Universidad Autónoma de Ciudad Juárez
Instituto de Arquitectura, Diseño y Arte
Departamento de Diseño
Licenciatura en Diseño Digital de Medios Interactivos

Título

Análisis del modelado tridimensional enfocado a la optimización de polígonos y aplicación de técnicas de texturización en la industria de los videojuegos

Protocolo de investigación
presentado por:

Victor Daniel Ferreyra Márquez

Gladys Michelle Carrillo Guerrero

Para obtener el título de Licenciado en Diseño Digital de Medios Interactivos

Director(a): Ramon Iván Barraza Castillo.

Ciudad Juárez, Chihuahua, noviembre 2022.

Universidad Autónoma de Ciudad Juárez
Instituto de Arquitectura, Diseño y Arte

Departamento de Diseño

En nuestro de carácter de director y lectores, hacemos constar que el proyecto de investigación: Análisis del modelado tridimensional enfocado a la optimización de polígonos y aplicación de técnicas de texturización en la industria de los videojuegos presentado por VICTOR DANIEL FERREYRA MÁRQUEZ y GLADYS MICHELLE CARRILLO GUERRERO, con Matrícula 162993 y 172145 cuenta con las características de aportación novedosa y solidez metodológica exigida por la normativa universitaria.

Dr. Ramón Iván Barraza Castillo
Director del Proyecto de Investigación

Mtra. Lucia Alejandra de la Torre Rodríguez
Sínodo

Mtra. Anahí Solís Chávez
Coordinadora de la Licenciatura
Diseño Digital de Medios Interactivos

Mtro. Rogelio Baquier Orozco
Sínodo

Noviembre de 2022

DEDICATORIA

Victor Daniel Ferreyra Márquez

Dedicado a mi familia y amigos que estuvieron a mi lado y apoyaron durante todo este tiempo, a los que creyeron que podría hacer realidad este logro tan importante para mí ya que sin su amistad y confianza no sería quien soy hoy en día.

Una dedicación especial a mi hermano y a mi madre que me han estado apoyando incondicionalmente y que velaron por mí en mis tiempos de crisis de salud.

Y a aquella que vive debajo de mí de esta página porque sin su apoyo este proyecto probablemente no hubiera sido posible.

Gladys Michelle Carrillo Guerrero

Dedicado a mi familia, mis padres, y especialmente a mis hermanos, que siempre me apoyaron de maneras distintas, uno con las indagaciones, curiosidades y risas, y el otro con su moralidad, su mentecata personalidad, inteligencia, sus constantes debates y, sobre todo, su capacidad de comprenderme y apoyarme en mis loqueras y terqueza.

A mis amigos, que, a pesar de ser mis amigos, no saben la importancia que tienen para mí.

A mis pequeñas niñas, Jaina y Orim, mis amores.

Y a aquel que vive arriba de mi en esta página, que, a sabiendas de mi personalidad, trata a su manera de apoyarme, y fue de amplio apoyo en este proyecto, y, esperemos, también en lo siguiente de nuestras vidas. -.. . ... .--. .- -.-. .... .- -.. --- …

Victor Daniel Ferreyra Márquez

Me gustaría agradecer a los docentes que reafirmaron mi interés sobre la carrera los cuales son la Mtra. Lucia De la Torre por sus enseñanzas en el modelado 3D y al Dr. Ramón Barraza por las enseñanzas de programación, la dirección en la realización de este proyecto de investigación y su alegría y gusto por esta carrera.

Quiero agradecer también a mi compañera por su apoyo y confianza durante el desarrollo de este proyecto y también durante la segunda mitad de mi vida universitaria ya que sin su apoyo esto hubiera sido más complejo de lo esperado. No solamente eso sino también por su amistad y cariño que resultó ser de gran apoyo moral durante este tiempo y a la flexibilidad de ayudarme en tiempos de crisis en las cuales el proyecto pudiera haberse sido congelado y no concretado.

También quiero agradecer a mis mascotas Orim y Jaina que comparto con mi compañera ya que fueron de gran apoyo moral y sirvieron para la creación de nuestra compañía Jaihdim Studios así como el logo que juntos creamos.

De igual manera quiero agradecer a mi hermano y a mi madre por su apoyo incondicional en las buenas y en las malas, que si no fuera por el esfuerzo de ellos especialmente en tiempos de salud critica, tal vez yo no estuviera aquí el día de hoy para completar este logro que es un sueño para mí.

AGRADECIMIENTOS

Gladys Michelle Carrillo Guerrero

Quiero agradecer a todos los que me apoyaron mientras cursaba ambas licenciaturas, a mis amigos, mi familia, y aquellos que creyeron en mi aun cuando a veces yo no lo hacía.

A mi compañero, cuyo proyecto e idea fue inicialmente suyo, el cual me enseño muchos de los aspectos dentro de este documento, quien me apoyo y soporto mis incesantes indagatorias y curiosidades, porque a veces, cuestiono todo.

Agradezco a mis criaturas hermosas, mis mascotas, Jaina y Orim, las cuales fueron la inspiración del logo de nuestra compañía, Jaihdim Studios, la muestra de unión de más que solo una relación, algo que espero sea permanente.

Al doctor y docente, Ramon Ivan Barraza Castillo, cuyo trabajo y colaboración fue un gran apoyo para nosotros, su amor y pasión dentro de la carrera es palpable y un ejemplo instrumental de docentes con verdadera dedicación.

Por último, agradezco a todos aquellos profesores que abrieron sus puertas y escucharon una opinión o retroalimentación, aun si errónea, personal o de otros estudiantes, y lograron mejorar como docentes dentro de la carrera por su capacidad de escuchar, algo que a veces, a todos, sin importar el nivel, conocimiento, titulo, etc. Podemos olvidar con facilidad. De verdad.

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1: EL PROBLEMA

INTRODUCCIÓN

En este capítulo se describirá las problemáticas a abordar dentro de la investigación y la razón del porque es necesario investigarlas. Se contextualizará, con el comienzo del desarrollo tridimensional, sus vertientes, eventualmente llegando a el problema, y las posibles soluciones, justificando él porque es vital tener una solución, delimitando el área en donde se aplicará, las preguntas y objetivos que se esperan superar.

1.1 Antecedentes del Problema

El desarrollo del modelaje tridimensional (3D) tiene una muy extensa historia, con anterioridad, la creación de dispositivos electrónicos y computadores capaces de generar cálculos matemáticos extensos y expresarlos de manera virtual vía una pantalla era una tecnología solo permitida por razones militares, como la creación del computador SAGE (Semi-Automatic Ground Environment) a los inicios de 1950, un sistema de defensa antinuclear creado por la empresa IBM por órdenes del gobierno de Estados Unidos (IBM, 2012); eventualmente esta tecnología emergente fue evolucionando, perdiendo la exclusividad militar y llegando a empresas de distintas indoles, mejorando de manera exuberante cada año, al punto de la modernidad, donde la mayoría de los ciudadanos comunes tienen algún tipo de dispositivo electrónico complejo, ya sean dispositivos móviles o computadoras personales (PC), según la INEGI, en México “se cuenta con 88.2 millones de usuarios de teléfono celular (75.5% de la población de seis años o más), […] y se estimaron 44.4 millones de usuarios de computadora, lo que representa un 38.0% del total de la población” (2020).

Gracias a la revolución tecnológica, la facilidad de creación y elaboración de elementos 3D es relativamente simple y de un costo accesible el día de hoy. Actualmente existen herramientas de desarrollo 3D gratuitas, que compiten contra otros programas de industria de altos costos, esto por consecuente ha generado una emergente vertiente de desarrolladores independientes, con distintos dispositivos, niveles de aprendizaje, objetivos, dependencias y conocimientos en diferentes ramas de diseño y programación 3D causando una significativa fluctuación de calidad de resultados e ideas en la implementación, sin embargo, esto no significa que sea un aspecto negativo, existen muchos beneficios detrás de la facilidad de desarrollo y comunicación entre las distintas diciplinas, según ALLPLAN (2018):

> "Las ventajas del desarrollo un proyecto digital es obvio. El intercambio continuo de información y comunicación entre miembros […] auxilia a que todas las diciplinas puedan juntarse y trabajar consistentemente. Esto funciona sin importar los países o lenguajes. […] Intercambiar información, en cualquier momento, en cualquier lugar es un estándar el día de hoy".

Sin embargo, la era de la comunicación y tecnología contemporánea puede causar problemas de distintos tipos, "Pero en toda interacción existe el riesgo de una comunicación malintencionada, mal informada o equivocada con el potencial de generar efectos no deseables; […] la facilidad para la publicación de contenidos anónimamente y sin regulación aumentan dicho riesgo" (Gómez, 2013). La desinformación o falta de conocimiento, la facilidad de acceso a programas de diseño y dispositivos de desarrollo, generan una fluctuación en relación con la calidad y cantidad de objetos producidos por los distintos diseñadores en el mundo.

Al existir una alta cantidad de información, accesibilidad de herramientas y alta independencia en la modernidad, los artistas 3D radican de distintas vertientes, estos usuarios indagan en variadas fuentes de información, ya sea entrenamiento propio, búsqueda de desarrollo independiente, educación directa de alguna fuente (Académica, informativa, educativa, cursos, etc.), las posibilidades siendo infinitas: “Con el paso del tiempo, el costo de las aplicaciones 3D se vuelve menor, permitiendo que novatos y más diseñadores 3D corran a la oportunidad de estar a la luz junto con otros profesionales de la industria.” (Color Experts, 2019).

La versatilidad del desarrollo 3D es un aspecto altamente llamativo y lucrativo, sin embargo, la curva de aprendizaje y constante esfuerzo es relativamente alta, como menciona Chilana et al., “Aun si existen amplias herramientas, manuales, materiales instruccionales, cursos y procesos de entrenamiento que son fáciles de entender para los principiantes, el aprendizaje 3D puede ser muy desalentador y complejo” (2018). En el aspecto 3D, el aprendizaje tiende a ser constante y altamente complicado, por la misma versatilidad del desarrollo; la libertad de creación nos abre las puertas a la creación de posibles objetos no propiamente optimizados para la industria, sin importar el ámbito (Independiente, o de un grupo relativamente pequeño de personas, o profesional), por ende, existe una alta cantidad de información relacionada a la búsqueda de inicialización del modelado 3D, para distintos tipos de objetivos, sin embargo, cave denotar que por la alta variedad de información, sistemas, detalles, formas, estilos, opciones y actualizaciones, es difícil mantener un orden ideal o fácil de comprender que exponga la mayoría de las pautas importantes en relación a lo esperado por parte del usuario; por ser una tecnología en constante evolución, es importante considerar que no puede existir un absoluto: “La reducción acelerada de la vida media del conocimiento, es

decir, el tiempo medio desde que aparece un conocimiento hasta que se vuelve obsoleto" (Siemens, 2005).

1.2 Planteamiento del Problema

Los efectos de la falta de conocimiento y aplicación de la optimización de modelados 3D enfocados para la animación de videojuegos que ocurren dentro de las empresas, grandes o indies, de hoy en día puede ser que ya no muestran interés en optimizar los modelados 3D para videojuegos o simplemente carecen del conocimiento de esto ya que "como ahora tenemos la tecnología" se dan el lujo de cometer dichos errores (K Gon, 2020). Estos problemas se reflejan al ejecutar el juego, generando un rendimiento bajo y por lo tanto reduce la calidad del producto final. Muchas veces no les interesa invertir en dichas optimizaciones por el simple hecho de querer sacar el producto final sin pasar por una revisión de calidad lo cual puede llevar a una representación equivocada de iluminación y de render (Whitmer, 2019).

Para lograr este tipo de optimizaciones, según Blázquez "es deseable reducir el número de nodos en escena ya que esto afecta al rendimiento, aunque depende en última instancia de cómo funcione nuestro motor de juego. Los programas de edición 3D suelen representar un nodo mediante un eje local" (2006). Esto quiere decir que entre menos polígonos estén presentes dentro del mundo o de la escena a renderizar, mejor será nuestro rendimiento al momento de correr el juego, esto es debido a que es menos carga tanto para el Central Processing Unit (CPU) como el Graphics Processing Unit (GPU).

En esta investigación se abordará la importancia de comprender las técnicas de optimización y texturización para poder llevar a cabo un mejor rendimiento, todo en la

perspectiva del diseño y desarrollo de modelado tridimensional en la industria de los videojuegos.

1.3 Delimitación del Problema

Es importante denotar que aun si el enfoque principal de esta investigación es la optimización de modelos 3D en la industria de los videojuegos, la mayoría del contenido dentro de esta investigación puede ser aplicada en cualquier otra temática relacionada con el desarrollo tridimensional, sin embargo, dependiendo del enfoque y la aplicación de los modelos, estos objetos pueden beneficiarse de una mayor cantidad de detalle.

En la industria cinematográfica digital, las películas no son renderizadas en tiempo real, y solo son controlados por distintos animadores, esta ventaja permite que los animadores enfoquen lo que realmente es importante e ignorar aquello que no sea relevante, por lo tanto, la calidad de modelos y polígonos pueden ser más indulgentes y libres.

En contraste, los videojuegos deben enfocarse en todos los detalles, objetos, mecánicas posibles que el jugador puede (o no) ver o interactuar en tiempo real. Un videojuego está constantemente usando el rendimiento grafico (GPU) y el procesamiento (o interpretación) lógico-aritmético (CPU) en todo momento mientras el juego esta activo, por esta razón, es importante comprender más a fondo las posibles limitaciones de los dispositivos en los cuales se jugarán:

“Los juegos son renderizados en tiempo real frente al jugador, para que el juego corra de manera apropiada, constante y fluida, los modelos 3D deben de ser creados a un nivel que no suponga una carga mayor para el motor de juego. Mientras juegas, puede

haber miles y miles de objetos renderizados al mismo tiempo en la pantalla, lo cual toma un alto poder de procesamiento" (Pluralsight, 2014).

Considerando lo anterior descrito, el enfoque de esta investigación está enfocada en la industria de los videojuegos, específicamente en computadores personales, donde es importante cuidar la cantidad de polígonos que los modelos 3D emplean, para evitar el uso excesivo de recursos, según el dispositivo en el que se jugara, por ende, se analizarán las posibles técnicas de optimización de polígonos, texturización y modelado tridimensional, eventualmente aplicándolas en una prueba controlada, donde se observarán los beneficios, ventajas, y efectos de impacto en los aspectos lógico-aritmético-grafico.

1.4 Preguntas de Investigación

Pregunta principal:

¿Cómo a través del análisis y la creación de una aplicación se pueden dar a conocer los beneficios de las técnicas de optimización de polígonos y texturización y modelado en la industria de los videojuegos?

Preguntas Secundarias:

- ¿De qué maneras se puede lograr la optimización de un modelado 3D orientado para videojuegos?
- ¿Cuáles son las ventajas de optimizar uno o varios modelos 3D?
- ¿Qué pasos se deben de seguir para mejorar el flujo de trabajo de un modelado 3D?

1.5 Objetivos de Investigación

1.5.1 Objetivo General

Desarrollar una aplicación por medio de Unity en donde se demuestre la importancia y beneficios de las distintas técnicas que existen para la optimización de polígonos, texturización y modelado tridimensional en la industria de los videojuegos.

1.5.2 Objetivos Específicos

- Conocer las distintas técnicas de optimización de polígonos, texturización y modelado tridimensional en la industria de los videojuegos.
- Analizar las ventajas de optimizar los modelos 3D.
- Investigar como son implementadas las técnicas de rendimiento, texturización y optimización dentro el motor de juego Unity en la industria de los videojuegos.

1.6 Justificación del Problema

Es un hecho que para los diseñadores de videojuegos es indispensable conocer la importancia de optimizar y la necesidad de mantener un control de polígonos al momento de diseñar un videojuego que contenga un mejor uso de texturas para dar mejor detalle sin la necesidad de tener que utilizar polígonos innecesarios para así poder lograr un mejor rendimiento del videojuego y por lo tanto mejorar la experiencia del usuario. Existen muchas formas de optimizar y es importante tomarlas en cuenta para hacer un uso correcto de estas técnicas para objetos no interactivos, construcciones y objetos animados.

Estas herramientas de diseño están enfocadas para los desarrolladores y diseñadores de videojuegos que buscan un mejor rendimiento para sus videojuegos para

abrir la gama al público que contenga diferentes equipos de cómputo u otras plataformas como lo serían las consolas de videojuegos.

Se busca lograr que este trabajo aporte al área de diseño de videojuegos para un mejor flujo de trabajo en el departamento gráfico para poder integrar nuevas ideas creativas que pueden ser el resultado de una optimización en las cuales genere una mejor estética y rendimiento siendo el producto de un uso óptimo de la optimización de polígonos en videojuegos. Esto se logrará a través de un análisis extenso, utilizando como referencia diferentes ejemplos de juegos que hacen o evitan el uso de estas técnicas para mantener un resultado consistente, eventualmente generando una aplicación que demuestre los distintos niveles de optimización y texturización.

CAPÍTULO 2: MARCO CONCEPTUAL

INTRODUCCIÓN

Dentro de este capítulo se hablará con detalle de todos los conceptos comunes que se utilizaran en esta investigación, para contextualizar e informar al lector de los términos comunes que se utilizaran dentro de la investigación.

2.1 Modelado 3D

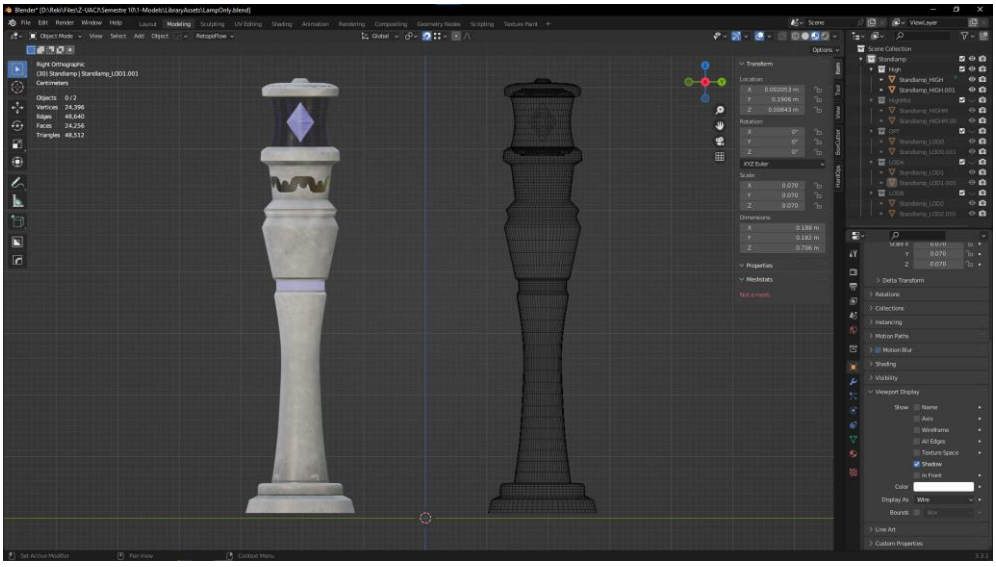


Figura 1. Muestra del modelado tridimensional en Blender

El modelado 3D es el proceso por el cual se realiza el concepto de algo que fue diseñado en 2D, el cual se toma como referencia para hacer el diseño en 3D por medio de técnicas dentro de un software especializado en este tema. Esto se logra a partir de la construcción de una malla la cual puede ser manipulada para deformaciones, fracturaciones, animaciones, imprenta etc. A palabras de J. Petty, “El modelado 3D es la técnica utilizada en los gráficos de computadora para producir una representación digital 3D de cualquier objeto o superficie” (2020).

Estos modelos hacen uso de técnicas que se utilizan en los gráficos de computadora para la representación 3D de un objeto, estos son creados por un software especializado en ello, dichos modelos están conformados por una malla de polígonos de los cuales tienen vértices y aristas, estas conforman las caras que dan forma al modelo 3D y al momento de ser renderizados empiezan a consumir recursos del equipo de cómputo. Estos objetos pueden ser manipulados para ser deformados o utilizados para la animación o para otros medios, entre ellos videojuegos.

2.1.1 Polígonos en un modelo 3D

"Los polígonos son derivados de geometría basados en vértices, aristas y caras que se pueden utilizar para crear un modelo tridimensional" (Autodesk 2016). Es decir, son figuras compuestas de tres o más puntos de anclaje, los cuales se les determina como vértices, estos están interconectados por aristas y forman una cara. Por lo regular para un modelado óptimo y fácil de entender se utilizan los polígonos cuyas caras están compuestas de 4 vértices y 4 aristas, estas se les definen como "quads", estos al momento de ser exportados para ser implementados en un motor de juego o como archivo final que va a cumplir un uso son transformadas a caras de 3 vértices y 3 aristas cuyos nombres son definidos como "tris", otros polígonos que estén compuestos de 5 o más vértices y aristas son conocidos como "n-gon" los cuales no son lo óptimo para un modelado 3D puesto que las normales de las caras no son consistentes debido a las irregularidades que puede presentar en el posicionamiento de los vértices.

2.1.2 Técnicas de modelado

Existen distintos tipos de técnicas de modelado, sin embargo, se mencionarán las más comunes y conocidas por los desarrolladores de modelado tridimensional.

2.1.2.1 Escultura

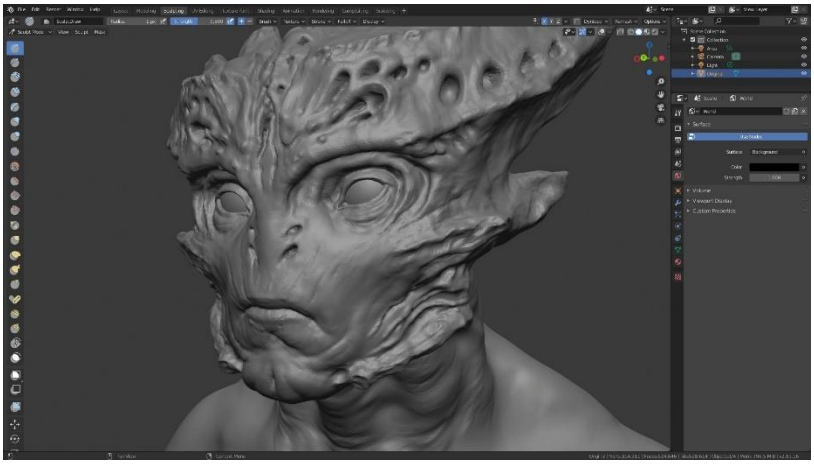


Figura 2. Modelado de escultura en Blender

En palabras de A. Naghdi y P.Adib, “es un proceso nuevo que hace alusión al modelado con plastilina en el mundo real.” (2021). En si la base de este método es el de poder modelar un objeto 3D sin la preocupación de la cantidad de polígonos que usamos puesto que nuestro objetivo no es llegar a un objeto óptimo para animar sino para llegar a la forma base de nuestro modelo de manera más rápida y eficiente, la técnica de esta técnica de modelar es muy similar a la de hacer esculturas con arcilla puesto que puedes ir agregando y quitando polígonos, usar distintas brochas o utensilios para dar ciertos efectos a discreción, tal y como se haría en la vida real.

2.1.2.2 Retopología

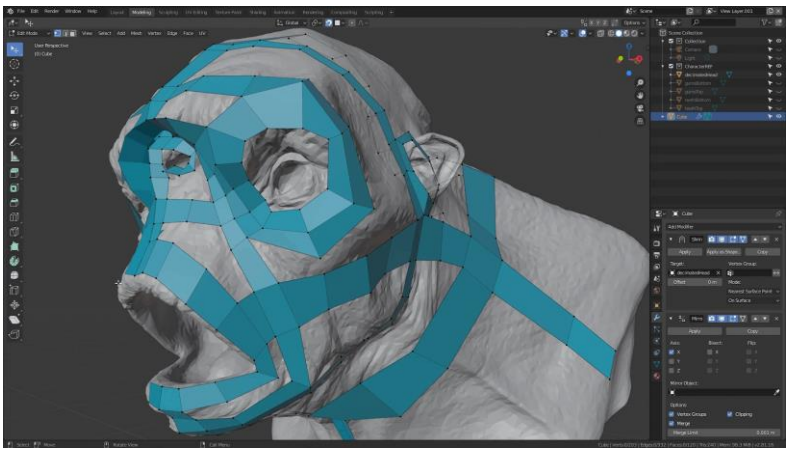


Figura 3. Escultura en proceso de retopología en Blender

De acuerdo con J. Petty, “la retopología es el proceso de convertir modelos de alta resolución a un producto más pequeño que se puede utilizar para la animación, Puede ser un proceso difícil, pero la idea básica es crear otra malla que simplifique la original.” (2019). Con esto podemos concluir que la retopología es una herramienta muy útil para optimizar modelos que empezaron como esculturas o para hacer correcciones de modelos que podemos optimizar para animación y videojuegos. Esta técnica es comúnmente utilizada después de hacer la escultura siempre y cuando el objetivo del modelo sea para videojuegos o animación.

2.1.2.3 Box Modelling

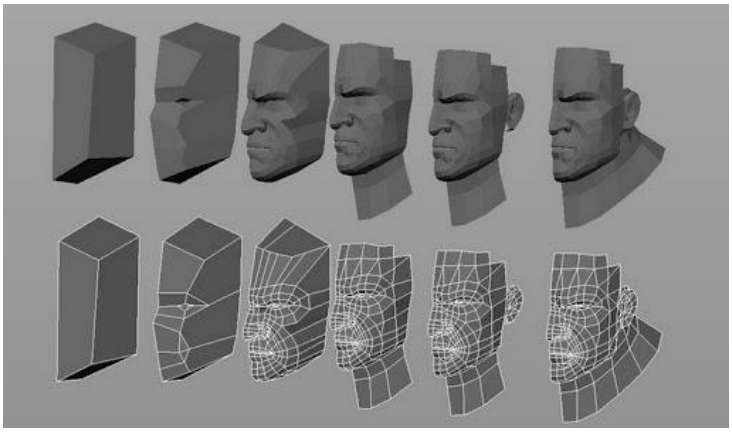


Figura 4. Progreso de un modelo en Box Modeling

En palabras de A. Naghdi y P.Adib, “es un método rápido para crear figuras básicas. Todo empieza con un cubo en el Box Modelling” (2021). Tal y como el nombre de la técnica lo indica, es una forma de modelar a partir de figuras básicas como lo sería un cubo y a partir de ahí se van agregando detalles a como sea necesario, usualmente esta técnica de modelado ya es muy anticuada, pero es muy utilizada aun hoy en día por distintos niveles de artistas, desde principiantes hasta profesionales de la industria.

2.2 Optimización

La optimización es el proceso por el cual se revisa los proyectos ya hechos y estos mejoran a partir de distintas técnicas, ya séase de código, gráficos, motor grafico; en el caso de la optimización de modelado 3D. De acuerdo con D. Blázquez, “En el mundo de los videojuegos, la optimización es el arte de aprovechar al máximo los recursos de un sistema, o visto de otra forma, de mejorar el rendimiento de tu juego” (2006).

2.2.1 Función de la Optimización en el Modelado 3D

Para D. Blázquez, “Es deseable reducir el número de nodos en escena ya que esto afecta al rendimiento, aunque depende de la última instancia de como funcione nuestro engine. Los programas de edición 3D suelen representar un nodo mediante un eje local” (2006), el cual se refiere a polígonos con los que están hechos los modelos 3D.

2.2.2 Ventajas de la Optimización

Menciona J. Petty, que “las mallas complejas son difíciles de animar. Es necesario limitar la cantidad de polígonos de cualquier modelo 3D cuando es usado para la animación” (2019). Esto hace alusión de que las mallas complejas son difíciles de manipular y por ende de utilizar ya que consumen muchos recursos y ralentizan con procesos innecesarios afectando al rendimiento del hardware, si hacemos correcciones de topología y uso adecuado de polígonos podemos conseguir un mejor rendimiento y por lo tanto una mejor experiencia de usuario. Como indica la documentación de Unity (2017) “Hay dos reglas básicas para la optimización de la geometría de un modelo: No uses más triángulos de los necesarios. Trata de mantener el número de esquinas de corte de UV Mapping y esquinas fuertes tan bajas como sean posibles.” Esto nos indica que en realidad si hay un propósito por el cual es necesario hacer uso de estas prácticas, puesto que hay un beneficio considerable detrás de estas.

2.2.3 LOD (Level of Detail)

De acuerdoel modeloecGamer, “esto se refiere a la dinámica basada en la distancia y los ajustes de conteo de polígonos” (2019), es decir, el modelo se ajusta a las necesidades o situaciones apropiadas para lograr representar la misma imagen con una cantidad menor de polígonos debido a que la cámara estará situada lejos de donde se encuentra los modelos, tomando esto en cuenta que los polígonos serán renderizados en una cantidad menor de pixeles dando la ilusión de que son idénticos o muy parecidos sin importar la distancia entre la cámara y el modelo.

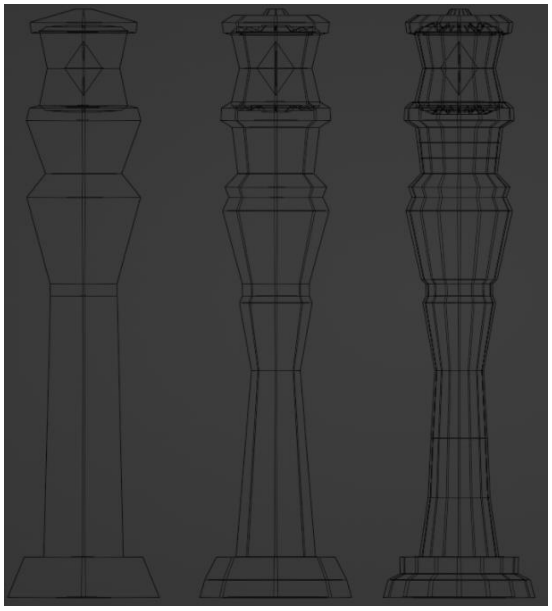


Figura 5. Distintos niveles de detalle en relación de polígonos

2.2.4 Teorías de la cadencia de optimización en el modelado 3D

Hoy en día se tiene la idea general de que el hardware es lo suficientemente potente como para soportar cuantos polígonos queramos y esto no es del todo cierto puesto que se da la oportunidad de realizar malas prácticas de modelado pensado para videojuegos, como se puede apreciar en la siguiente figura el cual es un objeto común del videojuego indie llamado Yandere Simulator del cual hace uso de 7744 polígonos (K Gon, 2020).

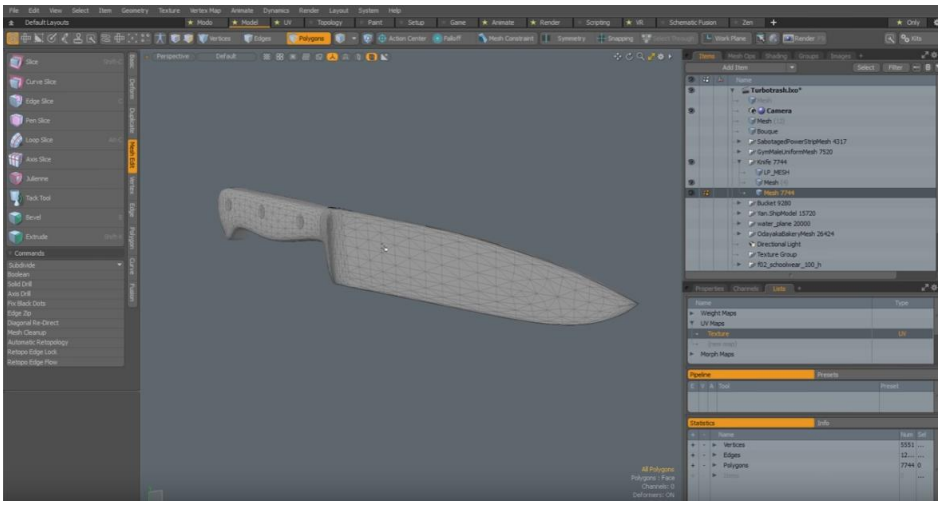


Figura 6. Ejemplo de uso alto de polígonos

Dicho este ejemplo se puede notar que al carecer conocimiento o el tiempo para desarrollar modelos 3D apropiados se recurre al uso de modelos 3D hechos con anterioridad por otros artistas, que pueden o no estar hechos en mente para videojuegos o para un ambiente distinto que sería el cinematográfico, la impresión 3D o simplemente para arte hiperrealista el cual su única función sería la de ser renderizada por sí misma o en un ambiente comercial de publicidad al cual nomas se renderiza una vez y se utiliza en su dicho propósito principal. Es esto por lo que es necesario el uso de distintas técnicas de optimización enfocadas a videojuegos.

2.3 Render

Según B. Whitmer, “El render 3D es el proceso de transformar la información de un modelo 3D una imagen 2D” (2019). Esto se puede interpretar a palabras más sencillas como el equivalente de una fotografía, o una película en la vida real, osease, se toma una fotografía de la escena realizada en el software por medio de un motor de renderizado, se renderiza este proceso para obtener el render con cálculo de colores, luces, sombras, etc.

2.4 Texturización en Modelos 3D

Las texturas son las imágenes que se realizan a partir del mapeado de un modelo 3D, estas son las que les da color, reflejo, iluminación, entre otros efectos para poder llegar al producto final al cual se traslada el proyecto trabajado. Según M. Geig, “Las texturas son las imágenes planas que son aplicadas a objetos 3D. Estas son las responsables de darles la oportunidad de ser coloridos e interesantes en lugar de algo blanco y aburrido” (2013). Tomando esto en consideración en los siguientes puntos se hablarán de los distintos tipos de texturas y su función principal.

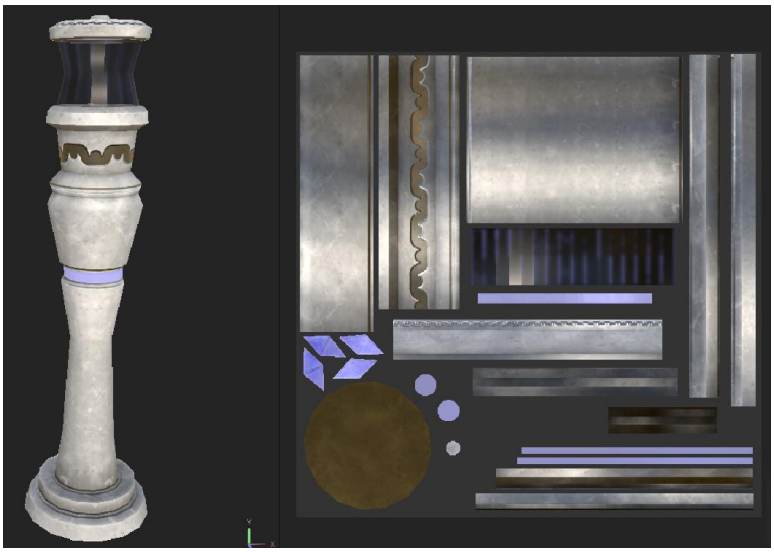


Figura 7. Modelo texturizado con su mapa de textura combinada

2.4.1 Albedo Maps

Según Royal Skies LLC, “Un mapa albedo es un mapa que no tiene nada más que colores, no debe de haber nada de información de sombreado ni de iluminación” (2019). Es decir, la imagen que se utiliza para dar color no debe de haber nada de sombras ni luces, debe ser lo más neutro posible para ahorrar espacio y dar detalle con otras texturas.

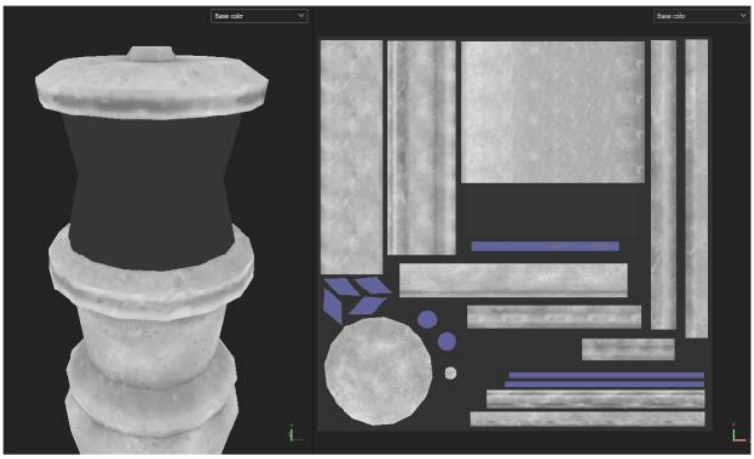


Figura 8. Albedo map en modelo de lampara

2.4.2 Roughness Maps

Según Royal Skies LLC, “este mapa es el encargado de que tan claro es el reflejo del entorno” (2019). También llamado gloss maps, es el responsable de que tanto es posible reflejar el entorno que lo rodea, por ejemplo, un espejo es perfectamente gloss y specular, mientras que una moneda es altamente specular y poco gloss puesto que en la moneda no se podrá ver el reflejo de una persona, pero si se podrá ver la luz reflejada en ella.

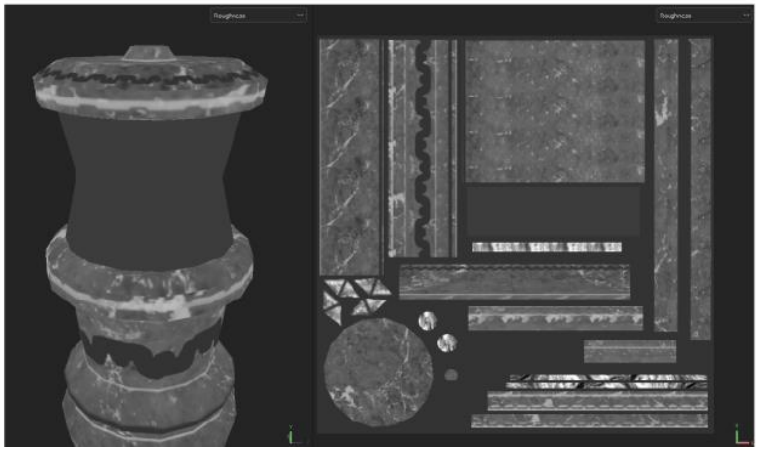


Figura 9. Roughness map en modelo de lampara

2.4.3 Metallic Maps

Según Chunk, “Es un mapa de escala a grises donde 0,0,0 (negro) no es metálico y 1,1,1 (blanco) es metálico” (2019). Gracias al uso de un mapa de texturas es posible aplicar un efecto metálico a superficies específicas de un objeto todo siendo controlado por escala a grises donde el blanco es metal y el negro no lo es.

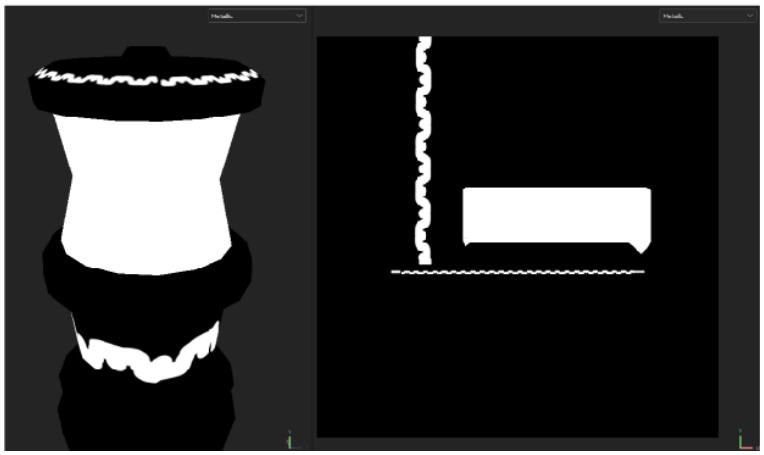


Figura 10. Metallic map en modelo de lampara

2.4.4 Normal Maps

En palabras de Pluralsight Creative, “un normal map crea la ilusión de detalle en un modelo low poly” (2014) y además “utilizan valores RGB para crear ángulos con respecto a los ejes XYZ”. Es decir, los normal maps crean la ilusión de un modelo muy detallado cuando no lo es, esto es debido a que manipula el cómo interactúan las normales del modelo con la luz que recibe a nivel de textura en lugar del normal de cada cara individual.

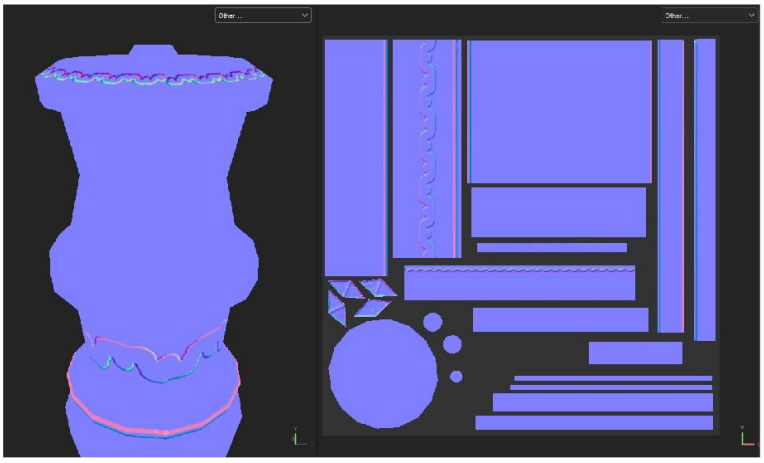


Figura 11. Normal map en modelo de lampara

2.4.5 Height Maps

De acuerdo con Pluralsight Creative, “lo único que hacen es que crea información de alturas utilizando únicamente escala a grises” (2014). Con esto podremos interpretar que crea la ilusión de tener más polígonos para crear sombras sobre el polígono tomando en cuenta que los negros son más profundos que los blancos que son más propensos a ser impactados por la luz, estos pueden ser implementados al momento de querer simular dobleces de ropa o de piel.

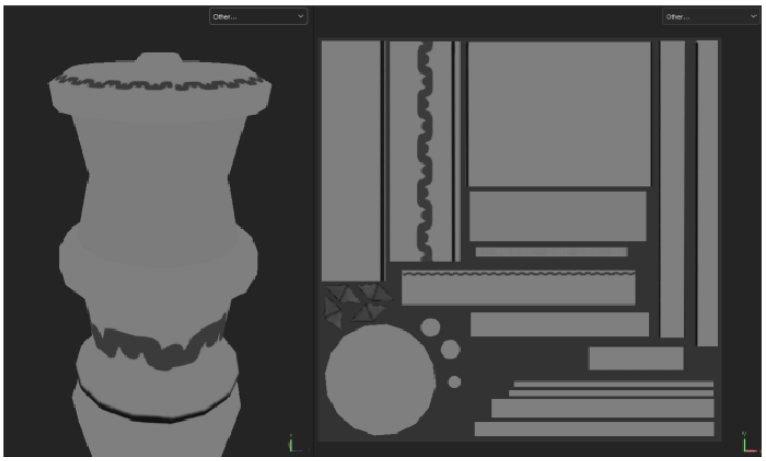


Figura 12. Height map en modelo de lampara

2.4.6 Opacity Maps

De acuerdo con Andrew de Rewote, “Es un mapa de escala de grises en las que el negro es totalmente transparente y el blanco no es transparente” (2020), por ende, es

un mapa a escala a grises que cuyo único propósito es el controlar el Alpha de ciertas partes de un modelo 3D.

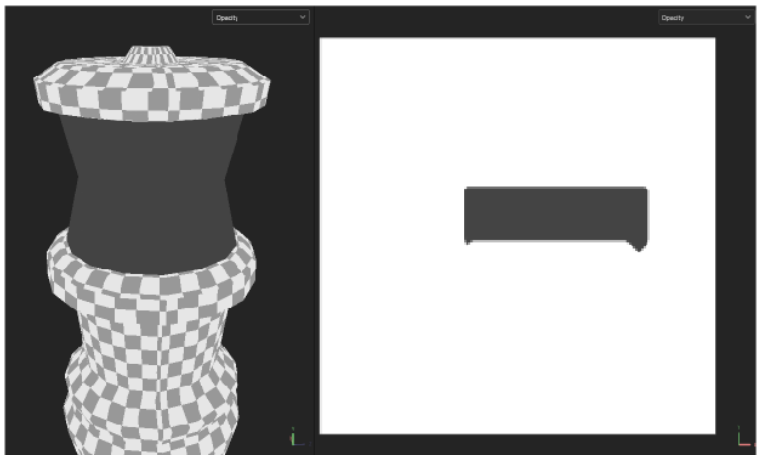


Figura 13. Opacity map en modelo de lampara

2.4.7 Emisive Maps

En palabras de Autodesk, “Un emisive map usa el color para simular el efecto de brillo dentro de una textura. El efecto no va más allá de los límites de un objeto” (2017). Esto da a entender que el emisive map usa colores asignados para dar un brillo a una zona específica del mapa de texturas, este por lo general lleva un fondo negro que indica las áreas que no tendrán dicho brillo y los que estén con color brillarán del color asignado.

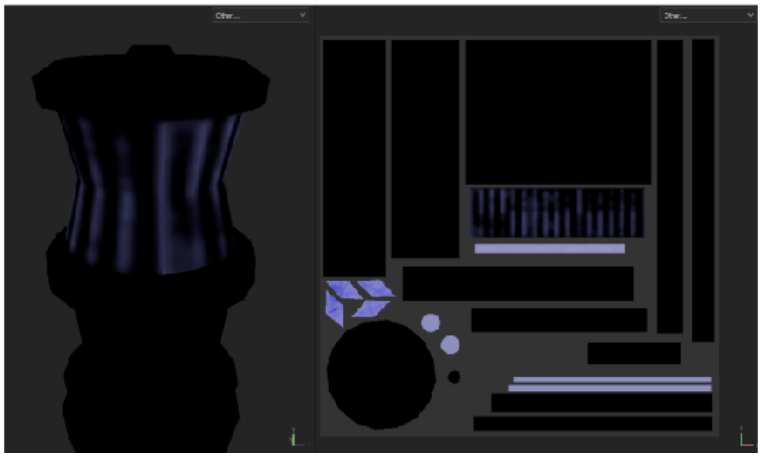


Figura 14. Emisive map en modelo de lampara

2.4.8 Diffuse Maps

En palabras de Royal Skies LLC, “Lo que representa un mapa diffuse es que la luz esta balanceada. En un mapa diffuse adecuado es posible agregar sombras de manera intencional pero no se puede saber de dónde viene la luz” (2019). En otras palabras, la iluminación de las texturas es neutra, pero puede contener luces y sombras controladas que dan la ilusión de detalle sin haberlo cuando se ve de lejos, pero esta no es del todo exacta puesto que queda a discreción de los conocimientos y habilidades del diseñador.

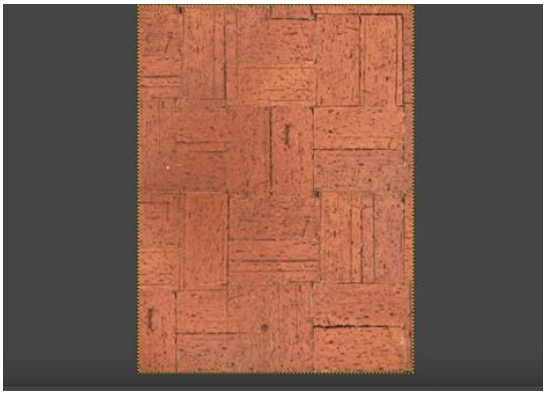


Figura 15. Ejemplo de Diffuse Map

2.4.9 Specular Maps

Según Royal Skies LLC, “un specular map tiene la propiedad de que en las áreas blancas son las que más reflejan luz y en las negras no reflejan luz” (2019). En otras palabras, reflejan las áreas con respecto del ángulo de la luz y únicamente reflejan luz.

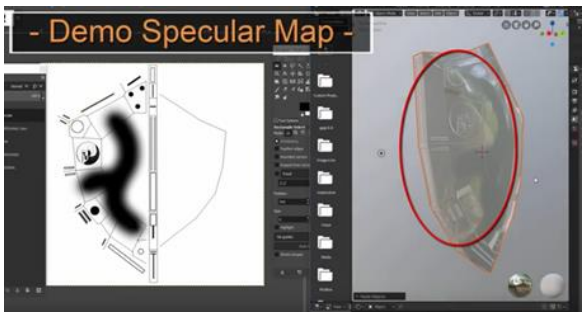


Figura 16. Ejemplo de Specular Map

CAPÍTULO 3: METODOLOGÍA

INTRODUCCIÓN

Dentro de este capítulo se detallarán aspectos metodológicos de la investigación, ya sean procedimientos, métodos y técnicas que se emplearán para lograr desarrollar las propuestas esperadas.

3.1 Metodología

La metodología es una de las ramas naturales de un proceso de desarrollo, para lograr un resultado coherente se requiere tener una idea general de lo que se espera llegar, por ende, al tener una idea, se requiere también entender que se debe seguir un proceso, o una serie de pasos para llegar al final del desarrollo. Según Coelho, “La metodología funciona como el soporte conceptual que rige la manera en que aplicamos los procedimientos en una investigación” (2019).

3.1.1 Tipo de investigación (Paradigma)

El tipo dentro de la investigación será de tipo cuantitativo, ya que se espera poder coleccionar información fácilmente medible, con base al análisis del posible grupo a investigar, al elaborar una aplicación, podemos medir cuantitativamente y de manera subjetiva vía la interacción con el usuario y el software, coleccionando los resultados, examinando y detallando las conclusiones a partir de la información colectada.

3.1.2 Alcance de la investigación

El alcance de la investigación a emplear será de tipo descriptivo, ya que se espera observar, comprender y examinar el conocimiento y la reacción de un grupo especifico de personas al estar expuestas a una aplicación de software de referencia (Benchmarking o rendimiento), y en base a los conocimientos de este grupo, observar sus experiencias con la aplicación.

3.1.3 Diseño de la investigación

Ya que se observarán y analizaran distintos dispositivos y su rendimiento dentro de la aplicación comparando los distintos niveles de detalle, además se encuestará a la muestra sin intervención, se considera que el diseño es de tipo no experimental, por razones de que no se interferirá o manipulara de ninguna manera los resultados, solo se observaran y analizaran.

3.1.4 Muestra u objeto de estudio

Parte de la muestra son los alumnos de la Universidad Autónoma de Ciudad Juárez que se encuentran en la carrera de Diseño Digital de Medios Interactivos, que hayan cursado las materias de Games Engines I & II (intermedio-avanzado), o con interés de desarrollar aplicaciones de videojuegos.

La muestra estará enfocada en un grupo de 33 alumnos que estudian la carrera de Diseño Digital de Medios Interactivos. (~150 Alumnos, con confianza del 95%, y un margen de error del 15%)

En cuanto a la aplicación, se enfocarán las pruebas de rendimiento a 20 distintos dispositivos segregándolos por el tipo de configuración de hardware y su nivel de rendimiento por estándares asignados numeralmente.

3.1.5 Instrumento(s) de recolección de datos

Se espera que, a partir de la interacción con la aplicación de referencia, se apliquen distintos tipos de encuestas estructuradas de pregunta cerrada (Likert, Selección múltiple, etc.) para medir los resultados de experiencia del usuario.

Además, en base a la aplicación, se correrá una evaluación de rendimiento o benchmarking, comparando los distintos tipos de técnicas mencionadas en la investigación, los efectos de dichas técnicas de optimización serán utilizadas y comparadas, eventualmente se evaluarán los resultados obtenidos a partir de la medición de rendimiento y fotogramas por segundo. Además, se anotarán los resultados de las distintas técnicas y como se comparan entre ellas y sin ellas.

3.2 Plan Metodológico

Se espera que, al término de la investigación, se desarrolle un prototipo de aplicación creada en Unity, el enfoque principal de esta aplicación es generar una tabla de referencias dentro del dispositivo, esta tabla de referencias comparara distintas técnicas de optimización y texturización, o la carencia de estas, midiendo las fluctuaciones de rendimiento y fotogramas por segundo, a partir de distintos tipos de dispositivos seleccionados.

Se empleará la metodología de desarrollo de software: Incremental, propuesto por Harlan Mills en el año 1990, para el desarrollo de la aplicación de referencia.

Las etapas de desarrollo de software incremental se emplearán tantas veces sea necesario para desarrollar secciones de la aplicación de manera dinámica, siguiendo las siguientes etapas:

1-Comunicación: Generación de ideas y requerimientos o limitantes

2-Plan Rápido: Se planea rápidamente la iteración a elaborar

3-Modelado Diseño Rápido: Diseño y programación visible para el usuario

4-Construccion de Prototipo: Se hace la construcción del prototipo y las ideas

5-Desarrollo, Entrega y Retroalimentación: Se evaluarán los resultados con participantes.

Para el resto del desarrollo, se generarán las siguientes fases:

3.2.1 Fase 1: Gestión e Investigación

En esta etapa se espera que se hagan las investigaciones e indagaciones iniciales de desarrollo de la aplicación a desarrollar, se espera asignar las posibles técnicas de optimización y texturización a aplicar dentro de la aplicación, al mismo tiempo, se desarrollara los aspectos iniciales del documento de investigación.

- Indagación inicial
- Asignar Técnicas a usar en App
- Selección de programas a usar

3.2.2 Fase 2: Diseño y Desarrollo

En esta etapa se espera que se inicie el desarrollo de la aplicación y encuestas que se asignaran a las posibles muestras, se desarrollaran todos los aspectos visuales

que puedan darse a requerir dentro de la ella y se continuara con el documento de investigación, se detallaran dentro del documento aquellos aspectos que sean relevantes, y se generaran las dudas iniciales del desarrollo, que podrían ser inquiridas y resueltas por otros posibles expertos del tema (programadores, diseñadores, etc.)

- Inicio de Desarrollo de App
- Asignar / Inicio de diseño de Encuesta
- Desarrollo Visual / Diseño
- Resolución de dudas iniciales dentro de la app

3.2.3 Fase 3: Pruebas, Análisis, y Producción

En esta etapa se esperan avances al desarrollo de la aplicación de referencia, se generarán otras posibles inquisitivas a solucionar, se trabaja en el proyecto mientras se prueba al mismo tiempo, para tener resultados a tiempo real y se trabajara de manera más cercana con expertos de programación para obtener un mejor resultado final dentro del desarrollo.

- Avances de Desarrollo de la App, a la cercanía o finalizado (Pre-Alpha)
- Generación de dudas extras para resolver
- Reunión con expertos para inquirir dudas

3.2.4 Fase Fin 1: Pre-Lanzamiento, Encuestas y Búsqueda de Resultados

En esta etapa se espera lograr la finalización del prototipo y aplicación para iniciar la experimentación con la muestra seleccionada, se aplicarán las pruebas de uso en ellos y se les encuestara para la obtención de datos, además, se harán las pruebas pertinentes de uso en distintos dispositivos para obtener datos de referencia comparativa y tener

mayor rango de información y obtener conclusiones con el menor porcentaje posible de variación y subjetividad.

- Finalización del prototipo de la app (pre-Alpha)
- Aplicación de pruebas de uso
- Aplicación de encuestas
- Pruebas pertinentes del uso de la app en distintos dispositivos
- Obtención de datos

3.2.5 Fase Fin 2: Resultados

En esta etapa se espera lograr dar los resultados de la investigación, se analizará con detalle para llegar una conclusión lógica a partir de la información obtenida, y por último se finalizará el documento de investigación presente con todos los detalles generados a partir de los resultados.

- Obtención de resultados de investigación
- Análisis con detalle de los resultados
- Conclusiones

CAPÍTULO 4: DESARROLLO DE LA PROPUESTA

INTRODUCCIÓN

Dentro de este capítulo se detallará el proceso del desarrollo del proyecto, la propuesta de la resolución del problema, dividido en las fases asignadas en el capítulo anterior. Solo se mencionarán las primeras tres fases del desarrollo, las cuales son enfocadas al inicio y la finalización del pre-alpha.

4.1 Fase 1: Gestión e investigación

Al analizar que nuestro objetivo es demostrar la importancia de las distintas técnicas de optimización de polígonos, texturización y modelado tridimensional en la industria de los videojuegos, debemos de conocer lo que se requiere para lograrlo, tanto en relación con la utilización de software, como los requerimientos para llevar a cabo los objetivos destacados.

4.1.1 Requerimientos de software

Para poder desarrollar la aplicación, se requieren las siguientes características para el desarrollo del proyecto:

1. Un motor de juegos que tenga la capacidad de manejar múltiples modelos tridimensionales, que dentro del software se puedan programar scripts o acciones específicas (movimientos, cambios de escena y modelos), capaces de realizar una interfaz con el que el usuario pueda mostrar estos distintos cambios y pueda observar el entorno.

   Seleccionado: UNITY

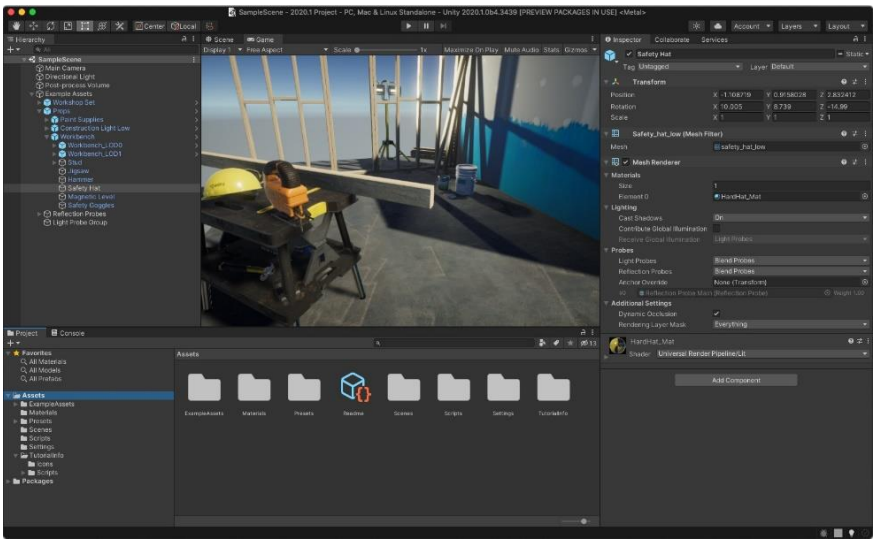


Figura 17. Ejemplo de interfaz de Unity

Unity es un motor de juegos enfocado al rango indie y corporativo AAA, cuyo enfoque es el desarrollo de aplicaciones y videojuegos tridimensionales / bidimensionales, su manejo de shaders y texturas es muy buena y aun si esta es una aplicación de paga, esta es gratuita para proyectos con fondos menores de 2 millones de pesos, además, este tiene una tienda de recursos gratuitos y de paga para cualquier usuario que usa Unity.

La dificultad de uso es intermedio-avanzado, empleando el C# como su lenguaje de programación e integración, usando terminologías y referencias de la industria dentro de la interfaz, lo cual auxilia a que cualquier programador pueda transferirse de otro motor a juegos a Unity.

Unity además tiene buenos recursos, tutoriales y referencias como manuales de uso y bibliotecas de información, es usado por un aproximado de dos billones de desarrolladores en el mundo (Unity, 2022)

2. Una aplicación de modelaje tridimensional donde se puedan desarrollar distintos modelos de distintas complejidades, con la capacidad de editar mapas de UV, asignar texturas, animar, y posar los modelos.

Seleccionado: Blender

Figura 18. Ejemplo de interfaz de Blender.

Blender es una aplicación gratuita y de código abierto (Open Source), que se actualiza constantemente y con más de 14 millones de descargas (Blender, 2022), aun cuando se tiene una amplia gama de aplicaciones tridimensionales, Blender es un programa de uso e interfaz amigable y fácil de usar, que tiene una amplia cantidad de recursos, tutoriales, e información externa, con extensiones de amplias gamas gratuitas y de paga.

3. Una aplicación de manejo de texturas, donde se pueden aplicar y editar los distintos mapas de texturas de manera rápida y dinámica, que tenga un buen manejo del realismo, capaces de generar texturas fluidas (Seamless) optimas, que acepte la adición de brochas y vectores.

   Seleccionado: Substance Painter

Figura 19. Ejemplo de interfaz de Substance Painter

A pesar de que Blender implementa el manejo de edición y generación de texturas, Substance Painter es una aplicación increíble en el desarrollo y manejo de texturas de un objeto tridimensional. SP es una aplicación visualmente entendible y de fácil manejo, con una módica cantidad de recursos externos, y aunque esta tiene dos tipos de costos, de un solo pago o de membresía mensual / anual (con beneficios extras), su uso es vital para el desarrollo del proyecto por la agilidad de texturizado del modelado tridimensional, el procesamiento de objetos de manera inteligente, buen manejo del realismo y texturización optimizada, manejo de brochas y vectores editables, además de reducir el peso total de cada objeto. SP reduce el tiempo de texturizado de los objetos al 70% a comparación de Blender.

4. Un entorno de desarrollo que sea compatible con el motor de juego seleccionado, para lograr scripts y código, que sea fluido y se pueda trabajar en tiempo real.

   Seleccionado: Visual Studio

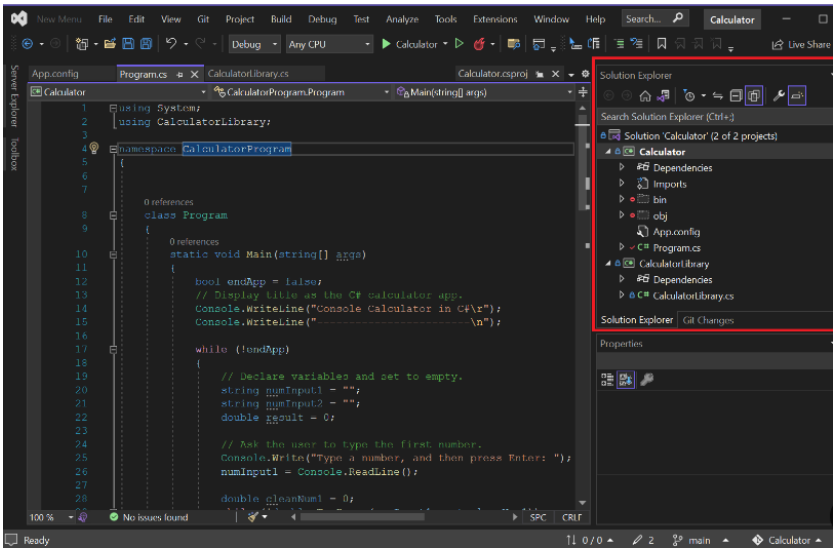


Figura 20. Ejemplo de interfaz de Visual Studio Professional

Visual Studio es una IDE inteligente bastante compatible con Unity, además de tener la capacidad de edición de código en tiempo real, esta emplea acciones rápidas y auto llenado de palabras y código, lo cual reduce el tiempo de programación total.

5. Un editor de imágenes para lograr hacer ediciones a los distintos mapas de UV y texturas a generar.

   Seleccionado: Photoshop

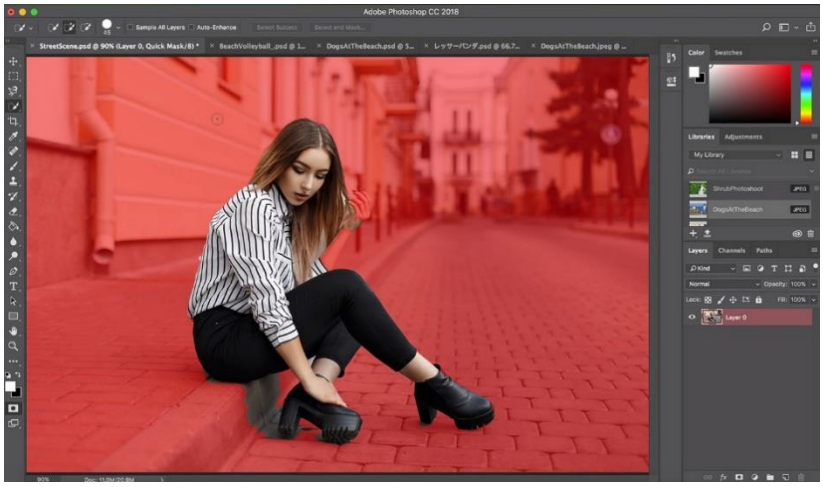


Figura 21. Ejemplo de interfaz de Photoshop

Photoshop es un buen programa para edición rápida de imágenes y mapas de UV, tiene muchas funciones para selección de color, capas, manejo de tonos, calidad de tamaños, compresión, entre otras cosas. Photoshop será empleado en este caso para ediciones rápidas de mapas en caso de manejar distintos mapas de UV.

6. Una aplicación de bocetaje con herramientas de reglas y simetría para la generación de ideas y dibujos para armar los objetos tridimensionales

   Seleccionado: Clip Studio Paint

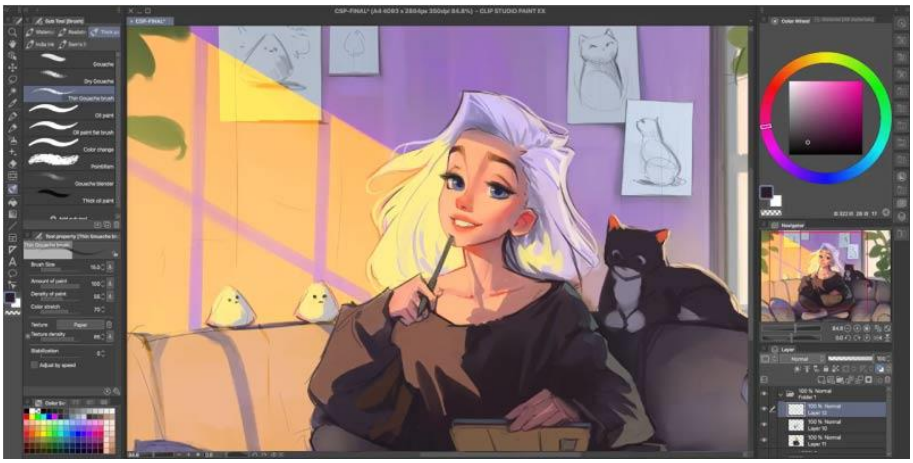


Figura 22. Ejemplo de interfaz de Clip Studio Paint

Clip Studio Paint es una aplicación cuyo enfoque principal es el bocetaje y diseño de arte profesional, además del buen manejo de capas y brochas, tiene buenas herramientas de reglas y simetría que se separan dependiendo de la capa seleccionada, estas herramientas pueden ser activadas y desactivadas libremente, es capaz de manejar vectores y tiene compatibilidad con tabletas de dibujo para mejor flujo de trabajo y velocidad.

7. Una aplicación de manejo de vectores que pueda crear imágenes.

   Seleccionado: Ilustrador

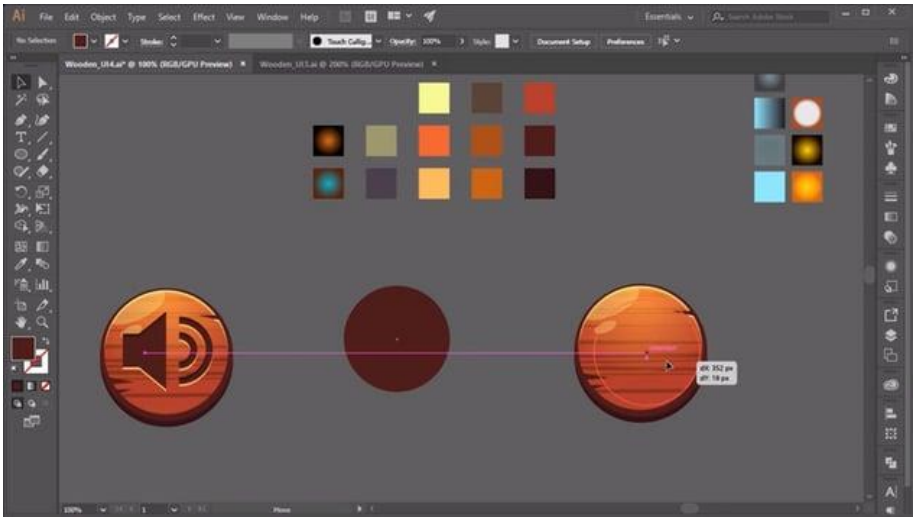


Figura 23. Ejemplo de interfaz de Illustrator

Ilustrador es un programa de alta gamma de manejo de vectores, al tener una interfaz similar a Photoshop, la adaptación entre uno y otro es relativamente fácil, además del buen manejo de capas y colores, auxilian para crear patrones de vectores para lograr estampados dentro de Substance Painter.

4.1.2 Requerimientos de aplicación

Ya que fueron seleccionados los softwares para elaborar el proyecto, se deben de considerar los siguientes aspectos:

1. Se requiere que el espacio demostrativo sea un lugar lo suficientemente extenso para demostrar las mejoras a partir de las distintas técnicas de modelado y texturización
2. Se requiere tener una módica cantidad de objetos tridimensionales para lograr observar las fluctuaciones de rendimiento en los distintos dispositivos.
3. Que los objetos tridimensionales puedan ser observados a distintas distancias, para demostrar las distintas técnicas de nivel de distancia (LOD's)
4. Los modelos se puedan comparar y mostrar las diferencias entre las distintas optimizaciones o la falta de ellos.

5. La información que se requiere para cuantificarla y analizarla pueda ser ágilmente mostrada en tiempo real dentro de la aplicación, sin la necesidad de instalar otras aplicaciones:
    a. Modelo de GPU & CPU
    b. Cantidad de RAM
    c. Porcentaje de uso de GPU & CPU
    d. Fotogramas por Segundo (FPS)
    e. Sistema Operativo

Para lograr resolver lo expuesto anteriormente se seleccionó como referencia inicial un lugar popular de la serie World of Warcraft; Stormwind Keep, el castillo de la alianza, ya que tiene distintos modelos naturales como artificiales, se tomarán algunos cambios y se evitara emplear objetos representativos (Logo, colores, etc.)

Figura 24. Referencias de Stormwind Keep, World of Warcraft

Se genero un mapa basado en las referencias (con el mapa original) y se elaboró un resumen general de los modelos esperados a elaborar para la aplicación:

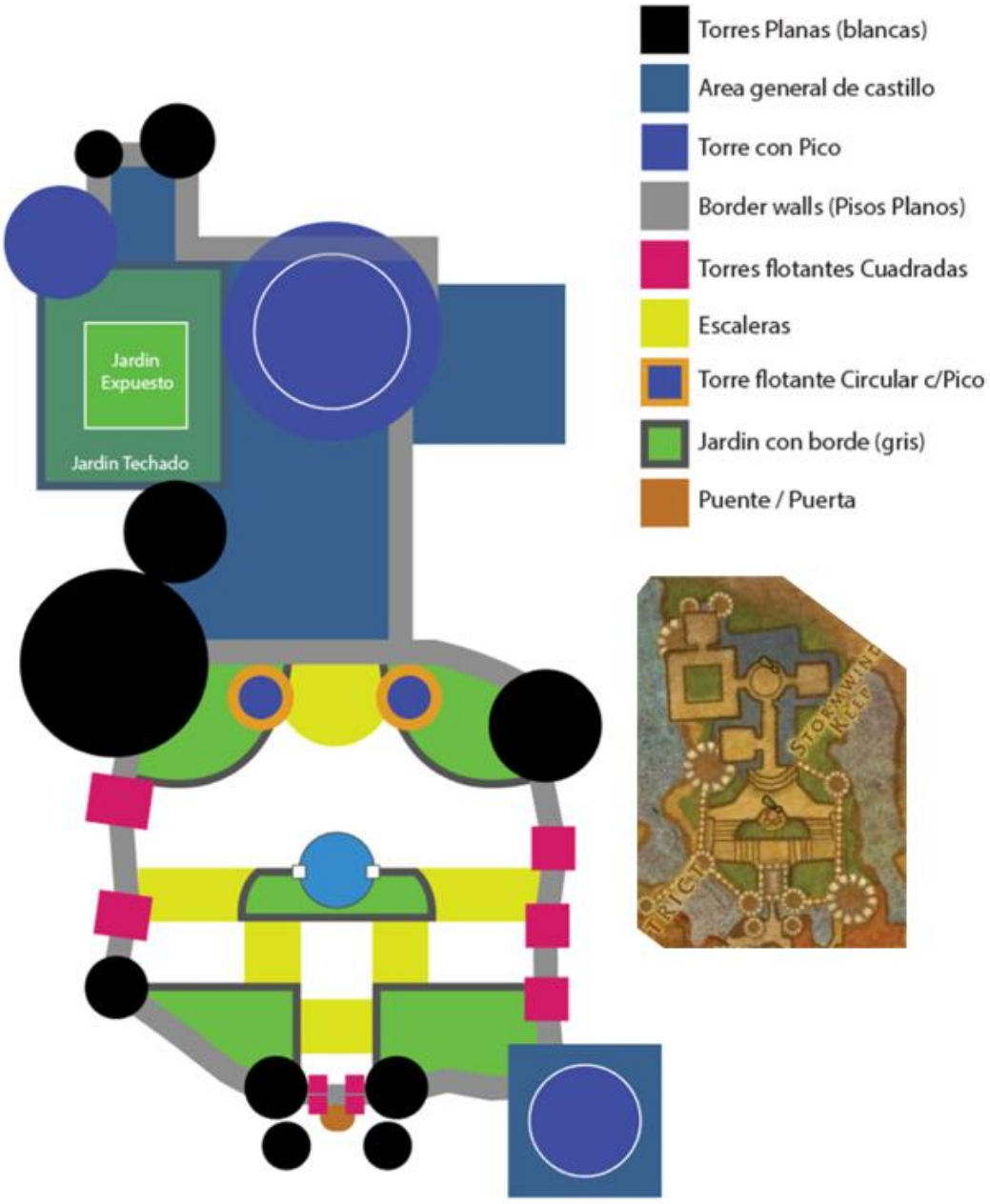


Figura 25. Mapa 2D de Stormwind Keep en conjunto con su mapa original

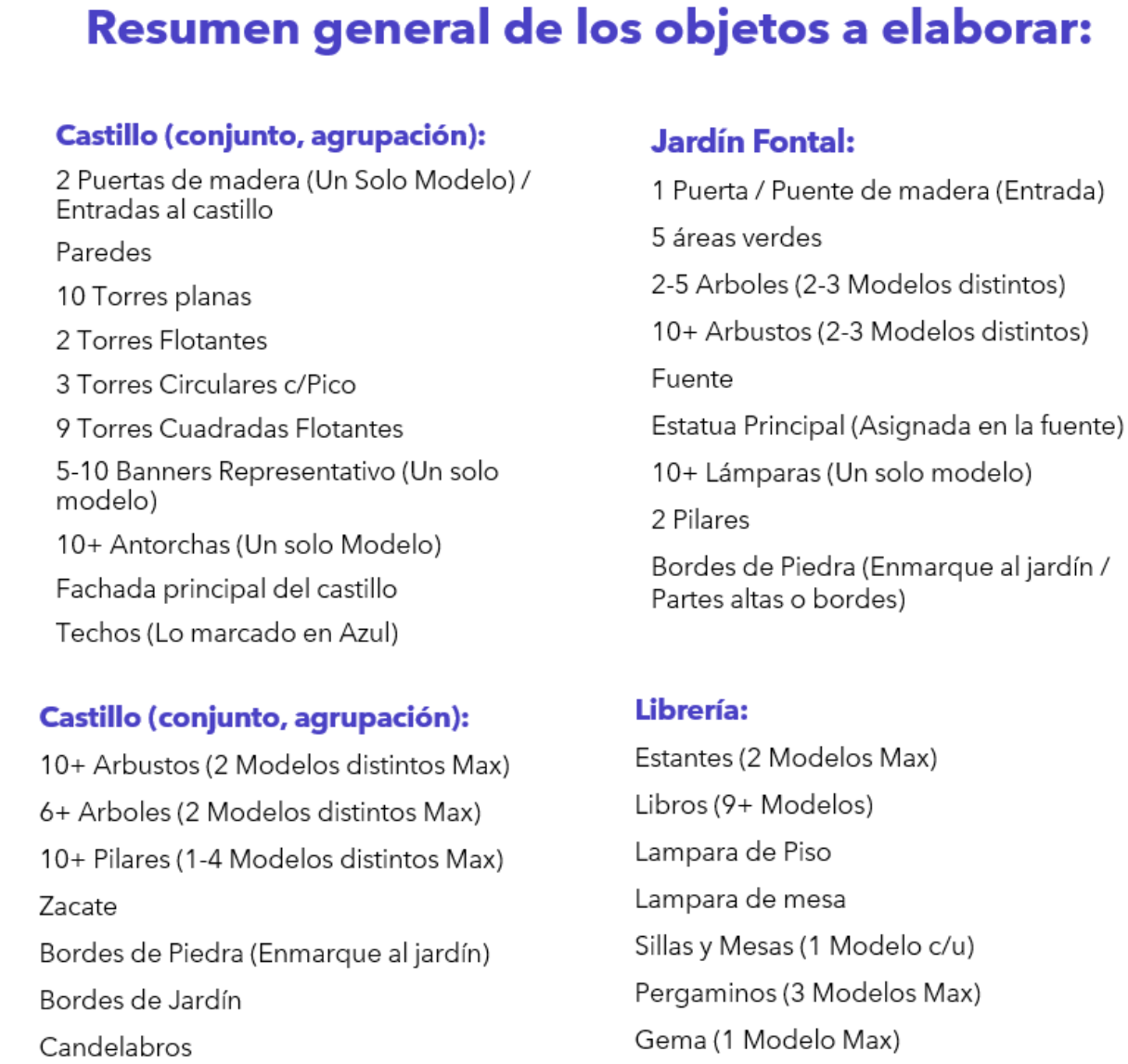


Figura 26. Resumen general de los objetos esperados a elaborar dentro de la aplicación

A partir de este resumen general, se elaborarán por objeto, 5 distintos tipos de modelos con calidades especificas:

1. Óptimo (LOD 0): Este objeto tendrá una cantidad justa de polígonos para un videojuego, empleará distintas mejoras de modelado y texturizado.

2. LOD 1: Nivel de distancia 1, este objeto será una iteración del óptimo en el cual se reducirá la cantidad de polígonos en un aproximado del 50-60% dependiendo la distancia (El objeto es lejano, pero no pierde la forma)
3. LOD 2: Nivel de distancia 2, este objeto será una iteración del LOD 1, en el cual se reducirá la cantidad de polígonos a lo más bajo posible. A diferencia del anterior, este si puede deformarse debido a la distancia de la cámara el cual no es perceptible el cambio de forma.
4. High-Mid: Simula el nivel de detalle de un objeto no muy optimizado, con alta cantidad de polígonos y detalles generados dentro del modelo, para este proyecto se simulo con el uso de subdivisión nivel 2 a partir del modelo óptimo.
5. High: Simula una el nivel de detalle de un objeto esculpido en 3D, sin optimizaciones ni retopología, para este proyecto se simulo con el uso de subdivisión nivel 2 a partir del modelo óptimo.

4.2 Fase 2: Diseño y Desarrollo

En base a la metodología de desarrollo de software incremental, se agregarán modelos por partes para lograr el resultado esperado. Se inicio primero con la elaboración de los modelos tridimensionales, el flujo de trabajo por objeto será el siguiente:

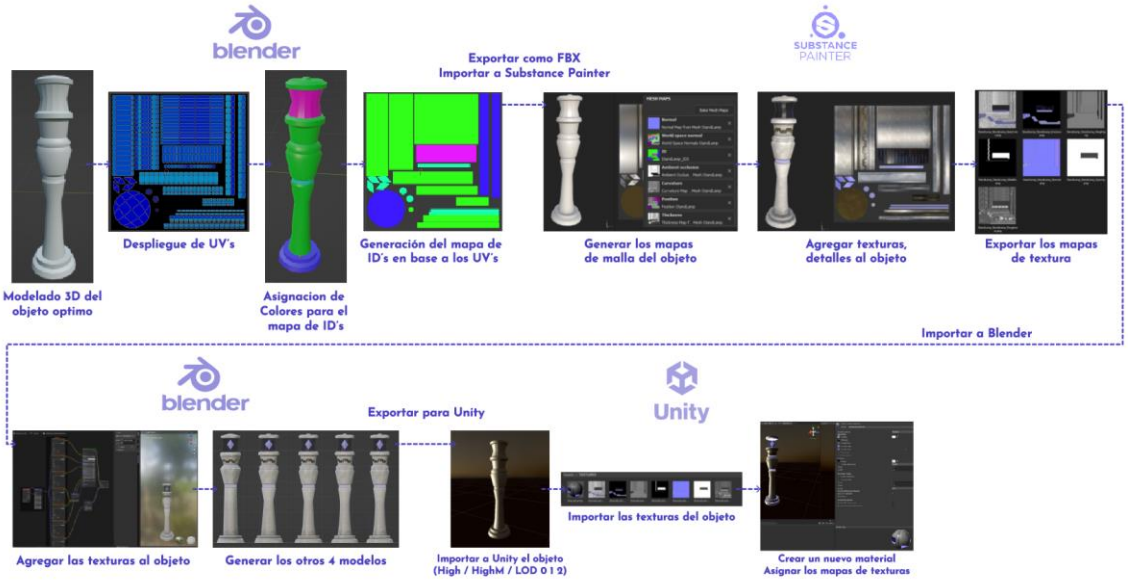


Figura 27. Flujo de trabajo por modelo 3D

Blender:

Se inicia Blender para la creación del objeto básico.

1. Modelado 3D del objeto Óptimo (LOD 0)

   Se crea el modelo base, se utilizan técnicas como extruir, cortes, pliegues (Creases), y se asignan orillas (Sharps) para formarlo, idealmente las aristas no deben de estar muy unidas entre ellas, debe haber un espacio justificable. El objeto puede demostrar curvas y divisiones sin la necesidad de ser muy agresivas o ultra detalladas, dentro de este proceso, no es necesario emplear el método de subdivisión para lograr un modelo óptimo, al aplicar las técnicas anteriores de pliegues y orillas se puede evitar la utilización de estas técnicas no ideales para un modelo dentro del desarrollo de modelos optimizados.

   Dependiendo del objeto (a excepción de modelos vivos), los tris esperados son de entre 1000 a 4000 para un modelo óptimo.

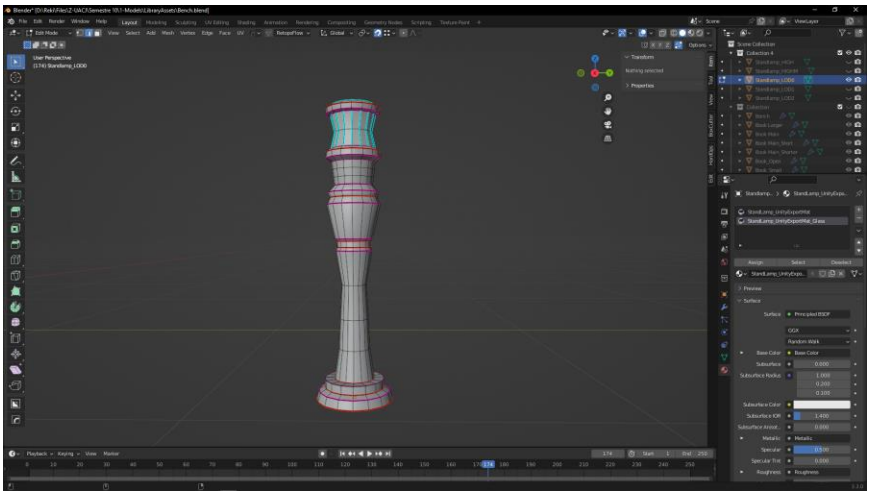


Figura 28. Modelo Óptimo

2. Despliegue de UV’s

   Una vez finalizado el modelo, se deben de generar los UV’s para lograr texturizar de manera apropiada. Las islas de UV idealmente deberán estar ordenadas, limpias, sin deformaciones ni entre puestos, con espacios justos entre isla e isla y que ocupen el mayor espacio posible del área cuadrada dentro de la edición de UV.

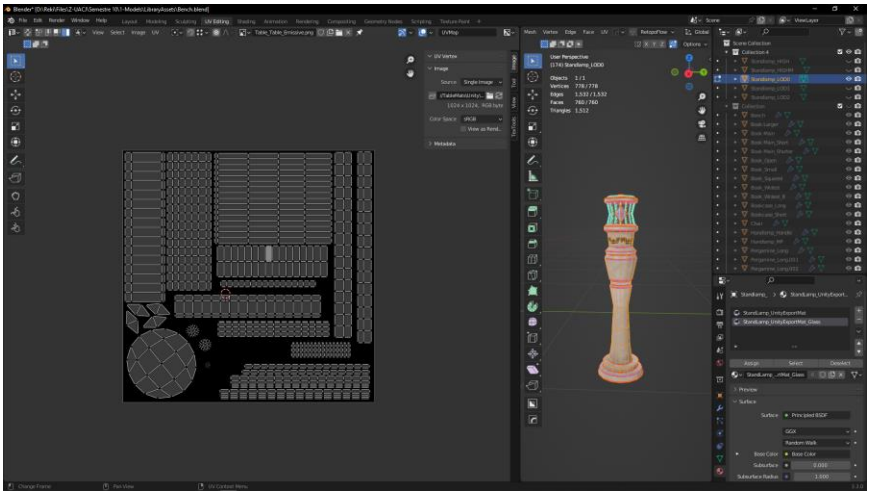


Figura 29. Modelo Óptimo y sus UV’s

3. Asignación de colores para mapa de ID’s

   En esta etapa se asignan los colores o ID’s que el programa de texturización empleara para separar los distintos materiales, en el caso del modelo mostrado, el color verde se utilizó para denotar texturas de piedra en tonos claros, el azul oscuro para piedras más negras, el turquesa para áreas de emisión, o luz, y el rosa para el vidrio.

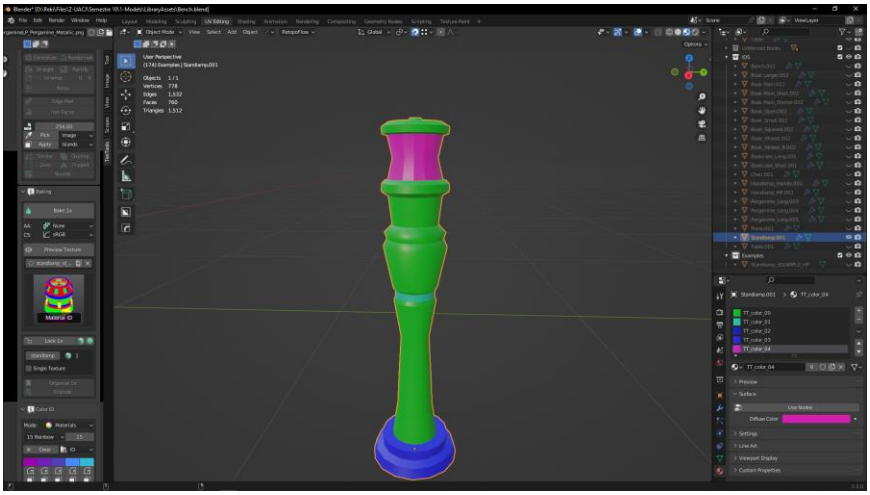


Figura 30. Modelo Óptimo con asignación de colores

4. Generación del mapa de ID’s en base a los UV’s

   Una vez asignados los colores, se generará la imagen que se transferirá al programa de texturización, se exportará la imagen y el modelo al siguiente programa. El tamaño de la imagen guardada es dependiente a la cantidad de objetos conjuntos o la calidad de las texturas.

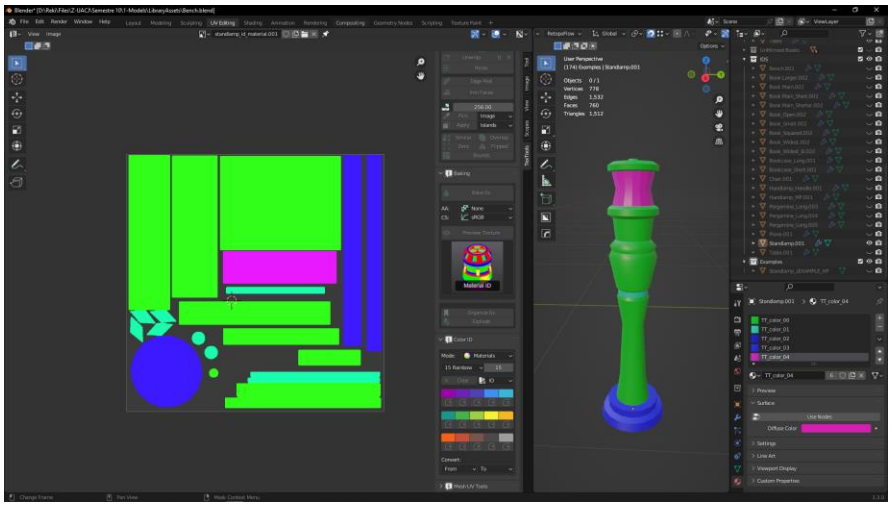


Figura 31. Modelo Óptimo con asignación de colores + UV’s

Substance Painter:

En este programa se editarán y generarán las texturas, mapas de texturas, y detallado general del objeto.

5. Generar los mapas de malla del modelo

   Al iniciar Substance Painter, se deberá importar el modelo, una vez importado, se harán los mapas de malla autogenerados por SP, es ideal que el objeto a generar separe ciertas partes del mismo para evitar sombras no deseadas (Por ejemplo, pisos y pilares distanciados de sí mismos, a menos que estos siempre estén juntos y no utilicen patrones), al generar los mapas de malla, se importará el mapa de ID’s generado anteriormente en Blender, y se le asignara en su espacio dedicado dentro de la aplicación. Mapas generados:

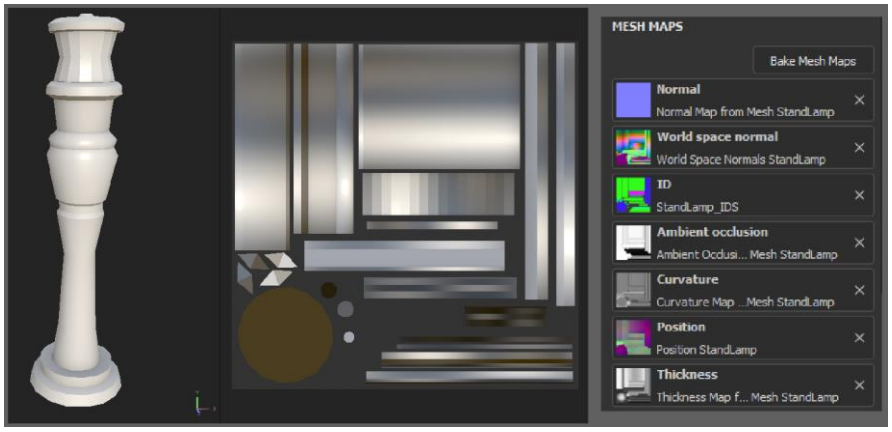


Figura 32. SP. Referencia de mapas de malla modelo base con los mapas autogenerados

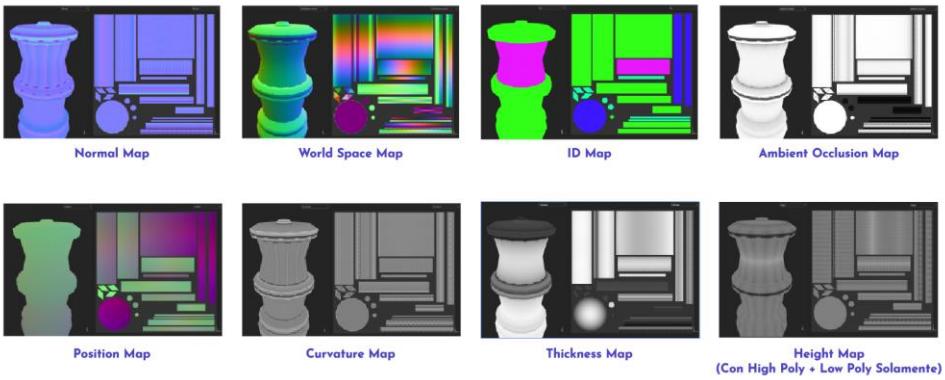


Figura 33. Mapas Autogenerados por Substance Painter

6. Agregar texturas y detalles al modelo

   Esta etapa es más libre y dependiente al diseñador, similar a aplicaciones como Photoshop e Illustrator, Substance Painter utiliza capas, máscaras y rellenos para lograr segregar y asignar las texturas, se utilizan brochas y se asignan su enfoque (por ejemplo, la edición de height maps + metálicos para crear detalles que sobresalen y son metalizadas) para lograr alzar detalles y darle mayor vida al objeto.

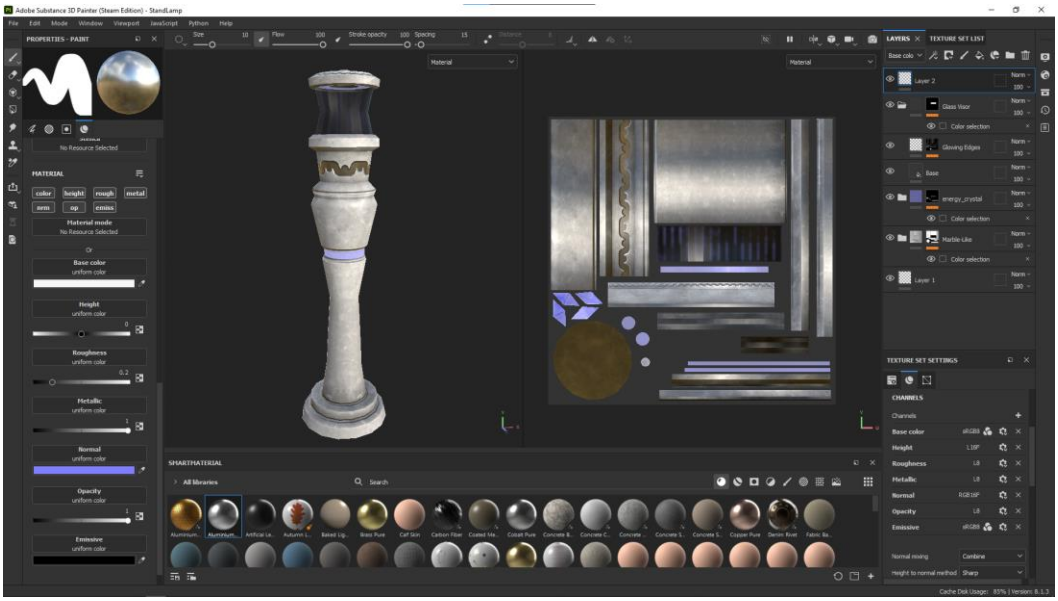


Figura 34. Modelo finalizado con texturas

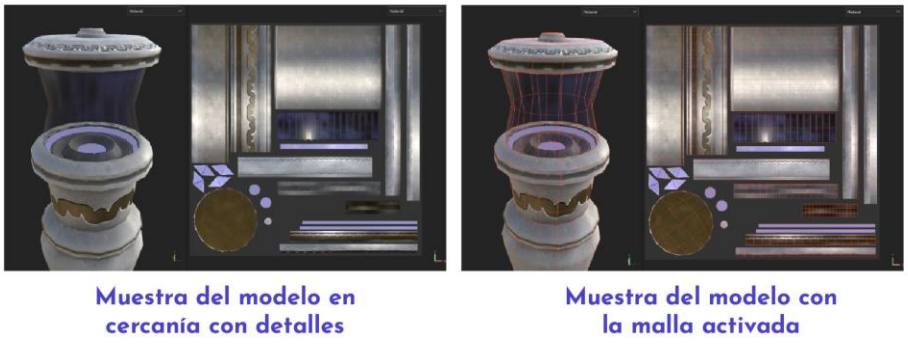


Figura 35. Referencia entre modelo texturizado con y sin malla

7. Exportar los mapas de textura

   Una vez finalizado el diseño, se deberán exportar las texturas para utilizar en Blender, estas texturas serán empleadas en base a lo que se utilizó, por ejemplo, si no se emplearon áreas metálicas, el mapa metálico tendrá menor prioridad que otros mapas, en el caso de este modelo, todos los mapas se utilizaron, incluyendo los mapas de opacidad.

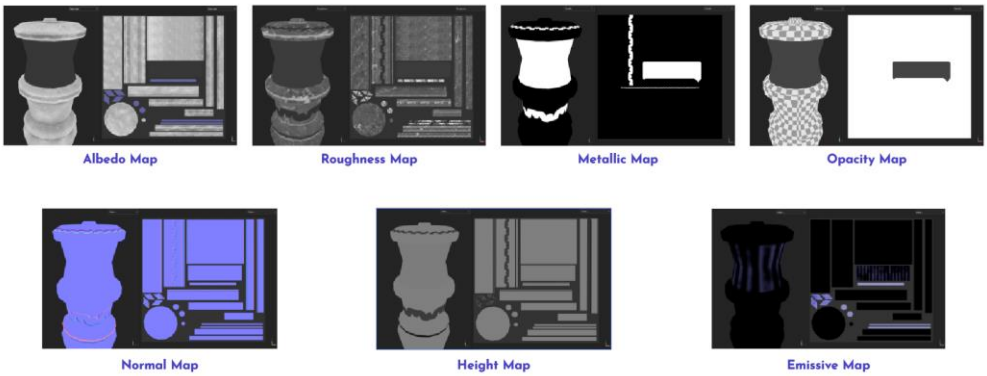


Figura 36. Mapas de texturas del modelo

Blender, Parte 2:

Se regresa al archivo original para importar y generar los siguientes modelos.

8. Agregar las texturas al modelo

   Se importan vía nodos en Blender las texturas empleadas, para lograr observar las texturas de vidrio se requiere configurar otros aspectos (Alpha Clip). Puede que entre modelos cambie un poco, pero es relativamente el mismo estilo, solo removiendo los mapas de texturas que no se utilizaran.

   Idealmente, se utilizará el estilo de shading dentro de Blender para lograr observar los resultados de manera rápida.

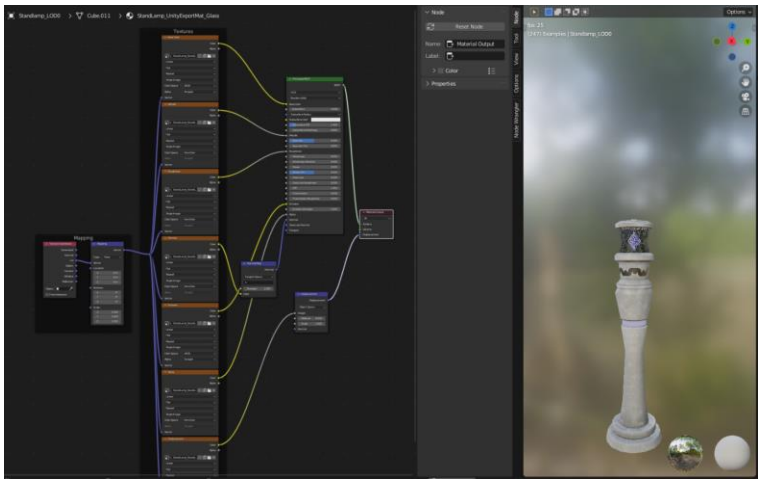


Figura 37. Nodos importados para blender + muestra del funcionamiento

9. Generar los otros 4 modelos (LOD 1, LOD 2, High, Mid-High)

   Los nuevos cuatro modelos siguen un proceso especifico dependiendo de la calidad y el tipo:

   LOD 1: este modelo no debe perder mucho de la forma original del objeto, pero si debe tener reducciones, se pueden eliminar aristas no vitales del modelo. La reducción aproximada es de la mitad de los tris del objeto óptimo.

   LOD 2: este modelo pierde la forma y se modifica a la distancia, se reducen la mayor cantidad de aristas a excepción de las más importantes (aquellas que mantienen la textura intacta. La reducción aproximada es la mitad de los tris del LOD 1

   Mid-High: simulación de un modelo no óptimo, se emplea la subdivisión en 1 a partir del modelo óptimo. La suma de tris es cuatro veces mayor al del objeto óptimo.

   High: simulación de un modelo esculpido (leve), se emplea la subdivisión en 2 a partir del modelo óptimo. La suma de tris es cuatro veces mayor al del objeto Mid-High.

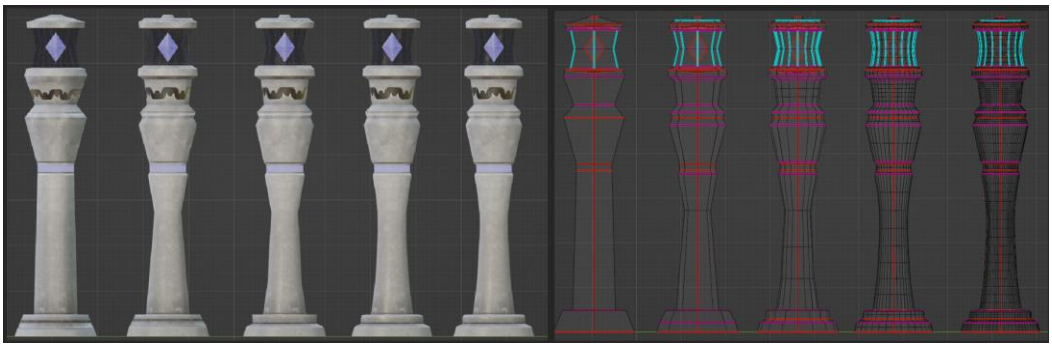


Figura 38. Comparativa entre modelos visuales y sus mallas

En la siguiente imagen, se encuentra la referencia de cada uno de los modelos, su cantidad de tris en comparación de uno a otro.

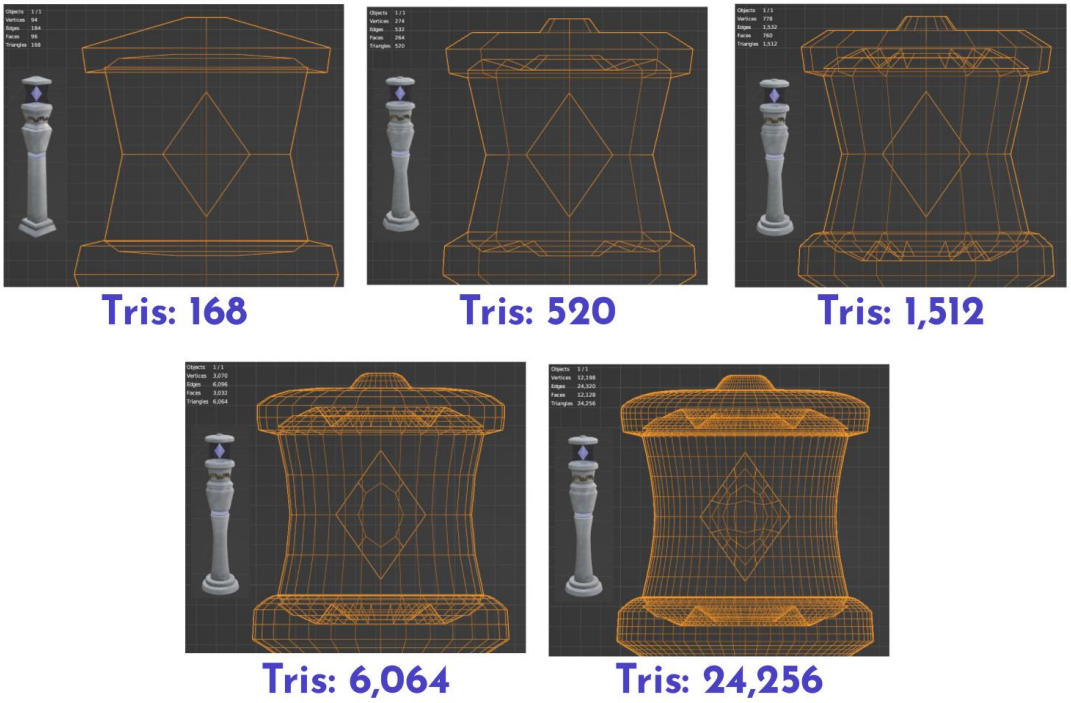


Figura 39. Comparativa entre modelos tridimensionales y su cantidad de tris

Por último, se exportan los modelos en 3 grupos para Unity:

LOD 0, 1, 2 (Juntos)

Mid-High

High

Unity:

Para el motor de juego, donde se importarán los modelos, se deberá crear el proyecto de Unity en tipo 3D para lograr armarlo.

10. Importar los 5 modelos distintos (High / Mid-High / LOD 0,1,2)

Como nota general, para que los modelos estén en las posiciones asignadas en Unity, se deben armar todos los objetos en Blender en sus posiciones, usando de referencia el punto de origen en el centro del mundo (o el espacio) 0 en X, Y, Z.

Se importa primero el modelo de alta calidad (High), después el intermedio (Mid-High), por último, el grupo de 3 LOD's (0,1,2).

Figura 40. Modelo de lampara en Unity

11. Importar las texturas del modelo

    Una vez importados los modelos, se importarán las texturas de manera ordenada y por objeto (utilizando carpetas con los nombres de los modelos originales, y no duplicándolas si se reutilizan), esto para lograr orden con la cantidad de modelos.

Figura 41. Mapas de texturas importados a Unity

12. Crear un nuevo material y asignar las texturas

    Se creará un nuevo material con el shader de Autodesk (los cuales tienen compatibilidad con las texturas importadas de Substance Painter), y se asignaran las texturas importadas en las secciones apropiadas, por último, se le asignara el nuevo material al objeto, reflejando el resultado esperado.

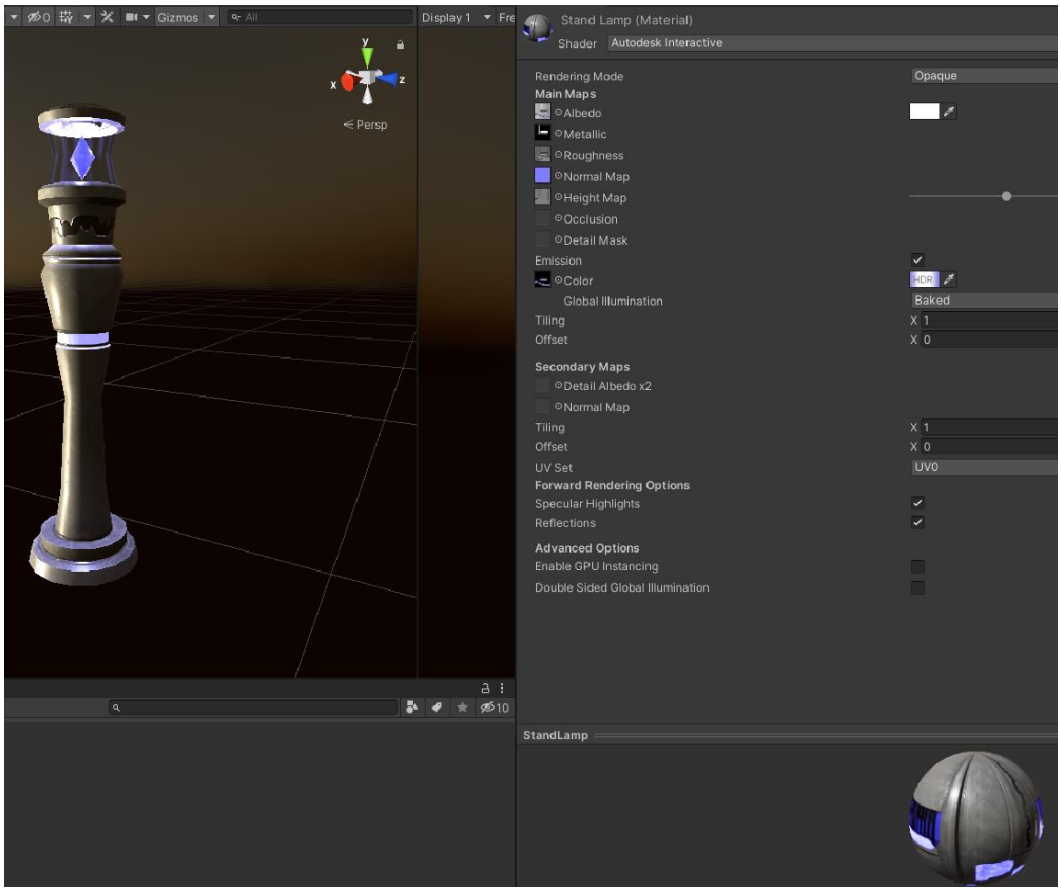


Figura 42. Asignación de texturas dentro del material de Autodesk

Tabla 1. seguimiento de modelos y referencias

Objetos 3D Elaborados	3D	LOD 2	LOD 1	OPT	High M	High	Texturas	Rig?	Seamless?	Reutilizado	Glass	Anim	Text Comp	Fecha Fin	Hecho Por:
1 Puerta de Madera + Marco de Piedra	*	466	1.78k	9.34k	149.6k	2.39M	1			*				15-Sep-22	Victor F.
1 Torre de piso	*	7.8k	49.8k	122.9k	513.9k	2.24M	2			*				12-Sep-22	Victor F.
1 Torre flotante (Con Base)	*	20.3k	84.3k	180.6k	878.2k	5.8M	3			*				13-Sep-22	Victor F.
1 Techo en cono	*	11.7k	29.5k	46.6k	184.8k	739.2k	1			*				13-Sep-22	Victor F.
Bandera con estandarte	*	59	458	978	14.7k	468.5k	1			*				12-Sep-22	Victor F.
Banner	*	674	1.4k	4k	29.9k	109.3k	1			*				11-Sep-22	Victor F.
Fuente + Agua animada	*	12.6k	13k	14k	41.7k	484.3k	2					*		18-Sep-22	Victor F.
Isla Flotante	*	960	3.8k	15.3k	61.4k	245.7k	1							17-Sep-22	Victor F.
Arbusto	*	56	1k	9.7k	35.9k	140.5k	1			*				14-Sep-22	Victor F.
Árbol	*	6.2k	6.8k	6.9k	21.8k	258.9k	1			*				14-Sep-22	Victor F.
Candelabro de piso (Con velas)	*	752	3.6k	9k	151.9k	2.43M	1			*				11-Sep-22	Victor F.
Ladrillos de piso	*	360	1.1k	2k	20.7k	33.1k	1			*				3-Sep-22	Victor F.
Ladrillos de Pared	*	36	268	564	2.3k	9.2k	1			*				8-Sep-22	Victor F.
Puerta principal/Puente	*	256	640	984	16.8k	270k	1							7-Sep-22	Victor F.
Casetas flotantes	*	840	2k	2.6k	10.8k	43.3k	1			*				6-Sep-22	Victor F.
Lampara de calle	*	168	240	892	1.4k	22.9k	1			*				11-Sep-22	Victor F.
Soldado + Lanza	*	3.9k	10.1k	18.3k	288.2k	1.15M	2	*		*					Victor F.
Ventana	*	228	672	1.2k	9.9k	20k	1			*				1-Sep-22	Victor F.
Rocas de la ventana (Marco)	*	20	508	2.1k	17.8k	71.2k	1			*				1-Sep-22	Victor F.
Techo en Triangulo	*	3.1k	19.1k	35k	138.4k	553.6k	1			*				13-Sep-22	Victor F.
Piso base para jardín principal	*	200	200	200	200	200	1			*				17-Sep-22	Victor F.
3 Ladrillos base Pared	*	36	230	500	2k	9.4k	1			*				1-Sep-22	Gladys C.
Lanza de soldado	*	142	282	532	2.1k	8.6k	1			30				3-Sep-22	Gladys C.
Bloques de piso	*	98	236	518	1.9k	7.6k	1			*				2-Sep-22	Gladys C.
Jardín Principal - Pilar	*	176	784	1.8k	7.3k	29.4k	1			10+				5-Sep-22	Gladys C.
Jardín - Externo - Pilar de esquina de dos lados	*	42	512	960	3.8k	15.4k	1						a	15-Sep-22	Gladys C.
Jardín - Externo - Pilar de esquina de un lado (1)	*	33	451	872	3.4k	13.9k	1			2			a	15-Sep-22	Gladys C.
Jardín - Externo - Pilar centrado (1)	*	40	480	968	3.8k	15.4k	1			6			a	15-Sep-22	Gladys C.
Jardín - Externo - Piso Esquinado (1)	*	14	30	66	264	1k	1		*	3			a	13-Sep-22	Gladys C.
Jardín - Externo - Piso Externo (1)	*	10	24	44	176	704	1		*	32			a	13-Sep-22	Gladys C.

Jardín - Externo - Piso con alfombra (1)	*	8	20	38	152	608	1		*	36			a	13-Sep-22	Gladys C.
Jardín - Interno - Zacate (1)	*	8	16	24	96	384	1		*	25			b	10-Sep-22	Gladys C.
Jardín - Interno - Pilar con Zacate (1)	*	44	360	684	2.7k	10.9 k	1		*	4			b	10-Sep-22	Gladys C.
Jardín - Interno - Piso con Zacate (1)	*	20	32	36	144	576	1		*	16			b	10-Sep-22	Gladys C.
Jardín - Interno - Pilar Esquinado Con Zacate (1)	*	52	352	688	2.7k	11k	1		*	4			b	10-Sep-22	Gladys C.
Jardín - Interno - Escaleras (1)	*	120	510	1k	4k	16.6 k	1			8			b	8-Sep-22	Gladys C.
Jardín - Banca (1)	*	396	824	1k	4k	16k	1			12				17-Sep-22	Gladys C.
Librería - Lampara de Piso	*	168	520	1.5k	6k	24.2 k	2			8	*			16-Sep-22	Gladys C.
Librería - Lampara de Mano	*	528	650	1.3k	5.4k	21.8 k	2			4				18-Sep-22	Gladys C.
Librería - Librero A	*	452	631	1k	4.3k	17.4 k	1			16					Gladys C.
Librería - Librero B	*	452	631	1k	4.3k	17.4 k	1			10					Gladys C.
Librería - Mesa	*	240	400	560	2.2k	8.9k	1			4				17-Sep-22	Gladys C.
Librería - Silla	*	412	650	848	3.2k	13.4 k	1			12				17-Sep-22	Gladys C.
Librería - 3 Pergaminos	*	450	864	2.8k	11.5 k	46.2 k	1			*				17-Sep-22	Gladys C.
Librería - 9 Libros A	*	312	464	1k	4.2k	16.8 k	1			*				18-Sep-22	Gladys C.
Librería - 9 Libros B	*	312	464	1k	4.2k	16.8 k	1			*				18-Sep-22	Gladys C.
Librería - 9 Libros C	*	312	464	1k	4.2k	16.8 k	1			*				18-Sep-22	Gladys C.
Librería - 9 Libros D	*	312	464	1k	4.2k	16.8 k	1			*				18-Sep-22	Gladys C.
Librería - Gema	*	8	24	48	192	768	1			*				17-Sep-22	Gladys C.

Referencias:

3D – Desarrollo del modelo básico
OPT – Desarrollo del modelo Optimo
LOD1 – Modelo con Nivel de distancia 1
LOD2 – Modelo con Nivel de distancia 2
High-M - Modelo No Optimizado SubSurf 1
High - Modelo No Optimizado SubSurf 2
Texturas - Cantidad de texturas
Rig? - Si el modelo usa articulación
Seamless? - Si el modelo texturas fluidas repetibles
Reutilizado – Si se usa más de una vez
Glass - Si lleva texturas de vidrio / opacidad
Anim - Si lleva alguna animación
Text Comp – Si tiene texturas compartidas y con quien
Fecha Fin - Fecha cuando se finalizó el modelo
Hecho Por - Por quien fueron elaborados los modelos

Verde = Finalizado

Amarillo = En progreso

Rojo = No iniciado

4.3 Fase 3: Pruebas, Análisis, y Producción

En esta sección se enfoca principalmente el desarrollo de la app PolyCastle en Unity, el diseño y armado del modelo completo, interfaz, las pruebas de uso, optimización y flujo.

Video: https://www.youtube.com/watch?v=WIPiPcX35mU

El proyecto de Unity fue iniciado el 15 de septiembre.

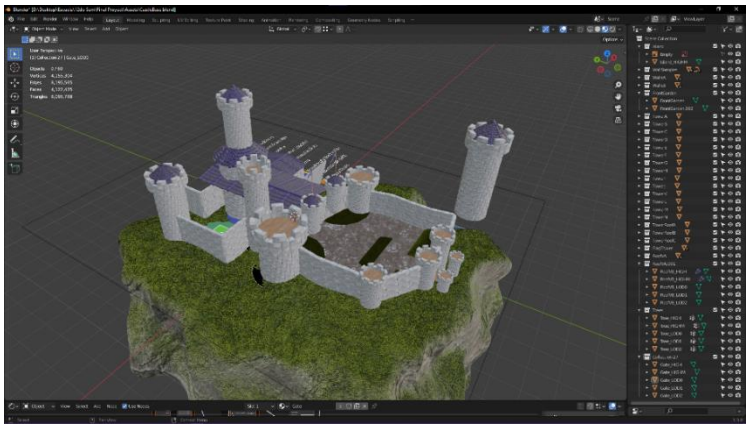


Figura 43. Castillo armado en Blender para exportar a Unity (en progreso)

Para lograr las posiciones de cada objeto dentro de Unity en el lugar apropiado, se deberá juntar los tres grupos de modelos y colocar en sus posiciones correctas dentro de Blender, por lo tanto, se debe armar el modelo en Blender y exportar las partes en sus posiciones tomando el punto de referencia como el centro del mundo.

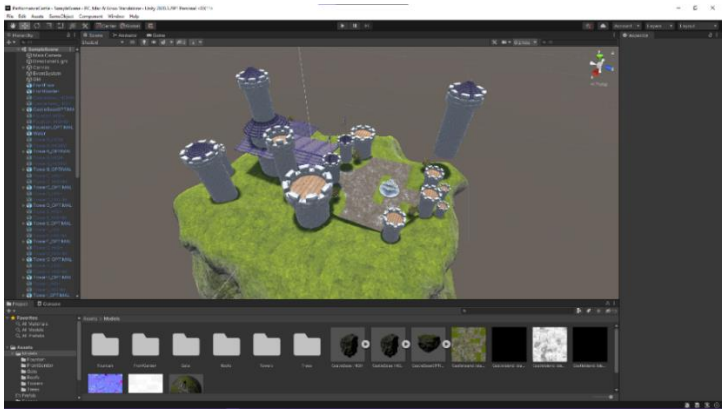
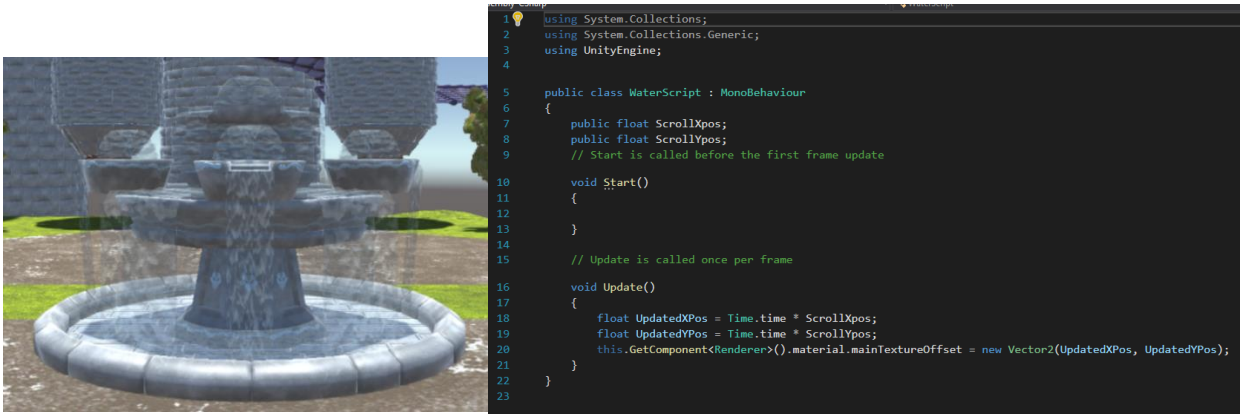


Figura 44. Castillo armado en Unity (en progreso)

Una vez que el objeto está en su posición, se pueden exportar los 3 grupos de calidades de los modelos de Blender e importar directamente a Unity, el cual respetara su posición. Considerando el flujo de diseño modelos tridimensionales, una vez que los modelos se encuentran dentro de Unity, se deberá crear por grupo su material y se importaran las texturas, donde se asignaran apropiadamente.

En casos relevantes, como la animación de texturas, se implementó el siguiente código, que permite que la textura de agua tenga movimiento en base al paso del tiempo.



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;


public class WaterScript : MonoBehaviour
{
    public float ScrollXpos;
    public float ScrollYpos;
    // Start is called before the first frame update

    void Start()
    {

    }

    // Update is called once per frame

    void Update()
    {
        float UpdatedXPos = Time.time * ScrollXpos;
        float UpdatedYPos = Time.time * ScrollYpos;
        this.GetComponent<Renderer>().material.mainTextureOffset = new Vector2(UpdatedXPos, UpdatedYPos);
    }
}

```

Figura 45. Muestra del código y la fuente animada

Para lograr visualizar la información del dispositivo en el cual se corra la aplicación, se implementó el siguiente código, que logra tomar la información del sistema, incluyendo fluctuaciones por fotograma, y colocarlas en variables que puedan ser mostradas dentro de la aplicación en todo momento.

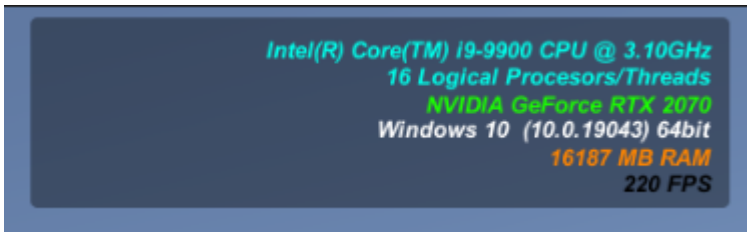
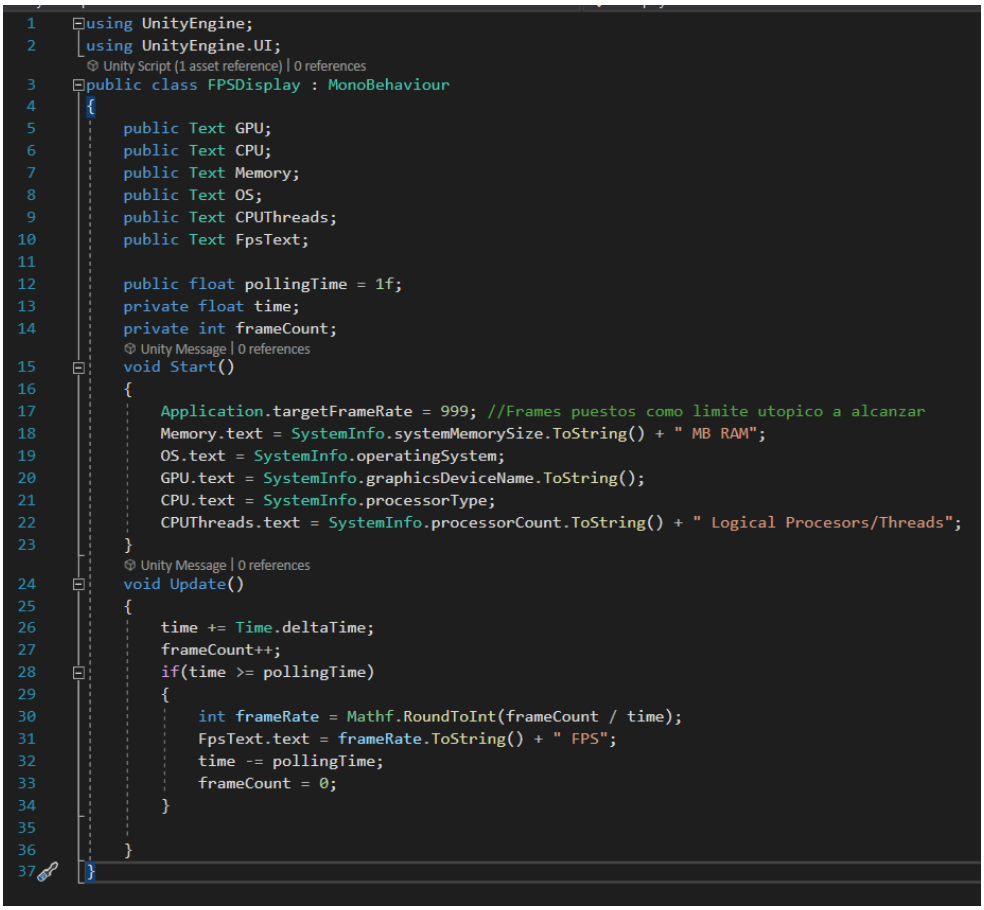


Figura 46. Cuadro de datos en dispositivo mostrado dentro de Unity (en progreso)



```
using UnityEngine;
using UnityEngine.UI;
Unity Script (1 asset reference) | 0 references
public class FPSDisplay : MonoBehaviour
{
    public Text GPU;
    public Text CPU;
    public Text Memory;
    public Text OS;
    public Text CPUThreads;
    public Text FpsText;

    public float pollingTime = 1f;
    private float time;
    private int frameCount;
    Unity Message | 0 references
    void Start()
    {
        Application.targetFrameRate = 999; //Frames puestos como limite utopico a alcanzar
        Memory.text = SystemInfo.systemMemorySize.ToString() + " MB RAM";
        OS.text = SystemInfo.operatingSystem;
        GPU.text = SystemInfo.graphicsDeviceName.ToString();
        CPU.text = SystemInfo.processorType;
        CPUThreads.text = SystemInfo.processorCount.ToString() + " Logical Procesors/Threads";
    }
    Unity Message | 0 references
    void Update()
    {
        time += Time.deltaTime;
        frameCount++;
        if(time >= pollingTime)
        {
            int frameRate = Mathf.RoundToInt(frameCount / time);
            FpsText.text = frameRate.ToString() + " FPS";
            time -= pollingTime;
            frameCount = 0;
        }

    }
}
```

Figura 47. Código de la implementación del cuadro de datos del dispositivo

Desafortunadamente, por limitación de Unity, revisar las temperaturas de GPU y CPU no es posible.

Para mayor funcionalidad y facilidad de revisar los datos, se implementó un depurador que se guarda la información vital del dispositivo (CPU, GPU, OS, RAM, tamaño de pantalla), y la cantidad de cuadros por segundos procesados junto con las capturas con la información.

```
//Para tomar capturas de pantalla
0 references
private IEnumerator ScreenshotLog()
{
    //Capture
    yield return new WaitForEndOfFrame();
    Texture2D textureSS = new Texture2D(Screen.width, Screen.height, TextureFormat.RGB24, false);

    textureSS.ReadPixels(new Rect(0, 0, Screen.width, Screen.height), 0, 0);
    textureSS.Apply();

    string namecap = "SS_PolyCastle_" + capNumber + "_" + qualityTypeText + "_" + System.DateTime.Now.ToString("yyyy-MM-dd__HH-mm-ss") + ".png";

    byte[] bytes = textureSS.EncodeToPNG();
    File.WriteAllBytes("Z-LogNCaps/" + namecap, bytes);

    //Log
    string txtDocumentName = "Z-LogNCaps/Log.txt";
    if (!File.Exists(txtDocumentName))
    {
        //agregando el log
        File.WriteAllText(txtDocumentName,
            "Informacion General y Log \n\n" +
            "GPU: " + SystemInfo.graphicsDeviceName.ToString() + " " + gpuMemSize.text + "\n" +
            "CPU: " + SystemInfo.processorType + ", LPT :" + SystemInfo.processorCount.ToString() + "\n" +
            screenSize.text + "\n" +
            "OS: " + SystemInfo.operatingSystem + "\n" +
            "RAM: " + SystemInfo.systemMemorySize.ToString() + " MB" + "\n" +
            "============================================================================================== \n\n"
            );
    }

    //agregando mas cosas al log:
    File.AppendAllText(txtDocumentName,
        "No. De Captura: " + capNumber + "\n" +
        "Fecha: " + System.DateTime.Now.ToString("yyyy-MM-dd__HH-mm-ss") + "\n" +
        "Calidad: " + qualityTypeText + " \n" +
        "FPS: " + framesInfo.text + "\n" +
        "---------------------------------------------------------------------------------------------- \n"
        );

    capNumber++;
}
```

Figura 48. Código de la implementación del guardado del depurador + Capturas

Se colocaron todas las pantallas de UI y diseño, como logotipos, botones, información, tipografías, etc.

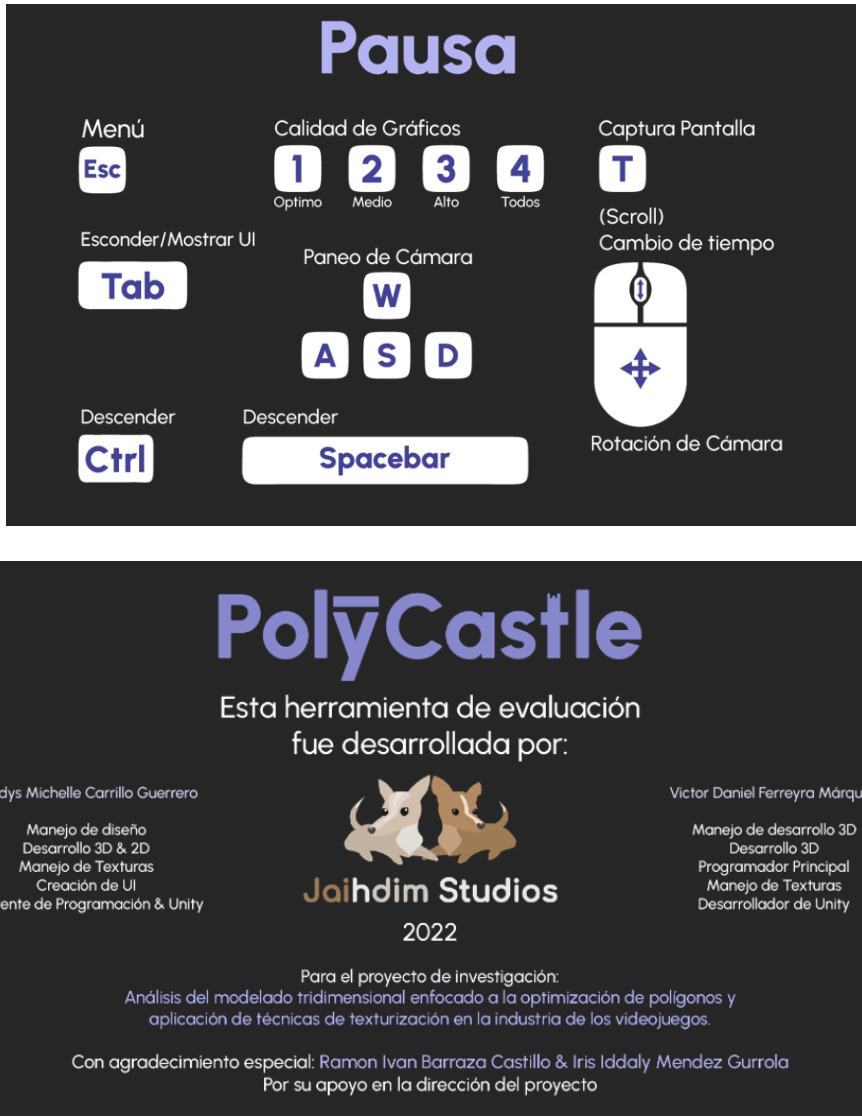


Figura 49. Muestra de UI

Se asigno una cámara de movimiento libre para que el usuario explore a libertad el mundo, y cinco cámaras fijas para poder elaborar las pruebas de evaluación y rendimiento de manera constante y sin fluctuaciones.

Figura 50. Muestra de cámara fija

Una de las dificultades principales fue el tratar de asignar los distintos modos de calidad de modelos dentro de la aplicación, aun si se elaboraron todos los elementos enlistados, se tomó la decisión final de aplicar una cantidad menor de modelos en los modos no óptimos, ya que podía causar inestabilidad y cierres inesperados. Aun así, el rendimiento sigue siendo menor al activar los modos óptimos por la cantidad de polígonos empleados aun si visualmente da la ilusión de parecer menor.

Figura 51. Diferencias entre Media-alta y Óptimo

La aplicación funcional con todos los errores solucionados (versión Alpha 0.1.35) fue finalizada el día 24 de octubre.

4.1 Fase F1: Pre-Lanzamiento, Encuestas Y Búsqueda de Resultados

Una vez finalizado el Alpha de la aplicación, se podrán asignar las pruebas de rendimiento y encuestas al objetivo meta. Para lograr obtener resultados relevantes para la investigación, el enfoque principal de las encuestas radican en observar si la muestra compete conocimientos sobre las distintas técnicas de optimización, sus conocimientos en texturización, y su capacidad visual para lograr comparar objetos tridimensionales de distintas calidades poligonales, mientras que las pruebas de rendimiento mostraran las consecuencias de uso de altas cantidades de polígonos y malas técnicas de optimización, y como la falta de optimización puede afectar en relación al rendimiento a los distintos dispositivos.

4.1.1 La encuesta

Liga de la encuesta (Vía Google Forms): https://forms.gle/NJJZpQiu2tTfC8c59

Dentro de esta sección se encuentran todas las preguntas empleadas dentro de la encuesta, las cuales fueron aplicadas a la muestra asignada.

Preguntas básicas de conocimiento:

Se espera lograr conocer en qué nivel de experiencia está el usuario, si reconoce los aspectos que se encuestaran para confirmar si es parte de la muestra esperada.

¿Alguna vez ha jugado videojuegos con gráficos 3D?

Si / No

¿Tiene conocimiento de que es un modelo 3D y de que está compuesto?

Si / No

¿Qué es un polígono?

Respuesta abierta

¿Qué es un tris (triángulo)?

Respuesta abierta

Preguntas de afirmación:

Esta serie de cuestionamientos nos permitirán saber si el usuario tiene noción de los aspectos vitales dentro del desarrollo de una aplicación o videojuego con modelos tridimensionales, porque son importantes, y si están informados de ello.

¿Tiene conocimiento de que es el rendimiento en una computadora, consola o dispositivo móvil? - En caso de que la respuesta anterior haya sido si, por favor de una descripción de que es.

Si / No + Respuesta abierta

¿Es importante emplear muchos polígonos en un modelo? ¿Por qué?

Si / No + Respuesta abierta

¿En el modelado, es importante usar Subdivisión Surface? ¿Por qué?

Si / No + Respuesta abierta

¿Se puede armar un buen modelo sin Subdivisión Surface?

Si / No

Cuando se hace un modelo tridimensional, entre más polígonos…

El archivo es más pesado, detallado y complejo, la computadora tiene menor rendimiento

El archivo es más ligero, limpio y simple, la computadora tiene mejor rendimiento

Es indiferente.

Cuando se desarrolla un videojuego, entre más polígonos tenga un modelo tridimensional…

El archivo es más pesado, detallado y complejo, la computadora tiene menor rendimiento

El archivo es más ligero, limpio y simple, la computadora tiene mejor rendimiento

Es indiferente.

¿Es importante implementar texturas en un modelo tridimensional? ¿Por qué?

Si / No + Respuesta abierta

¿Sabes usar o elaborar texturas y mapas de texturas?

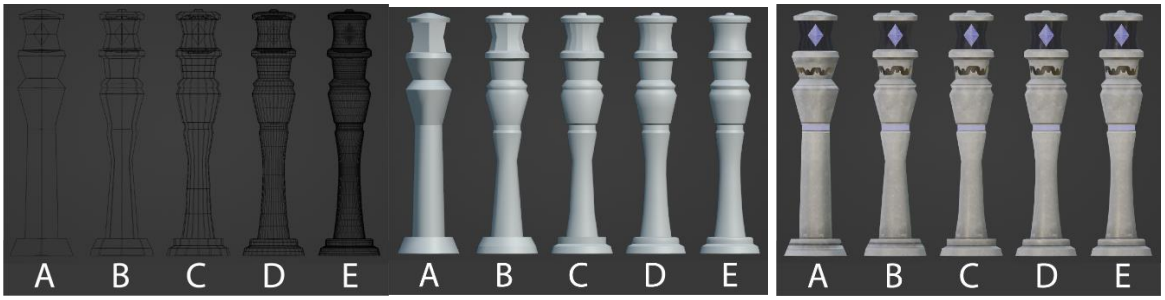
Si / No

Preguntas de comparación y reconocimiento:

Las siguientes preguntas contienen imágenes para observar y comparar, son tres preguntas con tres distintas imágenes en las cuales se debe contestar si o no. Se espera que esta serie de preguntas el usuario pueda observar o no los cambios entre modelos.
La primera imagen muestra de manera clara la diferencia entre cada uno de los modelos, al poder observar la malla directamente.
La segunda imagen muestra los objetos de manera sólida, donde puedes observar los detalles visualmente sin distracciones, se espera que los modelos puedan ser diferenciados entre ellos a pesar de no lograr ver sus respectivas mallas.
La tercera imagen muestra los modelos de manera texturizada y detallada, en este estilo, se espera que los usuarios tengan complicaciones en lograr observar las diferencias entre cada uno de ellos.

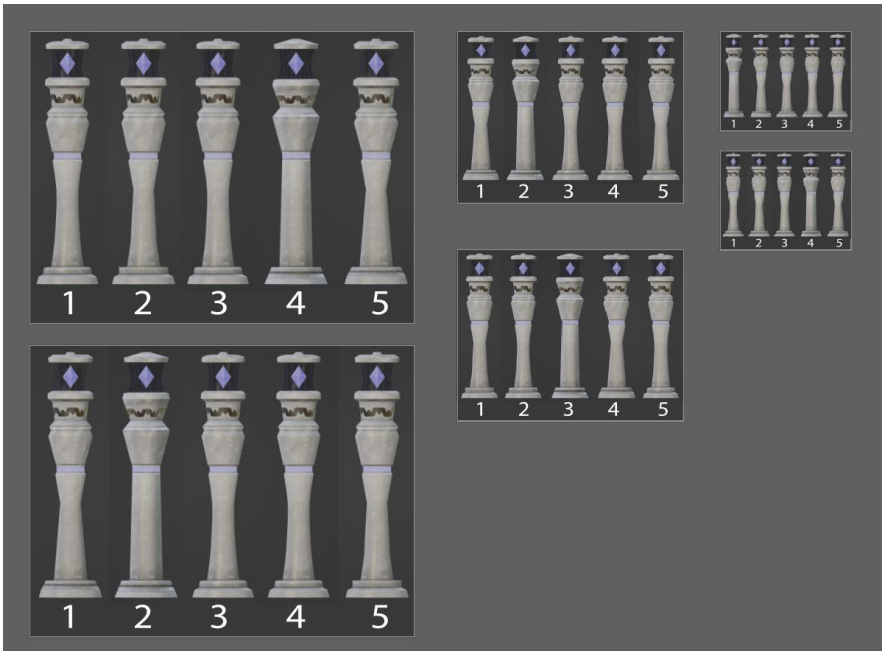
A – LOD 2 B – LOD 1 C – Óptimo D – Media-Alta E – Alta

¿Puedes notar la diferencia entre TODOS los objetos tridimensionales?



Las siguientes seis preguntas contienen imágenes para observar, en donde se emplea la tercera fotografía y se cambia el orden de los modelos para que el usuario trate de averiguar el orden real de calidad de imágenes con mayor cantidad de polígonos,

esta serie de imágenes tendrá distintos tamaños, para observar los distintos niveles de detalle, su utilidad y si es posible ver las diferencias entre ellos a distintas distancias.



4.1.2 La aplicación (PolyCastle)

Liga de la aplicación:
https://mega.nz/file/jAcyhKIS#UfxfG_1o7utlt6gBp697RLKHSXpisBW80nH471jIok

Para tener resultados sin variaciones inesperadas, se aplicarán por dispositivo cuatro distintas escenas (o cámaras) fijas en las cuales se tomarán cuatro capturas por escena, una por cada modo de calidad, a partir de estos resultados se observarán las variaciones y se segregarán los dispositivos por su potencia de hardware en GPU principalmente, usando de referencia el sitio web Passmark Software.

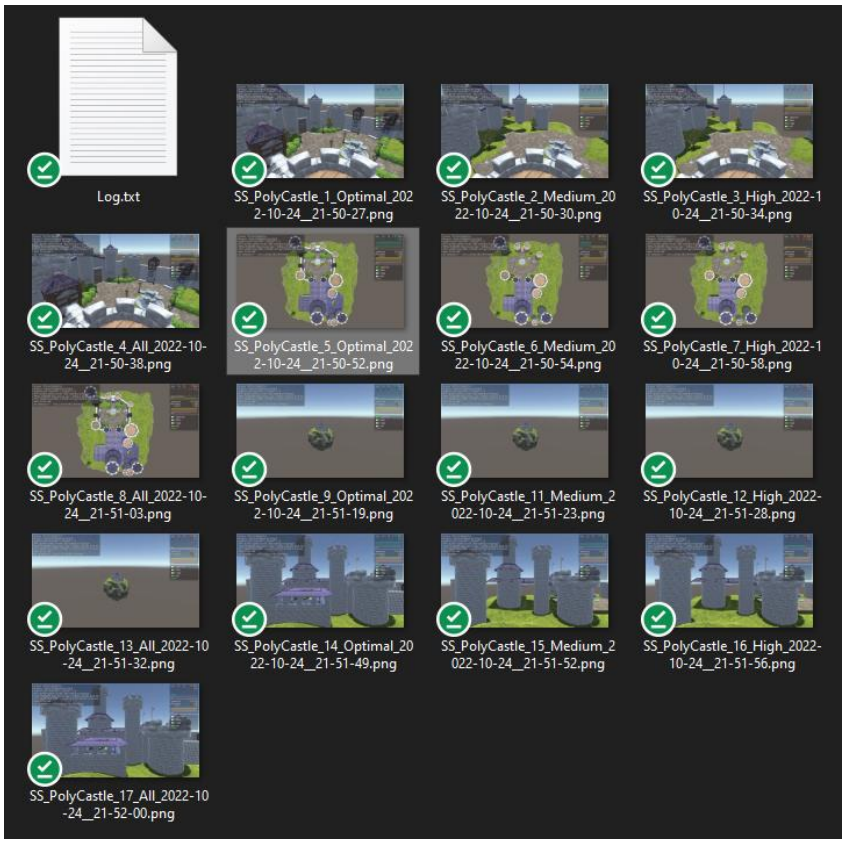


Figura 52. Imágenes de las cuatro distintas escenas más los resultados trasladados a un archivo de texto

Escena 1: Patio Frontal

En esta vista se muestra el patio frontal, donde visualmente se observan distintos repetidos objetos a distintas distancias, en el aspecto óptimo, se espera activar los LOD's en los objetos a mayor distancia sin que el usuario pueda notarlo.

Figura 53. Modo o Escena 1, patio frontal del castillo

Escena 2: Vista Aérea

En esta vista se pueden observar la mayoría de los objetos creados, está enfocado a demostrar la perdida y aumento claro de FPS en los distintos modos de calidades por el ángulo en el que se captura la toma.

Figura 54. Modo o Escena 2, vista aérea del castillo

Escena 3: Vista a distancia

En esta vista se puede observar la isla flotante con el castillo, no todos los objetos están a la vista del usuario, aun así, dentro de esta vista es posible lograr observar las fluctuaciones de FPS entre los distintos modos de calidades sin importar la distancia.

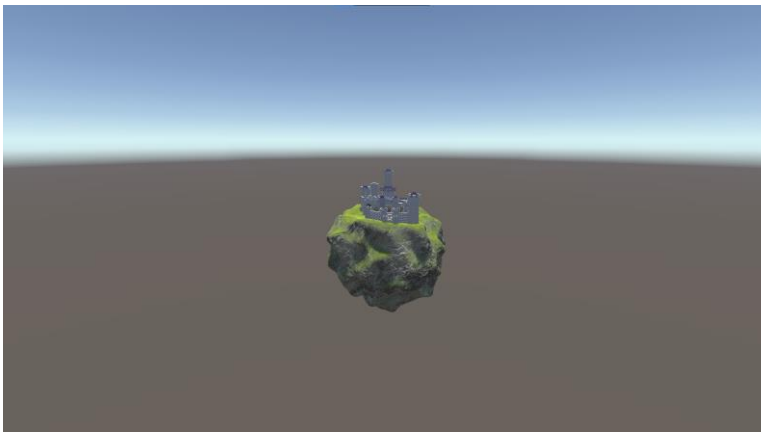


Figura 55. Modo o Escena 3, vista a distancia del castillo

Escena 4: Patio Trasero

Por último, en esta vista se pueden observar mayoritariamente las torres, los modelos tridimensionales más pesados de la escena, y el patio trasero del castillo, a pesar de no poder observar todos los objetos, el enfoque a las torres da una comparativa interesante de FPS en los distintos modos de calidad.

Figura 56. Modo o Escena 4, patio trasero del castillo

Modos de calidad asignadas (Con la referencia de imagen):

Modo 1: Óptimo (Con LOD's)

Modo 2: Sub-Óptimo (Aprox. 4 veces más polígonos comparados al óptimo)

Modo 3: No Óptimo (Aprox. 4 veces más polígonos comparados al Sub- óptimo)

Modo 4: Todos los anteriores activados.

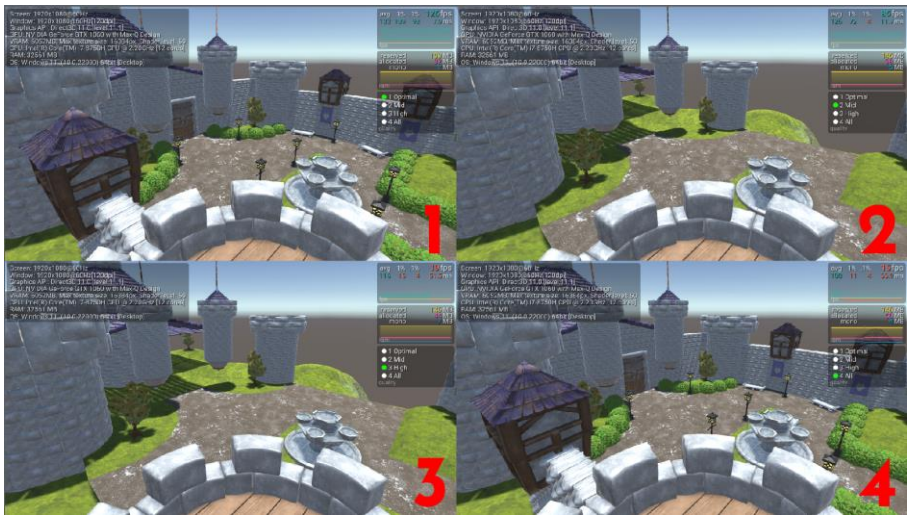


Figura 57. Escena con diferentes modos de calidad

CAPÍTULO 5: RESULTADOS

INTRODUCCIÓN

Dentro de este capítulo se demostrarán los resultados de tanto las encuestas como las estadísticas generadas por las pruebas de uso en la muestra y dispositivos, para lograr comprender el estado esperado, y por lo tanto eventualmente obtener una conclusión sobre lo planteado dentro de la investigación, dentro de cada sección se explicará con detalle el razonamiento de las preguntas y los resultados obtenidos.

5.1 Fase Fin 2: Resultados

Como nota, la mayoría de los usuarios que fueron encuestados fueron participes de las pruebas de rendimiento en sus dispositivos, sin embargo, existen usuarios sin dispositivos de prueba (laptops, computadoras personales), por lo que no todos los encuestados fueron participes de las pruebas de rendimiento.

En la encuesta, se encuestaron 33 participantes, lo esperado de la muestra.

En las pruebas de rendimiento, se probaron 22 dispositivos en total.

5.1.1 Resultados de la encuesta

Las preguntas iniciales fueron asignadas para comprender si los usuarios encuestados tenían comprensión de los conceptos básicos dentro del modelado tridimensional, también, para conocer si tenían conocimiento básico de aspectos dentro de los videojuegos.

Los resultados fueron favorables, la mayoría de los encuestados tenían el conocimiento necesario para lograr llevar a cabo la encuesta sin problemas. El 93.1% indicaron haber jugado un videojuego con gráficos 3D en algún momento de su vida, mientras que solo un 6.9% indico lo contrario.

En las preguntas relacionadas con el aspecto empírico de la investigación, se puede denotar que a pesar de que los involucrados en la encuesta tenían relación con las materias de desarrollo, o interés por aplicar sus conocimientos en ello, el 13.8% de los encuestados no tenían experiencia con el concepto de rendimiento, otras respuestas descritas no tenían el concepto muy claro.

Si bien la gran mayoría respondió conocer las ventajas y desventajas de los polígonos en el modelado, en la pregunta complementaria, un porcentaje bastante alto de encuestados usan el modificador como un método de suavizado, y no conocen otras técnicas para lograr el mismo resultado.

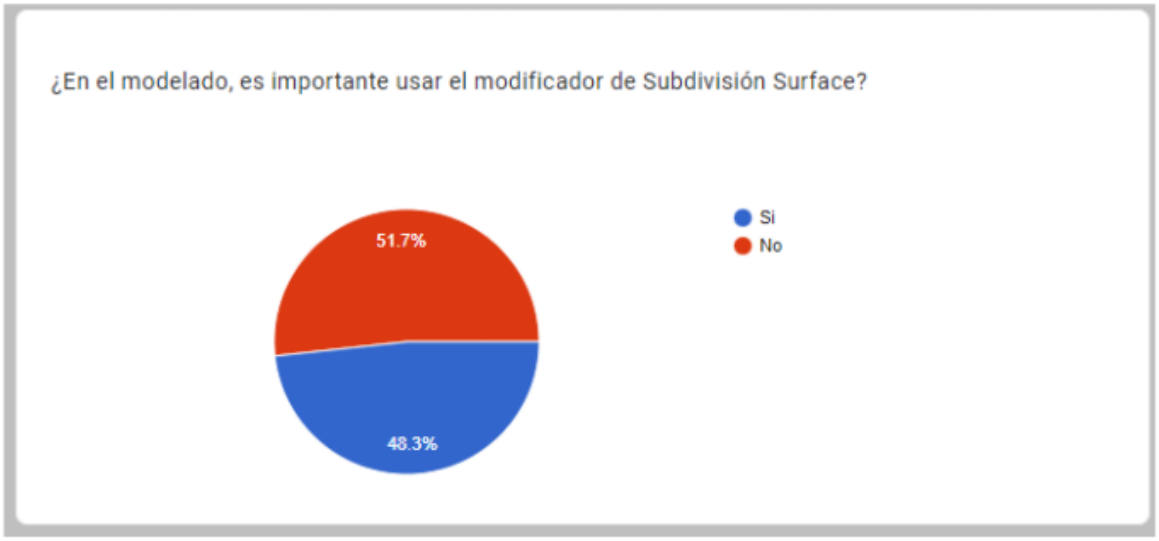


Figura 58. Resultados a la pregunta 7

A pesar de esto, la mayoría reconoce que se pueden elaborar buenos modelos sin la necesidad de utilizar el modificador de subdivisión, también, la importancia del peso poligonal de un objeto tridimensional con relación al archivo y el rendimiento del programa al momento de ejecución.

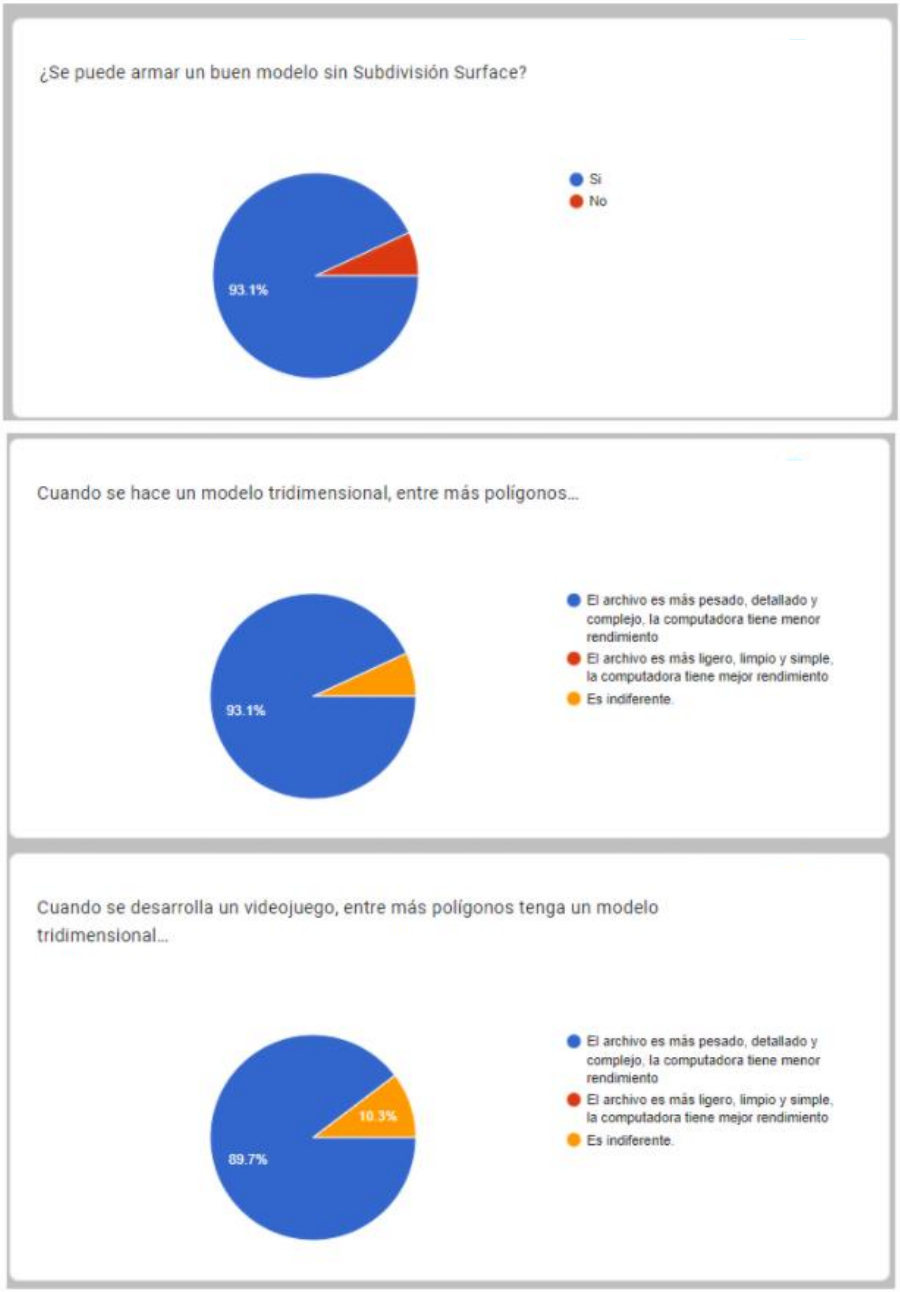


Figura 59. Resultados a las preguntas 8, 9 y 10

Este resultado es uno de los más relevantes dentro de la encuesta, a pesar de visualizar el mismo objeto en las tres capturas, siendo su única diferencia la falta de texturas o malla, al observar el último de ellos (el objeto texturizado), se nota claramente una diferencia de reconocimiento de los modelos, visualmente no le es tan posible al usuario notar las diferencias entre cada uno de ellos, como factor importante se debe aclarar que dos de los objetos (A y B) en esta pregunta están en desventaja, ya que se espera que sean observados de distancias mayores, las cuales hacen parecerse más al modelo óptimo.

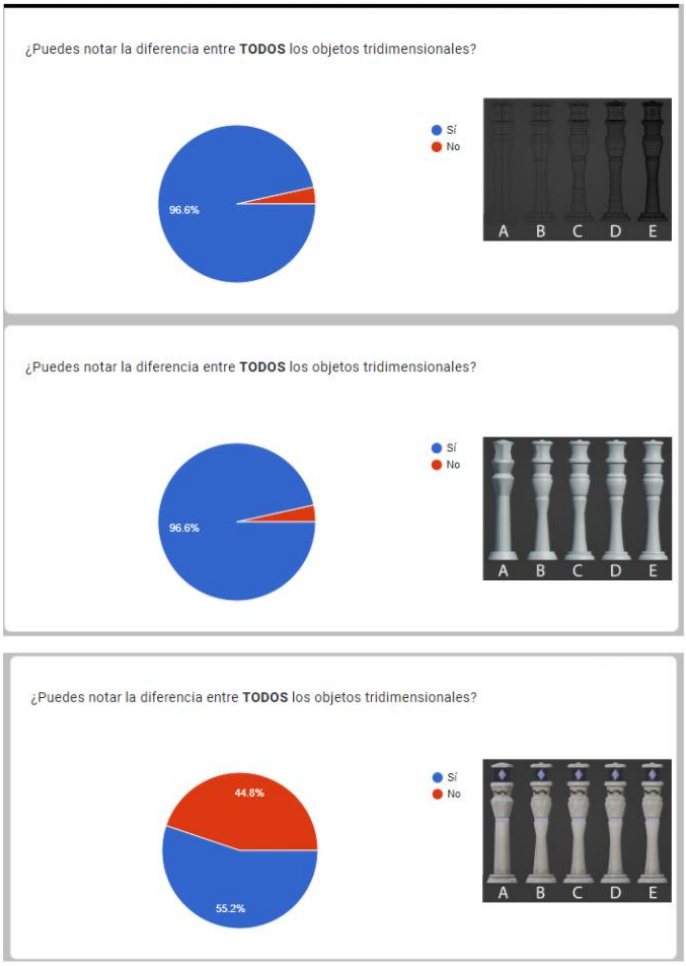


Figura 60. Comparativa de dificultad para identificar los modelos

Para lograr confirmar la teoría de dificultad de reconocimiento de modelos texturizados, se aplicaron las siguientes preguntas:

Serie de modelos en desorden: Imagen al 100% de distancia

Objetos y respuestas: Porcentajes de certeza:

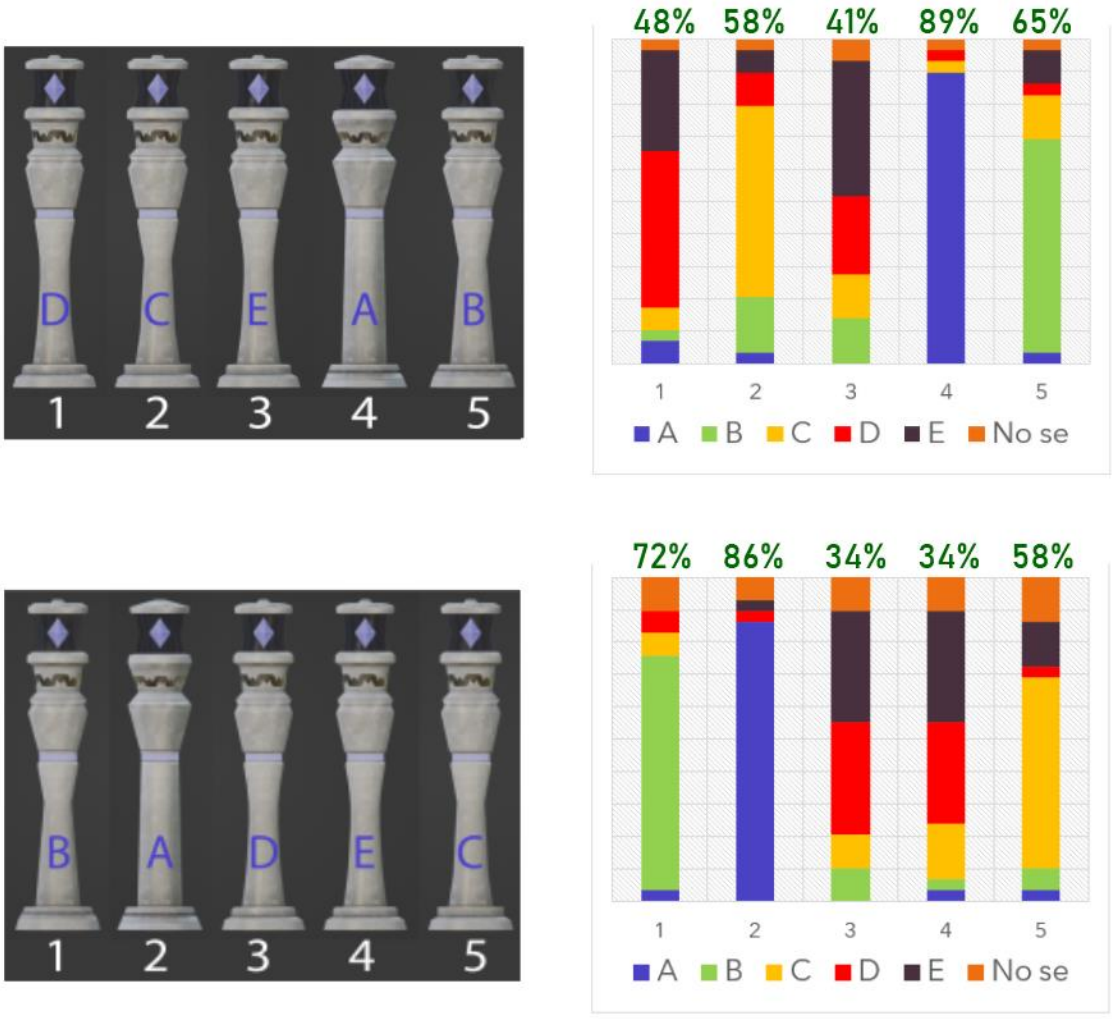


Figura 61. Resultados de identificación de modelos a 100% de distancia

A la distancia asignada, la mayoría de los encuestados se les facilita reconocer los modelos de menores polígonos, el modelo óptimo es un poco confuso, y los objetos de mayores polígonos son confundidos entre sí.

Al 75% de la distancia original, la mayoría de los encuestados aún pueden reconocer con facilidad los primeros dos objetos, hay una pequeña confusión con el óptimo modelo LOD 2, y Media-Alta. Por último, existe una confusión constante para reconocer el modelo de Media-Alta y Alta, los encuestados los comparan entre sí, les es complicado diferenciarlos.

Serie de modelos en desorden: Imagen al 75% de distancia

Objetos y respuestas: Porcentajes de certeza:

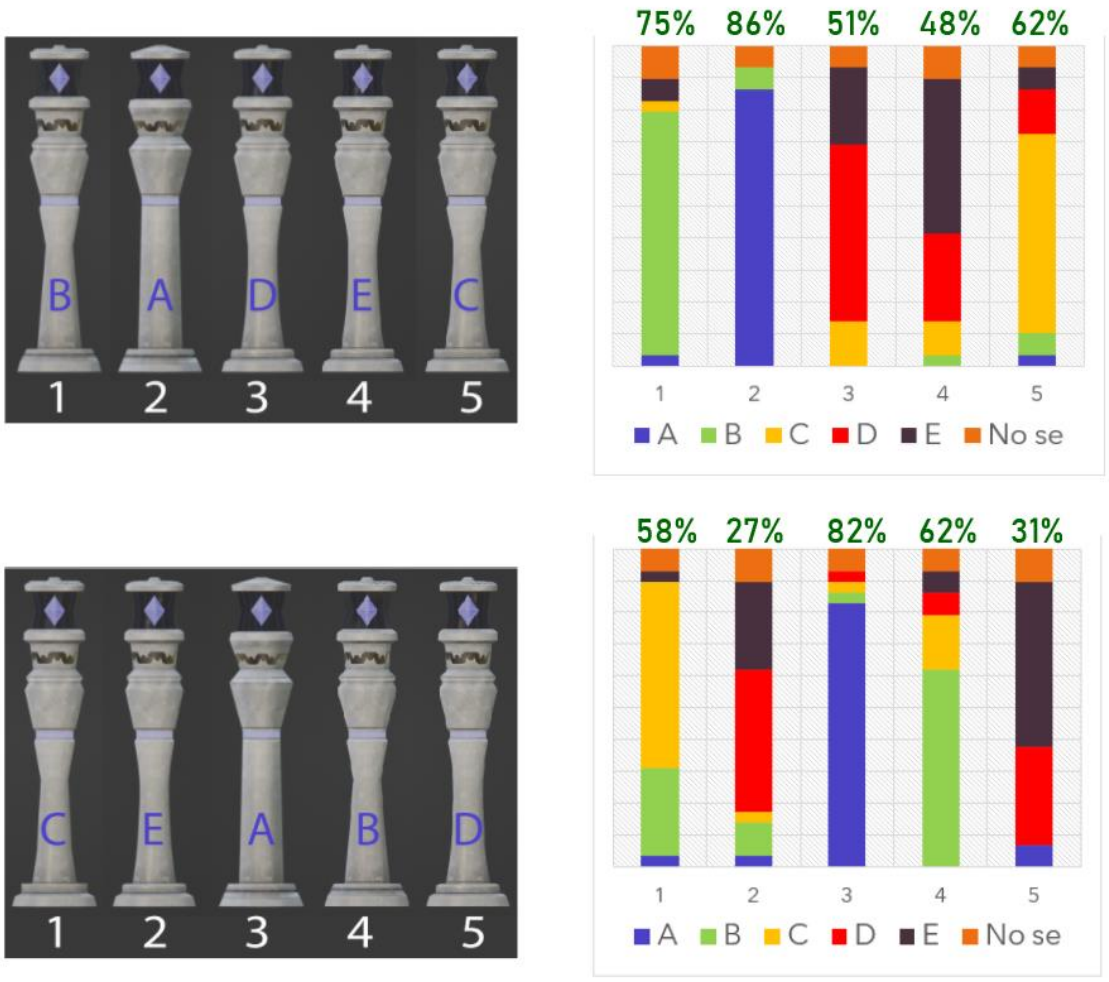


Figura 62. Resultados de identificación de modelos a 75% de distancia

Al 45% de la distancia original, es complicado diferenciar los objetos de mayor cantidad poligonal, la tendencia de reconocer los modelos de menor calidad no fluctúa a según las comparativas anteriores, más de la mitad de los encuestados lograron reconocer el modelo óptimo, sin embargo, a la mayoría de los usuarios les fue más difícil lograr diferenciar entre ambos modelos, constantemente confundiéndolos entre sí.

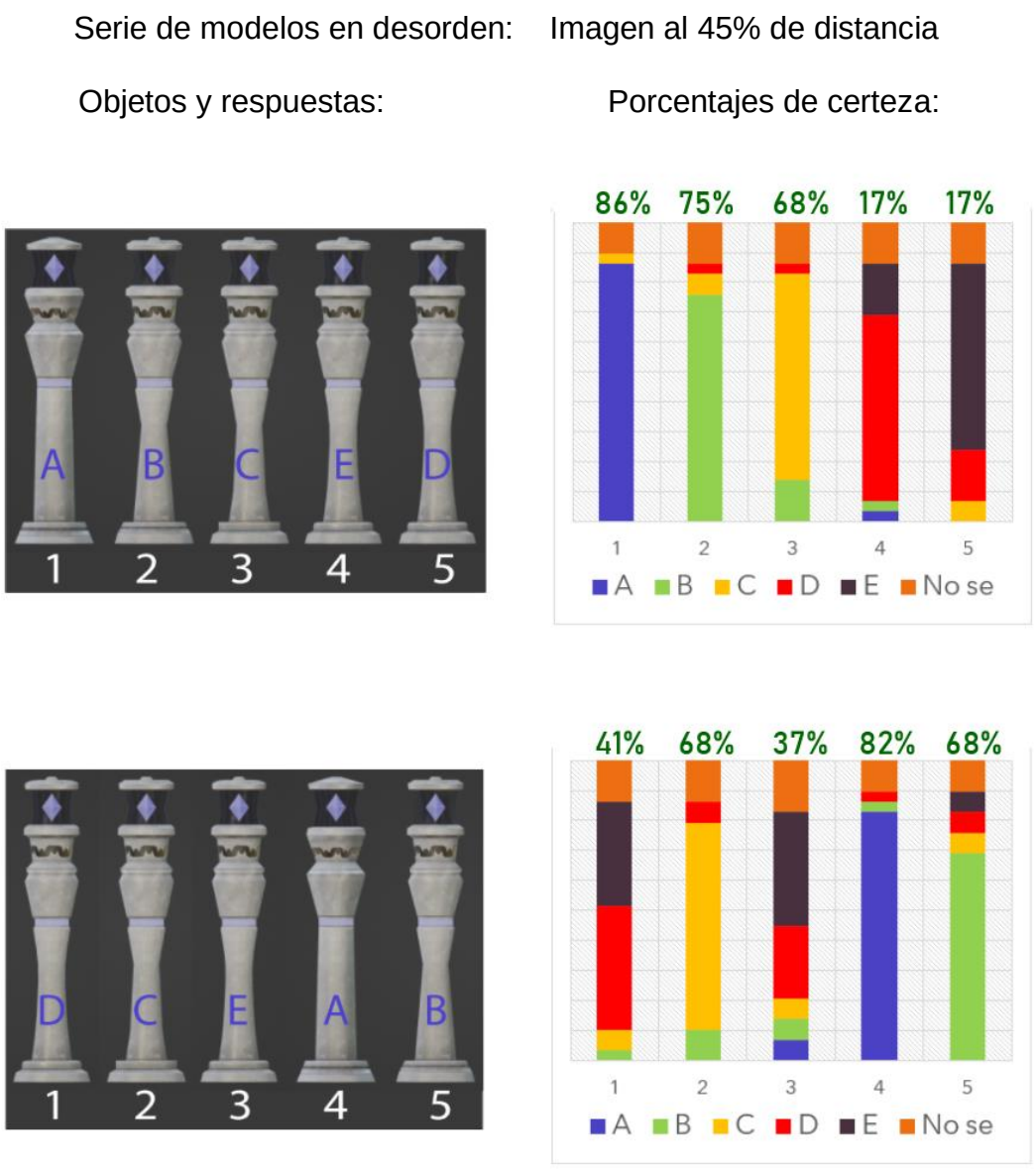


Figura 63. Resultados de identificación de modelos a 45% de distancia

5.1.2 Resultados de rendimiento: FPS

A continuación, se enlistan los resultados de la ejecución de la aplicación configuraciones de equipos de cómputo.

Tabla 2. asignación de rangos según su puntuación

Serie	CPU	Punt. CPU	GPU	Punt. GPU	Rango
A-1	AMD Ryzen 9 3900X	**32756**	AMD Radeon RX 6800	**21248**	Alta
A-2	AMD Ryzen 7 3700X	**22698**	AMD Radeon RX 6700 XT	**19286**	Alta
A-3	Intel(R) Core(TM) i9-10885H	**15505**	NVIDIA GeForce RTX 2070 Super	**18197**	Alta
A-4	Intel(R) Core(TM) i7-10750H	**12196**	NVIDIA GeForce RTX 2070 Super	**18197**	Alta
MA-1	Intel(R) Core(TM) i7-9750H	**11117**	NVIDIA GeForce RTX 2070 with Max-Q Design	**11929**	Media Alta
MA-2	AMD Ryzen 5 5600G	**19842**	NVIDIA GeForce GTX 1660 Ti	**11884**	Media Alta
MA-3	Intel(R) Core(TM) i7-7700K	**9668**	NVIDIA GeForce GTX 1060 (PC)	**10067**	Media Alta
MA-4	AMD Ryzen 9 4900HS	**19318**	NVIDIA GeForce RTX 2060 with Max-Q Design	**9688**	Media Alta
M-1	Intel(R) Core(TM) i7-6700HQ	**6531**	NVIDIA GeForce GTX 1060 (Mobile)	**8160**	Media
M-2	Intel(R) Core(TM) i7-8750H	**10028**	NVIDIA GeForce GTX 1060 with Max-Q Design	**7923**	Media
M-3	AMD Ryzen 5 4600H	**14676**	NVIDIA GeForce GTX 1650 (Mobile)	**6968**	Media
M-4	Intel(R) Core(TM) i7-6700HQ	**6531**	NVIDIA GeForce GTX 970M	**5741**	Media
M-5	Intel(R) Core(TM) i5-7300HQ	**5097**	NVIDIA GeForce GTX 1050 (Mobile)	**4461**	Media
B-1	AMD Ryzen 5 Microsoft Surface (R) Edition	**7358**	AMD Radeon(TM) Graphics	**2100**	Baja
B-2	AMD Ryzen 3 3250U	**3917**	AMD Radeon(TM) Graphics	**2100**	Baja
B-3	AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx	**7058**	AMD Radeon(TM) Vega 8 Graphics	**1588**	Baja
B-4	Intel(R) Pentium(R) CPU 6405U	**2374**	Intel(R) UHD Graphics	**1513**	Baja
B-5	AMD Ryzen 7 3700U with Radeon Vega Mobile Gfx	**7228**	AMD Radeon(TM) RX Vega 10 Graphics	**1465**	Baja
B-6	Intel(R) Core(TM) i7-8650U	**6309**	Intel(R) UHD Graphics 620	**1034**	Baja
B-7	Intel(R) Core(TM) i7-4790	**7231**	AMD Radeon(TM) R5 240	**523**	Baja
B-8	Intel(R) Core(TM) i5-3380M	**2935**	Intel(R) HD Graphics 4000	**339**	Baja
B-9	Intel(R) Core(TM) i5-3340M	**2656**	Intel(R) HD Graphics 4000	**339**	Baja

Escena 1: Patio Frontal

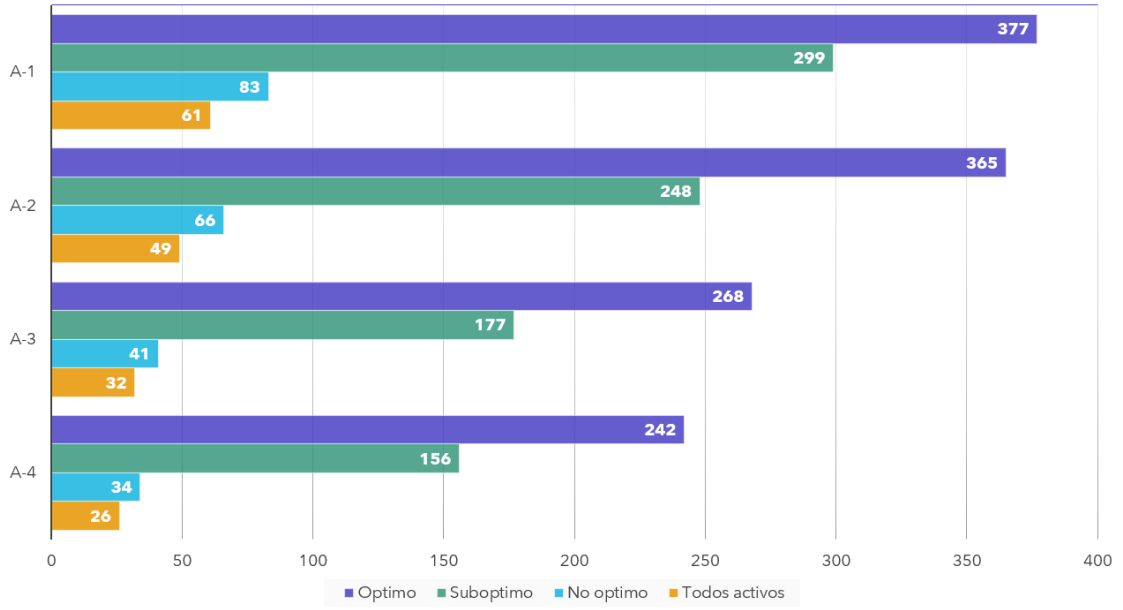
Mayor número de FPS significa mejor rendimiento en el dispositivo, el límite máximo de FPS es de 999.

En esta toma se pueden observar distintos modelos iguales a distintas distancias, gracias a esto se pueden visualizar distintos tipos de LOD en el modo óptimo. Observando los resultados, del modo óptimo al modo sub- óptimo, se puede notar una pérdida de rendimiento de al menos el 20% - 40% en la mayoría de las gamas, mientras que en el modo sub-óptimo al no optimo se nota una pérdida de más del 70% del rendimiento, llegando así a un límite de FPS menor de lo considerado mínimo jugable en la industria (30 FPS) en el 95% de los dispositivos diagnosticados.

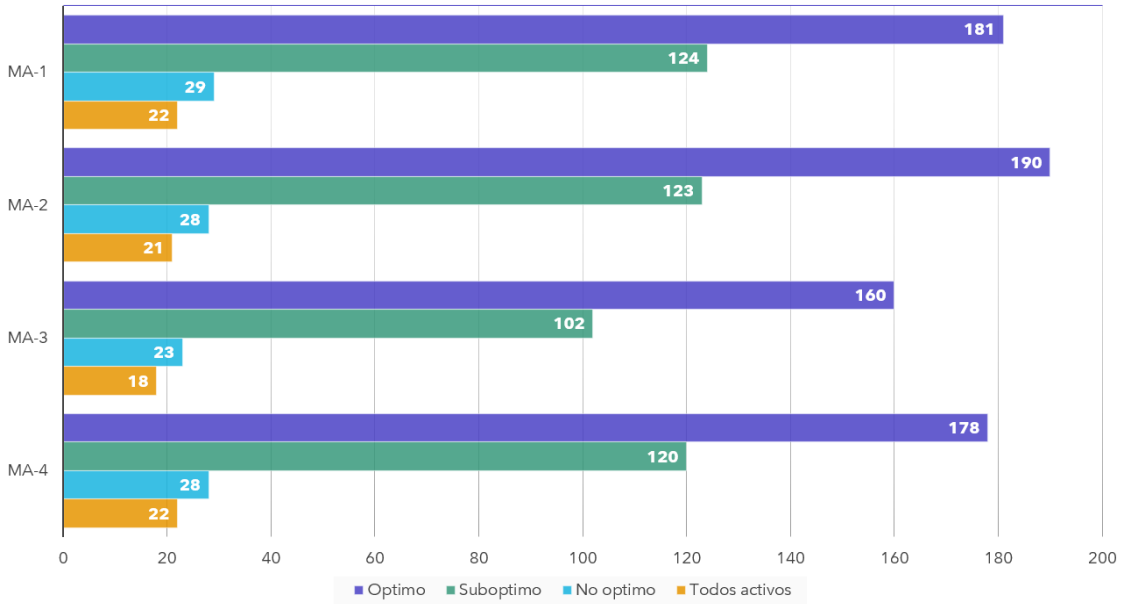
Solo en los dispositivos de gama más baja, aquellos dispositivos con tarjetas gráficas integradas (sin tarjeta gráfica) solamente o elaboradas hace más de 10 años el modo óptimo no es suficiente para lograr el rendimiento mínimo jugable.

Figura 64. Escena 1 - Patio frontal

Gama Alta – Escena 1:



Gama Media-Alta – Escena 1:



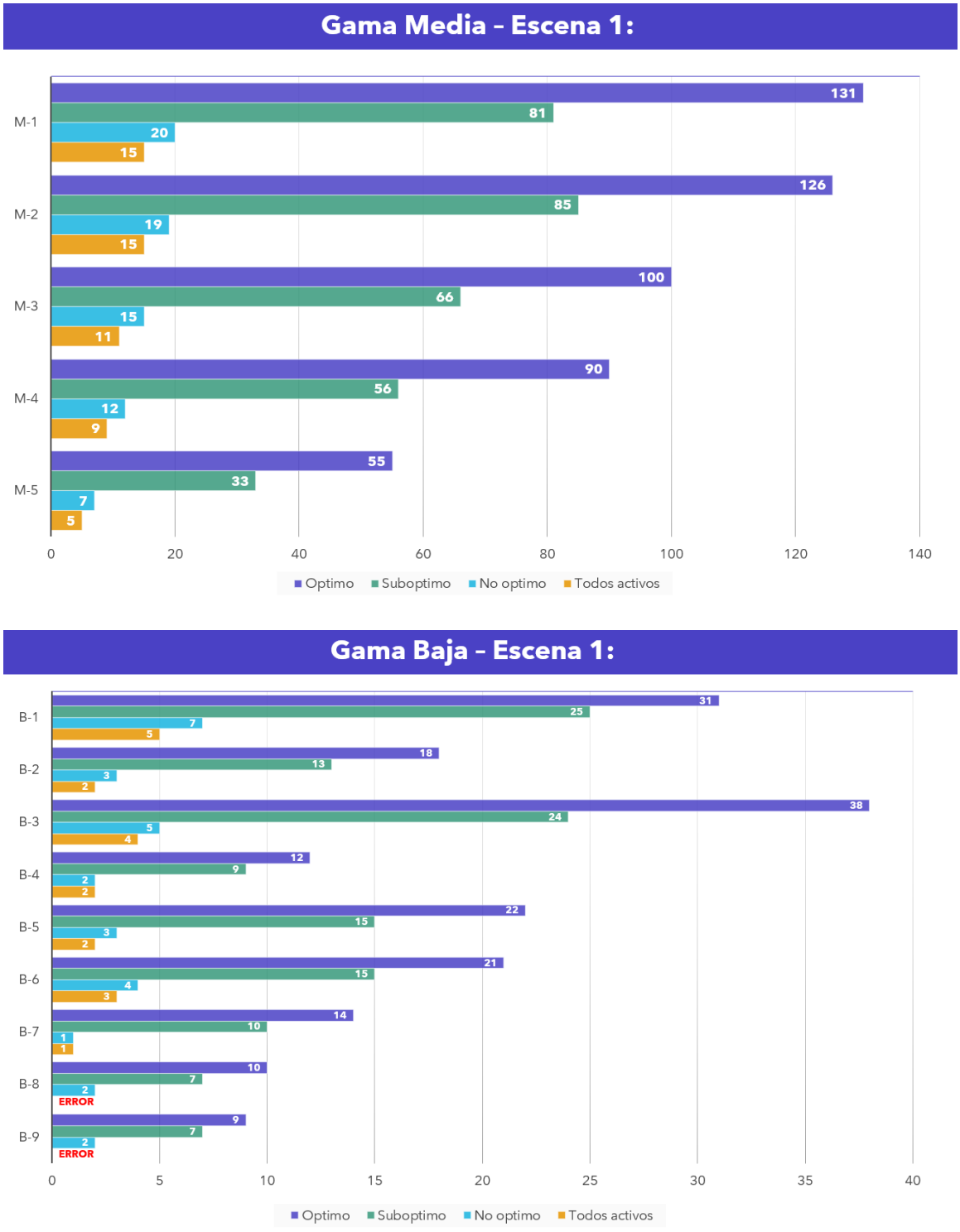


Figura 65. Resultados de la escena 1

Escena 2: Vista Aérea

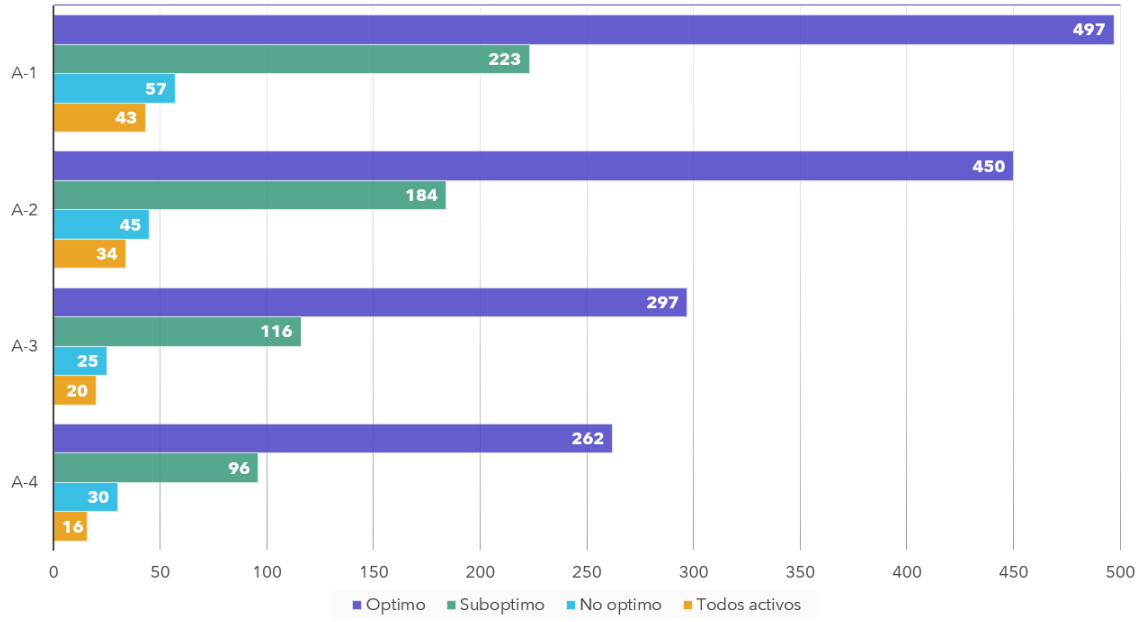
En este modo de cámara se aprecia la mayor cantidad de objetos visibles, sin embargo, es importante notar que las distancias de ciertos objetos activaran los LOD's en el modo óptimo, lo cual hará que en la mayoría de los casos mejore el rendimiento.

Observando los resultados, esta predicción es correcta, sumando entre 5 (en las gamas más bajas) hasta 100 FPS, un aproximado del 30% en todos los dispositivos. Desafortunadamente, al no aplicar las técnicas de optimización en los otros modos, en el 100% de los dispositivos se pierde un máximo de hasta el 60% del rendimiento del modo óptimo al sub- óptimo, por las mismas razones denotadas anteriormente.

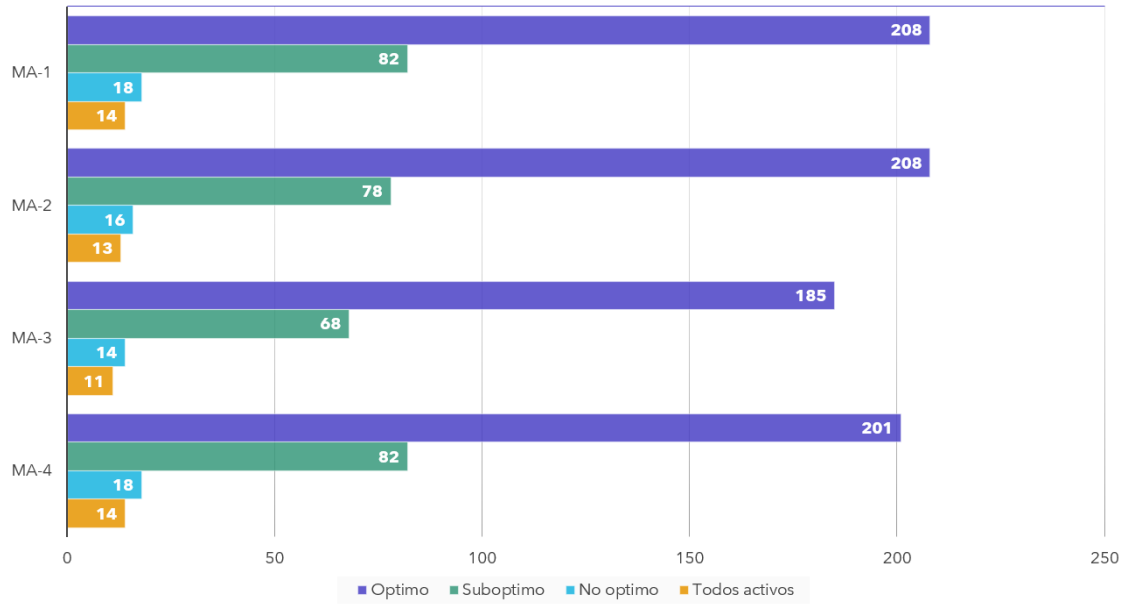
Al cuatriplicar la cantidad de polígonos entre el modo sub- óptimo y no óptimo, la vista aérea afecta el rendimiento de manera extrema, perdiendo hasta el 70% del rendimiento comparado con el anterior, perdiendo el mínimo jugable de FPS en el 90% de los dispositivos.

Figura 66. Escena 2 – Vista aérea

Gama Alta – Escena 2:



Gama Media-Alta – Escena 2:



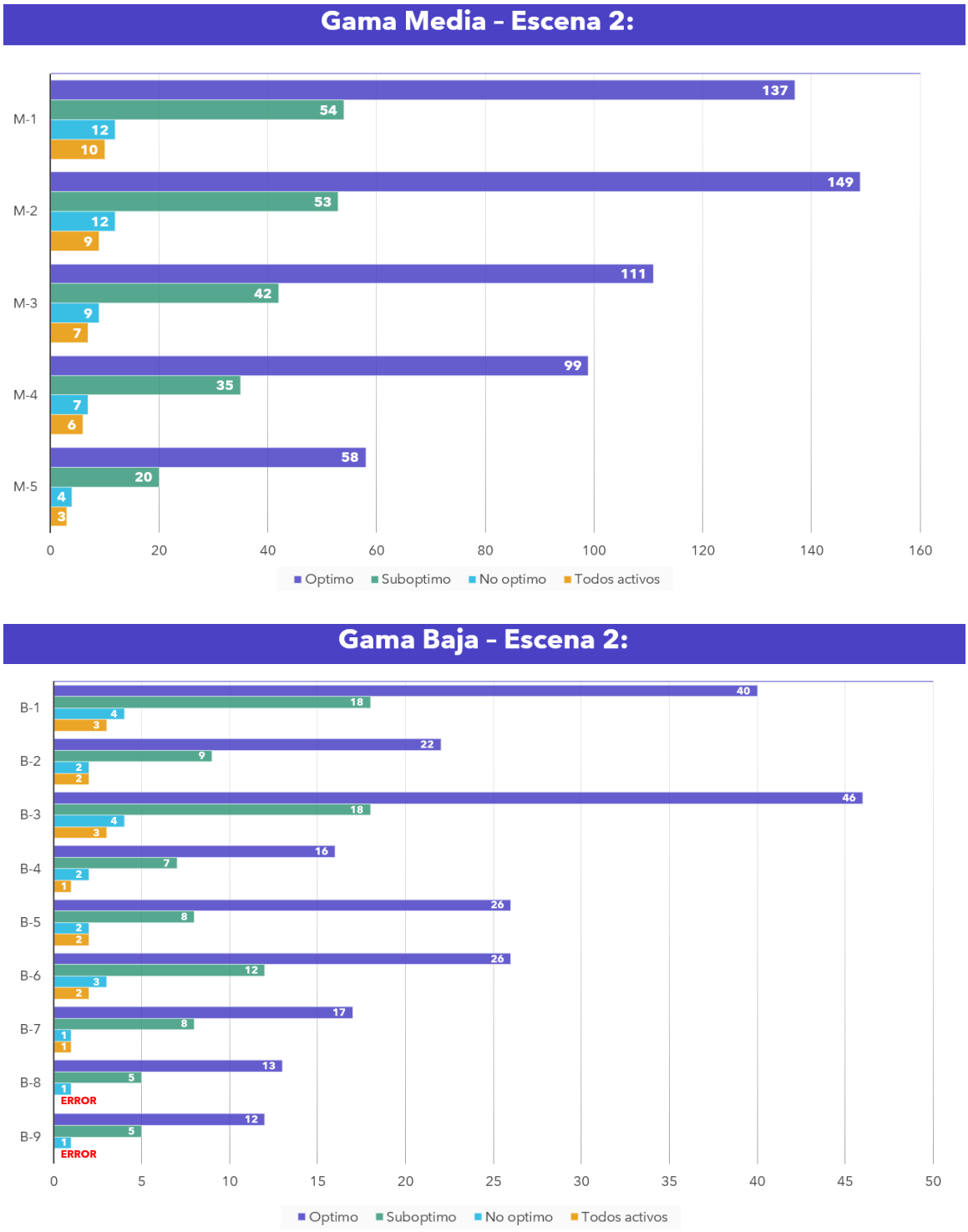


Figura 67. Resultados de la escena 2

Escena 3: Vista a distancia

En este análisis se muestra una mejora muy significativa de rendimiento y esto es notable debido al correcto uso de los LOD's a comparación de los modelos no óptimos de los Modos 2, 3 y 4. Esto es debido a que cuando la cámara del usuario esta lejana al modelo a visualizar este se reduce en cantidad de polígonos a la menor representación posible del objeto brindando un mejor rendimiento y menos procesamiento requerido para renderizar dichos objetos. Es importante recalcar que en las computadoras de gama alta lograron llegar al límite de FPS que fue programado con anticipación, lo cual nos indica que es posible que lo superan por un gran margen y esto se puede observar gracias a la comparativa del Modo sub- óptimo con respecto a los demás dispositivos ya que estos rondan entre los 370 a 300 FPS a comparación de los 170 a 100 de los dispositivos de gama media alta. Las pérdidas de rendimiento ente el modo óptimo al sub- óptimo no son tan graves a comparación de las otras escenas, la razón de esto es el ángulo de la cámara y la interacción de Unity con objetos no visibles, o la eliminación selectiva, en otras palabras, si el objeto no es visible, no se renderiza, esto beneficia a los modos sub-óptimos, sin embargo, la falta de utilización de LOD's y el alto uso de polígonos es la causa principal de la perdida de FPS en esta cámara.

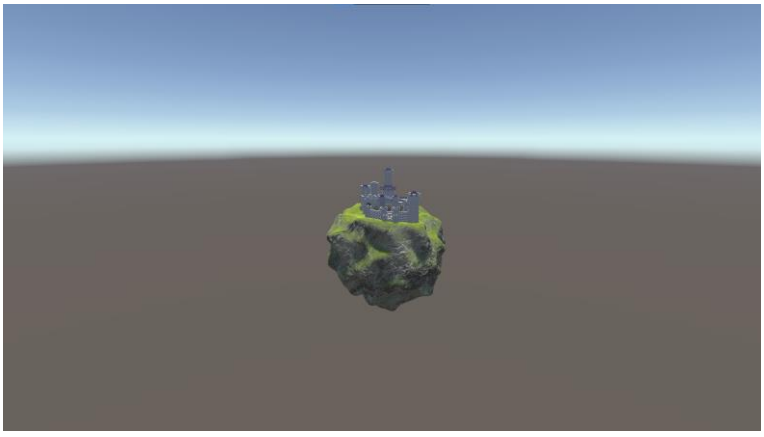
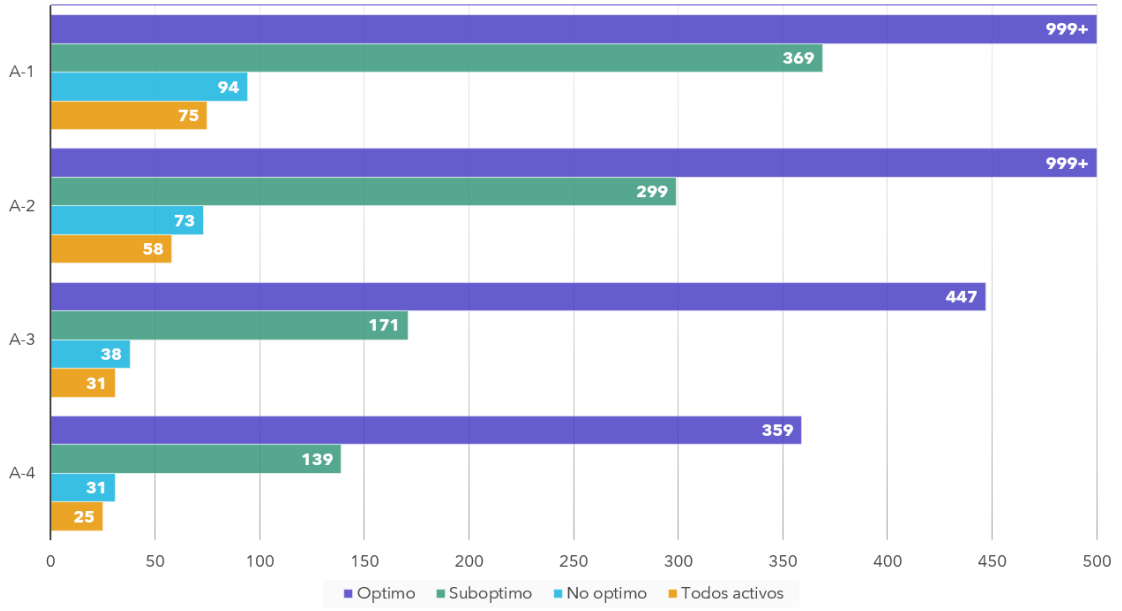
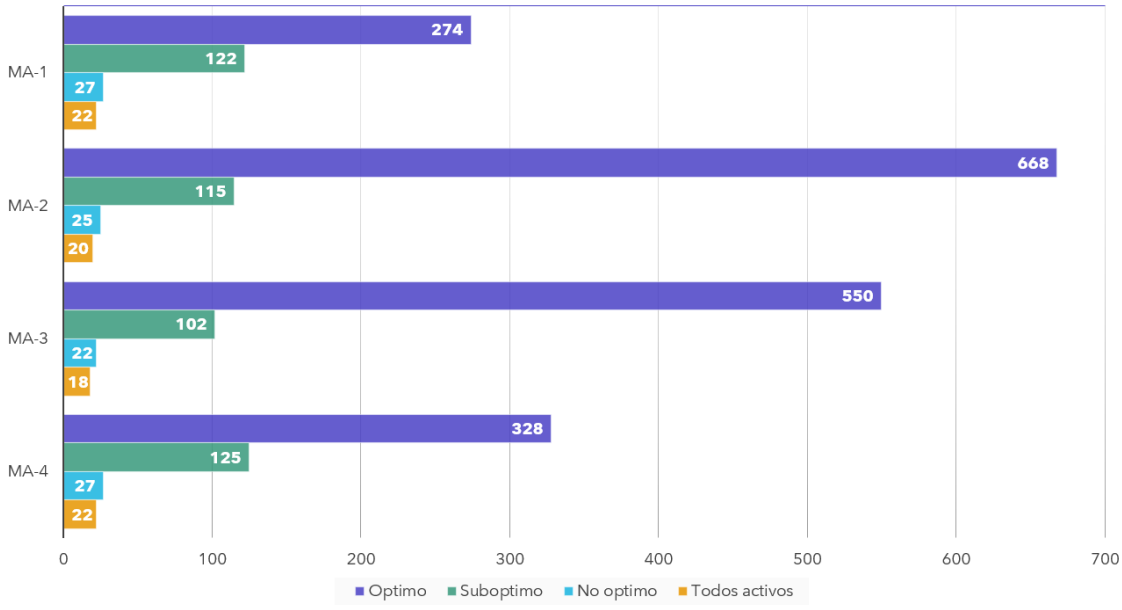


Figura 68. Escena 3 - Vista a distancia

Gama Alta – Escena 3:



Gama Media-Alta – Escena 3:



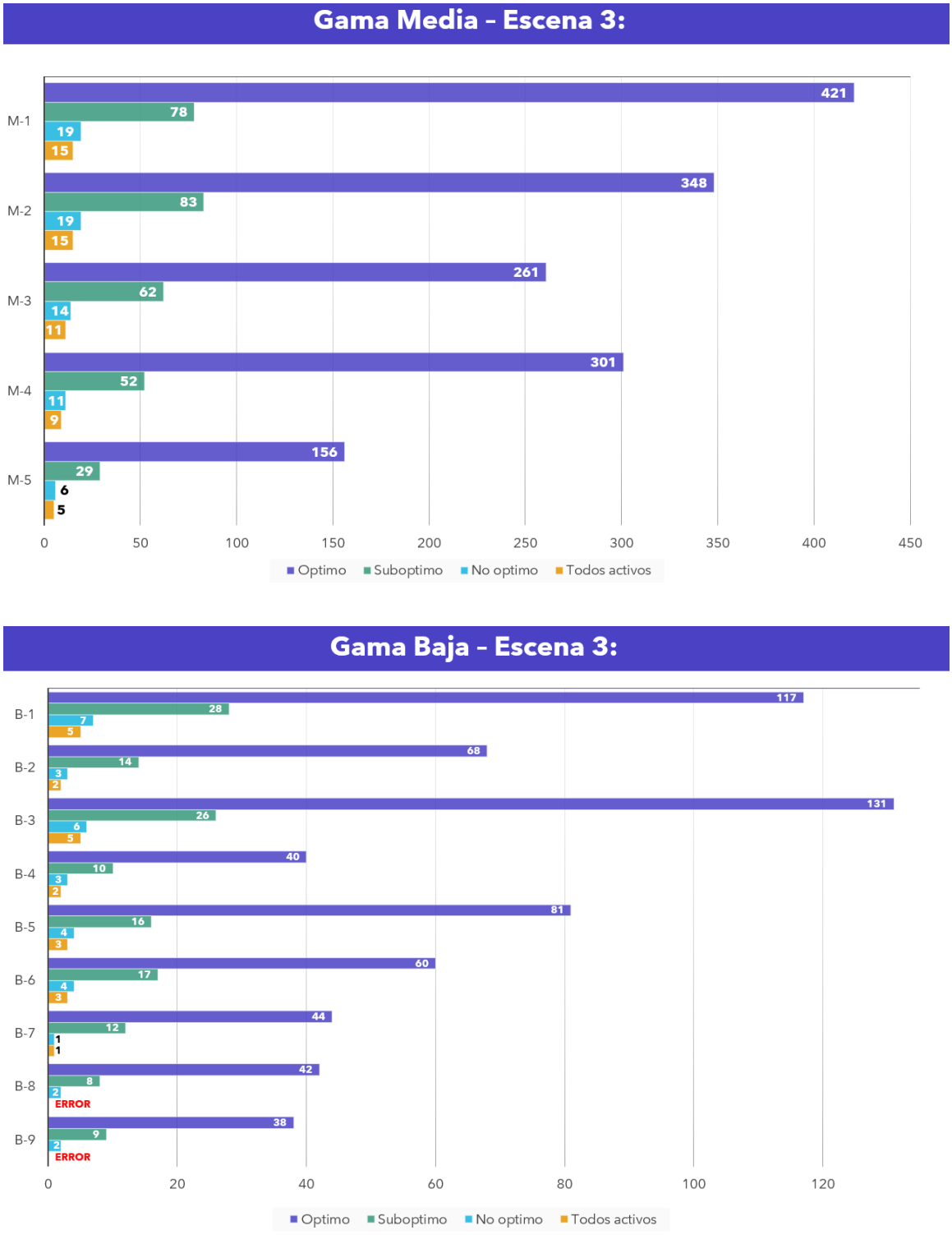


Figura 69. Resultados de la escena 3

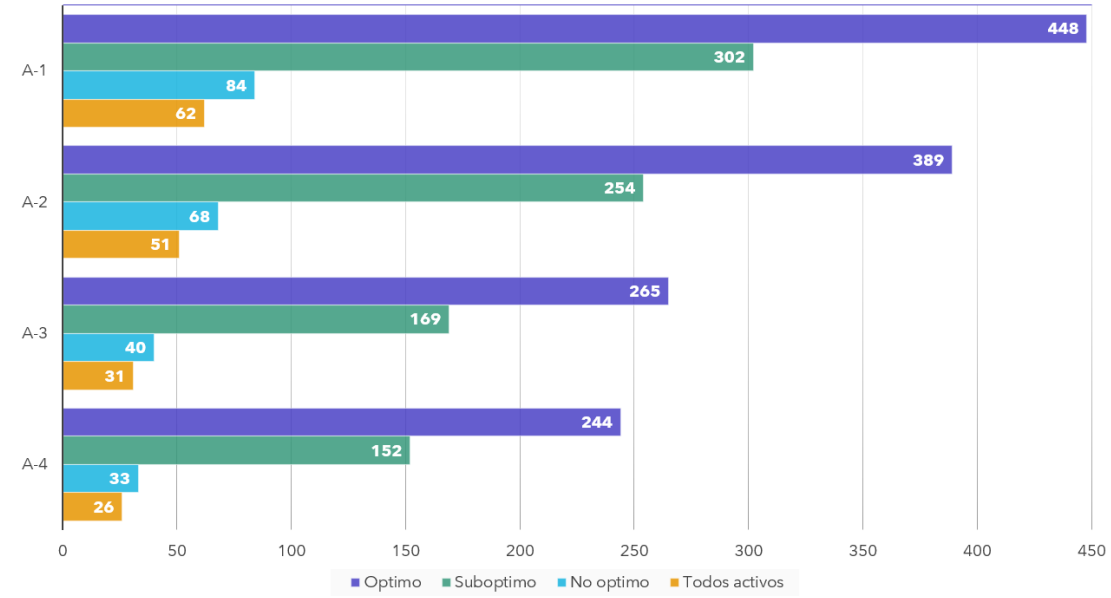
Escena 4: Patio Trasero

En esta vista se muestran principalmente los objetos que utilizan la mayor cantidad de polígonos, las barreras y las torres. A pesar de que el patrón de rendimiento es similar que las pruebas anteriores, se puede observar la perdida de FPS entre el modo óptimo y sub-óptimo del 30% aproximadamente, esto es causado por la visibilidad a partir del ángulo de vista seleccionado.

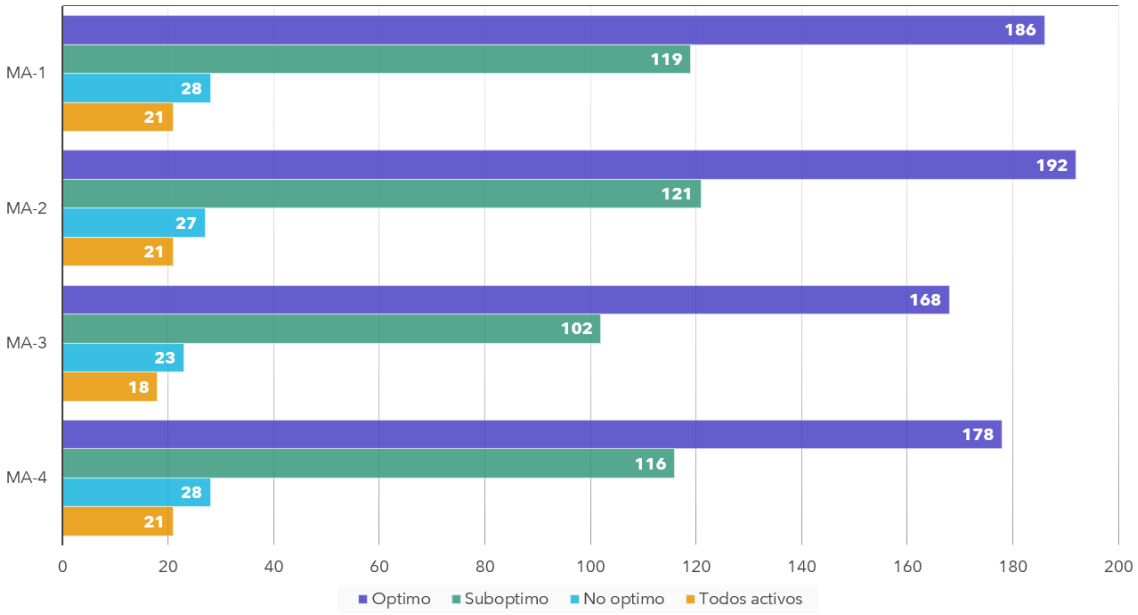
Nuevamente se puede notar la diferencia en FPS entre el modo sub- óptimo y no-óptimo, que, aun si se visualiza la misma cantidad de objetos, el cuatriplicar la cantidad de polígonos, la mayoría de los dispositivos pierden rendimiento de manera exorbitante, hasta un 70% (y en 85% de los dispositivos, reduce los FPS a una cantidad menor del mínimo jugable), esto sin generar cambios visualmente claros al observador, generando desventajas de jugabilidad, rendimiento, usabilidad e interacción sin beneficio alguno para el usuario.

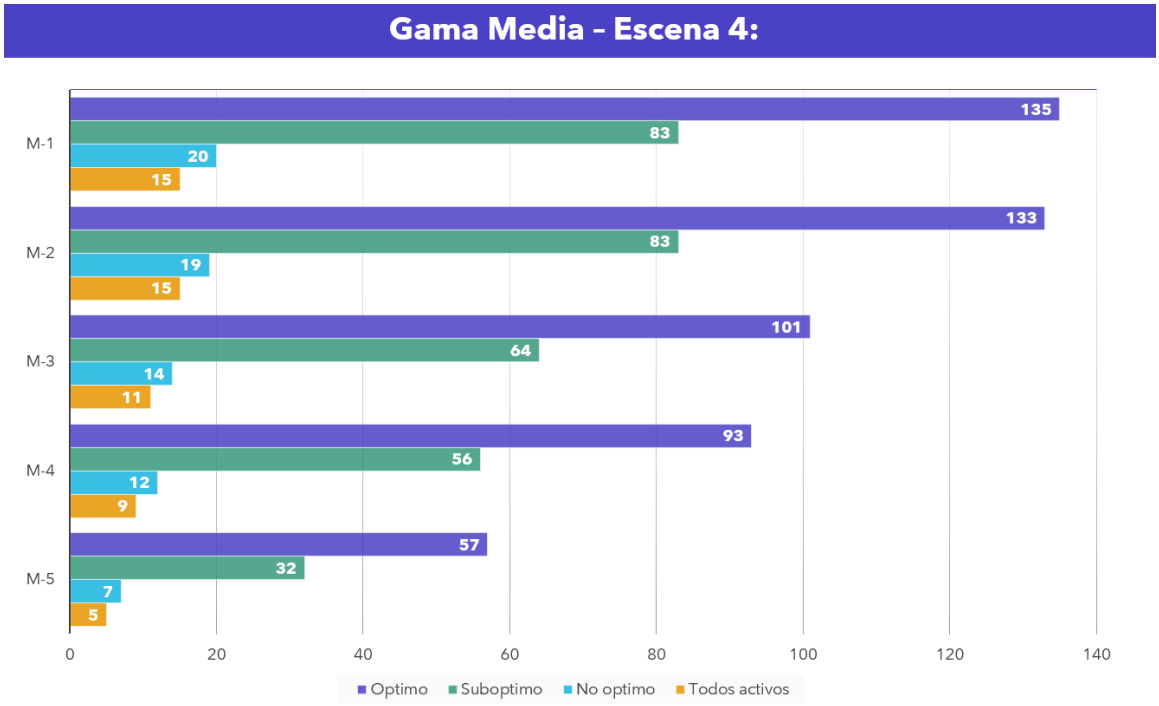
Figura 70. Escena 4 - Patio trasero

Gama Alta – Escena 4:



Gama Media-Alta – Escena 4:





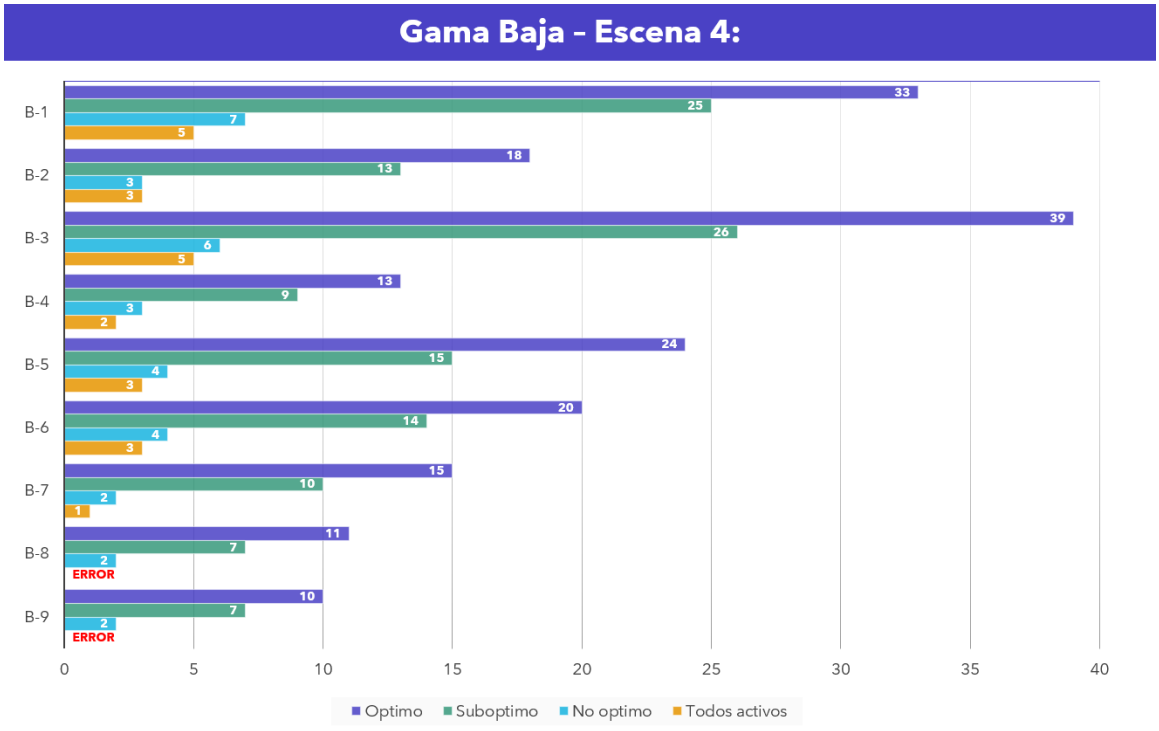
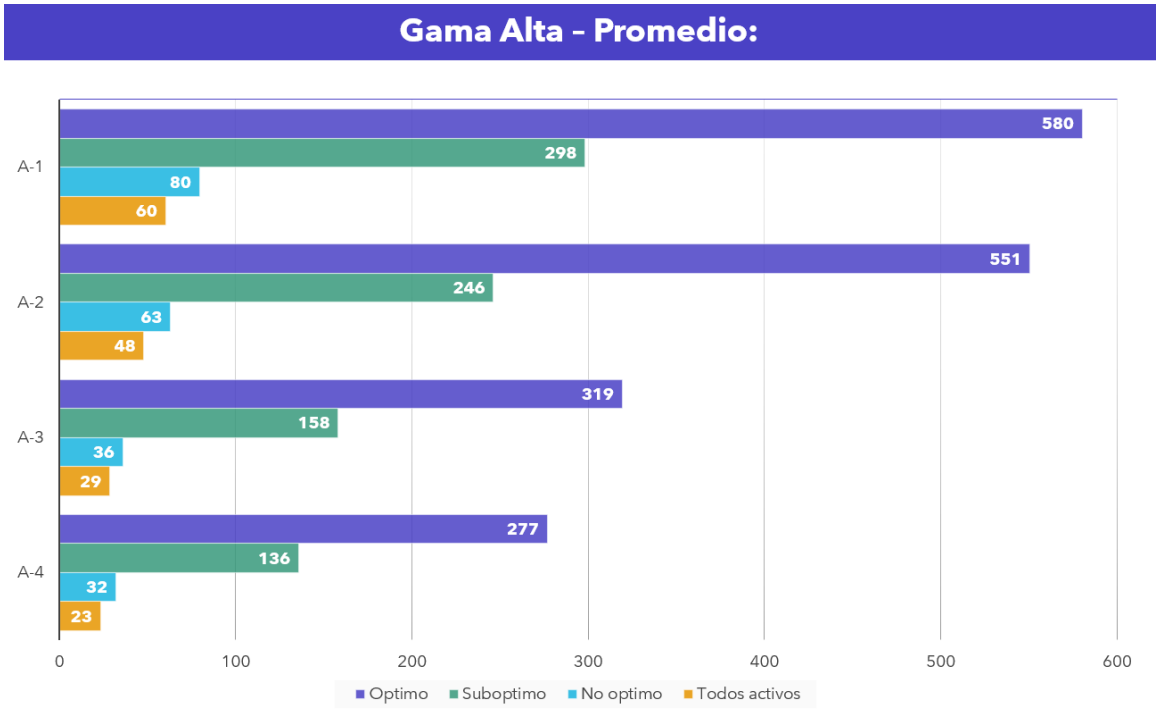
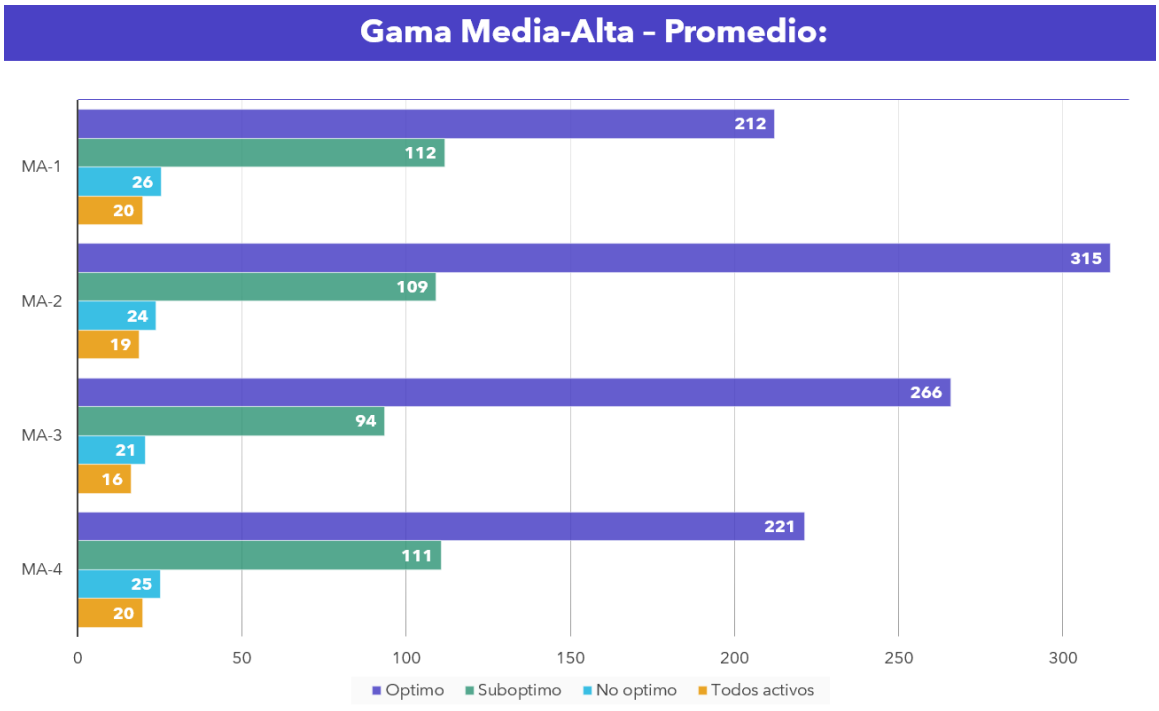


Figura 71. Resultados de la escena 4

Promedios:


Gama Alta – Promedio:
A-1
580
298
80
60
A-2
551
246
63
48
A-3
319
158
36
29
A-4
277
136
32
23
0
100
200
300
400
500
600
Optimo
Suboptimo
No optimo
Todos activos


Gama Media-Alta – Promedio:
MA-1
212
112
26
20
MA-2
315
109
24
19
MA-3
266
94
21
16
MA-4
221
111
25
20
0
50
100
150
200
250
300
Optimo
Suboptimo
No optimo
Todos activos

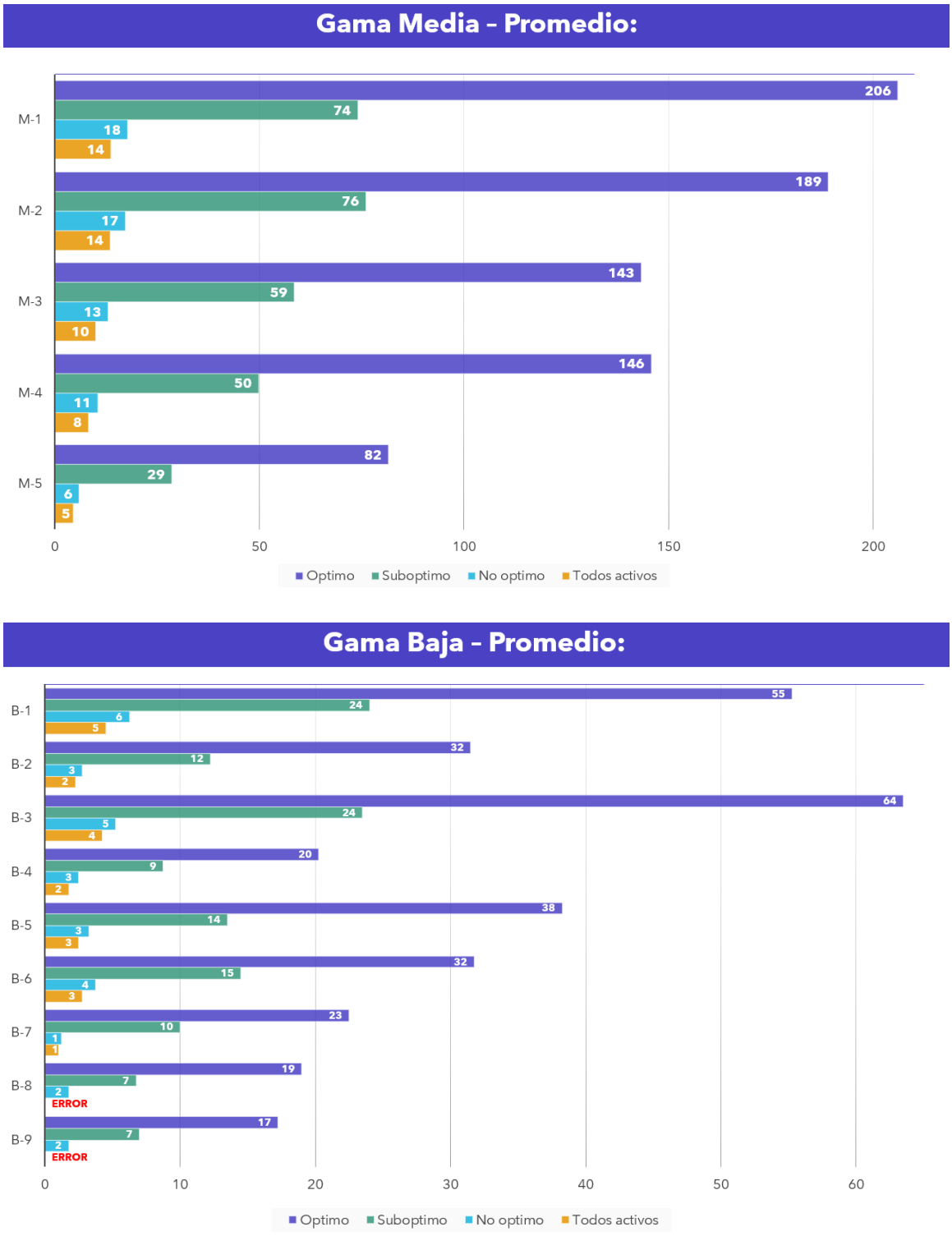


Figura 72. Resultados promedio de todas las escenas

A partir de las ejecuciones, se hizo una suma de todas las escenas mostradas anteriormente para lograr tener un resultado general. Se puede observar lo siguiente a partir del promedio y las pruebas:

- Todos los dispositivos pierden el mínimo del 45% de los FPS entre el modo óptimo y sub-óptimo.
- Se pierde el mínimo del 70% de los FPS a entre el modo sub- óptimo y no óptimo, sin importar el dispositivo, de la cual solo cuatro de los 22 dispositivos están en el mínimo jugable de fotogramas.
- La diferencia de rendimiento entre el modo no óptimo y TA (Todos Activos) es del 5% al 10%, lo cual hace que solo dos dispositivos puedan ejecutar el programa en el mínimo jugable.
- La diferencia visual entre cada uno de los modos no es fácilmente perceptible por los usuarios, un comentario comúnmente mencionado dentro de las pruebas de rendimiento era la confusión causada por la pérdida de FPS sin la pérdida de calidad de los modelos, a pesar de la existencia de un cambio extremo de polígonos entre modos.
- Cuatro de los 22 dispositivos ejecutan la herramienta de rendimiento a una cantidad de fotogramas por segundo menor al mínimo jugable.
- Dos dispositivos no fueron capaces de ejecutar el modo TA, siendo los modelos con la puntuación mínima tanto en los resultados como en los puntajes de GPU y CPU asignados.

CONCLUSIONES

A) Encuestas:

Es notable que los encuestados tienen conocimientos claros del diseño tridimensional, sin embargo, no están informados de las técnicas de optimización para lograr mejores modelos, existe una segregación clara entre los usuarios que no están informados del conocimiento de la optimización y técnicas de texturizado y aquellos que tienen la noción o están claros en ellos.

Los encuestados al comparar entre modelos de distintas cantidades de polígonos, mostraban confusión entre los modelos óptimos / subóptimo / no óptimos en adelante, lo cual demuestra que no es necesario emplear altas cantidades de polígonos para un modelo optimo o de buena calidad, sobre todo, si es observado a distancia.

Como nota, se debe considerar el factor de que la encuesta se enfoca en un grupo con conocimiento del modelado tridimensional, pero no son específicamente el público meta en el desarrollo de una aplicación o un videojuego, el cual no siempre esta informado o tiene interés de conocer u observar a detalle un modelo, este usuario puede solo estar interesado en si la aplicación lograra ejecutarse en su dispositivo de manera cómoda y fácil.

La importancia de la optimización se mostrará no solo en el rendimiento, o los fotogramas, también se mostrará en la cantidad de espacio que ocupa dentro de un proyecto, considerando la exorbitante cantidad de elementos dentro de una aplicación.

Los mapas de texturas bien aplicados en un modelo podrán no solo reducir la cantidad de polígonos que un objeto pueda tener, también puede lograr la ilusión de que

su calidad es mejor de lo que parece para el usuario meta; el combinar el buen uso de detalles y técnicas de optimizado lograra un mejor resultado visual.

La educación sobre la optimización y las técnicas de texturización en el modelado tridimensional es vital para el área de enfoque de la carrera de Diseño Digital de Medios Interactivos: el desarrollo de aplicaciones y videojuegos, sobre todo si el usuario desea armar aplicaciones enfocadas a multiplataforma, consolas o móvil, los cuales requieren mayor cantidad de optimizaciones para mayor compatibilidad.

B) Pruebas de rendimiento

Con lo anteriormente analizado, podemos ahora tomar en cuenta uno de los aspectos más importantes en la ejecución: el dispositivo con el que se abrirá la aplicación o juego.

A pesar de solo usar un tercio de los modelos en los modos sub-óptimo y no óptimo, la diferencia de rendimiento es desmesurado, aun si se fabricaron varias escenas para obtener resultados distintos entre ellos, el patrón de reducción de rendimiento y FPS fue constante entre modos, demostrando pérdidas de hasta el 70%, esto sin considerar el impacto de temperaturas, ergo, las desventajas de la mala optimización de modelos son visualmente observables y comprobables.

Se debe de considerar, algunos dispositivos no fueron capaces de ejecutar la aplicación de manera óptima, en aspectos mínimos jugables, ergo, la aplicación de rendimiento se puede seguir optimizando para estos modelos de recursos mínimos.

Como se mencionaba dentro de los resultados del promedio, muchos de los usuarios al momento de ejecución, cuestionaban la razón de la perdida de FPS aun si

no se observaban cambios visuales (algunos denotaron la perdida de modelos entre modos, lo cual les generaba aún más confusión puesto a visualmente se perdían objetos en el escenario, sin embargo su rendimiento era menor), reafirmando los aspectos vitales en el desarrollo de modelos optimizados: el uso de altos polígonos no significa mayor calidad ni detalle, ya que se puede lograr un buen resultado sin la necesidad de emplear una alta cantidad.

Puede que el usuario no note las diferencias visuales entre modos (en este caso, entre un modelo optimizado y no), pero el dispositivo ejecutando la aplicación puede demostrarlo, vía su bajo rendimiento y FPS.

El uso de LOD's es vital para tener mejor optimización y rendimiento, sobre todo al considerar las distancias visuales, donde el usuario puede observar múltiples objetos, pero no bien definidos singularmente. A largas distancias, el objeto no debe ser altamente detallado poligonalmente hablando, deberá tener una cantidad mínima posible, y el usuario no debe de percibir las diferencias entre LOD's y el modelo óptimo.

Dentro del desarrollo, se debe delimitar no solo el estilo grafico visual, si no también se debe aclarar la cantidad de polígonos que los modelos deberán de manejar en promedio, esto auxiliara a un mejor flujo, equilibrio, peso y coherencia visual, además de los beneficios de rendimiento, mayor alcance y compatibilidad.

Se puede confirmar la posibilidad de analizar, observar y determinar los beneficios de las distintas técnicas de optimización de polígonos y texturización, (empleando limites meta de polígonos, mapas de texturas, aplicando LOD's, etc.) todos los proyectos de aplicaciones tridimensionales, sin importar su propósito, meta, o temática, lograran un

mejor rendimiento, flujo y alcance al tener en consideración estos aspectos de optimización.

Reflexiones

Gladys Michelle Carrillo Guerrero

Personalmente aprendí muchas cosas que jamás pensé que aprendería, increíblemente, mi enfoque e interés personal no es el modelado tridimensional. Este proyecto de investigación logro educarme de muchos aspectos que no sabía que existían, aprendí junto con este documento la importancia de estos puntos que se llevaron a cabo, los tomé en práctica, los viví, observe, y me asombre con ellos. Me siento bastante orgullosa de los resultados obtenidos de esta investigación, y creo que es algo que todos dentro del área de desarrollo y la carrera de DDMI deberían de considerar si su interés es el desarrollo tridimensional.

Como nota, y un comentario común de algunos de los compañeros que no comprendieron totalmente el propósito de nuestra investigación, no es malo el uso de muchos polígonos, lo malo es el sobreuso de polígonos; hacer modelos de muy alta calidad y polígonos no es malo mientras tengas en claro en propósito de su elaboración.

En mi opinión, la experiencia que lleve a través de la carrera me hace sentir que faltaron muchos aspectos vitales para lograr una mejor calidad de productos, en este caso, los modelos tridimensionales, ya que lo educado es lo mínimo posible para lograr el “cómo hacerlo”, pero no el “porque se debe hacer o no”, ergo, un desarrollo profesional o de mayor calidad. Muchos de nosotros, aquellos realmente interesados en el desarrollo, o en general, otros aspirantes e independientes, muestran una falta de

conocimiento e información en relación con la optimización, esta fluctuación causa problemas para lograr el alcance esperado, en otras palabras, puede que nos sintamos incapaces de lograr desarrollar algo que sea considerado profesional o de alta calidad.

Cabe mencionar que no es una manera de desviar culpas o excusar, es vital comprender que gracias a la era en que vivimos, hay demasiada información, recursos, conocimiento; no es culpa de los docentes el no lograr demostrar o educar todo, debemos entender como estudiantes y próximos a ser profesionales que el conocimiento aprendido no es permanente, está en constante actualización, y es nuestro rol considerarlo y tener en cuenta que como desarrolladores, debemos de mantenernos lo más actualizados posibles de los nuevos contenidos, las nuevas herramientas, todo lo que nos pueda auxiliar a mejorar.

Siento personalmente que tener una buena relación y comunicación, sobre todo en la etapa de estudiantes, con los docentes encargados es importante para lograr un equilibrio de conocimiento e información, ellos no lo saben todo, y nosotros tampoco, entre los dos nos podemos auxiliar para lograr un mejor entendimiento general, por esta razón agradezco a los profesores que abrieron sus puertas y escucharon una opinión o retroalimentación, aun si errónea, personal o de otros estudiantes, y lograron mejorar como docentes dentro de la carrera por su capacidad de escuchar, algo que a veces, a todos, sin importar el nivel, conocimiento, titulo, etc. Podemos olvidar con facilidad. De verdad.

Victor Daniel Ferreyra Márquez

A lo largo del estudio de la carrera, me percaté del potencial que hay para mejorar la enseñanza dentro de la institución y que por sí solas no llevan a un buen producto final

debido a que no se enseñan las buenas y malas prácticas dependiendo del objetivo al que se va a desarrollar, lo cual en el caso de este proyecto de investigación es enfocado a los videojuegos. Con esto quiero remarcar que hay que ser autodidactas y no depender únicamente de lo que se aprende en la institución y del conocimiento limitado de los docentes.

Tomando esto en cuenta no me sorprende los resultados de la investigación y la falta de conocimiento de dicho tema debido a que no se enseñan durante las clases y que un reajuste de materias es más que necesario para mejorar la calidad del plan de estudios y la dirección de este.

Si bien, creía saber el cómo afectaban los polígonos al rendimiento de un juego o aplicación, pero no tenía en mente que fuera tan grande el impacto, esta investigación me ayudó a estudiar y aprender estas técnicas que tomare en cuenta para proyectos a futuro para mejorar la experiencia de mis usuarios.

BIBLIOGRAFÍA

AllPlan. (2018). *30-35 Percent Increased Design Productivity Using 3D Modeling*. AllPlan: A Nemetschek Company. https://info.allplan.com/hubfs/07_Guides/ALLPLAN_Whitepaper_Vorteile_3D_Planung_ing_EN.pdf

Arm Developer. (2021). *Geometry Best Practices for Artists*. https://developer.arm.com/solutions/graphics-and-gaming/developer-guides/game-artist-guides/geometry-best-practices/single-page

Autodesk. (2016, 11 mayo). *Polygonal modeling*. Autodesk Knowledge Network. https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/Maya/files/GUID-7941F97A-36E8-47FE-95D1-71412A3B3017-htm.html

Ayub, A. (2021, 7 abril). *18 optimization techniques for mobile and VR*. Frag Games. https://www.frag-games.com/2021/03/22/18-optimization-techniques-for-mobile-and-vr%E2%80%8B/

Blázquez, D. (2006, 22 junio). *Optimización de gráficos 3D para videojuegos*. Exelweiss. https://www.exelweiss.com/blog/18/optimizacion-de-graficos-3d-para-videojuegos/

Blender Foundation. (2021, 7 julio). *Blender by the numbers – 2020*. blender.org. https://www.blender.org/news/blender-by-the-numbers-2020/

Catarina, A. (2018). *3D Modeling Pipeline for Games* [Master's thesis]. https://www.theseus.fi/bitstream/handle/10024/156811/Gomes_Sarmento_Da_Fonseca_Ana.pdf?sequence=1&isAllowed=y

Chilana, P., Hudson, N., Bhaduri, S., Shashikumar, P., & Kane, S. (2018). *Supporting Remote Real-Time Expert Help: Opportunities and Challenges for Novice 3D Modelers*. Symposium on Visual Languages and Human-Centric Computing. https://ixlab.cs.sfu.ca/static/img/siteAssets/publications/supporting-remote-real-time-expert-help-opportunities-and-challenges-for-novice-3d-modelers/ChilanaVLHCC2018.pdf

Coelho, F. (2019, 17 mayo). *Metodología. Significados.* *https://www.significados.com/metodologia*

Geig, M. (2013, 19 diciembre). *Working with models, materials, and textures in unity game development*. InformIT. https://www.informit.com/articles/article.aspx?p=2162089

Gómez, H. (2013, 16 mayo). Desinformación en Internet y hegemonía en redes sociales. *Gestión de las Personas y Tecnología*, *16*(6), 39-53. https://www.redalyc.org/pdf/4778/477847110004.pdf

Gon, K. (2020, 23 mayo). *Top heavy files of Yandere Simulator: part 2* [Video]. YouTube. https://www.youtube.com/watch?v=fHe62W-D1fQ

INEGI. (2021). *Encuesta Nacional sobre Disponibilidad y Uso de Tecnologías de la Información en los Hogares (ENDUTIH) 2020* (325/21). https://www.inegi.org.mx/contenidos/saladeprensa/boletines/2021/OtrTemEcon/ENDUTIH_2020.pdf

Kukuh, A. (2015). *Optimize 3D Graphic for Culture Game by Using Polygon Reduction*. Journal of Theoretical and Applied Information Technology. https://www.jatit.org/volumes/Vol72No1/2Vol72No1.pdf
Lohikoski, L. (2013). *Optimization of 3D Game Models*. Södertörn University. https://www.diva-portal.org/smash/get/diva2:708048/FULLTEXT01.pdf
LowSpecGamer. (2019, 4 noviembre). *How does LOD work? How does it increase PC Gaming performance?* [Video]. YouTube. https://www.youtube.com/watch?v=I7uJWrIfugM
LowSpecGamer. (2020, 25 septiembre). *I got Ray Tracing working on an Athlon APU | Crysis Remastered* [Video]. YouTube. https://www.youtube.com/watch?v=Me74qw1sa-o
Malta, B. (2013). *3D Modeling Optimization for Multimedia Production* [Master's thesis]. https://www.theseus.fi/bitstream/handle/10024/58433/BrunoMaltaFinalThesis.pdf?sequence=1&isAllowed=y
Marshall, W. (2018, 4 noviembre). *How To Properly Optimize Your Low Poly Cylinders* [Video]. YouTube. https://www.youtube.com/watch?v=dpdyt4OMzXc
Meysman, D. (2020, 14 marzo). *Keeping your games 'optimized': Part 1 - Triangles*. Artstation. https://www.artstation.com/blogs/daanmeysman/7goy/keeping-your-games-optimized-part-1-triangles
Meysman, D. (2020, 26 marzo). *Keeping your games 'optimized': Part 2 - Drawcalls*. Artstation. https://www.artstation.com/blogs/daanmeysman/dGqm/keeping-your-games-optimized-part-2-drawcalls
Niko. (2010, 20 abril). *Low-poly tips*. CGmascot. https://www.cgmascot.com/design/low-poly-tips/
Niko. (2011, 14 noviembre). *Low-poly tips 2 – Game art asset optimization*. CGmascot. https://www.cgmascot.com/design/low-poly-tips-2/
Ovrick, J. (2017). Pro Tips: 3D Modeling best practices. https://www.artella.com/index.php/2017/10/18/pro-tips-3d-modeling-best-practices/
PassMark Software (2022, 31 octubre). *CPU Benchmark.* *https://www.cpubenchmark.net/*
Petty, J. (2019, 4 enero). *What is 3D modeling & what's it used for?* Concept Art Empire. https://conceptartempire.com/what-is-3d-modeling/
Petty, J. (2019). What is Retopology? (A Complete Intro Guide for Beginners) https://conceptartempire.com/retopology/
Powell, R. (2014). *Optimized Graphics for Handheld Real-time CG Applications* [Master's thesis]. https://www.diva-portal.org/smash/get/diva2:722454/FULLTEXT01.pdf
Pluralsight Creative (Digital Tutors). (2014, 20 marzo). *CG101: What is a Normal Map?* YouTube. https://www.youtube.com/watch?v=yHzIx41eiD4
Pluralsight. (2014, 8 abril). *What's the difference? A comparison of modeling for games and modeling for movies*. https://www.pluralsight.com/blog/film-games/whats-the-difference-a-comparison-of-modeling-for-games-and-modeling-for-movies
Royal Skies LLC. (2019, 19 agosto). *Blender 2.8 : Albedo Maps in 2 Minutes!!! (Gimp Tutorial)* [Video]. YouTube. https://www.youtube.com/watch?v=dX-LvaB36yk

Royal Skies LLC. (2019, 16 agosto). *Blender 2.8 : Gloss & Specular Maps In 2 Minutes!!! (Gimp - Tutorial)* [Video]. YouTube. https://www.youtube.com/watch?v=h5I0k3xifdg

Sanders, A. (2020, 4 febreoro). *What's the difference of animating for video games vs movies?* Lifewire. https://www.lifewire.com/animating-for-video-games-vs-movies-141113

SpeedTutor. (2015, 24 octubre). *Optimizing Games in Unity - Introduction / Statistics window / Series* [Video]. YouTube. https://www.youtube.com/watch?v=3x2DzXlBCkc&list=PLb34wPRpZdVdBxIoDYxOUMWOFeVf_X324&index=1

Unity. (2021). 2021 Gaming Report. *Unity insights from 2020 and predicted trends for 2021*, *1*(1), 3. https://images.response.unity3d.com/Web/Unity/%7B4eb56531-e6aa-492f-8fda-c68ae20af950%7D_2021_Gaming_Report_-_Operate_Solutions.pdf

Whitmer, B. (2019, 9 abril). *What is 3D rendering?* Create 3D Product Visuals & Augmented Reality For Commerce. https://www.threekit.com/blog/what-is-3d-rendering