UNIVERSIDAD CATÓLICA DE CUENCA

UNIVERSIDAD CATÓLICA DE CUENCA

Comunidad Educativa al Servicio del Pueblo

FACULTAD DE INFORMÁTICA, CIENCIAS DE LA COMPUTACIÓN E INNOVACIÓN TECNOLÓGICA

CARRERA DE REALIDAD VIRTUAL Y VIDEOJUEGOS

Desarrollo de modelos 3D *low-poly* y sistemas de partículas optimizados para el videojuego Qhapac Ñan: El camino del Chasqui

PROYECTO DE INTEGRACIÓN CURRICULAR PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN REALIDAD VIRTUAL Y VIDEOJUEGOS

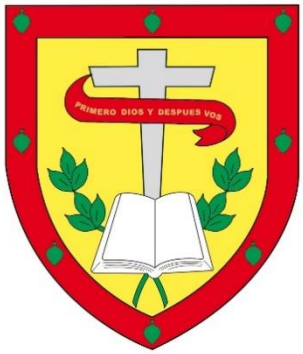
AUTOR: CRISTHIAN GABRIEL JIMÉNEZ MORA

DIRECTORA: DIS. TATIANA ALEXANDRA CABRERA SILVA, MSC.

CUENCA- ECUADOR

2026

DIOS, PATRIA, CULTURA Y DESARROLLO



UNIVERSIDAD CATÓLICA DE CUENCA

Comunidad Educativa al Servicio del Pueblo

FACULTAD DE INFORMÁTICA, CIENCIAS DE LA COMPUTACIÓN E INNOVACIÓN TECNOLÓGICA

CARRERA DE REALIDAD VIRTUAL Y VIDEOJUEGOS

Desarrollo de modelos 3D *low-poly* y sistemas de partículas optimizados para el videojuego Qhapac Ñan: El camino del Chasqui

PROYECTO DE INTEGRACIÓN CURRICULAR PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN REALIDAD VIRTUAL Y VIDEOJUEGOS

AUTOR: CRISTHIAN GABRIEL JIMÉNEZ MORA

DIRECTORA: DIS. TATIANA ALEXANDRA CABRERA SILVA, MSC.

CUENCA- ECUADOR

2026

DIOS, PATRIA, CULTURA Y DESARROLLO

Declaratoria de Autoría y Responsabilidad

Cristhian Gabriel Jiménez Mora portador de la cédula de ciudadanía Nº **0706303989**. Declaro ser el autor de la obra: "**Desarrollo de modelos 3D low-poly y sistemas de partículas optimizados para el videojuego Qhapac Ñan: El camino del Chasqui"**, sobre la cual me hago responsable sobre las opiniones, versiones e ideas expresadas. Declaro que la misma ha sido elaborada respetando los derechos de propiedad intelectual de terceros y eximo a la Universidad Católica de Cuenca sobre cualquier reclamación que pudiera existir al respecto. Declaro finalmente que mi obra ha sido realizada cumpliendo con todos los requisitos legales, éticos y bioéticos de investigación, que la misma no incumple con la normativa nacional e internacional en el área específica de investigación, sobre la que también me responsabilizo y eximo a la Universidad Católica de Cuenca de toda reclamación al respecto.

Cuenca, **13 de septiembre del 2026**

F: .....

Cristhian Gabriel Jiménez Mora

C.I. 0706303989

Certificado

Certifico que el presente trabajo de investigación fue desarrollado por **Cristhian Gabriel Jiménez Mora,** portador(a) de la cedula de ciudadanía **N.º 0706303989** con el tema "Desarrollo de modelos 3D low-poly y sistemas de partículas optimizados para el videojuego Qhapac Ñan: El camino del Chasqui", bajo mi supervisión.

MSc. Tatiana Alexandra Cabrera Silva

DIRECTORA DEL PROYECTO DE TITULACIÓN

UNIVERSIDAD CATÓLICA DE CUENCA

Dedicatoria

A mi familia, por ser mi pilar fundamental y mi mayor soporte durante todas las horas invertidas frente a la pantalla perfeccionando cada modelo, topología y textura. Por su paciencia incondicional en los momentos de mayor exigencia y por creer siempre en mi pasión por el desarrollo de videojuegos y el arte técnico.

A todos aquellos que alguna vez me inspiraron a adentrarme en la creación de mundos virtuales y a transformar la imaginación en experiencias interactivas.

Agradecimiento

A la Universidad Católica de Cuenca y a mis docentes de la carrera de Realidad Virtual y Videojuegos, por brindarme las herramientas y el conocimiento necesario para materializar mi vocación profesional.

Un agradecimiento especial a mi directora de tesis, Dis. Tatiana Alexandra Cabrera Silva, Msc., por su guía invaluable, disposición y visión crítica, las cuales fueron fundamentales para pulir el desarrollo técnico y visual de este proyecto.

A Dante Gaibor, por su constante colaboración en el diseño y la narrativa dentro de nuestros proyectos de desarrollo, cuyo trabajo en equipo siempre ha sido un impulso para alcanzar mejores resultados creativos.

A mi compañero Adrián, por el apoyo mutuo y el intercambio de ideas durante el proceso de investigación y estructuración documental de nuestras tesis; su compañerismo aligeró la carga de esta etapa académica.

Finalmente, a mis amigos, con quienes he compartido tantas partidas, ideas creativas y *game jams*, por ser mi fuente de inspiración constante y acompañarme a lo largo de este camino formativo.

Resumen

El desarrollo de videojuegos para computadoras de bajo rendimiento normalmente enfrenta problemas de latencia por el alto consumo de recursos gráficos innecesarios. Para solucionar esto, el proyecto detalla la creación de modelos 3D y efectos visuales optimizados para el videojuego educativo "Qhapac Ñan: El Camino del Chasqui". Utilizando Unity, se aplicó una metodología mixta. Se diseñó el entorno, la fauna y el protagonista con una estética *low-poly* y *flat shading*, mientras que los poderes divinos se representaron mediante sistemas de partículas de bajo costo computacional. Tras someter el proyecto a pruebas de estrés, los resultados demostraron que, en una primera versión, el número de *batches* rondaba un promedio de 4.630, logrando un alto contraste con las versiones optimizadas, las cuales alcanzaron un promedio de solo 730 *batches*. Además, las pruebas cualitativas con usuarios revelaron que los modelos 3D en conjunto con los VFX y los sonidos, permitieron una clara comprensión del entorno y de todos sus elementos. En conclusión, esta optimización gráfica permite una experiencia fluida, demostrando que se pueden desarrollar videojuegos visualmente atractivos y con identidad propia sin sacrificar el rendimiento.

***Palabras Clave**: Optimización gráfica, Low-poly, Sistemas de partículas, Unity, Videojuegos educativos*

Abstract

Video game development for low-end computers typically faces latency issues due to the excessive use of unnecessary graphical resources. To address this, the project describes the creation of optimized 3D models and visual effects for the educational video game "Qhapac Ñan: El Camino del Chasqui." Using Unity, a hybrid methodology was applied. The environment, wildlife, and protagonist were designed with a low-poly aesthetic and flat shading, while divine powers were represented using computationally efficient particle systems. After subjecting the project to stress testing, the results showed that, in the initial version, the number of *batches* averaged around 4,630—a stark contrast to the optimized versions, which averaged only 730 *batches*. Furthermore, qualitative user testing revealed that the 3D models, in conjunction with the VFX and sound effects, enabled a clear understanding of the environment and all its elements. In conclusion, this graphical optimization enables a smooth experience, demonstrating that it is possible to develop visually appealing video games with a unique identity without sacrificing performance.

Keywords*: Graphical optimization, Low-poly, Particle systems, Unity, Educational video games.*

Índice de contenido

Índice de tablas

Índice de figuras

Introducción

El desarrollo contemporáneo de videojuegos enfrenta una dicotomía constante entre la búsqueda de una alta fidelidad visual y las capacidades de procesamiento del hardware destino. Mientras la industria impulsa entornos interactivos con tecnologías de última generación, este costo computacional margina a los usuarios e instituciones que operan con computadoras de bajo rendimiento. En este contexto surge "Qhapac Ñan: El Camino del Chasqui", un proyecto lúdico y pedagógico que busca difundir la cultura, la mitología y la geografía andina mediante una experiencia interactiva.

El principal desafío de este desarrollo radica en la rápida saturación de la Unidad Central de Procesamiento (CPU) y la Unidad de Procesamiento Gráfico (GPU) al renderizar modelos tridimensionales y efectos visuales (VFX) en equipos con recursos limitados. Una topología descontrolada y la superposición de elementos semitransparentes en los sistemas de partículas generan sobrecarga (*overdraw*) y caídas severas en la tasa de fotogramas, afectando directamente la jugabilidad e impidiendo el objetivo educativo del software.

Para dar solución a esta problemática, el presente trabajo de titulación propone la estructuración de un pipeline de arte técnico centrado en la optimización gráfica. La metodología abarca desde el modelado *low-poly* y el uso de *flat shading* para la fauna, personajes y biomas, hasta la implementación de *shaders* personalizados y sistemas de partículas de bajo costo computacional en el motor Unity, prescindiendo de iluminaciones dinámicas.

El presente documento detalla el desarrollo y la validación de la propuesta tecnológica y se encuentra estructurado de la siguiente manera:

En el Capítulo I, se expone el marco referencial, detallando el planteamiento del problema, los objetivos del proyecto y la justificación tecnológica y social de la optimización en entornos de hardware no especializado.

El Capítulo II abarca el marco teórico y tecnológico, fundamentando las bases conceptuales de la abstracción gráfica, la topología, los sistemas de partículas y la arquitectura del motor de renderizado.

El Capítulo III presenta el manual del usuario, documentando la experiencia de acceso, navegación y la comprensión de las mecánicas del videojuego desde la perspectiva del público objetivo.

El Capítulo IV desarrolla el manual técnico de implementación, donde se describe el flujo de trabajo de la dirección de arte, la producción de *assets* 3D, la creación de *shaders*, la integración de recursos en Unity y las técnicas de optimización aplicadas para reducir los lotes de renderizado (*batches*). Posterior, se exponen la validación, los resultados y la discusión, demostrando mediante métricas de rendimiento extraídas del *profiler* y evaluaciones cualitativas de usuario, el cumplimiento de los objetivos planteados y la viabilidad del proyecto.

Capítulo I

1. Marco Referencial

1.1. Contextualización del problema

En el desarrollo contemporáneo de videojuegos, existe una dicotomía constante entre la búsqueda de una alta fidelidad visual y las capacidades reales de procesamiento del hardware de destino. La industria del entretenimiento digital ha impulsado la creación de entornos cada vez más inmersivos y fotorrealistas [1], esta evolución estética conlleva un costo computacional significativo que no siempre es compatible con plataformas o dispositivos de gama de entrada o equipos no especializados.

En la industria actual, los juegos modernos han demostrado no hacer el énfasis adecuado a la optimización gráfica, esto debido a toda la tecnología de punta como la generación de fotogramas de NVIDIA. Esto desencadena en una mala práctica por parte de los estudios, ya que solo los usuarios con acceso a las últimas tecnologías lograrán disfrutar de una experiencia satisfactoria dentro de su videojuego.

Específicamente en los provectos orientados al sistema de educación público o privado, no se tiene el acceso necesario para computadoras con última tecnología o tipo *gamer*, el hardware no suele contar con tarjetas gráficas de última generación, lo cual limita las posibilidades de tener un aprendizaje más lúdico interactivo con tipo gamificación que contenga opciones visualmente atractivas que representen diferentes áreas del aprendizaje, entre ellas. el de cultura, flora, fauna, entre otros.

1.2. Planteamiento del problema

El problema principal radica en la rápida saturación de los recursos de la Unidad Central de Procesamiento (CPU) y la Unidad de Procesamiento Gráfico (GPU) cuando no se aplican restricciones arquitectónicas a los modelos tridimensionales y a los efectos visuales (VFX) desde la fase conceptual del diseño [2]. Si los modelos 3D del protagonista (Chaicó-chasqui), la fauna y el entorno se construyen con una topología descontrolada, el motor gráfico colapsa, resultando en caídas severas en la tasa de fotogramas por segundo (FPS).

Además, una mala estructura de la malla tridimensional no solo afecta el renderizado, sino que desencadena fallos críticos en la etapa de rigging y animación [3].

Entre los fallos más comunes generados por la errónea estructura de la malla, podemos encontrar: inconsistencia en los pesos de las normales, errónea deformación en las articulaciones de un modelo o la generación de un mapa de UVs (sistema de coordenadas utilizado para aplicar texturas 2D a modelos 3D) con coordenadas erróneas, entre otras.

Simultáneamente, la representación de poderes divinos (Inti, Quilla, Illapa, Pachacámac, Pachamama, Kon) requiere de sistemas de partículas. Simular estos eventos dentro del motor Unity impone cargas elevadas de procesamiento, donde el fenómeno de *overdraw* (sobre dibujado) derivado de la superposición masiva de elementos semitransparentes degrada exponencialmente la fluidez del software [4]. De no resolverse este conflicto entre diseño estético y rendimiento, el usuario percibe una experiencia inestable, frustrante y, en última instancia el videojuego fracasa en su misión lúdica y pedagógica.

1.3. Formulación del problema

Con base en la problemática expuesta, la investigación se rige bajo la siguiente interrogante: ¿De qué manera la estructuración de un *pipeline* técnico centrado en el modelado 3D *low-poly* y en la optimización de sistemas de partículas simples en Unity permite la coherencia visual y el rendimiento fluido del videojuego "Qhapac Ñan: el Camino del Chasqui" en hardware de capacidades limitadas?

1.4. Justificación

La viabilidad técnica de un videojuego en dispositivos de hardware no especializado exige la necesidad de establecer un control estricto sobre el renderizado de gráficos. La adopción de una estética *low-poly* combinada con un sombreado plano (*flat shading*) resulta efectiva y cognitivamente asimilable para el usuario en videojuegos de perspectiva isométrica. Esta técnica mejora la limpieza visual de los tableros tácticos y disminuye dramáticamente la carga de renderizado [5], permitiendo que recursos que normalmente se destinarían al cálculo de texturas complejas (como mapas de normales o especularidad) se reasignen a mantener la estabilidad del motor gráfico.

Desde una perspectiva pedagógica y cultural, el videojuego funciona como una "máquina de aprendizaje" [6]. Al interactuar con el camino del chasqui, los usuarios asimilan conceptos de la historia y geografía andina. En este escenario, los efectos visuales (VFX) no son meramente ornamentales, actúan como una retroalimentación que ayuda al jugador a comprender las consecuencias de sus decisiones dentro del tablero [7].

Los videojuegos traducen la historia a un lenguaje digerible para las nuevas generaciones, teniendo la capacidad de no solo leer sobre cómo era la agricultura en terrazas, si no, que pueden directamente experimentar desde la visión de un agricultor de la época. Se

genera un cambio, de la memorización de la enseñanza tradicional por las vivencias de una educación lúdica.

Finalmente, esta investigación se justifica por su aporte a la preservación del patrimonio cultural a través de medios digitales [8]. Al optimizar estos recursos para equipos de bajo rendimiento en instituciones públicas, se garantiza que la aplicación pueda llegar a un público objetivo amplio, capturando su atención mediante una identidad visual propia y atractiva.

1.5. Objetivo general

Desarrollar los modelos 3D y los efectos visuales (VFX) mediante técnicas de modelado *low-poly* y la implementación de sistemas de partículas simples en Unity, asegurando la coherencia visual y el rendimiento fluido del videojuego "Qhapac Ñan: El Camino del Chasqui" en hardware de bajo rendimiento.

1.6. Objetivos específicos

- Diseñar con modelado y texturas, al protagonista (Chaicó-chasquí) y la fauna (oso, cóndor, jaguar, serpiente, jabalí) representativa empleando un estilo *Low-Poly* Minimalista con *Flat Shading*, asegurando su legibilidad desde una perspectiva isométrica.
- Construir *assets* modulares 3D para la creación procedimental de las grillas hexagonales correspondientes a los tres biomas del juego (Costa, Amazonía, Sierra), optimizando la carga de materiales en el motor gráfico.
- Desarrollar sistemas de partículas de bajo coste computacional en el motor gráfico para representación los poderes de los dioses (Inti, Quilla, Illapa, Pachacámac, Pachamama, Kon) y la retroalimentación del entorno.

- Evaluar el rendimiento gráfico y la legibilidad visual de los modelos y VFX integrados en el motor mediante pruebas de estrés (*profiling*) manteniendo una tasa de fotogramas estable.

1.7. Alcance del proyecto

El presente proyecto abarca la producción, configuración y optimización de los elementos gráficos tridimensionales, los sistemas de partículas y *shaders* necesarios para ensamblar la *vertical slice* (pequeña porción jugable) del videojuego.

El alcance incluye la creación de la malla tridimensional optimizada del personaje principal, la fauna ecuatoriana y los elementos del entorno (casillas hexagonales, vegetación, rocas y torres), priorizando un bajo conteo poligonal preparado para el uso de colores sólidos. Asimismo, comprende el diseño e implementación de sistemas de partículas y *shaders* en Unity orientados a representar las habilidades otorgadas por las cartas de los dioses y la retroalimentación interactiva del tablero (estado de captura, muerte, o marcación de casillas).

Fuera del alcance:

El alcance del proyecto se centra exclusivamente en el desarrollo de visuales 3D como el modelado de *assets* y lo correspondiente al *technical art*, por lo tanto este trabajo excluye explícitamente el desarrollo de la lógica central de programación del juego, la inteligencia artificial (IA) de los enemigos, el proceso de animación profunda de los personajes, así como el diseño de la interfaz de usuario (UI/UX) e ilustraciones 2D, responsabilidades que han sido delegadas a otros miembros del equipo de desarrollo.

1.8. Delimitaciones

La investigación se encuentra sujeta a las siguientes delimitaciones:

- Tecnológicas: El desarrollo gráfico está estrictamente condicionado por las limitaciones del hardware no especializado que se utiliza habitualmente en las instituciones públicas. Todo el ensamblaje y prueba de los elementos visuales se realizan de forma nativa utilizando el motor gráfico Unity [9], prescindiendo de técnicas de renderizado costosas como *Ray Tracing* o iluminación dinámica global.
- Metodológicas: El estudio se limita a evaluar las métricas de rendimiento directamente relacionadas con el apartado visual (*Draw Calls*, consumo de VRAM y poligonaje).
- Temporales: El proyecto engloba únicamente la fase de producción de los *assets* necesarios para el prototipo funcional o *Vertical Slice*, sin abarcar el ciclo de vida completo de publicación comercial del título.

1.9. Beneficiarios

- Beneficiarios directos: El público objetivo primordial son estudiantes de 5to y 8vo grado de educación general básica donde aprender sobre las culturas incaicas, quienes disfrutarán de una experiencia interactiva y fluida que les permitirá acercarse al patrimonio cultural del Qhapac Ñan sin interrupciones técnicas. Adicionalmente, las propias instituciones públicas se benefician al poder implementar el software en hardware modesto sin requerir inversiones excesivas en equipamiento de gama alta.

- Beneficiarios indirectos: El profesorado al poder generar nuevo material de enseñanza lúdica sobre las culturas incas con un software aprovechado para equipos de bajo rendimiento[10].

1.10. Metodología general

La investigación se enmarca en un enfoque mixto (cuantitativo y cualitativo), una aproximación recomendada para proyectos tecnológicos que requieren evaluar tanto la eficiencia del hardware como la percepción humana [11].

Por una parte, el componente cuantitativo se centra en el análisis objetivo del rendimiento del sistema utilizando la herramienta *Unity Profiler*. Se recolectan datos duros respecto al conteo de polígonos, el uso de la memoria de video (VRAM), el número de llamadas de dibujado (*Draw Calls*) y las fluctuaciones de la tasa de fotogramas por segundo (FPS).

Por otro, el componente cualitativo evalúa la legibilidad visual y la coherencia estética del modelo mediante técnicas de observación y pruebas de usuario. El flujo de trabajo adopta un diseño iterativo [12], estructurado en fases de preproducción, modelado, horneado de mapas base, programación de VFX y *shaders* junto a pruebas de estrés.

1.11. Resultados esperados

Al finalizar el desarrollo del proyecto, se espera entregar un conjunto completo de *assets* tridimensionales (personajes y entornos modulares) con una topología optimizada para el entorno de desarrollo. Asimismo, se obtendrá un repositorio de *prefabs* en Unity que contendrá los sistemas de partículas configurados mediante técnicas de bajo costo computacional (usando *sprites* simples, emisión controlada y materiales multiplicativos/aditivos).

Los productos entregables serán los siguientes:

- Modelo del protagonista *low poly* con *flat shading* (formato .fbx)
- Modelos *low poly* con *flat shading* de oso, cóndor, jaguar, serpiente, pecarí de la fauna representativa (formato .fbx)
- *Assets* modulares 3D para las grillas hexagonales (formato .fbx)
- Sistemas de partículas de bajo coste computacional representativo de cada poder de los dioses (Inti, Quilla, Illapa, Pachacámac, Pachamama, Kon)
- Los elementos desarrollados funcionan en computadoras de bajo rendimiento

1.12. Relación con el protocolo de investigación

La estructura del documento de titulación mantiene una correspondencia directa y una alineación con el protocolo de investigación previamente aprobado. Los objetivos generales y específicos, así como la delimitación técnica del uso del *flat shading* y la mitigación de *overdraw* en partículas, han guiado el plan de trabajo tal cual fue concebido, asegurando que la propuesta gráfica resuelva la problemática computacional identificada inicialmente en el contexto de hardware limitado.

Capítulo II

2. Marco Teórico

2.1. Antecedentes de investigación

La optimización de recursos tridimensionales y la gestión eficiente de efectos visuales (VFX) en entornos de hardware con capacidades limitadas, han sido objeto de diversos estudios en la industria de los videojuegos. En la estandarización de flujos de trabajo orientados a la producción de arte técnico, la investigación científica presenta directrices claras para evitar la saturación de los motores gráficos.

Respecto a la optimización tridimensional, Boonsawang et al. [2] documentaron la estructuración de pipelines específicos para modelos estilizados, priorizando una topología limpia que soporte animaciones sin recurrir a subdivisiones que eleven el costo de recursos gráficos. Su investigación enfatiza que equilibrar la complejidad del diseño visual con métricas tangibles de eficiencia en la memoria gráfica es vital para proyectos de renderizado en tiempo real. En la misma rama, Pirhonen [3] demuestra que la estructuración correcta de las mallas *low-poly* es esencial para mitigar fallos en la deformación de articulaciones durante la etapa de *rigging*.

A nivel perceptual, Fenech [5] analiza el impacto del estilo artístico en videojuegos de perspectiva isométrica. Sus pruebas de rendimiento evidencian que las estéticas de baja carga poligonal benefician la claridad visual y la lectura espacial de los jugadores, mejorando la respuesta cognitiva frente a tableros estructurados.

Por este motivo se decide que el estilo artístico no solo tiene que ayudar a la identidad visual, también debe ser una herramienta para mitigar la sobrecarga grafica en los dispositivos no especializados en los que será usado.

En el campo de los efectos visuales (VFX), Imam y Areeb [13] parametrizaron la integración nativa de simulaciones de partículas en motores gráficos para evitar cuellos de botella en la memoria de video (VRAM). Asimismo, Sibai et al. [4] investigaron el impacto computacional del paralelismo y los subprocesos en estos sistemas, demostrando que el rendimiento de un juego está estrictamente atado a la tasa de emisión y al cálculo de colisiones físicas en cada fotograma.

Más allá de la literatura científica, la viabilidad de la abstracción visual y la optimización de recursos se evidencia en proyectos comerciales que han marcado hitos en la industria y que sirven como referentes directos para el desarrollo de Qhapac Ñan: El Camino del Chasqui.

En primer lugar, destaca el caso de Minecraft, desarrollado por Mojang Studios [14]. Este título representa el exponente máximo de la abstracción geométrica extrema en la era moderna. Su estética, basada enteramente en vóxeles (cubos tridimensionales) y texturas de baja resolución, demuestra que la fidelidad fotorrealista no es un requisito para lograr la inmersión del jugador. Desde el arte técnico, Minecraft resuelve el problema de la sobrecarga de la GPU mediante la simplificación radical de sus mallas y el uso de colores e iconografía sumamente legibles.

Este conteo poligonal le permite al motor renderizar mundos procedurales gigantescos sin colapsar la memoria de los dispositivos. Para el proyecto Qhapac Ñan, Minecraft aporta un antecedente crucial: la validación empírica de que la fauna andina y los elementos del

entorno pueden reducirse a sus formas geométricas más primitivas (*low-poly*) manteniendo una conexión cognitiva y emocional fuerte con el usuario.

Figura 1
Minecraft

Nota. Imagen del videojuego Minecraft donde se demuestra su estilo visual *voxel* con texturas simples. Obtenido de [14]

Por otro lado, dentro del género específico de estrategia, el videojuego The Battle of Polytopia, creado por el estudio Midjiwan AB [15], constituye el referente más directo para la dirección de arte y la optimización de este proyecto. Polytopia es un juego de estrategia por turnos que comparte múltiples similitudes arquitectónicas con Qhapac Ñan, utiliza una perspectiva de cámara isométrica, su terreno está segmentado en módulos cuadriculados que representan distintos biomas, y todo su apartado gráfico está construido bajo una estricta directriz *low-poly* con flat *shading*.

Técnicamente, el éxito de The Battle of Polytopia radica en su capacidad para ejecutarse con fluidez en dispositivos móviles de gama baja, un hardware con limitaciones

de procesamiento gráfico muy similares al *hardware* objetivo donde se exhibirá Qhapac Ñan. Asimismo, el tratamiento de sus Efectos Visuales (VFX) es un modelo a seguir, en lugar de utilizar sistemas de partículas tridimensionales complejos para los ataques, magias o la captura de ciudades, Polytopia emplea efectos de interfaz y *sprites* aditivos bidimensionales de corta duración. Estos efectos proporcionan la retroalimentación exacta que el jugador necesita para entender el tablero, sin generar *overdraw* ni saturar las llamadas de dibujado (*Draw Calls*).

Figura 2
The Battle of Polytopia

Nota. Imagen del videojuego The Battle of Polytopia donde se evidencia su mapa modular y perspectiva isométrica. Obtenido de [15]

El análisis de estos dos referentes comerciales corrobora que la adopción de una estética minimalista, cuando está respaldada por un flujo de trabajo optimizado, establece una identidad visual clara que favorece la legibilidad de las mecánicas de juego.

2.2. Bases teóricas

El presente proyecto se fundamenta en diversas teorías y principios gráficos que rigen el arte técnico en los videojuegos modernos.

- **Modelado *Low-Poly* y Abstracción Gráfica**

El modelado de baja densidad poligonal (*low-poly*) dejó de ser únicamente una limitación técnica de las consolas de antaño para convertirse en una corriente estética y una herramienta de abstracción [16]. La teoría de la abstracción gráfica sostiene que al simplificar la geometría de un objeto se reduce el peso de los cálculos matemáticos del motor sin sacrificar la esencia cognitiva que permite al usuario identificar el objeto. Sin embargo, esta abstracción debe ser meticulosa; una reducción geométrica destructiva e incontrolada interfiere con la identificación del jugador y la proyección sobre su avatar [17].

- **Topología Geométrica para Animación**

La topología es el estudio de la distribución de vértices, aristas y caras (polígonos) sobre una superficie 3D. En personajes interactivos, la teoría de deformación geométrica establece que la malla debe fluir siguiendo la anatomía muscular del sujeto (mediante *edge loops* o bucles de aristas) [3].

La topología sobre seres articulados debe seguir una anatomía básica, ayudando a la deformación correcta de las articulaciones, ya que si no se preparan las articulaciones en la topología, en la etapa de *rigging* la malla se va a deformar de manera incorrecta generando artefactos visuales.

- **Sombreado Plano (*Flat Shading*) y Oclusión Ambiental**

La iluminación en tiempo real es uno de los procesos que más estrés genera en la GPU. La técnica de *flat shading* anula el suavizado de las normales de la malla, calculando la luz de manera uniforme a través de todo el polígono en lugar de interpolarla entre los vértices. Para suplir la pérdida de profundidad visual que esto genera, se recurre a la teoría del *bake* (horneado) de texturas, específicamente calculando la oclusión ambiental (*ambient occlusion*). Esta técnica pre-calcula las sombras de contacto generadas por la luz indirecta y las "pinta" en la textura estática, eliminando el coste de procesarlas durante la ejecución del juego [1].

- **Sistemas de Partículas**

Los sistemas de partículas son representaciones computacionales de nubes volumétricas utilizadas para simular fenómenos difusos (fuego, magia, clima) [18]. Teóricamente, un sistema de partículas se rige por la cinemática básica: un emisor (*emitter*) expulsa elementos bidimensionales (*sprites* o *billboards* que siempre miran a la cámara) a los que se les aplican vectores de velocidad, gravedad, color y un ciclo de vida (*lifespan*).

2.3. Bases metodológicas

El desarrollo de esta investigación se enmarca en un enfoque mixto (cuantitativo y cualitativo), el cual es idóneo para la producción de *software* interactivo, donde confluyen las métricas exactas del rendimiento del *hardware* y la percepción subjetiva del usuario humano [11].

El análisis de los datos se estructura bajo dos metodologías propias del desarrollo de videojuegos: el benchmarking de rendimiento y la evaluación heurística visual.

- Evaluación cuantitativa (*benchmarking* y *profiling*): Orientada a la medición del rendimiento del *hardware* frente a un "presupuesto técnico" preestablecido. Se

fundamenta en el análisis estadístico de los recursos consumidos utilizando *Unity Profiler*. Los datos obtenidos como la tasa de fotogramas (FPS), consumo de VRAM y número de llamadas de dibujado (*draw calls*), se compararon con los límites del hardware objetivo. Si los *assets low-poly* y los sistemas de partículas lograban mantenerse por debajo de este límite técnico, se validaron matemáticamente en cuanto al éxito de la optimización [9].

- Evaluación cualitativa (evaluación heurística y de usuario): Enfocada en medir la coherencia estética y la legibilidad visual del arte. Se evaluaron cualitativamente si la abstracción geométrica de los biomas y la fauna incaica permitían una rápida identificación desde la cámara isométrica. Asimismo, se validó si la retroalimentación de los efectos visuales (VFX) comunicó correctamente la acción en el tablero sin saturar la pantalla.

Las pruebas cualitativas se basaron en un *play test* con el público objetivo que constaba de niños de 8 a 12 años, quienes fueron entrevistados con preguntas como "¿Entiendes qué animal es este?" o “¿Cuál es el dios que lanza un rayo?”, estas preguntas nos darán la visión necesaria para hacer los ajustes necesarios en el apartado visual.

Adicionalmente, el proyecto adopta un modelo de diseño iterativo [12]. En el arte técnico de videojuegos, el primer modelo o efecto nunca es el definitivo.

En esencia siempre se pasa un primer modelo al motor gráfico (Unity) para analizar la carga computacional, de ser el caso de que no se ajuste a los parámetros plantados, el modelo regresa a las primeras etapas de desarrollo para los ajustes necesarios y volver a iterar hasta conseguir los resultados objetivos.

Esta iteración constante de crear, medir (*profiling*), ajustar y volver a probar permite que los prototipos visuales evolucionen en ciclos cortos, garantizando que el arte final alcance el

equilibrio exacto entre calidad estética y rendimiento computacional antes de integrarse a la *vertical slice* del juego.

2.4. Bases tecnológicas

La materialización de la propuesta gráfica recae en ecosistemas tecnológicos específicos que soportan el flujo de trabajo del arte técnico tridimensional:

- **Blender:** Para la producción de los *assets* gráficos, se evaluaron los estándares de la industria, optando finalmente por la selección de Blender por encima de alternativas privativas como Autodesk Maya. Esta decisión arquitectónica no responde a una preferencia operativa, sino a una alineación técnica con las necesidades de optimización del proyecto y el ecosistema de desarrollo de arte técnico.

  El flujo de trabajo de Blender está diseñado para facilitar iteraciones geométricas rápidas. Blender fue seleccionado como el principal software de modelado 3D debido a que su entorno justifica un robusto sistema de modelado directo y herramientas ágiles para el despliegue de mapas UV. Estas características configuran un flujo de trabajo altamente optimizado para la creación de topología limpia, lo cual es un requisito indispensable para la construcción de los personajes y la fauna del juego. En la consolidación de mallas *low-poly*, donde cada vértice tiene un peso estructural crítico, la manipulación directa de Blender superó la canalización tradicional de Maya, permitiendo a la dirección de arte un control absoluto sobre el conteo poligonal final.

  La unificación del pipeline técnico representó un factor decisivo. Mientras que Autodesk Maya exige la construcción de sistemas óseos rigurosamente manuales para cada entidad, Blender ofrece soluciones inmediatas y modulares a través de

extensiones integradas como el *addon Rigify*. Esta herramienta facilitó la generación de meta-esqueletos preconfigurados que se adaptaron con precisión a la anatomía de los diferentes mamíferos y aves del videojuego, consolidando las fases de modelado, corte de UVs y *rigging* en un único entorno de software.

Finalmente, desde la perspectiva de sostenibilidad e implementación, el proyecto "Qhapac Ñan" posee un enfoque educativo orientado a instituciones de Ecuador. La adopción de Blender, al ser un *software* de código abierto (*Open Source*), elimina las barreras económicas asociadas a las costosas licencias corporativas. Esto garantiza que el proyecto original, así como sus futuras actualizaciones, expansiones o mantenimientos, puedan ejecutarse de manera libre y escalable, democratizando el acceso a las herramientas de desarrollo sin comprometer la calidad visual ni el rendimiento técnico.

- **Substance Painter:** Para la fase de texturizado y tratamiento de color, se prescindió del uso de Adobe Photoshop a favor de la adopción íntegra de Adobe Substance 3D Painter. Esta elección se fundamenta en las limitaciones inherentes de los editores de imagen bidimensionales (2D) frente a las exigencias de un pipeline de arte técnico moderno, especialmente en un proyecto que requiere optimizar la legibilidad visual mediante *flat shading*.

  El primer argumento técnico radica en la naturaleza del espacio de trabajo. Photoshop obliga al artista a pintar sobre un lienzo bidimensional (el mapa UV desplegado). Este enfoque tradicional dificulta la percepción del volumen geométrico y genera problemas graves de distorsión y discontinuidad visual en las costuras (*seams*) de la malla tridimensional. Substance 3D Painter, por el contrario, permite

interactuar y pintar directamente sobre la geometría 3D en tiempo real, garantizando que los trazados fluyan de manera continua a través de los polígonos sin importar los cortes del mapa UV.

El segundo factor decisivo es la capacidad de cálculo matemático de los volúmenes. Adobe Substance 3D Painter fue utilizado exclusivamente para la fase de texturizado y horneado de luz. Este software cuenta con un potente motor de horneado (*Bake*) que permite transferir oclusiones ambientales con gran precisión geométrica hacia texturas estáticas 2D (mapas de color base o Albedo). Esta característica resultó crítica para suplir la iluminación dinámica dentro del motor gráfico Unity, ya que permite simular la profundidad y el sombreado de contacto sin costo para la GPU. Photoshop carece de un motor nativo capaz de leer mallas 3D para calcular y renderizar estos mapas de oclusión de forma automatizada.

Finalmente, la dirección de arte exigía un texturizado estilizado y artesanal. Se escogió este programa exclusivamente para el texturizado de pintura a mano (*hand painting*) por su alta versatilidad de brochas y su avanzado sistema de pintura por capas. A diferencia de los flujos de trabajo en Photoshop donde se deben deformar imágenes para encajarlas en las islas UV, Substance Painter permite proyectar y ajustar detalles precisos, como la iconografía incaica de la vestimenta del Chasqui, símbolos o sombras de contacto falsas, ubicándolas en capas superiores al color base de forma destructiva o no destructiva según las necesidades del modelo.

- **Unity:** La selección del entorno de desarrollo es la decisión arquitectónica más crítica para el éxito del proyecto, dado que define los límites de ejecución en el *hardware* objetivo. Para "Qhapac Ñan: El Camino del Chasqui", se evaluaron motores gráficos

líderes en la industria como Unreal Engine, Godot y Unity. La decisión final de utilizar Unity se fundamentó en su adaptabilidad técnica frente a los requerimientos de bajo consumo, descartando a sus competidores por razones de sobrecarga de procesamiento y madurez del ecosistema de arte técnico.

En primera instancia, se descartó Unreal Engine debido a su alto costo computacional base. Si bien Unreal es el estándar de la industria para gráficos fotorrealistas de alta fidelidad, su arquitectura interna (*core overhead*) exige recursos significativos de memoria y procesamiento, incluso en proyectos vacíos. Dado que el *hardware* objetivo del proyecto abarca dispositivos no especializados con especificaciones limitadas (como 4 GB de RAM y gráficos integrados), la implementación de Unreal Engine habría provocado cuellos de botella críticos, imposibilitando la meta de mantener 60 fotogramas por segundo (FPS) estables.

Por otro lado, aunque Godot Engine representa una alternativa de código abierto sumamente ligera y funcional, Unity superó a este motor gracias a la madurez de sus herramientas nativas para la creación de arte técnico tridimensional. Unity destaca por su versatilidad en proyectos 3D e incluye un sistema de partículas nativo altamente modular. Además, el ecosistema de Unity Shader Graph permitió estructurar visualmente la lógica matemática del *CellShader* y la topología geométrica sin necesidad de programar sombreadores en lenguajes de bajo nivel (como HLSL/GLSL puro) desde cero, agilizando drásticamente las iteraciones visuales.

Finalmente, el motor gráfico Unity consolida su viabilidad gracias a su arquitectura y herramientas de optimización. Desde el punto de vista técnico, el

proyecto contempla el uso de Unity en conjunto con el *Universal Render Pipeline* (URP) como canalización de renderizado. El URP permitió deshabilitar por completo los cálculos de iluminación dinámica global y sombras físicas, aislando el procesamiento gráfico al uso exclusivo de materiales *Unlit*.

A nivel de validación científica, el sistema de diagnóstico integrado (*Unity Profiler*) resultó esencial para la evaluación de métricas de rendimiento. Esta herramienta permitió comprobar empíricamente la reducción de los lotes de renderizado (*batches*) y certificar que la carga computacional generada por los sistemas de partículas se mantenía dentro de los márgenes seguros para computadoras de gama de entrada.

2.5. Arquitecturas, modelos y estándares

La construcción técnica de este proyecto no responde únicamente al uso de programas, sino a arquitecturas de software que estandarizan el desarrollo:

Arquitectura basada en componentes (*Component-based architecture* - CBA)

El motor seleccionado (Unity) rige su funcionamiento bajo el modelo CBA. En lugar de utilizar una jerarquía rígida de programación orientada a objetos convencional, el CBA permite que cada elemento tridimensional (*game-object*) actúe como un contenedor vacío.

Esta arquitectura además acelera los flujos de desarrollo, ya que el *technical artist* puede generar todos los sistemas de partículas que se necesitan en el proyecto, y posteriormente añadirlos e integrarlos a los *assets* ya establecidos por el resto del equipo.

Estándar de tasa de refresco y pipeline de renderizado

El estándar visual para una experiencia interactiva sin fricción cognitiva establece como meta una ejecución a 60 FPS (fotogramas por segundo) estables. Para lograr este modelo matemático, el *Rendering Pipeline* debe procesar la geometría, aplicar los mapas de texturas y mostrar la imagen final en pantalla en aproximadamente 16.6 milisegundos por fotograma. Para no saturar este estándar, se aplican reglas estrictas de *draw calls* (órdenes de la CPU a la GPU para dibujar un objeto) y se elimina el *overdraw*.

2.6. Herramientas de desarrollo

En función de las bases tecnológicas expuestas, el proyecto justifica y adopta las siguientes herramientas para el desarrollo operativo:

- **Unity:** Elegido como el motor gráfico base del proyecto. Unity destaca por su versatilidad en proyectos 3D e incluye un sistema de partículas nativo (*Shuriken* o *VFX Graph*) altamente modular [9]. Su sistema de *profiling* integrado es esencial para la evaluación de métricas de rendimiento.
- **Blender:** Seleccionado como el principal software de modelado 3D. Se justifica por su robusto sistema de modelado directo, despliegue de mapas UV y un flujo de trabajo altamente optimizado para la creación de topología limpia, requisitos indispensables para los personajes y fauna del juego.
- **Substance 3D painter:** Utilizado exclusivamente para la fase de texturizado y horneado de luz. Su potente motor de *Bake* permite transferir oclusiones ambientales con gran precisión geométrica hacia texturas 2D (*Albedo*), lo cual es crítico para suplir la iluminación dinámica en el motor.
- ***Unity Profiler*:** Herramienta interna de análisis que se utilizará como estándar de validación durante las pruebas de estrés para medir picos de la GPU y memoria.

2.7. Marco conceptual

Para una correcta interpretación de los elementos descritos en la investigación, se definen los siguientes conceptos técnicos:

- ***Draw Call*:** Llamada que realiza el procesador central (CPU) a la tarjeta gráfica (GPU) indicándole que dibuje una malla tridimensional en la pantalla. Un exceso de *draw calls* produce cuellos de botella en el rendimiento [10].
- ***Low-Poly*:** Corriente estética y técnica de optimización que reduce al máximo el número de polígonos que componen un modelo 3D, priorizando la silueta.
- ***Bake* de Texturas:** Proceso matemático mediante el cual la información compleja (luz, sombras o detalles de alta resolución) se procesa en el software de diseño y se imprime en una imagen estática (textura) [9].
- ***Overdraw*:** Efecto perjudicial en el renderizado que ocurre cuando la GPU dibuja múltiples píxeles transparentes o semitransparentes uno sobre otro en el mismo fotograma (muy común en partículas no optimizadas) [4].
- ***Profiling*:** Monitoreo analítico de un software en tiempo real para observar su comportamiento a nivel de consumo de memoria, CPU y GPU.
- ***Flat Shading*:** Tipo de sombreado de superficie donde la luz se calcula por cada cara del polígono y no se interpola, lo que da a los modelos un aspecto facetado característico del estilo *low-poly*.

2.8. Relación entre la teoría y la propuesta tecnológica

Las teorías gráficas y arquitectónicas descritas no son aisladas, sino los cimientos que respaldan todas las decisiones tomadas durante el desarrollo de *Qhapac Ñan*.

La teoría de la abstracción visual y el *low-poly* es la que justifica directamente la simplificación de los biomas (Costa, Amazonía, Sierra) y la geometría de la fauna andina. Al aplicar estas bases teóricas, la propuesta tecnológica garantiza que los jugadores identifiquen culturalmente a los personajes sin forzar el hardware. Además de que los efectos visuales propuesto hacen que los niños sientan como los dioses cobran vida.

Por otro lado, la teoría de sistemas de partículas y la mitigación del *overdraw* explican por qué la propuesta tecnológica rechaza el uso de emisores con físicas reales para los poderes divinos. En su lugar, al basarse en los estudios de paralelismo de la GPU [13], el proyecto asume como solución el uso de *sprites* 2D con materiales aditivos. De esta manera, el marco teórico justifica cada recorte estético y cada decisión metodológica implementada en la *Vertical Slice* del videojuego.

Capítulo III

3. Manual De Usuario

3.1. Presentación de la propuesta y perfil de usuario

"Qhapac Ñan: El Camino del Chasqui" es un videojuego de estrategia por turnos con elementos *roguelike* diseñado para centros educativos, donde los jugadores encarnan al

astrólogo Chaicó-chasqui. A diferencia de juegos educativos tradicionales, este título se puede jugar varias veces, cada intento teniendo un toque único por su generación procedural y la dificultad progresiva a través de cada nivel, retando a los niños a usar el pensamiento crítico y poderes de los dioses andinos para cumplir una profecía histórica.

El estado actual del juego es un prototipo que contiene la mayoría de las escenas del juego final, siendo estas el menú principal y los tres niveles de jugabilidad. Todas las mecánicas del juego ya están implementadas, tales como la generación del mapa, el movimiento, los enemigos, y los poderes que puede usar el jugador.

El público objetivo del videojuego son niños de 8 a 12 años, quienes tienen un interés por los videojuegos, historias de aventura, aprender de manera interactiva y la tecnología. Los controles son diseñados para apelar a la juventud que interactúa rápidamente con clics o toques a la pantalla, y las instrucciones se acoplan a esta mentalidad, en vez de poner instrucciones largas que obligan al jugador a leer.

3.1.1. Buyer persona

- Nombre: Mateo
- Edad: 10 años
- Ocupación: Estudiante
- Objetivo: Quiere que sus clases no sean tan aburridas y hacer otras actividades. Quiere ver qué pasa si llega al final del mapa.
- Frustraciones: Se frustra si los controles no responden rápido.
- Escenario de uso: Mateo ve el videojuego, toma el ratón y entiende rápidamente que debe mover a Chaicó-chasquí por los hexágonos. Usa la carta de "Viento" para moverse rápido porque quiere ganar a sus amigos que están mirando.

3.1.2. Requisitos mínimos

Tabla 1
Requisitos mínimos del juego

Sistema Operativo	Windows 10 x64 bits

Procesador	Intel Core i3 (5ta Gen.) / AMD Athlon Silver 3050U
Espacio de Almacenamiento	250MB
RAM	4GB
GPU	Intel HD Graphics 5500 / NVIDIA GeForce GTX 600 / AMD Radeon HD 7000

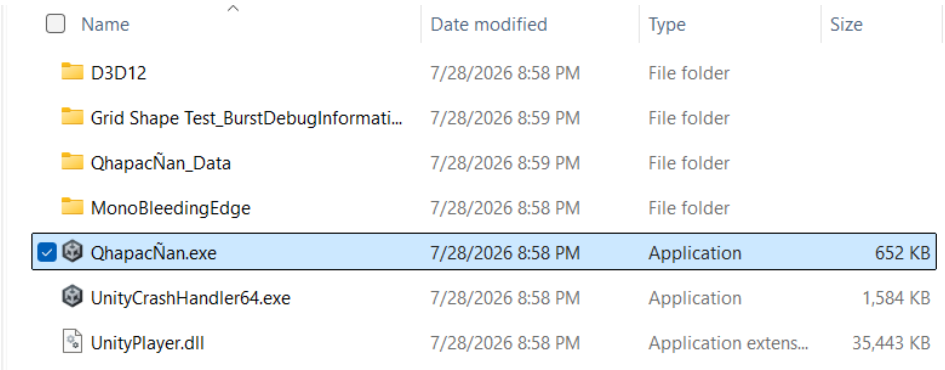
Nota. Descripción de los requisitos mínimos que debe tener un PC para correr el juego Qhapac Ñan

Los componentes de *hardware* mínimos se obtuvieron mediante las funciones de *Profiling* y *Benchmarking* de Unity, comparándolo con componentes que cumplen estos requisitos mínimos.

3.2. Acceso e inicio de la experiencia

1. Copiar la carpeta completa del ejecutable del medio de distribución obtenido.
2. Hacer doble clic izquierdo sobre el archivo con nombre “QhapacÑan.exe”.

Figura 3
Contenido de ejecutable del juego



Nota. Contenido de la carpeta que se obtiene al adquirir el juego Qhapac Ñan. Se resalta el ejecutable que se debe hacer doble clic para iniciar el juego.

3. En la pantalla de inicio, hacer clic sobre el botón que tiene el ícono de “Reproducir”.

Figura 4
Pantalla de inicio de Qhapac Ñan

Nota. Pantalla de inicio de Qhapac Ñan. Se señala el botón que se debe seleccionar para empezar el juego.

3.3. Navegación y controles

3.3.1. Controles del juego

Tabla 2
Controles del juego

Clic Izquierdo	Seleccionar
Clic Derecho + Mantener	Mover cámara
Rueda de Mouse	Acercar o alejar la cámara del jugador

Nota. Descripción de los controles del juego Qhapac Ñan.

3.3.2. Navegación del juego

El usuario empieza el juego en el menú principal, donde tiene los botones de empezar el juego, ver y cambiar los ajustes, y salir del programa.

Figura 5
Explicación de pantalla de inicio de Qhapac Ñan.

Nota. Pantalla de inicio del juego Qhapac Ñan. Se señalan los botones para empezar el juego, las configuraciones, y para salir del programa.

Presionando el botón de ajustes abre un menú en donde el jugador puede elegir entre cambiar la configuración de audio o video del juego. La opción de audio permite editar el volumen maestro del juego, y las opciones de configuración de video permiten reducir la cantidad máxima de fotogramas por segundo (FPS), cambiar la resolución de pantalla del juego, activar o desactivar VSync y cambiar el esquema de colores del juego en el caso de daltonismo por parte del usuario.

Figura 6
Menú de opciones de configuración del juego.

Nota. El menú que aparece al seleccionar el botón de configuraciones en la pantalla de inicio del juego.

Figura 7
Pantalla de cambio de volumen maestro del juego.

Nota. Menú que aparece al seleccionar la opción de configuración de audio.

Figura 8
Menú de opciones de configuración de video.

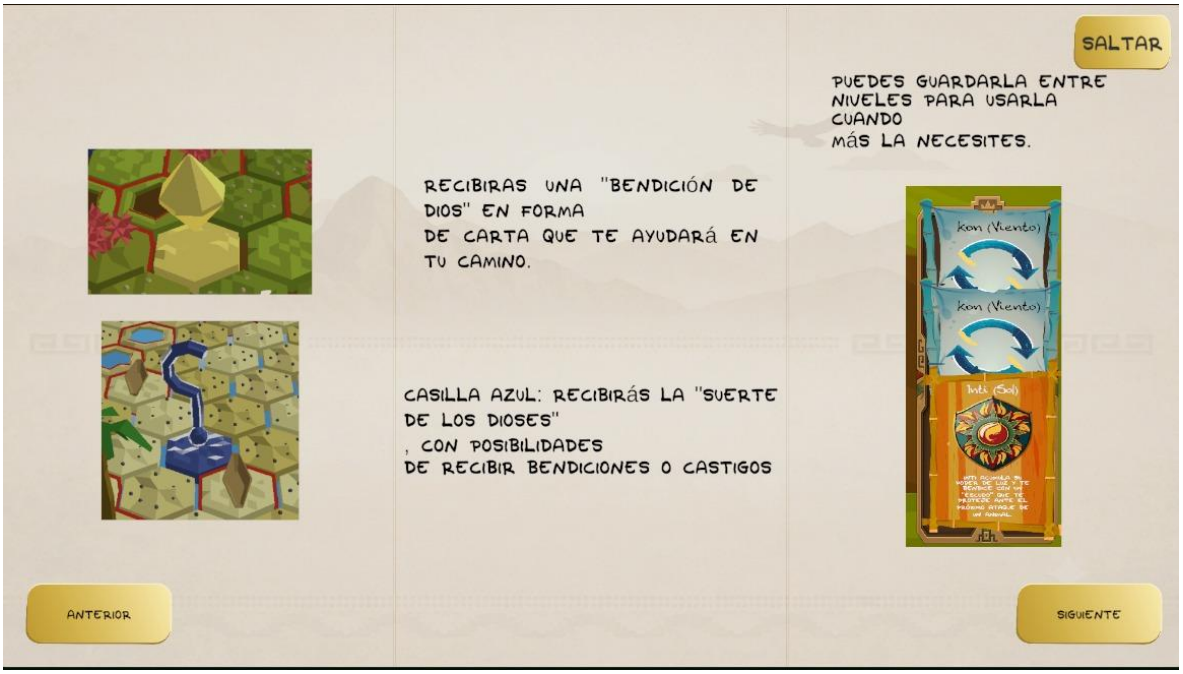
Nota. Menú de opciones al seleccionar la opción de configuración de video.

Al presionar el botón de jugar, se presenta un tutorial para que el jugador entienda la historia y el objetivo del juego. Tiene botones para avanzar y retroceder por el tutorial, y un botón que le permite saltar el tutorial.

Figura 9
Página 1 del tutorial.

Nota. La primera página del tutorial que aparece al iniciar el juego.

Figura 10
Página 2 del tutorial.



Nota. La segunda página del tutorial, que sale al presionar el botón “Siguiente” en la primera página.

Figura 11
Página 3 del tutorial.

Nota. La tercera página del tutorial, que sale al presionar “Siguiente” en la segunda página.

Dentro del juego, el usuario puede seleccionar una cantidad determinada de casillas para avanzar hacia el centro del mapa. Al llegar a una casilla dorada, aparecerán tres cartas, de las cuales el jugador podrá elegir uno. Cada carta tiene un poder que ayudará al jugador en su objetivo. Si aterriza sobre una casilla azul, le saldrán dos cartas iguales, de las cuales elegirá uno. Esta carta le puede dar inmediatamente un efecto positivo o negativo.

Figura 12
Selección de casilla

Nota. Representación visual de la selección de casilla para el movimiento del jugador. Se resalta la casilla que el jugador vaya a seleccionar para dar claridad visual.

Figura 13
Panel de cartas de poder.

Nota. El panel de cartas que le sale al jugador al llegar a una casilla dorada. Tiene 3 cartas para poder seleccionar, y cada carta tiene el nombre de una deidad Inca, además del efecto que otorga.

Figura 14
Panel de cartas de fortuna.

Nota. El panel de cartas que aparece al jugador cuando aterriza en una casilla azul. Le permite elegir de entre dos cartas con efectos desconocidos.

Al llegar a las casillas que están en el centro del mapa, se carga el siguiente nivel. Si es que el jugador es derrotado, aparece un menú que le da contexto adicional y le permite reiniciar el juego o volver al menú principal.

Figura 15
Pantalla de derrota.

Nota. Pantalla que aparece al jugador cuando es atacado por un animal sin tener un escudo activado. Le permite volver al menú principal, o reiniciar el juego.

3.4. Funciones e interacciones disponibles

Dentro del programa existen funciones que permiten al jugador entender los efectos de sus acciones. Estos principalmente se observan cuando el jugador activa una carta de poder, y tiene una semejanza al dominio de la deidad quien está en la carta que el usuario activó. Usando la carta de Inti, aparece un escudo como efecto visual; este mismo efecto aparece cuando el enemigo ataca al jugador mientras tenga un escudo activo. Además, al cambiar una casilla de camino a obstáculo (el efecto de Pachamama), o viceversa (Pachacamac), aparece una planta creciendo y el suelo derrumbándose respectivamente. Cuando el usuario usa el poder de Kon, al moverse, se puede notar un efecto de viento

saliendo del jugador. El efecto del poder de Illapa es representado visualmente por un rayo que cae al piso para llevar al jugador a la casilla a la cual desea teleportarse. Finalmente, al dormir un enemigo con el poder de Mama Quilla, el enemigo tendrá una indicación visual de aquel efecto activándose.

Figura 16
Activación del poder de la carta Inti.

Nota. Efecto visual que aparece al activar el poder de la carta Inti.

Figura 17
Activación del poder de la carta Pachamama

Nota. Representación visual de la activación del poder de la carta Pachamama.

Figura 18
Activación del poder de la carta Pachacamac.

Nota. Efecto visual que aparece cuando el jugador activa el poder de la carta Pachacamac.

Figura 19

Efecto visual del poder de la carta Kon.

Nota. Efecto visual que el jugador visualiza al activar el poder de la carta de Kon.

Figura 20

Efecto de la carta de Illapa.

Nota. Efecto de rayo que aparece al activar el poder de la carta de Illapa.

Figura 21

Efecto visual la carta de Mama Quilla.

Nota. Efecto visual puesto a los enemigos que sufren los efectos del poder de la carta de Mama Quilla.

Capítulo IV

4. Manual Técnico De Implementación

4.1. Dirección de arte y propuesta visual

Para garantizar la viabilidad técnica del videojuego en hardware de bajo rendimiento, la dirección de arte se fundamentó en una estética tridimensional optimizada (*low-poly*) combinada con un sombreado estilizado (*flat shading/cell-shading*). Esta decisión estética no solo disminuye drásticamente la carga de renderizado al prescindir de cálculos de iluminación física, sino que favorece la legibilidad cognitiva de los elementos tácticos desde la perspectiva isométrica de la cámara.

4.2. Diseño de personaje, escenarios y elementos gráficos

El diseño visual se estructuró con dos caminos cada uno con requerimientos de detalles diferentes. Por un lado, el protagonista (Chaicó-chasquí) y la fauna endémica exigían un mayor detalle de identidad visual, justificando un mayor presupuesto de resolución. Por el otro camino, los escenarios (Costa, Amazonía y Sierra) requerían un diseño altamente modular basado en grillas hexagonales, lo que obligó a unificar su diseño bajo una estructura de textura compartida para evitar la saturación de la memoria gráfica.

Figura 22
Comparativa visual entre el personaje y entorno

Nota. Comparativa del detalle visual que se presenta entre el personaje y un hexágono del entorno.

4.3. Producción de *assets* 3D

Dentro del desarrollo de videojuegos, un *asset* se define como un recurso digital completo y funcional, el cual puede integrar desde su geometría base (malla tridimensional), sus coordenadas de mapeo y texturas, hasta su esqueleto interno (*rig*), dejándolo listo para su ejecución lógica dentro del motor gráfico.

El proceso de producción de estos *assets* inició con una fase de recolección de referencias visuales. Para la fauna (oso, cóndor, jaguar, serpiente, pecarí) se recopilaron entre 4 y 9 referencias desde múltiples ángulos. Para el protagonista, dada su complejidad cultural,

el análisis se desglosó por cada implemento de su vestimenta: Q'epi, Unku, Chuspa, Pututu, Ojotas y Chumpi. Esta investigación garantizó que la abstracción geométrica no desvirtuara la precisión histórica.

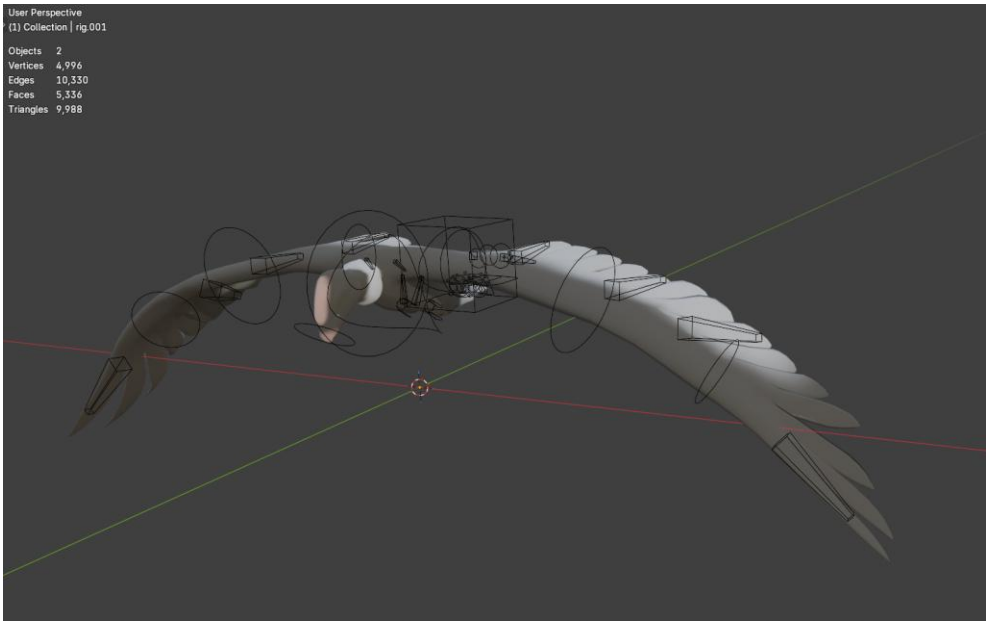
La construcción de la topología se ejecutó íntegramente en Blender. Todo modelado *low-poly* debe estructurarse partiendo de figuras primitivas básicas (cubos, esferas, cilindros). El porqué de esta técnica radica en el control absoluto sobre el flujo de aristas (*edge loops*); al extruir y modelar desde una forma primitiva y no desde mallas densas o generadas proceduralmente, el artista técnico asegura que cada polígono cumpla una función estructural, evitando subdivisiones innecesarias que el motor gráfico tendría que procesar en vano. Durante esta fase, no se estableció un presupuesto de polígonos genérico, sino que la densidad se dictó por la complejidad morfológica y la necesidad de articulación de cada *asset*:

- Protagonista (Chaicó-chasqui): 6.232 vértices.
- Cóndor: 4.996 vértices.
- Oso: 1.204 vértices.
- Jaguar: 933 vértices.
- Pecarí: 496 vértices.
- Serpiente: 336 vértices.

Figura 23
Poly count Chaico Chasqui

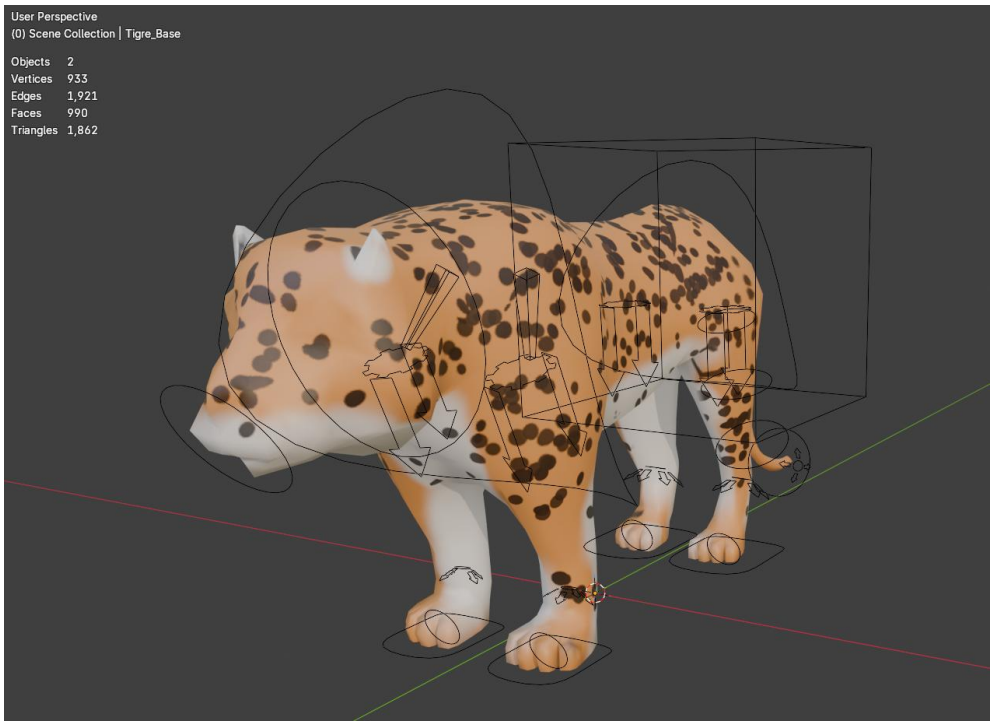
Nota. Conteo de polígonos del personaje principal Chaico Chasqui

Figura 24
Poly count cóndor andino



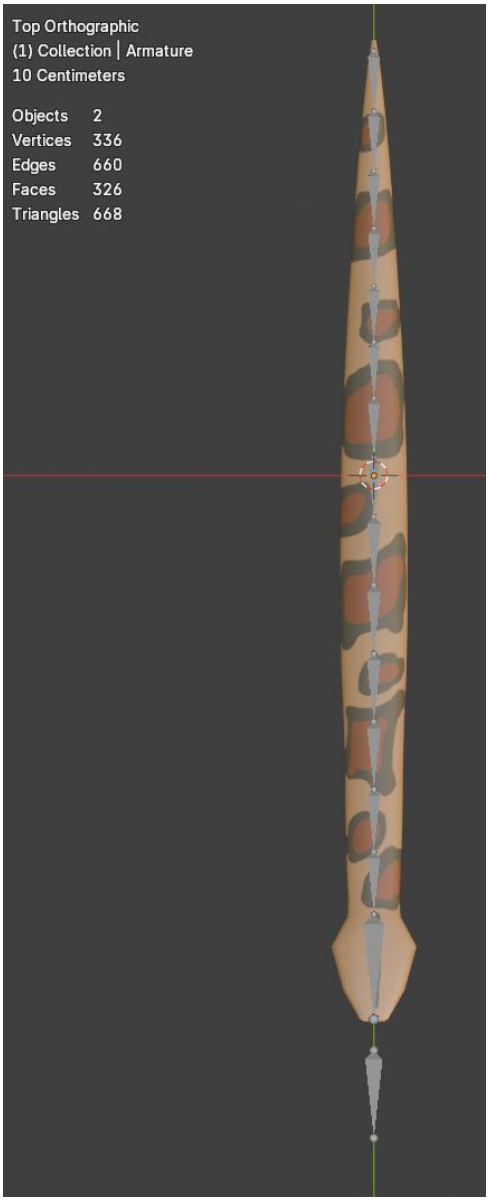
Nota. Conteo de polígonos del enemigo condor

Figura 25
Poly count jaguar



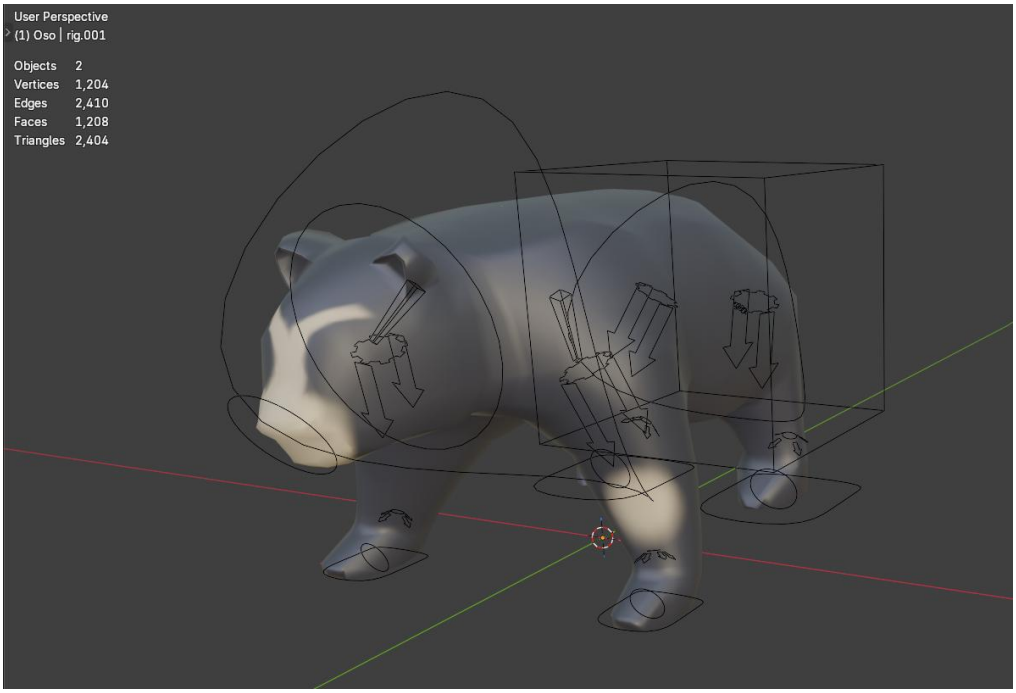
Nota. Conteo de polígonos del enemigo jaguar

Figura 26
Poly count matacaballos



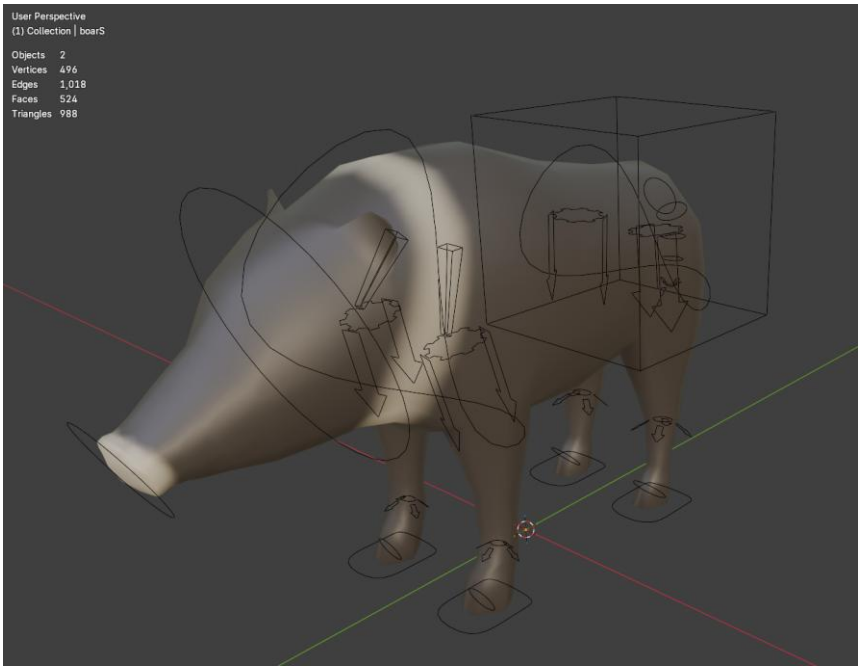
Nota. Conteo de polígonos del enemigo matacaballos

Figura 27
Poly count oso de anteojos



Nota. Conteo de polígonos del enemigo oso de anteojos

Figura 28
Poly count pecarí



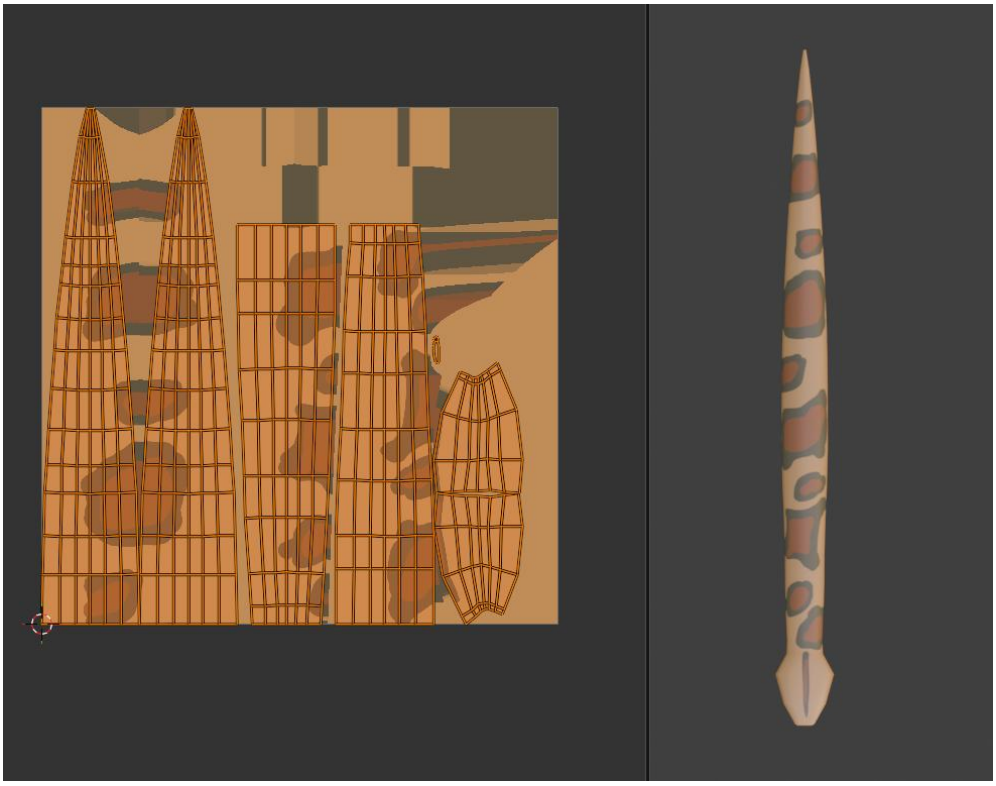
Nota. Conteo de polígonos del enemigo pecarí

Al concebir los modelos estrictamente bajo un estándar de bajos polígonos desde el primer vértice, se descartó por completo la necesidad de realizar una retopología (el proceso de calcar una malla de baja resolución sobre una escultura de alta resolución). La limpieza geométrica se redujo a la fusión manual (*merge*) de vértices aislados para corregir triángulos superpuestos y asegurar que las uniones de la malla permitieran una deformación orgánica limpia en la fase de animación.

El proceso de texturizado se dividió de la siguiente manera:

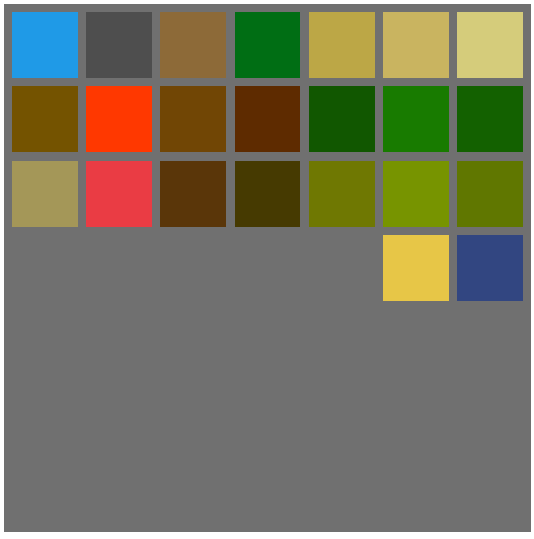
- **Personajes y fauna:** Se realizaron los cortes de coordenadas UVs en Blender. Posteriormente, las mallas se exportaron a Adobe Substance Painter. En este software, se realizó un horneado (*bake*) base de la malla consigo misma. Este paso no tuvo fines de transferencia de detalles geométricos, sino de diagnóstico técnico: el mapa horneado permitió identificar rápidamente artefactos visuales, errores en la distribución de las islas UV o tensiones en las normales. Tras corregir la topología de forma iterativa, se procedió a pintar las texturas base íntegramente a mano utilizando una tableta gráfica. Se exportó únicamente el mapa de color base (Base Map) a una resolución de 1024x1024 píxeles.
- **Entorno y módulos hexagonales:** Los componentes del escenario requerían únicamente colores sólidos para el *flat shading*. Se extrajeron los códigos hexadecimales de color desde Blender y se compilaron en una única *Texture Atlas* de 128x128 píxeles utilizando el software Affinity. Las coordenadas UV de todos los hexágonos y props del entorno se escalaron y posicionaron sobre los píxeles de color correspondientes en este atlas.

Figura 29
Mapa UV del enemigo "Matacaballos"



Nota. Despliegue del mapa UV del modelo 3D del enemigo “matacaballos”

Figura 30
Textura atlas

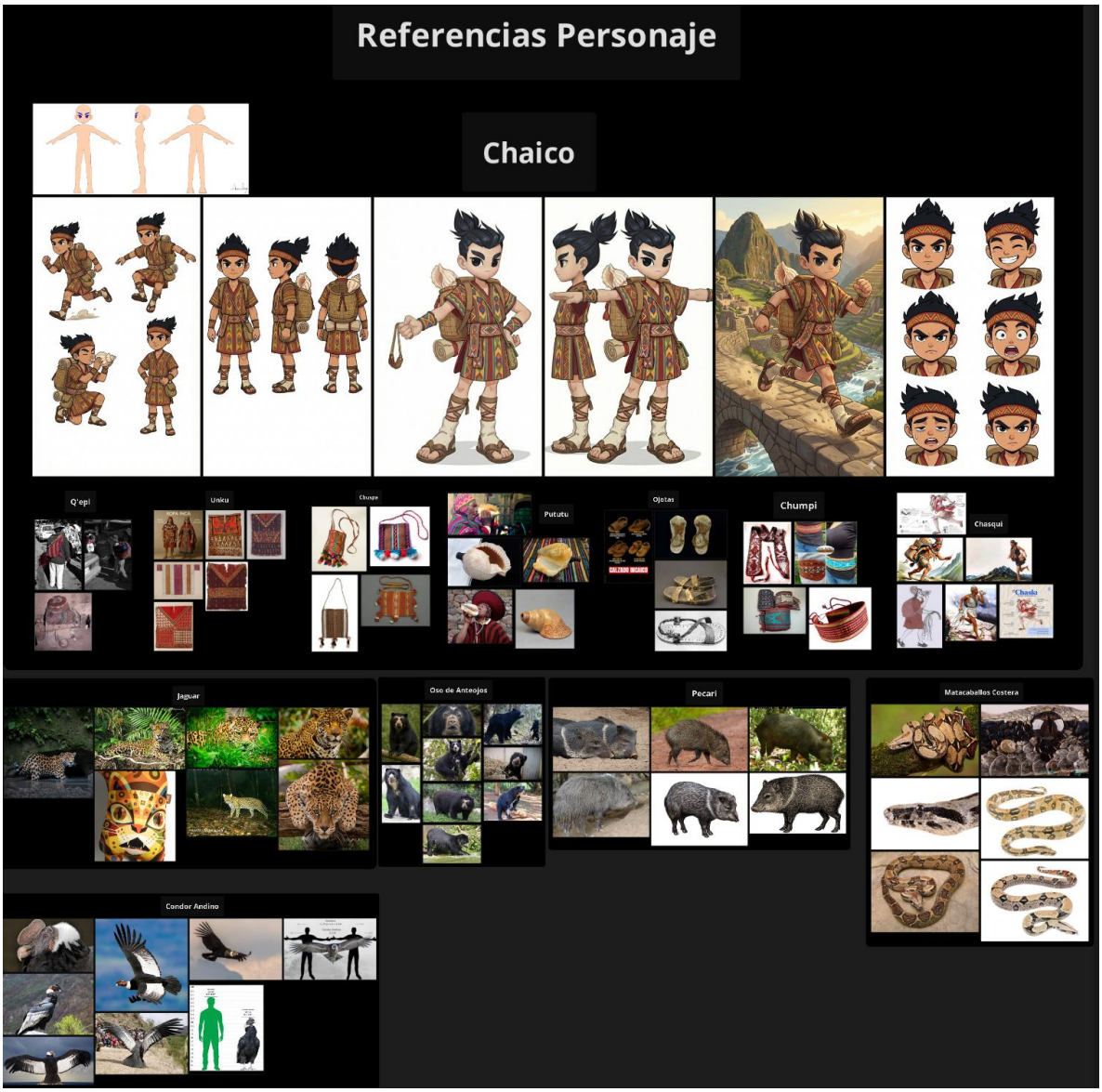


Nota. Textura atlas donde los modelos del entorno extraen sus colores base

4.4. *Pipeline* de producción artística

Primero se recolecta todas las referencias proporcionadas por el equipo 2D, para los animales se recolectaron de 4 a 9 referencias de diferentes ángulos, a diferencia del personaje, que debido a su complejidad visual se tomaron en cuenta referencias visuales de cada implemento de su vestimenta como el Q'epi, Unku, Chuspa, Pututu, Ojotas y Chumpi.

Figura 31
Referencias



Nota. Collage de referencias del personaje y de los animales desarrollados

Después se realizó el modelado manteniendo un bajo conteo poligonal, partiendo de modelos primitivos como cubos, esferas, cilindros, entre otros. En base a las referencias y apoyo de vistas isométricas, para el personaje y sus accesorios se modelaron por separado, para luego añadirlos a la malla base del cuerpo del Chasqui, cabe recalcar que para todos los modelos se tuvo mucho énfasis en aplicar el correcto flujo de vértices en zonas articuladas como rodillas, cuellos, entre otros. Posteriormente, se procedió a realizar los cortes UVs para hacer el despliegue del mapa.

Figura 32
Partes del modelo Chaico Chasqui

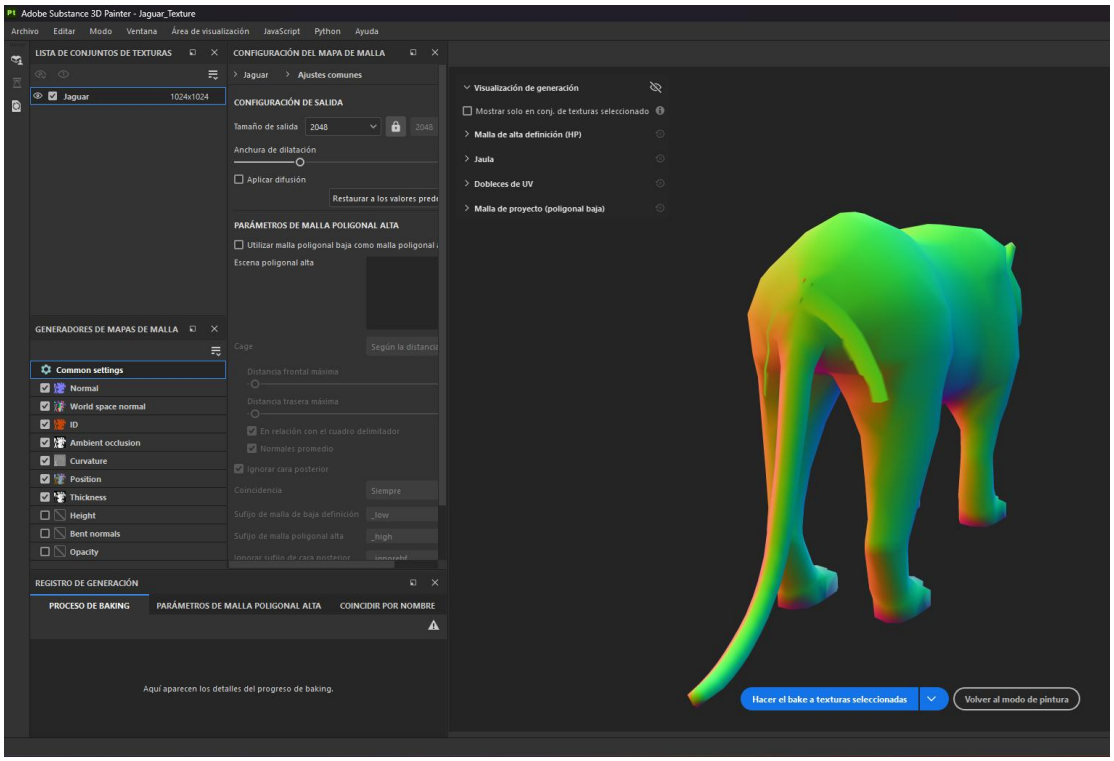
Nota. Despliegue de las piezas del personaje principal Chaico Chasqui (cabello, cuerpo, ojos, ropa, bolso)

Una vez teniendo el personaje con la correcta topología y su mapa de UV desplegado, se pasó el modelo al programa de texturizado Substance Painter, donde se realizó el *bake* de las texturas base, como el mapa de normales y la oclusión ambiental, esto únicamente con el

objetivo de identificar artefactos visuales, en el caso de haberse detectado algún artefacto, se regresaba a Blender a corregir dichos errores, iterando nuevamente hasta tener un modelo limpio, para proceder al texturizado a mano, todas las texturas de los personajes fueron hechas a mano, aprovechando herramientas como las capas de máscaras y la simetría.

Figura 33

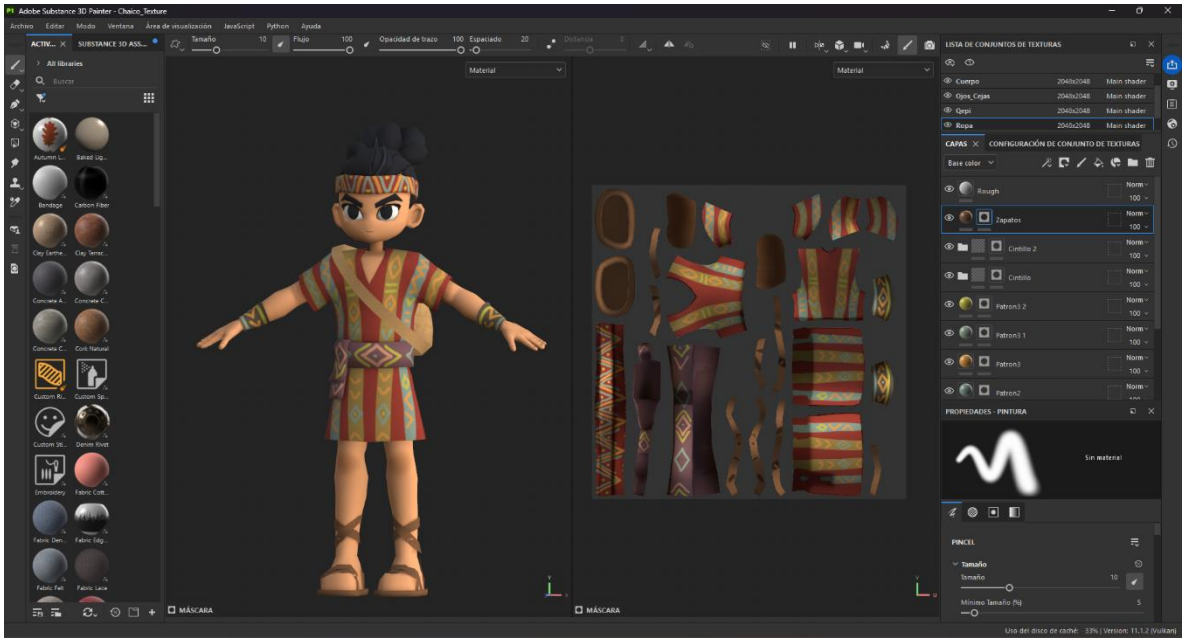
Artefacto en la superficie del jaguar



Nota. Artefacto claramente visible en la zona de la cola del jaguar, lo que proviene de un error en el mapa de UVs

Figura 34

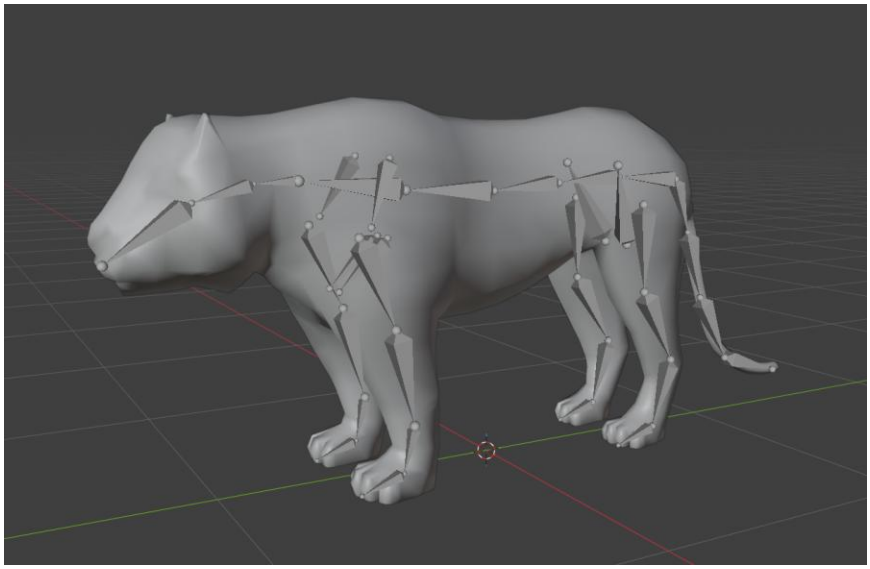
Capas de pintura en Substance Painter



Nota. Flujo de trabajo realizado en Substance Painter, con herramientas como el sistema de capas

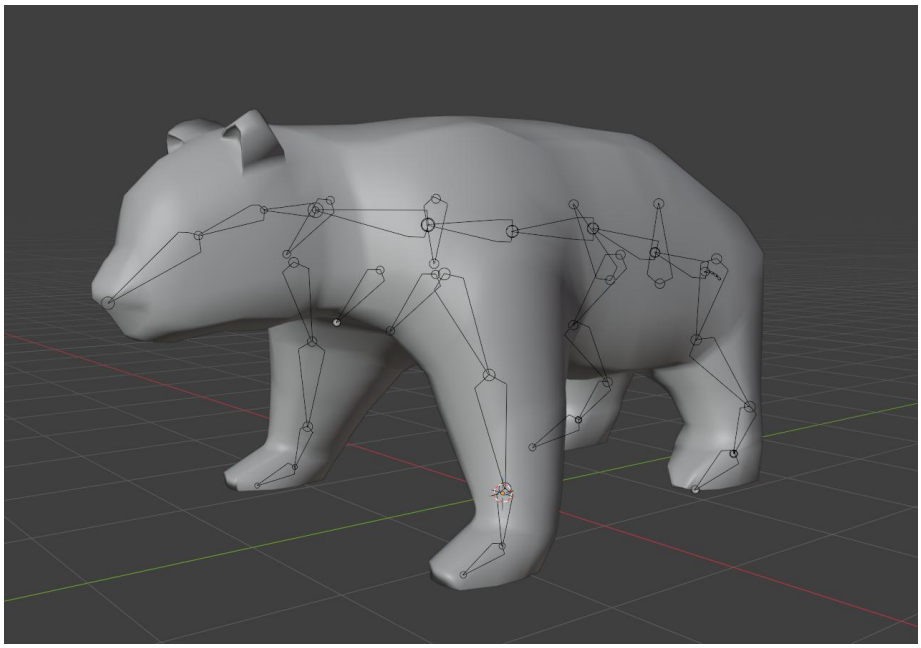
Después de tener todo el modelo visualmente correcto se procedió a realizar el *rigging* dentro de Blender, para esta etapa se usó un *addon* de Blender llamado *Rigify* el cual nos facilita un esqueleto base para humanos, mamíferos de 4 patas y aves, que es justo lo que necesitamos. Se ajustaron los huesos y se aplicó un cálculo automático de pesos, se probó el *rig* final para la exportación y compartir los archivos con mi compañero de animaciones.

Figura 35
Rig de jaguar



Nota. Esqueleto (*Rig*) base de un mamífero cuadrúpedo ajustado a la morfología del jaguar

Figura 36
Rig oso de anteojos



Nota. Esqueleto (*Rig*) base de un mamífero cuadrúpedo ajustado a la morfología del oso de anteojos

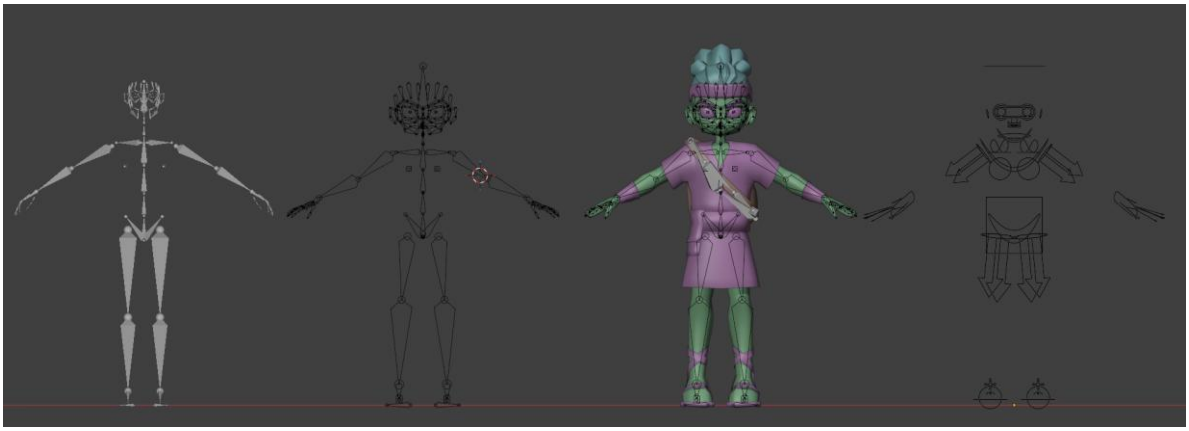
4.5. Preparación de personajes y objetos

Para asegurar el correcto funcionamiento de los modelos en el motor, el proceso de *rigging* se integró directamente en el pipeline del arte técnico.

Como ya se mencionó en el punto anterior, en la etapa de *rigging* se usó un *addon* de Blender llamado *rigify*, el mismo *addon* cuenta con un catálogo de "esqueletos" preestablecidos como el de humano, mamífero cuadrúpedo, pez y ave. El proceso para cada personaje fue el siguiente:

- Para el personaje principal (Chaicó Chasqui) se usó el esqueleto completo de humano, se importó en la escena de Blender y se ajustaron las dimensiones y posiciones de los huesos respetando las proporciones y articulaciones del modelo, después procedí a emparentar los huesos con el modelo, esto se hizo con el cálculo automático de pesos, los errores que se presentaban se ajustaban pintando manualmente los pesos del correspondiente hueso. Una vez todas las deformaciones eran correctas, se generaban los controladores, una de las ventajas de usar *rigify* es que te permite generar controladores para cada esqueleto, estos controladores te permiten mover varias secciones del esqueleto con cinemática inversa (movimiento de los cuerpos), lo cual quiere decir que me permite mover la cadera y el resto del cuerpo se moverá reactivamente.

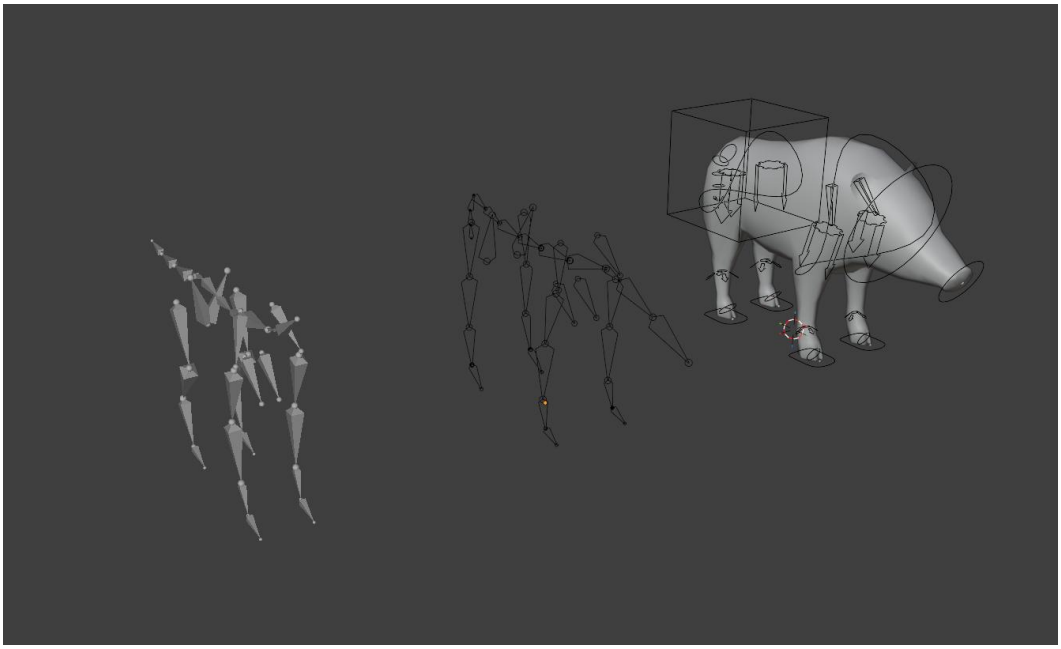
Figura 37
Rig del Chaico Chasqui



Nota, En la figura 37 podemos apreciar el *rig* humanoide base y la diferencia de sus dimensiones con respecto al que se ajustó a la morfología del Chaico Chasqui

- Los animales como el Pecarí, Oso, Jaguar y Cóndor compartieron el mismo flujo, ya que al igual que el humano, había un esqueleto base que se aplicó a estos animales. De igual manera cada esqueleto se ajustó a la morfología del animal, respetando sus dimensiones y articulaciones.

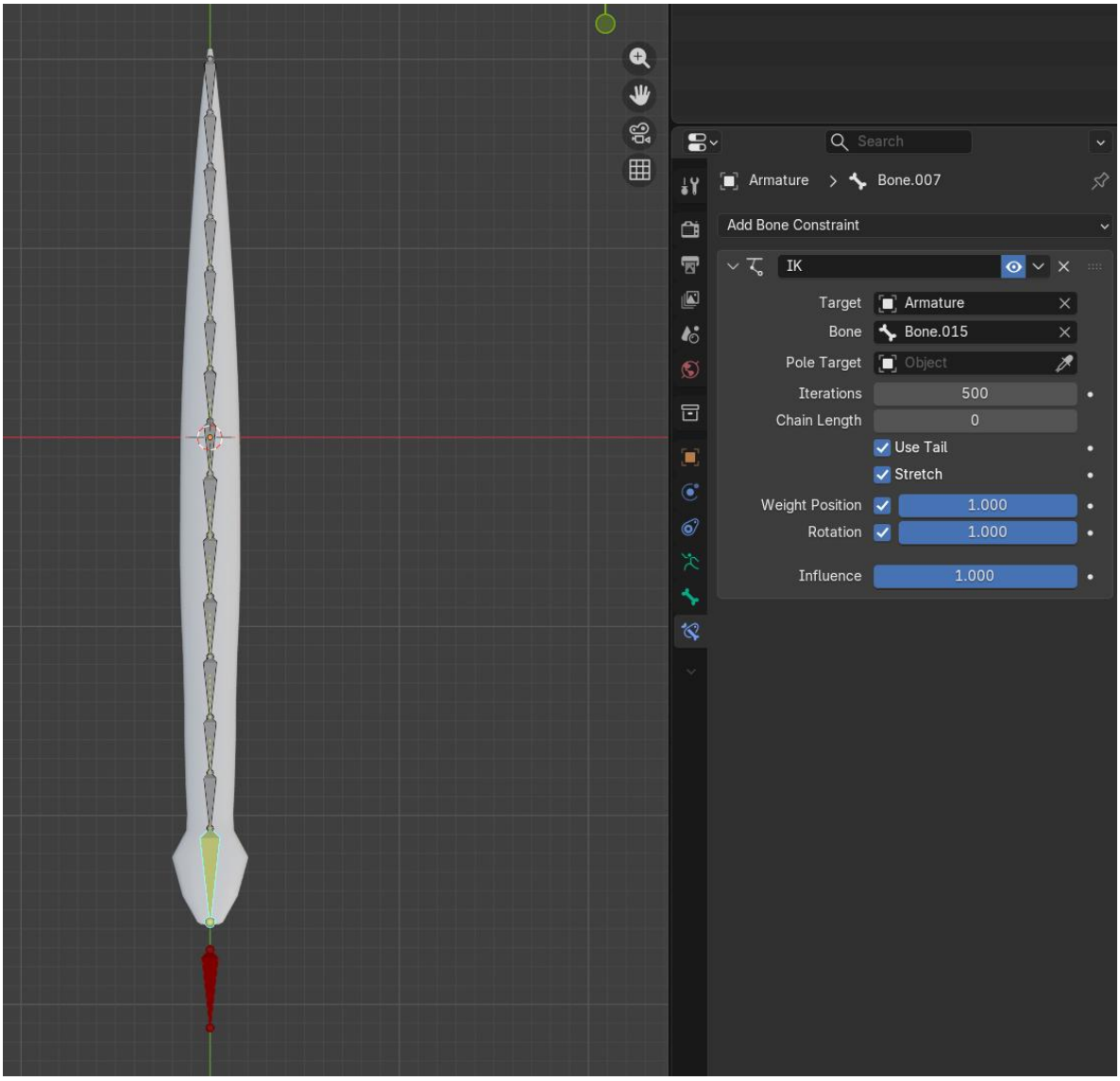
Figura 38
Rig Pecari



Nota, En la figura 38 podemos apreciar el *rig* cuadrúpedo base y la diferencia de sus dimensiones con respecto al que se ajustó a la morfología del pecarí

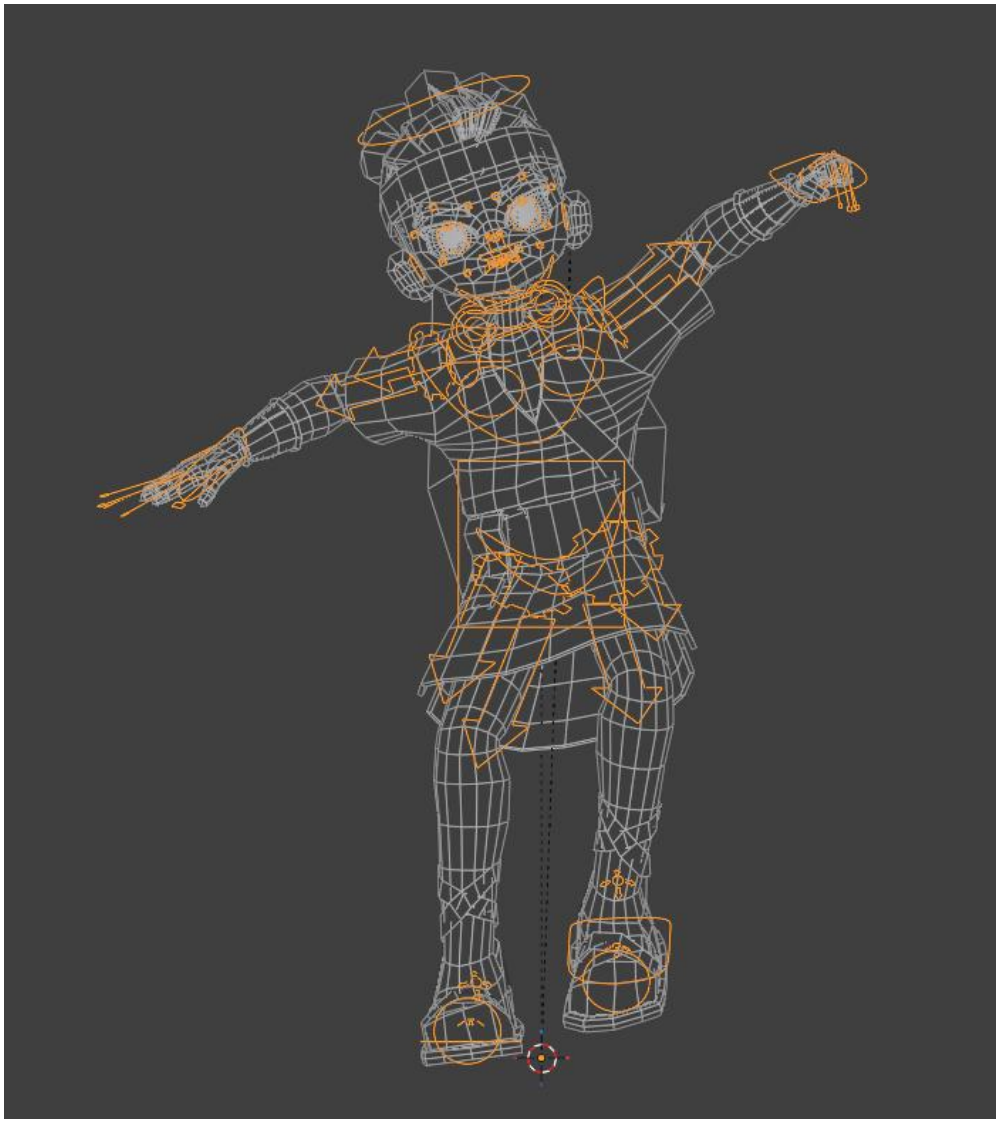
- La serpiente se trabajó aparte, ya que el *addon rigify* no cuenta con un esqueleto para reptiles de ese tipo. Pero al ser una serpiente, se optó por construir una cadena de huesos y manualmente ajustar un controlador para la cinemática inversa, esta configuración dio como resultado un esqueleto con un único controlador en la cabeza.

Figura 39
Rig matacaballos



Nota. *Rig* manual para la serpiente, una cadena de huesos con un controlador IK en la cabeza.

Figura 40
Muestra de flexión y topología del Chaico-Chasqui



Nota. Topología del Chaico Chasqui en pose muestra para flexión de articulaciones.

4.6. Materiales, texturas, iluminación y renderizado

El proyecto prescindió completamente de cálculos de iluminación en tiempo real y sombras dinámicas. Todo el motor de renderizado URP (Universal Render Pipeline) fue configurado para utilizar materiales tipo *Unlit* (sin iluminación física). La ilusión de

iluminación se desarrolló mediante *shaders* personalizados construidos en *Unity Shader Graph.*

Figura 41
Comparativa de materiales

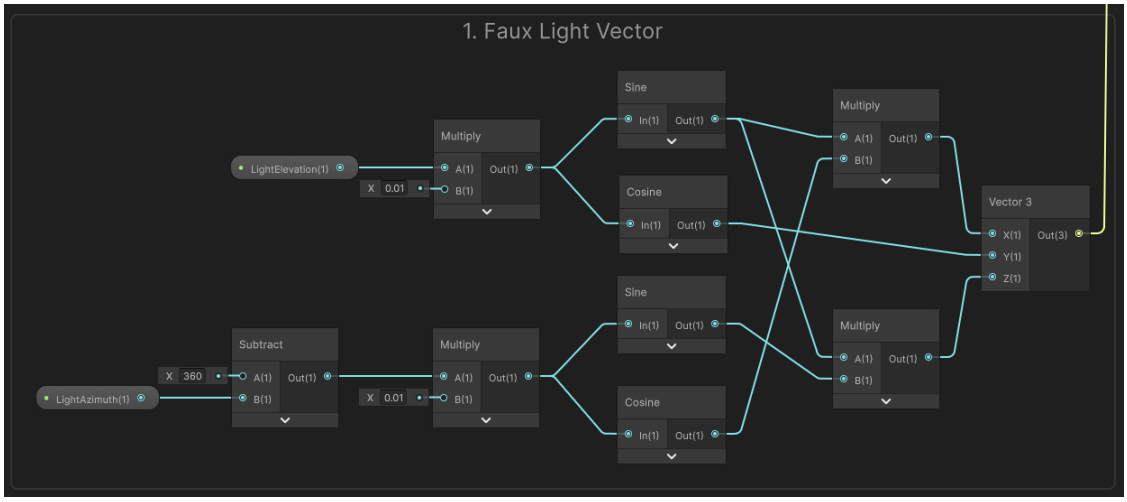
Nota. Comparativa entre un material que se renderiza con el *shader* por defecto de Unity y el material que se renderiza con el *cellshader*

El sombreado principal denominado *CellShader,* se aplicó al personaje, animales y entorno. Fue desarrollado en cuatro bloques de cálculo.

1. *Faux light vector*: Remplaza la iluminación base de Unity. Utiliza variables públicas como *"LightElevation"* y *"LightAzimuth"* conectadas a funciones trigonométricas

(seno y coseno) para generar un vector direccional de luz global falsa, personalizable desde el editor.

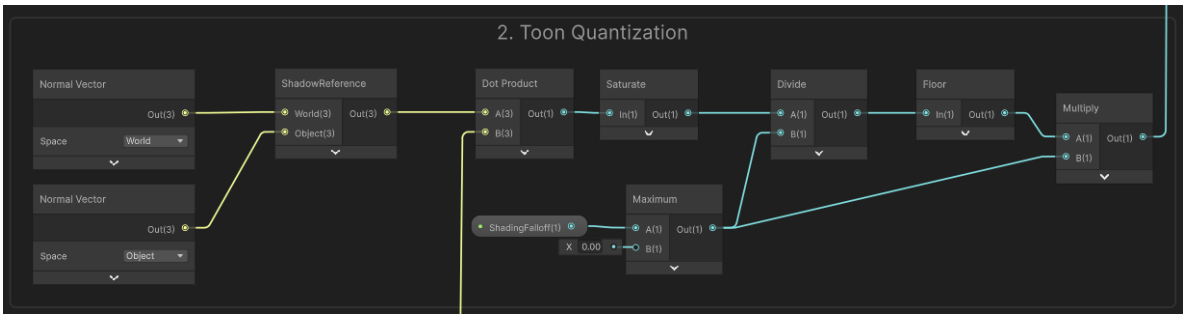
Figura 42
Cellshader bloque 1. Faux Light Vector



Nota. Interfaz de nodos en *Unity Shader Graph* correspondientes al bloque *Faux Light Vector*, encargado de simular matemáticamente la dirección global de la luz.

2. *Toon quantization*: Calcula el efecto de *CellShading* (sombreado plano). Calcula el producto escalar (*Dot Product*) entre las sombras del modelo y el vector dirección del bloque uno. Pasa por operaciones de saturación y división, limitadas por una función matemática (*Floor*), ajustada por la variable "*ShadingFalloff*".

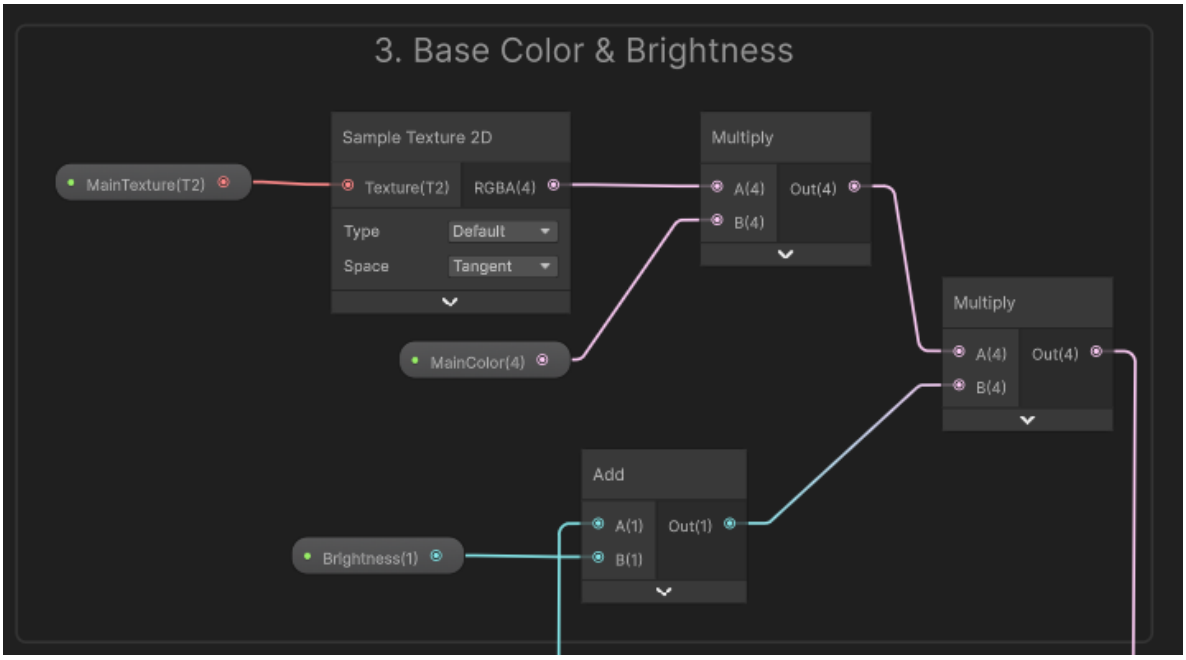
Figura 43
Cellshader bloque 2. Toon Quantization



Nota. Configuración del bloque *Toon Quantization* dentro del *CellShader*, utilizado para calcular el sombreado plano y limitar los gradientes de iluminación.

3. *Base color & brightness:* Inserta la información visual mediante variables simples como *"MainTexture", "MainColor" y "Brightness"*, sumadas y multiplicadas al flujo visual desarrollado hasta este punto.

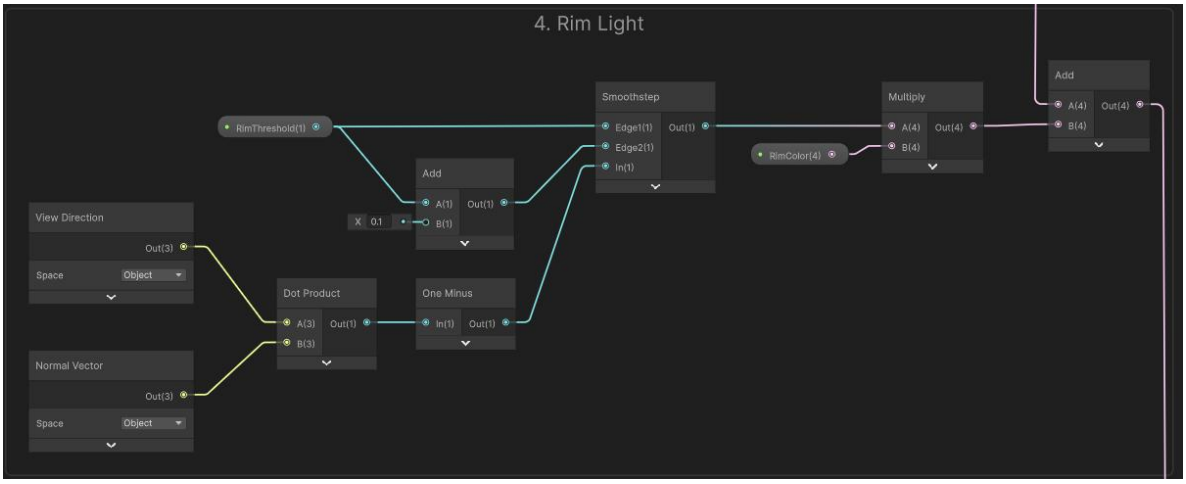
Figura 44
Cellshader bloque 3. Base Color & Brightness



Nota. Bloque *Base Color & Brightness* del *CellShader*, ilustrando la integración de las texturas y el control de luminosidad del material.

4. *RimLight:* Resalta la silueta del modelo calculando la dirección de la cámara (*View Direction*) respecto a las normales mediante un nodo *Dot Product* y un *Smoothstep* controlado por las variables *"RimThreshold" y "RimColor"*.

Figura 45
Cellshader bloque 4. Rim Light



Nota. Operaciones matemáticas del bloque *Rim Light*, diseñadas para resaltar los bordes de la silueta en base a la dirección de la cámara.

Un segundo *shader* que se utilizo fue el *OutlineShader,* para resaltar casillas e interactuables. Su arquitectura simple consta de dos variables, *"Outline Thckness"* se multiplicó por el nodo *Position* en espacio de objeto y se conectó directamente con la salida de vértice, y *"Outline Color"* se conectó directamente a la salida *Base Color*.

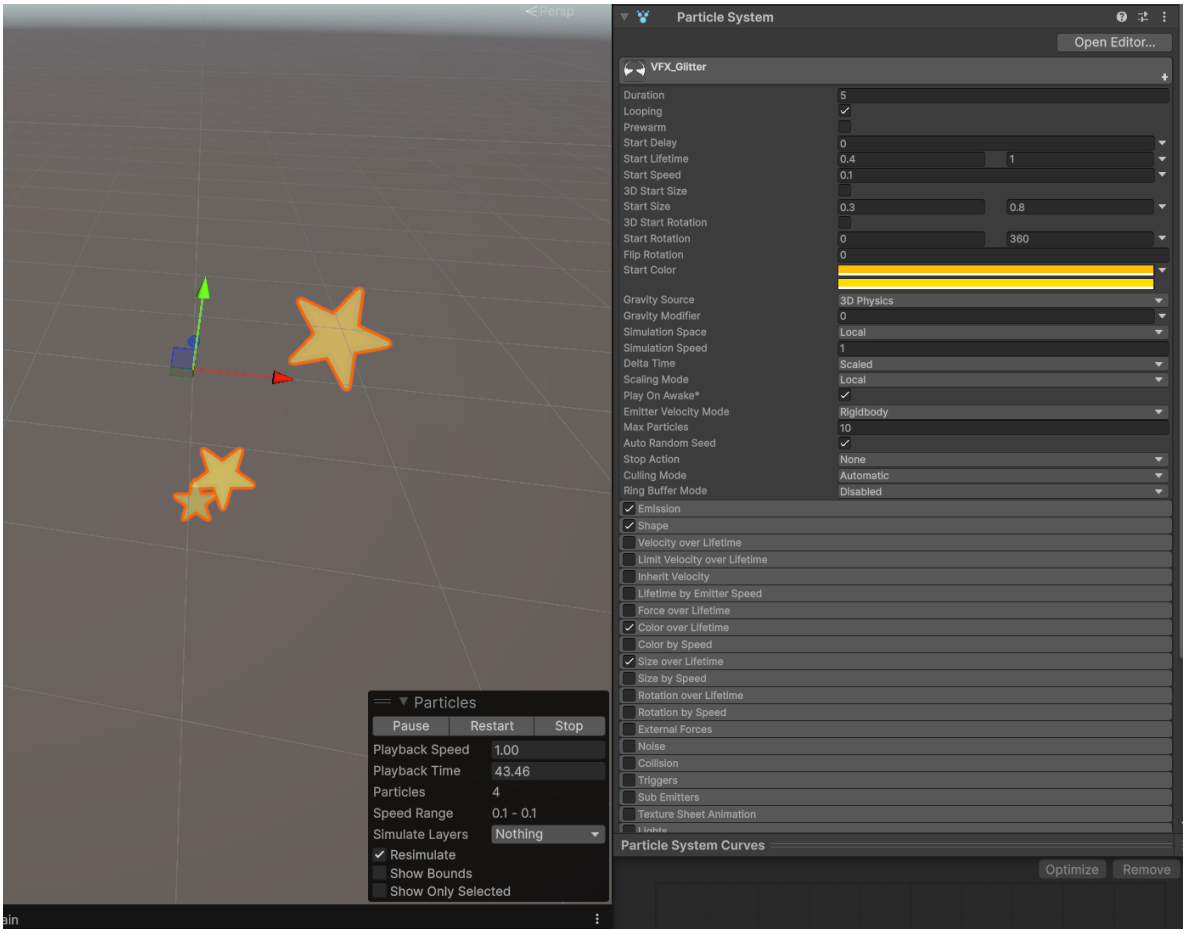
4.7. Creación de efectos visuales

El diseño de sistemas de partículas se enfocó para dispositivos de bajo rendimiento. Los VFX (efectos visuales) se clasificaron en generales y habilidades de cartas:

VFX Generales:

- *Glitter:* Emisión en bucle con un límite máximo de 10 partículas, emitiendo 5 por ciclo de reproducción, utilizando un emisor de forma esférica. Material URP *Unlit Particle* con modo de fusión aditivo. Textura en escala de grises de 64x64 pixeles ilustrada en Affinity, coloreada dinámicamente por la función *Start Color*.

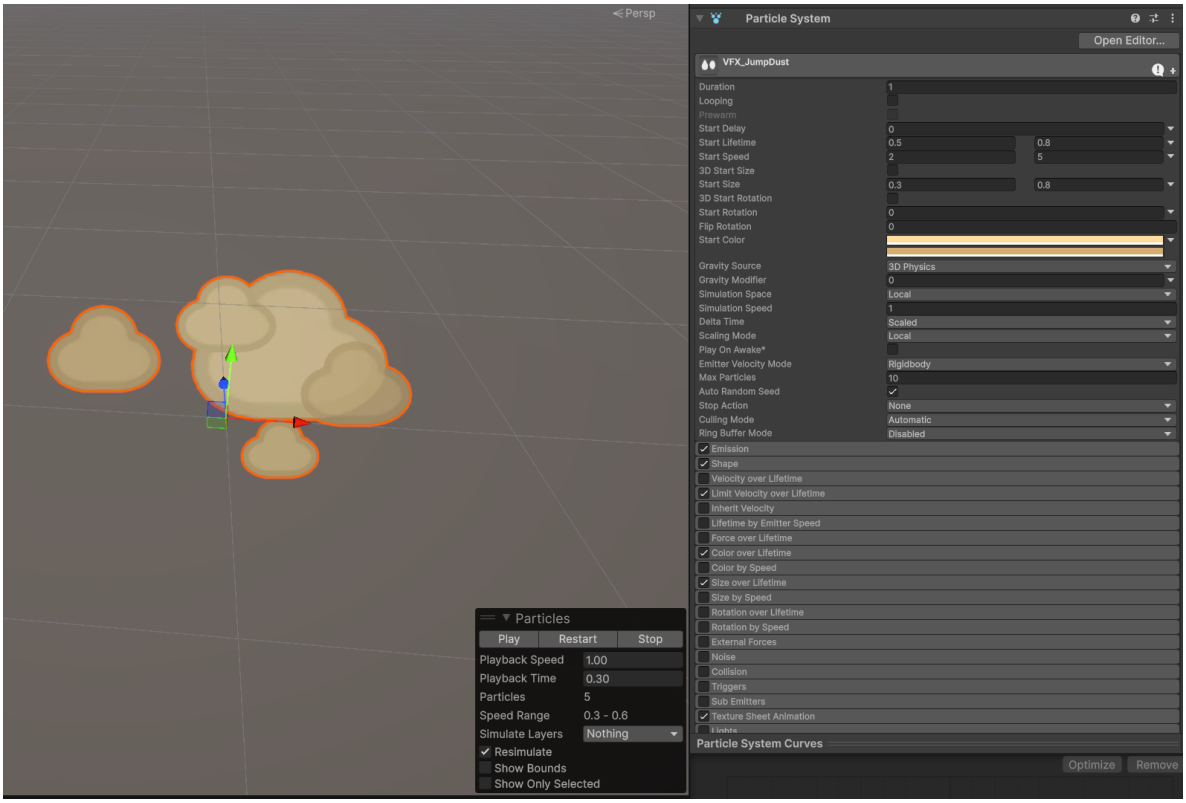
Figura 46
Efecto visual: Glitter



Nota. Parámetros de emisión e interfaz del motor gráfico para la creación del sistema de partículas continuo *Glitter*.

- *JumpDust:* Emisor semiesférico sin bucle, limitado a 10 partículas. Utiliza un material con fusión de transparencia y una textura base de nube caricaturizada de 64x64 pixeles.

Figura 47
Efecto visual: JumpDust



Nota. Estructura y simulación del sistema de partículas *JumpDust* utilizado para proyectar nubes de polvo en interacciones físicas.

- Portal: En lugar de un sistema de partículas, se desarrolló una malla hexagonal con un *shader* personalizado. Utiliza una función personalizada para generar el recorte perimetral del hexágono y se anima mediante nodos *Twirl* y *Voronoi Noise* multiplicados por el nodo *Time*.

  float2 p = abs(UV – 0.5); Out = max(p.x * 0.866025 + p.y * 0.5, p.y);

  El desglose técnico de qué significa cada valor y por qué son necesarios:

  1. float2 p = abs(UV - 0.5); (Centrado y Simetría)

- UV - 0.5: Las coordenadas UV nativas van de un valor de 0 a 1 (donde 0.5, 0.5 es el centro exacto de la cara). Al restarle 0.5, se desplaza el punto de origen de las coordenadas (0,0) exactamente al centro de la malla.
- abs(): La función de valor absoluto (absolute) convierte cualquier valor negativo en positivo. Al aplicar esto a las coordenadas centradas, se crea una simetría en espejo en los 4 cuadrantes (X, Y). Esto significa que el motor gráfico solo necesita calcular matemáticamente una cuarta parte del hexágono (un solo cuadrante) y el resultado se refleja automáticamente para formar la figura completa. Es una técnica optimizada para ahorrar cálculos en la GPU.

2. Los valores "mágicos": 0.866025 y 0.5 (Trigonometría espacial)

Un hexágono regular se compone de 6 triángulos equiláteros, lo que significa que sus paredes inclinadas tienen ángulos estrictos basados en incrementos de 30 y 60 grados. Para trazar esas líneas diagonales en el espacio 2D, se utilizan las constantes trigonométricas de un ángulo de 30°.

- 0.866025: Este número es el resultado matemático de la raíz cuadrada de 3 dividida para dos (√3 / 2), lo cual es exactamente el Coseno de 30 grados.
- 0.5: Este número es exactamente el Seno de 30 grados.

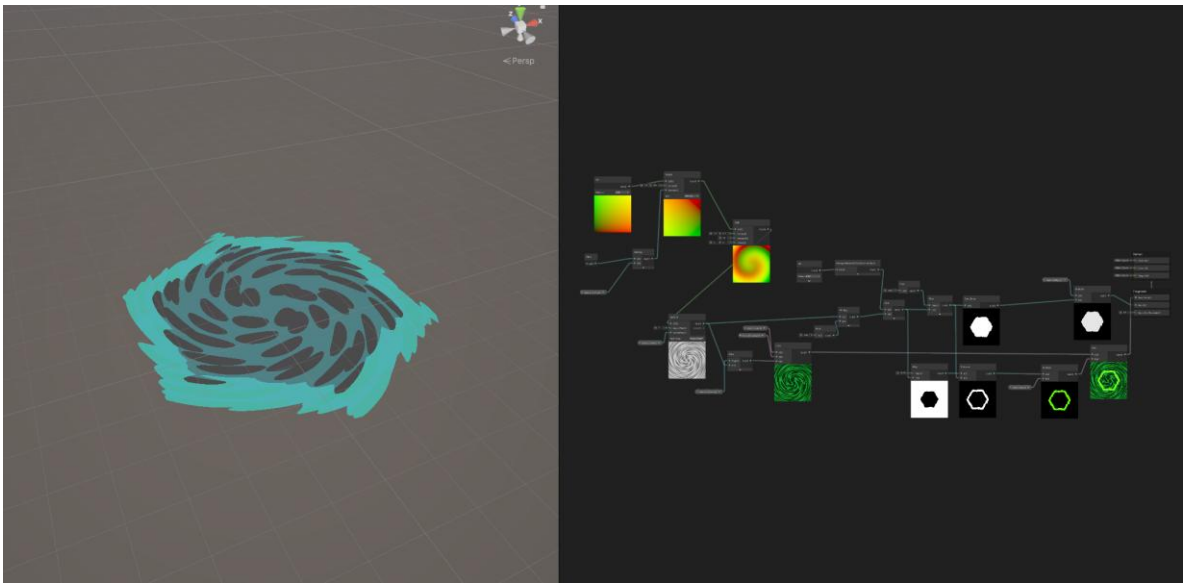
Al multiplicar las coordenadas X e Y por estos valores trigonométricos (p.x * 0.866025 + p.y * 0.5), el shader está calculando y trazando la pendiente exacta de las paredes diagonales laterales del hexágono.

3. Out = max(..., p.y); (La intersección booleana)

- p.y: Esta parte de la ecuación representa simplemente las paredes planas horizontales (el "techo" y el "piso" del hexágono).

- max(): La función max compara dos valores y devuelve el más grande. En la programación matemática de formas (SDF), el nodo max funciona como una intersección. Combina los planos de las paredes diagonales (la primera parte de la fórmula) con los planos horizontales (p.y). Al cruzarse, se "cortan" mutuamente, limitando la forma geométrica y dando como resultado final la silueta de un hexágono perfecto.

Figura 48
Efecto visual: Portal

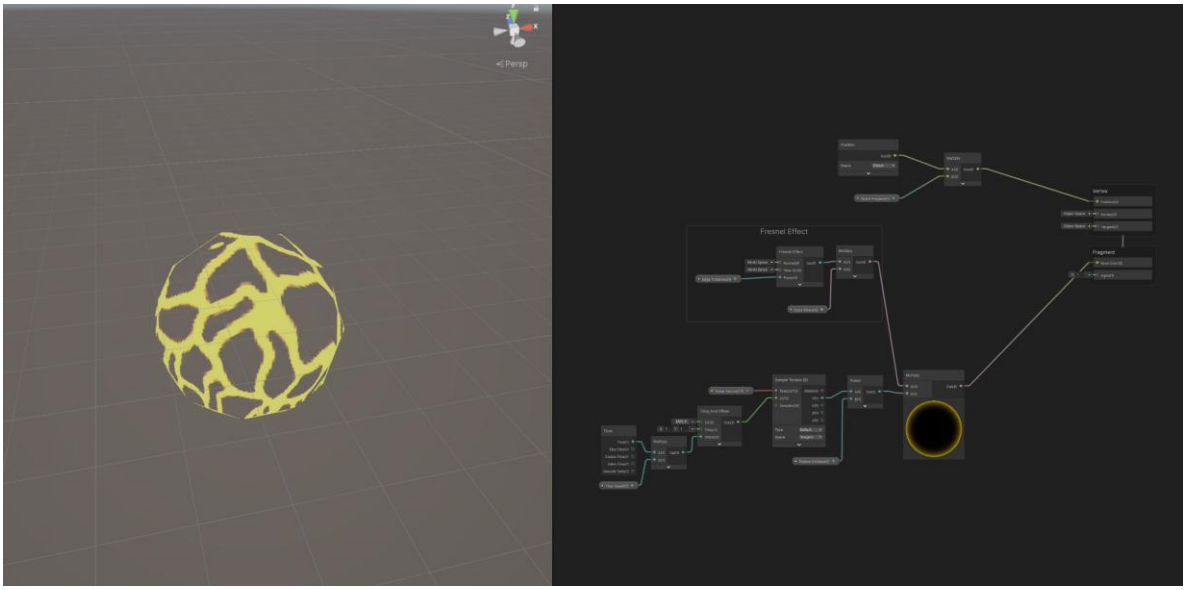


Nota. Apariencia y configuración lógica del efecto visual del portal, generado mediante una malla hexagonal modificada por nodos *Twirl* y ruido *Voronoi*.

VFX de Dioses:

- Inti (Escudo): *Shader* que multiplica el nodo *Time* por la variable “FlowSpeed”, conectado a un *Tilling and Offset* que desplaza una textura de ruido. Utiliza un efecto *Fresnel* para mantener la transparencia central.

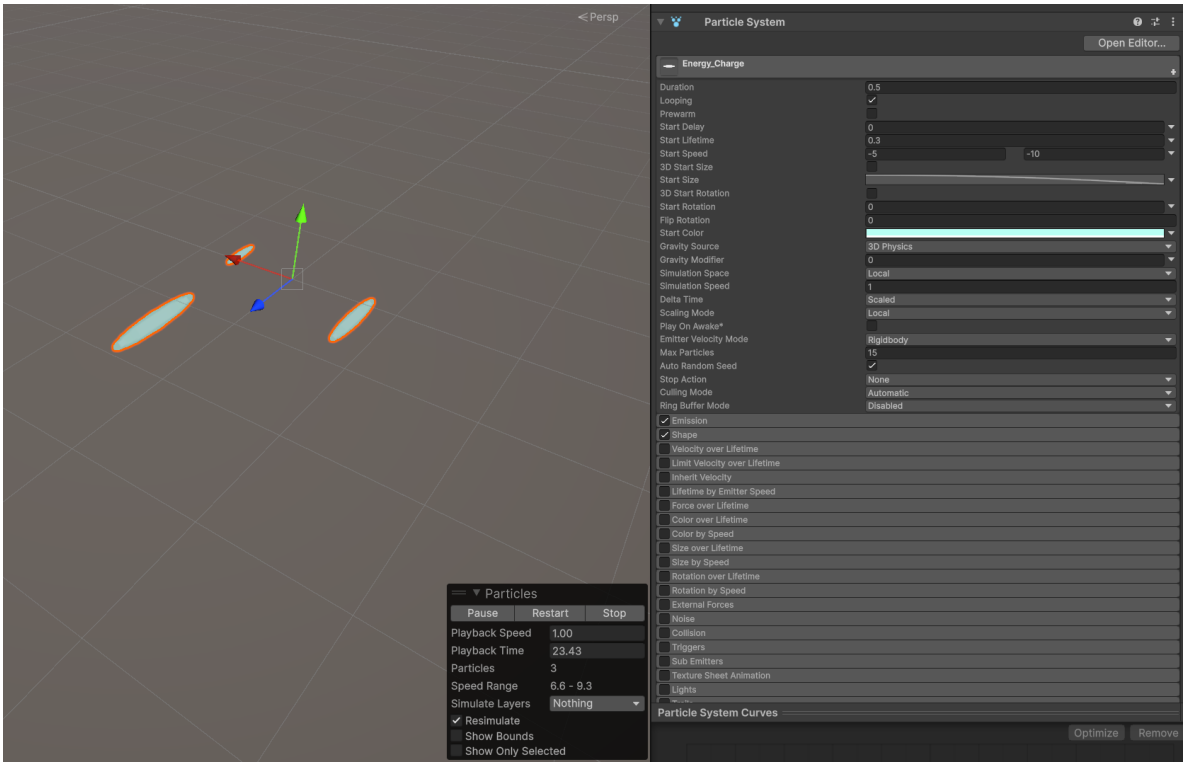
Figura 49
Efecto visual: Inti



Nota. Simulación visual de la habilidad defensiva de la carta Inti (Escudo), producida mediante mapas de ruido dinámicos y efecto Fresnel.

- Kon (Viento): Partícula en bucle emitidas desde una forma lineal. El renderizado de partículas fue configurado en *Stretched Billboard* con una escala de longitud de 1.5, usando una textura circular achatada de 64x64 pixeles para emular una fuerte corriente de viento.

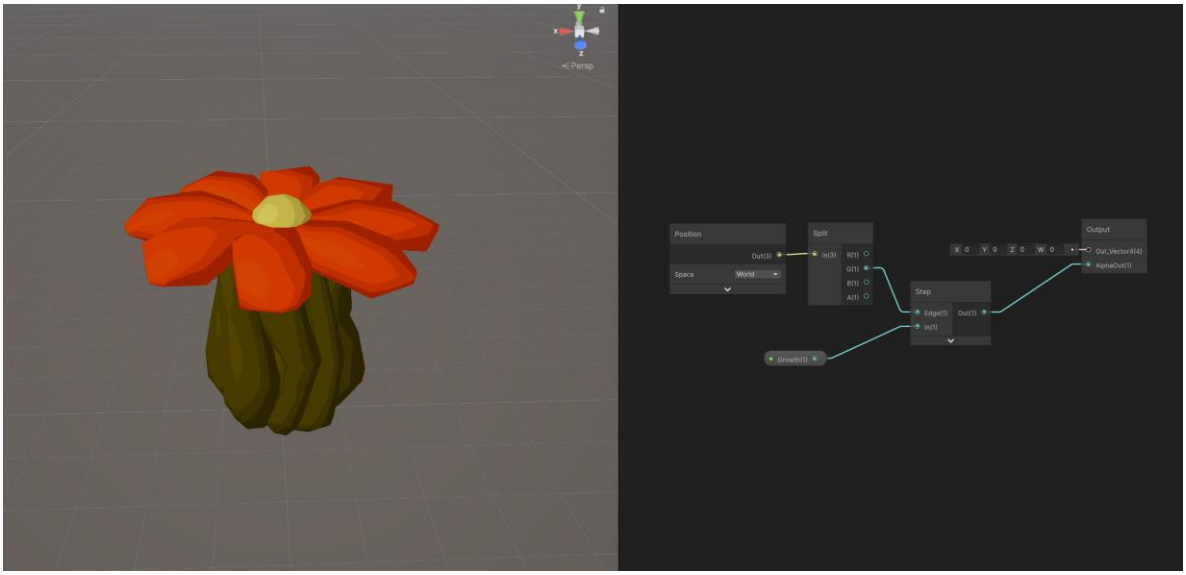
Figura 50
Efecto visual: Kon



Nota. Representación de corrientes de aire aceleradas para el efecto de la deidad Kon, lograda con emisores lineales y partículas estiradas.

- Pachamama (Crecimiento de raíz): Se implemento con un *Subgraph* añadido al *CellShader* principal. Se uso el nodo *Position*, extrayendo el eje vertical (Y) mediante el nodo *Split*. Se compara contra una variable “Growth” utilizando un nodo *Step*, dando la capacidad de animar el crecimiento directamente en el canal alfa.

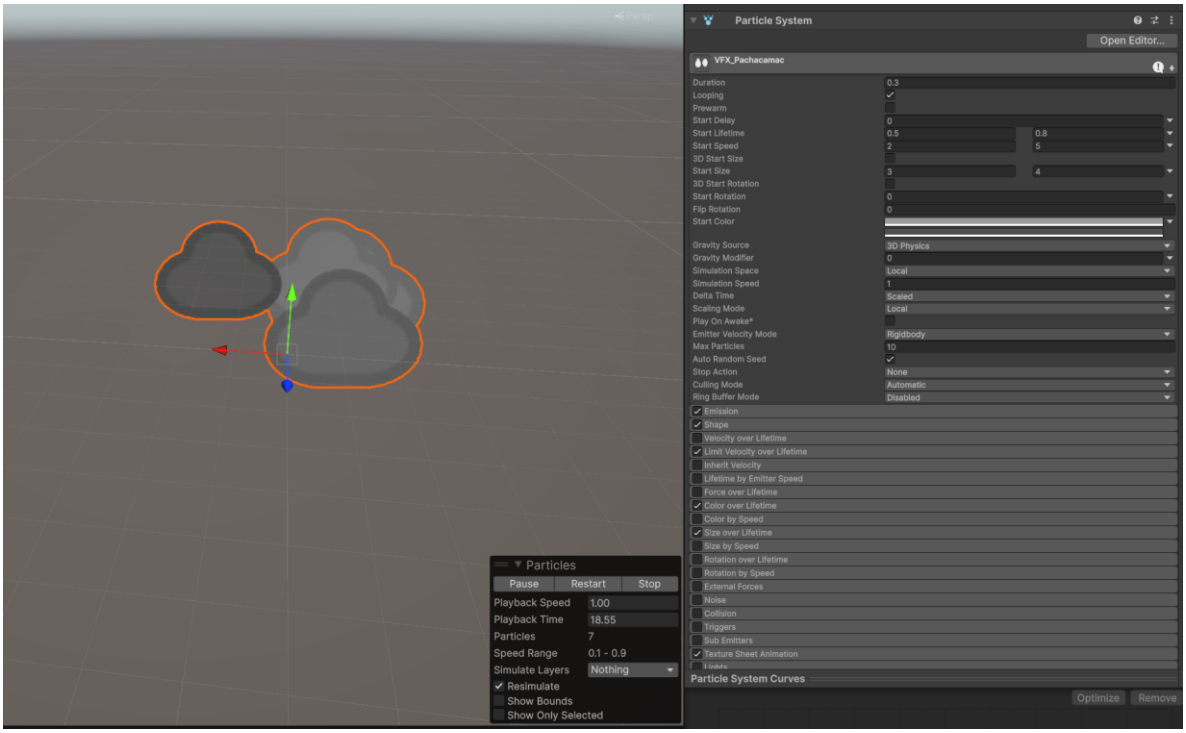
Figura 51
Efecto visual: Pachamama



Nota. Modificación del canal alfa en el *CellShader* para generar la animación de crecimiento orgánico de la carta Pachamama.

- Pachacamac (Terremoto): Se reutilizo la lógica, función y material del VFX JumpDust, ajustando la escala, duración y colores, para conseguir el efecto visual deseado de terremoto.

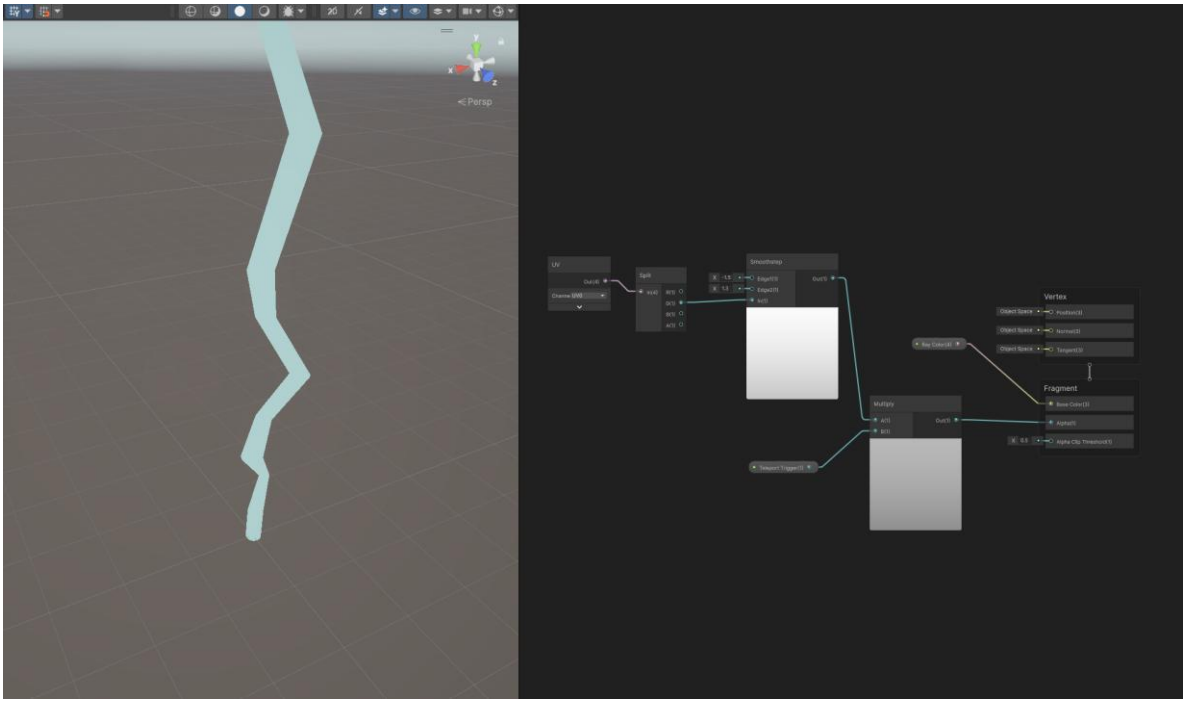
Figura 52
Efecto visual: Pachacamac



Nota. Elementos de partículas volumétricas sobrepuestas orientadas a simular sismos para el efecto de la carta Pachacámac.

- Illapa (Rayo): Se modelo una geometría tridimensional cilíndrica adaptada a la dimensión del hexágono. El *shader* extrae la coordenada UV vertical a través de un nodo *Split* y un *Smothstep*, multiplicado por una variable de animación conectada a a la opacidad del canal alfa.

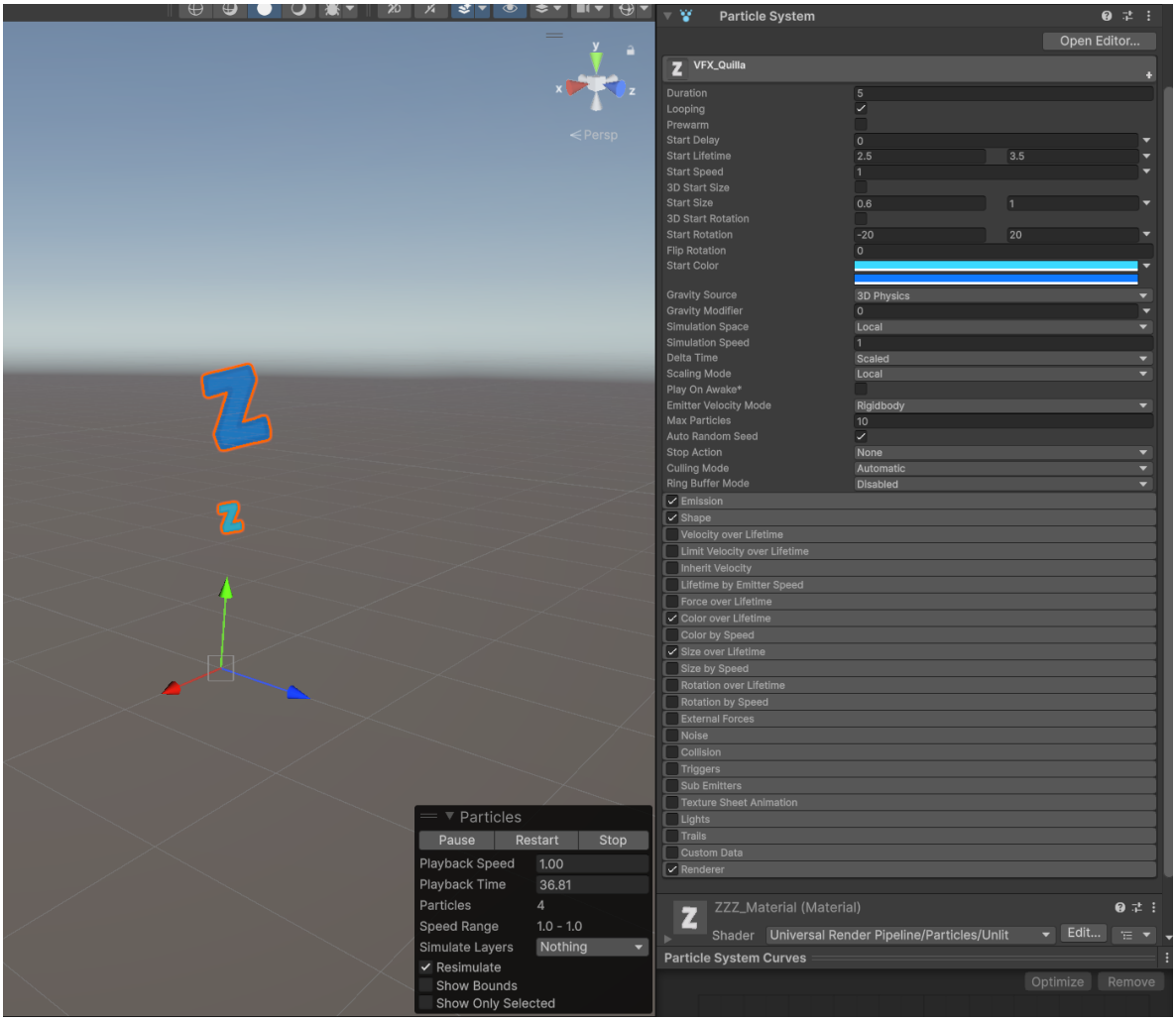
Figura 53
Efecto visual: Illapa



Nota. Construcción geométrica y sombreado interpolado del efecto eléctrico asociado al poder de traslación de Illapa.

- Quilla (Sueño): Emisor de caja (*Box*) emitiendo partículas con textura de "Z" de 64x64 pixeles. Se utilizó los módulos *color over lifetime* para controlar el desvanecimiento y *size by lifetime* para la escala caricaturesca.

Figura 54
Efecto visual: Quilla



Nota. Distribución de texturas tipográficas y manipulación de tamaño a lo largo del tiempo para ilustrar el efecto de sueño de Quilla.

4.8. Integración de recursos visuales en el motor

Todos los modelos tridimensionales en formato estándar (.fbx) fueron importados al motor Unity. El desarrollo de la lógica visual requirió ajustar las escalas de importación predeterminadas. Cada elemento del entorno se configuro como un *Prefab* modular Estos *Prefabs* quedaron listos para ser utilizados en el sistema de generación procedural, asegurando que todos los niveles instanciaran el mismo material en memoria.

El proceso estructurado para crear un *Prefab* modular se ejecutó en los siguientes pasos:

1. Ensamblaje de la pieza base

El proceso comenzó importando el modelo tridimensional (archivo .fbx) a la jerarquía de Unity. A este modelo se le asignó un "contenedor vacío" (*Empty GameObject*), el cual funcionó como su núcleo estructural. Dentro de la escena, se le aplicó los materiales optimizados (que nacen del *CellShader*) y se le añadieron los componentes lógicos necesarios, como los colisionadores (*Colliders*) para que el Chasqui pudiera saltar sobre los hexágonos sin caer al vacío.

2. Estandarización del punto de anclaje (*Pivot*)

Para que una pieza sea verdaderamente modular, debe encajar perfectamente con las demás sin dejar huecos. Esto se logró calibrando el punto de origen (*Pivot*) de cada hexágono en el centro exacto de su base. De esta manera, cuando el código del juego ordena colocar un hexágono junto a otro, el motor sabe exactamente a qué distancia ubicarlos para que sus bordes se toquen con precisión milimétrica, formando un tablero continuo.

3. Conversión a *Prefab*

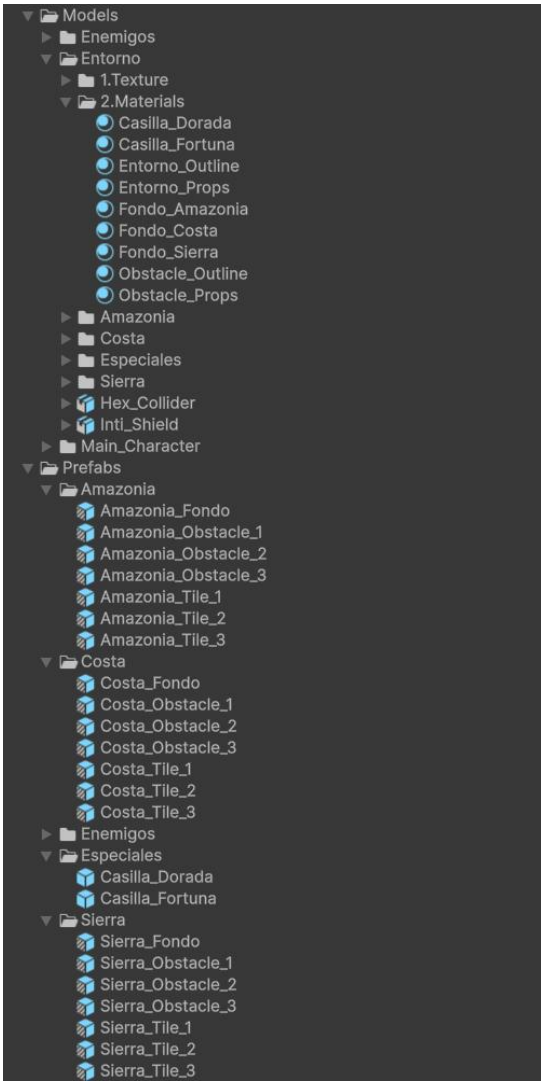
Una vez que el modelo contaba con su geometría, colisiones, materiales y su punto de origen estandarizado, el objeto completo se arrastró desde la ventana de jerarquía visual hacia una carpeta de archivos llamada "Prefabs" dentro del proyecto. Al hacer esto, el icono del objeto cambia a color azul, indicando que el molde maestro ha sido creado con éxito y salvaguardado en el disco duro.

4. Instanciación y generación procedural

Al contar con una biblioteca de *Prefabs* ordenados por biomas (Costa, Sierra, Amazonía), el equipo de programación pudo alimentar el generador procedural del juego. Cuando un jugador inicia una partida, el algoritmo de generación llama a estos *Prefabs* y los ensambla en posiciones predeterminadas.

Todos los modelos importados se organizaron dentro de una jerarquía estándar, guardándose dentro de una capeta denominada “Models” donde se albergaron dos carpetas generales denominadas “Texture” y “Material”, después cada modelo del entorno fue clasificado por el nivel al que pertenecía, esta misma clasificación se migro a la carpeta y desarrollo de los *prefabs*.

Figura 55
Jerarquía de carpetas en Unity



Nota. Estructura del árbol de directorios en Unity, mostrando la clasificación estandarizada de modelos, *prefabs* y materiales.

4.9. Optimización gráfica

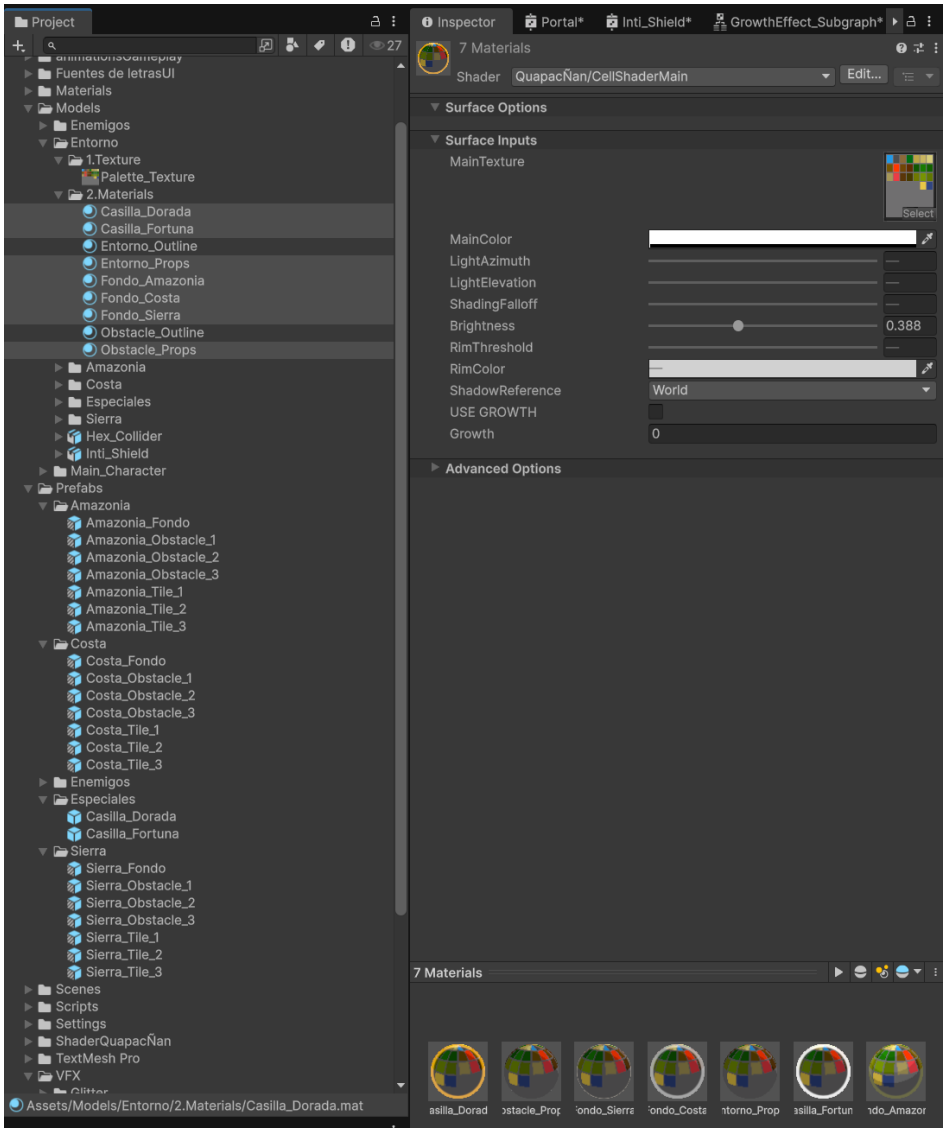
Para conseguir equilibrar la calidad y estética con el consumo eficiente del procesador grafico (GPU), en el pipeline se implementó técnicas agresivas de optimización.

- Consolidación de materiales: El uso de una textura atlas unificada para todas las casillas y obstáculos del juego mitigo el cuello de botella más crítico del renderizado.

Se eliminó la duplicación de materiales por cada hexágono instanciado en el sistema de generación procedural, disminuyendo drásticamente los cambios de estado en la tarjeta gráfica.

Figura 56

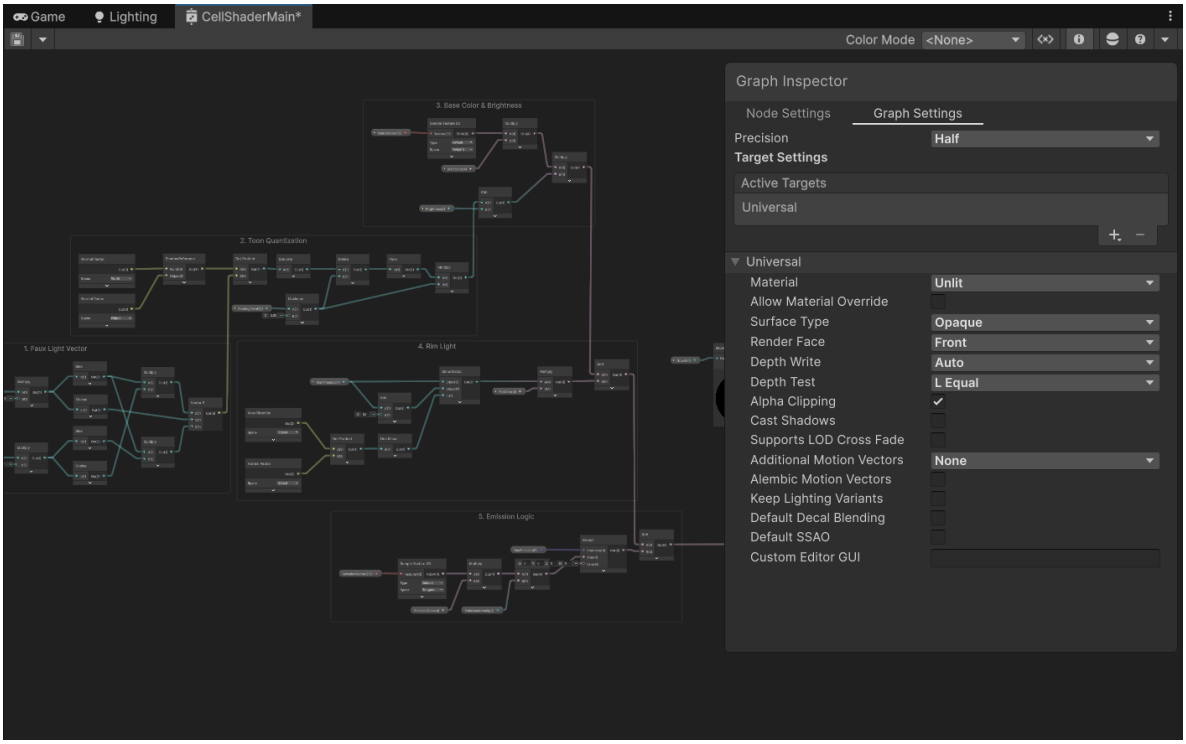
Replicabilidad de textura atlas



Nota. Asignación del atlas de texturas dentro del inspector de Unity para el material unificado de los elementos del entorno.

- Reducción de precisión: Todos los *shader graph* fueron configurados para ejecutar sus cálculos utilizando precisión "*half*" (16 bits) en lugar de "*float*" (32 bits), aligerando la carga de la GPU sin perdida perceptible de calidad visual.

Figura 57
Configuración limitada de shaders



Nota. Ajustes del inspector del motor gráfico limitando la precisión de cálculo de los *shaders* a 16 bits (*Half*) para garantizar optimización.

4.10. Software, formato y organización de recursos

- Modelado y *Rigging:* Blender, modelos exportados en formato (.fbx).
- Diagnóstico de normales y Texturizado: Adobe Substance Painter, texturas exportadas en 1024 x 1024 pixeles en formato (.png)
- Sprites y atlas texture affinity, resoluciones exportadas a potencias de dos como 64x64 y 128x128 pixeles en formato (.png)

- Motor de renderizado: Unity 6, con motor de renderizado URP.

4.11. Validación técnica de recursos visuales

Las pruebas analíticas ejecutadas mediante la herramienta *Unity Profiler* demostraron la eficiencia del entorno optimizado:

- Llamadas de dibujado (*draw calls* - *batches*): La consolidación de materiales en un atlas redujo masivamente los *batches* de 4630 a 730 estables, lo que establece 60 FPS en el *hardware* objetivo.
- Consumo de VRAM: La optimización estructural derivo en un uso de memoria grafica de 0.54 GB (7% de la distribución total). Aunque el volumen base refleja la estructura, la eficiencia radica en que este paquete de memoria es estático y no escala exponencialmente al aumentar el tamaño del mapa.
- Carga de CPU en VFX: El prescindir de físicas y el límite estricto de partículas redujo el tiempo de procesamiento de los hilos lógicos "ParticleSystem.Update" y "ParticleSystem.WaitForPreviousRenderingToFinish" de 0.14 ms a un margen normal de 0.0 ms (1 *call*).

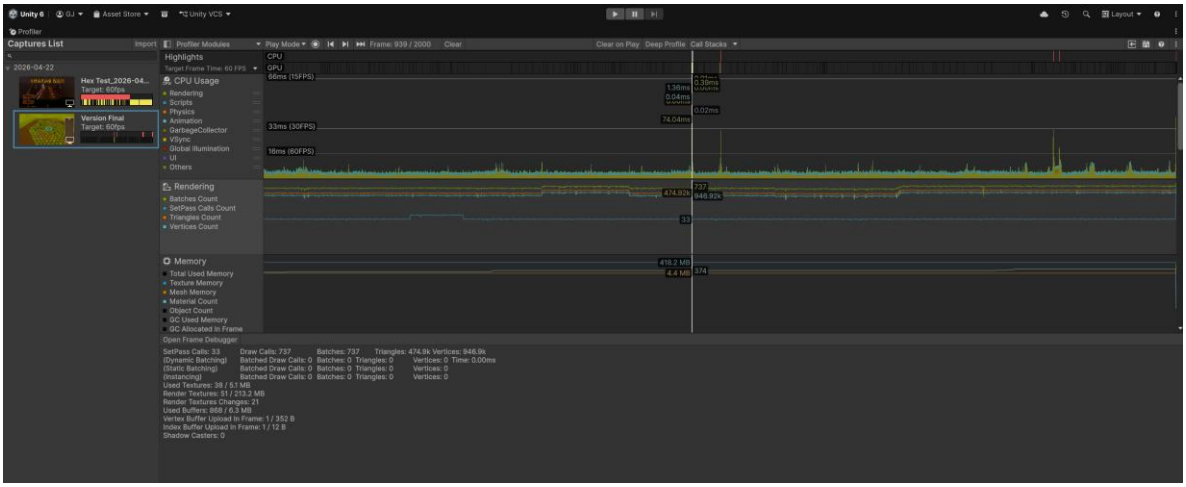
Tabla 3
Tabla comparativa de rendimiento

Pre-optimización	**Post-optimización**
Batches = 4630	*Batches* = 730
Tiempo CPU = 0.14 ms	Tiempo CPU = 0.0 ms

Nota. Comparación de métricas de rendimiento extraídas de *Unity Profiler*, contrastando el número de lotes de renderizado (*batches*) y el tiempo de procesamiento de la CPU antes y después de implementar las técnicas de optimización gráfica en los modelos y efectos visuales.

Figura 58

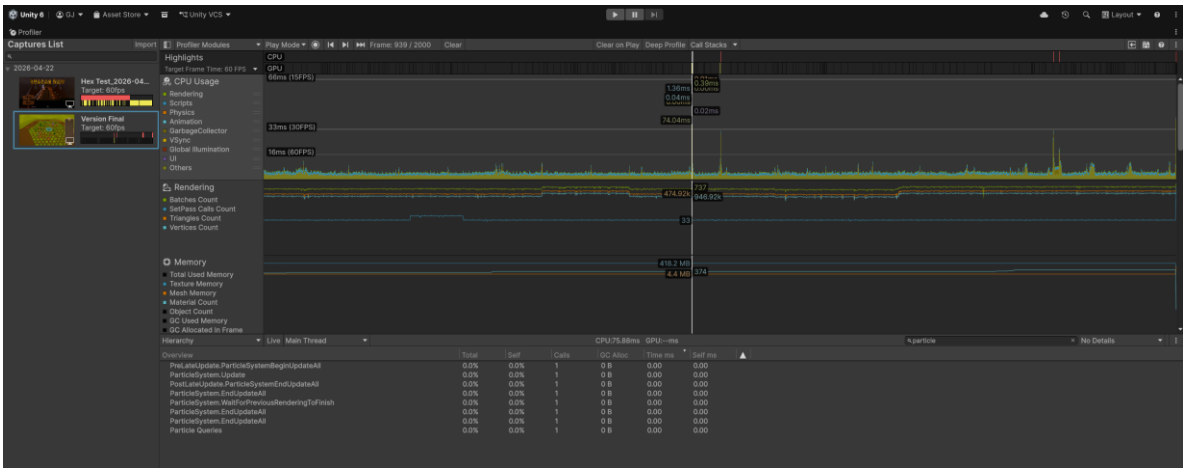
Análisis de procesamiento grafico post-optimización



Nota. Métricas de rendimiento registradas en *Unity Profiler* que evidencian la drástica caída en el número de lotes de renderizado tras la optimización.

Figura 59

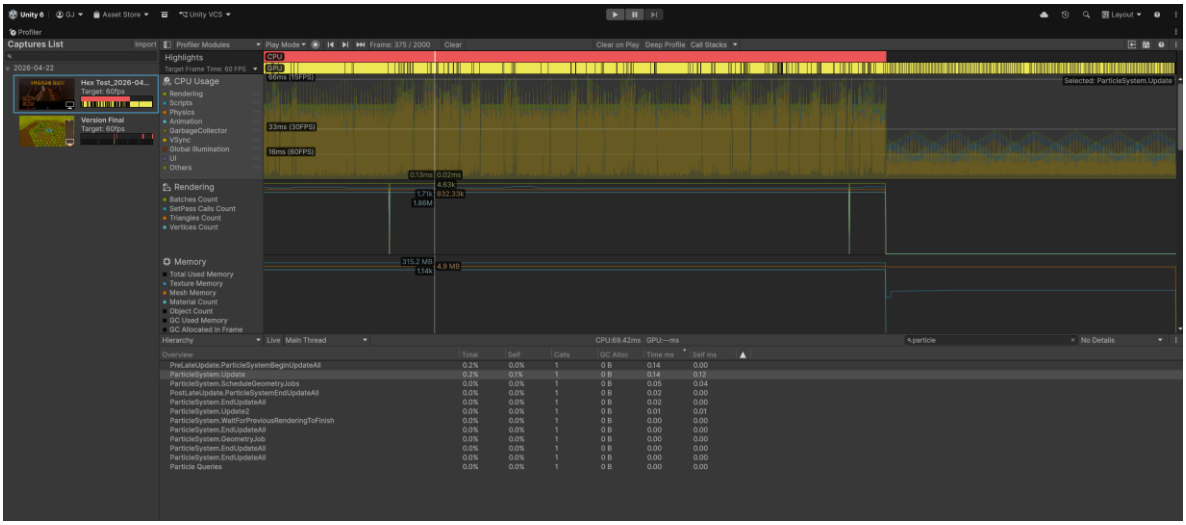
Análisis de procesamiento computacional post-optimización



Nota. Diagnóstico de la carga computacional en *Unity Profiler* demostrando tiempos de ejecución estables en los procesos lógicos y de partículas.

Figura 60

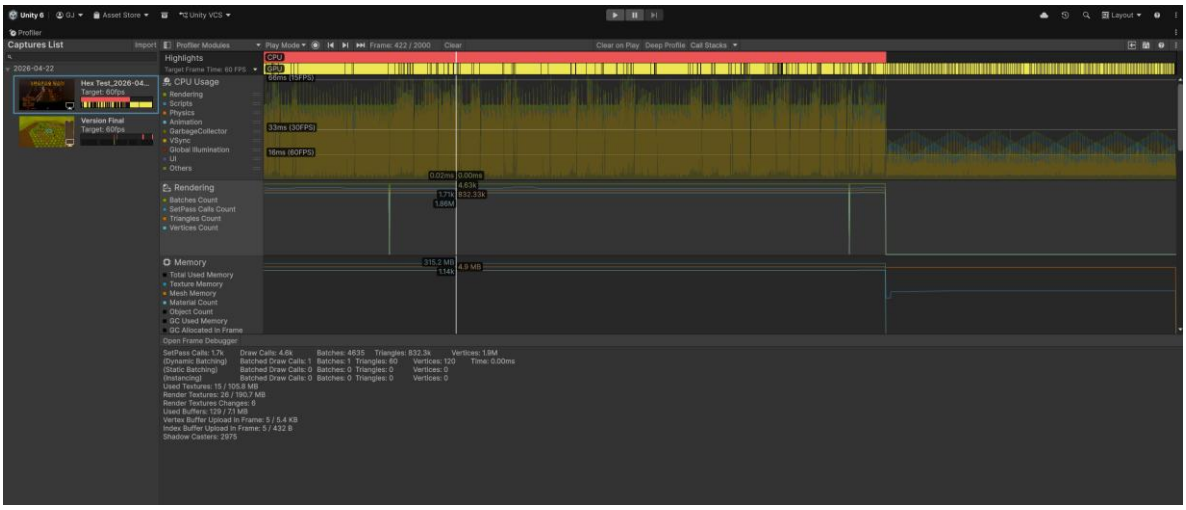
Análisis de procesamiento computacional pre-optimización



Nota. Captura inicial del *Unity Profiler* detallando las saturaciones e inestabilidad del procesador previas a la implementación de mitigación de físicas.

Figura 61

Análisis de procesamiento grafico pre-optimización



Nota. Monitoreo de memoria gráfica y renderizado que ilustra la alta demanda provocada por materiales independientes previo al uso del *Texture Atlas*.

4.12. Procedimiento para ampliar recursos artísticos

Para que el equipo de desarrollo pueda escalar el proyecto y desarrollar un nuevo nivel (bioma) o criatura manteniendo el rendimiento actual, se debe ejecutar el siguiente protocolo.

1. Modelado: Construir la geometría en Blender bajo topología *low-poly* estricta. Mantener una correcta malla para deformación. Colapsar vértices ocultos.
2. UV *Mapping* (entornos): Evitar texturas propias. Escalar los polígonos a un punto mínimo en el editor UV y posicionarlos sobre la paleta de colores en la textura atlas de 128x128 pixeles.
3. UV *Mapping* (personajes): Distribuir las islas UV optimizando el espacio. Exportar a Substance Painter, realizar el *bake* base para revisión de geometría, pintar el color base a mano y exportar como único mapa de 1024x1024 pixeles.
4. Integración de *shaders*: Asignar el modelo a un material que utilice el "*CellShader*". Ajustar las variables del material según las necesidades del nuevo entorno.
5. VFX: El material debe ser *unlit*. Las texturas base no deben superar los 64x64 pixeles. El máximo de partículas nunca debe superar un límite técnico de 15.
6. *Prefabs*: Importar el (.fbx) a Unity, ajustar la escala y rotación dentro de un *empty*, asignar los materiales optimizados, convertir en *prefab* y arrastrar a las carpetas correspondientes para si consumo por parte del generador procedural.

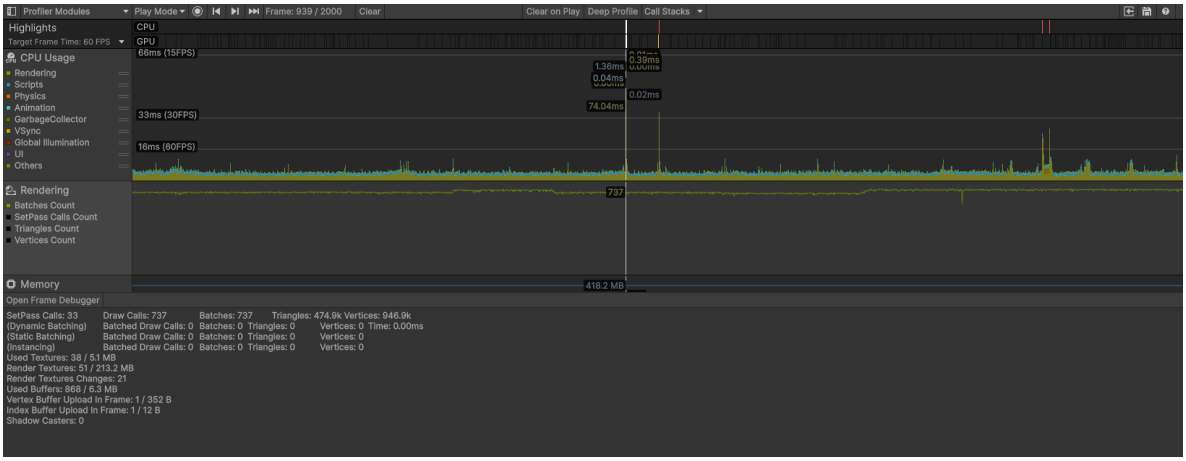
4.13. Validación cuantitativa y pruebas de estrés.

Las pruebas cuantitativas demostraron la eficiencia técnica del flujo de trabajo implementado. Al ejecutar la *vertical slice* en computadoras de bajo rendimiento, el software logró mantener una tasa constante de 60 fotogramas por segundo (FPS) sin fluctuaciones.

Este rendimiento se alcanzó gracias a la reducción de las llamadas de dibujado (*draw calls*). El proyecto pasó de un promedio de 4630 *batches* en sus fases iniciales a mantener un límite estable de 730 *batches*.

El consumo de la VRAM se fijó en 0.54 GB, una cantidad de memoria estática, ya que, no escala durante la generación procedural de los mapas. La carga de procesamiento del CPU sobre los sistemas se mantuvo en 0.0 ms, garantizando una fluidez incluso al superponer múltiples efectos en pantalla.

Figura 62
Resultados unity profiler



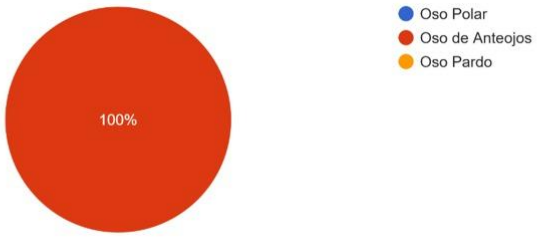
Nota. Informe consolidado del *Unity Profiler* corroborando el éxito de la optimización al sostener el rendimiento técnico sobre hardware limitado.

4.14. Validación cualitativa y experiencia de usuario

Para la evaluación cualitativa, se trabajó juntamente con el área de diseño interactivo empleando una muestra de 12 niños de entre 8 y 12 años, en sesiones de 15 minutos. Los datos se recolectaron mediante encuestas.

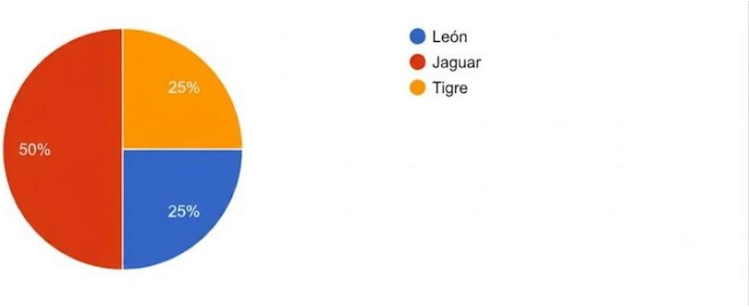
Los resultados reflejaron que el entendimiento visual del modelado *low-poly* en conjunto con el *CellShader* resultó claro para el público objetivo. La legibilidad de la pantalla permitió a los usuarios identificar la gran mayoría de las representaciones.

Figura 63
Pregunta de encuesta. ¿Qué oso de color oscuro y manchas claras en la cara aparecía en los niveles de la Sierra?



Nota. Distribución estadística de los resultados de reconocimiento visual del modelo del oso de anteojos por parte del público objetivo.

Figura 64
Pregunta de encuesta. Un gran felino muy ágil te acechaba en tu recorrido. ¿Qué animal era?



Nota. Gráfico de aciertos correspondientes a la evaluación heurística de legibilidad del modelo felino andino.

Los VFX que representaban a las deidades incaicas tuvieron una correcta comprensión, poderes como el escudo de luz o el rayo que teletransporta al personaje fueron reconocidos. Esto valido que el enfoque técnico logro optimizar los requerimientos del hardware sin sacrificar la comunicación visual.

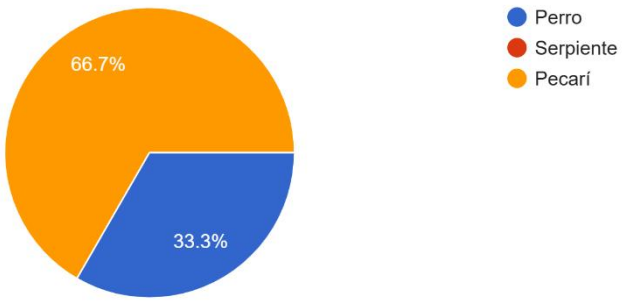
4.15. Discusión de los resultados visuales

Durante las pruebas con el público objetivo, se detectaron áreas de mejoras especificas relacionadas con la asimilación de la fauna. La evaluación evidenció que la silueta de ciertos modelos tridimensionales presentaron dificultades de identificación.

Al introducir el bioma amazónico (Nivel 1), el 33.3% de los usuarios identificó erróneamente al enemigo como un perro en lugar de un pecarí. Tras el análisis del comportamiento de los usuarios, este porcentaje se atribuyó a un desconocimiento sobre la existencia de la especie.

Figura 65

Pregunta de encuesta. En la selva de la Amazonía, debías tener cuidado con un animal salvaje parecido a un cerdito pequeño. ¿Cuál era?



Nota. Cuantificación de los errores de asimilación e identificación de los usuarios respecto al modelo del pecarí en el bioma amazónico.

La representación del jaguar presentó una tasa de reconocimiento del 50% mientras que el resto de la muestra lo confundió con especies ajenas a la región. El flujo de trabajo permitió el ajuste visual en la textura del jaguar. La corrección se evaluó como exitosa tras una rápida identificación por parte del público objetivo.

4.16. Conclusiones

La estructuración del pipeline implementado confirma la viabilidad de desarrollar software visual para hardware de capacidades limitadas. Se alcanzó exitosamente el equilibrio entre un rendimiento computacional fluido y una dirección de arte que retiene la atención del público objetivo.

El éxito central de esta optimización recae en la creación e implementación de *shaders* personalizados. Remplazar procesos de simulación nativos demandantes por matemáticas de sombreado permitió resultados visuales llamativos sin estrés para la GPU.

La metodología presentada demuestra que evitar flujos de trabajo tradicionales y orientar la producción visual hacia un control desde la fase conceptual, consolida un estándar funcional.

Referencias

[1] M. M. Juvekar, C. J. Rajwade, S. M. Charkari, P. R. Raut, and P. Pawar, "3D GAME DEVELOPEMENT," 2024. [Online]. Available: www.ijcrt.org

[2] N. Boonsawang, K. Motonobu, and P. Chanpum, "Optimizing Art and Technology in 3D Character Creation: Strategies for High-Fidelity Modeling and Resource Efficiency in Real-Time Environments," 2024. [Online]. Available: https://rsucon.rsu.ac.th/proceedings

[3] L. Pirhonen, "Creating an Optimized 3D-mesh for a Stylized Low-poly Video Game Character," Dec. 2024.

[4] F. N. Sibai, A. Potvin, and S. Ngo, "Optimizing Particle Systems through CUDA-Assisted Multithreading," *WSEAS TRANSACTIONS ON SYSTEMS AND CONTROL*, vol. 15, pp. 691–698, Dec. 2020, doi: 10.37394/23203.2020.15.69.

[5] S. Fenech, "IMPACT OF ART STYLE IN 2.5D ISOMETRIC GAMES," 2024.

[6] J. Paul. Gee, *What video games have to teach us about learning and literacy*. Palgrave Macmillan, 2003.

[7] Y. Lin and K. Tanaka, "Influences of visual effects on sense of agency in video games," *R. Soc. Open Sci.*, vol. 13, no. 3, Mar. 2026, doi: 10.1098/rsos.251930.

[8] B. Bontchev, "Serious Games for and as Cultural Heritage," *Digital Presentation and Preservation of Cultural and Scientific Heritage*, vol. 5, pp. 43–58, Sep. 2015, doi: 10.55630/dipp.2015.5.3.

[9] Matt. Smith and Shaun. Ferns, *Unity 2021 Cookbook Fourth Edition*. PACKT PUBLISHING LIMITED, 2021.

[10] Ernest. Adams, *Fundamentals of game design*. New Riders, 2014.

[11] R. H. Sampieri, C. F. Collado, and M. del P. B. Lucio, "Metodología de la investigación," 2014.

[12] J. Schell, "The Art of Game Design: A Book of Lenses," 2008.

[13] R. Imam and Q. M. Areeb, "Game Effect Based on Particle System in Unity 3D," *Academia Letters*, Jul. 2021, doi: 10.20935/AL2447.

[14] Mojang Studios, "Minecraft," https://www.minecraft.net. Accessed: Aug. 05, 2026. [Online]. Available: https://www.minecraft.net

[15] Midjiwan AB, "The Battle of Polytopia," https://polytopia.io/. Accessed: Aug. 05, 2026. [Online]. Available: https://polytopia.io/

[16] M. M. Keleşoğlu and D. Güleç Özer, "A study on digital low poly modeling methods as an abstraction tool in design processes," *Civil Engineering and Architecture*, vol. 9, no. 7, pp. 2570–2586, Dec. 2021, doi: 10.13189/cea.2021.091513.

[17] V. J. Cicchirillo, "The impact of video game character viewpoints and task on perceptions of cognitive and similarity identification," *Cyberpsychology: Journal of Psychosocial Research on Cyberspace*, vol. 14, no. 4, pp. 1–14, Nov. 2020, doi: 10.5817/CP2020-4-2.

[18] E. Hastings, R. Guha, and K. O. Stanley, "NEAT Particles: Design, Representation, and Animation of Particle System Effects," in *2007 IEEE Symposium on Computational Intelligence and Games*, IEEE, 2007, pp. 154–160. doi: 10.1109/CIG.2007.368092.