte más técnica.

Concepto y Diseño de la Escena

La escena (videojuego) se ambienta en un planeta destruido como repercusión de una guerra librada anteriormente. Los dragones, como Nur nuestro protagonista, se dedican ahora a exterminar los seres que vagan este mundo, con el objetivo de retornar este a su antiguo esplendor. Aunque en este documento no se detalla mucho la historia detrás del videojuego, se ha redactado más extensamente en el siguiente apéndice.

El videojuego en si relata uno de los muchos combates librados en este mundo, donde el protagonista, un pequeño dragón negro con una bufanda roja, lucha usando dos ataques principales: una lanza y su aliento de fuego.

Se detallan tres oleadas de enemigos, slimes con ataques melee en la primera, magos no muertos con ataques de proyectiles de hielo en la segunda, y el Golem final en la tercera.

Modelado 3D

Tenemos como modelos principales, el escenario (el coliseo) el protagonista (el dragón, Nur), y los enemigos (slimes, magos esqueletos y el jefe final, un golem de hielo).

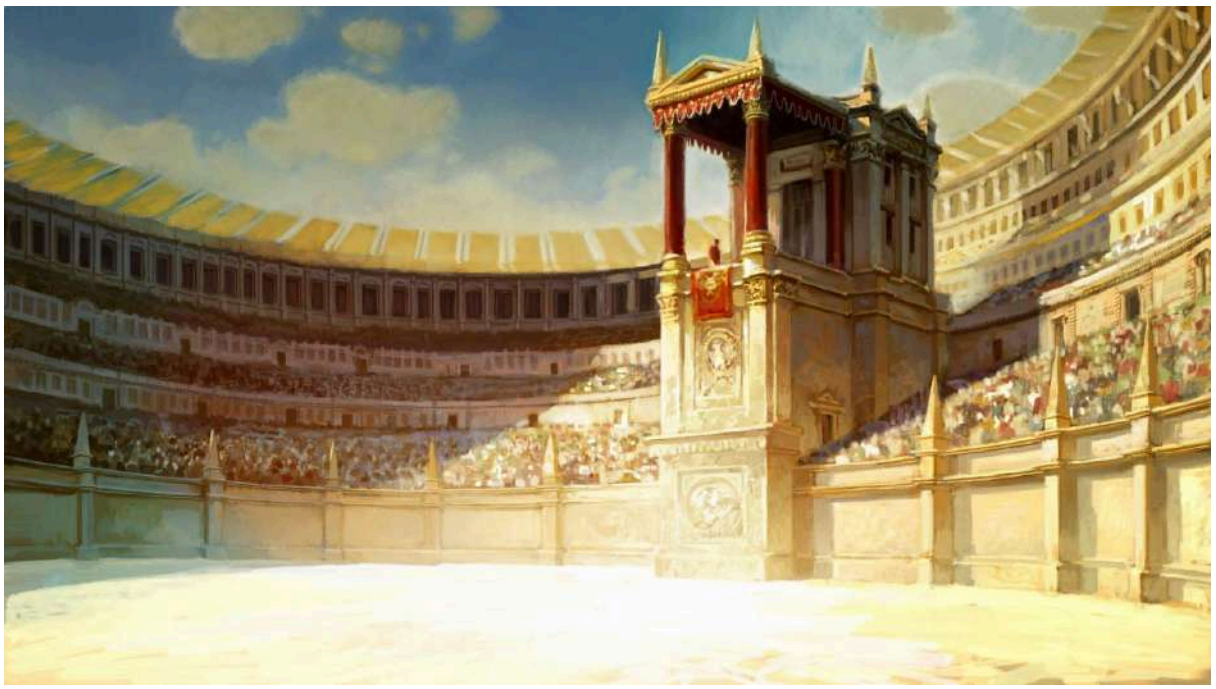
El escenario (Coliseo)

El coliseo es una estructura central dentro del escenario de combate del juego. Su función es servir como arena principal para las batallas del jugador, enmarcando no solo el espacio físico de los enfrentamientos, sino también reforzando la ambientación general del bioma helado.

Su diseño fue planteado desde un enfoque semi-realista, con proporciones y formas arquitectónicas inspiradas en coliseos reales, pero simplificadas para adaptarse mejor al estilo visual del juego.

Desde el principio se pensó en situar este coliseo en un entorno nevado, buscando crear un fuerte contraste visual con la estructura de piedra oscura y envejecida. Este bioma contribuye a generar una atmósfera hostil, pero solemne, donde el jugador siente tanto el frío del entorno como la amenaza del combate inminente.

El cielo nocturno y el eclipse parcial añadieron un componente estético y narrativo clave, con un juego de luces y sombras que atraviesa el coliseo dando más juego con la iluminación.

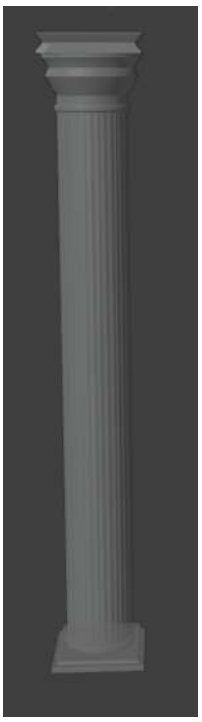


Imágenes conceptuales del coliseo y el ambiente

Para el modelado del coliseo[1], se comenzó primero con la creación de un único arco que representaría la estructura básica repetida a lo largo del edificio. Este arco fue modelado de forma independiente, cuidando que sus proporciones evocaran una arquitectura clásica, pero no excesivamente realista, buscando un equilibrio entre funcionalidad y estilo visual del juego.

Además del arco central, se añadieron columnas a los laterales para reforzar la idea de soporte arquitectónico. Estas columnas no fueron modeladas como cilindros simples, sino que se les aplicaron detalles básicos como líneas verticales de corte para simular segmentaciones, y una ligera base ensanchada para darles mayor estabilidad visual.

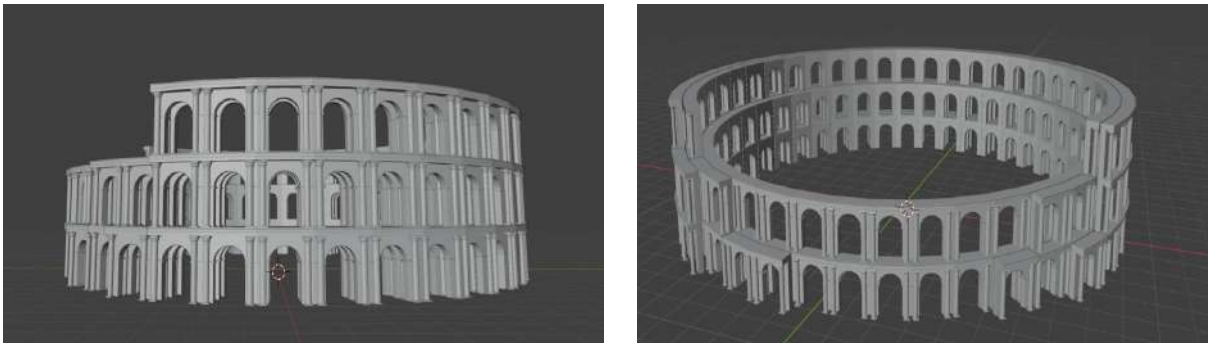
Una vez finalizado el módulo del arco, se revisaron sus normales, se aplicaron todos los modificadores activos y se corrigieron las geometrías innecesarias para asegurar que fuera una malla limpia y fácilmente replicable.



Arco base *Columna base*

Este módulo fue luego duplicado mediante el uso del modificador Array, ajustando la distancia de separación para que no se solaparan y formaran un ritmo visual regular. Para curvar toda la secuencia y darle forma circular al coliseo, se utilizó un Curve Modifier, con una curva Circular que se achantó un poco para que no fuera un círculo perfecto. Este sistema permitía controlar de forma precisa la apertura del coliseo y ajustar cuántos arcos se necesitaban para cerrarlo o dejar espacios abiertos, como la entrada principal.

Una vez construida esta base estructural, se procedió a añadir varias capas de muros para dar más volumen y presencia al coliseo. Se crearon muros superiores, sobre la fila de arcos, que simulan los niveles altos donde estarían las gradas o zonas de vigilancia. También se añadieron muros exteriores, rodeando la estructura inicial, con una separación leve respecto al núcleo principal del coliseo. Estos muros funcionan como contrafuertes o capas defensivas, y ayudan a romper la silueta general del edificio, evitando que se vea demasiado uniforme o simple.

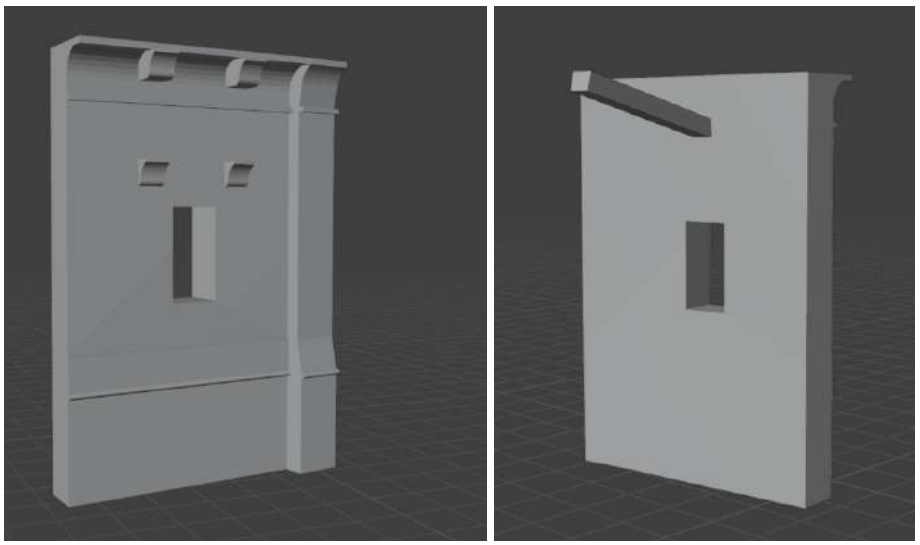


Coliseo Base desde diferentes puntos de vista

De la misma manera que se realizaron los arcos, se crearon dos tipos adicionales de muros para completar la estructura del coliseo, cada uno con características específicas para cumplir con distintas funciones estéticas y funcionales dentro del diseño general.

El primer muro se modeló comenzando con un bloque rectangular, al que se le aplicó un corte rectangular en la parte central utilizando un boolean modifier para crear una abertura limpia y precisa. Las esquinas de esta abertura se suavizaron para evitar bordes afilados, manteniendo un diseño más fluido. En la parte superior del muro, se agregaron detalles de corte, creando una forma ligeramente arqueada y un pequeño saliente que simula una moldura arquitectónica. Los bordes laterales fueron suavizados y ligeramente redondeados para dar una sensación de solidez y robustez. La parte inferior se mantuvo más recta, con una ligera ampliación hacia los lados, ofreciendo una base estable y consistente para el muro.

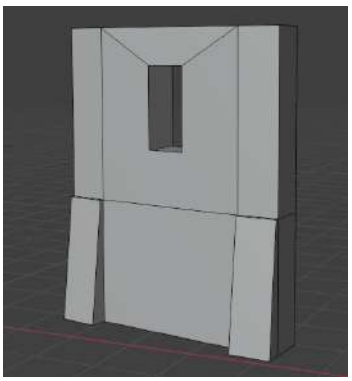
Este muro fue diseñado para ser modular, por lo que las proporciones y la disposición de los cortes y detalles se pensaron para asegurar que se pudieran replicar de forma coherente a lo largo del coliseo.



Base del Tipo de muro 1

El segundo tipo de muro se modeló con un enfoque más sólido y anguloso, buscando un diseño robusto que complementará la estructura general del coliseo. Se partió de un bloque rectangular, similar al primer tipo de muro, pero con un enfoque más simple en cuanto a la geometría.

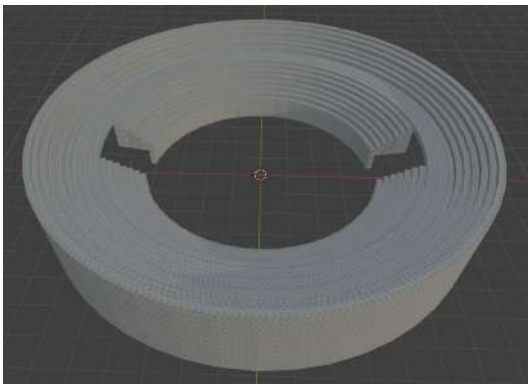
Se le realizó un corte rectangular en la parte central, utilizando un boolean modifier para mantener las aberturas limpias y definidas. A diferencia del primer muro, las esquinas de la abertura no fueron suavizadas, manteniendo una estructura más rígida y recta. La parte superior e inferior del muro presentan ángulos rectos, pero las secciones laterales del muro fueron cortadas en diagonal. Este corte inclinado en los laterales agrega un detalle visual interesante y al mismo tiempo permite que el muro tenga una base más ancha, aumentando su estabilidad visual y física.



Base del Tipo de muro 2

Las escaleras se construyeron a partir de un cubo base, duplicándolo con incrementos de altura para formar un tramo escalonado de forma manual. Una vez completado, se unieron los cubos y se aplicaron los modificadores Array y Curve, al igual que en los muros, para que siguieran una trayectoria circular adaptada al coliseo.

Este método permitió crear unas escaleras funcionales y visualmente coherentes con el resto de la estructura, rodeando el interior del coliseo y creando unas gradas amplias y estéticas a la vez.



Gradas del coliseo

El palco fue modelado como una estructura elevada y cerrada parcialmente, pensada para situar al boss mientras observa la arena antes de entrar en combate. Su posición dominante y su altura ayudan a reforzar la idea de que el jugador está siendo observado constantemente desde un lugar de poder.

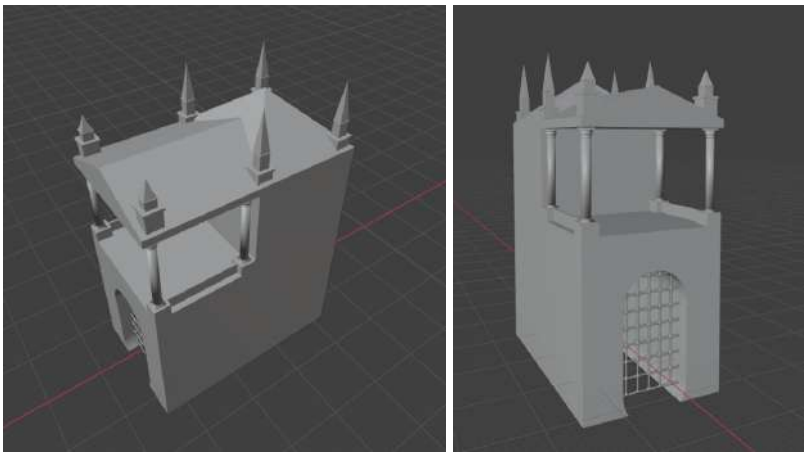
Se comenzó con un bloque base en forma de cubo al que se le aplicó un corte en arco en la parte inferior para crear una entrada, utilizando booleanas para facilitar la forma.

Sobre esta base se extruyó un segundo nivel, creando una plataforma abierta que simula una especie de balcón fortificado. Para definir sus bordes,

se añadieron pequeños bloques en las esquinas y laterales, actuando como barandillas.

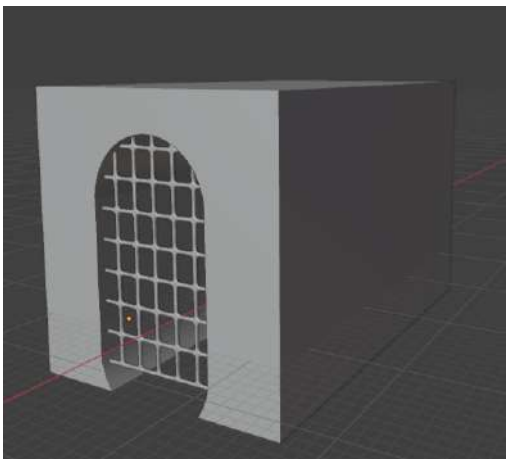
Luego se añadieron cuatro columnas iguales a las columnas de los arcos. Estas columnas sirven como soporte para el techo, formado por planos inclinados ensamblados con un marco rectangular.

Finalmente, se colocaron pináculos puntiagudos en las esquinas del techo, modelados con conos extruidos sobre bases cuadradas.



Palco desde diferentes puntos de vista

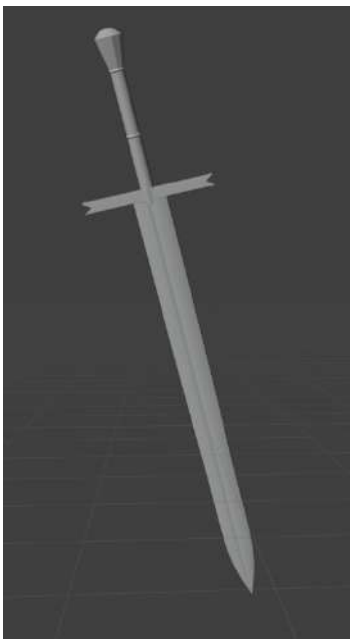
La entrada del coliseo fue modelada a partir de un bloque rectangular al que se le dio un arco en la parte inferior, usando un boolean modifier para crear la abertura principal. En el centro, se añadió una rejilla de barras verticales y horizontales, simulando una puerta de hierro reforzada.



Entrada al coliseo

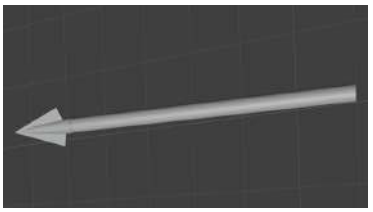
En el suelo de la arena se han colocado varias espadas y lanzas como elementos decorativos y funcionales, para reforzar la atmósfera de un coliseo de combate. Estos elementos fueron modelados de manera sencilla pero detallada para asegurar que se integrarán adecuadamente en el entorno sin sobrecargar la escena.

Las espadas fueron creadas comenzando con un cilindro largo que se fue moldeando para crear la hoja, la empuñadura y el pomo. Se utilizó una técnica de subdivisión para darle una forma afilada a la hoja, y se añadió un cruce en la empuñadura para completar el diseño. Este diseño de espada tiene una estética medieval simple pero efectiva, con un detalle mínimo para evitar que se distorsione la simplicidad visual del entorno.



Espada Básica

Por otro lado, las lanzas fueron modeladas a partir de un cilindro largo al cual se le añadió una punta triangular afilada, siguiendo un proceso similar al de las espadas, pero con un solo extremo puntiagudo. Las lanzas tienen un diseño más delgado y aerodinámico, pensado para representar el tipo de armas arrojadizas o de largo alcance que podrían haber sido usadas en combates en la arena.



Lanza Básica

Ambos objetos fueron colocados aleatoriamente en el suelo de la arena, simulando el abandono o el uso pasado durante combates previos. Estos detalles, aunque simples en su modelado, ayudan a dar más realismo y contexto a la escena del coliseo.

Se han modelado varias antorchas[2] con el objetivo de colocarlas alrededor del coliseo, para aportar iluminación adicional y enriquecer la atmósfera del entorno.

Se partió de un cono alargado como base, al que se le añadieron cilindros y anillos para definir la estructura central. En la parte superior se colocaron varias piezas abiertas orientadas hacia afuera, simulando un soporte para las llamas. Finalmente, cada antorcha se unió a una placa rectangular con remaches en las esquinas, que servirá como anclaje a los muros del coliseo. Estas antorchas no solo cumplen una función estética, sino que también permiten introducir más fuentes de luz en la escena, ayudando a crear contrastes visuales y reforzar la ambientación nocturna del escenario.

Antorcha del Coliseo

Para construir la ambientación del coliseo, se ha trabajado en distintos elementos que refuerzan la atmósfera invernal y nocturna del escenario. El suelo se ha realizado utilizando una textura de nieve aplicada sobre un plano que cubre toda la base de la arena, dando la sensación de un entorno frío y hostil. Para complementar esto, se ha colocado en el techo un segundo plano oculto a la vista, encargado de emitir partículas de nieve mediante un sistema de partículas, simulando una nevada constante durante la escena.

En cuanto al cielo, se ha utilizado una imagen HDRI de noche que proporciona una iluminación ambiental tenue y azulada. Finalmente, se ha modelado un eclipse utilizando dos esferas superpuestas: una como luna principal y otra desplazada parcialmente para generar la sombra que crea el

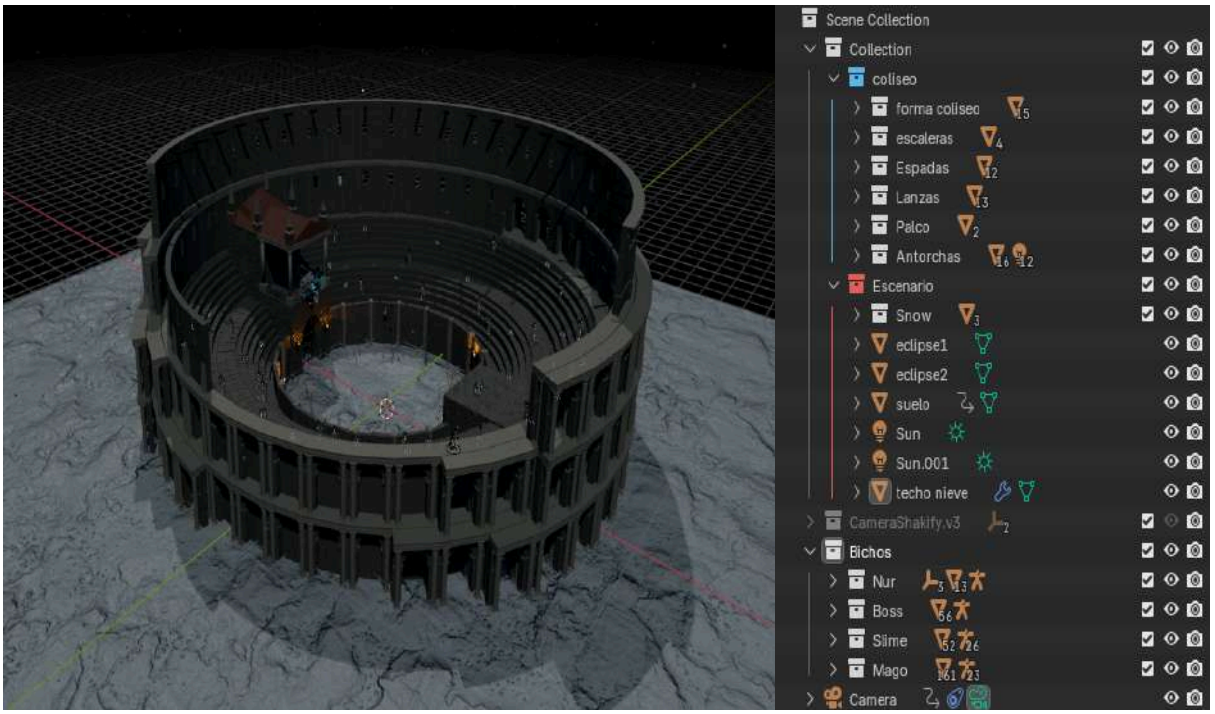
efecto del eclipse. Estos elementos en conjunto ayudan a crear una atmósfera coherente, inmersiva y visualmente impactante.

Eclipse y HDRI del cielo

Para finalizar el proyecto, se han colocado tanto al protagonista como a los enemigos dentro del coliseo, completando así la escena principal del juego. Los personajes fueron posicionados estratégicamente en distintas zonas de la arena para representar la disposición inicial de un combate: el protagonista en el centro, con los enemigos rodeando el perímetro o distribuidos en entradas específicas. Esto no solo sirve para visualizar la jugabilidad, sino también para probar escalas, iluminación y coherencia entre todos los elementos modelados.

Además, se organizó toda la escena mediante una jerarquía de objetos clara y estructurada, agrupando las partes del coliseo (muros, arcos, escaleras, palco, armas y decoración) por categorías, al igual que los personajes y elementos ambientales (partículas, luces, cielo, etc.).

Aquí podemos ver el resultado final juntando todos los modelos con las texturas, junto a su jerarquía de objetos:



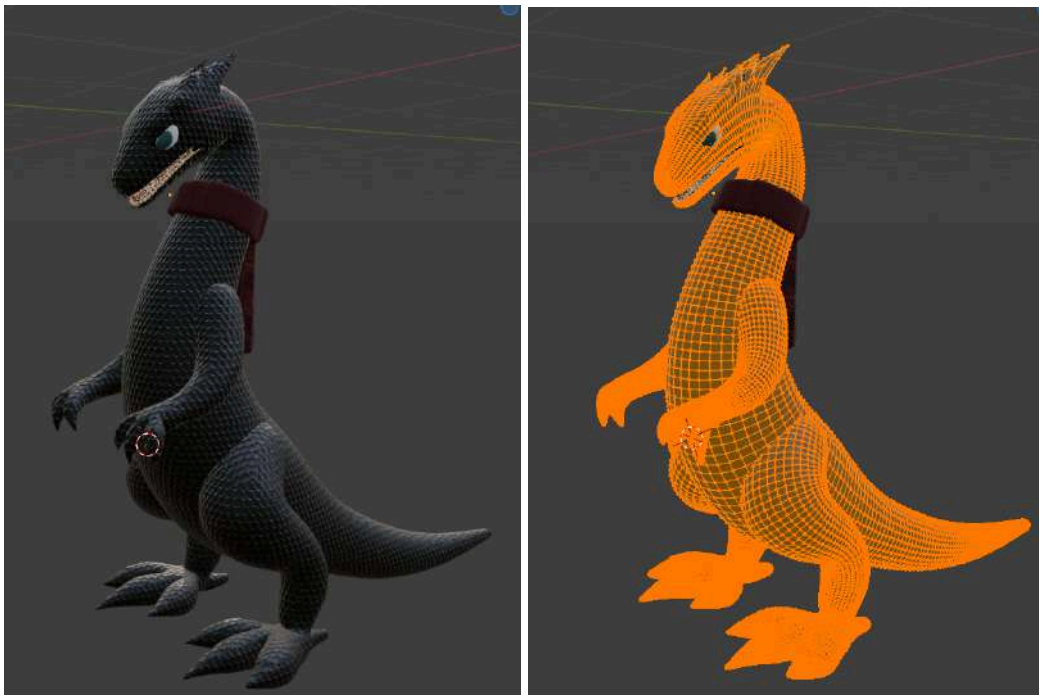
En el proceso final, se intentó añadir nieve sobre los muros del coliseo para aumentar la coherencia visual con el bioma invernal. Sin embargo, al implementar esta nieve mediante un addon, se detectó que el rendimiento se veía seriamente afectado: la escena tardaba demasiado en cargar y se volvía pesada al manipularla, lo que obligó a descartar esta idea por cuestiones de optimización. Además, al exportar el modelo a Unity, se descubrió que una cara del marco del coliseo faltaba. Debido a esto, fue necesario rehacer esa parte del coliseo desde cero, corrigiendo la geometría y asegurando que todas las caras estuvieran bien cerradas y orientadas, lo que implicó una revisión general del modelo antes de considerarlo terminado.

El protagonista (Nur, el dragón)

El protagonista del juego es un pequeño dragón llamado Nur. Es el modelo que controla el jugador en 3ª persona, y blande una lanza con su pata superior derecha. Lleva al cuello una bufanda roja.

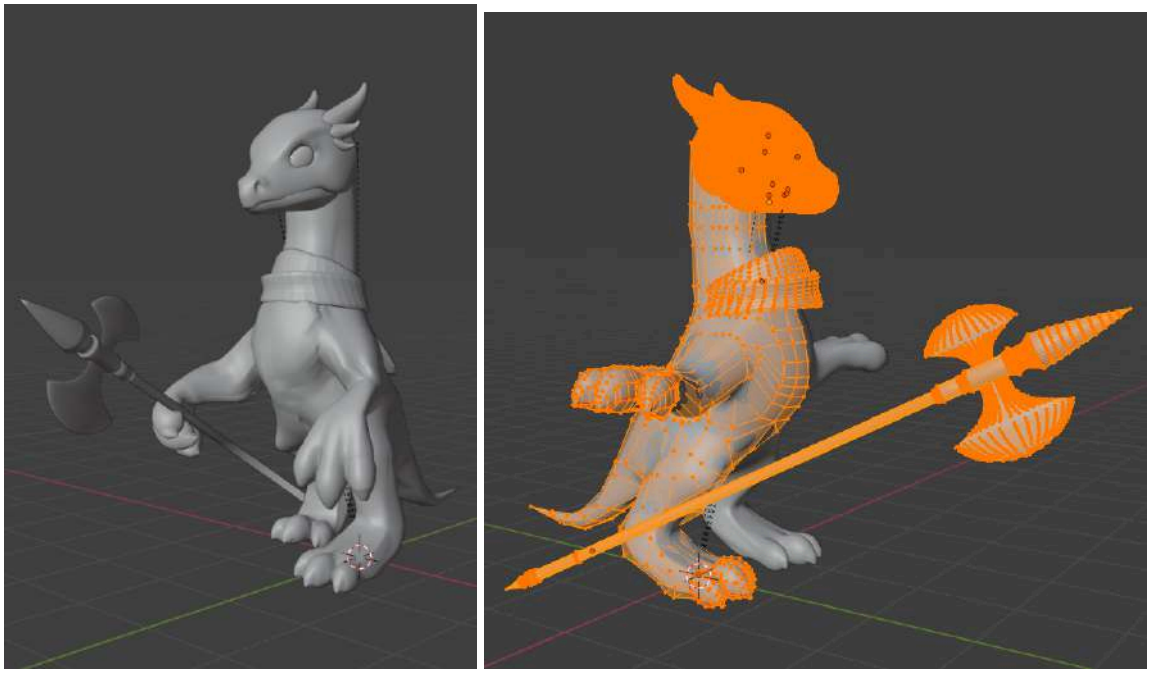
El modelo final no es el modelo original. Se empezó modelando otro dragón diferente, pero se tuvo que volver a empezar desde el principio tras

intentar importar el modelo a Unity. Debido a las malas prácticas a la hora de modelar, con vértices intercalados entre sí, el anterior modelo era inviable.

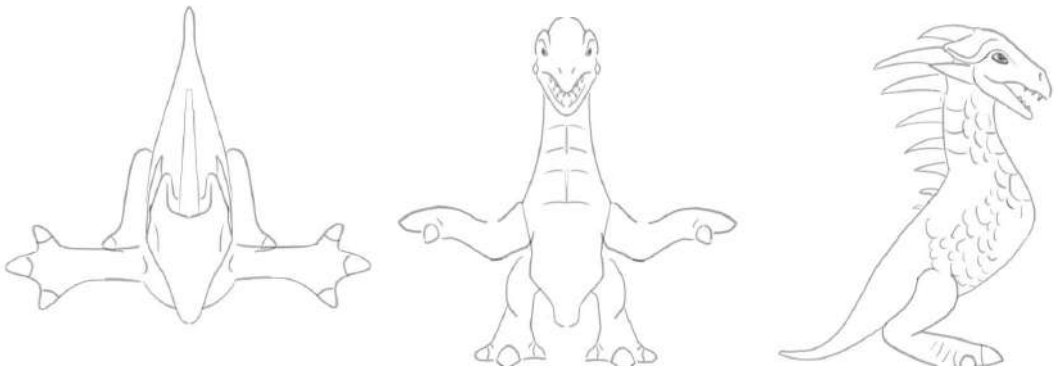


Modelo anterior de Nur, junto a su vista con vértices

Así, el modelo se tuvo que descartar completamente y se hizo otro desde cero. Esta vez, se tuvieron en cuenta los vértices, las intersecciones de planos, y se simplificó mucho la figura.



Para realizar el nuevo modelo, se siguió un tutorial de YouTube de creación de personajes[3]. El proceso comienza con imágenes de referencia, que se sitúan en los ejes, de manera que cada vista del dragón puede ser modelada a partir de una imagen.



Las imágenes de perspectiva se obtuvieron a partir de una figurita real de un dragón, fotografiado desde las tres vistas, y después trazado digitalmente para más claridad. Las imágenes superiores son el resultado de este proceso. Podemos ver que, al final, sirvieron más para el modelado del cuerpo que del busto, ya que este se decidió cambiar más adelante.

Siguiendo el tutorial, se empezó con un cubo simple. A partir de este, y alineándose con las imágenes de perspectiva, con la herramienta de *Extrude*, se fue acotando más a la forma deseada. Esto se hizo para cada una de las tres vistas.

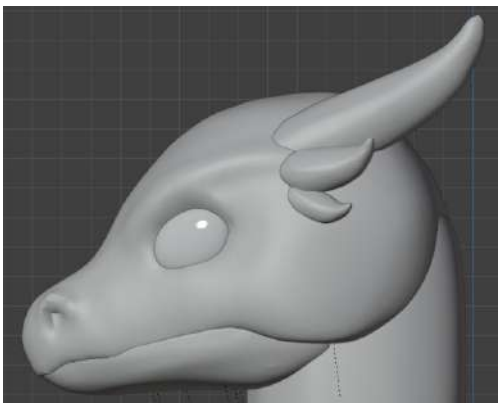
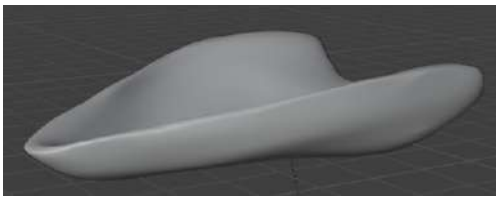
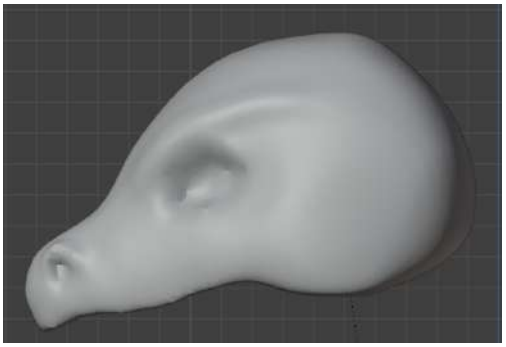
Dado que el modelo seguía siendo demasiado cuadrado, se hizo una siguiente acotación, sin aún redondear los bordes. Simplemente, se perfilaron los bordes, siendo el modelo aún “cuadrado”, pero más fiel a las imágenes. Para agilizar el proceso, todo esto se hizo bajo el modificador *Mirror*.

Finalmente, se aplica el modificador *Subdivision Surface*, para conseguir un modelo más realista.

A continuación, se modeló la cabeza[4]. Debido a decisiones de diseños, no se modeló a partir de las imágenes de referencia anteriores. Se decidió, en cambio, generar con IA una imagen que se acercara más a la visión general de Nur como protagonista, y utilizarla de referencia.

La base es un cubo con *Subdivision Surface*, básicamente una esfera. A diferencia del cuerpo, la cabeza empieza siendo directamente un modelo "realista", en el sentido de que en ningún momento es cuadrado. A partir de la base, en el modo *Sculpting*, damos forma a la cabeza.

Con las herramientas proporcionadas por Blender de escultura, sobre todo *Elastic Grab*, se va dando forma a la propia cabeza, basándonos siempre en la imagen de referencia. Debido a que una de las animaciones de Nur debía ser un ataque escupiendo fuego, se tuvieron que esculpir la cabeza y la mandíbula por separado, siguiendo la misma técnica.

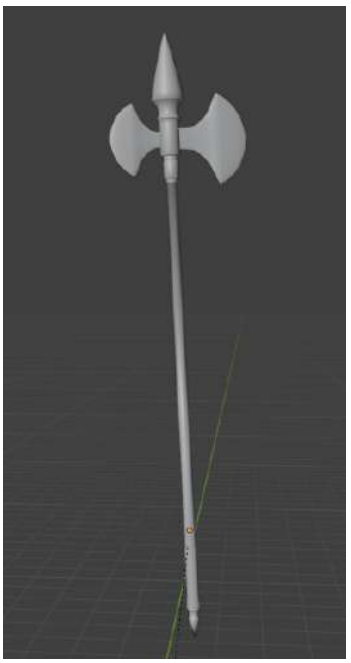
Una vez formada la base de la cabeza, añadimos los demás elementos. Los ojos son simples esferas deformadas para ser óvalos. Los cuernos se hicieron también a partir de esferas, deformadas en *Sculpting*, y finalmente, quedaban los detalles del interior de la boca.


Se esculpieron los dientes, en interior de la boca, y la lengua, todos con la misma técnica de *Sculpting*.

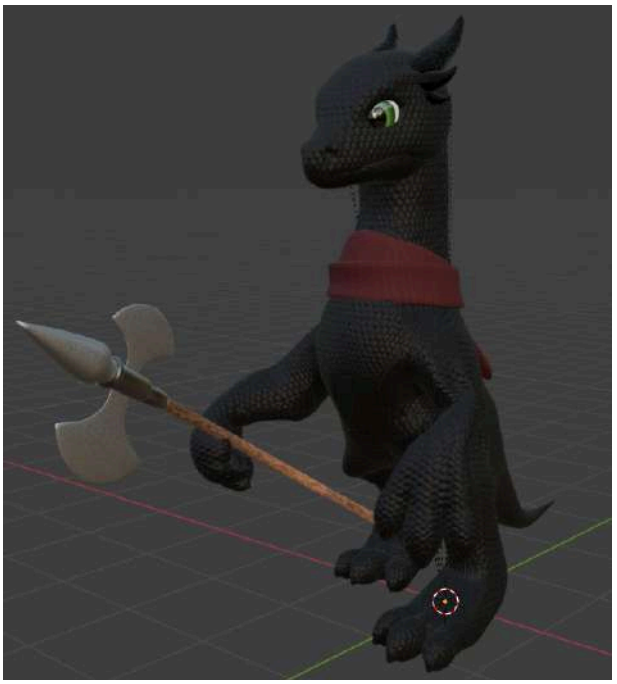
A continuación se modeló la bufanda[5]. La base es un cubo con las caras superior e inferior eliminadas, creando una forma cuadrada con un

plano. A partir de los modificadores *Subdivision Surface*, *Solidify* y *Shrinkwrap*, además de darle forma, obtenemos una bufanda sencilla.

El arma principal que utiliza el dragón protagonista del juego se trata de una hacha de doble hoja montada sobre un largo mango de madera, diseñada para combinar fuerza y alcance en sus ataques. El modelado se realizó por partes: el mango se creó a partir de un cilindro alargado con ligeras deformaciones para simular la forma irregular de la madera, mientras que las hojas del hacha se modelaron a partir de formas curvas simétricas extruidas y suavizadas, unidas mediante piezas metálicas cilíndricas que actúan como refuerzos estructurales. La parte superior incluye también una punta afilada que permite que el arma funcione tanto como hacha de corte como lanza de empuje.



Finalmente, así queda la jerarquía de objetos en el modelo final de Nur.

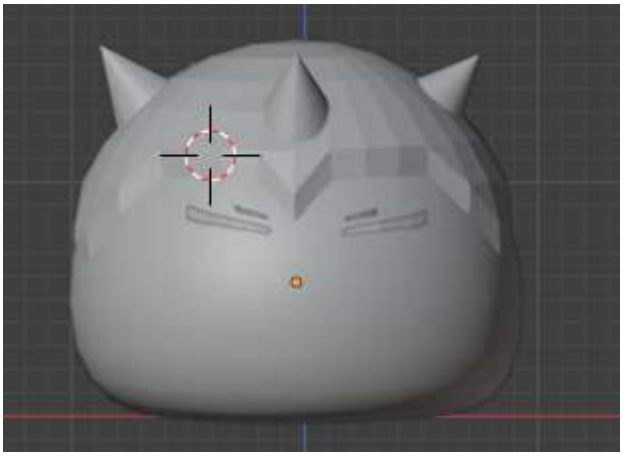


Los enemigos

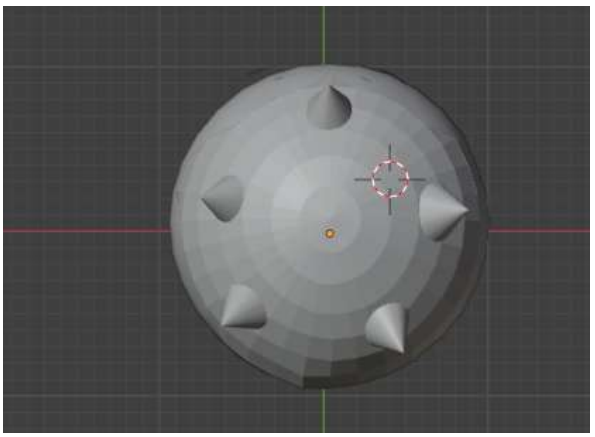
Slimes

El slime es el enemigo básico del juego. Su función es perseguir al jugador dando saltos y golpearle cuerpo a cuerpo, mientras el jugador trata de huir de él.

Su modelo fue planteado para que este tuviera un toque más de caricatura, con un cuerpo redondo y claramente identificable, donde sus movimientos se pudieran identificar a simple vista. La otra opción de posible slime era como una baba viscosa sin una forma tan definida. Pero se decidió no ir por ese camino.



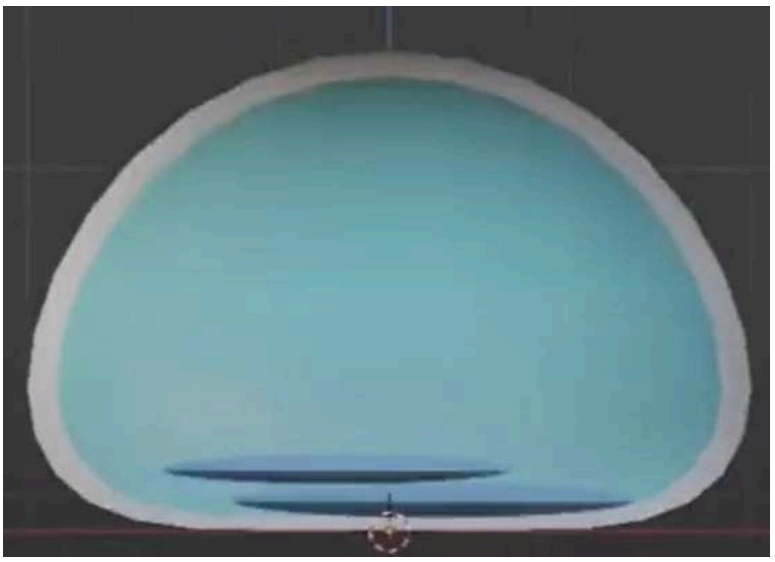
Así mismo, a este decidimos hacerle un casco con pinchos, para que su método de ataque fuese más identificable y claro, que simplemente recibir daño al ser tocado por él.



Para su modelado 3D, se partió de una UVsphere a la cual se le dio una forma algo más achatada y una base, que sería el punto de contacto con el suelo del slime.

Así mismo, este contaba con una "segunda piel" la cual servía para definir mejor el contorno de su tamaño, así como dar una imagen más similar a un líquido.

Del mismo modo, este iba a ser semitransparente y constaría de objetos en su interior[6]. Pero finalmente se descartó esa opción, pues al ir desarrollando más a fondo el contexto del juego así como la coherencia gráfica, se vio una mejor idea darle una textura más sólida. De las texturas del slime y su desarrollo se hablará más adelante.



Inicialmente, el casco se realizó plegando la parte inferior de la esfera sobre sí misma, y luego añadiendo puntas en los extremos, una mala idea, pero todavía inexpertos pareció una buena idea. Este casco presentó normales invertidas y aristas que no se conectaban entre sí, y al final se decidió eliminar y crear el casco final.

El casco final se realizó eliminando la mitad inferior de una UVsphere, y dando un mayor volumen a sus caras y conectandolas entre sí y al slime. Este ahora posee una diferenciación mucho más marcada y limpia que el casco inicial.

Versión antigua del slime.

Los “mofletes” del slime fueron un elemento que se decidió eliminar, pues estos salen demasiado de nuestra imagen del juego, y con respecto a los ojos y las cejas, si que también parecen más falsas, pero ayudan visualmente a diferenciar la orientación del slime en todo momento de una forma no tan descarada.

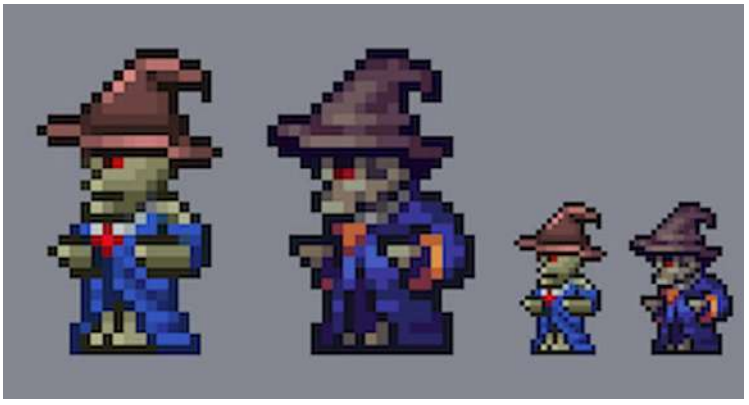
Los ojos se hicieron creando un cuadrado y cortándolo para que tuviera diferentes trozos y caras para poder moldearlas más fácilmente al cuerpo del slime.

Versión final del enemigo básico Slime

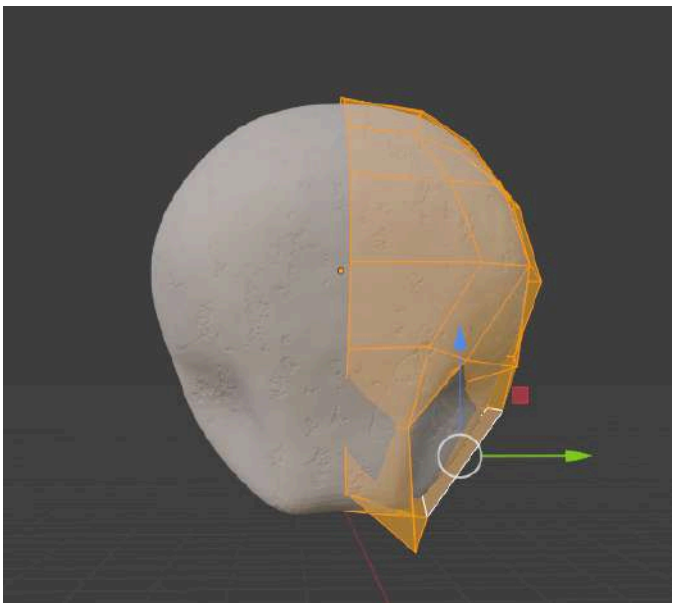
Mago esqueleto

El esqueleto es uno de los enemigos a distancia del juego. A diferencia del slime, este no persigue al jugador directamente, sino que le lanza proyectiles desde una posición más segura. Si el jugador se acerca demasiado, el esqueleto comienza a caminar marcha atrás para evitar el contacto directo. Su comportamiento más estratégico debía reflejarse también en su diseño visual, por lo que se optó por una figura más definida y antropomórfica, con un estilo más detallado que el de los enemigos básicos.

El modelo del mago esqueleto fue concebido a partir de dos referencias claras: los esqueletos de LEGO, por su estética modular y simplificada, y el personaje Tim del videojuego *Terraria*, cuya silueta y accesorios encajaban muy bien con lo que queríamos representar. Estas influencias ayudaron a crear una silueta reconocible y coherente con el resto de enemigos del juego.

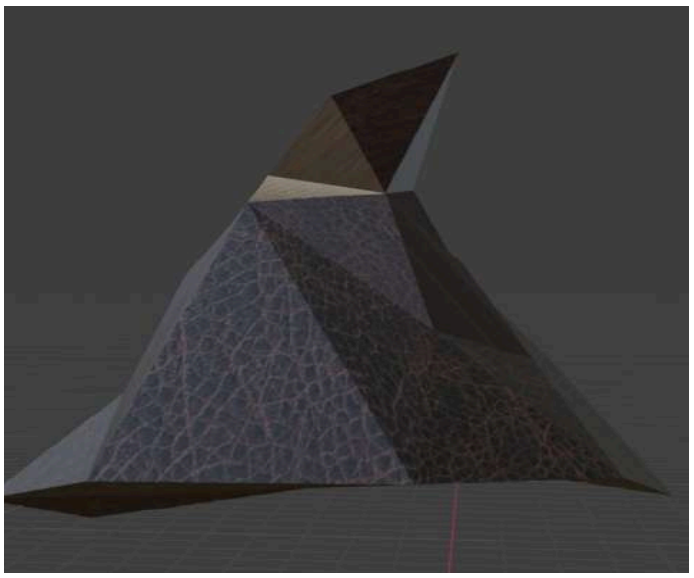


El desarrollo del modelo 3D se comenzó por la cabeza de este, pues se veía como una parte esencial para después poder obtener las proporciones correctas del personaje. Esta inició como un cubo, al cual se le dio una forma alargada y se le fueron añadiendo elementos o modificando las conexiones de los nodos para obtener el resultado final.

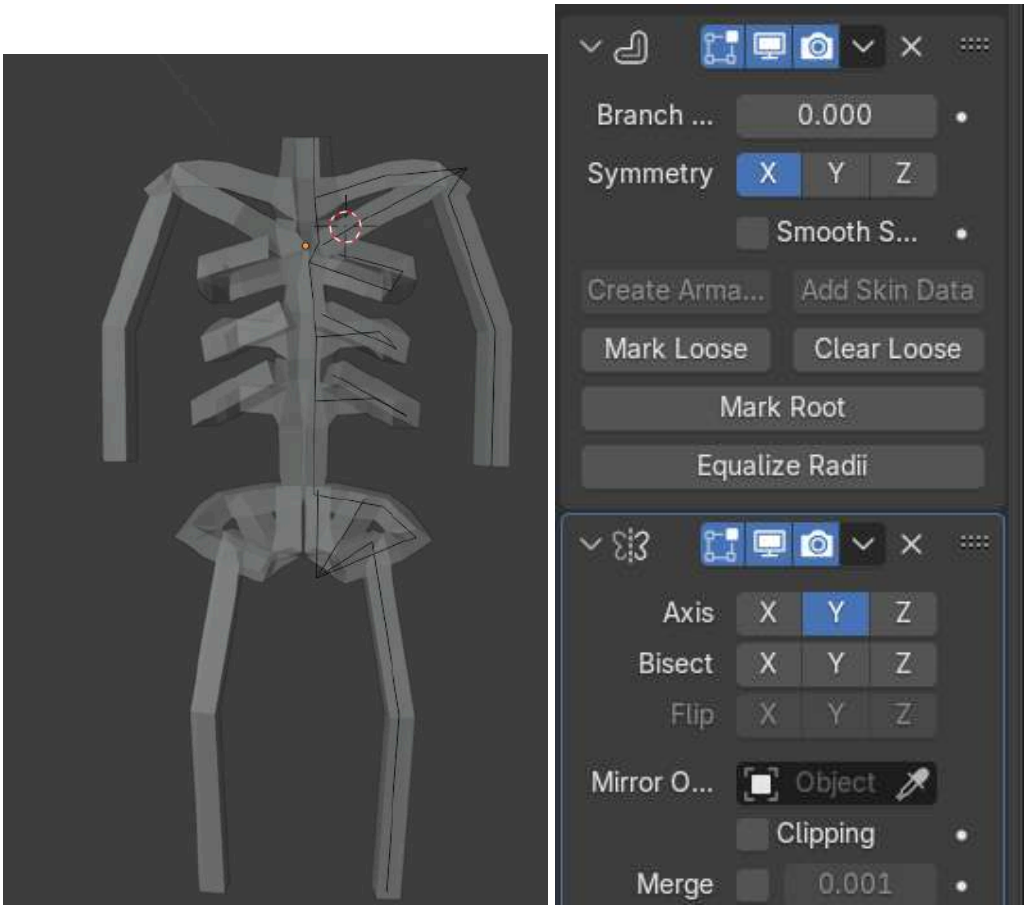


Así mismo, como se puede observar, se utilizó un mirror el mirror modifier junto al subdivision surface para darle una estética más plana, aunque este último se decidió eliminar tras finalizar el diseño del cuerpo, pues la estética más puntiaguda encajaba mejor con el resto de elementos.

El sombrero se inició situando un torus alrededor de la cabeza para obtener el diámetro, y un cono dentro de este para simular la altura. Estos elementos no están en la versión final del sombrero, pues presentaban problemas a la hora de fusionarlos, o no daban la estética que se buscaba para este, pero fueron muy útiles como referencias.



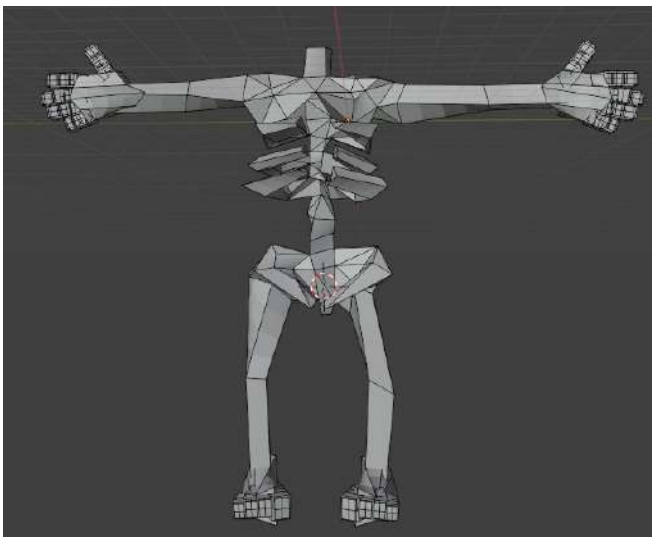
El modelado del cuerpo del esqueleto pasó por distintos estados. Inicialmente, se planteó realizar su cuerpo mediante el uso del Skin Modifier y el Mirror Modifier[7]. Su desarrollo se hizo desde una arista con dos nodos, y añadiendo poco a poco nodos para ampliar las partes del cuerpo. Se comenzó desarrollando la columna, utilizando la cabeza del personaje para la referencia del tamaño (pues se intentó que mantuviera unas proporciones humanas dentro de lo que cabe) así como realizando particiones en las aristas para extraer de estas elementos como los brazos y las costillas.



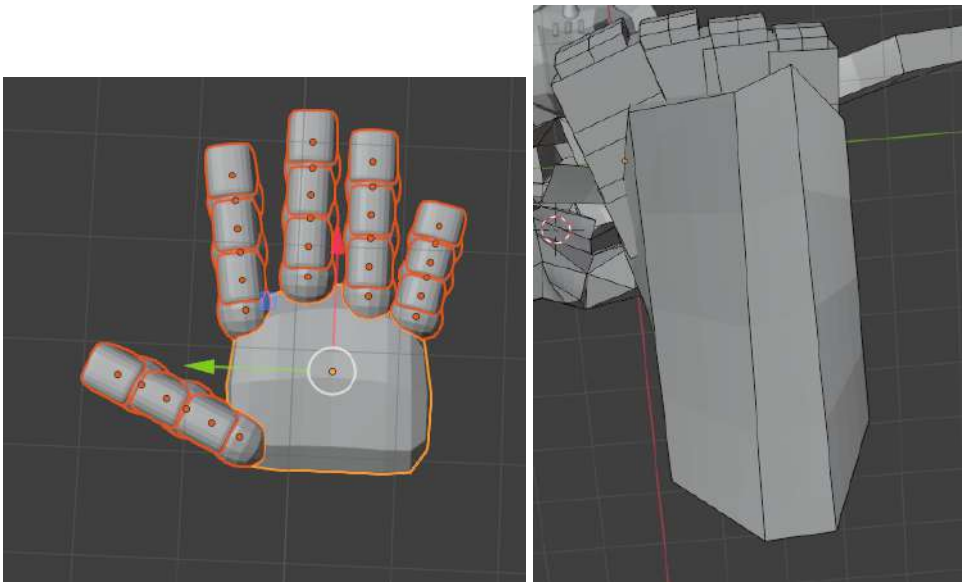
Cuerpo en fase inicial del esqueleto

Esta distribución resultó dar muchos problemas a la hora de realizar el rigging, pues al no generarse mallas como tal con el uso del skin modifier, en muchas ocasiones no se interpretan bien las distancias, o si se utiliza el autorigg del skin modifier, el mirror del cuerpo, solo se realizaba el rig a una parte del cuerpo, y está controlaba ambas partes.

Finalmente, se convirtió el cuerpo con el skin modifier a un conjunto de mallas. Se eliminó el modificador, y se arregló el cuerpo a partir de esa base que se había descrito.

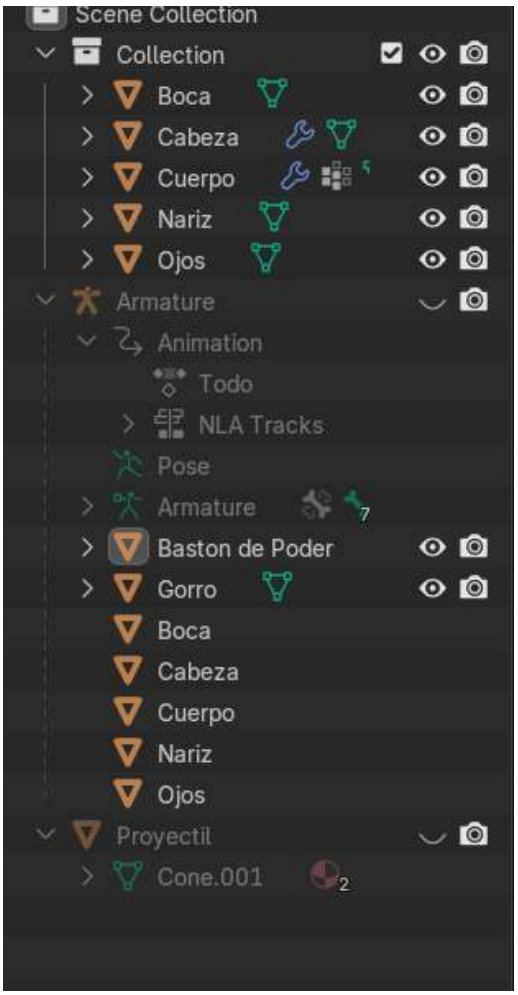


Las extremidades fueron uno de los puntos más importantes en su diseño, debido a su complejidad se utilizó un video como guía. En el video, solo se modelan las manos, y a partir de ese modelo se realizaron modificaciones para el modelado de los pies, adaptando las proporciones y ángulo de los dedos. También cabe destacar que inicialmente se utilizaba el *Subdivision Surface*, pero para que quedará más acorde al cuerpo final se eliminó.



Finalmente, el modelado del bastón se realizó utilizando como base una esfera para el cristal mágico, y un cilindro alargado para el diseño, y

después se añadieron más detalles a la unión entre estos, como una segunda esfera que medio envuelve la primera entre estos. La referencia en su diseño es el bastón de Saruman del señor de los anillos.

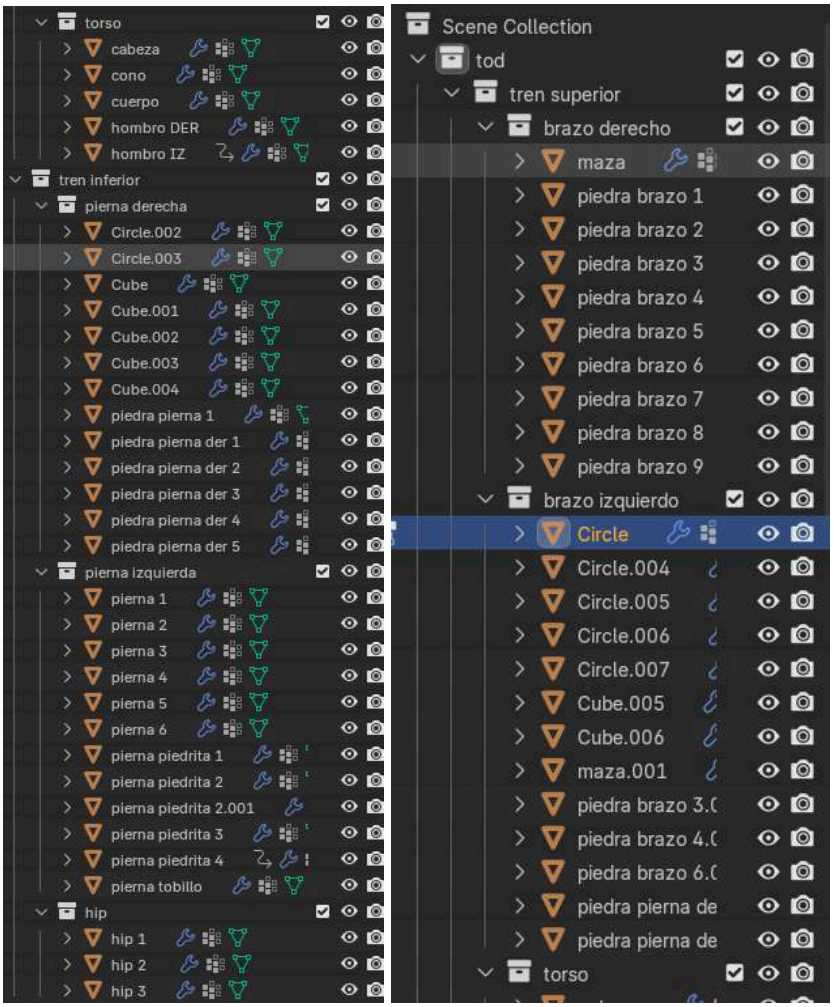


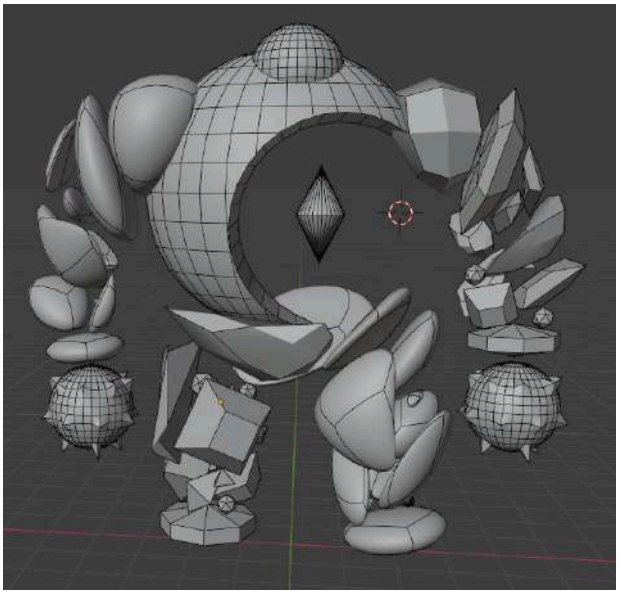
Jefe final

TOD (Tarmox Oathbound Devourer) es el jefe final del juego, y queda visible para el jugador desde el momento en que entra al coliseo. Este boss final consiste en un golem autónomo formado por acumulaciones de hielo y nieve, además de una gema que puede observarse en el centro de su pecho. El boss tiene un tamaño considerable en comparación con los otros enemigos, pero la diferencia es aún más notable si lo comparamos con Nur.

Afortunadamente, pese a su gran tamaño y a que realiza ataques cuerpo a cuerpo devastadores para la salud del protagonista, TOD es lento y carece de inteligencia. Simplemente se mueve en dirección al protagonista e intenta alcanzarlo. Esta dinámica le da una ventaja estratégica al jugador, ya que le permite mantener la distancia para reponer energías y curarse.

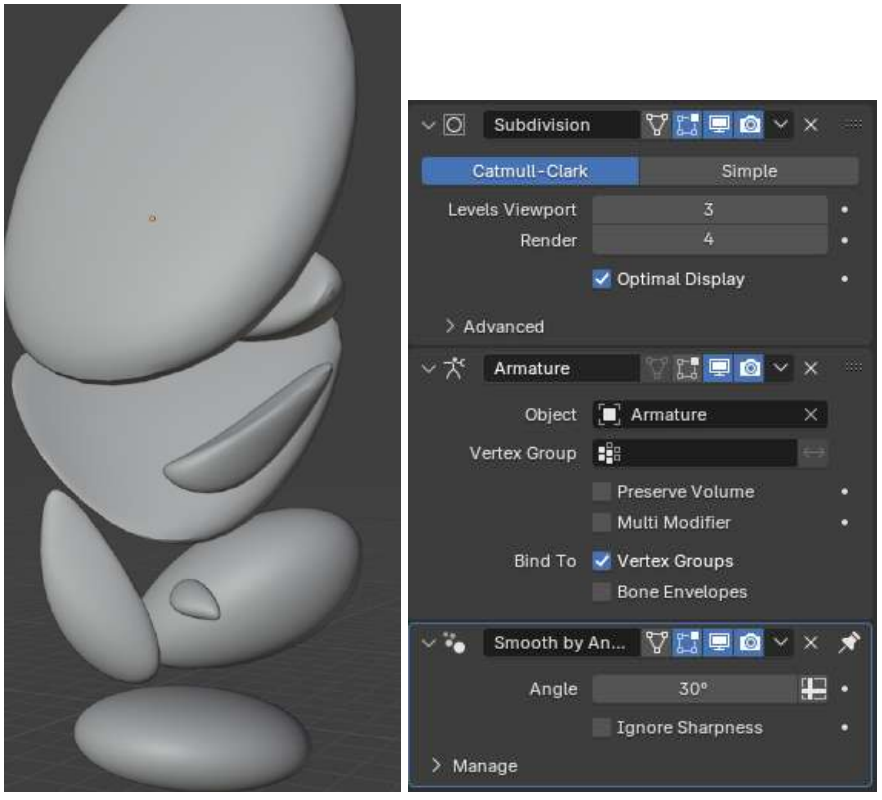
La idea de TOD surge a partir de dos conceptos: la primera pieza de inspiración es el arte de Svetlin Velinov para la carta de *Magic: The Gathering* Solitude, de la cual se ha incorporado el torso y las piezas flotantes; y el golem de piedra de Soul Eater, del cual se toma el comportamiento de TOD. Del primero se adopta la estética, y del segundo, el patrón de conducta.



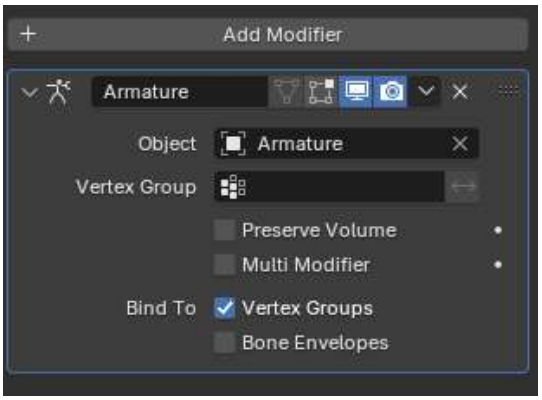


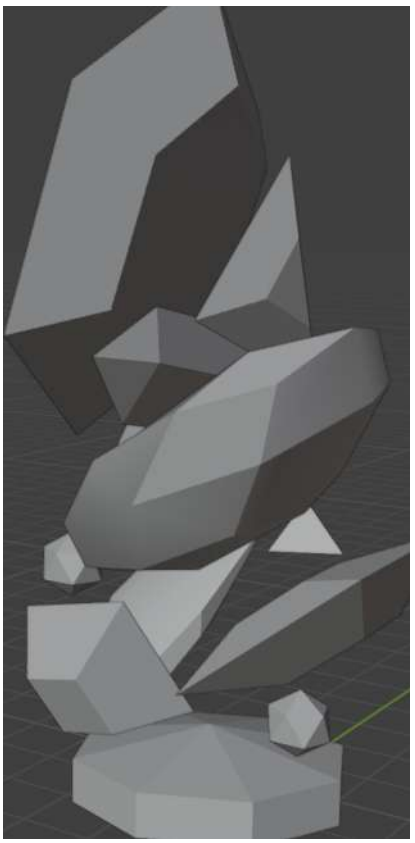
TOD es una figura compleja formada por figuras simples , lo cual lo hace algo diferente a los otros modelos de este juego.

Se ha optado por crear una especie de balance entre las texturas y formas de TOD, como se puede observar tiene un brazo de hielo y el otro de nieve estos comparten material con su extremidad inferior opuesta. Decidimos que el torso y la cabeza fueran de nieve pero la gema fuera de hielo.



Para modelar las partes formadas por nieve se aplicaba un subdivisión a la pieza después de darle la forma deseada además de un smooth para que diera esta sensación de forma erosionada.





Mientras que su contraparte simplemente generaba la forma deseada a partir de un objeto básico.

La idea del modelo siempre fue que diera sensación de simetría pero no fuera una simetría perfecta , por ello no he podido hacer un mirror como se haría normalmente en un personaje humanoide del estilo y en vez de eso he hecho las dos partes individualmente para que tuvieran algunos cambios notorios como por ejemplo el posicionamiento de las piezas o las formas de las mismas.

Texturizado e Iluminación

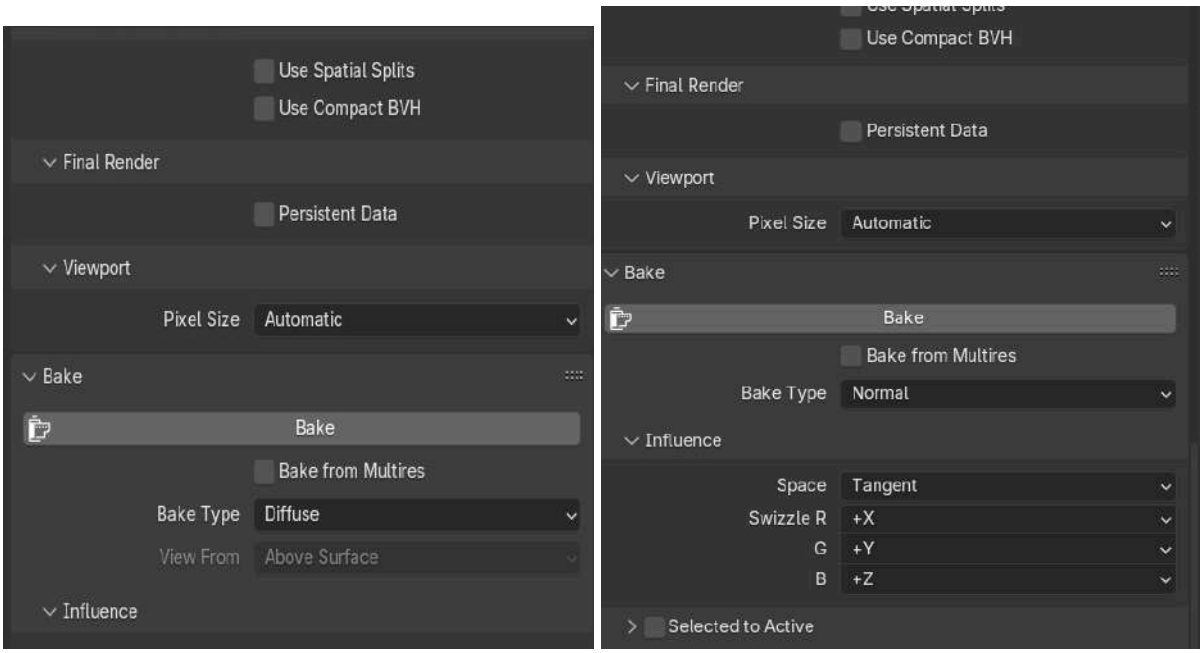
En cuanto a las texturas de los modelos, la mayoría las hemos extraído de BlenderKit[8]. Es un complemento para Blender que permite acceder a una biblioteca en línea de modelos 3D, materiales y entornos HDRI directamente desde la interfaz de Blender.

Las texturas usadas para Nur, el dragón, son su piel, los ojos, el interior de la boca, la lengua, y los dientes.

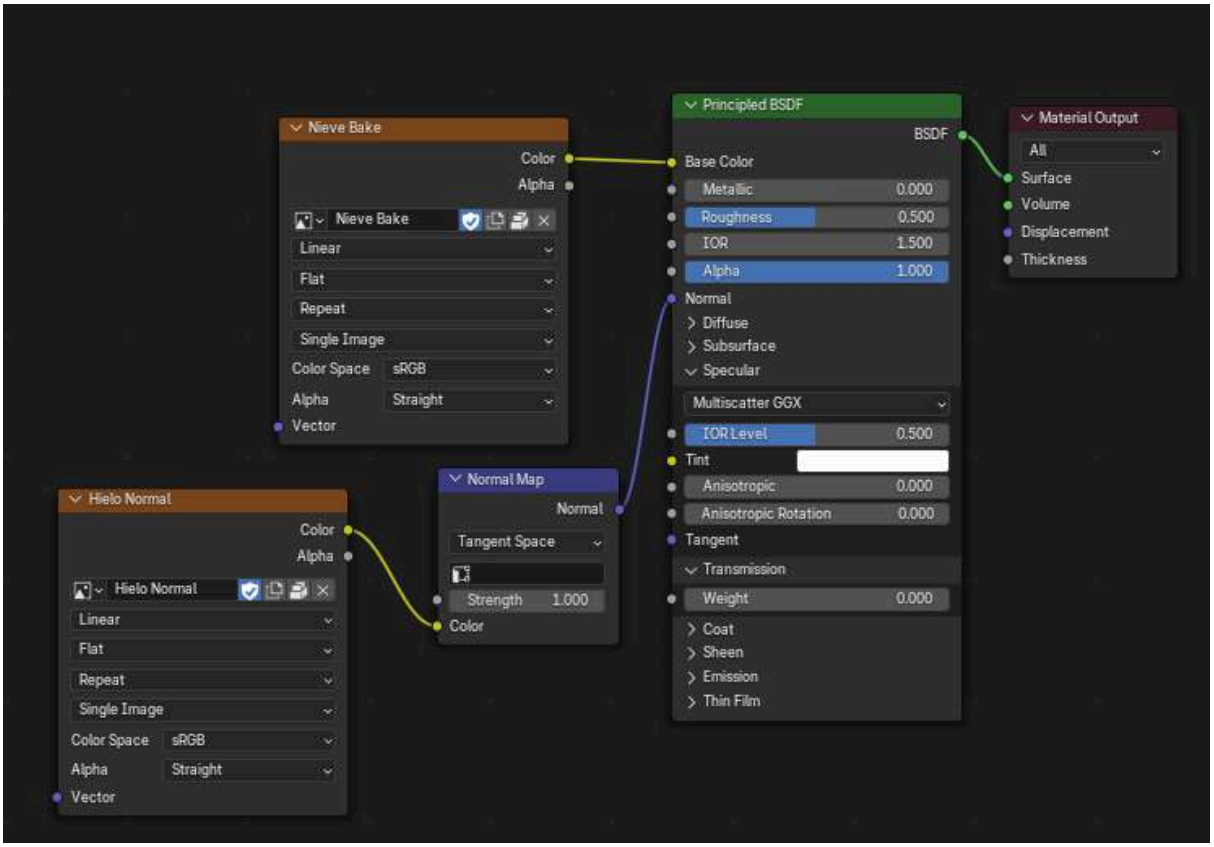
Lo mismo aplica para las texturas usadas en los enemigos, la nieve, el hielo, el cuerpo del esqueleto y el sombrero de mago.

Por desgracia, las texturas de nieve, hielo y hueso, eran texturas generadas proceduralmente. Esto requirió realizar un proceso de bake para que se pudieran exportar correctamente como imágenes (el punto de exportación a Unity se tratará más adelante).

Para este proceso se realizó consultando un tutorial[9]. Se comienza alineando las texturas que tienen los materiales aplicados en un UV mapping. A partir de aquí, se generan una imagen básica, y una imagen de sombras mediante la utilidad de bake de blender, la cual analiza las texturas del objeto y mediante diversos samples de estas crea imágenes. Para los colores de las texturas utilizamos el formato de *difuse*, mientras que para las sombras y definiciones, el modelo de *normal*.



Una vez generadas ambas imágenes, se realizaba un shader de estas, introduciendolas ambas como inputs, y extrayendo una textura como output.



Estas texturas bakeadas se realizaron para su incorporación en Unity, pero se ha decidido nombrar en la memoria. En la parte entregada de blender se compartieron las texturas del BlenderKit originales.

La iluminación fue una parte fundamental para transmitir correctamente la atmósfera nocturna, invernal y de combate que se buscaba para la escena del coliseo. Desde el principio, se decidió trabajar con una luz ambiente tenue, ya que la acción transcurre de noche. Esta luz general fue configurada con una intensidad baja y con tonos azulados, para reforzar la sensación de frío y oscuridad, propios del bioma nevado.

Como fuente principal de iluminación se utilizó el eclipse modelado con dos esferas, que además de ser un elemento estético importante, sirvió para justificar una luz direccional intensa que proyecta sombras dramáticas sobre el escenario. Esta luz genera un foco claro desde una dirección específica, marcando siluetas fuertes en los muros, el suelo y los personajes, lo que añade mucha fuerza visual a la escena. No obstante, este tipo de iluminación también provoca zonas con sombras demasiado duras y oscuras, lo que dificulta la visibilidad en algunas partes del coliseo.

Para contrarrestar ese problema sin romper con la lógica visual del eclipse, se colocó una segunda luz direccional opuesta, con menor intensidad, simulando una leve reflexión ambiental o luz residual proveniente del cielo. Esta luz secundaria ayudó a suavizar las sombras más oscuras, permitiendo que los elementos del escenario fueran visibles sin perder la sensación de oscuridad general. Fue un equilibrio delicado, pero necesario para que la jugabilidad y la estética se mantuvieran bien integradas.

Además de estas luces principales, se añadieron puntos de luz en cada una de las antorchas distribuidas alrededor del coliseo. Cada antorcha fue configurada con una luz cálida y de corto alcance, que no solo refuerza su función decorativa, sino que también aporta focos de iluminación local que ayudan a destacar detalles de la arquitectura y los elementos cercanos, como muros, armas o personajes. Estas pequeñas luces también contribuyen a

generar contraste de temperatura entre los tonos fríos del entorno y el color cálido del fuego.

Finalmente, se incorporó una luz puntual en el núcleo del jefe final, situada dentro de su cuerpo o zona central. Esta luz tiene una doble función: destacar visualmente al jefe dentro del combate y darle una presencia amenazante y sobrenatural.

En conjunto, todas estas luces permiten mantener una escena bien equilibrada, en la que se puede leer visualmente lo que ocurre, sin perder el

impacto visual de una noche helada, iluminada por un eclipse parcial y unas pocas fuentes de luz artificial controladas.

Rigging, Weight Paint y Animación

En cuanto a las animaciones, se ha utilizado la técnica basada en *keyframes*. Cada pose base conforma un *keyframe*, y es Blender el que se encarga de "rellenar" entre *frames*, deformando el modelo para hacer una transición suave. Así, conseguimos una animación realista.

El escenario (Coliseo)

La animación de la escena se realizó mediante el uso de keyframes aplicados a la cámara principal. Se definieron varios puntos clave en la línea de tiempo, y Blender se encargó de interpolar automáticamente el movimiento entre ellos, generando un desplazamiento fluido y controlado.

Al comenzar la animación, la cámara se sitúa en el punto de vista del dragón, avanzando en línea recta como si este estuviera caminando hacia el centro de la arena.

A continuación, la cámara amplía el rango de visión y comienza un movimiento de giro suave hacia los lados, revelando progresivamente la estructura del coliseo. Durante esta secuencia, las antorchas se encienden mediante la activación de su visibilidad en el momento exacto, generando un efecto de iluminación progresiva que refuerza la ambientación.

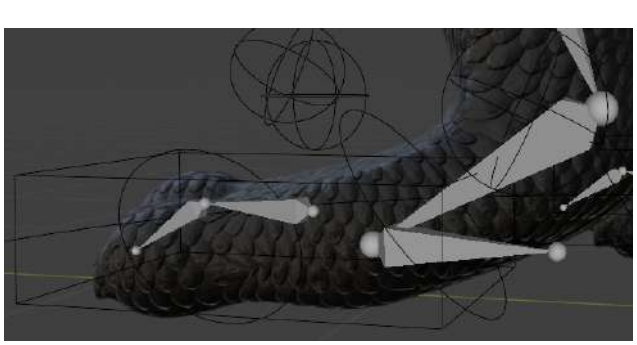
En el centro del movimiento, se muestra al dragón ejecutando su animación de idle, permaneciendo en su sitio con movimientos sutiles para simular vida.

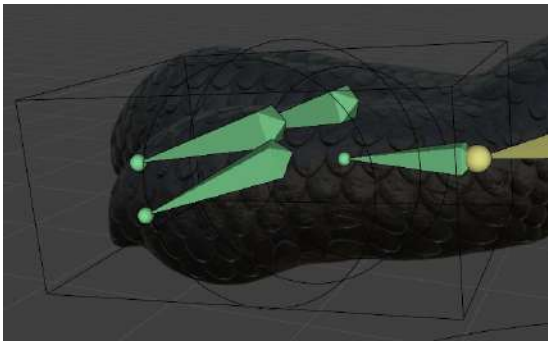
Finalmente, la cámara asciende ligeramente y termina enfocando hacia el cielo, centrando la atención en el eclipse, que actúa como cierre visual y narrativo de la escena.

El protagonista (Nur, el dragón)

Para empezar el proceso de animación del dragón, primero se hizo el *rigging*[10]. Primero se creó el esqueleto básico, con los huesos básicos que conformaban la forma del modelo.

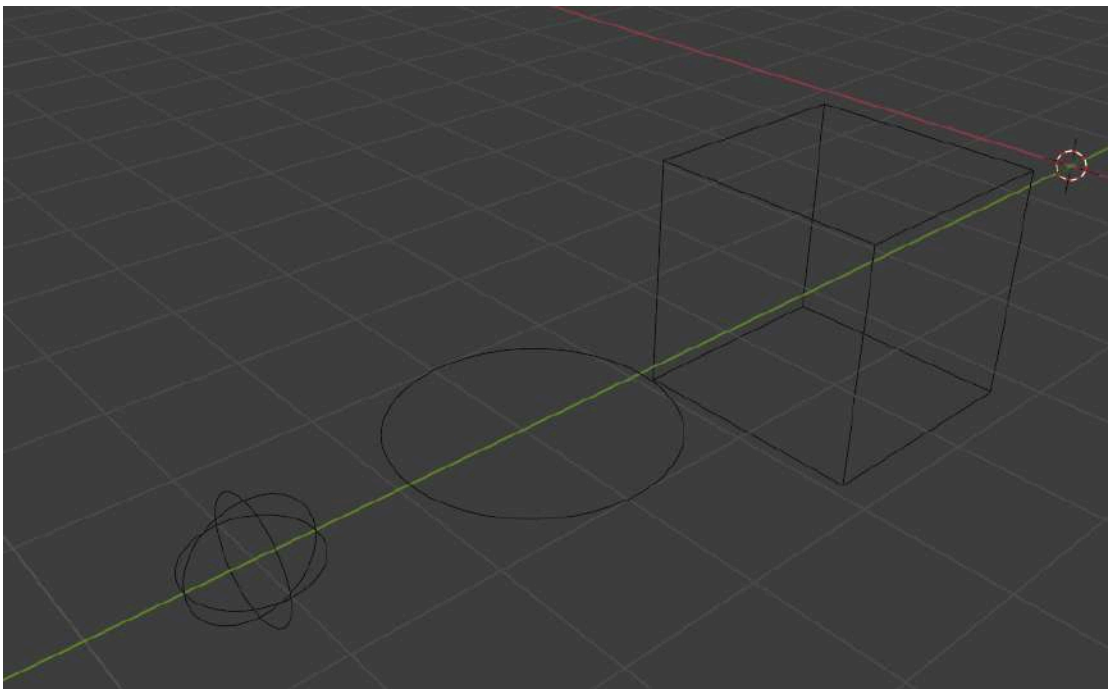
Se añadieron *kinematics* para los brazos y piernas, además de un controlador para los pies y manos de manera independiente. Está configurado de manera que si se mueve uno de ellos, afecta al brazo o pierna, efectuando sus movimientos como lo haría una extremidad real. Los dedos también se han configurado de manera que el dedo se mueve en su totalidad, flexionando como lo haría uno real. Todo este proceso se realizó en una sola mitad del modelo, aplicando *Mirror* más tarde.





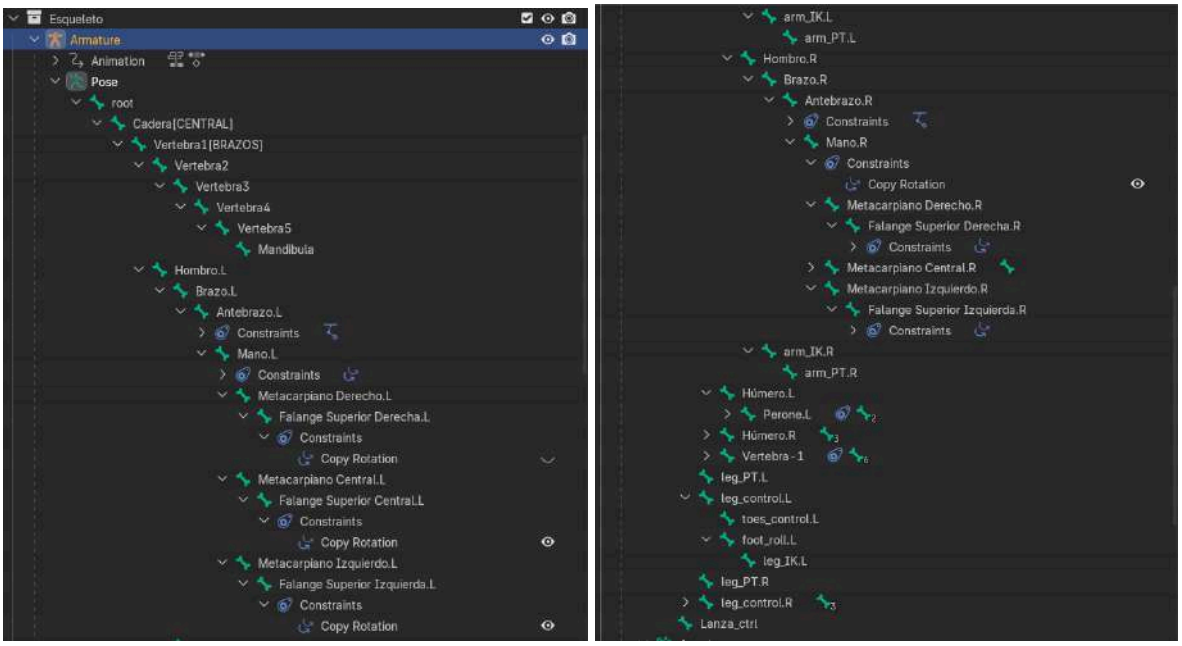
Kinematics de los pies y manos

Finalmente, todos los huesos se convirtieron en controladores más sofisticados para facilitar el proceso de animación. Así, con formas vacías, aplicamos a cada hueso una de ellas, personalizando cada controlador para tener una forma más intuitiva y poder distinguir bien qué parte del modelo se mueve con cada uno.

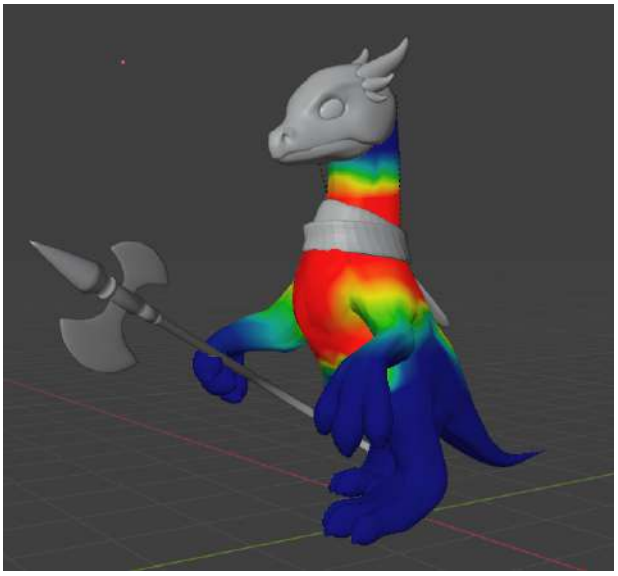


Formas vacías utilizadas para los controladores

La jerarquía del esqueleto final queda de la siguiente manera.



El siguiente paso antes de animar fue el *Weight Paint*[11] para asegurarnos de que los huesos y controladores creados realmente mueven lo que pretendemos que muevan.

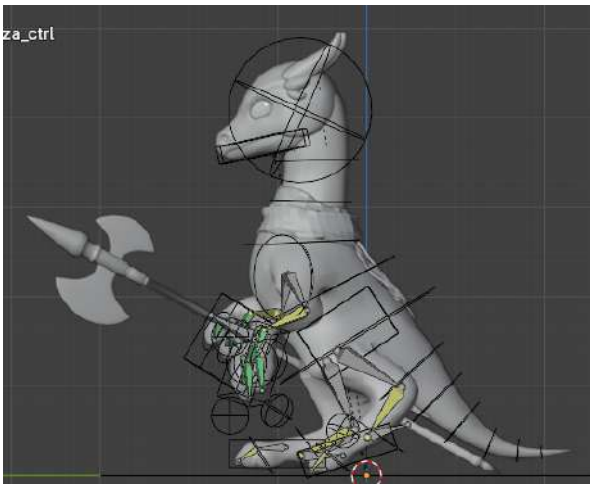


Parte del Weight Paint de Nur

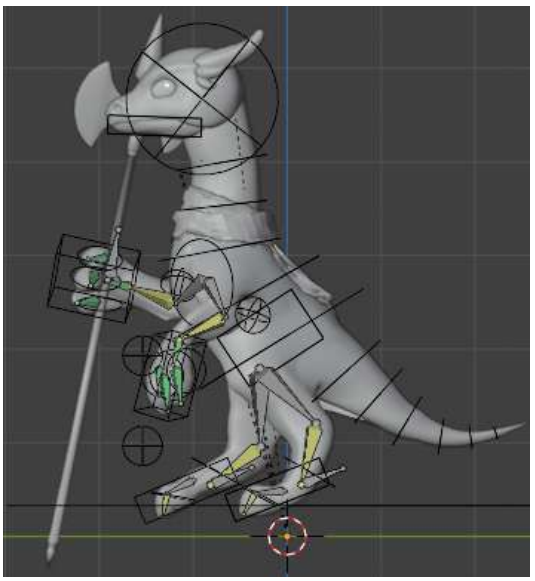
Para cada hueso (o controlador), se asigna un peso, de manera que cuanto más rojo, más influencia tiene el hueso, y cuanto más azul, menos.

Una vez determinados los pesos, podemos pasar a animar. Se han creado para Nur seis animaciones, descritas a continuación. Se pueden consultar todas las animaciones de Nur en el siguiente enlace.

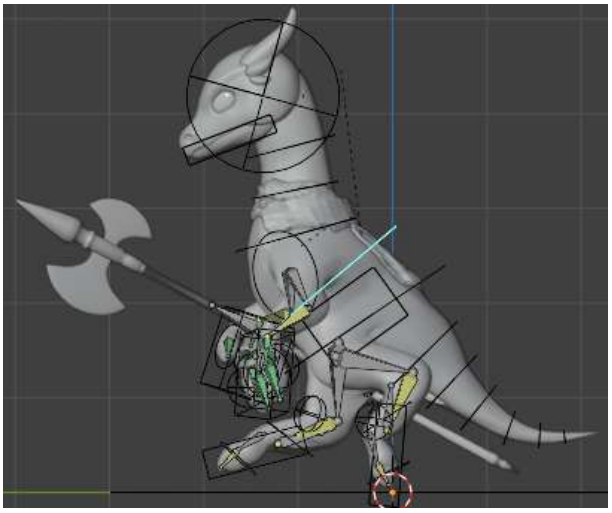
Caminando: Una animación en bucle, donde el dragón camina infinitamente sobre un plano. Utilizada en el videojuego para la acción de andar. Con el movimiento, se mueven además las otras partes del cuerpo para una simulación más realista. Para realizar esta animación, se siguió la guía de un tutorial[12].



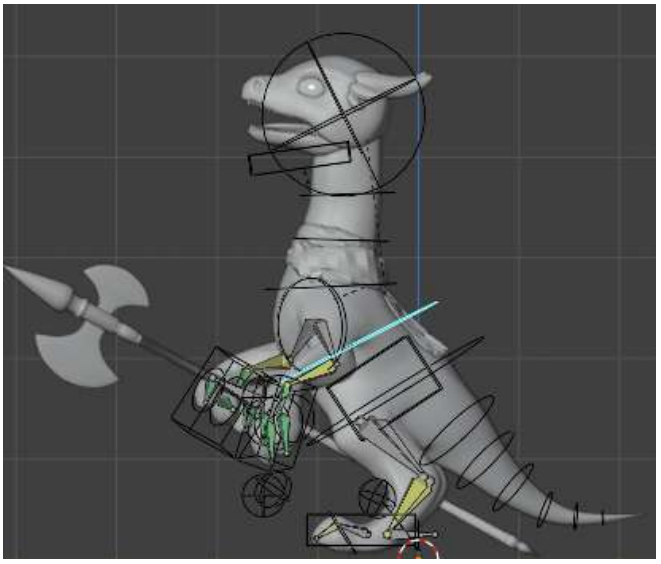
Ataque Básico: Animación del ataque básico a melee. Nur hace un pequeño salto y hace un ataque con la lanza hacia abajo. El cuerpo se mueve con el movimiento, también queriendo simular un poco más de realismo.



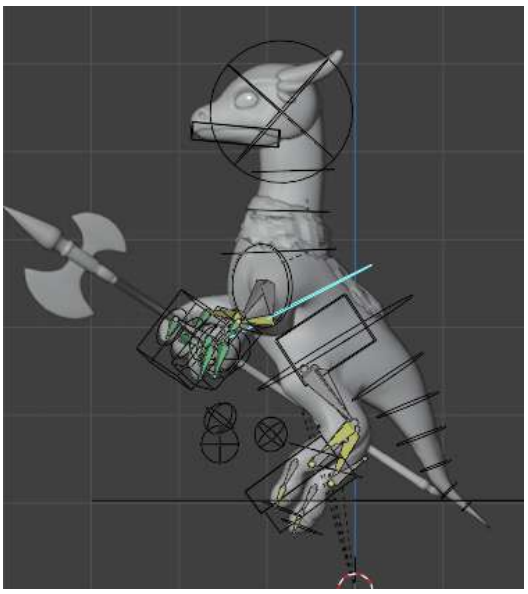
Correr: Animación de correr. Se tomó como base la de caminar, siguiendo los mismos frames base. Esta dura bastante menos, ya que los frames necesitan estar más cerca, y se necesita más repetición para poder simular una acción más rápida.



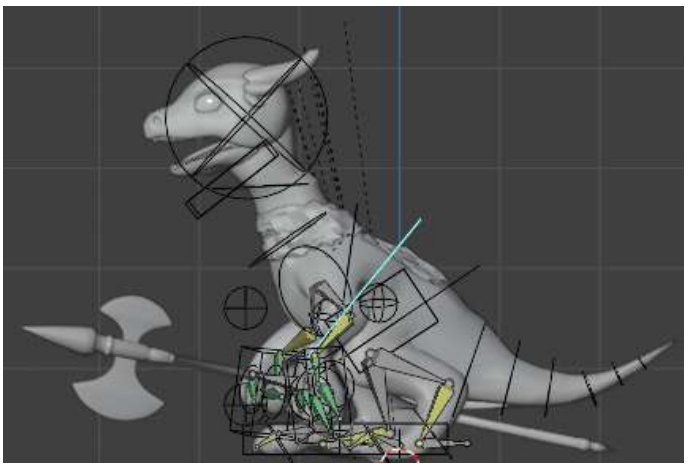
Idle: Una animación que salta cuando Nur está inactivo durante un tiempo, sin input del jugador. Levanta la cabeza y ruge, sacudiendo luego la cabeza y la cola.

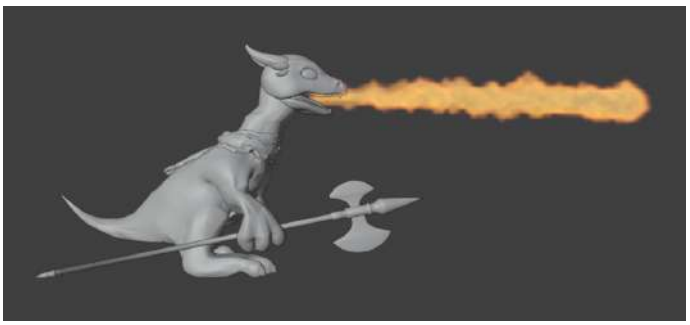


Salto: Animación donde Nur salta. Prepara la estructura corporal, haciendo un pequeño impulso, antes de pegar el salto. Una vez en el aire, comienza el descenso, haciendo también la simulación de impacto con el suelo, y luego volver a su posición inicial.



Ataque Especial: Nur ruge, escupiendo fuego por la boca y agachándose un poco. El fuego, aunque lo intentamos hacer en Blender, no fue posible debido a la dificultad adicional que supuso intentar incluirla en Unity. Así, el propio fuego solo se muestra en Unity. Se ha guardado la animación con fuego en Blender, por tanto se han entregado ambos modelos (con fuego, y sin).





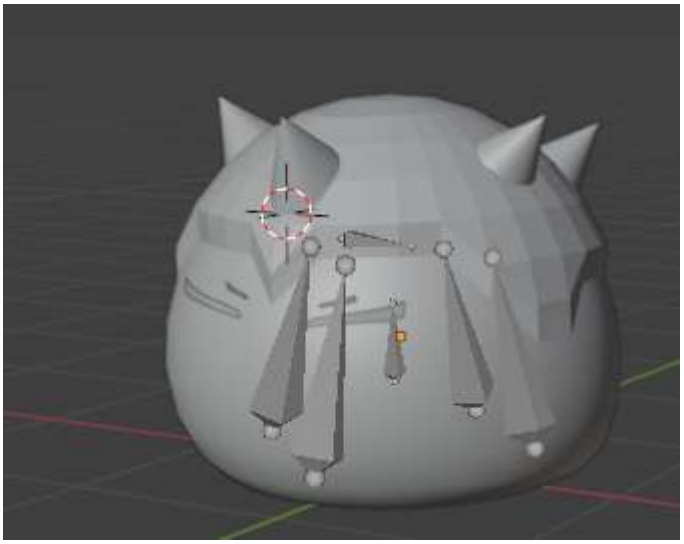
A lo largo de todas las animaciones, la bufanda a la que se aplicaron físicas de téxtil, se le hizo un bake. Así, para cada animación, la bufanda se mueve con el modelo, creando más realismo.

Para facilitar la transición a Unity, todas las animaciones y el bake de la bufanda están en su versión final en la animación **Todas**, en el modelo de Blender.

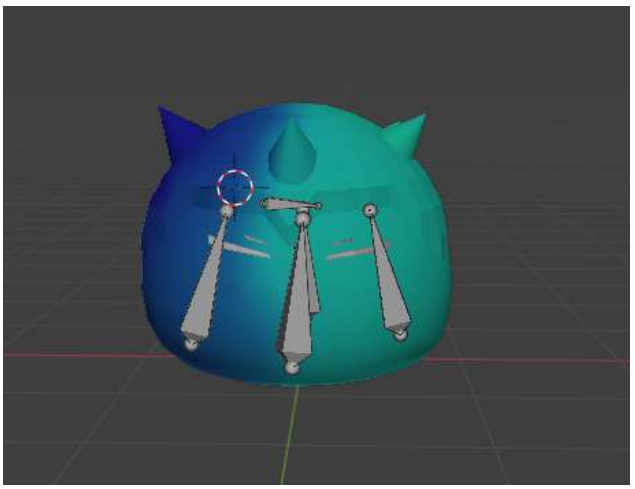
Los enemigos

Slimes

El esqueleto del slime está conformado por 6 huesos. El hueso principal, el cual se encarga de los movimientos de estos, 4 huesos laterales (uno para cada X e Y), y el hueso del casco.



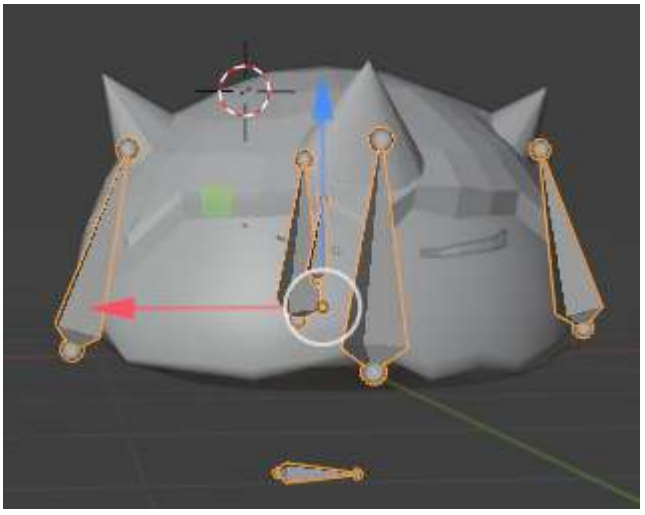
Diversas distribuciones de huesos y pesos se tuvieron que probar, pero finalmente la distribución descrita anteriormente juntamente con un “weight paint”, que le da a cada hueso influencia sobre una mitad del cuerpo completo en función del lado en el que se encuentra, resultó tal y como lo queríamos para su movimiento.

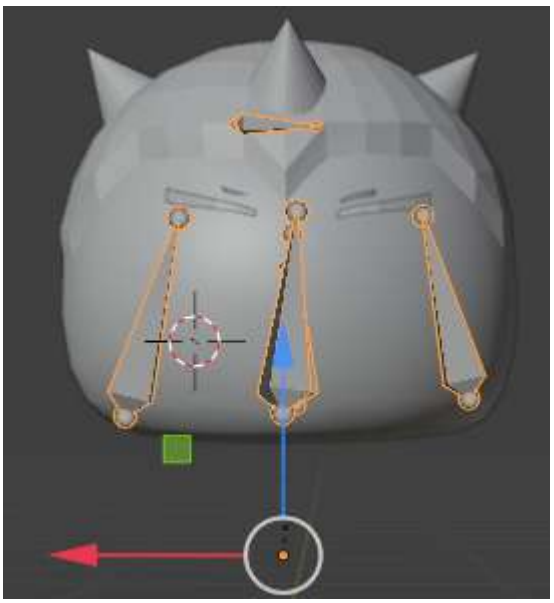


El objetivo era que sus movimientos mantuvieran la esencia del slime y parecieran “líquidos” o no sólidos completamente. De este método, al empujar de sus huesos, gran parte del slime se estira o contrae, por lo que se asemeja muy bien este efecto.

Con respecto a sus animaciones[13], las principales son las de saltar, atacar y morir. El slime no posee una animación dedicada a cuando se queda quieto (idle).

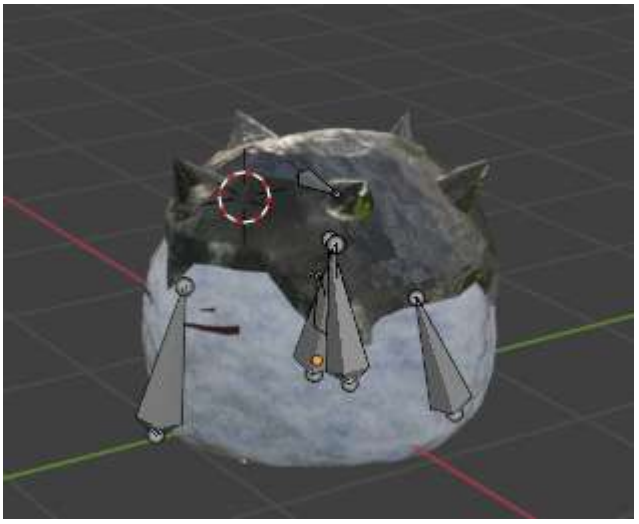
Salto: el cuerpo se comprime y luego se expande hacia arriba, simulando un rebote elástico.



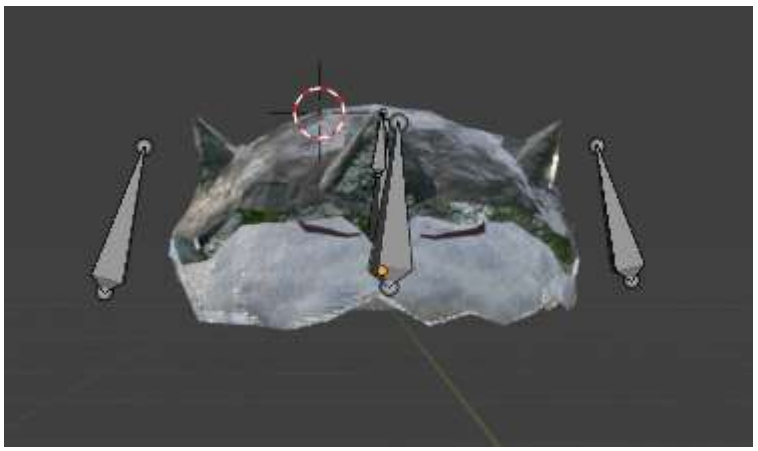


Ataque: el slime proyecta su cabeza con pinchos hacia adelante en un movimiento brusco. Se simula que en su movimiento se comprimen los lados de su cuerpo y se alarga la cabeza.

Nota: Nos habría

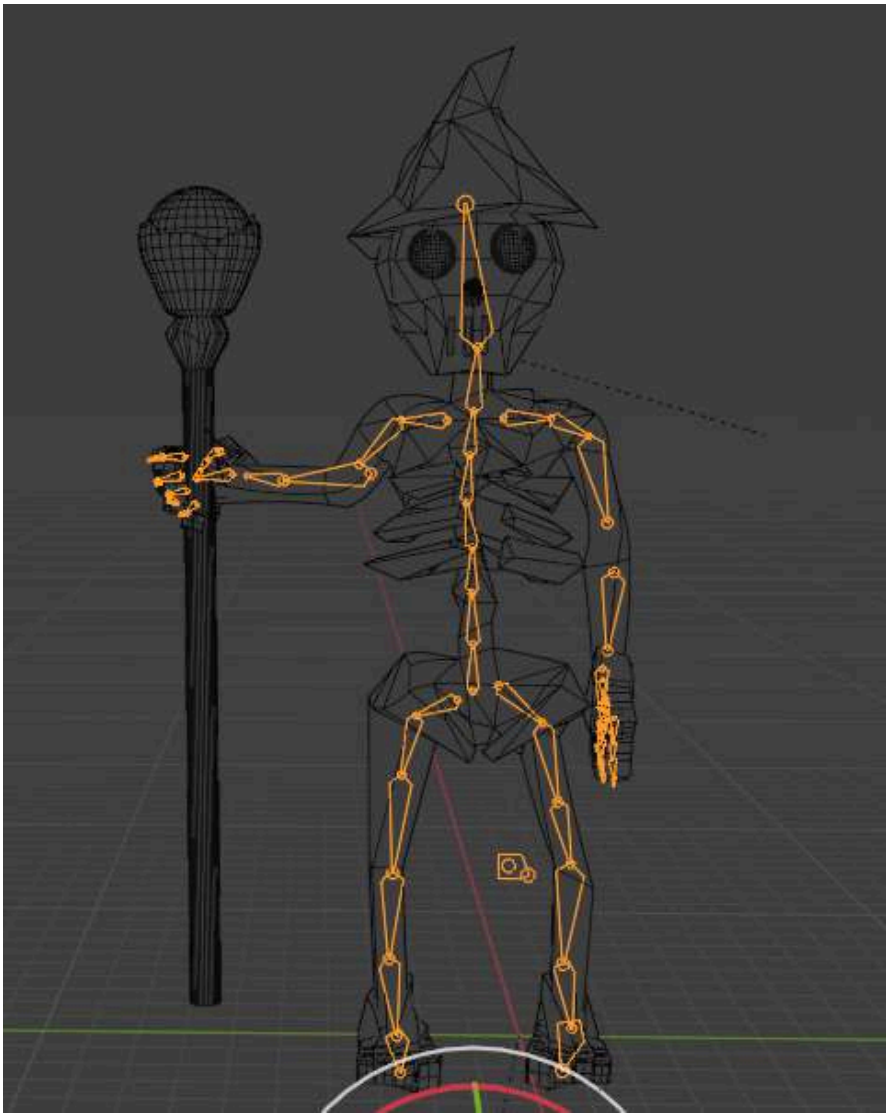


Muerte: el personaje "se funde" o colapsa sobre sí mismo. Para esta animación se usaron keyframes que reducen el volumen general del cuerpo mientras los accesorios se deforman.



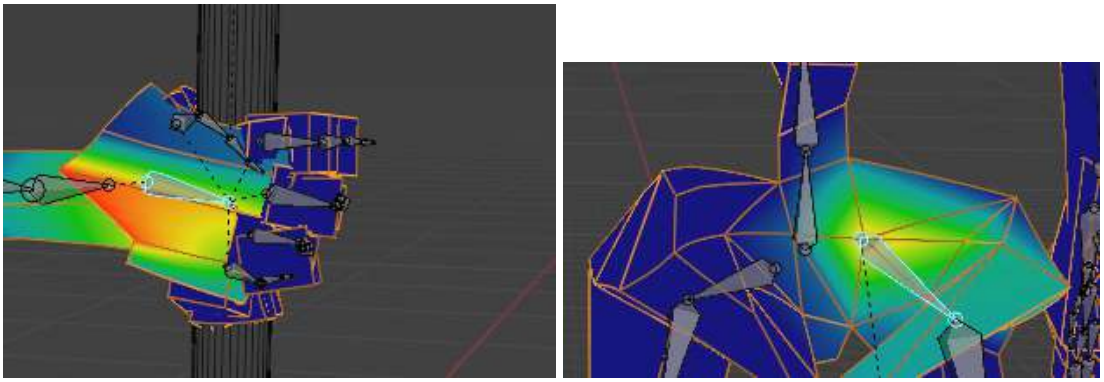
Mago esqueleto

El mago esqueleto posee una estructura ósea mucho más definida y segmentada que el slime, lo que requirió una aproximación distinta tanto para el rigging como para la creación de las animaciones. Su rig se construyó siguiendo la lógica anatómica de un esqueleto estilizado: incluye huesos principales en la columna, pelvis, cuello, brazos, piernas y cabeza, además de huesos individuales en los dedos, lo cual permitió un control detallado de gestos y posturas.



Tanto el bastón como la cabeza y los complementos, se vincularon a los huesos de la mano y del cuello respectivamente.

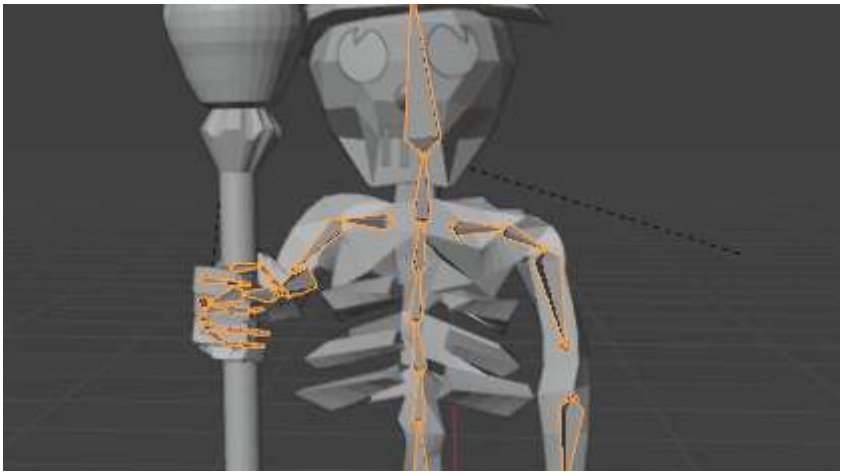
Para la generación del esqueleto se utilizó un rig manual, adaptado a la morfología modular del personaje. Se empleó la opción de pintura automática de pesos (*automatic weight paint*) para la asignación inicial de influencias, pero se detectaron varios problemas, especialmente en las articulaciones de cadera y hombros, así como en los dedos[14], donde los pesos se solapaban o deformaban mal la malla. Por ello, se realizaron ajustes manuales zona por zona, asegurando que cada hueso afectara únicamente a las partes deseadas del modelo. Esta fase fue especialmente importante para lograr una animación fluida en movimientos como lanzar hechizos o caminar.



El rig final resultó muy satisfactorio para lo que se buscaba: permitir movimientos relativamente complejos sin llegar a una estructura excesivamente técnica o pesada.

El mago esqueleto cuenta con un conjunto de cuatro animaciones principales: *idle*, *caminar*, *atacar* y *morir*, todas ellas diseñadas para enfatizar su rol como enemigo a distancia con personalidad evasiva.

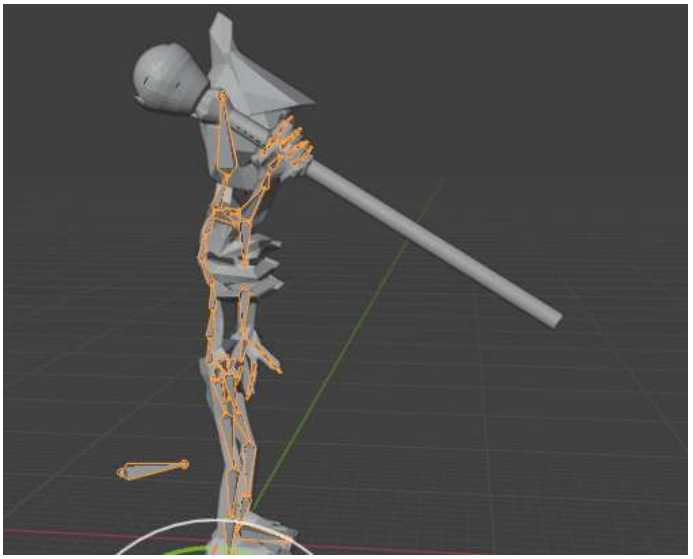
Idle: El personaje se mantiene en su mayoría quiero, pero manteniendo ligeros movimientos en sus hombros y columna, así como pequeños gestos en el báculo y la cabeza.

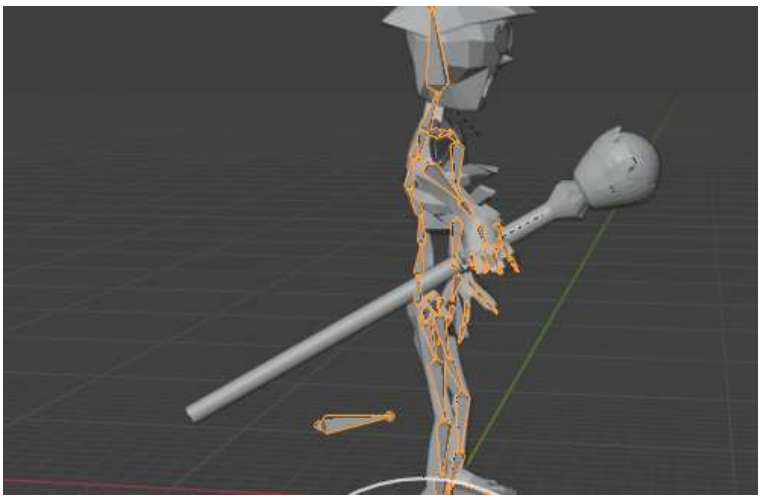


Caminar: El personaje avanza con pasos definidos, el torso levemente inclinado y los brazos balanceándose con un ritmo irregular. Se buscó transmitir cierta rigidez corporal sin que resultara completamente artificial.

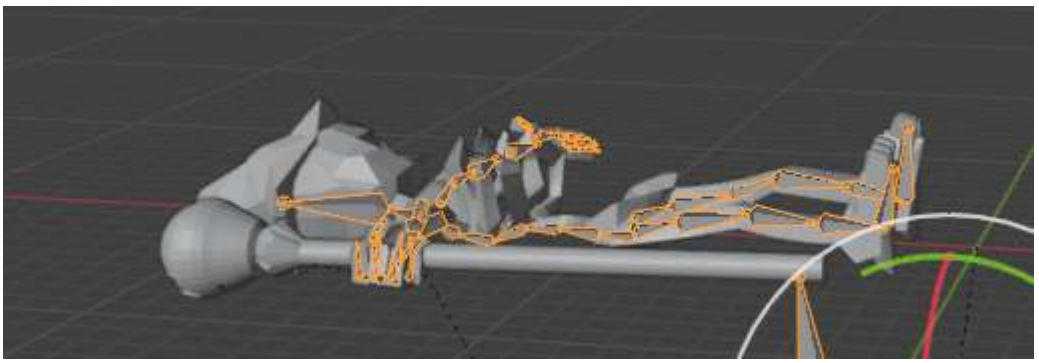
Ataque: La animación más expresiva del personaje. Al atacar, el mago levanta su báculo con un gesto firme y realiza un giro parcial del torso. El

brazo y la mano acompañan el gesto con una leve sacudida. El objetivo fue representar un ataque ritualizado y predecible, permitiendo al jugador anticiparse a él.



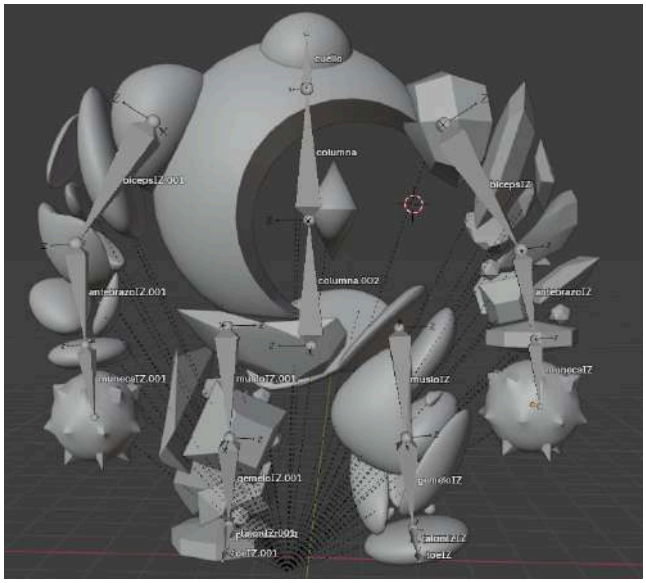


Muerte: El esqueleto colapsa y cae al suelo con su cuerpo totalmente estático. Se trató de transmitir que su cuerpo deja de estar animado mágicamente. Mantiene cierto toque de falsedad como de "dibujos animados", porqué nos parecía gracioso.



A diferencia del slime, cuya animación buscaba transmitir elasticidad, en el mago esqueleto se prioriza la gestualidad y la intención. Para la exportación en Unity de estas animaciones, se fusionaron todas en una sola.

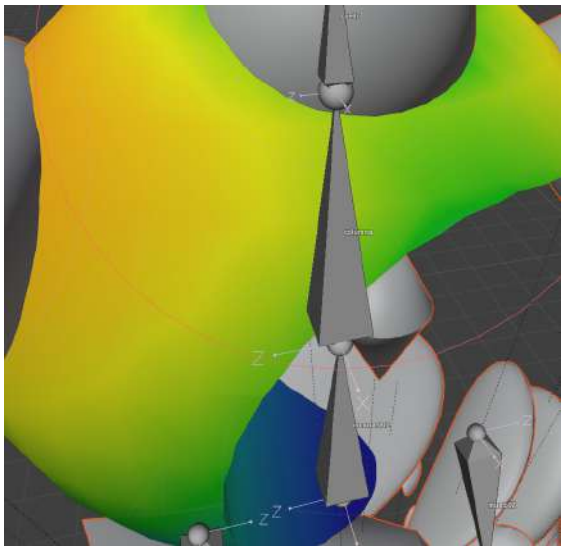
Jefe final



El rigging de TOD es simétrico a diferencia de sus extremidades . Al tratarse de un golem quería dar la impresión de ser una figura más bien tosca y con dificultad de movimiento , entonces pese a consistir de decenas de partes flotantes le he dado un número limitado de articulaciones para que sus movimientos resulten robóticos.

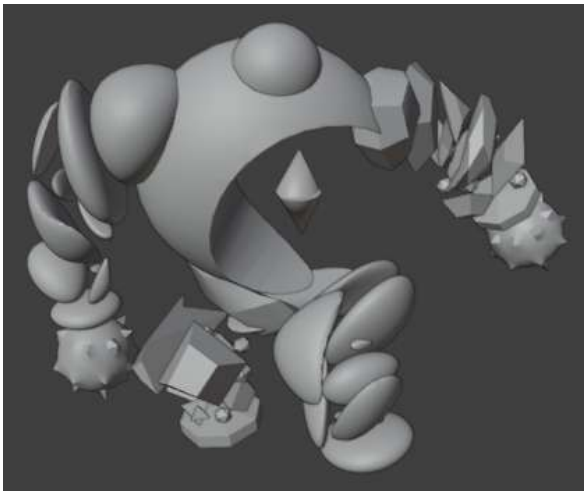
Pese a que existen herramientas de rigging automáticas dado a lo específico que era el objetivo con este personaje me vi obligado a hacerlo a mano.

La mayor particularidad que había que tener en cuenta es que al tratarse de un golem de objetos sólidos estos no deben deformarse al moverse, lo cual probó ser un reto finalmente, el resultado es un modelo que es perfectamente animable pero cuyas piezas se deforman lo menos posible.

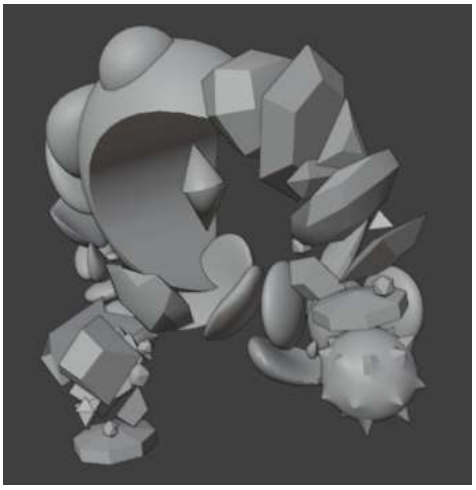


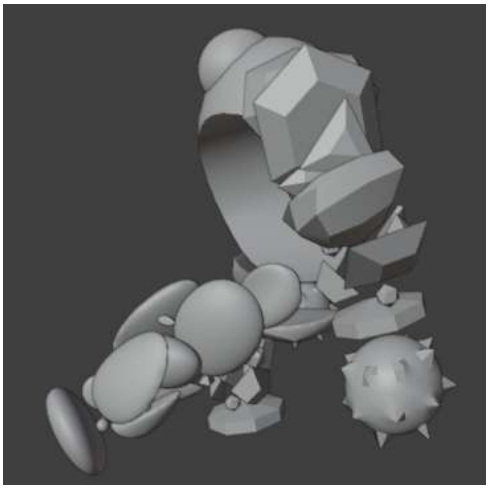
En el golem el weight paint es un poco diferente en el que vayas a encontrar en otros modelos puesto que está hecho para que las piezas reaccionen lo menos posible al movimiento , con excepción de piezas como los hombros o torso como podemos ver en esta imagen , las cuales reaccionan a movimientos de la columna y tórax respectivamente. TOD cuenta con seis animaciones únicas y un ciclo de idle. Las animaciones en cuestión son :

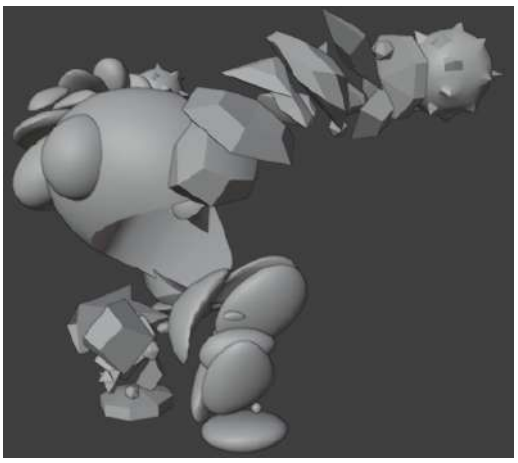
attack 1 : ataque básico de TOD , hace un gesto con el brazo derecho hacia atrás torsionando el cuerpo para acumular energía, tras ello golpea con la maza.



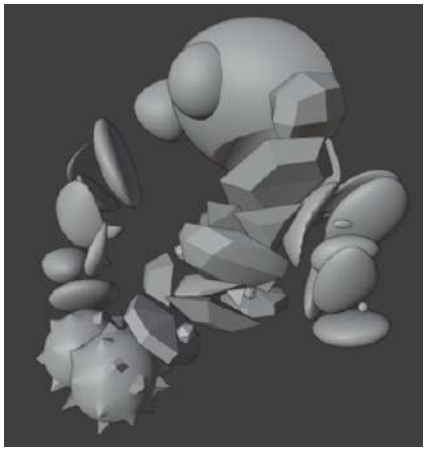
attack 2: parada baja , estirando la pata aprovechando el espacio vacío entre las partes de la pierna al final de la animación.



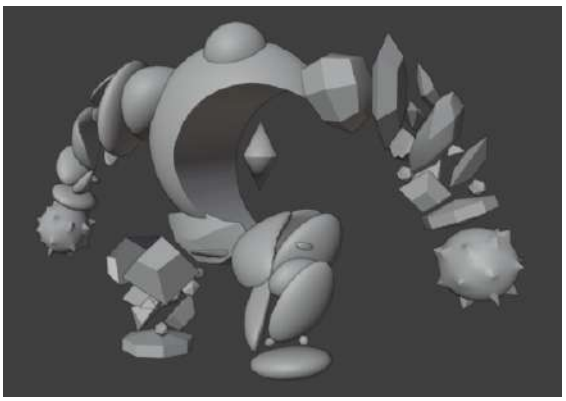


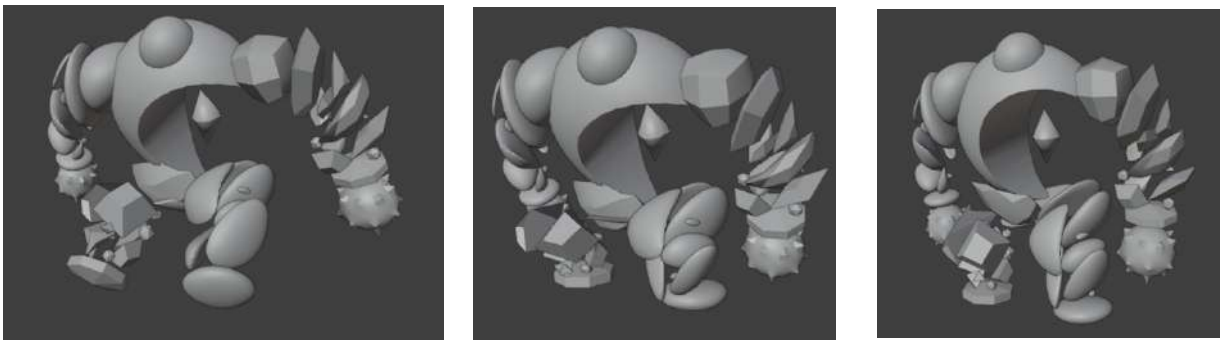


attack 3 : Junta los dos brazos detrás de la espalda , acumula energía y lanza sus brazos dando una palmada con las dos mazas que usa de manos.

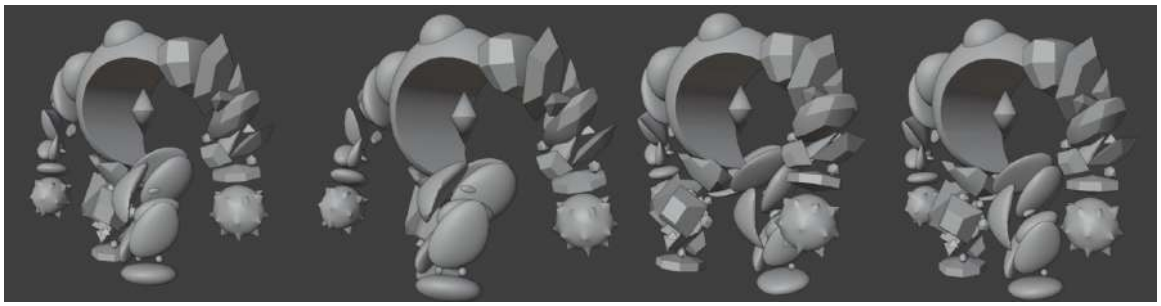


levantarse: Esta es la animación que usamos para hacer que TOD entre a la arena cuando comienza su pelea , consiste en el levantándose de donde está sentado ayudándose de las manos para despegarse de su asiento e impulsarse para llegar a la arena.

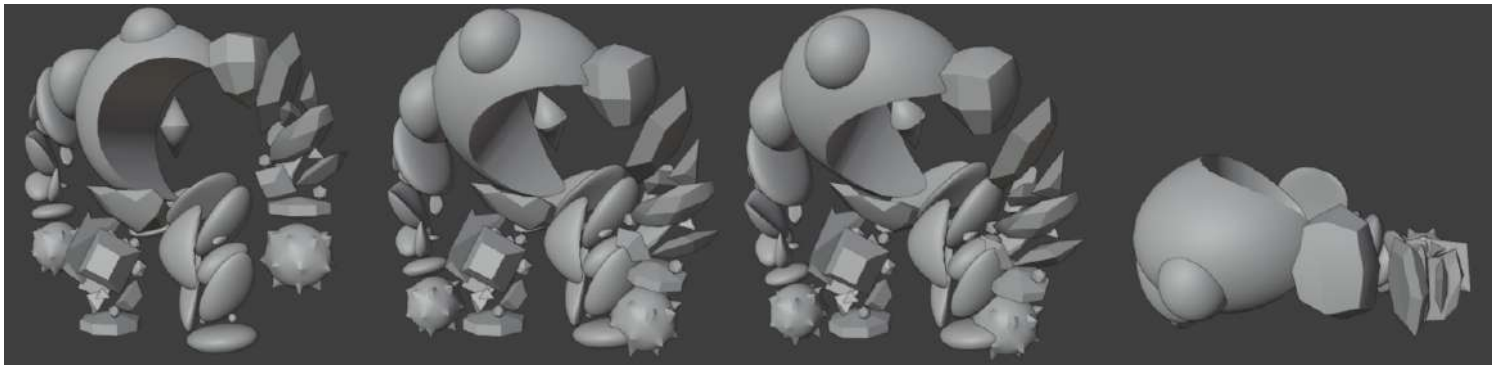




walk : Animación de caminar. Al ser un golem humanoide la animación se ha hecho similar a la de los otros modelos.



death: Animación de muerte , se ha hecho pensando que se le rompió la fuente de energía. Por lo que se desploma al quedarse sin ella.



Simulación y Efectos Especiales

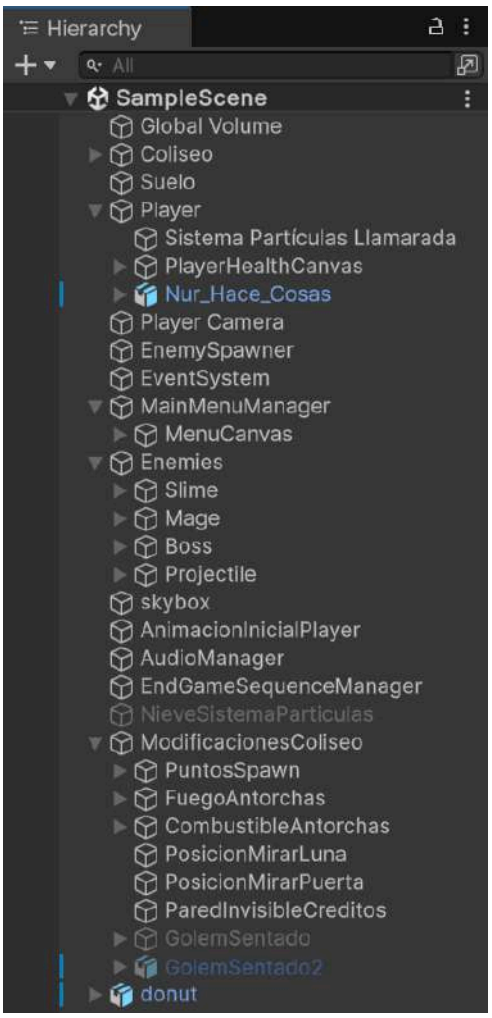
Para simular el fuego de las antorchas, inicialmente se intentó utilizar sistemas de partículas con emisión de llamas, pero esto generaba un consumo excesivo de recursos, ralentizando la escena y provocando tiempos de carga muy largos. Por esa razón, se optó por una solución más ligera: se colocó un video en bucle con animación de fuego en la parte superior de cada antorcha, proyectado sobre un plano con transparencia. Esta alternativa permite mantener el efecto visual del fuego de forma convincente sin comprometer el rendimiento general del proyecto.

En cuanto a la nieve caída, se utilizó un plano invisible situado en la parte alta del coliseo, configurado como emisor de partículas. En lugar de usar partículas con texturas complejas o físicas avanzadas, se generaron esferas ligeramente deformadas para simular copos de nieve. Estas caen de forma constante sobre la escena, manteniendo la coherencia con el bioma nevado. El uso de geometría simple en lugar de partículas detalladas también ayudó a optimizar el rendimiento, permitiendo una nevada continúa sin afectar significativamente la fluidez del entorno.

Implementación en Unity

Estructura General del Proyecto

La implementación en Unity se desarrolló como una escena única que integra todos los elementos modelados en Blender. El proyecto se planteó con las ideas de modularidad y organización para poder realizar un mantenimiento sencillo, pero al final debido a la inexperiencia que tenemos en Unity el proyecto terminó sin ser tan limpio como nos habría gustado.



La escena principal contiene, en orden y solamente mencionando los elementos relevantes, el coliseo importado desde Blender con varias modificaciones para la mejor importación a Unity, un colisionador (collider) invisible para el suelo, el sistema del jugador tanto para sus animaciones y lógica interna, su sistema de partículas para generar llamaradas, su interfaz gráfica para mostrar la vida y la stamina (aguante) y su modelo 3D.

A esto le sigue la cámara en tercera persona y el script asociado que gestiona su control, el sistema de invocación (spawning) de enemigos configurado por rondas, el menú y el código que lo controla, los modelos base de los enemigos los cuales se instanciarán para generar más, el cielo, el reproductor de vídeo para la cinemática inicial, el gestor de audio, el gestor de la secuencia de fin de juego (ejecutado una vez acaban las rondas), los sistemas de partículas para mostrar fuego en las antorchas y las posiciones donde colocar las cámaras de las animaciones.

Sistemas

Menú Principal

El sistema de menú se implementó de forma simple proporcionando el título del juego "Moon's Shadow: The Last Flame", un botón para jugar, otro para salir (cerrar el juego), y dos deslizadores (sliders) para controlar el volumen de los efectos sonido y de la música, que no del vídeo.

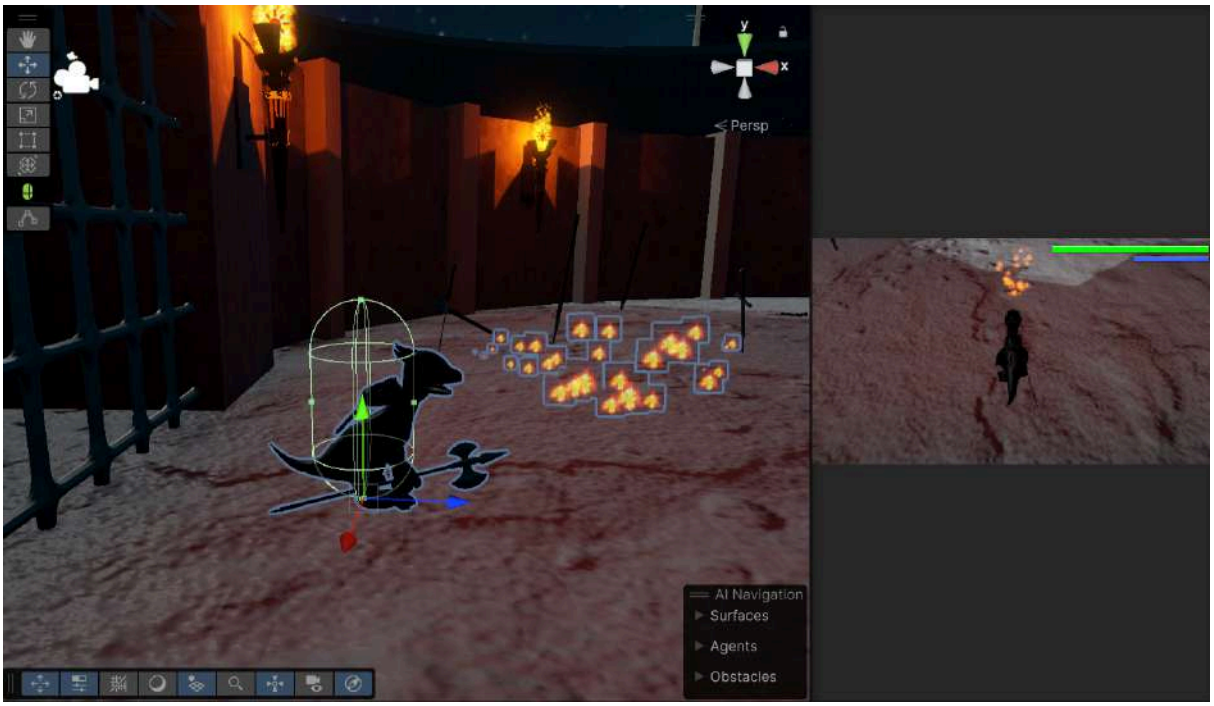
La implementación se realizó mediante el sistema de UI de Unity, utilizando Canvas con renderizado en Screen Space - Overlay para asegurar la correcta visualización en diferentes resoluciones. Los controles de volumen están conectados directamente con el AudioManger del proyecto, permitiendo ajustes en tiempo real que se mantienen durante toda la sesión de juego.

La transición del menú al juego se maneja de forma fluida, desactivando los elementos de UI del menú y activando los sistemas de gameplay de manera secuencial.

Controles y Jugador

El sistema de control del jugador se implementó siguiendo el patrón estándar de juegos en tercera persona, con controles intuitivos que permiten una experiencia de juego fluida. El dragón protagonista se controla mediante las teclas WASD para el movimiento, la barra espaciadora para saltar, la tecla SHIFT para correr, y el ratón para el control de cámara y los ataques. Con la tecla ESC se abre el menú en medio de la partida.

Los ataques se asignaron a botones específicos del ratón: el clic izquierdo ejecuta el ataque con la lanza, mientras que el clic derecho activa el ataque con llamarada de fuego.



El sistema de stamina añade una capa estratégica al combate, limitando la frecuencia de uso de movimientos y obligando al jugador a gestionar sus recursos durante los enfrentamientos. Un ejemplo de esto es cómo el ataque de llamarada que causa más daño también requiere más stamina. La barra de vida y estamina se posicionaron en la esquina superior derecha de la pantalla, siguiendo convenciones de diseño de interfaces que facilitan su lectura durante el combate.

La cámara en tercera persona se implementó con seguimiento suave del jugador y rotación libre mediante el ratón, permitiendo una visión clara del entorno y facilitando la orientación espacial durante el combate.

Sin embargo, no permite el *clipping* (atravesar) con muros, puertas o suelos para no irrumpir la inmersión.

Enemigos y IA

Los slimes funcionan como enemigos de cuerpo a cuerpo con comportamiento agresivo directo. Su IA se basa en un sistema de estados que evalúa constantemente la distancia al jugador para determinar la acción apropiada: perseguir o atacar.

El movimiento de los slimes se caracteriza por saltos direccionales hacia el jugador, utilizando las animaciones creadas en Blender para mantener la coherencia visual. Cada slime tiene su propia barra de vida que aparece encima del modelo cuando recibe daño, proporcionando feedback visual inmediato al jugador sobre el progreso del combate.

Los magos esqueletos implementan un comportamiento de combate a distancia más sofisticado. Su IA mantiene una distancia óptima del jugador: lo suficientemente cerca para lanzar proyectiles con precisión (aunque se añade cierta aleatoriedad a su puntería para que no sea muy difícil enfrentarse a ellos), pero lo suficientemente lejos para evitar ataques cuerpo a cuerpo.

Cuando el jugador se aproxima demasiado, el mago activa un comportamiento de retirada, caminando hacia atrás. Este comportamiento crea encuentros más dinámicos y obliga al jugador a utilizar estrategias diferentes para cada tipo de enemigo.

Los proyectiles de hielo tienen trayectoria constante y velocidad moderada para crear un desafío no muy difícil: lo suficientemente rápidos

para ser amenazantes, pero lo suficientemente lentos para permitir esquivarlos.

El jefe final representa el punto culminante del combate, implementando un sistema de fases que escala la dificultad progresivamente. El golem evoluciona su comportamiento según disminuye su vida.

En la primera fase, el jefe utiliza únicamente ataques con su puño izquierdo. La segunda fase introduce la invocación de slimes como esbirros, obligando al jugador a gestionar múltiples amenazas simultáneamente, además del segundo ataque del golem, que es la patada. La tercera y última fase añade la invocación de magos esqueleto además de los slimes, creando el encuentro más complejo del juego. En este momento el jefe puede atacar con sus tres tipos de ataque, siendo el de los dos puños el que más daño causa.

La barra de vida del jefe se presenta de forma especial en la parte inferior de la pantalla, junto con su nombre, para destacar su importancia narrativa y gameplay. Esta presentación diferenciada ayuda al jugador a identificar inmediatamente que se enfrenta al encuentro final.

Rondas Configurable

El sistema de rondas en EnemySpawner permite, desde el Inspector de Unity, indicar cuántos enemigos de qué tipo y en cuántas rondas aparecerán. En cada ronda, los enemigos irán saltando de las gradas al escenario hasta que todos los de esa ronda lo hayan hecho. Una vez mueren los enemigos de esa ronda, ocurre lo mismo ya mencionado pero con los de la siguiente.

Esta flexibilidad facilita el balanceo del juego y permitió iteraciones rápidas durante el desarrollo. Cada ronda se puede configurar independientemente, especificando qué tipos de enemigos aparecen, en qué cantidad, y con qué intervalos de tiempo entre spawns.

El sistema detecta automáticamente cuándo una ronda ha sido completada (todos los enemigos derrotados) y procede a activar la siguiente ronda después de una pausa configurable. Esta mecánica mantiene el ritmo del juego y proporciona momentos de respiro entre oleadas de enemigos.

Audio

La banda sonora del juego utiliza "Soul of Cinder", un *soundtrack* de Dark Souls 3, como tema principal durante los combates, proporcionando una atmósfera épica que complementa la intensidad de las batallas. La música se reproduce en bucle continuo y se puede ajustar mediante el slider de volumen del menú principal.

Debido a que este proyecto no tiene objetivo de recaudar dinero ni fama, no ocurriría ningún problema de tipo copyright.

El sistema de efectos de sonido cubre todas las acciones principales del juego:

- **Jugador**: Sonidos diferenciados para ataques con lanza, ataques con fuego, pasos al caminar, y pasos más intensos al correr.
- **Enemigos**: Cada tipo de enemigo tiene sonidos únicos para ataques y recepción de daño.
- **Ambientales**: Efectos sonoros para las animaciones cinemáticas, incluyendo la explosión de la luna, el derrumbe de la puerta y la transición a los créditos.

La implementación de AudioManger utiliza dos AudioSource, uno para efectos de sonido y otro para la música.

Iluminación

Para optimizar el rendimiento sin sacrificar calidad visual, se implementó un sistema híbrido de iluminación. Las antorchas distribuidas

por el coliseo utilizan iluminación prebaked (prehorneadas), calculada durante el tiempo de desarrollo y almacenada en lightmaps.

Esta técnica reduce significativamente el costo computacional durante el runtime (durante la partida), permitiendo tener múltiples fuentes de luz decorativas sin impactar los fotogramas por segundo. La luz del corazón del jefe final también utiliza esta técnica, creando un efecto visual distintivo sin penalización de rendimiento.

La luz solar se mantiene como iluminación en tiempo real para generar sombras dinámicas correctas que respondan a los movimientos de los personajes. Esta decisión de diseño equilibra el impacto visual con el costo de rendimiento, priorizando las sombras dinámicas donde más impacto tienen.

El material emisivo del sol se configuró con intensidad variable entre rondas, creando el efecto narrativo de que el eclipse se debilita progresivamente y el sol recupera su fuerza a medida que el jugador avanza hacia la victoria.

Partículas

El sistema de partículas más complejo del proyecto simula la destrucción de la luna durante la secuencia final. Se implementaron dos sistemas coordinados: uno para los fragmentos grandes de la luna que se dispersan por el cielo, y otro para los fragmentos menores que crean un efecto de explosión más detallado.

Cada antorcha del coliseo incluye un sistema de partículas para simular las llamas. Aunque inicialmente se consideró usar efectos creados en Blender, se optó por utilizar los sistemas de Unity al ser muy difícil importarlos correctamente.

El ataque de fuego del jugador utiliza un sistema de partículas parecido al de las antorchas que se activa desde la boca del dragón, coordinado con la animación de escupir fuego para mantener la sincronización visual.

Durante el desarrollo se experimentó con un sistema de partículas para simular nieve cayendo, pero se descartó debido a problemas de rendimiento. La nieve ralentizaba significativamente la escena, especialmente en hardware menos potente, por lo que se priorizó la fluidez del gameplay sobre este elemento decorativo. Sin embargo, se mantuvo en la cinemática inicial debido a que el cómputo ya está realizado dentro del propio vídeo.

Colisiones

El sistema de colisiones se implementó utilizando los Colliders estándar de Unity, configurados específicamente para cada tipo de objeto:

- **Jugador y Cámara**: Colisiones con suelo y paredes usando Rigidbody y Collider.
- **Enemigos**: Colisiones entre jugador y enemigos para detectar proximidad y contacto. Estos tienen dos Collider, uno para las colisiones a tener en cuenta por el motor de física, y otro utilizado como hitbox por el jugador para que no sea muy difícil golpearles.
- **Armas del Suelo**: Las lanzas, espadas y antorchas del coliseo tienen colisiones que impiden que el jugador las atraviese.
- **Proyectiles**: Sistema de detección por triggers para los proyectiles de los magos esqueleto: cuando un proyectil golpea un enemigo no esqueleto, le resta vida. Si golpea al jugador, ocurre lo mismo. Si colisiona con una pared o suelo, desaparece.

Cinemáticas

Al pulsar Jugar en el menú empieza la cinemática realizada en Blender la cual introduce al jugador al reto que se le viene por delante. Esta puede ser saltada manteniendo la tecla ESC por dos segundos.

La transición a la ronda final incluye una secuencia cinemática donde la cámara enfoca al jefe saltando desde el palco hasta la arena. Durante esta secuencia la interfaz de usuario se oculta temporalmente para crear un momento narrativo impactante.

Al derrotar al jefe final, se activa una secuencia compleja que combina múltiples elementos:

1. **Transición de Cámara**: La cámara se mueve automáticamente para enfocar la luna, ocultando la interfaz de usuario.
2. **Explosión Lunar**: Activación de sistemas de partículas coordinados con la destrucción del modelo de la luna.
3. **Efectos de Iluminación**: El eclipse se disipa y la luz solar recupera su intensidad completa.
4. **Derrumbe de la Puerta**: La puerta bajo el palco se destruye mediante animación y un sonido impactante.
5. **Transición a Créditos**: Fade a blanco seguido de la secuencia de créditos tras cruzar la puerta derribada.

Esta secuencia se diseñó para proporcionar un cierre narrativo satisfactorio que recompense al jugador por completar el desafío.

Scripts y Arquitectura de Código

El código se organizó siguiendo principios de responsabilidad única, con cada script encargándose de aspectos específicos del gameplay:

Sistemas de Enemigos:

- **Enemy.cs:** Clase base con comportamiento común (movimiento, ataque, vida).
- **SlimeEnemy.cs:** Especialización para comportamiento de salto.
- **MageEnemy.cs:** Especialización para ataques a distancia.
- **BossEnemy.cs**: Lógica compleja de fases e invocación de esbirros.

Sistemas de Gestión:

- **EnemySpawner.cs:** Control de rondas y spawning de enemigos.
- **MainMenuManager.cs:** Gestión de estados de menú y del reinicio de la partida.
- **AudioManager.cs:** Reproducción de sonidos.

Sistemas de Jugador:

- **PlayerController.cs:** Input, movimiento y ataques del jugador.
- **ThirdPersonCamera.cs:** Control de cámara en tercera persona.

Sistemas de Secuencias:

- **EndGameSequenceManager.cs:** Manejo de las secuencias cinemáticas de la luna y de la puerta.
- **DeathScreen.cs:** Manejo de la secuencia de la muerte.
- **CreditsTrigger.cs:** Manejo de la secuencia de los créditos.

Sistemas de Animaciones:

- **NotificarAnimacionesEnemigo.cs:** Notificador de cuándo empiezan o acaban las animaciones de los enemigos.
- **NotificarAnimacionesJugador.cs:** Notificador de cuándo empiezan o acaban las animaciones del jugador.

Sistemas de Barras de Vida:

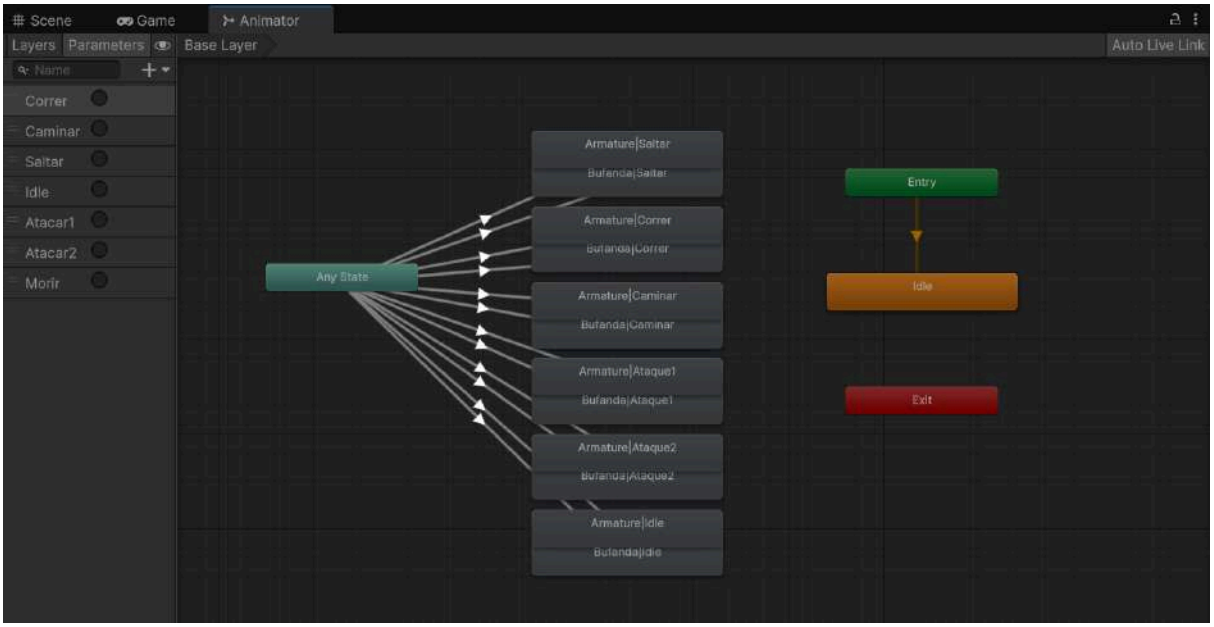
- **PlayerHealth.cs:** Gestor de la barra de vida del jugador.
- **EnemyHealthBar.cs:** Gestor de la barra de vida del enemigo común.
- **BossHealthUI.cs:** Gestor de la barra de vida del enemigo jefe final.

Integración con Modelos de Blender

Proceso de Importación

La importación desde Blender requirió ajustes específicos en la configuración de Unity:

- **Configuración de Importación**: Escalado apropiado, generación de colisiones, cambio de texturas, de fuentes de luz, etc.
- **Materiales**: Recreación de materiales de Blender usando los shaders de Unity.
- **Animaciones**: Configuración de Animation Controllers para manejar las transiciones entre animaciones.



Problemas y Soluciones

Durante la integración surgieron varios desafíos técnicos:

- **Normales Invertidas**: Algunos modelos requerían corrección de normales.
- **Escalado Inconsistente**: Necesidad de estandarizar escalas entre diferentes modelos.
- **Materiales Procedurales**: Conversión de materiales procedurales de Blender a texturas estáticas (baked), aunque no se pudo con todas. Las texturas que no se pudieron pasar se sustituyeron con otras de internet[15] [16].

Conclusiones de la Implementación en Unity

La implementación en Unity logró integrar exitosamente que no perfectamente todos los elementos modelados en Blender en una experiencia de juego cohesiva y funcional.

Los sistemas de configuración flexible, particularmente el sistema de rondas editable desde Inspector, demostraron ser invaluables durante el

proceso de balanceado y testing. Esta flexibilidad permitió iteraciones rápidas y ajustes finos del gameplay.

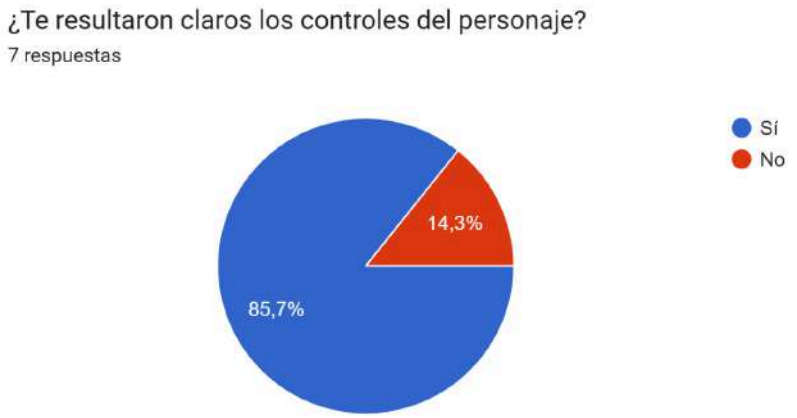
El equilibrio entre calidad visual y rendimiento se logró mediante el uso estratégico de técnicas de optimización, manteniendo la experiencia visual rica sin comprometer la fluidez del juego.

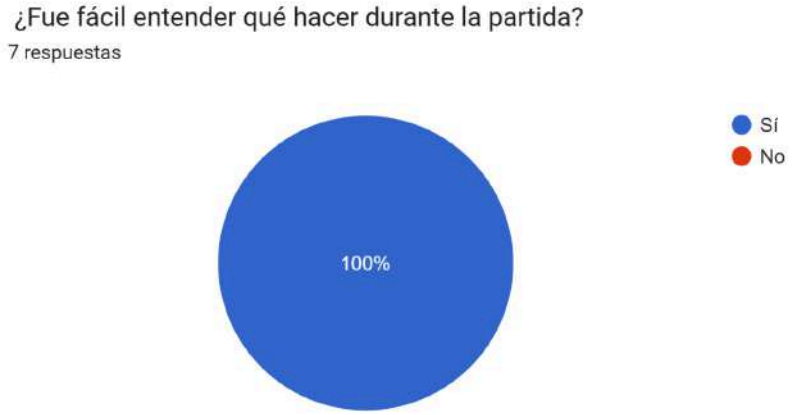
Aunque el videojuego tiene muchas posibles mejoras, como hacer la animación de caminar más fluida en el videojuego, se iría fuera del alcance del proyecto.

Como pensamientos finales diríamos que hemos cumplido con lo esperado con un producto que ha demostrado nuestra determinación, pasión y conocimiento.

Evaluación y Feedback

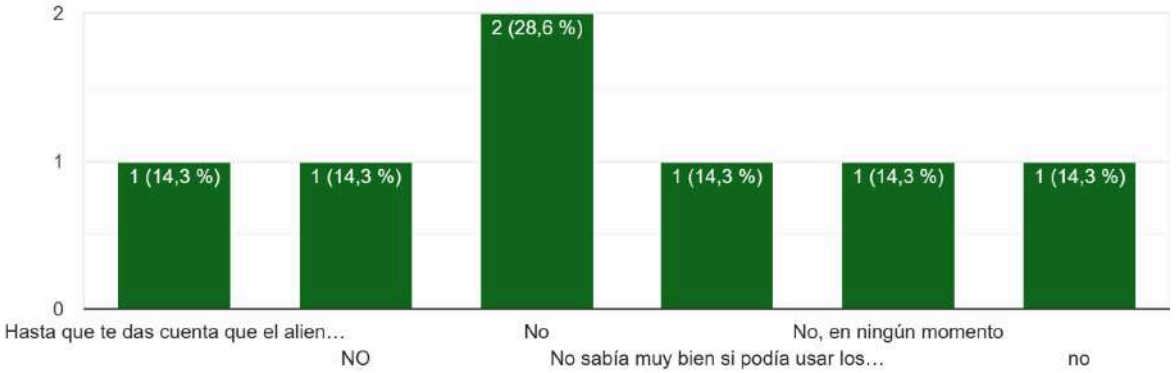
Para la evaluación final del juego, realizamos un formulario entre varios jugadores que tuvieron acceso a la beta. A continuación mostramos las respuestas:





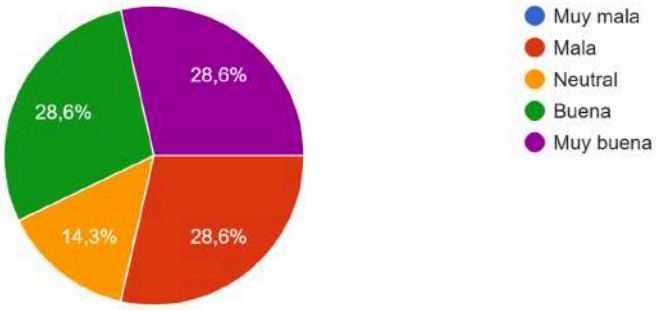
¿Te sentiste perdido en algún momento? ¿Dónde?

7 respuestas



¿Cómo valorarías la interfaz (vida, stamina, menú)?

7 respuestas



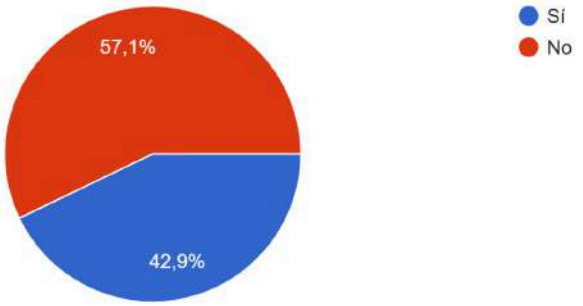
¿Cómo describirías la ambientación y el diseño artístico?

7 respuestas

Bonito
La ambientación está bonita, tanto el coliseo como los personajes (NPC's y jugador) están bien animados. Sobre el diseño, los colores están bien elegidos así como el diseño del nivel.
Impoluto , sobre todo el golem tod ese que es muy majo
chulo
Parecía de película
La ambientación, en general, es interesante y, a nivel artístico, podría mejorarse un poco más
Me gusta mucho la iluminación y que el personaje sea un dragón

¿Algún momento en el que no supieras si el juego había reaccionado a tu acción?

7 respuestas



¿Qué fue lo que más te gustó?

7 respuestas

- El coliseo
- El poder "volar" con el deagón cuando saltabas y esprintabas. Y que una vez iniciado el juego te deje elegir el volumen del juego, normalmente, siempre se ponene al 100% y eso acaba molestando.
- el golem
- la entrada de TOD
- El escenario
- La animación del principio del juego
- La animación del fuego del dragón

¿Y lo que menos?

7 respuestas

- Texturas
- La respuesta del ataque secundario
- lo demas
- el ataque básico es muy débil.
- Hay que clicar mucho el clic derecho
- A nivel visual, los ataques del jugador y de los enemigos no queda nada claro, no es fácil determinar cuando estás infligiendo daño o cuando lo estas recibiendo además de que las "hitbox" se sienten mal (a veces parece que le estas golpeando al enemigo a nivel visual pero no parece que le baja la visa)
- Que no hubiera ninguna llave :(

Conclusiones

A lo largo del proyecto, hemos pasado por todas las fases clave del desarrollo de un videojuego: desde el diseño conceptual hasta la implementación final en Unity, siendo esta la parte más problemática.

Aunque en algunos momentos la inexperiencia se ha hecho notar, especialmente en temas de optimización y modelaje, también hemos sabido buscar soluciones, adaptarnos y sacar adelante un proyecto funcional y visualmente atractivo. Cada error nos ha enseñado algo.

Nuestro mayor desafío ha sido sobre todo pasar a Unity, ya que es lo que ha determinado muchas de nuestras decisiones. Algunos modelos los hemos tenido que repetir para que no diesen problemas al importar a Unity. Además, hemos tenido que intentar buscar solución al problema de importar las texturas, ya que por mucho que las intentásemos bakear, había muchas que no se importaban bien.

En general, encontramos que el paso a Unity nos ha ayudado a entender mejor cómo realizar los modelos en Blender. Los modelos, en general, son más eficientes en cuanto a distribución de vértices.

Manual de Usuario

Podemos observar el juego al completo en el siguiente playthrough.

El objetivo principal del juego es sobrevivir las rondas de enemigos que atacan al jugador hasta que aparezca el enemigo final TOD. Después de una intensa batalla el jugador se alzará victorioso y podrá salir del coliseo o perecer en el intento.

WASD para el movimiento, barra espaciadora para saltar, mantener SHIFT mientras te mueves para correr, ratón para el control de cámara y los ataques y ESC para abrir el menú.

Los ataques se asignaron a botones específicos del ratón: el clic izquierdo ejecuta el ataque con la lanza, mientras que el clic derecho activa el ataque con llamarada de fuego.

Además, si el jugador se queda quieto durante unos segundos el dragón hace la animación de idle.

Bibliografía

1. Base de referencia coliseo
2. Base para antorcha
3. How to Model a Character Mesh in Blender | Game Character Tutorial #1 de Ironbark Games Studio
4. How to Sculpt an Alien Head | Learning to Sculpt in Blender de Ironbark Games Studio
5. Easy Character Modeling, Little Scarf Dud || Blender 2.93 de Joey Carlino
6. Blender Tutorial Slime 3D
7. [Blender] Quickly Creating a Base Mesh For Sculpting/Modelling
8. BlenderKit
9. How to Bake Textures in Blender and Export to Unity
10. Rigging Made Easy: Learn How to Rig Your 3D Character in Blender Like a PRO de Ironbark Games Studio
11. Unlock the Secrets of 3D Character Skinning in Blender: Step-by-Step Full Tutorial de Ironbark Games Studio
12. Easy Walk Cycle, Character Animation with Blender de Joey Carlino
13. Cómo ANIMAR un PERSONAJE 3D para VIDEOJUEGOS
14. Blender Quick Hand Rigging
15. ambientCG
16. The Best Assets for Game Making | Unity Asset Store