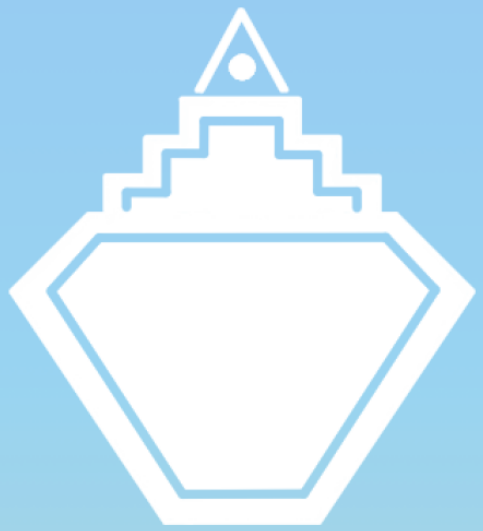
Creadoras

Carolina Galán Otero

Silvia Prego Ferro

Valeria Santiago Parra



Directores

Antonio Seoane

Ángel Fariña

Nēsos

UNIVERSIDADE DA CORUÑA

Índice

Introducción

Nēsos es un trabajo de fin de Grado que desarrolla la demo de uno de los niveles del videojuego de aventura en 3D, para PC, con el mismo nombre. En las siguientes páginas hablaremos del proceso de diseño y desarrollo de la demo; mostraremos el trabajo de creación de personajes y diseño del entorno realizado durante la preproducción, así como el diseño del nivel escogido para su desarrollo. Además, se adjunta el documento de diseño del juego completo disponible en los anexos.

En conjunto con este documento se entrega una demo jugable de lo que sería el tercer nivel del juego, el Sector 3 – Zona 2, con una duración de entre 10 y 15 minutos. Se busca presentar tanto a la protagonista como las mecánicas disponibles, así como mostrar parte de la historia de la isla. Decidimos optar por el desarrollo del tercer nivel porque era una zona muy detallada, con un entorno interesante para que el jugador pueda explorarlo, pero sin llegar a perderse en la trama del juego por ser un nivel muy avanzado en la historia.

Descripción del proyecto

Sinopsis

Nēsos es una isla flotante oculta en los cielos desde hace cientos de años, desde que un tsunami amenazó con acabar con toda la vida de la isla. Dividida en cuatro zonas bien limitadas, sus habitantes viven sin conocer el peligro que les acecha.

Ío, una joven de las aldeas exteriores, se ve obligada a adentrarse en la ciudad después de que toda su aldea sea raptada. En su viaje deberá recorrer los tres sectores de Nēsos para salvar a su familia y descubrir el secreto que se esconde en lo más profundo de la isla.

Público objetivo

Jóvenes de entre 12 y 17 años, que sean jugadores habituales. Residentes en casa de sus padres, los cuales les mantienen económicamente. Interesados en historias de fantasía tanto para cine como para videojuegos. Suelen pasar mucho tiempo con el móvil, y sobre todo en redes sociales.

Otro mercado que puede verse influenciado es el de jóvenes adultos de entre 18 a 25 años, con capacidad económica y nostalgia por los juegos de plataforma antiguos o con preferencia por los juegos indie.

Análisis DAFO

Debilidades

- Juego indie rodeado de juegos de grandes distribuidoras.
- Dificultad a la hora de conseguir el estilo deseado.
- Género muy explotado.

Amenazas

- Competencia de las grandes distribuidoras ya consolidadas.
- Otras producciones indie desarrolladas por gentenueva que quiere entrar en el sector.

Análisis DAFO

Fortalezas

- Rediseño de la leyenda de Atlantis con una estética que une elementos variados.
- Historia del videojuego muy detallada.

Oportunidades

- El juego está pensado para un público a partir de 12 años, pero puede atraer a personas de más edad por recordarles a juegos de su infancia.

Análisis DAFO.

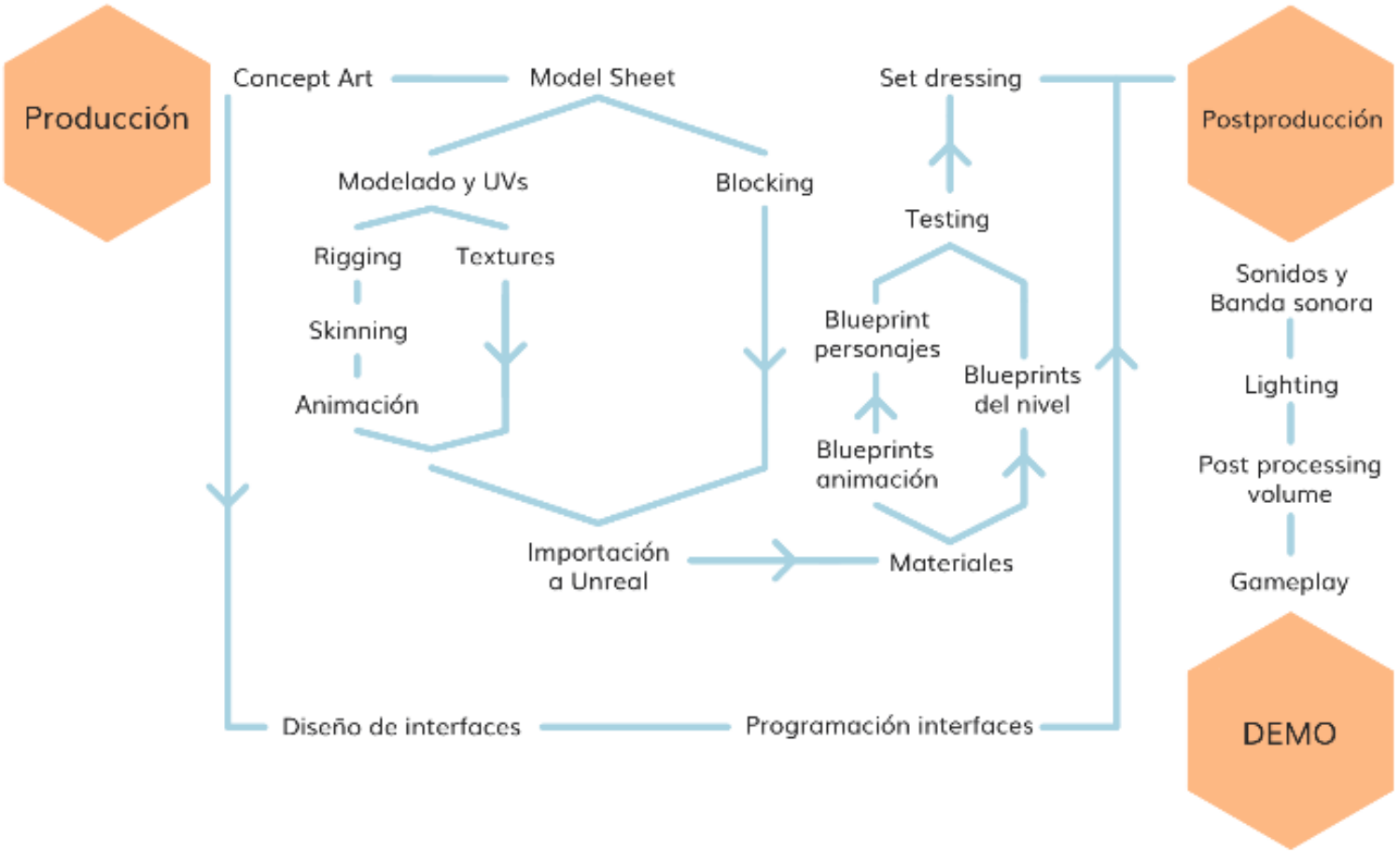
Objetivos del proyecto

El principal objetivo de este proyecto es poner en práctica lo aprendido en 3D en el grado de Comunicación Audiovisual durante estos años. Gracias a la realización de esta demo, hemos podido experimentar cómo sería una producción similar, trabajando en equipo y siguiendo el pipeline establecido, a pequeña escala.

Además, nuestro principal reto fue conseguir que todo lo ideado en la fase de preproducción del proyecto se hiciese realidad combinando los conocimientos adquiridos durante estos años y adquiriendo otros nuevos para obtener el resultado deseado.

Pipeline

A continuación se muestra el pipeline seguido para la correcta organización de nuestro proyecto.



Pipeline de producción.

Medios técnicos

Durante el proceso de preproducción y desarrollo de la demo recurrimos a diferentes programas en función de las necesidades del proyecto. Para mantener el proyecto organizado y actualizado a disposición de todas las integrantes del grupo se optó por utilizar el almacenamiento en OneDrive.

A continuación se muestra la lista con los programas utilizados.

Maya 2018

Mixamo

Substance Painter

Unreal Engine 4

ClipStudio Paint

Audacity

Imagen visual

A lo largo del desarrollo del TFG diseñamos diversos elementos visuales que ayudaron a crear individualidad a la demo del juego; sin embargo, no lo presentamos como un manual de identidad corporativa, ya que éste no entraba en nuestros objetivos.

Logo

El logo del juego es una imagen muy simplificada de la isla, Nēsos, donde ocurren los hechos de la historia. La base en forma de diamante sería la tierra flotante sobre la que se sostiene la ciudad; la parte de en medio, que recuerda a una escalera, son los tres sectores de Nēsos. El triángulo y la esfera de la zona más alta del logo representan la torre final, el objetivo de Ío, donde ésta termina su aventura.

Logo de Nēsos.

Nivel a desarrollar

Introducción a la demo

Para la creación de la demo, se optó por el tercer nivel del juego ambientado en la Zona2 del Sector 3 de Nēsos. Este nivel nos permite conocer parte de la isla y presentar las mecánicas base del juego en un entorno entretenido y único.

En las primeras versiones de la demo, el nivel era mucho más amplio y de mayor duración que el resultado final, pero gracias a la asignatura de Interacción 3D pudimos testear una primera versión del prototipo y hacer los cambios oportunos a partir de ahí. Por esto se optó por realizar cambios en el nivel para poder tener exploración, una versión simplificada de las torres y un enemigo final.

Historia

La isla de Nēsos, oculta en los cielos desde hace cientos de años, corre un grave peligro. Sus habitantes viven aislados y agrupados según su escala social en tres sectores diferentes. Las aldeas, fuera de los muros de la ciudad, están habitadas por la gente más humilde. En el Sector 3 viven los trabajadores de las fábricas y en el Sector 2, los intelectuales. El Sector 1 es conocido por la riqueza de sus habitantes, pero nadie sabe realmente cómo ni quiénes viven en él. La única persona conocida de ese sector es Atlas, el gobernante de Nēsos.

Nuestra historia empieza con Ío, una niña de las aldeas exteriores de la isla. Cuando toda su aldea es raptada por una misteriosa nave proveniente de la ciudad, Ío se embarca en una aventura para rescatar a su familia. En su viaje se adentrará en los diferentes sectores de Nēsos, algo que nadie ha conseguido jamás, y encontrará aliados dentro de la ciudad que la guiarán en su búsqueda.

En las entrañas de la isla, Ío descubrirá el gran secreto que esconde Nēsos y deberá enfrentarse cara a cara con el gobernante, viéndose obligada a salvar la isla de su cruel destino.

Diagrama de nivel

En la siguiente tabla se explican los eventos que tienen lugar en la demo realizada.

Evento	*Descripción*
S03_RET006	Vence a los enemigos.
S03_CIN003	Presentación del entorno al jugador.
S03_RET007	Llega a la primera torre.
S03_RET008	Vence a los enemigos.
S03_RET009	Activar los tres interruptores para que el ascensor funcione y subir a lo alto de la grúa.
S03_RET010	Cruza el brazo de la grúa.
S03_RET011	Vence a los enemigos.
S03_RET012	Activa la batería para girar el brazo de la grúa.
S03_TG001	Giro de la grúa.
S03_RET013	Vence a los enemigos.
S03_TG002	Presentación de Karkínus, el boss del nivel.
S03_RET014	Vence a Karkínus.
S03_RET015	Activa las dos baterías de la Gran Fábrica.
S03_RET016	Activa el interruptor para rotar el panel de la Gran Fábrica.

A continuación, se muestra el diagrama de la demo creada.

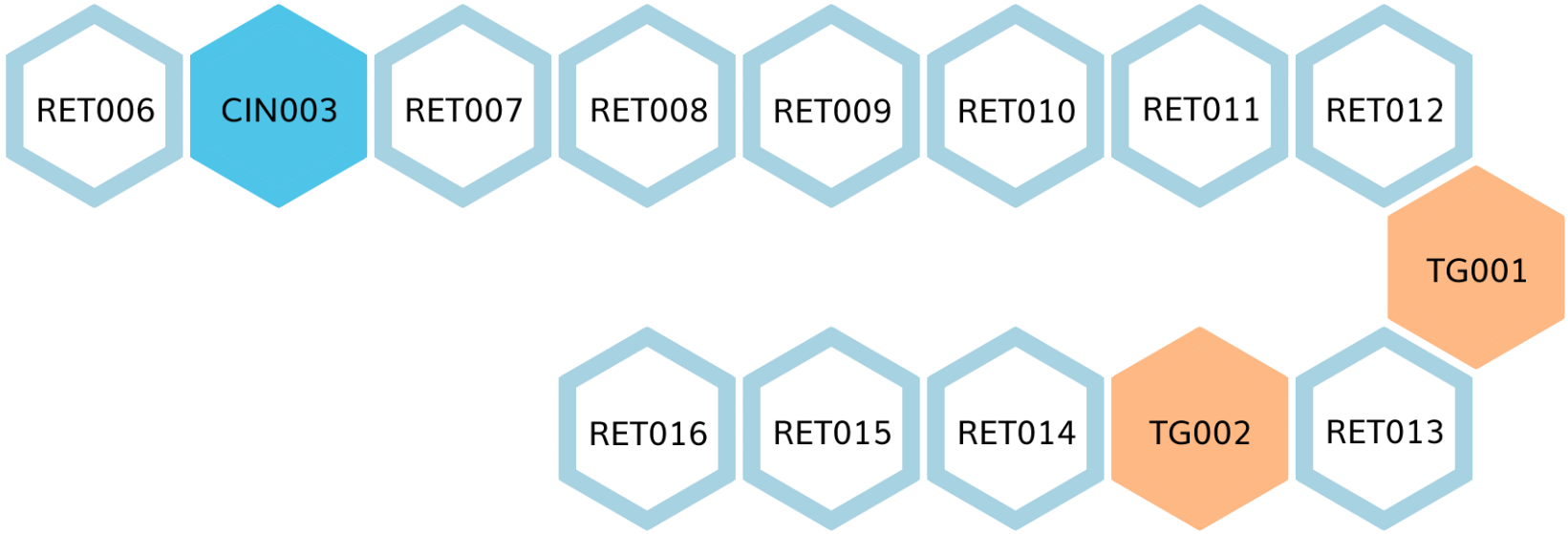


Diagrama de la demo.

Ambientación y diseño

Uno de los objetivos del diseño fue lograr una estética estilizada para los entornos y personajes. Centrándonos en el uso de figuras geométricas, nos enfocamos en darle un lenguaje visual al mundo que ayudara a contar la historia y a darle pistas al jugador.

De esta manera establecimos que el hexágono fuese la forma base relacionada con toda la isla. El cuadrado sería un símbolo muy relacionado con el tercer sector y sus entornos industriales; el círculo referencia al segundo sector, que a su vez está influenciado por la antigua Grecia y corresponde al sector de los intelectuales; y el triángulo como símbolo del primer sector, el más envidiado y rico de la isla. A su vez, los tres sectores se identifican con tres metales. El Sector 3 se relaciona con el cobre; el Sector 2, con la plata; y el Sector 1 con el oro.

Además de esto, los entornos están basados en diferentes áreas del planeta y su paleta de colores, que sirvieron de inspiración para el desarrollo de Nēsos.

Referencia 1.

Referencia 2.

Entorno

Para el terreno del nivel buscamos crear un ambiente estilizado y que pudiera encajar en la idea global del juego.

Para las paredes del cañón en el que se encuentra el Sector 3, buscamos referencias en entornos rocosos similares a los del desierto de Arizona, ya que tanto los colores como el terreno eran muy similares a lo que buscábamos. Además, para añadir el toque estilizado, adaptamos las columnas de basalto de Islandia al terreno.

En cuanto a la estética general del nivel, el Sector 3 destaca por las construcciones geométricas y las tonalidades tierra. Es el sector de la industria por lo que existen elementos mecánicos que se adaptan a las construcciones de los edificios.

Referencia 5.

Referencia 4.

Para inspirarnos, encontramos referencias en la forma de las construcciones de Marruecos. Las casas son cuadradas siguiendo la forma base que establecimos para el sector; con paredes desgastadas y apaños hechos con desechos del propio sector y del siguiente. Añadir elementos del siguiente sector, nos ayuda a unificar los ambientes tan variados de la isla. El resultado es un ambiente con edificios construidos a partir de materiales industriales del Sector 3 y desechos del Sector 2 basados en la antigua arquitectura griega.

Preproducción

El primer paso que realizamos en el desarrollo de este TFG fue el arte conceptual de los personajes y props que tomarán vida en el video-juego, además de los entornos y los assets que completarán su aspecto. Este proceso fue necesario para que luego fueran correctamente modelados en la fase de producción. Este apartado se puede resumir en Concept Art > Thumbnails > Iteraciones de color > Modelsheet > Rig y Skin > Texturizado (solo los assets modelados por el equipo siguieron este proceso).

Ío

Nuestra joven protagonista fue una de las cosas que estaban claras desde el inicio del juego; aunque al principio queríamos que fuera un personaje sin género especificado, nos decantamos por hacerla un personaje femenino. Su edad, a pesar de no estar especificada, busca acercarse a la del principal público objetivo (12-17 años).

En primer lugar, tomamos referencias de niños del sureste asiático, además de protagonistas de franquicias con ropa relativamente simple pero llamativa. Sobre esto último, exploramos ideas alrededor de la imagen de Goku (*Dragon Ball*) y Jak (*Jak & Daxter*), debido a sus vestimentas y la imagen juvenil, atlética y lista para la acción que proyectan.

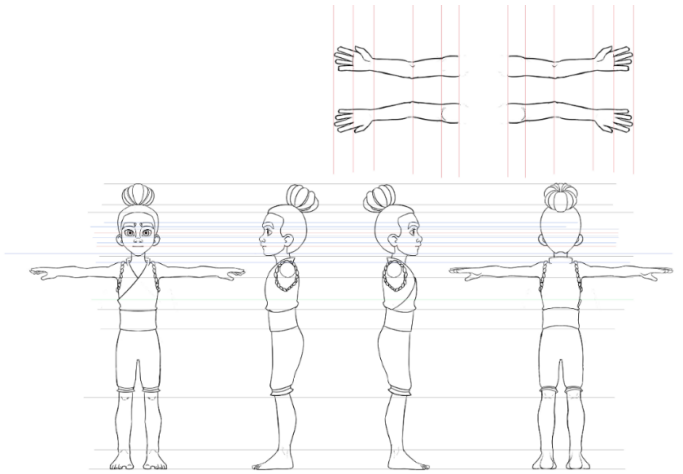
Además, cogimos ideas de otros protagonistas de videojuegos, como Link (*The Legend of Zelda*), con características en común: protagonista mudo, pero con capacidad para comunicarse, ropa que recuerda al lino...

El diseño de Ío sufrió varios cambios para simplificar procesos futuros, tales como modelado o rigging. Uno de estos cambios fue el pelo. La protagonista iba a tener melena, pero se decidió cambiar este peinado por un moño ya que era mucho más sencillo y no necesita rig. Lo mismo pasó con sus pantalones, en vez de que fueran estilo bombacho se optó por pegados a la piel y por encima de la rodilla, así evitamos colisiones en su topología y el uso de físicas para la ropa. La decisión de que no llevase zapatos, además de aumentar su imagen de "niña salvaje", también facilitó el proceso de modelado.

Concept art de Ío.

Modelado y rigging

Para el modelado de Ío nos enfocamos en mantener las proporciones originales establecidas en el *Model Sheet*. Primero se modeló su cuerpo completo y, en base a él, se modelo la ropa. El modelo en *low-poly* tiene **4.351** caras, suavizado cuenta con **17.386**.



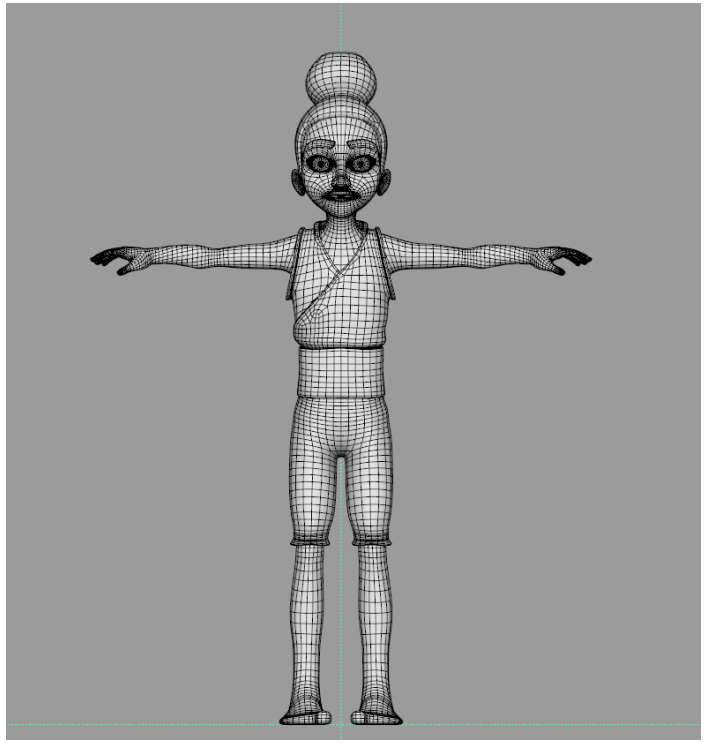
Model sheet técnico de Ío.

Pruebas de concepto de Ío.

Ío fue el enfoque principal y el primer modelo realizado, puesto que es nuestra protagonista y único personaje humanoide. Debido a que es un modelo **estilizado**, uno de los mayores retos fue enfrentarse a modelarla sin pensar en el realismo, en este aspecto sirvió de mucha ayuda tener el *model sheet* de guía. Este nos ayudó a la hora de realizar las facciones, manos y pies de Ío, las partes del cuerpo más complejas. Otra zona que fue complicada fue la unión de las piernas por la poca familiaridad con una unión recta del torso con las piernas.

La **ropa** de Ío fue modelada en base al cuerpo y detallado con herramientas de escultura en Maya. La camisa fue uno de los objetos más complejos debido a que el corte lateral está modelado, por lo que hubo que hallar la manera de que la geometría siguiera el movimiento del corte diagonal sin agregar más caras, pero estuviera reforzada en las zonas del cuello y las mangas.

Todo el modelo se combinó para facilitar la exportación en fbx a **Mixamo**, dónde obtuvimos el rig, skin y las animaciones de Ío.



Malla del modelo.

Texturizado

Tras escoger en las iteraciones los colores que mejor se adaptaban a la estética general del juego, se realizó una primera versión del mapa de color difuso. Sacando desde Maya un *snapshot* de los UVs para tomarlo como plantilla, se añadió el color utilizando Clip Studio.
Posteriormente, pasamos el modelo a Substance Painter para realizar el texturizado final. Exportamos el modelo de forma que tuviéramos distintos sets para facilitar el texturizado de las diferentes partes de Ío.

Iteraciones de la ropa de Ío.

Modelo texturizado.

Tras añadirle la paleta de color definitiva (tonos violetas y malvas que resaltaran con los tonos tierra del entorno), pasamos a añadirle más detalles. En la ropa se utilizó un mapa de rugosidad para simular la tela y agregamos manchas y patrones (por ejemplo, la cinta que lleva en la cintura, formada por pequeños hexágonos). Al ser una niña pequeña y teniendo en cuenta el viaje que realiza, decidimos añadirle detalles como suciedad y arañazos en la piel para darle mayor personalidad y realismo.

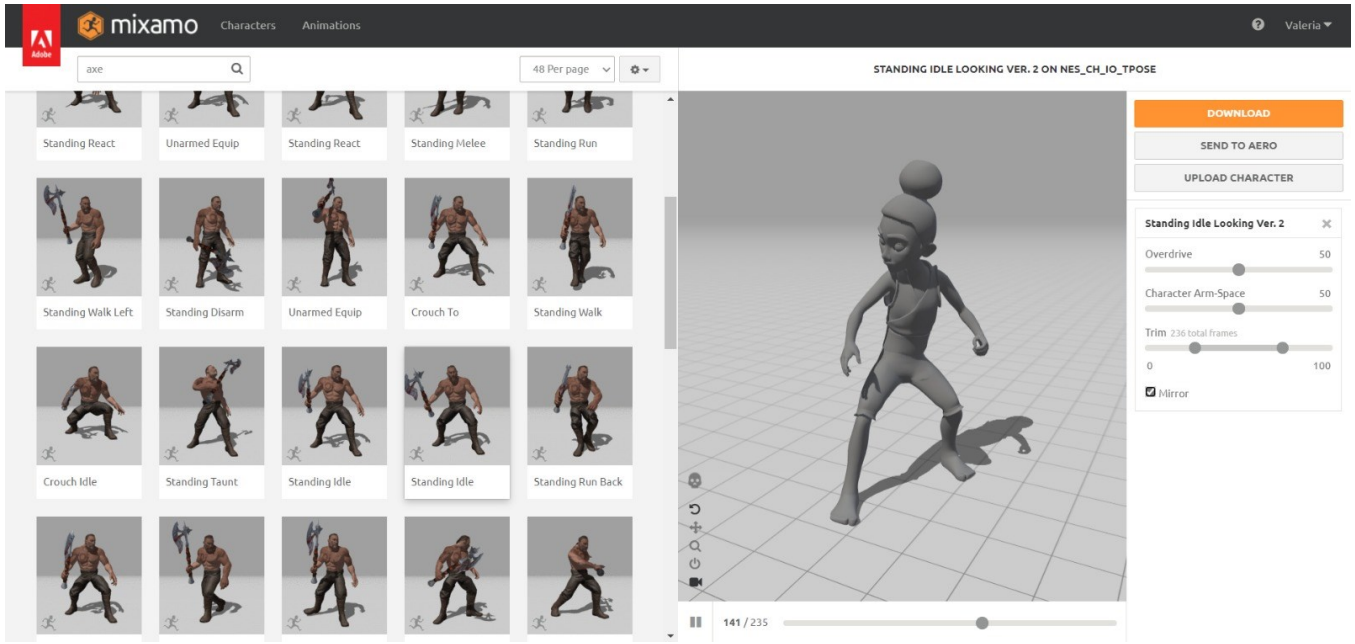
Animaciones

Ío tiene un total de 13 animaciones utilizadas a lo largo de todo el juego, las cuales fueron alteradas en Maya por medio de un **script de Python** y **capas de animación**.

El proceso principal del **script de Python** fue el de cambiar los nombres de los *joints* de Mixamo para que el esqueleto fuera compatible en Unreal, agregar un *joint root* centrado en el 0 del mundo, y agregar dos *joints* en cada mano posicionaos en el centro de la palma de Ío para utilizarlo como punto de anclaje del Méros.

Por otra parte, se utilizaron **capas de animación** en Maya para aquellas animaciones que no funcionaban debido a la manera en la que se movían. El proceso era seleccionar los *joints* que se querían alterar, colocarlos en una nueva capa de animación y reanimarlos.

Estas correcciones se realizaron principalmente en las manos, para que estuvieran cerradas o abiertas; en los brazos, para que el Méros no atravesara a Ío; o para hacer cambios más grandes, como alterar animaciones que no estaban estáticas en el 0 del mundo para poder utilizarlas en el videojuego. El mejor ejemplo es la animación del doble salto, que inicialmente se movía hacia adelante y fue alterada para que funcionara en el mismo sitio.

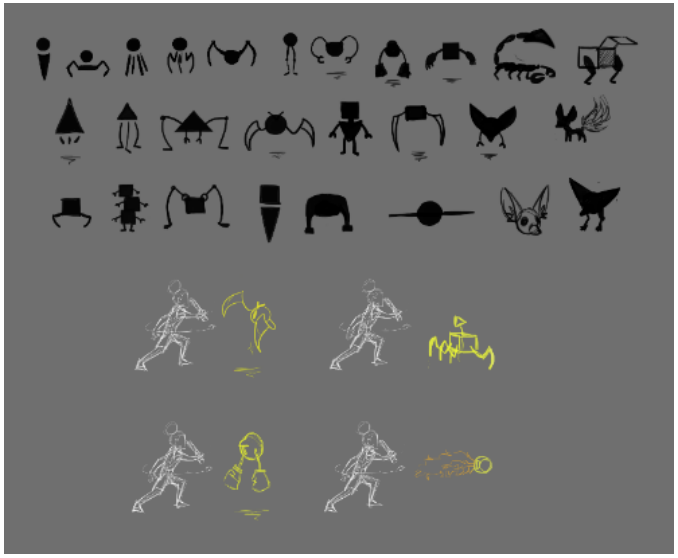


Proceso en Mixamo.

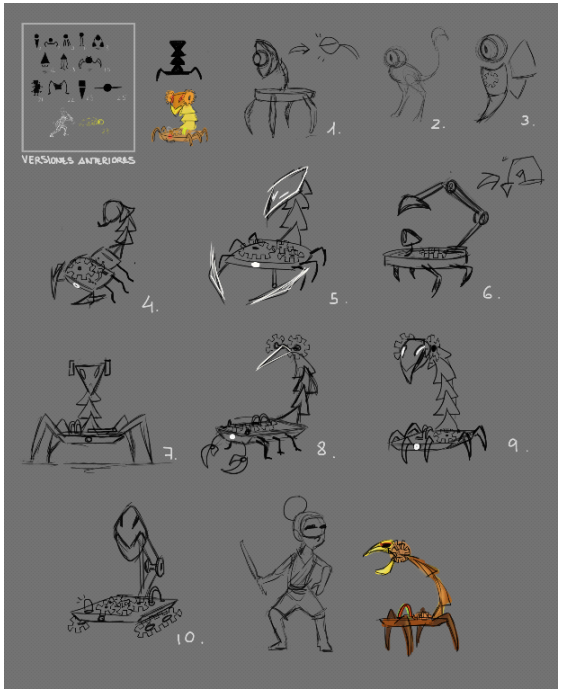
Mantokóras

Las Mantokóras son criaturas que rondan por el Sector 3. Son los primeros enemigos “serios” a los que el jugador debe enfrentarse, así que su diseño está pensado para que fuera una amenaza asequible.

Ya que este sector es el más industrial, los enemigos de esta zona tenían que estar formados también por piezas propias de un entorno industrial: metal, cobre, engranajes, acero oxidado, etc. El primer diseño incluía un enemigo que flotaba en el aire y atacaba con cuchillas, girando en su propio eje.



Iteraciones de las Mantokóras.

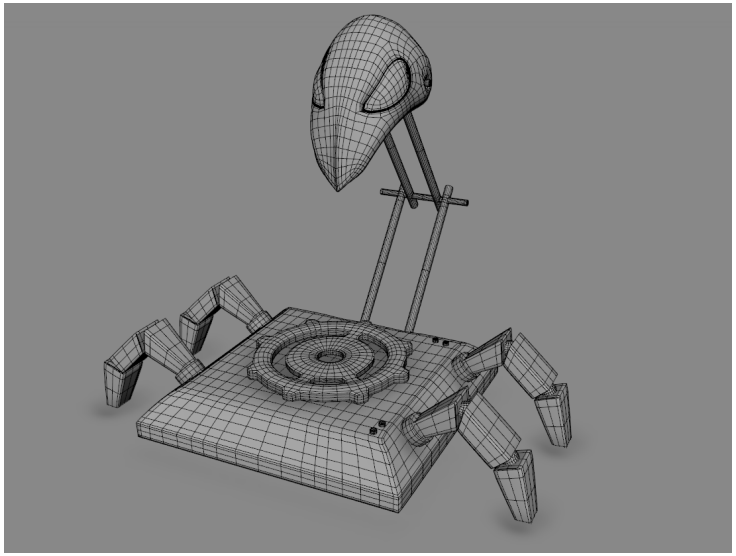


Pruebas de concepto de las Mantokóras.

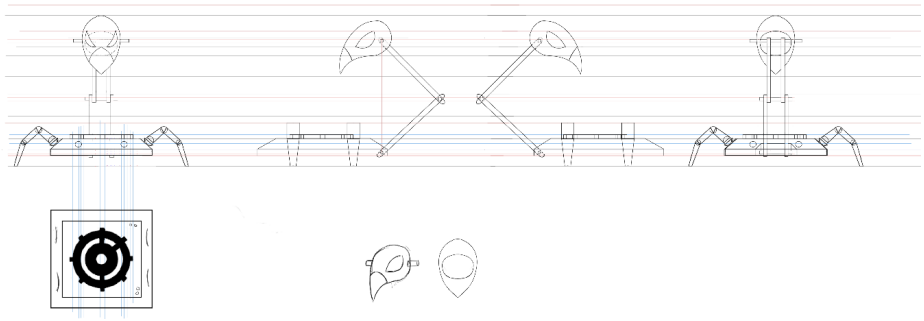
Más tarde, se exploraron diseños más mecánicos y más animalísticos. Después de implementar influencias del norte de África al desarrollo del entorno, se decidió hacer lo propio con los enemigos: la mantokóra final tiene un aspecto que recuerda a un escorpión, pero con elementos de un área totalmente industrial. El diseño está pensado para que ataque con la “cabeza” de la misma forma que un escorpión ataca con su cola. Su nombre proviene del ser mitológico griego “mantícora”, con cuerpo de león y cola de escorpión.

El modelado es **principalmente geométrico** con refuerzos en las esquinas para mantener la forma cuando se suavizara. El diseño original de las **patas** cambio para que el movimiento de la caminata fuera más natural. Inicialmente eran piezas completas, pero el movimiento de animación era muy mecánico y no era ese el objetivo del modelo, por lo cual se realizaron **divisiones** y se agregaron esferas en los puntos de las articulaciones.

Otra articulación que cambió fue el **cuello**. El modo en que se había modelado inicialmente era lo opuesto a lo que se buscaba, en vez de rotar sobre un eje, solo podía moverse de arriba a abajo, por lo que hubo que **rehacer la geometría** para que funcionara correctamente.



Malla del modelo.

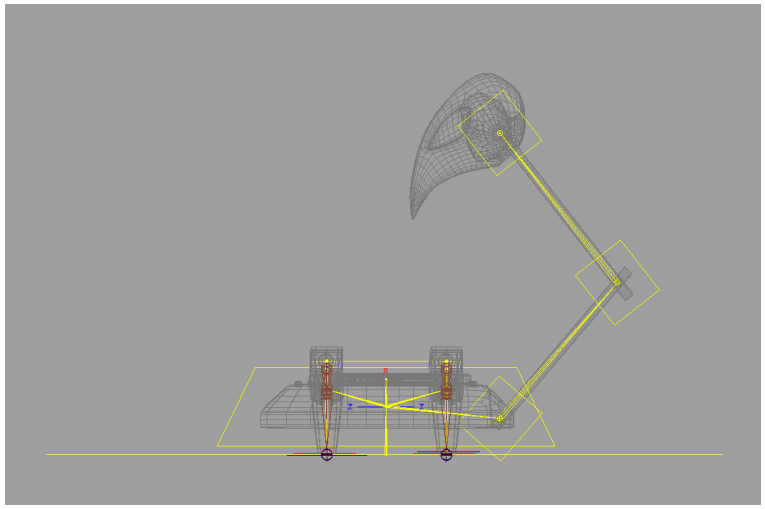


Model sheet técnico de la mantokóra.

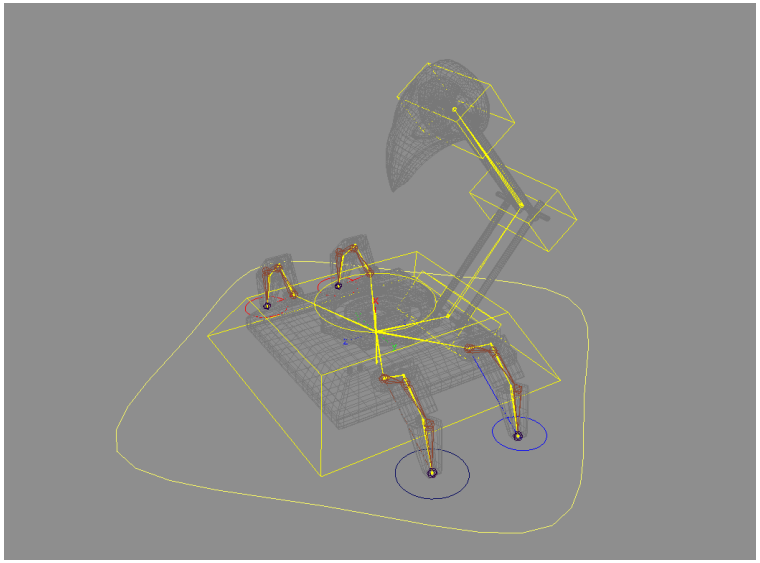
El modelado de la **máscara** fue uno de los más complejos debido a la forma y colocación de los ojos. Hubo que hacer retopología constante y utilizar herramientas de escultura para suavizar la caída de la aguja.

Una vez obtenida la forma general del casco se cortó la forma de los ojos y se fueron adaptando los *edge loops* de alrededor para que fluyeran de manera correcta. Uno de los mayores retos fue encontrar la manera en que la vista frontal y lateral de los ojos fueran lo más fieles posible al concepto de arte, manteniendo la forma, pero que transmitiera el sentimiento de amenaza al jugador.

El proceso de rig estuvo fuertemente hilado con el de modelado debido a problemas de funcionamiento que surgieron con las primeras pruebas. En su base hay un esqueleto simple que va desde el centro del mundo a la cabeza. Por otro lado, las patas fueron realizadas con *IKhandles* con controles no relacionados con el control corporal de manera que se pudieran mover independientemente y así facilitar las animaciones del personaje.



Rig del modelo (perfil).



Rig del modelo.

Texturizado

Dentro de la historia del juego, los enemigos son formas robóticas provenientes del Sector 1. Al ser el sector más rico, se relaciona con el oro, lo que nos permite utilizar este material para identificar los objetos o personajes procedentes de esa parte de la isla.

Decidimos utilizar una base metálica y color cobre para el cuerpo del modelo. Se utilizó el ***generator curvature***, lo que genera una máscara en base al modelo, en este caso para darle un aspecto desgastado en los bordes. Añadimos detalles dorados con el objetivo de relacionar el personaje con su lugar de origen.

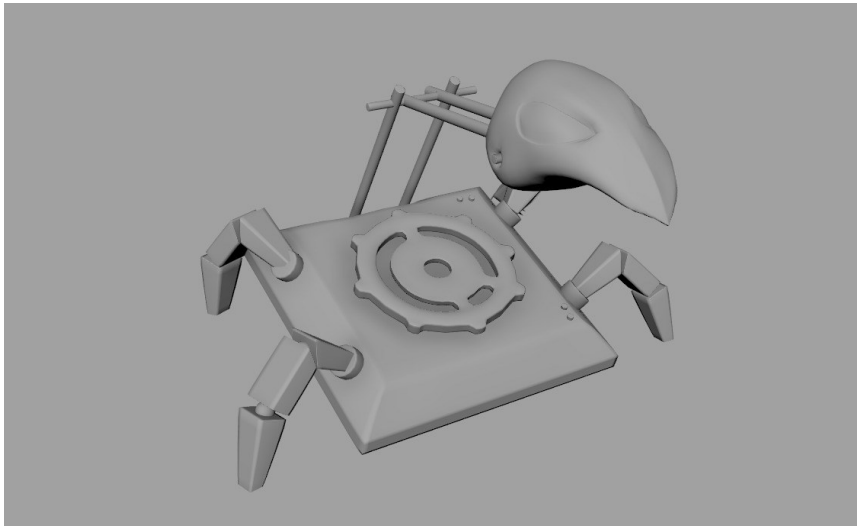
También se añadió un mapa de **color emisivo** en los ojos para hacer destacar la cara de la mantokóra. Para aplicarlos diferentes mapas se utilizó la herramienta de relleno del programa para crear de forma rápida y eficaz las máscaras necesarias.

Modelo texturizado.

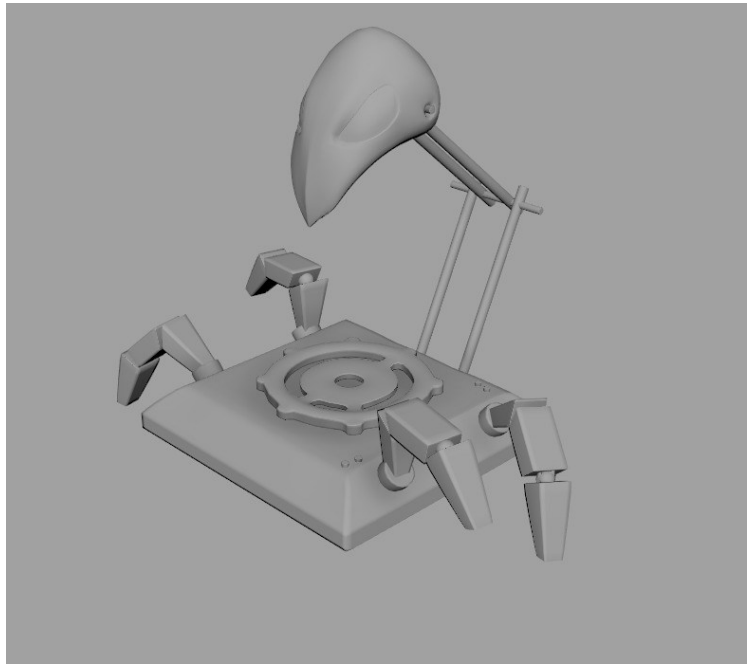
Animación

Las Mantokóras tienen 3 animaciones principales: **reposo**, **movimiento** y **ataque**.

La animación de reposo es simple: el cuerpo se mueve hacia arriba y abajo manteniendo su posición. Cuando avanza, se mueve en galope, dando pequeños saltos, y en ataque se mueve similar a un escorpión, empujando su cuerpo hacía adelante haciendo una curva con su cola.



Animación de ataque.

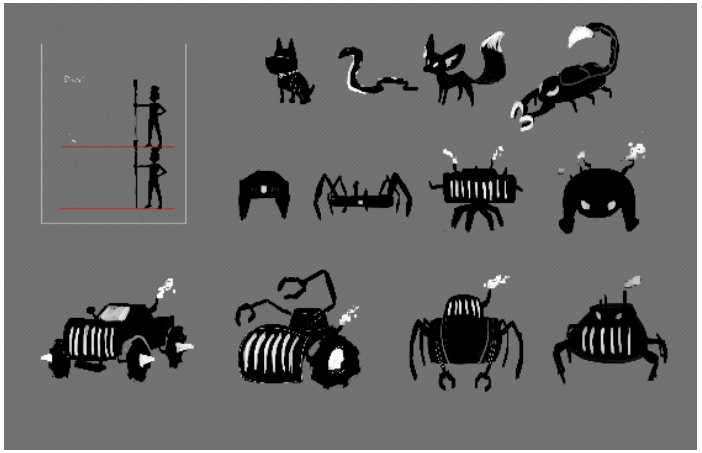


Animación de caminata.

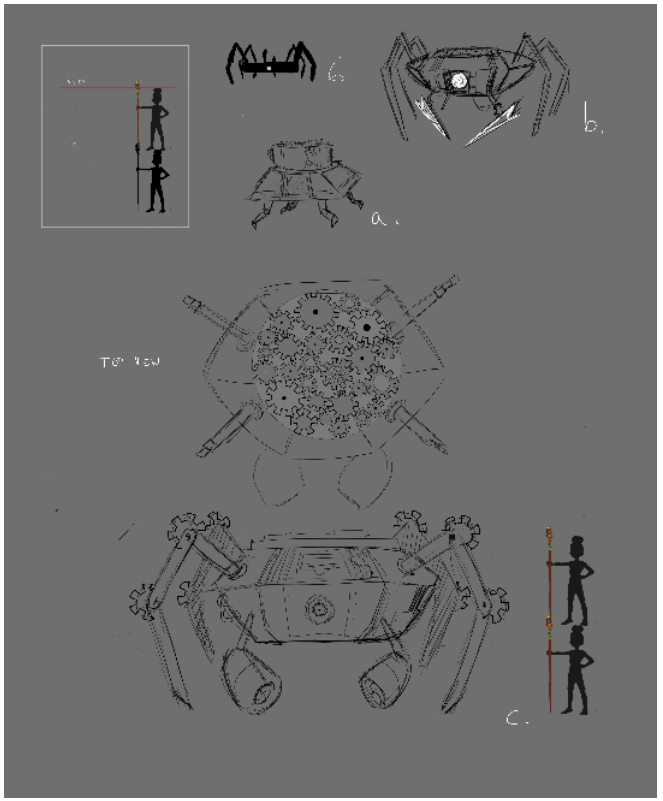
Karkínus

El boss del nivel protege al jugador de llegar a su destino, la puerta hacia el Sector 2. Es el primer boss al que el jugador se enfrenta, por lo que sus mecánicas son muy simples, pero buscábamos que su aspecto fuera intimidante de todas formas.

Igual que con las Mantokóras, con este boss buscábamos un aspecto industrial, pesado y potente. Al comienzo del proceso de diseño, exploramos varios tipos de aspectos, desde uno más animal hasta más vehicular, que recordara a un *Monster Truck*. Más adelante, para que tuviera relación con el resto de los enemigos del nivel, pero no se pareciesen demasiado, en vez de a un escorpión nos acercamos más hacia un aspecto de cangrejo/araña, con patas largas y un centro más voluminoso, pensando también en las mecánicas de combate, teniendo en cuenta las zonas golpeables y el punto débil. El diseño cambió bastante para simplificar procesos como el rigging o la animación; de todas formas, se sigue apreciando su presencia imponente original.



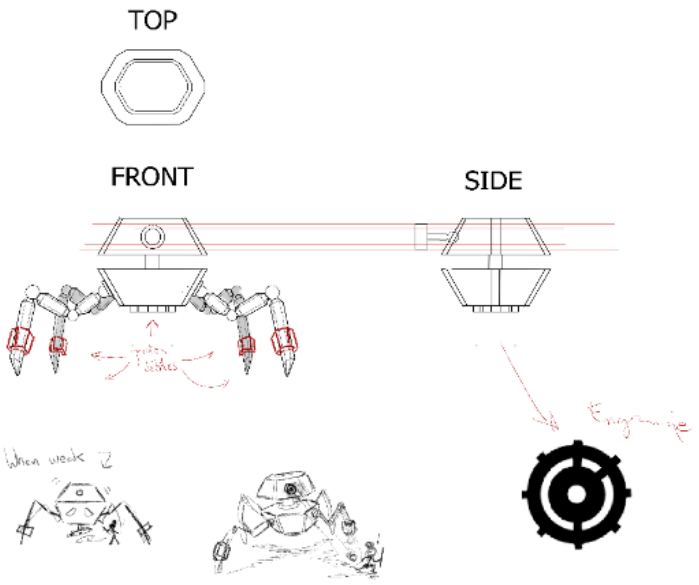
Iteraciones de Karkínus.



Prueba de concepto de Karkínus.

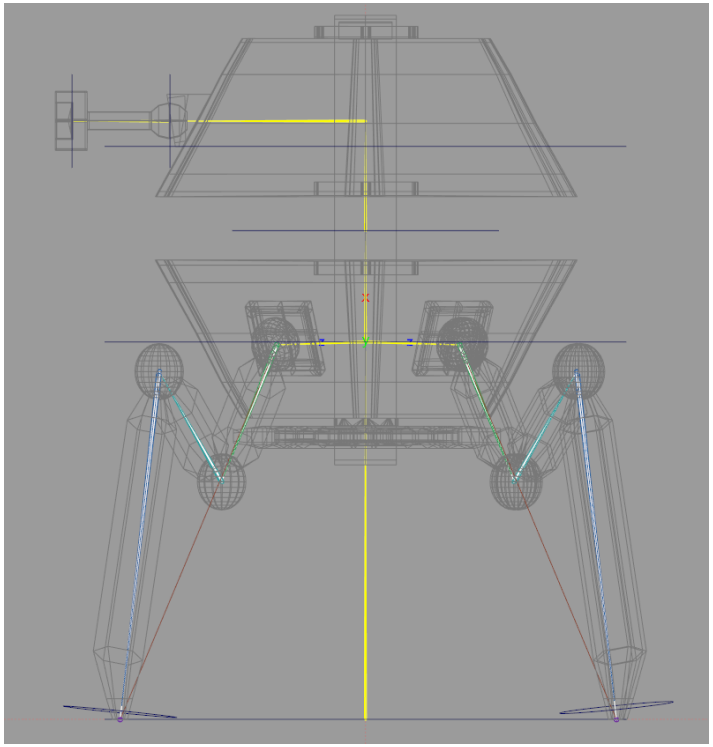
Al ser un diseño sencillo, el modelado fue relativamente rápido. Cuando se modelaron las Mantokóras, uno de nuestros tutores recalcó que, tal y como están modeladas, las patas podrían hacer contacto entre sus piezas. Esto se mejoró en el diseño del Karkínus: en vez de estar formadas por rectángulos, sus patas están construidas en base a cilindros octogonales. De esta manera, las patas no chocan entre sí, facilitando el trabajo del departamento de animación.

Para mejorar las mecánicas de combate, se optó por que la "cabeza" del boss y el engranaje, su punto débil, reciclado de la mantokóra, estuvieran combinados en una misma pieza. Así, la cabeza del boss gira, buscando a Ío y cuando colapsa, se cierra y la tuerca se muestra.

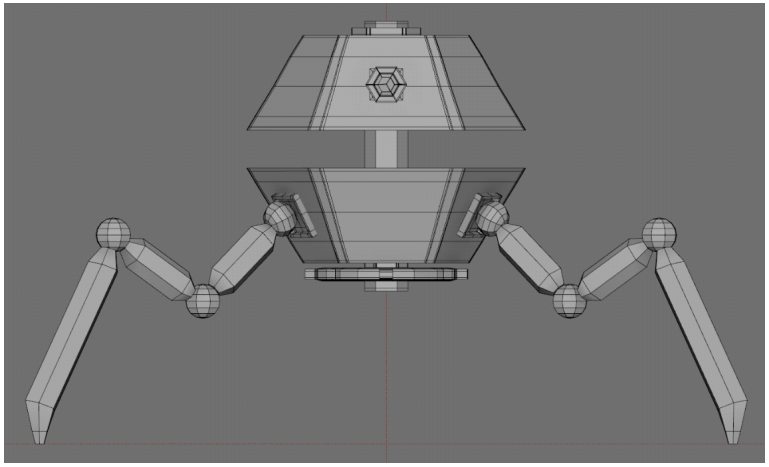


Model sheet técnico de Karkínus.

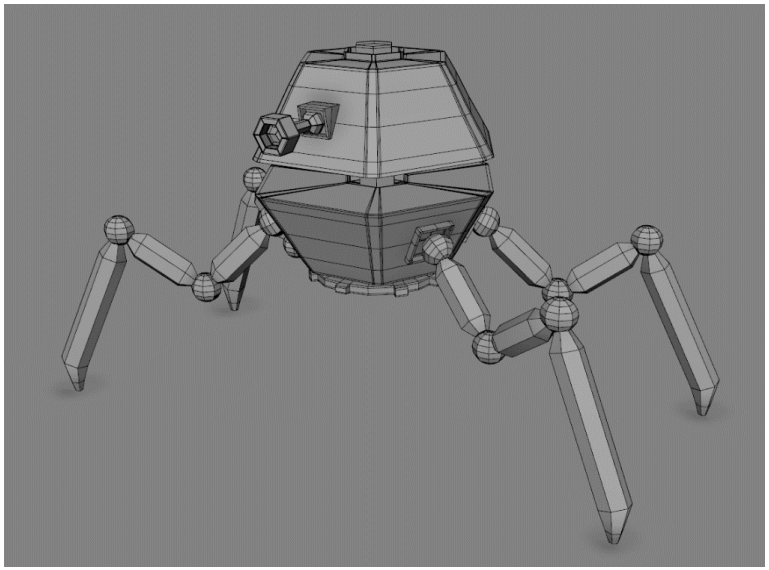
Para mejorar las mecánicas de combate, se optó por que la "cabeza" del boss y el engranaje, su punto débil, reciclado de la mantokóra, estuvieran combinados en una misma pieza. Así, la cabeza del boss gira, buscando a Ío y cuando colapsa, se cierra y la tuerca se muestra.



Rig del modelo.



Malla del modelo (frontal).



Malla del modelo.

Texturizado

Gracias a las facilidades que aporta Substance Painter, el proceso de texturizado de Karkínus fue más rápido que el de la mantokóra.

Utilizando la escena con el modelo de la mantokóra y editando las propiedades del proyecto, importamos el modelo de Karkínus. A continuación, ajustamos las máscaras para que cada parte tuviera el aspecto deseado.

Tanto las patas como el cuerpo del modelo se mantienen con los mismos colores que en el enemigo. Para destacar su sector de origen, se mantiene el color dorado en el engranaje que comparte con las Mantokóras.

Texturizado del modelo (versión 2).

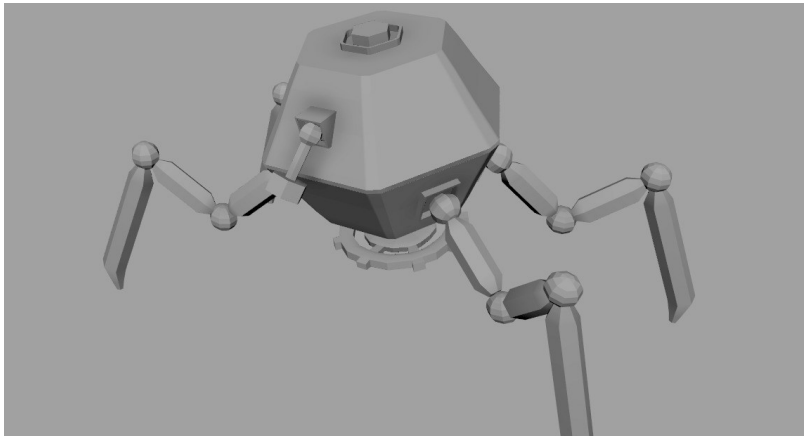
Texturizado del modelo (versión 1).

Además, fue necesaria una segunda versión del texturizado en la que tuviese un aspecto más estropeado, para poder indicar al jugador durante la batalla cuando conseguía disminuir la salud de Karkínus. Para ello se oscurecieron las patas, puntos débiles durante la batalla, y se añadió un *height map* para destacar el deterioro. También se añadió un valor emisivo a la tuerca para señalarle al jugador el momento justo de atacar.

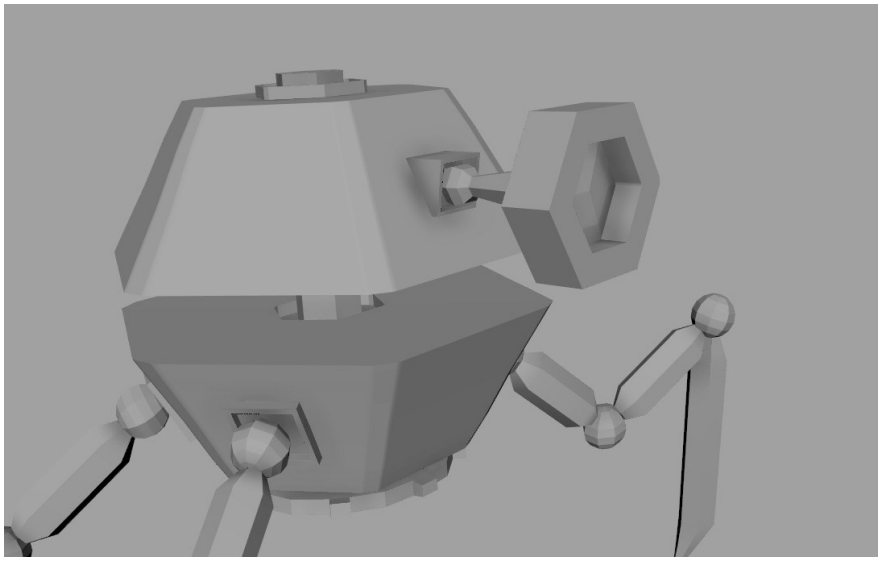
El Karkínus tiene **5 animaciones**: reposo, ataque, inicio de golpe, reposo de golpe, fin de golpe.

Se siguió el mismo proceso de animación que con las Mantokóras: realizadas en Maya, luego se hizo *bake* en los *joints*, se borraron los *constraints* de los controles y se exportaron en FBX para ser importadas a Unreal.

El proceso de golpe inicialmente estaba en una sola animación, pero debido al ciclo de ataque fue alterado posteriormente para que pudiera ser manipulado con facilidad en la programación.



Animación de aturdimiento.

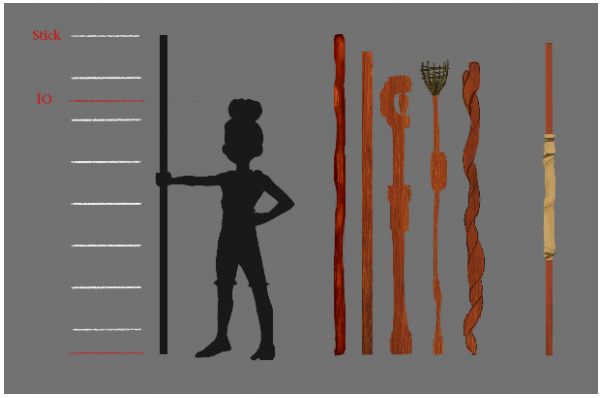


Animación de ataque.

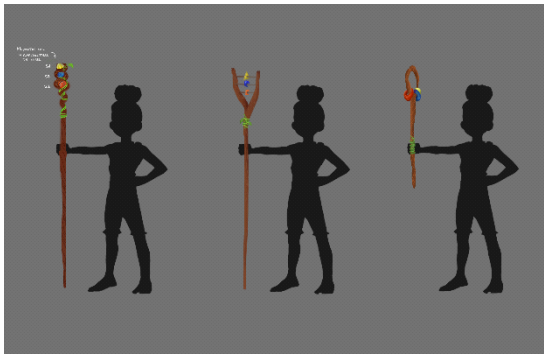
Méros

Cuando ideamos a Ío, barajamos varias posibilidades de ataque, pero acabamos decidiéndonos con que tuviera arma propia. Como se mencionó con anterioridad, parte del diseño de nuestra protagonista está basado en Goku (*Dragon Ball*), por lo que, en un primer momento, Ío utilizaría un bastón largo para derrotar a sus enemigos.

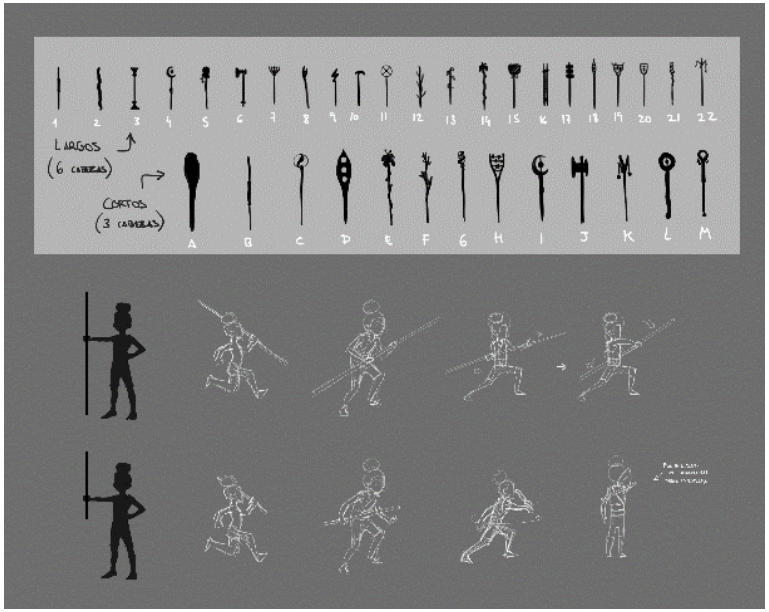
Sin embargo, para facilitar su rig, se decantó por un diseño más corto, parecido a un machete o a un martillo. Entre varias opciones, se optó por un diseño que mezclara partes naturales e industriales. Además, esta arma debía tener un espacio para la gema que permite al jugador pasar de nivel a nivel activando las diferentes máquinas de los puzles del juego. Por último, se bautizó como Méros, "parte de" en griego, ya que para Ío esta herramienta es indispensable para su vida diaria.



Prueba de concepto 1.



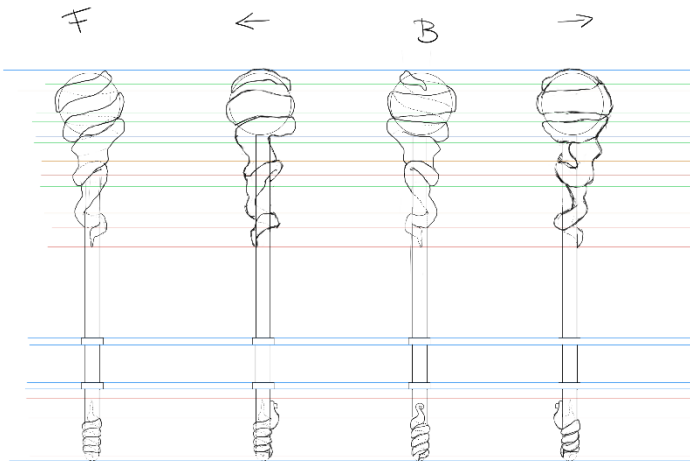
Prueba de concepto 2.



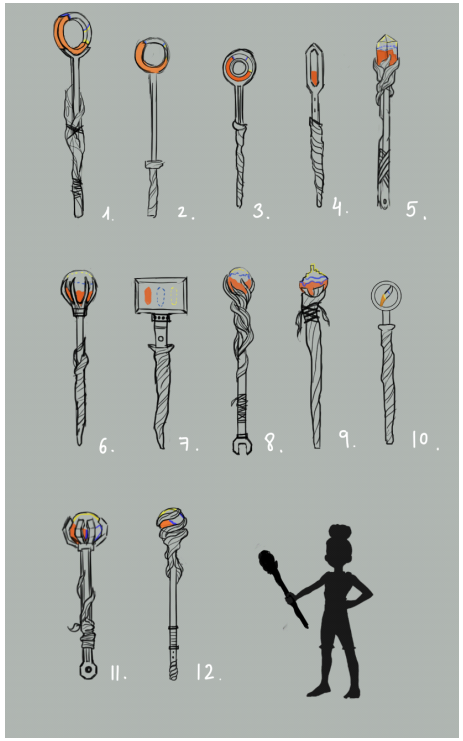
Iteraciones del Méros.

El modelado de este prop siguió en su mayoría el diseño final; se hicieron algunas modificaciones para una mejor topología y resultado final.

El Méros está formado por **tres piezas**: el mango, la rama que se retuerce alrededor de él y la gema. Aunque dos de estas tres piezas fueron relativamente fáciles de hacer, existieron más complicaciones en la restante, la rama del árbol: fue complicado recrear una forma natural que además contuviera la topología correcta. El resultado final, de todas formas, fue satisfactorio.



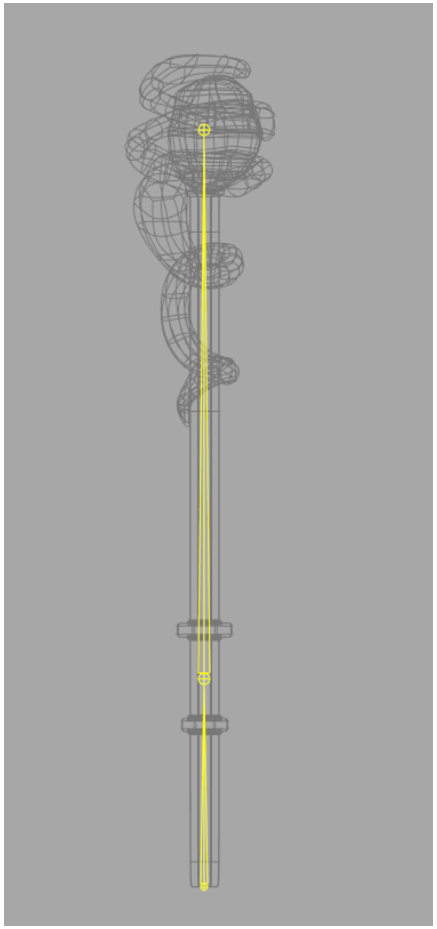
Model sheet del Méros.



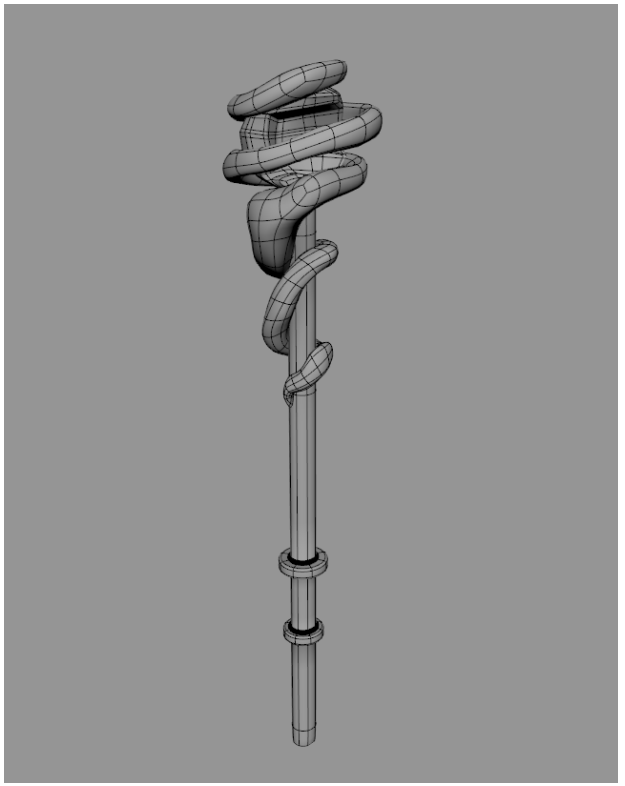
Pruebas de concepto 3.

Cabe destacar que la gema, al modelarla, no debía ocupar todo el espacio disponible en la rama, ya que más adelante en el juego (fuera de la demo) Ío encontrará otras dos, que se juntarán y formarán una esfera que será crucial para el final del juego.

El **rig** del Méros tenía el objetivo de facilitar la colocación de un *socket* en la gema para facilitar la programación en Unreal. En este caso no se hicieron controles porque no hace falta realizar animaciones.



Rig del modelo.



Malla del modelo.

Además, esta arma debía tener un espacio para la gema que permite al jugador pasar de nivel a nivel activando las diferentes máquinas de los puzles del juego. Por último, se bautizó como Méros, "parte de" en griego, ya que para Ío esta herramienta es indispensable para su vida diaria.

Texturizado

Como ya hemos explicado, el modelo está dividido en **tres partes**: la gema, la rama y el mango. Para la gema se optó por un tono anaranjado que representara el Sector 3, en el que proporciona más ayuda a Ío. Al ser un objeto escondido durante años se le añadieron marcas de desgaste con el mapa de rugosidad.

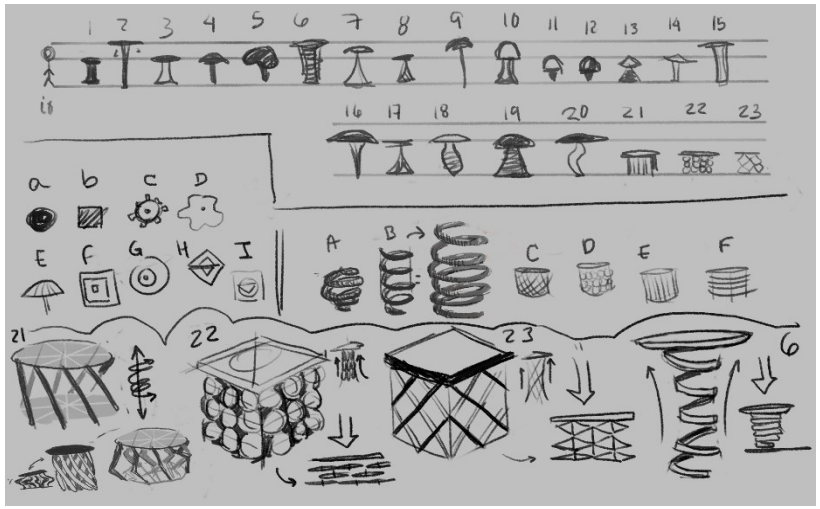
Para la rama se utilizaron dos tonos de marrón y un *height map* para añadirle los nudos característicos del tronco de un árbol y darle un aspecto más estilizado.

En cuanto al mango metálica, escogimos un aspecto desgastado y tras varias versiones optamos por un color marrón rojizo con el objetivo de que no atrajese demasiado la atención ni tapara la gema, parte más importante del Méros.

Texturizado del modelo.

Hongos

Para llegar a los sitios altos, Ío se puede ayudar de los hongos que crecen por el Sector 3. En vez de hacer plataformas de saltos convencionales, optamos por crear una especie nueva que combinase elementos industriales de su entorno con los propios de una planta, bajo el pretexto de que el entorno está tan contaminado por el metal que la naturaleza tuvo que adaptarse. Así, este hongo es propio del Sector 3 y no se encuentra en otra fase del juego.



Iteraciones del asset.

Pruebas de concepto 1.

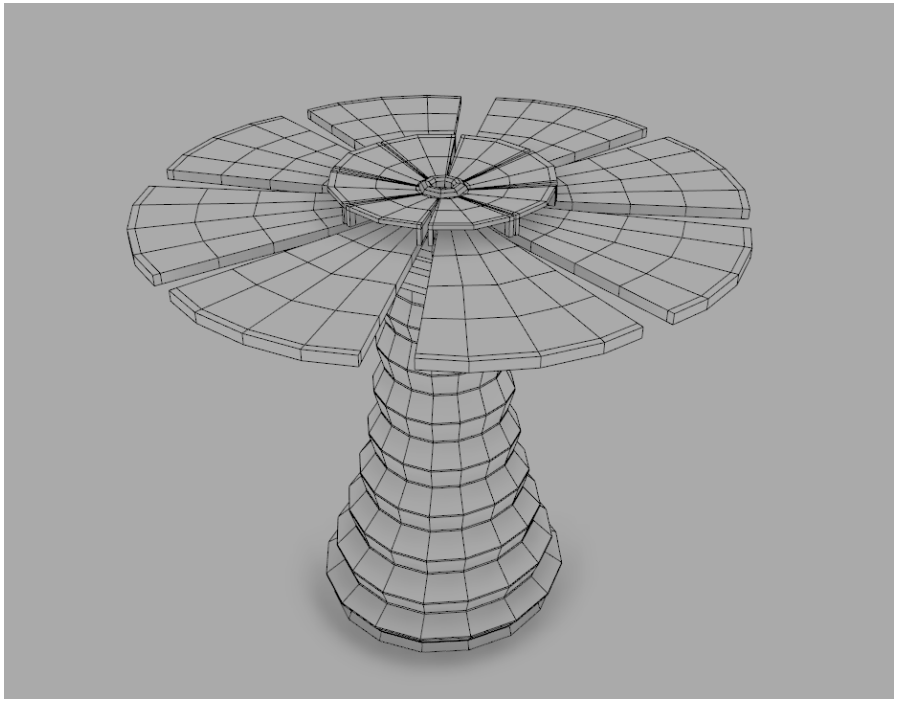
En una primera versión, las plataformas de salto iban a ser hojas tropicales, ya que el entorno iba a contar con muchos más elementos naturales. Al cambiar esto último, se modificó también el aspecto de las plataformas, llegando a la conclusión de que una flora nativa que tuvo que adaptarse a la contaminación nos parecía mucho más atractivo.

Pruebas de concepto 3.

Pruebas de concepto 2.

Para su creación se realizó un *model sheet* en el cual se tomaron referencias de tubos corrugados para la parte inferior y de hongos para la parte superior.

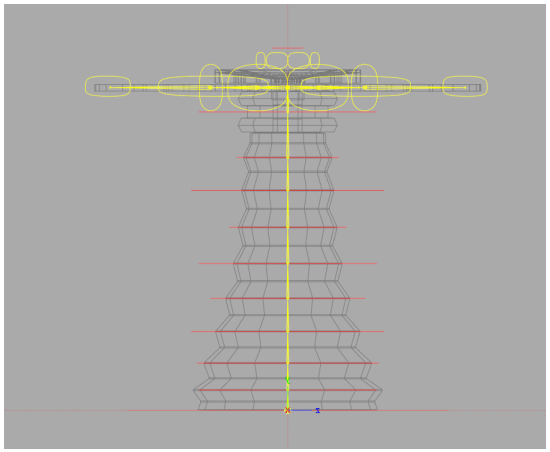
El modelado de la plataforma fue bastante complejo. Lo primero que se buscó fue hacer la forma de la base, sobre la cual luego se hicieron las divisiones necesarias para la cantidad de salientes. A continuación, se extrajeron y reforzaron las caras. La parte superior se dividió en cuatro cuadrantes y solo se modelaron dos de las hojas de manera que encajaran perfectamente una vez fueran duplicadas y reflejadas con la herramienta *mirror* de Maya.



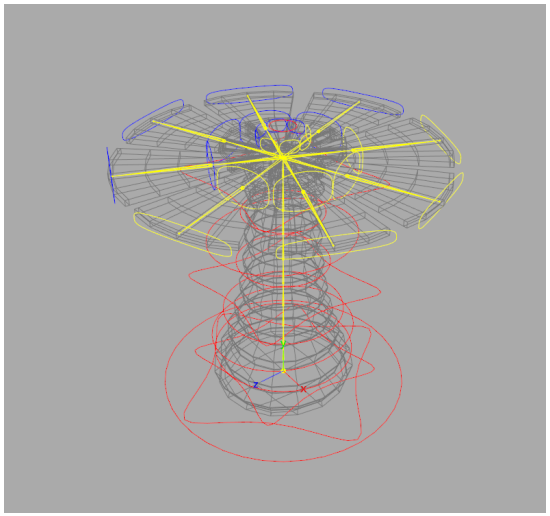
Malla del modelo.

El proceso de rig de la planta fue bastante complejo. Primero establecimos el modo en que queríamos que se moviera la plataforma y decidimos que fuera un movimiento ondeante pero repetitivo y mecánico. Para la parte **inferior** se colocaron *joints* en cada uno de los salientes para controlar completamente solo esa parte del *skin* mientras que en las bajadas comenzaban a desvanecerse. Esto sirvió a la hora de realizar los controles para que al agarrar los controles de la parte superior, la rotación del centro fuera de un 50% y por lo tanto quedara un movimiento natural.

La parte superior del rig se hizo a la mitad, duplicando y reflejando el rig previamente creado. Hay tres *joints* por cada una de las hojas y todas tenían la misma dirección en los tres ejes por lo que la animación de todas al mismo tiempo era factible, esto ayudó a movilizar la etapa de animación.



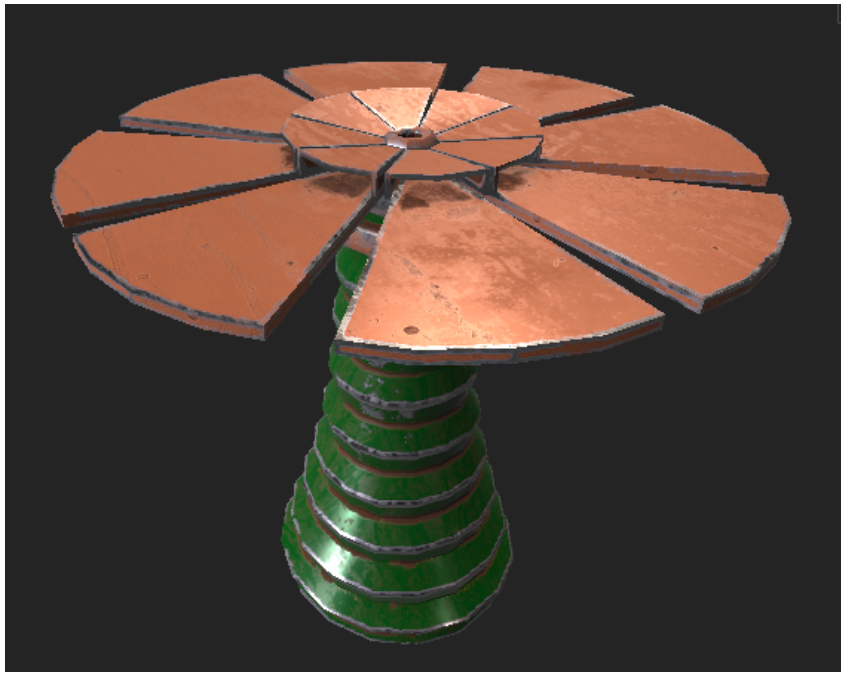
Rig del modelo (frontal).



Rig del modelo.

Texturizado

Todo el modelo está recubierto de metal, tanto el tallo como las hojas. En primer lugar se aplicó el color a cada parte del modelo. De base se mantuvo un gris y se utilizó un marrón oscuro para los detalles de suciedad. Para darle un aspecto desgastado se añadió un *generador* para que en los bordes y zonas curvas se apreciara la capa metálica original de color gris y la suciedad que simula el óxido.



Texturizado del modelo.

Animación

Se realizaron dos tipos de animaciones para las plataformas. Una para el reposo, en la que la planta rota sobre su propio eje de lado a lado, para indicar al jugador que es un objeto que puede utilizar. El movimiento es constante y un poco rápido debido a que no queríamos que fuera totalmente una planta, sino también una máquina.

El otro movimiento fue el que usa al lanzar por el aire al jugador. Es un movimiento simple en el que la planta baja y se aplasta, y luego se estira al darle impulso. Nos dimos cuenta luego en la programación que no se ve en cámara a menos de que el jugador este mirando hacía abajo al utilizarla.

Baterías

Su aspecto es 100% industrial. La idea principal era que Ío las activara dando un golpe en la base del suelo, pero al crear los interruptores cambiamos el modo de activación y mantuvimos el diseño del modelo.

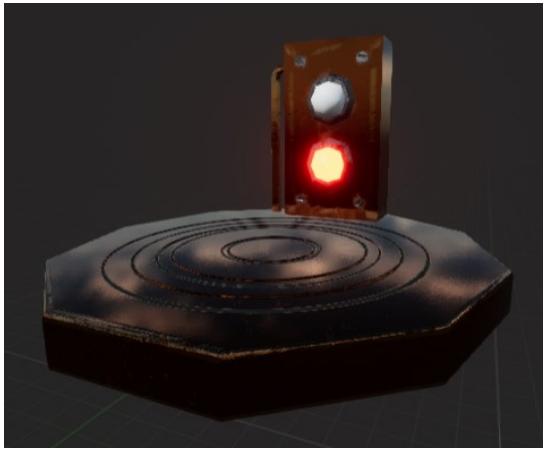
Modelado

Se reutilizó el ascensor como la base de la batería. A continuación, se utilizó un cubo para crear la parte en la que se encuentran dos luces que indicaran al jugador cuando la batería está activada y cuando no. El resultado final está dividido en tres partes: la base con la caja principal, la luz roja y la luz verde para facilitar la programación.

Texturizado

Utilizamos un *smart material* del programa y se cambió para quitarle realismo. Se añadió color gris, mapas de rugosidad y de desplazamiento para los detalles del modelo.

Para las luces se creó un material en Unreal y se crearon instancias para poder cambiarle el color y la intensidad y tener tres **variantes**: apagada, encendida en rojo y encendida en verde.



Texturizado del modelo.

Interruptores

Los interruptores son los que hacen que el ascensor te lleve a lo alto de las torres. Para establecer una relación clara entre estos y las baterías, los interruptores siguen el mismo estilo de diseño y texturizado.

Modelado

El modelado se divide en dos partes: el marco del interruptor y la luz. Al igual que con la batería, necesitábamos un modelo dividido en dos partes para poder realizar la programación de forma cómoda.

Texturizado

Al ser objetos activables al igual que las baterías, decidimos utilizar el mismo aspecto para relacionarlos fácilmente. Seguimos el mismo procedimiento que con Karkínus, cambiamos el modelo en la escena de las baterías y ajustamos las capas necesarias.

Para la luz se utilizaron las instancias ya creadas para las luces de la batería.

Texturizado del modelo.

Ascensor

En el interior de las torres se encuentra un ascensor que permite a Ío subir hasta lo más alto de la construcción.

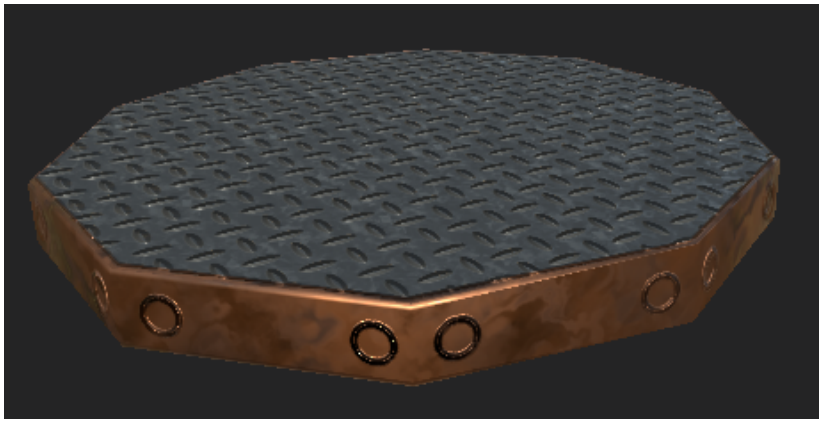
Modelado

Para en modelado se utilizó la base de la pieza de la grúa como referencia, ya que debía encajar correctamente con el hueco disponible. El resultado es un hexágono sin más detalles ya que decidimos añadírselos mediante texturizado para no complicar la malla.

Texturizado

Todo el modelo tiene una base metálica. Para el lateral de la base se combinaron dos tonos de cobre mediante una máscara a la que se le agregó una textura de ruido. Además, se añadieron detalles con un *height map* en las esquinas del modelo.

Para la parte superior se escogió un gris para el color y un height map para simular los detalles de una placa metálica.



Texturizado del modelo.

Monedas

El jugador puede recoger las monedas repartidas por el nivel. Para el diseño se tuvo en cuenta la influencia del Sector 1, por su forma triangular, el oro y los detalles verdes específicamente.

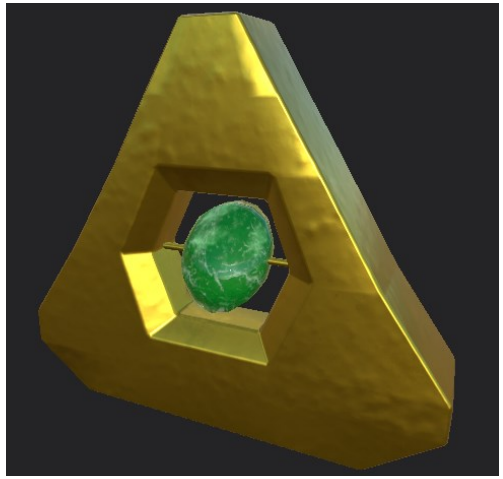
Modelado

El modelado se basa en figuras geométricas simples y se modificaron algunos vértices y caras para una correcta topología.

Texturizado

La base del modelo es de metal dorado con imperfecciones para que no quedase completamente plano. La esfera central se texturizó para obtener el aspecto de una piedra de jade. Se aprovechó un *smart material* propio de Substance y se editaron las capas para obtener el resultado deseado.

Posteriormente se añadió un valor al canal emisivo en Unreal para resaltar más el modelo sobre el terreno del nivel.



Texturizado del modelo.

Salud

Se puede encontrar dentro de los jarrones repartidos por el mundo. Elegimos anillos brillantes que giran sobre su eje, uno dentro de otro, y de color azul para que resaltara en el entorno.

Modelado

Está formada por tres anillos formando una diana. Se cambió la topología de un torus para obtener el primer anillo y, a continuación, se duplicó ese para crear los dos de su interior.

Texturizado

Para este modelo se creó un material en el motor de Unreal. Se cambiaron los ajustes para crear un material translucido y se animaron las coordenadas de la posición respecto al mundo y de las normales para dar movimiento al modelo.

También se varió su opacidad entre 0 y 1, además de añadirle emisividad. El resultado fue un material con comportamiento similar al de una burbuja.

Texturizado del modelo.

Jarrones

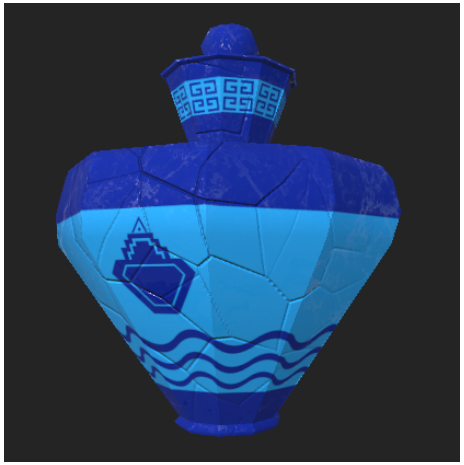
Al romperlos, estos dejan caer salud. Ya que el Sector 2 es el sector de los artistas y tiene influencias de la Grecia antigua, nos decantamos por que los elementos rompibles que tuvieran pickups estuvieran basados en los jarrones de cerámica de esta época.

Modelado

El modelado de los jarrones fue tomado del diseño realizado en preproducción y se realizó como una sola geometría con bordes muy marcados, pero movidos para que tuviera desperfectos.

Texturizado

Tras probar el modelo en el nivel, se cambiaron sus colores a dos tonos de azul. Se aplicó un mapa de rugosidad para darle aspecto de mármol y se añadieron grietas para destacar que es un objeto destruible. Además, dibujamos motivos repetidos en la parte superior e inferior del jarrón, y se creó un diseño geométrico de la isla en dos dimensiones.



Texturizado del modelo.

Landscape y nivel

Nēsos es una isla flotante por encima de las nubes, por lo tanto, su clima es naturalmente árido. El Sector 3 en concreto es el más árido de todos, con colores tierra y muros de piedra que lo rodean. Para conseguir este efecto de estar “encerrados entre piedra", el nivel fue construido modelando el *Landscape* y creando murallas que se alzan casi en ángulo recto desde el suelo. Además, a medida que se completa el nivel se va subiendo el nivel del suelo, para dar una sensación de avance más real.

Como ya se ha comentado, el nivel fue creado a partir de la herramienta *Landscape* de UE4. Esta decisión tiene como base el ahorro de recursos y el deseo de obtener un resultado orgánico. Se barajó en un primer lugar la creación de muros en Maya y su posterior exportación a Unreal, pero esta idea fue descartada.

A pesar del aspecto orgánico que queríamos, encontramos varios problemas a la hora de que pareciera verdaderamente natural, especialmente al no estar familiarizadas con el sistema de UVs de UE4. Nuestra mayor preocupación fueron las texturas “estiradas” y los riscos que se pueden ver en algunas secciones del nivel. Aun así, conseguimos un aspecto general con el que estamos satisfechas.

Por otro lado, construimos el entorno a base de crear unos pocos *assets* de forma estratégica, es decir, la gran mayoría de *assets* fueron diseñados para una construcción modular. De esta manera, pudimos utilizar sus piezas a nuestro antojo para dar la sensación de que todo era diferente, cuando, en realidad, son elementos reciclados.

De todas formas, nos apoyamos también en contenido descargado para completar la estética del entorno. Utilizamos un pack de Unreal con elementos propios de una villa medieval y usamos sólo los que consideramos que pondrían un broche final al acabado del entorno: árboles, hierba, piedras, efectos especiales...

Las texturas utilizadas a lo largo del nivel fueron descargadas y colocadas en capas para pintar el *Landscape* y así construir la zona de juego. El conjunto consta de dos texturas de tierra, una de piedras hexagonales, una de hierba y otras dos de piedra, que es la misma, pero en ejes diferentes (X y Z), para poderlas colocar en cada lado de los muros de manera cómoda y natural.

En primer lugar, el problema principal que encontramos en este aspecto fue el mapa de desplazamiento de uno de los materiales. El mapa crea una especie de "carriles" y "baches" que se dejan ver en el *Landscape*; además, al no tener colisión, Ío puede atravesarlos, por lo que es bastante notorio cuando el jugador atraviesa uno de ellos. En segundo lugar, hay zonas en el mapa que se "rompen" entre las *tiles* del *Landscape*, dejando ver el vacío que hay debajo del mapa. Este problema es generado también por el ya comentado mapa de desplazamiento.

Aun así, en un primer vistazo estos problemas no fueron notados ni por los tutores ni por las personas que realizaron los *testing*, por lo que resultó que no era tan obvio como nos pudo parecer en un primer momento. De todas formas, para solucionarlo, modificamos lo máximo posible los mapas de desplazamiento que daban problemas y, por otra parte, cubrimos con assets y *foliage* las roturas más graves.

Tuberías

Modelado

Decidimos que debían mantener la forma hexagonal. En modelado se crearon 4 piezas: una recta, una curva, una boca o salida y un detalle de soldadura para cuando quisiéramos extenderlas.

Texturizado

Manteniendo la relación con el Sector 1, se utilizó un material de oro propio de Substance y se alteraran sus capas para disminuir el estilo realista. Además, al estar situadas en el Sector 3, se añadieron dos capas para darle detalles oxidados y desgastados al metal, con el objetivo de integrarlas de forma correcta en el nivel.

Modelo final.

Casas

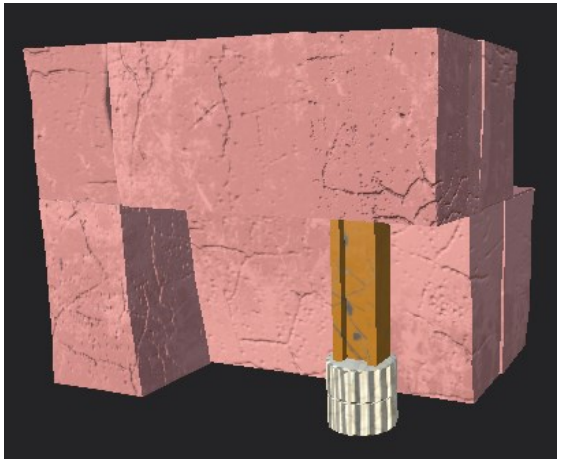
Modelado

Recurrimos al modelado modular para combinar las piezas y crear nuevos assets. El proceso fue ensayo y error debido a la poca familiaridad con el concepto. Además, realizamos **colisiones personalizadas** para cada modelo y no tener que recurrir a las colisiones complejas de Unreal.

Texturizado

Para las paredes se utilizó una textura de desgaste, y se añadieron grietas mediante el *height map*. Para las columnas, se editó un *smart material* de Substance y añadimos una capa para crear las estrías propias de las columnas griegas.

Se crearon dos *smart materials* para aplicarlos al segundo modelo con facilidad. Para la viga del segundo modelo optó por una base gris metálica y se añadió una capa anaranjada con una textura de arañazos.



Texturizado del modelo.

Viga

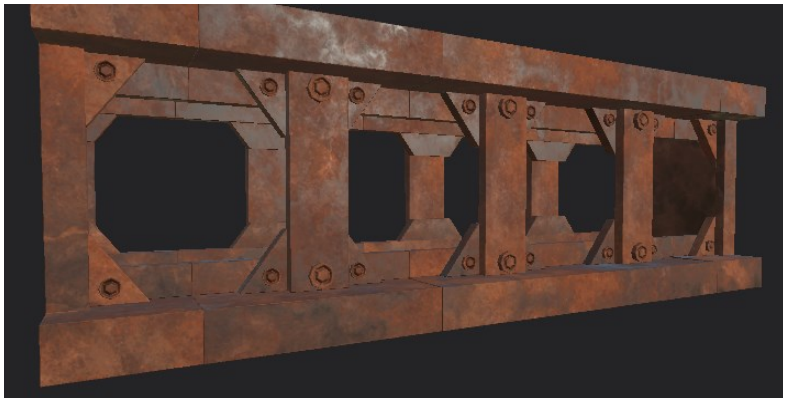
Modelado

Primero se modeló el extremo de la viga, con forma de C y se añadieron los detalles para crear el hueco hexagonal que se había diseñado. También se añadieron cilindros para crear los tornillos. Finalmente se combinó todo el modelo.

Texturizado

Se escogieron dos tonos de marrón y se combinaron utilizando una máscara. Todo el modelo es metálico, con unas partes más desgastadas que otras. Para los detalles de los tornillos se creó una capa con un *height map*.

Para poder utilizar este material en otros modelos se creó un *smart material* que facilitaba el trabajo posterior.



Texturizado del modelo.

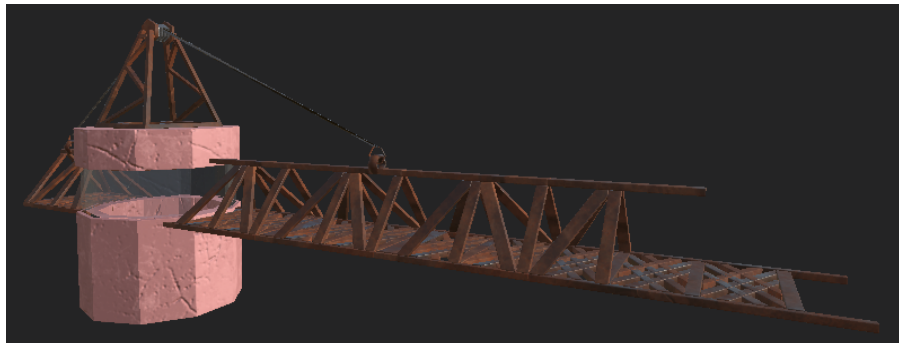
Grúa

Modelado

Se dividió el modelo en diferentes partes. La cabina y la cristalera tuvieron que ser separadas para que la cristalera no estuviera flotando debido al material utilizado. A la cabina se le añadieron las vigas de la grúa por las que Ío cruza, que se separaron en dos versiones: parte central y extremo, para poder añadir tantas partes centrales como fueran necesarias.

Texturizado

Se añadió el *smart material* creado para las paredes de las casas a la cabina y el *smart material* de las vigas para la parte de la grúa. Para el cristal se creó un material translucido con color y arañazos En el caso del cable se añadió una textura de cuerda en el *height map*.



Texturizado del modelo.

Programación

Personajes

Ío

Una de las primeras cosas a las que nos enfrentamos fue crear la **máquina de estado** y la **blueprint de personaje** de Ío desde cero, para establecer la lógica de movimiento para las animaciones. Esto requirió establecer los inputs y **condiciones** de cada una de las acciones para hacer entender a la máquina de estado que animaciones se podían reproducir.

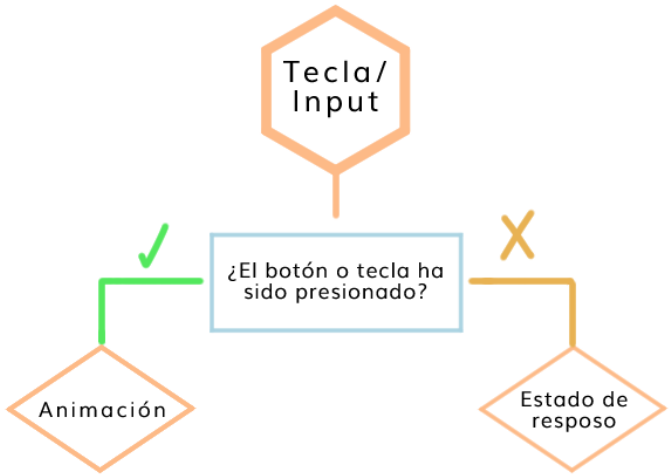


Diagrama de los inputs.

Acciones	*Descripción*
Movimiento	El personaje se desplaza en tres dimensiones. Hacia adelante y hacia atrás.
Salto	El salto está hecho para que Ío se desplace hacia arriba.
Doble salto	Permite aumentar el alcance del salto.
Ataque	Animación que bloquea el movimiento de avance del personaje y permite atacar a los enemigos.

Una de las cosas que queríamos establecer desde el inicio fue un doble salto dinámico que demostrara la personalidad de Ío. Para poder realizar esto se alteraron animaciones y se buscó un sistema que permitiera alterar la cantidad de saltos desde el blueprint con una variable que contara sus saltos, como se ve en el siguiente diagrama.

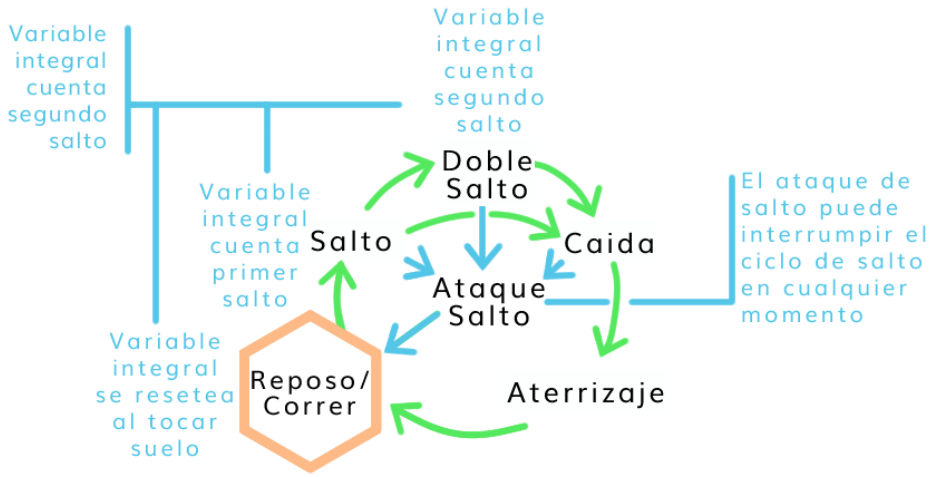


Diagrama de salto.

Algunas de las ideas que no llegaron a darse fue agregar una defensa, esto resulto ser muy complicado debido a que no encontramos recursos que nos ayudaran a programarlo y quedó en el aire hasta que se realizó el primer testing y uno de los jugadores dijo que era incómodo saltar si no ibas a poder atacar al final, lo que nos llevó a reutilizar la animación de la defensa para el final del salto.

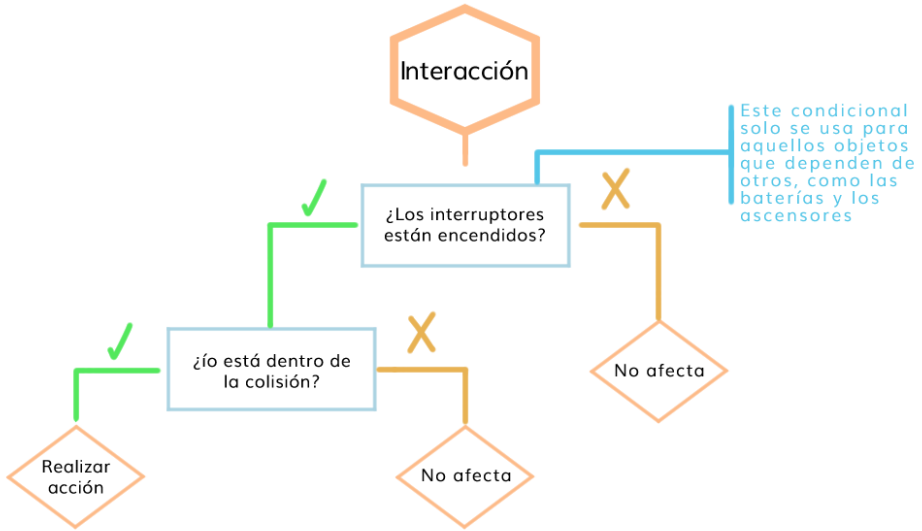


Diagrama de interacción.

Las interacciones de Ío son mediante un botón que activa lo objetos, como hemos visto en el Diagrama de Inputs, pero esto está atado a que esté en rango. Aun así, nuestros retos estaban ceñidos a que hubiera interruptores apagados, por lo que con ciertos blueprints había un condicional a mayores que impidiera usarlos.

Mantokóras

Para la programación de la Inteligencia Artificial nos enfocamos en la persecución de los enemigos.

Una de las cosas más complejas a la hora de la programación de la Mantakóra fue lograr que anduvieran por el terreno de manera normal hasta reconocer a Ío para luego perseguirla y atacarla.

La programación de la Matokóra incluye un nodo que permite ver al jugador, pero a su vez tiene varias variables que establecen si debe perseguir al jugador, el rango de espacio en el que debe perseguirlo y los eventos de vida y daño.

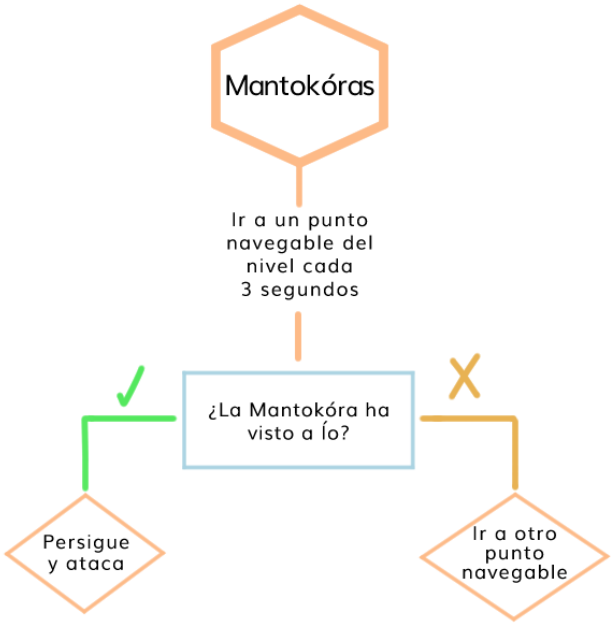


Diagrama de persecución.

Para la recepción de daño, queríamos asegurarnos de que Ío les quitara vida cuando los atacara y no solo con chocar contra ellos, por lo que creamos un condicional que tomara en cuenta si la animación de Ío se estaba reproduciendo antes de restarles vida.

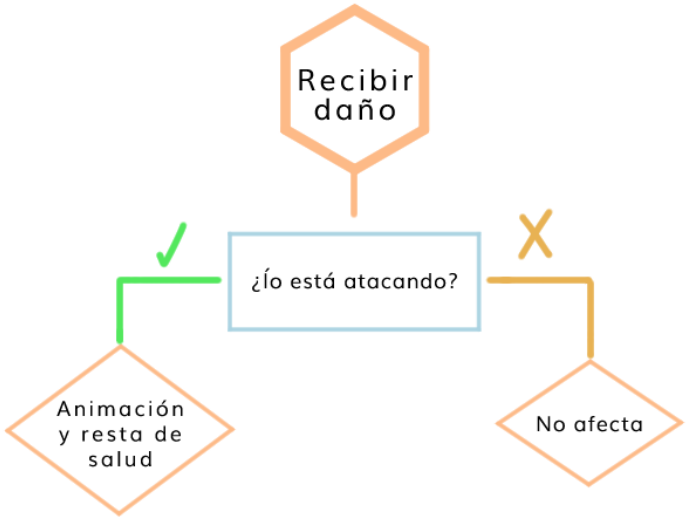


Diagrama de ataque.

Acciones	*Descripción*
Movimiento	El personaje se desplaza en dos dimensiones. Hacia arriba y abajo y hacia derecha e izquierda. Además de que está programada para seguir a Ío cuando la
Ataque	Animación que bloquea el movimiento de avance del

La importación del esqueleto en Unreal presentó sus propios problemas. Debido a la forma del esqueleto, nos encontramos con que la capsula de colisión de la blueprint de personaje no se adapta correctamente a la forma del modelo. Para corregir esto se agregaron 4 ***sockets*** en cada una de las rodillas del esqueleto y se agregaron **colisiones de esfera** que se adaptaran al largo de la pata que se utilizaría para el combate. Estas colisiones facilitaron el uso del evento ***begin overlap*** para establecer el sistema de salud.

Acciones	*Descripción*
Rotación	El boss no se desplaza, simplemente rota desde uno de sus *joints* siempre en siguiendo al juga-
Ataque	Ataca desde el centro y tiene tiempo de recar-
Punto débil	Una vez es atacado en sus patas una cierta cantidad de veces el enemigo despliega su pun-

La base del sistema de salud de Karkínus se centra en que sus puntos débiles fueran sus cuatro patas. Inicialmente una de las ideas fue que las patas tuvieran su propio sistema de salud que afectara a su vez a una variable de salud general, pero esto probó ser más problemático a la hora de mantener las cuentas de vida de Karkínus.

Una de las cosas más complicadas de programar fue el sistema de ataque, debido a que tomamos inspiración de juegos de PlayStation 2 en los que hay un claro ciclo a la hora de pelear. El objetivo era asegurarnos de que hubiera que golpear el punto débil cuatro veces, y solo lo podías golpear cuando habías roto una de las patas, lo que dio muchos problemas a la hora de programar las diferentes etapas de la pelea y asegurarnos de que todo funcionara correctamente.

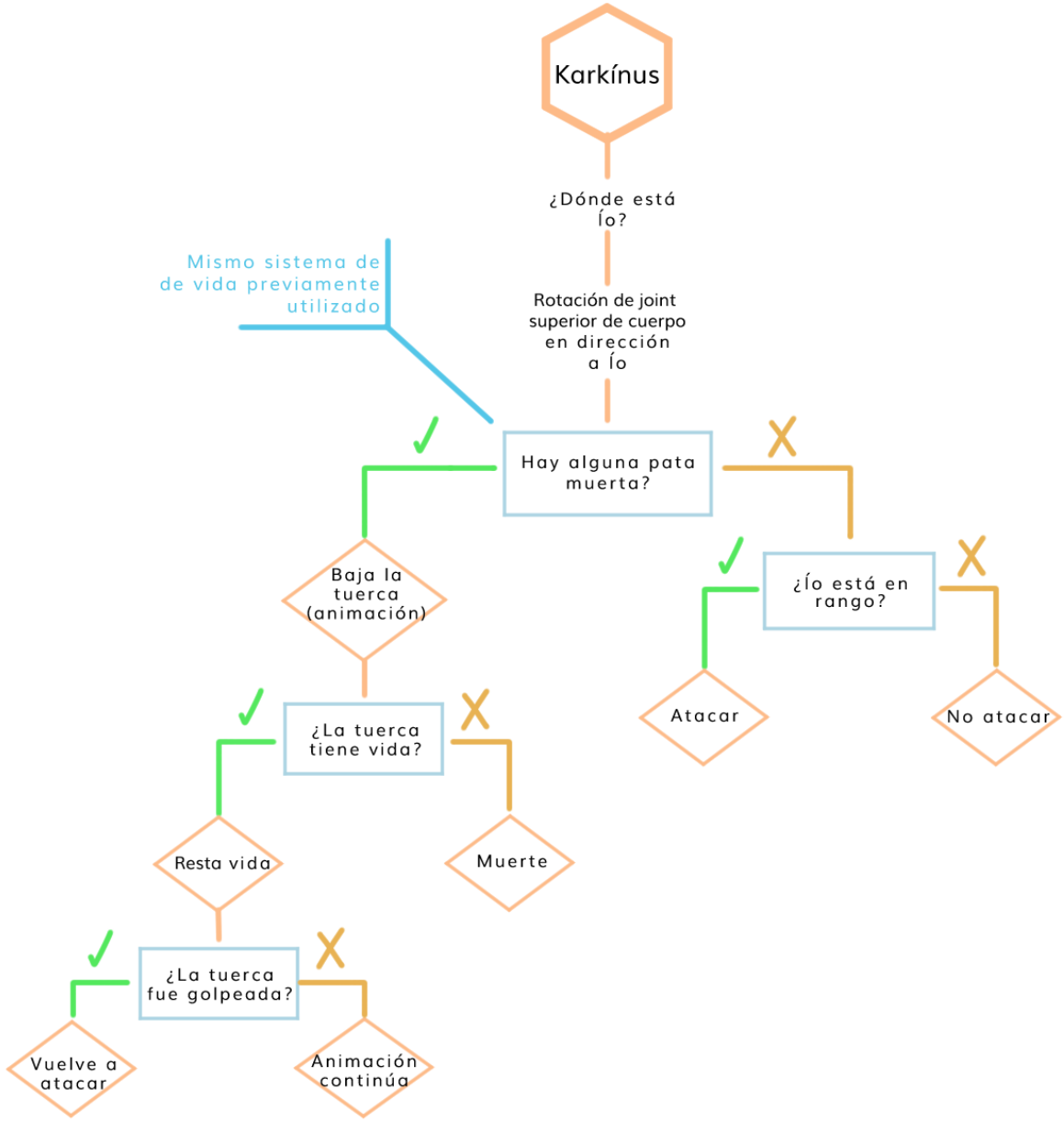


Diagrama de batalla.

Props

Méros

La programación del Méros se centró en agregar un ***socket*** para emparentar un ***collision volume*** al extremo y utilizarlo para causar daño con las condiciones de que Ío debía primero estar utilizando su animación de ataque. Además de eso establecimos un sistema de azar en el que el daño de Ío varía entre 20 y 50.

Para este asset se creó un **sistema de partículas** específico que desprendiera formas hexagonales para seguir fácilmente el camino de Ío en pantalla y reforzar la idea de que es una fuente de energía.

Acciones	*Descripción*
Movimiento	Atado a un socket y sigue las animaciones de Ío.
Ataque	Programado para solo causar daño si se dispara una animación.

Jarrones

La programación de los jarrones es simple en su funcionamiento, pero comenzó a dar problemas a la hora de colocarlos en el nivel debido a que las Mantokóras también eran capaces de romperlos y agarrar monedas. Además, cada pieza en la que rompía el jarrón contaba como moneda (esto último fue solventado).

Acciones	*Descripción*
Destrucción	El objeto se destruye al contacto con Ío.
Spawn	La destrucción causa un *spawn* de vida.

Monedas

Acciones	*Descripción*
Dar vida	Revisa la vida del jugador y le suma 50, luego lo almacena en el Game Instance, dónde se guarda la información del jugador en una va-

Vida

Acciones	*Descripción*
Sumar monedas	Toma la variable de monedas del Game Instance, le suma uno y luego lo almacena de nuevo en el Game Instance, dónde se guarda la información del jugador en una variable para las monedas.

Hongos

Para los hongos se crearon **máquinas de estado** y **blueprints de personaje**. Esto se debió a lo simple que es controlar las animaciones desde el blueprint una vez establecidas las condiciones. De esta manera un solo evento fue necesitado para cambiar la animación y activar el impulso de la plataforma.

Acciones	*Descripción*
Activación	Programados para que cuando Ío salte encima de ellos la impulsen por encima de lo que sus saltos le permiten.

Ascensor

Acciones	*Descripción*
Movimiento	Movimiento de subida y bajada utilizando la tecla E para interactuar. Sólo se podrá activar una vez que haya llegado al punto de inicio o de final. Durante el trayecto el jugador no podrá cambiar la dirección. La comunicación entre blueprints permite que sea activado y se inicie el movimiento solo si sus interruptores están encendidos. Esto se consigue mediante variables booleanas establecidas en el blueprint del nivel para que los interruptores correspondientes activen el ascensor.

Interruptor

Acciones	*Descripción*
Activación	Permite que el ascensor al que están conectados sea activado. Como parte de la interfaz visual, está programado para que cambie la luz que proyecta para mostrarle al jugador que está encendido, además de tener programado un efecto de sonido de confirmación.

Baterías

Acciones	*Descripción*
Activación	Permiten la rotación de las torres y de objetos que impiden el paso, como en el caso de la zona final de la demo. Se activan de la misma forma que los interruptores.

Mediante la **comunicación entre los blueprints** de los props y creamos las torres a las que se enfrenta Ío a lo largo del juego. Esto ocurrió principalmente mediante el uso de la **blueprint de nivel**, en la que utilizamos diferentes variables de tipo ***boolean*** para señalar el orden de los eventos en todo momento.

El funcionamiento de los ascensores en base a los interruptores dio problemas debido a las acciones del *input* de Ío. Cuando está cerca del objeto y se aprieta el botón sin ninguna animación los botones funcionan, y si aparece la animación, el botón no va. En este caso descubrimos un bug que permite que, saliendo del espacio del botón, interactuando y volviendo a intentarlo se puede seguir el juego.

Para la segunda torre no hubo problema con la comunicación de blueprints debido a que no hacía falta para el funcionamiento del ascensor. En este caso la comunicación del blueprint fue para rotar la torre en la dirección correcta para continuar.

Para los diferentes menús se ideó una estética simple, pero que encajara con la demo realizada.

En las páginas siguientes se mostrará el resultado final de cada uno de ellos y se explicará el funcionamiento de su programación. Cabe destacar que todas las pantallas estaban pensadas para adaptarse a las diferentes resoluciones de los monitores, de forma que los objetos colocados no se deformen. Existe una pantalla para cambiar la resolución de la ventana de juego; sin embargo, pese a funcionar dentro del motor de Unreal, no conseguimos que en la exportación respondieran los botones correctamente.

Además, se agregó una pantalla de carga entre algunas de las pantallas para que no saltasen tan bruscamente de unas a otras, y una pantalla final al terminar el juego para poder volver al menú principal.

Menú principal

Es la primera pantalla que el jugador ve al empezar el juego. Dicho menú está colocado en un nivel aparte, con el tema correspondiente de la banda sonora creada para el juego. Cuenta con sonidos de confirmación al seleccionar los botones y con tres versiones para la apariencia de los mismos: el texto normal, la forma de un hexágono al lado para indicar la opción sobre la que se encuentra el jugador, y el mismo hexágono rellenado de color para acompañar al sonido que confirma la selección.

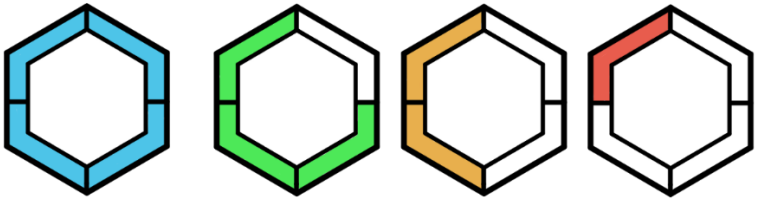
Evento	*Descripción*
Nueva Partida	Borra los datos guardados, si los hay, e inicia el juego.
Cargar Partida	Carga los datos almacenados del juego y coloca al jugador desde la última localización guardada, con la cantidad de vida y monedas guardadas en ese punto.
Opciones	Permitiría ajustar la resolución de la pantalla eligiendo entre las cinco opciones posibles.
Salir	Salir del juego.

Menú principal.

Dentro del juego creamos una interfaz simple para evitar llenar la pantalla del jugador y obstruir la vista. En la parte superior el jugador podrá consultar la cantidad de vida de Ío, así como las monedas que le quedan hasta llegar a las cincuenta que hay escondidas por el nivel.

Para el indicador de salud optamos por crear un hexágono que respondiera a la cantidad de vida de Ío. Para ello creamos un material en Unreal usando una máscara para darle la forma hexagonal y un *angle gradient* (tipo de degradado con forma circular) para que la salud disminuyera en el sentido de las agujas del reloj. A continuación, programamos la interfaz para que la textura actuara como barra de progreso. Por último, decidimos añadir cuatro variables para indicar al jugador la cantidad de vida de Ío con cuatro colores diferentes. De esta forma si la salud era menor a un número concreto, el color del hexágono cambiaría.

Interfaz del juego.



Variantes de la salud.

En cuanto al contador de monedas, cada vez que Ío recoge una de las cincuenta monedas repartidas por el nivel, se suma y guarda el valor. Mediante un texto en la interfaz, pudimos indicarle al jugador la cantidad de monedas recogidas. Sin embargo, cuando el jugador carga la partida, las monedas del nivel hacen *respawn* permitiendo al jugador recoger más de cincuenta monedas. Esto se debe a que la base de datos donde se guarda el juego solo guarda los datos de Ío.

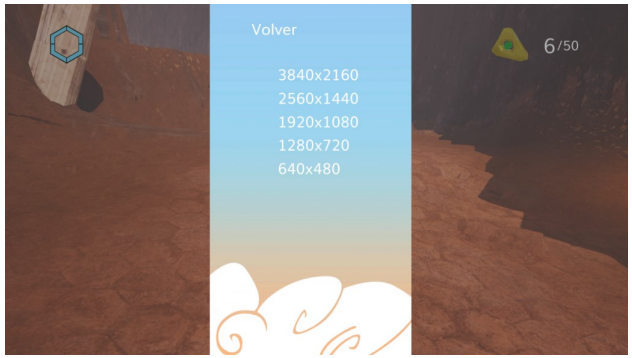
Además, añadimos volúmenes para darle al jugador ayudas con los controles y un contexto del funcionamiento de las torres.

Menú de pausa

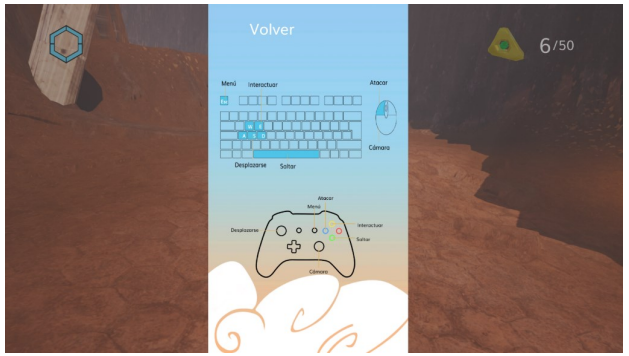
Cuando el jugador pulsa el botón de pausa (Esc), el juego queda inhabilitado mientras no seleccione la opción “Volver”. Esta pantalla cuenta con la misma estética y los mismos indicadores de confirmación que el menú principal.

Evento	*Descripción*
Volver	Vuelve al juego restaurando los controles establecidos.
Opciones	Permitiría ajustar la resolución de la pantalla desde dentro del juego.
Controles	Pantalla de consulta con imágenes de las dos opciones de controles disponibles.
Salir	Salir al menú principal.

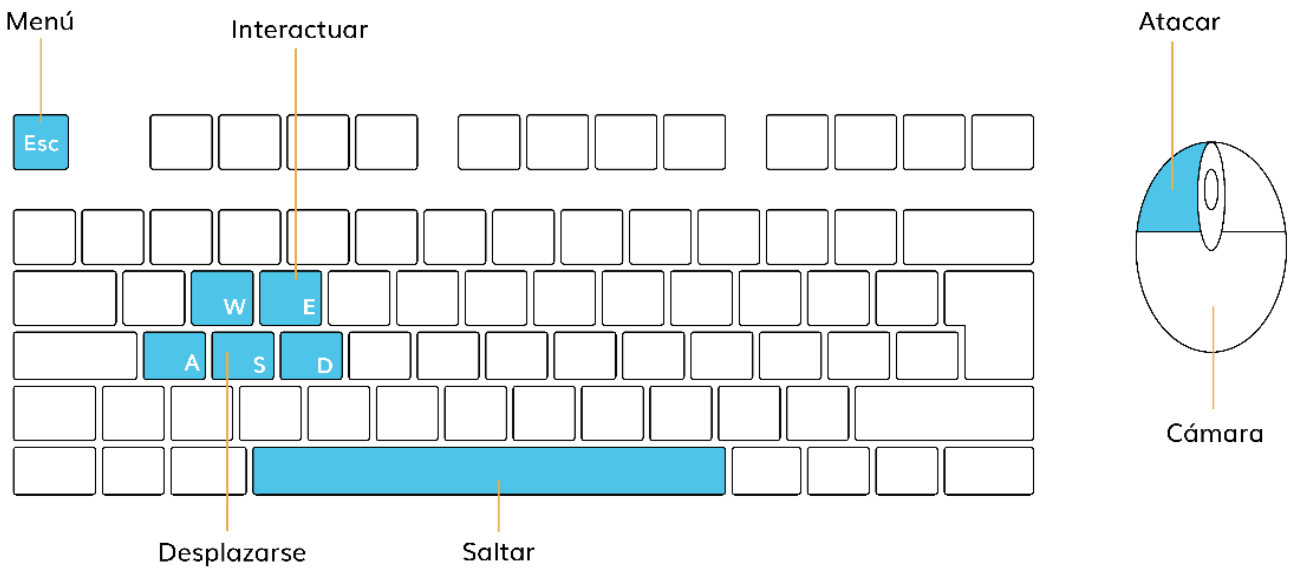
Menú de pausa.



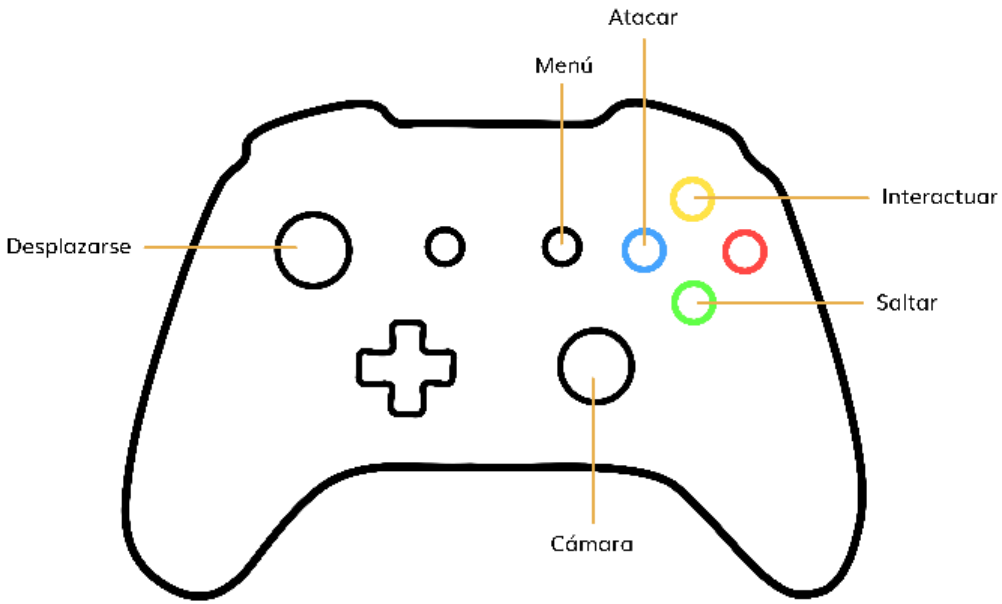
Pantalla de opciones.



Pantalla de controles.



Controles del teclado.



Controles del Gamepad.

Ambientación sonora

Las pistas utilizadas fueron buscadas en bibliotecas de sonido y editadas utilizando Audacity para cambiarlas y adaptarlas al mundo de Nēsos , a excepción de los sonidos de Ío.

Ío

Los sonidos de Ío fueron buscados en bibliotecas de sonido y grabados dependiendo de lo factible que fuera conseguir hacerlo nosotras para darle un toque auténtico y único. La voz de Ío fue personalmente grabada por miembros del equipo, incluyendo así sonidos de esfuerzo cuando salta, recibe daño o muere. Los pasos y sonidos de combate fueron obtenidos de las bibliotecas.

Los sonidos de interacción con los entornos se basaron mucho en que fueran sonidos divertidos y poco convencionales.

Mantokóras

El sonido es principalmente metálico, debido a su diseño, y juvenil, inspirado por los juegos de PlayStation2. Por un lado, se aplicó un sonido de máquina para dejar intuir al jugador su funcionamiento; por otro, el sonido siseante al atacar realza su aspecto y recalca su diseño similar al de un escorpión.

Karkínus

Se utilizaron sonidos metálicos como en las Mantokóras. Los sonidos son los ya aplicados de combate y una explosión final para cuando es derrotado.

Méros

Los sonidos del Méros se hicieron directamente en su blueprint, tomando en cuenta la programación de Ío y si el volumen de colisión del propio Méros tuvo contacto con alguno de los dos blueprints de los enemigos. De esta manera, nos aseguramos de que el sonido de choque ocurriera solo cuando Ío realmente golpeaba algo, y que sonara diferente cuando golpeaba al aire.

Otros sonidos

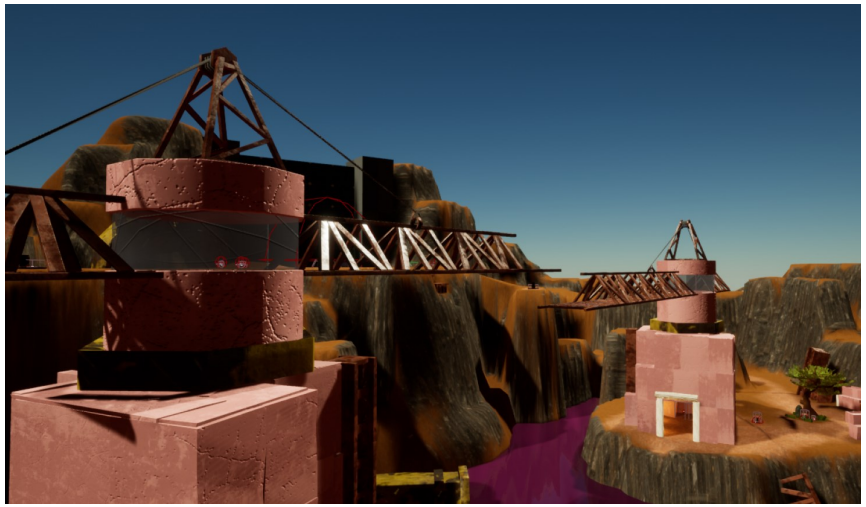
Además se agregaron otros sonidos para los jarrones al romperse, la salud y las monedas al ser recogidas, el hongo cuando Ío salta sobre él y todas las máquinas con las que Ío puede interactuar.

Banda sonora

Para la banda sonora se realizó una colaboración externa con Fernando Fernández Rouco, quien compuso las tres piezas del juego. Previamente se realizó un documento de pitch para el compositor, de manera que supiera los temas del juego, las influencias y el ambiente en el que se desarrolla el juego.

Iluminación

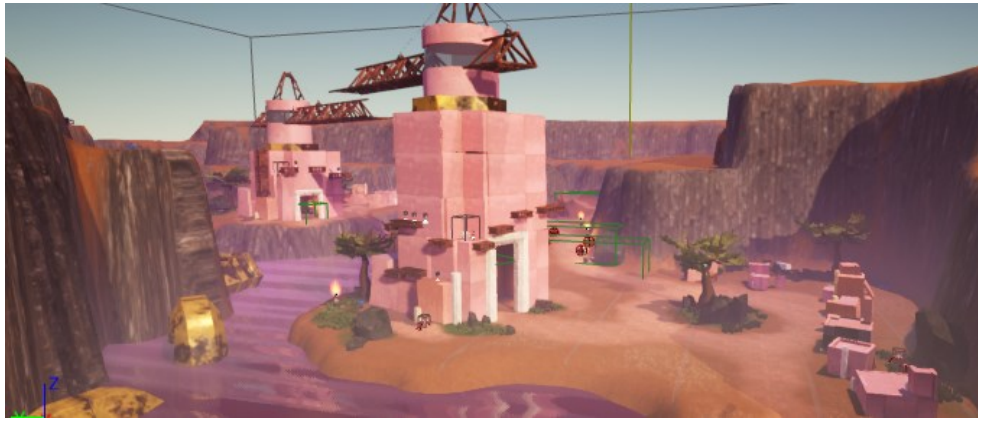
Entre las ultimas cosas que trabajamos está la iluminación. Para esto buscamos un asset para que arreglara la iluminación general en el nivel, resolviendo las sombras duras que había en un principio, y luego cambiamos los parámetros de la niebla, el *skydome* y la luz para que el color estuviera más acorde con Nēsos y dieran más atmosfera.



Antes de la iluminación final.

Resultado final.

Antes de la iluminación final.



Resultado final.

Testing

Primera sesión de testing

En este momento el Karkínus aún no estaba programado y el terreno seguía en procesos de cambio. Debido a esto muchas de los problemas con los que nos encontramos eran que los jugadores no sabían si estaban interactuando con el mundo, por lo tanto, muchas de las soluciones fueron agregar sonidos o efectos visuales que los ayudaran.

Problemas

Los jugadores no sabían qué hacer a lo largo del nivel. Para guiarlos agregamos assets de **color blanco** para marcar el camino y **mensajes en la HUD** que los pusieran en contexto

Los jugadores no sabían si los **interruptores** que estaban utilizando estaban funcionando, así que agregamos **sonidos de confirmación** de las acciones.

Es incómodo **no poder atacar cuando saltas**. Se reutilizó la animación para un intento de defensa y se agregó al final del salto para dar mayor rango de ataque.

Segunda sesión de testing

Una de las diferencias claves que notamos a la hora de realizar el segundo testing eran los pequeñas fallos de programación, muchas de las cuales no pudieron ser solventadas, pero que no quedaron fuera de nuestro conocimiento.

Problemas

Si **mueres en el agua** el sistema de guardado automático no te coloca en una zona segura, por lo que crea un ***loop* de muerte**. Por ello el agua decidimos dejarla con colisión, lo que provoca que tanto Ío como las Mantokóras caminen sobre ella.

El **ángulo de ataque de Karkínus** es muy alto y no te puede dar si te acercas a las patas.

Los botones de la primera torre no funcionan constantemente. Si la animación de interacción de Ío funciona, en botón no funciona.

No se puede usar el **gamepad en los menús** y es incómodo para el jugador.

Conclusiones

Hemos logrado **poner a prueba nuestros conocimientos** y explorar nuevas posibilidades dentro del mundo de la producción 3D, además de profundizar más a fondo en los trabajos que nos interesan a cada una, ya sea teniendo el primer contacto con nuevos programas o explorando nuevas maneras de realizar los procesos que hemos aprendido en la carrera.

Uno de los mayores retos para nosotras fue enfrentarnos a la creación tangible de un medio tan complejo como un videojuego y a la necesidad de **reaprender a comunicar**. Ya solo el hecho de interactuar en un mundo implica muchas cosas, pero no solo se trata de participar, sino de obtener información a cambio, y con la posibilidad de errar en el proceso. Todos estos aspectos son fáciles de decir, pero difíciles de aplicar a la hora de crear algo teniendo un público en mente.

Desarrollar el universo de Nēsos fue un reto, en específico poner límites a la visión creativa inicial haciendo recortes de contenido hasta lograr crear un mundo cohesivo y jugable. Conseguimos muchos **logros en el proceso**, en especial en la parte de producción, a pesar de que hay muchas cosas que nos gustaría agregar a mayores, creamos personajes únicos con animaciones, texturas, y mecánicas que creemos alcanzaron las expectativas que teníamos en un primer momento.

A pesar de todas las complicaciones en el camino, la sensación de conseguir que un enemigo reaccione al jugador o que una puerta se abra es muy gratificante, y es en buena parte lo que nos mantuvo **motivadas** a crear un buen producto. Tuvimos la oportunidad de crear un producto nuevo mientras aprendíamos en el proceso, diseñando un universo original y e interactivo del que nos sentimos satisfechas.

Bibliografía

- Todas las fotos de referencia fueron obtenidas de Unsplash.
- Cancer – Ventanas al Universo. (2020). Retrieved 31 August 2020, from https://www.windows2universe.org/mythology/cancer.html&lang=sp

Contenido descargado

- Unreal Engine. 2020. Advanced Village Pack By Advanced Asset Packs In Environments - UE4 Marketplace. [online] Retrieved 3 March 2020, from: https://www.unrealengine.com/marketplace/en-US/product/advanced-village-pack
- Unreal Engine. 2020. Industry Props Pack 6 By Silvertm In Props - UE4 Marketplace. [online] Retrieved 6 March 2020, from: https://www.unrealengine.com/marketplace/en-US/product/3e2a3cb997cf47b1ab782a67957bfed0
- Unreal Engine YouTube Channel 2020. Getting Started with Landscapes | Live Training | Unreal Engine. [online] Retrieved 23 July 2020, from: https://www.youtube.com/watch?v=gMKjIZMPJ0Q
- Textures.com. 2020. Red Hot Steel - PBR0221. [online] Retrieved 11 July 2020, from: https://www.textures.com/download/pbr0221/133269
- Textures.com. 2020. Basalt Cliff - PBR0488. [online] Retrieved 11 July 2020, from: https://www.textures.com/download/pbr0488/138497
- Textures.com. 2020. Soil 6 - 3Dscans0628. [online] Retrieved 11 July 2020, from: https://www.textures.com/download/3dscans0628/138396
- Textures.com. 2020. Hexagon Paver - PBR0601. [online] Retrieved 11 July 2020, from: https://www.textures.com/download/pbr0601/139027?q=hexagon

Tutoriales

- Creating Games For Beginners Using UE4 - Unreal Engine 4 Course. (2020). Retrieved 17 September 2020, from https://www.youtube.com/playlist?list=PLL0cLF8gjBpqDdMoeid6Vl5roMl6xJQGC
- Creating A Role Playing Game - Unreal Engine 4 Course. (2020). Retrieved 17 September 2020, from https://www.youtube.com/playlist?list=PLL0cLF8gjBpqA8DcrhL_O9kD4jsUqhDR6
- User Interface Development - Unreal Engine 4 Course. (2020). Retrieved 17 September 2020, from https://www.youtube.com/playlist?list=PLL0cLF8gjBprIHm0yo-Vj9oBwi2-gAIEd

Sonidos descargados

- Jump Landing Sound. (2012). Retrieved 17 September 2020, from https://opengameart.org/content/jump-landing-sound
- Battle Sound Effects. (2010). Retrieved 17 September 2020, from https://opengameart.org/content/battle-sound-effects
- 3 Melee sounds. (2009). Retrieved 17 September 2020, from https://opengameart.org/content/3-melee-sounds
- Jump Landing Sound. (2012). Retrieved 17 September 2020, from https://opengameart.org/content/jump-landing-sound
- Pickup_Gold_00.wav by LittleRobotSoundFactory. (2020). Retrieved 17 September 2020, from https://freesound.org/people/LittleRobotSoundFactory/sounds/270408/
- 08935 game gold bonus.wav by Robinhood76. (2020). Retrieved 17 September 2020, from https://freesound.org/people/Robinhood76/sounds/518733/
- Huge Explosion by florianreichelt. (2020). Retrieved 17 September 2020, from https://freesound.org/people/florianreichelt/sounds/459973/
- Oidz Magnet, A (H4n).wav by InspectorJ. (2020). Retrieved 17 September 2020, from https://freesound.org/people/InspectorJ/sounds/411163/
- Cute Cartoon Jump Sound Effect. (2012). Retrieved 17 September 2020, from https://opengameart.org/content/cute-cartoon-jump-sound-effect
- success_bell by MattLeschuck. (2020). Retrieved 17 September 2020, from https://freesound.org/people/MattLeschuck/sounds/511484/
- Robotic mechanic step sounds. (2009). Retrieved 17 September 2020, from https://opengameart.org/content/robotic-mechanic-step-sounds
- Huge Explosion by florianreichelt. (2020). Retrieved 17 September 2020, from https://freesound.org/people/florianreichelt/sounds/459973/
- shot.flac by qubodup. (2020). Retrieved 17 September 2020, from https://freesound.org/people/qubodup/sounds/188341/
- Elevator by marcel_farres. (2020). Retrieved 17 September 2020, from https://freesound.org/people/marcel_farres/sounds/186104/
- success_bell by MattLeschuck. (2020). Retrieved 17 September 2020, from https://freesound.org/people/MattLeschuck/sounds/511484/
- 10_Turn.wav by 14GPanskaZackovaM. (2020). Retrieved 17 September 2020, from https://freesound.org/people/14GPanskaZackovaM/sounds/422124/
- Glass Break - Medium Jar by RoganMcDougald. (2020). Retrieved 17 September 2020, from https://freesound.org/people/RoganMcDougald/sounds/260434/