

Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Centro de Computación Gráfica

Técnica de Ambient Occlusion para la Iluminación Global en Tiempo Real

Trabajo Especial de Grado
presentado ante la Ilustre
Universidad Central de Venezuela
Por el Bachiller
Alex Martínez
Para optar al título de
Licenciado en Computación

Tutor: Prof. Esmitt Ramírez

Caracas, 11 Noviembre de 2017

Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Centro de Computación Gráfica

ACTA DEL VEREDICTO

Quienes suscriben, Miembros del Jurado designado por el Consejo de la Escuela de Computación para examinar el Trabajo Especial de Grado, presentado por el Bachiller Alex Martínez C.I.: 20.755.137, con el título *Técnica de Ambient Occlusion para la Iluminación Global en Tiempo Real*, a los fines de cumplir con el requisito legal para optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído el trabajo por cada uno de los Miembros del Jurado, se fijó el día 17 de Octubre de 2017, a las 1:00 pm, para que su autor lo defendiera en forma pública, en el Centro de Computación Gráfica, lo cual se realizó mediante una exposición oral de su contenido, y luego respondió satisfactoriamente a las preguntas que les fueron formuladas por el Jurado, todo ello conforme a lo dispuesto en la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió Aprobarla.

En fe de lo cual se levanta la presente acta, en Caracas a los 25 días del mes de Octubre del año dos mil diecisiete, dejándose también constancia de que actuó como Coordinador del Jurado el Profesor Esmitt Ramírez. El Prof. Ramírez actuó como Jurado por Videoconferencia. Por ello, el Prof. Robinson Rivas como Director de la Escuela de Computación, firma la presente acta en original avalando al Prof. Ramírez.

REPUBLICA DE VENEZUELA
Facultad de Ciencias
Escuela de Computación
Universidad Central - Caracas

Prof. Esmitt Ramírez
Tutor

Prof. Francisco Moreno
Jurado

Profa. Adelis Nieves
Jurado

Resumen

En la Computación Gráfica, la iluminación global corresponde a un conjunto de técnicas en las cuales el cálculo de iluminación incidente sobre un objeto, toma en cuenta la interacción directa con las fuentes lumínicas y la luz reflejada por los objetos que componen la escena; muchas de estas técnicas representan un alto costo computacional, por ello se ha optado por alternativas de bajo costo para obtener un resultado similar. Tal es el caso de la técnica de *Ambient Occlusion* la cual es una alternativa que permite obtener resultados aproximados en un menor tiempo. A pesar de que la técnica original de *Ambient Occlusion* ofrece resultados en menor tiempo que la iluminación global aún no es suficiente para ser utilizada en aplicaciones en tiempo real, debido a esto se desarrolló la técnica de *Screen-Spaced Ambient Occlusion* (SSAO), el cual es una variante de *Ambient Occlusion* que opera solo sobre el *depth buffer* en espacio de pantalla, siendo ampliamente usado en aplicaciones en tiempo real. En este trabajo se realizan comparaciones entre diferentes técnicas inspiradas en SSAO con la finalidad de conocer sus ventajas y desventajas tanto visuales como de rendimiento, permitiendo concluir qué método se adapta mejor a las diferentes necesidades que se tienen al momento de desarrollar aplicaciones en tiempo real.

Palabras Claves: Iluminación Global, Ambient Occlusion, Screen Space, Depth Buffer, Tiempo Real.

Índice General

Índice de Figuras

Índice de Tablas

Introducción

Una gran variedad de técnicas se han desarrollado para simular la iluminación del mundo real y utilizarlo con la finalidad de representar objetos como si estuviesen iluminados por la luz del entorno, haciendo posible que se pueda fusionar la computación gráfica con escenas reales. Tal es el caso de iluminación global que consiste en un conjunto de técnicas que toman en cuenta las interacciones entre los distintos objetos de la escena obteniendo efectos como sombras suaves, reflexiones entre objetos, iluminación indirecta, entre otros. Sin embargo, cada una de estas técnicas representa un alto costo computacional, y debido a la necesidad de aplicar iluminación global a aplicaciones en tiempo real o plataformas donde no siempre se cuentan con los recursos necesarios para ejecutarse, se ha optado por alternativas de bajo costo computacional para obtener un resultado similar.

La técnica de *Ambient Occlusion* u Oclusión Ambiental permite obtener resultados aproximados a uno de los efectos generados por la iluminación global en un menor tiempo. Sin embargo, esta técnica sigue siendo un método global donde la iluminación en cada punto de contacto opera en función de las demás geometrías de la escena, donde se distinguen las áreas claras de las oscuras basándose en la proximidad de las superficies entre sí creando zonas más oscuras donde las superficies de las geometrías interactúan.

Debido a la necesidad de obtener resultados con aproximaciones en tiempo real, se implementó la técnica de *Screen-Spaced Ambient Occlusion* (SSAO), la cual es una variante de la técnica de *Ambient Occlusion* que opera sólo sobre el *depth buffer* en espacio de pantalla, siendo ampliamente empleada en aplicaciones en tiempo real. Con la popularidad obtenida por SSAO, nuevas técnicas se desarrollaron con el paso del tiempo con la intención de mejorar tanto la calidad visual como el rendimiento y resolver o minimizar problemas encontrados en esta técnica.

En este trabajo se presenta la implementación de cuatro técnicas inspiradas en SSAO, *Alchemy Screen-Space Ambient Obscurance, Scalable Ambient Obscurance, Multi-Scale Ambient Occlusion* y Unreal Engine 4 Ambient Occlusion, sobre las cuales se realizan diferentes pruebas entre el tiempo de ejecución, calidad visual, comparación perceptual, entre otras, con el objetivo de comparar los resultados obtenidos y exponer observaciones. Esto permite conocer las ventajas y desventajas de cada técnica y cuál escoger dependiendo de los requerimientos que debe cumplir.

En el capítulo 1 se presenta el planteamiento del problema, además de los objetivos generales y especifico. En el capítulo 2 se explican conceptos básicos de iluminación, tanto local como global, se explica con detenimiento la técnica de A*mbient Occlusion* y su adaptación en tiempo real *Screen-Spaced Ambient Occlusion.* En el capítulo 3 se explican teóricamente las cuatro técnicas basadas en *Screen-Spaced Ambient Occlusion* seleccionadas para este trabajo, además de las ventajas y desventajas que exponen sus desarrolladores. En el capítulo 4 se explica la implementación de las cuatro técnicas seleccionadas además de los pasos que se realizan antes y después de su ejecución. En el capítulo 5 se describen y analizan las pruebas realizadas sobre la implementación con la finalidad de realizar comparaciones entre el tiempo de ejecución, calidad visual, comparación perceptual, entre otros aspectos de las mismas. Finalmente, se exponen las conclusiones en el capítulo 5 y presentan algunas ideas que pueden ser implementadas a futuro con el objeto de mejorar las técnicas SSAO.

Capítulo 1

Problemática y Objetivos

1.1 Planteamiento del problema

En la actualidad existen una serie de técnicas que permiten crear escenas virtuales 3D que muestran resultados muy cercanos al mundo real, tal es el caso de iluminación global que comprende muchas de estas técnicas, sin embargo, solo es posible obtener dichos resultados para escenas estáticas o en animaciones, las cuales mientras más complejas y largas sean mayor será el tiempo de espera. Esto funciona cuando se desea crear fotografías o películas animadas, pero no cuando debe haber algún tipo de interacción con la escena, por ejemplo, los videojuegos donde es necesario obtener resultados en tiempo real. Debido a la necesidad de conseguir realismo a aplicaciones en tiempo real se desarrolló la técnica de *Ambient Occlusion.*

Para lograr iluminación global se requiere de hardware con gran capacidad de cálculo y recursos que muy pocas empresas pueden permitirse. Por otra parte, hoy en día existe una amplia cantidad de técnicas que permiten aplicar *Ambient Occlusion* en tiempo real, debido a esto cada desarrollador al momento de seleccionar una técnica, necesita primero realizar un estudio que le permita escoger la mejor técnica para la aplicación a realizar, donde también debe encontrar los parámetros que ofrecen los mejores resultados para la técnica seleccionada.

1.2 Justificación

A través de los años se han desarrollado diferentes técnicas a raíz de *Screen-Space Ambient Occlusion*. Debido a esto ahora existen técnicas con diferentes enfoques y resultados, resultando complicado decidir cuál técnica utilizar para desarrollar una aplicación, donde los desarrolladores deben realizar un estudio de las diferentes técnicas para luego proceder a desarrollar la aplicación deseada. Debido a todos estos problemas resulta útil disponer de un estudio ya realizado sobre las diferentes técnicas desarrolladas donde observar resultados visuales, de rendimiento, comparaciones entre ellas y conclusiones que permiten definir cuáles son las capacidades y limitaciones de cada una de ellas en diferentes escenarios.

La solución propuesta ayudará en la toma de decisiones al momento de necesitar alguna técnica de *Ambient Occlusion,* donde se tienen pruebas visuales y de rendimiento, con los parámetros propuestos por los desarrolladores y también al realizar diferentes cambios, obteniendo una serie de resultados que facilitan conocer la técnica ideal para cada aplicación.

1.3 Objetivo general

Realizar un estudio comparativo de cuatro técnicas de *Ambient occlusion.*

1.4 Objetivos específicos

- Implementar la técnica *Alchemy Screen-Space Ambient Obscurance*.
- Implementar la técnica *Scalable Ambient Obscurance*.
- Implementar la técnica *Unreal Engine 4 Ambient Occlusion*.
- Implementar la técnica *Multi-Scale Ambient Occlusion*.
- Comparar las técnicas *Alchemy AO, Scalable AO, Unreal Engine 4 AO y Multi-Scale AO* en su rendimiento.
- Comparar las técnicas *Alchemy AO, Scalable AO, Unreal Engine 4 AO y Multi-Scale AO* en su aspecto visual bajo las mismas condiciones.
- Comparar las técnicas *Alchemy AO, Scalable AO, Unreal Engine 4 AO y Multi-Scale AO* a diferentes resoluciones evaluando aspecto visual y rendimiento.

Capítulo 2

Iluminación y sus componentes

En el mundo real somos capaces de ver nuestro entorno gracias a los rayos de luz que alcanzan el ojo, ya sea porque la luz es reflejada desde una fuente de una superficie o porque proviene de la fuente de luz en sí. En un escenario virtual, tal como en la vida real, no somos capaces de ver un objeto a menos que esté iluminado o emita luz. En el campo de computación gráfica la iluminación local solo considera la información de la superficie, sin tomar en cuenta las demás entidades de la escena. También existen las técnicas de iluminación global, las cuales toman en cuenta la interacción de la luz con todas las entidades de la escena, pero vienen acompañadas por un gran costo computacional, al trabajar modelos de iluminación en conjunto se logra obtener imágenes lo más parecido posible a la realidad. La generación de imágenes realistas puede ser útil para el cine, medicina, juegos de video, arquitectura, ingeniería, simulaciones, entre otros. En este capítulo se describe cómo se simula la luz en un mundo virtual, donde tenemos dos modelos de iluminación: iluminación local e iluminación global y los componentes que la conforman.

2.1 Iluminación

Se puede decir que la técnica de iluminación influye de manera notable en la percepción de un determinado lugar. Por ello, si bien la forma, la posición y el movimiento son imprescindibles, la iluminación compone un elemento esencial en cualquier entorno de la vida real y virtual, aportando gran cantidad de realismo si se simula adecuadamente.
Fundamentalmente, dentro de las técnicas de computación gráfica existen dos modelos de iluminación: local y global.

2.1.1 Iluminación Local

Considera únicamente la luz que llega desde las fuentes hasta los objetos. Se calcula el color del rayo de luz en función del color y el material a partir del punto que proviene el rayo, y de las fuentes de luz presentes en la escena. Si bien este modelo de iluminación cumple con iluminar la escena, la apariencia resultante se encuentra muy lejos de ser realista [14].
Sin embargo, el costo computacional es bajo, siendo utilizado en aquellas aplicaciones donde no se requieran efectos más sofisticados o donde el rendimiento debe ser muy alto tales como, una aplicación con millones de elementos o en un entorno interactivo en tiempo real. Para mejorar su apariencia existen múltiples técnicas que añaden efectos visuales, como sombras, las cuales resultan de gran importancia para dotar de realismo las escenas virtuales.
El modelo de iluminación añade una serie de componentes que se calculan de manera independiente para luego combinarlas, y obtener un efecto de iluminación total para un lugar particular sobre la superficie de un material. Estos componentes son ambiental, difuso y especular.

- El componente difuso se esparce por la superficie equitativamente en todas las direcciones para cada fuente de luz en particular. No importa en cuál dirección está la cámara u ojo, pero sí la dirección de la luz. Es más brillante cuando la superficie encara directamente la fuente de luz, simplemente porque se obtiene más luz que en otra orientación. El cálculo del componente difuso depende de las normales de la superficie y de la dirección de la fuente de luz, pero no de la dirección del ojo, también depende del color de la superficie.
- El componente especular es luz reflejada directamente por la superficie. Este reflejo se refiere a qué tanto el material de la superficie actúa como un espejo. Mientras un material metálico refleja un componente especular muy brillante, un material de tela refleja prácticamente nada. Para calcular el componente especular se requiere la normal de la superficie, la dirección de la luz y la del ojo.
- El componente ambiental es luz que no proviene de una dirección específica, se considera como una constante a través de la escena, formando una buena aproximación de luz esparcida.

2.1.2 Iluminación Global

Se considera iluminación global al conjunto de técnicas de iluminación que tienen en cuenta las interacciones entre los distintos objetos de la escena, es decir, considera la luz reflejada por un punto teniendo en cuenta toda la luz que recibe y no sólo la que proviene de las fuentes de luz. Los efectos que se consiguen con este tipo de técnicas son sombras suaves, iluminación indirecta y transparencias. El conjunto de estos efectos hace que la apariencia visual de los modelos sea mucho más realista que utilizando técnicas locales. Sin embargo, su costo computacional es muy elevado y está limitado a una cantidad baja de objetos.

Debido al alto costo computacional que representa la iluminación global, se han desarrollado alternativas de bajo costo que permitan obtener un resultado similar al obtenido con iluminación global. Tal es el caso de la técnica de *Ambient Occlusion* [7] que a pesar de obtener resultados menos realistas es mucho más rápido y menos complejo de calcular que otros métodos por lo que es muy popular entre los desarrolladores de video juegos y en la animación de producción.

2.1.3 Técnicas de Iluminación Global

Para dotar de realismo en las escenas 3D, es indispensable la iluminación. Lamentablemente, la luz es demasiado compleja para que sea fácil de modelar con un computador orientado a trabajos personales, debido a esto existen diferentes técnicas que aproximan los principios de la luz y que permiten obtener resultados que engañan la vista. A continuación se exponen algunas de las técnicas con mayor relevancia.

2.1.3.1 Ray Tracing (Traza de rayos)

Esta técnica consiste en lanzar rayos que intersecten los distintos objetos para determinar las superficies visibles en la escena, trazando rayos desde el observador hasta la escena a través del plano de la imagen. De todas las intersecciones ocurridas a través del rayo con los diferentes objetos de la escena, aquella intersección que esté más cerca del observador determinará cuál será el objeto visible [8]. Con esta técnica se pueden realizar efectos globales de iluminación como reflexiones, refracciones o sombras tal como se muestra en la Figura 1 donde se observa que se disparan diferentes rayos desde la cámara a la escena a través de un plano imagen donde sólo se mostrará el resultado de la intersección más cercana. Por otra parte, se define donde la escena se encuentra iluminada, disparando un rayo de rebote que va desde el punto de intersección a la fuente de luz, en caso de no haber intersección en el trayecto se puede concluir que dicho punto de intersección se encuentra iluminado.

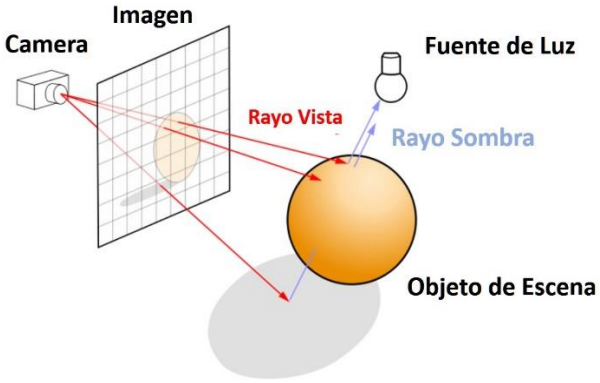


Figura 1: Modelo básico de Ray Tracing

Ray tracing produce resultados de muy alto realismo, pero a un mayor costo computacional. Este se basa en muestrear un subconjunto finito del conjunto infinito de caminos posibles, es decir, a más muestras, más tiempo y más memoria es necesaria, pero se logrará más exactitud. Proporciona resultados más realistas en superficies especulares, brillos, reflejos, entre otros, pero no consigue sombras suaves ni efectos de iluminación difusa.

2.1.3.2 Radiosity (Radiosidad)

Se basa en una aplicación del método de elementos finitos para resolver la ecuación de rendering en escenas con superficies puramente difusas. Para implementar algún algoritmo de *Radiosity*, se calcula la cantidad de luz que llega a cada punto de la escena lanzando rayos en todas direcciones y acumulando la cantidad de luz recibida [9].
Resulta una técnica muy sensible a la complejidad de la escena, un aumento de objetos de la escena lleva consigo un aumento enorme en el tiempo de cálculo y necesidades de memoria, debido a que toda o gran parte de la geometría debe estar en memoria. En la Figura 2 se puede apreciar una comparación de una escena renderizada sin *Radiosity* y con *Radiosity*.

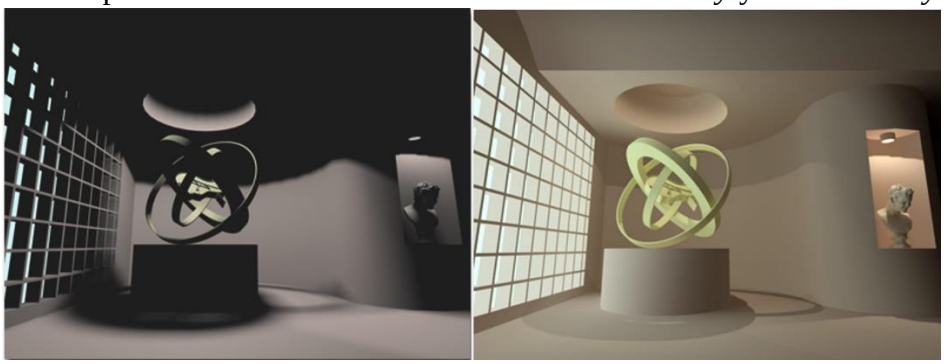


Figura 2: A la izquierda una escena renderizada sin Radiosity y a la derecha la misma escena con Radiosity.

Puede decirse que hay dos grandes métodos de iluminación global: los basados en *Ray Tracing* que son eficaces para computar la iluminación indirecta reflejada, pero no la difusa y los basados en *Radiosity* que son eficaces para computar iluminación indirecta difusa, pero no la reflejada. De esta situación se derivan nuevos métodos que combinan ambas técnicas en dos o más pasos y se denominan *Multipass Methods* (de múltiples pasadas) que combinan *Radiosity* y *Ray Tracing.*

2.1.3.3 Path Tracing

Es un algoritmo de renderizado similar a *Ray Tracing* donde se disparan rayos desde el observador a través de la escena. *Path Tracing* usa muestreo pseudoaleatorio(proceso que parece producir números al azar, pero no lo hace realmente) para computar incrementalmente la imagen final. El muestreo pseudoaleatorio hace posible renderizar fenómenos complejos que no son manejados en un *Ray Tracing* regular, pero generalmente tarda más en producir resultados de alta calidad [10].
El muestreo pseudoaleatorio genera ruido en la imagen producida. El ruido es removido dejando que el algoritmo genere más muestras. *Path tracing* a diferencia de *Ray Tracing* puede simular sombras suaves, *depth of field, motion blur*, *caustics, Ambient Occlusion* e iluminación indirecta. En la Figura 3 se observa el resultado de aplicar *Path Tracing* al generar 4 rayos por cada píxel muestreado y la misma escena pero con 1024 rayos por píxel.

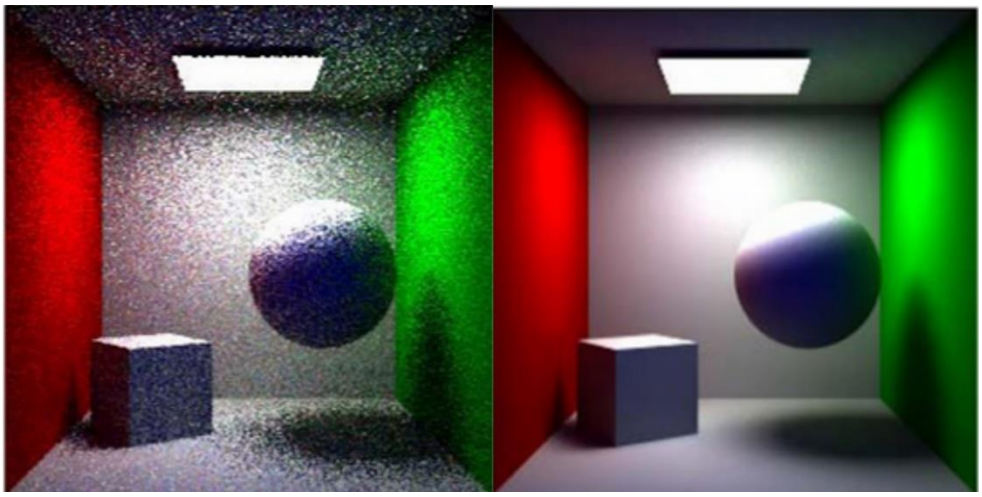


Figura 3: A la izquierda una escena renderizada con 4 rayos generados por píxel y a la derecha la misma escena con 1024 rayos generados por píxel muestreado.

Se pueden obtener imágenes de muy alto realismo, pero requiere de un tiempo computacional demasiado elevado. En la imagen de la izquierda de la Figura 3 podemos percibir ruido y como se compensa añadiendo más rayos.

2.1.3.4 Photon Mapping (Mapa de fotones)

Recrea el comportamiento físico de la luz, lanzando fotones desde los focos de luz que pueden rebotar o adherirse a los objetos. Es un algoritmo versátil, capaz de simular iluminación global, incluyendo cáusticas [11].
El algoritmo de iluminación global basado en fotones es un método de dos pasadas:

- En el primer paso se construye la estructura conocida como mapa de fotones a partir de los fotones emitidos desde las fuentes de luz hasta la escena. Esta estructura por lo general es un *k-d tree* y donde sólo se almacenan los fotones que golpean objetos no especulares. En la Figura 4 se muestra el resultado de la primera pasada de photon mapping.

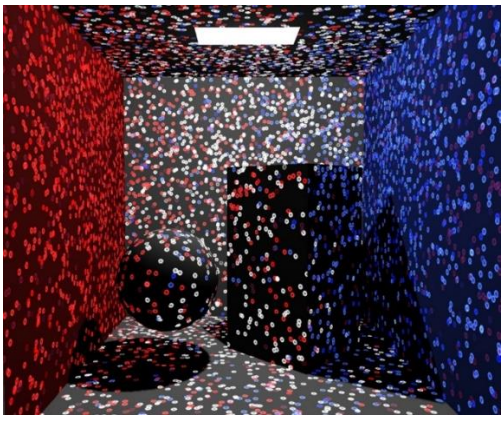


Figura 4: Primera pasada de Photon Mapping.

- El segundo paso es la fase de rendering, donde utiliza técnicas estadísticas en la estructura del mapa de fotones para extraer información sobre el flujo de entrada y radiancia reflejada en cualquier punto de la escena. En la Figura 5 se aprecia el resultado final de *Photon Mapping.*

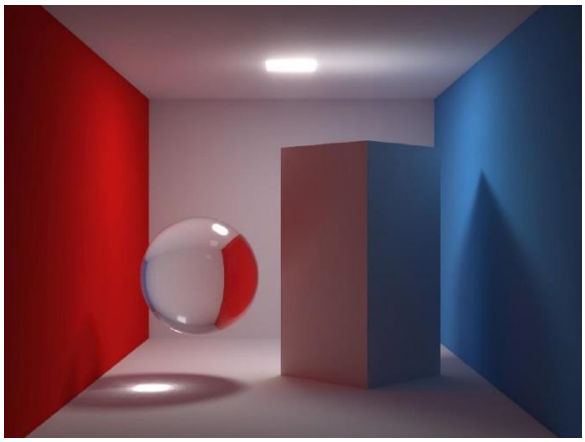


Figura 5: Photon Mapping Resultado Final con Caústicos.

Photon Mapping no está asociado a la geometría de la escena, siendo capaz de simular iluminación global de escenas complejas que contiene millones de triángulos.

El conjunto de estos efectos hace que la apariencia visual de los modelos sea mucho más realista que utilizando técnicas de carácter local. Sin embargo, su coste computacional las restringe a aplicaciones con un número limitado de objetos o donde el rendimiento no es el objetivo. Una alternativa para conseguir apariencia cercana a iluminación global es utilizar la técnica de *Ambient Occlusion* la cual genera resultados un poco menos realista, pero es mucho más rápido y menos complejo de calcular que los diferentes métodos de iluminación global.

2.2 Ambient Occlusion

Ambient Occlusion es una técnica de iluminación que se utiliza comúnmente para crear sombras suaves en objetos. La técnica de *Ambient Occlusion* no se utiliza para crear el tipo de sombras que se proyectan a partir de objetos con una luz que brilla directamente sobre ellos [7]. En cambio, *Ambient Occlusion* genera el tipo de sombras profundas que aparecen en las esquinas o los pliegues de los objetos, estructuras o elementos que nos rodean, siendo difícil que la luz alcance, tal como se muestra en la Figura 6.

Figura 6: *Ambient Occlusion* Resultado Final.

Técnicamente hablando, *Ambient Occlusion* es una técnica de iluminación global. Sin embargo, se refiere a menudo como una alternativa de bajo costo computacional a la iluminación global. Algunos *renders* incluyen la técnica de *Ambient Occlusion* como parte del cálculo de la iluminación global, mientras que otros no lo hacen. La técnica de *Ambient Occlusion* por sí sólo genera iluminación menos realista que la iluminación global, sin embargo, es mucho más rápida y menos compleja en término de cálculos a ejecutar que otros métodos por lo que sigue siendo muy popular entre los desarrolladores de videojuegos, animaciones de producción e incluso películas.

Por una parte el término "*Ambient*" se refiere a la luz indirecta en la escena. La luz ambiental es la luz que no proviene de una fuente que puede ser identificada. Es la luz que se acumula a partir de fotones extraviados que rebotan por toda una escena. Por otra parte, el término "*Occlusion*" se refiere a cuando un objeto es bloqueado por otro que pasa entre el observador y este. La oclusión puede ser útil para otras cosas. Por ejemplo, se puede optimizar una escena y renderizarla más rápido mediante el sacrificio de los objetos que se encuentren ocluidos o escondidos por otros objetos.

Ambient Occlusion es una medida de la cantidad de luz ambiental que está bloqueada por elementos cercanos [14]. Si se ocluye un objeto o superficie, menos luz puede llegar a ella, lo que significa que la superficie estará en la sombra. Las arrugas, las esquinas y huecos tienden a tener una gran cantidad de oclusión, mejorando el renderizado haciendo pequeños detalles más visibles y mejora la percepción de profundidad. Gran parte de la forma que los humanos perciben la profundidad tiene que ver con el tamaño, la oscuridad y la ubicación de las sombras. El uso de *Ambient Occlusion* ayuda al espectador a comprender mejor la ubicación de los elementos de una escena en un momento dado.

Hay muchos algoritmos para el cálculo de *Ambient Occlusion* y cada uno tiene sus ventajas y desventajas, donde también influyen las características del motor gráfico. El método de Monte Carlo es el algoritmo más usado debido a la interpretación de los rayos con las superficies de manera satisfactoria.

Los fotones son lanzados desde la fuente e inciden en las superficies y desde ellas hacia el resto de la escena, donde finaliza la ejecución, es decir, sólo 1 rebote, tal como muestra la Figura 7.

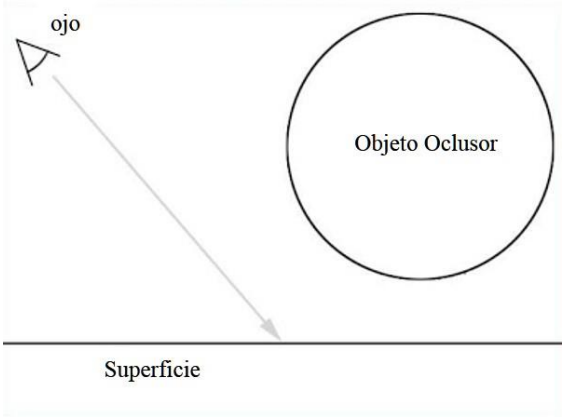


Figura 7: Rayo lanzado desde la fuente a la superficie.

En la Figura 8 se observa como la dispersión de los fotones es hemisférica, aleatoria e interactúa con las normales de los objetos.

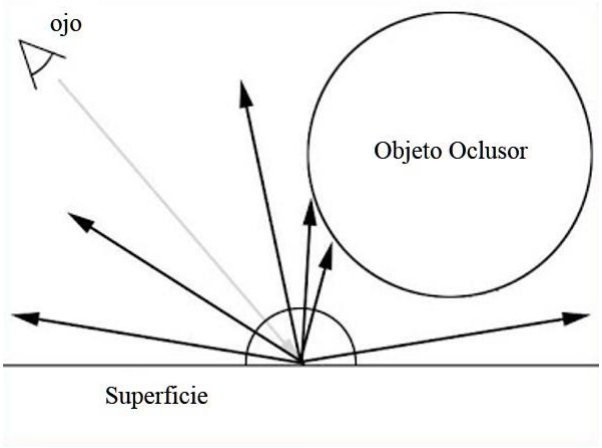


Figura 8: Rayos aleatorios lanzados desde el hemisferio.

Finalmente cada punto de la superficie es sombreada por una proporción de rayos que intersectan objetos de la escena, en la Figura 9 se observan 2 intersecciones para 6 muestreos, obteniendo una relación de 1/3. Sustrayendo esta relación 1 obtendremos zonas oscuras en la porción ocluida de las superficies. El proceso se repite para cada objeto en la escena.

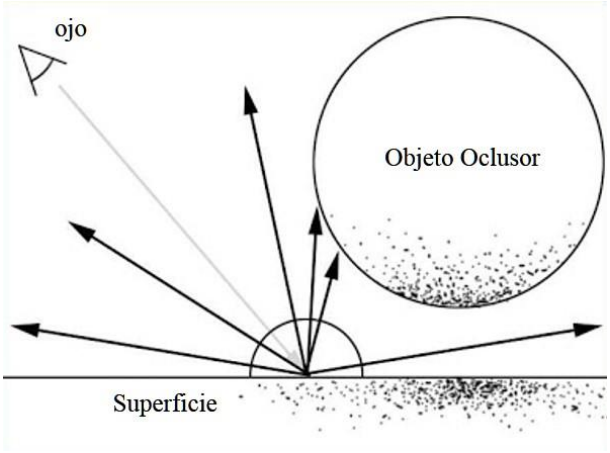


Figura 9: Proporción de oclusión para los objetos de la escena,
6 muestras en total de las cuales 2 intersectan con el objeto oclusor.

Esta técnica debe estar acompañada de otros pasos de post-producción como la difusión de entorno, color ambiental, entre otros.

Con suficientes muestras por punto, este método puede producir muy buenos resultados. La desventaja es que este método puede ser muy lento. Muchos de los videojuegos que utilizan este método no calculan *ambient occlusion* en tiempo real, sino que se calcula al cargar la escena o fuera de línea al construir la escena almacenando los resultados en un mapa de textura.

2.2.1 Origen y Evolución

La Empresa ILM (Industrial Light & Magic) desarrolló en 2001 la técnica de *Ambient Occlusion*, en un esfuerzo por producir un mayor nivel de realismo de las imágenes generadas por computador en la película Pearl Harbor que nunca se había producido antes. Inspirados en la técnica de reflejo de oclusión utilizada en la película Máxima Velocidad 2 buscaron desarrollar una técnica donde podrían hacer algo similar para Pearl Harbor, pero sin reflexiones y con iluminación ambiental [15]. Al proporcionar sombra precisa e información de luz direccional, la técnica de *Ambient Occlusion* permite crear efectos de iluminación realistas y eficientes. Desde su desarrollo, la técnica ha sido ampliamente utilizada en el campo de computación gráfica. En la Figura 10 se aprecia la primera imagen usada para probar la técnica de *Ambient Occlusion.*

Figura 10: Primera imagen utilizada para probar *Ambient Occlusion*.

2.2.1.1 Screen Space Ambient Occlusion (SSAO)

Es una técnica de renderizado para eficientemente aproximar el efecto de *Ambient Occlusion* en tiempo real, fue desarrollado por Vladimir Kajalin mientras trabajaba en Crytek siendo usada por primera vez en 2007 en el videojuego Crysis. Funciona examinando los píxeles, y comparando su ubicación en el *depth buffer* con los píxeles a su alrededor. Los píxeles cercanos son sombreados para simular sombras suaves. Para poder usar esta técnica en tiempo real no es posible muestrear cada píxel cada *frame*, así que se toman muestras aleatorias [1] como se muestra en la Figura 11, lo cual sin duda genera ruido que puede ser más notable con el movimiento, para resolver esto se pasa a una técnica de post-procesamiento donde se difumina el resultado.

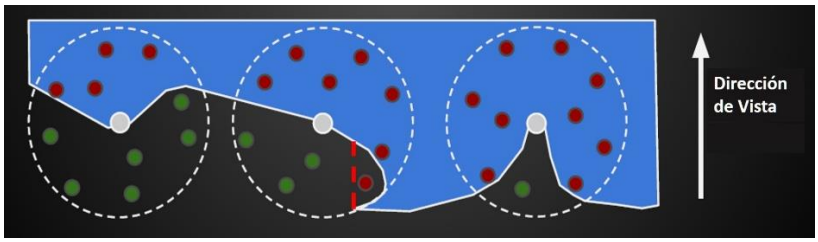


Figura 11: Muestreo Aleatorio en Depth Buffer.

La reducción del ruido se obtiene aplicando un *Gaussian Blur*, el cual es resultado de empañar una imagen por una función Gaussiana, en la Figura 12 se puede apreciar el resultado de esta técnica. Es un efecto ampliamente utilizado en computación gráfica, típicamente para reducir el ruido de la imagen y reducir detalles.

Figura 12: Resultado de aplicar Gaussian Blur a una imagen.

La técnica de SSAO es completamente independiente del sistema de iluminación, es decir, que la apariencia y desempeño del shader no son afectados por las luces usadas en la escena. SSAO es dependiente de la vista debido a que el *depth buffer* se mide desde la cámara, es decir, que la ubicación de las sombras creadas por SSAO puede cambiar al mover la cámara alrededor. Puede ser calculado completamente en la GPU. Mientras más alta es la resolución, más cálculos deben ser realizados, incluso pequeños cambios en la resolución pueden traer grandes problemas.

La Figura 13 muestra un ejemplo donde se puede contrastar el resultado de una escena sin SSAO y con SSAO respectivamente.

Figura 13: Resultado de escenas con SSAO activado o desactivado.

2.2.1.2 Horizon Based Ambient Occlusion (HBAO)

En la conferencia de tecnología SIGGRAPH 2008 NVIDIA introdujo una mejorada variante de SSAO llamada *Horizon Based Ambient Occlusion* (HBAO), HBAO emplea algoritmos de base física que aproximan una integral con muestreo de *depth buffer*. La actualización permite HBAO generar SSAO de alta calidad, mientras incrementa el

número de muestras por píxel, incrementa la definición, calidad, y visibilidad del sombreado del *Ambient Occlusion* [2].

Para encontrar el horizonte se muestrean en diferentes direcciones alrededor del píxel de interés y se toman puntos de muestra alrededor, a través de cada dirección generada, en la Figura 14 se observa cómo a partir de un píxel de interés se generan 4 direcciones.

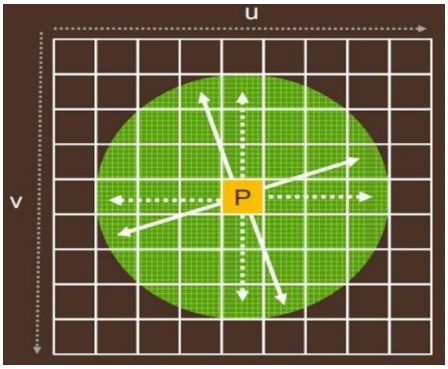


Figura 14: Ejemplo con cuatro direcciones generadas a partir de píxel de interés.

En cada dirección generada se busca la máxima altura dentro del radio de muestreo, tal como se muestra en la Figura 15, donde se observa cómo a través del vector se explora la geometría, al encontrar una intersección se eleva el ángulo de búsqueda y se continúa explorando hasta que se llega al límite definido por el radio. Una vez se alcanza el límite se registra la máxima altura encontrada como el horizonte, el cual será utilizado para definir la cantidad de oclusión del píxel de interés.

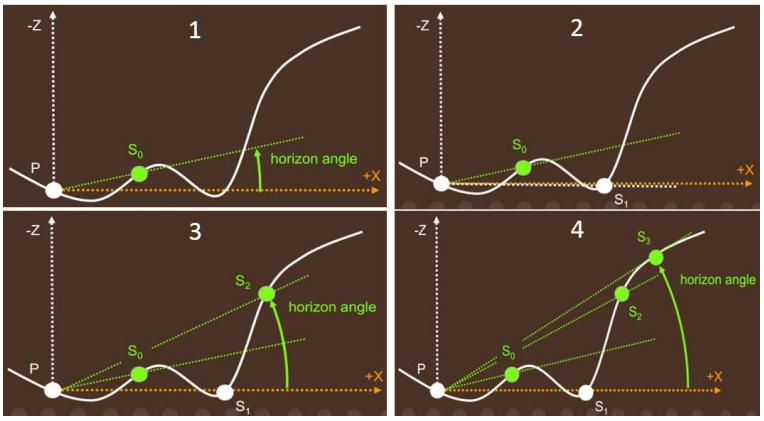


Figura 15: Búsqueda del Horizonte para la técnica HBAO.

Cada horizonte encontrado en cada dirección generada aporta a la proporción de oclusión correspondiente al píxel de interés dependiendo de la altura.

Por razones de rendimiento, HBAO es usualmente renderizado a media resolución lo que inevitablemente causa un parpadeo que es difícil ocultar en cada situación, para resolver este problema Louis Bavoil de NVIDIA ha reconstruido y modernizado HBAO para crear HBAO+. El resultado del trabajo de Louis es una técnica de *Ambient Occlusion* más precisa, con sombras de peso que están mejor definidas y son más visibles, mejora el nivel de detalle al doble, es tres veces más rápido y utiliza DirectX 11 [2].

Capítulo 3

Ambient Occlusion en Tiempo Real

Este capítulo se enfoca en cuatro técnicas recientes inspiradas a partir de *screen-space Ambient Occlusion* (SSAO), *Alchemy Screen-Space Ambient Obscurance, Scalable Ambient Obscurance, Multi-Scale Ambient Occlusion* y *Unreal Engine 4 Ambient Occlusion*. Las cuales buscan optimizar el rendimiento o mejorar la calidad de los resultados obtenidos. Siempre procurando obtener resultados lo más cercanos posible a los obtenidos con iluminación global, pero que pueden ejecutarse en tiempo real. Se explicará cada una de las técnicas e ideas propuestas.

3.1 El algoritmo Alchemy Screen-Space Ambient Obscurance

Produce efectos importantes de iluminación tales como esquinas oscuras, grietas, arrugas y oscuridad por proximidad. El método se basa en una variación de SSAO que utiliza una función de atenuación de luz que simplifica las operaciones. *Alchemy AO* crea sombras de contacto que conforman las superficies, captura la oclusión generada por la geometría y provee *temporal coherence* lo cual junto a su función de atenuación de luz cancela operaciones costosas, logrando así ser eficiente. Para estimar la oclusión en cada píxel se utiliza tanto el *depth* como el *normal buffer*. *Alchemy AO* se desarrolló para un videojuego en específico, *Guitar Hero,* para escenas a resolución HD720p en Xbox 360 [3].

La Figura 16 muestra el impacto visual de *Alchemy AO*. La imagen de la izquierda muestra una escena con sólo luz ambiental. La imagen de la derecha mezcla esa luz ambiental con *Alchemy AO*, el cual resuelve los detalles y relación espacial entre los objetos.

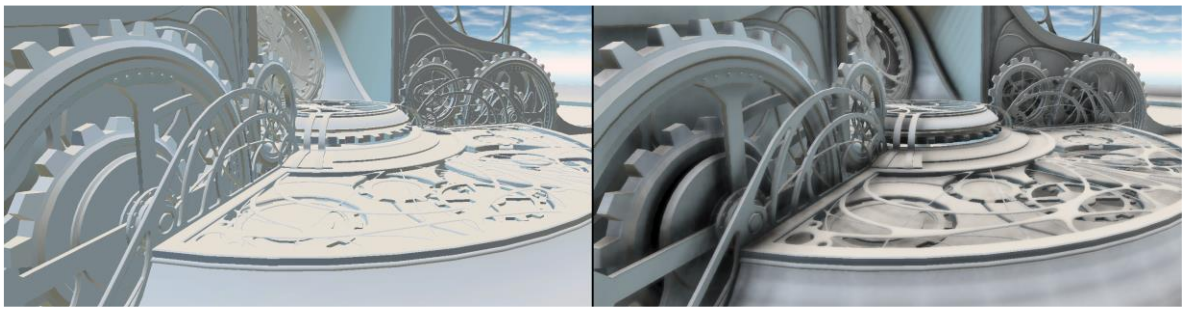


Figura 16: La imagen de la izquierda con iluminación simple y la imagen de la derecha con Alchemy AO.

El algoritmo se basa en tres ideas: derivar un estimador robusto desde la ecuación de *render;* proveer *temporal coherence* que consiste en reutilizar información de frames anteriores y combinarlas con el frame actual al hacer el estimador lo suficientemente

eficiente como para evaluar varias veces por píxel; y logra esa eficiencia al moldear la función de atenuación de luz para cancelar operaciones costosas. *Alchemy AO* aborda los inconvenientes de métodos anteriores de SSAO tales como: ajustar la oclusión a las superficies afectadas, es decir, que no existan sombras flotando en el aire cerca de las siluetas, además de capturar fenómenos a múltiples escalas, tales como sombras en pozos profundos, oscurecimiento de esquinas, sombras de contacto y arrugas [3].

Como todos los métodos en espacio de pantalla, tiene limitaciones en la cantidad de muestras y errores de oclusión debido a oclusores fuera de pantalla y detrás de la superficie del *depth buffer*. *Alchemy AO* es estable bajo un largo rango de parámetros y puede aumentar el rendimiento de tarjetas gráficas específicas al ajustar el número de muestra y el radio de muestreo [3].

3.2 Scalable Ambient Obscurance

Se presenta una variante de *Alchemy AO* donde se mantienen los cálculos principales pero se evalúan a través de un nuevo algoritmo que mejora *Alchemy AO* de dos maneras. Primero, el nuevo algoritmo SAO requiere de entrada solamente el *depth buffer*, a diferencia de *Alchemy AO* que necesita también el *normal buffer*. Segundo, en vez de utilizar una resolución fija de 1280x720 píxeles, el algoritmo asume un hardware moderno y aumenta a altas resoluciones y el radio de muestreo en el espacio mundo. De esta forma se busca atacar conocidas limitaciones de métodos SSAO anteriores, donde la eficiencia se ve afectada cuando se toman muestras lejanas al píxel de interés [4]. En la Figura 17 se muestra una comparación entre los resultados de aplicar *Alchemy AO* y *Scalable Ambient Obscurance*.

Figura 17: La imagen de la izquierda con el método Alchemy AO y la imagen de la derecha con Scalable Ambient Obscurance (SAO).

Scalable Ambient Obscurance mejora el rendimiento a una resolución de 2560x1600 píxeles, generaliza el algoritmo para renderizadores hacia adelante y diferidos (*forward/deferred renderers*) y elimina la dependencia del radio de *Alchemy AO*. Las optimizaciones se construyen sobre tres estrategias: pre-filtrar el *depth buffer* para maximizar la eficiencia de

la memoria; reduce el total ancho de banda al reconstruir cuidadosamente posiciones y normales a alta precisión desde un *depth buffer*; y explota técnicas de paralelismo [4].

Una limitación que *SAO* comparte con las otras técnicas SSAO es la necesidad de una banda de protección en el *viewport* de tal forma que la geometría fuera de pantalla contribuye oclusión. Esto aumenta la necesidad de memoria aunque el impacto en el rendimiento es poco [4].

En resumen *Scalable Ambient Obscurance* tiene como base la ecuación utilizada en Alchemy AO, pero le hacen algunas optimizaciones con el objetivo de escalar a altas resoluciones además de aumentar el tamaño del radio de muestreo. La primera mejora consiste en reducir el ancho de banda de memoria necesaria debido a que solo necesita el *depth buffer* con el cual se derivan la posición y normales de los objetos de la escena. Otra mejora es el uso de mipmap sobre el *depth buffer* mejorando el rendimiento del muestreo, lo que permite al algoritmo escalar a amplios radios, logrando una estimación de oclusión más precisa. *Ambient Occlusion* siempre es calculado a la máxima resolución, aunque use mipmap en el *depth buffer,* y por último realizan una ligera optimización en el cálculo de la función de atenuación con la que reducen el contraste en esquinas muy agudas.

3.3 Efficient Screen-Space Approach to High-Quality Multi-Scale Ambient Occlusion

Es una variante de *Screen-Space Ambient Occlusion* (SSAO) que mejora tanto el rendimiento como la calidad de los resultados. Este método calcula los valores de la técnica de *Ambient Occlusion* a múltiples resoluciones de imagen las cuales combina para obtener un resultado final de alta resolución para cada píxel. Se produce AO de alta calidad que incluye sombras de alta frecuencia debido a geometría cercana que ocluyen y sombras de baja frecuencia debido a geometría lejana. Este método solo necesita utilizar pocos núcleos de muestra en cada resolución, por lo que se consigue lograr alto rendimiento sin recurrir a muestreo aleatorio. Como consecuencia el resultado obtenido no sufre de ruido ni difuminación excesiva, lo cual es común en otras variaciones de SSAO. El método produce resultados más cercanos a soluciones que utilizan *ray tracing* mientras se ejecuta a tasas de *frames* similares o superiores que otras variaciones de SSAO. En la Figura 18 se observa el resultado de este método en comparación con la misma escena sin *Ambient Occlusion* [5].

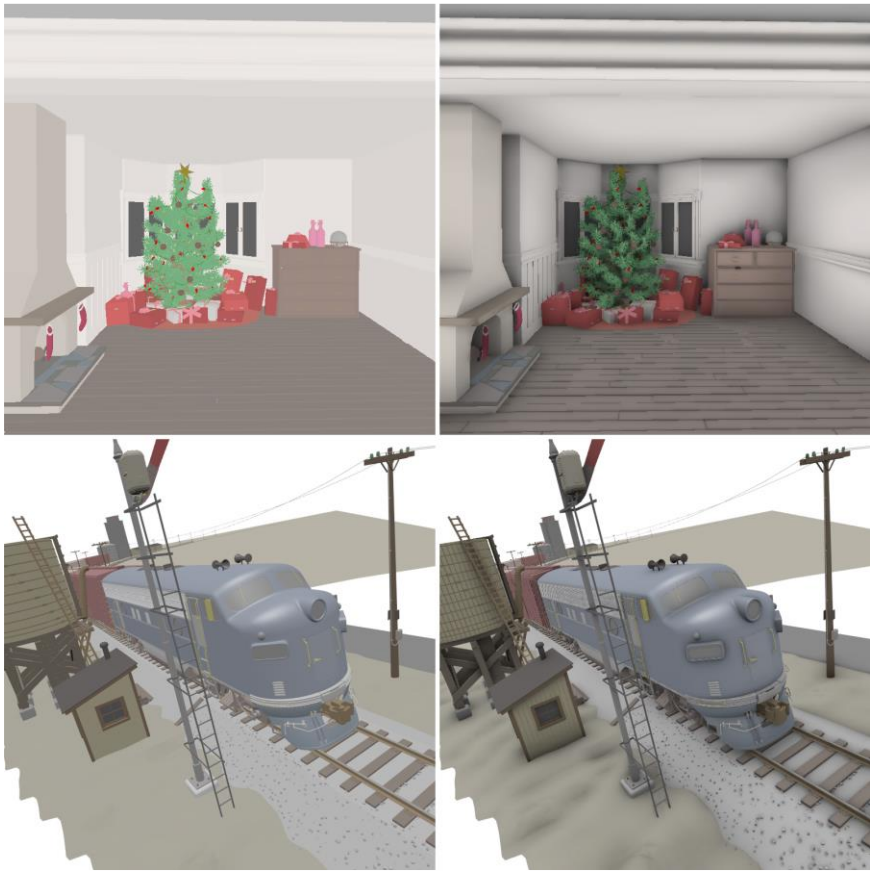


Figura 18: Del lado izquierdo escenas sin AO y del lado derecho la misma escena con este método de AO.

Una de las limitaciones de este método es que requiere más memoria que otros métodos SSAO existente. Cada nivel de resolución usa cuatro *buffers*, de posición, normales, *Ambient Occlusion* y *blur*. Los efectos de este incremento de memoria se pueden notar más cuando es usado en un contexto más realista, por ejemplo, un videojuego, donde la memoria es compartida con otras etapas de iluminación y sombreado. Otro problema es la baja cantidad de *temporal coherence* sobre geometría delgada tales como patas de sillas u hojas de árboles [5].

La Figura 19 muestra la combinación de valores AO a través de múltiples resoluciones para conseguir el resultado final. Las primeras 5 imágenes, de izquierda a derecha, de arriba hacia abajo, muestran el resultado de AO en cada resolución por separado y la última imagen abajo a la derecha muestra el resultado final al combinar todas las imágenes anteriores.

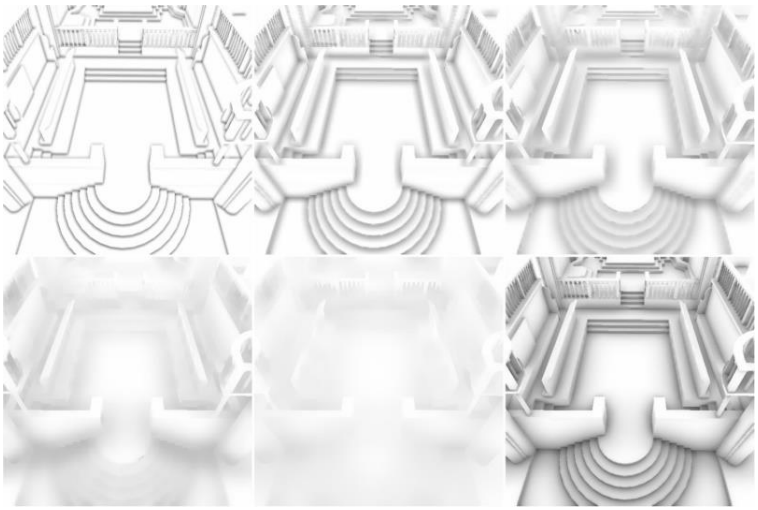


Figura 19: AO a diferentes resoluciones, al combinarlas se obtiene el resultado final de la esquina inferior derecha.

Debido a que esta técnica se enfoca en aplicar el cálculo de *Ambient Occlusion* en diferentes resoluciones de manera eficiente, ofrece la libertad de escoger qué técnica de A*mbient Occlusion* utilizar, es decir, así como se puede utilizar la técnica SSAO original se puede utilizar HBAO, Alchemy AO, Scalable AO o cualquier otra técnica similar que necesite primero la obtención de *depth buffer* para realizar el cálculo de oclusión en una escena.

3.4 Unreal Engine 4 Ambient Occlusion

Es un método inspirado en *Horizon-Based Ambient Occlusion* (HBAO) la diferencia es que las direcciones generadas que buscan el mayor oclusor se generan en pares y la oclusión acumulada se calcula a partir del ángulo entre cada par de vectores. Este método no solo usa *depth buffer* como HBAO sino que también utiliza las normales del *GBuffer* el cual es producto de la técnica *Deferred Shading*. Debido a esto los detalles del mapa de normales afecta los resultados mejorando la calidad. En la Figura 20 se observa el resultado de *Unreal Engine AO* en comparación con el método SSAO [6].

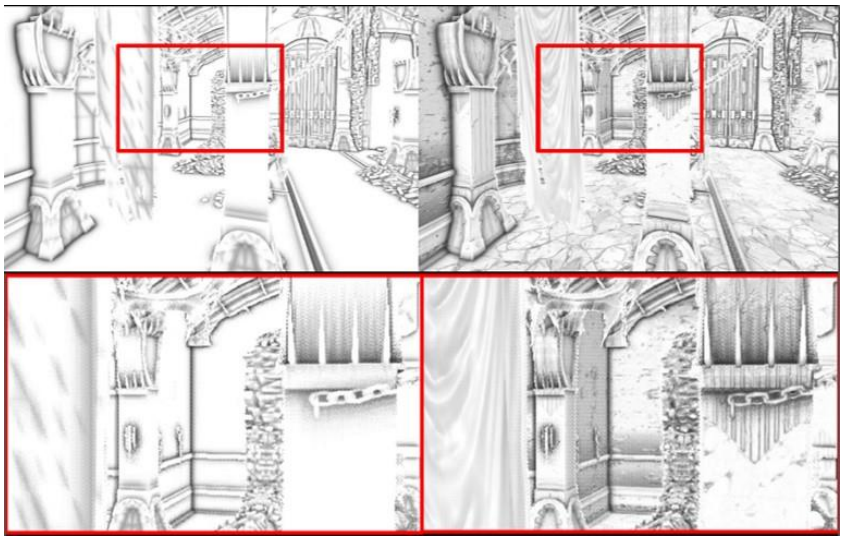


Figura 20: En la izquierda con el método SSAO y en la Derecha con Unreal Engine AO.

Capítulo 4

Desarrollo e Implementación

En este capítulo se expondrán las herramientas de hardware y software utilizadas para la implementación. Se explica cómo se realizó la implementación de las diferentes técnicas de *Ambient Occlusion* en tiempo real inspiradas a partir de la técnica *Screen-Spaced Ambient Occlusion* (SSAO), además de los pasos previos y posteriores a las mismas necesarios para el despliegue de la escena, para cada etapa del desarrollo se expondrán algunos extractos del código fuente donde suceden las características de mayor importancia de cada técnica.

4.1 Componente de hardware y software

El hardware utilizado para su desarrollo y ejecución consistió en una PC que incluye una tarjeta gráfica con soporte GLSL versión 4.50. Los componentes de la PC son los siguientes:

- Intel Core i7 4790K 4.00GHz
- 16 GB de memoria RAM.
- NVIDIA Geforce GTX 970, 4 GB de memoria.

También se utilizó el siguiente software para el desarrollo:

- Windows 10 como sistema operativo.
- Microsoft Visual Studio 2017.
- Ant Tweak Bar en su versión 1.16, para la interfaz gráfica de la aplicación. [13]
- OpenGL en su versión 4.50.

4.2 Esquema Propuesto para la solución

Para lograr el desarrollo de las diferentes técnicas de *Ambient Occlusion* en tiempo real se siguen los siguientes pasos:

1. Cálculo del depth buffer (Primer Paso *Deferred Shading*).
2. Aplicación de Ambient Occlusion (Segundo Paso *Deferred Shading*).
3. *Gaussian Blur.*
4. *Blending* final.

Todas las técnicas *Ambient Occlusion* seleccionadas realizan los mismos pasos donde solo varían en el paso número dos, donde ocurre la aplicación de técnica de *Ambient Occlusion* propuesto por cada una de las técnicas a estudiar. Antes de realizar cada paso es necesario desplegar todos los objetos de la escena incluso los que no se les calculará oclusión. El primer paso consiste en generar el *depth buffer* el cual es vital para aplicar cada técnica, en el segundo paso se realiza la aplicación de cada técnica, el cual debido al ruido con que se obtiene cada resultado es necesario realizar un *blur* para reducirlo. De esta forma, se obtiene mejores resultados visuales. Para finalizar todas despliegan los resultados finales en un plano que cubre toda la pantalla el cual se mezcla con el resto de la escena realizando un *alpha blending*. Cada uno de estos pasos se explica en mayor detalle a lo largo del capítulo.

4.2.1 Cálculo del depth buffer

El primer paso para calcular el *depth buffer* es construir un *framebuffer* al cual se le asignan dos texturas: una para guardar las normales de la escena y otra para la profundidad (*depth*). Primero se renderizan los objetos de la escena para luego pasar al primer programa *shader* a utilizar en la aplicación de la técnica de *Ambient Occlusion*, donde se toman las posiciones y normales por vértice de los objetos, y desde el *fragment shader* se guardan en las texturas asignadas previamente en el *framebuffer*.

En la Figura 21 se muestra el código del *vertex shader* del primer programa *shader* donde le envía al *fragment shader* las normales de los objetos desde el punto de vista.

```
#version 450

layout(location = 0) in vec3 vVertex;   //object space vertex
layout(location = 1) in vec3 vNormal;   //object space normal

//uniforms
uniform mat4 MVP;               //combined modelview projection matrix
uniform mat3 NormalMatrix;      //normal matrix

smooth out vec3 vEyeSpaceNormal;   //output eye space normal

void main()
{
    //get eye space normal by multiplying the object space normal
    //with the normal matrix
    vEyeSpaceNormal = NormalMatrix*vNormal;

    //get the clipspace position
    gl_Position = MVP*vec4(vVertex,1);
}
```

Figura 21: Vertex Shader del Primer Paso.

En la Figura 22 se muestra como el *fragment shader* retorna las normales interpoladas.

```
#version 450

layout(location=0) out vec4 vFragColor; //fragment shader output

smooth in vec3 vEyeSpaceNormal;         //eye space normal from the vertex shader

void main()
{
    //output the eye space normal as colour, bring it in 0-1 range
    vFragColor = vec4(normalize(vEyeSpaceNormal)*0.5 + 0.5, 1);
}
```

Figura 22: Fragment Shader del Primer Paso.

4.2.2 Aplicación de Ambient Occlusion

Además del *framebuffer* creado para almacenar la información de normales y profundidad, es necesario crear un segundo buffer donde se almacena el resultado de cada técnica de *Ambient Occlusion* basada en la técnica de SSAO. Este *framebuffer* también es utilizado para aplicar la difuminación (*gaussian blur*) necesaria para mejorar el resultado final de la técnica de *Ambient Occlusion*. Después del primer paso se enlaza el segundo *framebuffer* el cual usa las texturas de normales y profundidad (*depth)* para calcular el resultado de *Ambient Occlusion*.

Para el segundo paso donde se aplica la técnica de *Ambient Occlusion*, se utiliza un segundo programa *shader* donde el *vertex shader* no realiza ningún tipo de modificación y pasa directamente al *fragment shader* donde se aplica cada técnica para realizar los cálculos correspondientes, debido a las similitudes entre las mismas, todas las técnicas de estudio en este trabajo utilizan el mismo *vertex shader*, las diferencias se realizan dentro del correspondiente *fragment shader*. En la Figura 23 se muestra el *vertex shader*.

```
#version 450

layout(location=0) in vec2 vVertex;     //object space vertex position

smooth out vec2 vUV;                    //interpolated texture coordinate

void main()
{
    //get clipspace position from the object space vertex position
    gl_Position = vec4(vVertex*2-1.0,0,1);

    //pass object space vertex position as the texture coordinate
    vUV = vVertex;
}
```

Figura 23: Vertex Shader sin modificar geometría.

A continuación se explicará con más detalle cada técnica de *Ambient Occlusion* seleccionada.

4.2.2.1 Alchemy Screen-Space Ambient Obscurance

Para aplicar la técnica de *Alchemy AO*, primero se selecciona un punto **Q** en el espacio de imagen cerca de la proyección de **C** que es nuestro punto de interés, se lee el *depth buffer* el cual da la ubicación del punto **P** que podría ocluir al punto **C**, el vector **V** que va desde **C** a **P** define el factor de área de proyección y distancia de atenuación, finalmente se muestrean varios puntos de esta misma forma acumulando el estimado de oclusión dentro del hemisferio. Se puede observar en la Figura 24 como se realiza el muestreo.

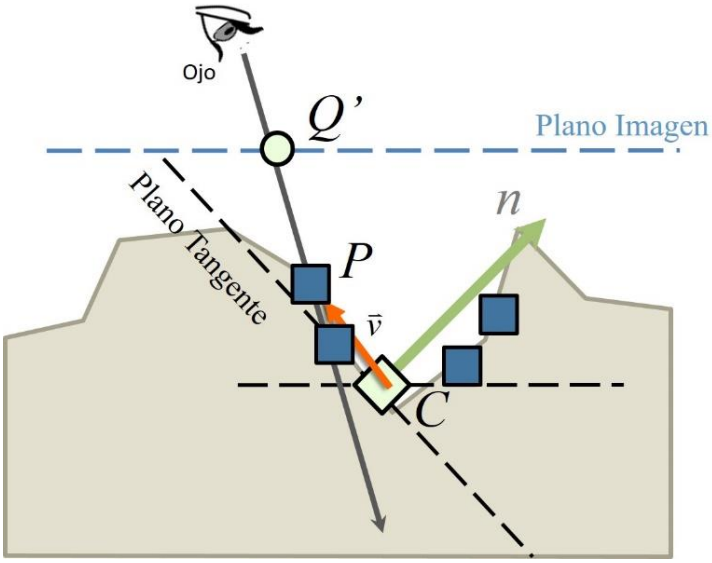


Figura 24: Esquema 2D de un objeto al cual se estima la oclusión para el punto C dentro del valle.

Por otra parte en la Figura 25 se presenta la aproximación de oclusión que utiliza Alchemy AO donde **vi** es el vector que va desde el punto de interés a cada muestra aleatoria, $\boldsymbol{\sigma}$ es el multiplicador de fuerza, es decir, si es alto produce sombras más oscuras y si es bajo produce sombras más claras, **β** es el sesgo de sombra (bias) si es muy alto contrarresta la oclusión, y si es muy bajo crea sombra a sí mismo, **s** es la cantidad de muestras por píxel, **z** es usado para compensar los productos punto que al incrementar la cantidad de muestras se vuelven sensibles a errores. Cabe destacar que la cantidad de muestras recomendadas por los desarrolladores [3] es entre 4 y 16, mientras más muestras mayor es la precisión, pero a menor muestras mejor rendimiento, **k** es el multiplicador de contraste donde a mayor valor la transición de oscuridad es más nítida y a menor valor las sombras son más borrosas y por último **ε** se utiliza para evadir la división entre cero. En la Figura 26 se muestra un extracto de código del *fragment shader* para aplicar la técnica de *AlchemyAO* donde ocurre el proceso

de muestreo para cada muestra tomada alrededor del punto de interés además de la función de atenuación aplicada.

$$A \approx \max\left(0, 1 - \frac{2\sigma}{s} \cdot \sum_{i=1}^{s} \frac{\max(0, \vec{v_i} \cdot \hat{n} + z_C \beta)}{\vec{v_i} \cdot \vec{v_i} + \epsilon}\right)^k$$

Figura 25: Aproximación para realizar proceso de muestreo en espacio de pantalla.

```
for
{
    //
    float ssR;
    vec2 aoUnitOffset = tapLocation(i, randPatternRotAngle, ssR);
    ssR *= uvDiskRadius;

    //the occluding point in camera space
    vec2 aoTexS = aoTexCoord + ssR*aoUnitOffset;
    vec3 aoQ = GetViewPos(aoTexS);

    //calculate the occlusion for the pixel of interes (aoC) for each sample
    vec3 aoV = aoQ - aoC;
    float aoVv = dot(aoV,aoV);
    float aoVn = dot(aoV, aoN_C);
    float aoEp = 0.01;
    float aoF = max(aoRad*aoRad - aoVv, 0.0);

    //falloff function
    ao += aoF*aoF*aoF*max((aoVn - aoBias)/(aoEp+aoVv),0.0);

}
```

Figura 26: Fragment Shader Alchemy AO.

4.2.2.2 Scalable Ambient Obscurance

La técnica de *Scalable Ambient Occlusion* está basada en *AlchemyAO* [4], pero con las siguientes optimizaciones:

1. Reduce ancho de banda al utilizar sólo el *depth buffer* a partir del cual deriva las normales.

   El primer paso en SAO es calcular con alta precisión la muestra tomada del *depth buffer* destacando la importancia que tiene dicha muestra para el resto del proceso, debido a que a partir de ella se derivan el resto de los valores. Luego se realiza una reconstrucción de las normales a partir de la posición. En la Figura 27 se muestra en código las funciones que realizan este paso.

```
//  Clipping plane constants for use by reconstructZ
//  clipInfo = (z_f == -inf()) ? Vector3(z_n, -1.0f, 1.0f) :
//                               Vector3(z_n * z_f,  z_n - z_f,  z_f);
float reconstructCSZ(float depth) {
//  return clipInfo[0] / (clipInfo[1] * d + zFar[2]);
    return ( zNear * zFar ) / ( (zNear - zFar) * depth + zFar );
}

// World space point being shaded
vec3 reconstructCSPosition(vec2 S, float z) {
    return vec3((S.xy * projInfo.xy + projInfo.zw) * z, z);
}

//Reconstructs screen-space unit normal from screen-space position
vec3 reconstructCSFaceNormal(vec3 C) {
    return normalize(cross(dFdx(C), dFdy(C)));
}

void main() {
    ......
    float depthCS = reconstructCSZ( zDepth );

    vec3 originVS = reconstructCSPosition( vUV, depthCS);

    vec3 normalVS = normalize( reconstructCSFaceNormal(originVS) );
    ......
}
```

Figura 27: Cálculo de alta precisión de muestra y reconstrucción de normales.

2. Inclusión de Mipmapping para mejorar el rendimiento del muestreo, permitiendo al algoritmo escalar a amplios radios logrando una estimación de oclusión más precisa.

```
//  compute the occlusion to sample with index i about the pixel at uv
//  that corresponds to camera-space point positionVS with unit normal normalVS
//  using maximum screen-space sampling radius sampleRadiusSS
float sampleAO(vec2 uv, vec3 positionVS, vec3 normalVS, float sampleRadiusSS,
               int tapIndex, float rotationAngle)
{
    const float epsilon = 0.01;
    float radius2 = sampleRadiusWS * sampleRadiusWS;
    // offset on the unit disk, spun for this pixel
    float radiusSS;
    vec2 unitOffset = tapLocation(tapIndex, rotationAngle, radiusSS);
    radiusSS *= sampleRadiusSS;
    // The occluding point in camera space
    vec3 Q = getOffsetPositionVS(uv, unitOffset, radiusSS);
    vec3 v = Q - positionVS;
    float vv = dot(v, v);
    float vn = dot(v, normalVS);
    //falloff function
    // Smoother transition to zero (lowers contrast, smoothing out corners).
    float f = max(radius2 - vv, 0.0);
    return f * f * f * max((vn - bias) / (epsilon + vv), 0.0);
}
void main() {
    ......
    for (int i = 0; i < NUM_SAMPLES; ++i) {
            occlusion += sampleAO(aoTexCoord, originVS, normalVS,
                                  radiusSS, i, randomPatternRotationAngle);
    }
    ......
}
```

Figura 28: Mipmapping en la técnica Scalable AO.

En la Figura 28 se observa el cálculo de *Ambient Occlusion* por cada muestra, la inclusión de mipmapping ocurre dentro de la función *getOffsetPositionVS* la cual se observa en la Figura 29.

```
//  Read the camera-space position of the point at screen-space pixel
//  ssP + unitOffset * ssR.  Assumes length(unitOffset) == 1
vec3 getOffsetPosition(ivec2 ssC, vec2 unitOffset, float ssR) {
    // Derivation:
    //  mipLevel = floor(log(ssR / MAX_OFFSET));
    int mipLevel = clamp(int(floor(log2(ssR))) - LOG_MAX_OFFSET, 0, MAX_MIP_LEVEL);
    ivec2 ssP = ivec2(ssR * unitOffset) + ssC;
    vec3 P;
    // We need to divide by 2^mipLevel to read the appropriately scaled coordinate from a MIP-map.
    // Manually clamp to the texture size because texelFetch bypasses the texture unit
    ivec2 mipP = clamp(ssP >> mipLevel, ivec2(0),
                       textureSize(CS_Z_buffer, mipLevel) - ivec2(1));
    P.z = texelFetch(CS_Z_buffer, mipP, mipLevel).r;
    // Offset to pixel center
    P = reconstructCSPosition(vec2(ssP) + vec2(0.5), P.z);
    return P;
}
```

Figura 29: Aplicación de la técnica de Scalable AO.

Una de las observaciones expuestas por los desarrolladores es que debe haber entre 3 y 5 niveles de mipmapping, si hay menos de 3 es posible que ocurra un parpadeo y si es mayor a 5 se obtiene mal rendimiento debido a que no se está usando mipmapping efectivamente.

3. Función de atenuación que disminuye contraste y suaviza las esquinas, el código se realiza como último paso antes de retornar el resultado de cada muestra de oclusión.

4.2.2.3 Multi-Scale Ambient Occlusion

De las diferentes técnicas inspiradas en *Screen-Spaced Ambient Occlusion* (SSAO) algunas se enfocan en mejorar el rendimiento y otras en la calidad visual de los resultados, pero todas cumplen con la capacidad de ser utilizadas en aplicaciones de tiempo real. Multi-Scale Ambien Occlusion (MSSAO) es del tipo de técnica que se enfoca en la calidad visual del resultado sacrificando un poco el rendimiento.

MSSAO no consiste en un método de cálculo para obtener *Ambient Occlusion* como otras técnicas, de hecho, se puede aplicar a todas las técnicas similares a SSAO. Los desarrolladores de MSSAO [5] se enfocan en explicar las mejoras que ofrece esta técnica sin especificar qué técnica de SSAO es utilizada.

En este trabajo, en vista de la libertad que ofrece MSSAO se utiliza como técnica de *Ambient Occlusion* la técnica de *Alchemy AO*, sin embargo, cualquiera de las otras dos técnicas pueden ser utilizadas, es decir, *Scalable AO* o *Unreal Engine 4 AO*. Se utilizó cinco resoluciones diferentes tal como se recomienda en el trabajo de Hoang y Low [5], la resolución máxima es 1920x1080 píxeles de ahí las siguientes vienen a ser divisiones

entre dos, es decir, 960x540, 480x270, 240x137 y 120x68 píxeles. Se aplica *Ambient Occlusion* para cada una de estas resoluciones generando cinco diferentes texturas con su respectivo *blur*, las cuales como último paso se mezclan utilizando el programa shader que realiza el blending final que se explica más adelante en este capítulo. La aplicación realizada para este trabajo permite definir la intensidad de cada una de estas texturas, con la finalidad de observar el resultado individualmente o combinarlos a gusto de cada usuario.

Con respecto al código, se debe recordar que lo importante de esta técnica no ocurre dentro de un programa shader sino dentro del código C++. Debido a que aplica *Ambient Occlusion* en cinco diferentes resoluciones necesita de cinco diferentes *framebuffers* con sus respectivas texturas de normales, de profundidad (*depth*) y la que mostrará el resultado final de *Ambient Occlusion* a sus respectivas resoluciones. Por otra parte dentro de la función de *render* MSSAO se ejecuta dentro de un ciclo donde se aplica *Ambient Occlusion* para cada resolución definida, es importante establecer el *framebuffer* y el *viewport* correspondiente a cada resolución tal como se muestra en la Figura 30.

```
for (int i = 0; i < LEVEL_COUNT; i++)       //LEVEL_COUNT == 5
{
    ......
    //bind the FBO
        glBindFramebuffer(GL_FRAMEBUFFER, frameBufs[i]);
        //set the viewport to the size of the offscreen render target
        glViewport(0,0,RTT_WIDTH/(pow(2,i)),RTT_HEIGHT/(pow(2,i)));
        //Shader Primer Paso
        SSAOPrimerPaso();
        //bind the second step SSAO shader
        AlchemyAO->Enable();
            //set shader uniforms
            ......
        //unbind the second step SSAO shader
        AlchemyAO->Disable();
    ......
}
```

Figura 30: Resumen técnica MSSAO.

4.2.2.4 Unreal Engine 4 Ambient Occlusion

La técnica de Unreal Engine 4 *Ambient Occlusion* no cuenta con mucha información en la literatura, de hecho, no hay referencia de un artículo explicando a detalle esta técnica, debido a esto sólo se cuenta con información principalmente teórica. De Unreal engine 4 *Ambient Occlusion* se sabe que es una variación de la técnica de *Horizon-Based Ambient Occlusion (HBAO)*, la cual consiste en explorar un terreno a través de un vector y encontrar el horizonte más alto dentro de un radio. Unreal Engine 4 parte de este principio pero a cada vector generado le crea a su vez un vector inverso que explora en la dirección contraria, es decir, un vector reflejo, a estos dos vectores se les calcula el ángulo entre sí y este ángulo será el que defina la oclusión para un punto de interés, es

decir, a menor ángulo mayor oclusión y a mayor ángulo menor oclusión, esta simple mejora les permite utilizar menos muestras que las necesitadas por HBAO. A parte del cambio en el muestreo, a diferencia de HBAO que sólo utiliza el *depth buffer*, Unreal Engine utiliza también el *normal buffer* el cual les permite obtener más detalle de la escena [6].

4.2.3 Gaussian Blur

Debido a la necesidad de calcular oclusión para aplicaciones en tiempo real la técnica de SSAO y todas las técnicas que se desarrollaron a partir de ella generan ruido en el resultado final, producto de realizar muestreo aleatorio. Una solución es utilizar mayor cantidad de muestras produciendo menor ruido debido a la mayor precisión obtenida. Sin embargo, al utilizar más muestras se pierde rendimiento y por ende la aplicación pierde la capacidad de ejecutarse en tiempo real, para solucionar este problema se opta por aplicar una difuminación al resultado final de la técnica de *Ambient Occlusion* perdiendo un poco la calidad del resultado pero se elimina en gran medida el ruido generado.

Para este paso se utiliza la técnica de *Gaussian Blur* para reducir el ruido de una imagen. En este trabajo para aplicar esta técnica se divide el proceso en dos pasadas: en la primera pasada, se desenfoca la imagen sólo en la dirección horizontal o vertical; en la segunda pasada, se desenfoca en la dirección restante. En la Figura 31 se observa la función *main* del *fragment shader* que realiza el *blur* vertical.

```
void main() {
    //get the inverse of texture size
    vec2 delta = 1.0/textureSize(textureMap,0);
    vec4 color = vec4(0);
    int  index = 20;
    //go through all neighbors and multiply the kernel value with the obtained
    //colour from the input image
    for(int i=-10;i<=10;i++) {
        color += kernel[index--]*texture(textureMap, vUV + (vec2(0,i*delta.y)));
    }
    //return the filtered colour as fragment output
    vFragColor =  color;
}
```

Figura 31: Fragment Shader Gaussian Blur Vertical.

Para el paso horizontal se emplea delta.x en vez de delta.y.

4.2.4 Blending Final

Una vez se obtiene el resultado final de la técnica de *Ambient Occlusion* en una textura es necesario mezclar (*blend*) ese resultado con el resto de la escena, para ello se utiliza un último shader que simplemente aplica la textura al plano que se encuentra frente de pantalla (*full-screen quad*). Luego, a este plano se le aplica *blending*, el cual es una técnica conocida por aplica transparencia entre objetos de una escena permitiendo "mezclar" el resultado de la técnica de *Ambient Occlusion* aplicado en dicho plano con el resto de la escena. En la siguiente Figura 32 se observa el código C++ donde se ejecuta el *shader* y se realizan las instrucciones que aplican *blending.*

```
//unbind FBO, restore the defaul viewport and draw buffer
glBindFramebuffer(GL_FRAMEBUFFER,0);
glViewport(0,0,WWidth, WHeight);
glDrawBuffer(GL_BACK_LEFT);

//now draw the final filtered SSAO result with blending
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
glBlendEquation(GL_FUNC_ADD);
ShaderFinal->Enable();
    ShaderFinal->setUniform("textureMap", 4);
    glDrawArrays(GL_TRIANGLES,0,6);
ShaderFinal->Disable();

//disable blending
glDisable(GL_BLEND);
```

Figura 32: Fragmento de código en C++ de la operación de blending.

Dentro del *fragment shader* se aplica la textura con *Ambient Occlusion* al plano. Hay que destacar que la técnica de MSSAO es un poco diferente en este paso, a diferencia de las demás, produce cinco texturas, las cuales deben mezclarse dentro del *fragment shader* antes de ser aplicadas en el plano dependiendo de la intensidad seleccionada para cada textura. En la Figura 33 se observa el código del *fragment shader* utilizado para el *blending* final.

```
void main()
{
    if(esMSSAO){
        vec4 tex1 = texture(textureMap, vUV);
        vec4 tex2 = texture(textureMap2, vUV);
        vec4 tex3 = texture(textureMap3, vUV);
        vec4 tex4 = texture(textureMap4, vUV);
        vec4 tex5 = texture(textureMap5, vUV);

        vec4 mixText = tex1*intensidadTex1 + tex2*intensidadTex2 +
                       tex3*intensidadTex3 + tex4*intensidadTex4 +
                       tex5*intensidadTex5;
        vFragColor = mixText;
    }else{
        vFragColor = texture(textureMap, vUV);
    }
}
```

Figura 33: Fragment shader de la operacion de blending.

Capítulo 5

Pruebas y Resultados

En este capítulo se expondrán los resultados obtenidos al realizar diferentes pruebas en cada una de las técnicas de *Ambient Occlusion* seleccionadas para este Trabajo Especial de Grado. Primero se explicarán las escenas utilizadas para realizar las pruebas. Luego se mostrarán las diferencias entre los resultados al aplicar diferentes parámetros utilizando una tabla comparativa entre el rendimiento, calidad visual, porcentaje de percepción visual entre las diferentes técnicas. Finalmente se mostrará una colección de imágenes donde podrá enfocarse en observar diferencias visuales entre las técnicas aplicadas a las escenas.

5.1 Ambiente de pruebas

Para realizar las evaluaciones de las diferentes técnicas de *Ambient Occlusion* se utilizan tres escenas que consisten principalmente de un objeto de interés sobre un plano, la escena mostrada en la Figura 34 (a) es una antigua herrería de un pueblo donde se puede apreciar los detalles tanto del exterior como el interior del taller, la escena mostrada en la Figura 34 (b) es un dragón el cual permite demostrar cómo se comporta cada técnica sobre un modelo con alta oclusión en todo el cuerpo y la tercera escena mostrada en la Figura 34 (c) es un pequeño pueblo con seis estructuras de dos pisos el cual nos permite observar una introducción de cómo se comporta la técnica de *Ambient Occlusion* en espacios abiertos. En la Figura 34 se observan las tres escenas.

(a) Herrería (b) Dragón (c) Pueblo

Figura 34: Escenarios de prueba.

En la Tabla 1 se muestra el nombre de cada objeto presente en las diferentes escenas, cantidad de vértices y polígonos.

Modelos	Nro. de Vértices	Nro. de Triángulos
Suelo (Plano)	4	2
SkyBox	8	12
Herrería	15.069	10.294
Dragón	8160	16191
Pueblo	17726	26256

Tabla 1: Detalles sobre los escenarios de prueba..

Las pruebas se realizaron bajo un mismo ambiente. Se utilizó una PC con las siguientes especificaciones: Windows 10 de 64bit, Procesador Intel Core i7-4790K de 4 GHz, 16GB de RAM y tarjeta de video Nvidia GeForce GTX 970 con 4GB de memoria. En la Tabla 2 se observan algunas características de la tarjeta de video.

Características	Nvidia GeForce GTX 970
Reloj de núcleo	1178 MHz
Reloj de memoria	1753 MHz
Ancho de banda de la memoria	224.4 GB/s
Tasa de relleno de píxeles	66.0 GPixel/s
Tasa de relleno de texturas	122.5 GTexel/s

Tabla 2: Especificaciones de la tarjeta de video.

5.2 Resultados visuales y rendimiento

A continuación se presentan los resultados de las diferentes pruebas de rendimiento realizadas en las tres escenas, aplicando cada técnica de *Ambient Occlusion* para dos puntos de vista en una misma escena. Se decide mostrar más de un punto de vista con la intención de demostrar que incluso la posición de la cámara afecta los resultados de rendimiento. En primer lugar se mostrará el tiempo promedio que tarda en ejecutarse cada técnica implementada, además del tiempo promedio que tarda cada escena en renderizar los objetos, en aplicar la técnica de *blur* y el *blending* final con los parámetros que ofrecen los mejores resultados visuales. Luego se muestra una serie de tablas comparativas, además de los resultados visuales de cada técnica al hacer una variación de parámetros. Finalmente se muestran los resultados al realizar comparaciones perceptuales sobre cada escena. Cabe destacar que la unidad de tiempo utilizada es de milisegundos y todas las pruebas se realizaron a una resolución de 1920x1080 píxeles también conocida como resolución 1080p o Full HD.

Se presenta una tabla con los resultados de tiempo promedio para cada técnica de *Ambient Occlusion,* además del tiempo que se tarda cada escena en renderizar los objetos, *blur* y *blending* final sobre el primer punto de vista de la primera escena. Luego se muestra una serie de figuras donde se compara visualmente los resultados mostrando la escena con *Ambient Occlusion* y la escena con sólo *Ambient Occlusion* sin texturas.

Los parámetros utilizados para esta prueba se presentan en la Tabla 3, donde el radio se representa en metros y las muestras son pixeles que se toman de la textura de profundidad.

Técnica	Radio(m)	Muestras
Alchemy AO	1	16
Scalable AO	10	8
Unreal Engine AO	0,05	8
Multi Scale AO	2	16

Tabla 3: Parámetros utilizados para pruebas.

1. Escena 1: Herrería, punto de vista 1.

   En la Figura 35 se muestra el primer punto de vista de la escena 1 sin *Ambient Occlusion.*

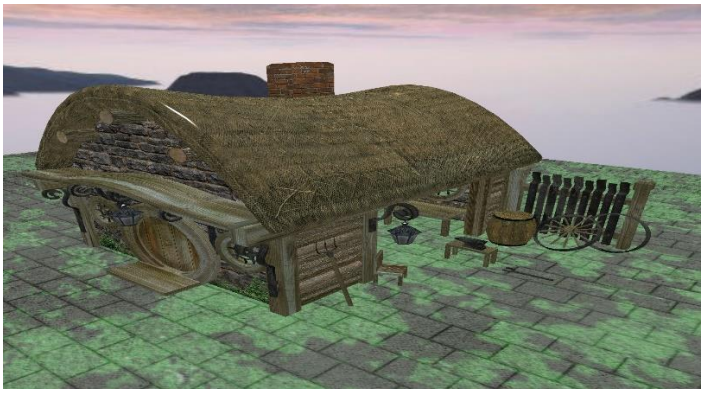


Figura 35: Escena 1 punto de vista 1.

Pruebas	Tiempo Promedio (ms)
Render Objetos	0,5538
Alchemy AO	0,9839
Scalable AO	0,8840
Unreal Engine AO	1,2942
Multi Scale AO	1,4198
Blur	2,7808
Blur Multi Scale AO	3,4103
Final Blending	0,1546
Final Blending Multi Scale AO	0,3096

Tabla 4: Resultados del tiempo promedio de ejecución para cada técnica *Ambient Occlusion*.

- **Alchemy Ambient Obscurance**

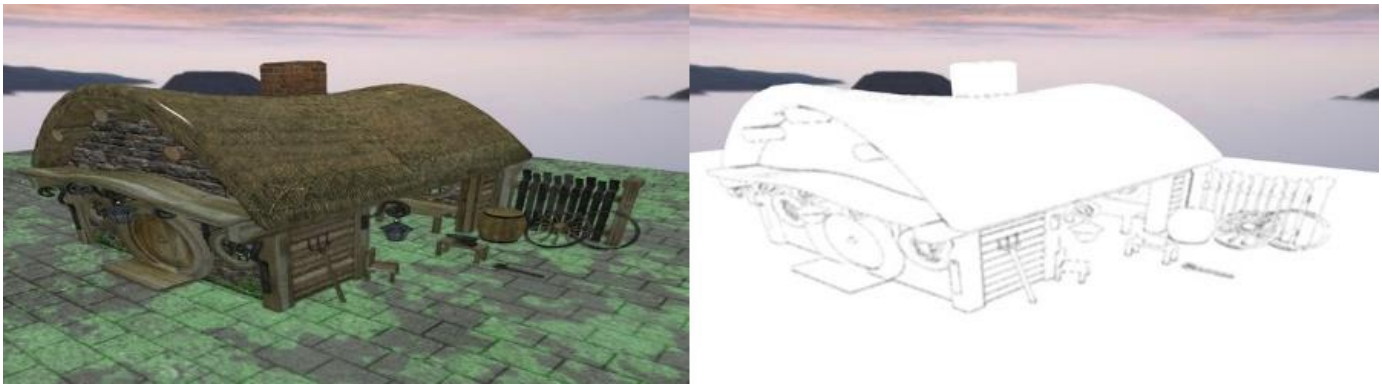


Figura 36: Escena 1 Punto de vista 1 comparación visual Alchemy AO.

- **Scalable Ambient Obscurance**

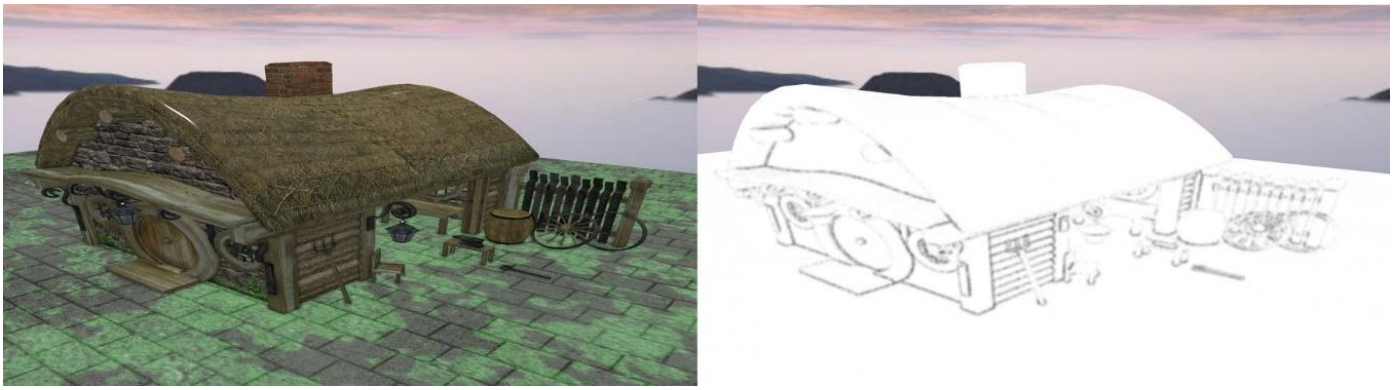


Figura 37: Escena 1 punto de vista 1 comparación visual Scalable AO.

- **Unreal Engine 4 Ambient Occlusion**

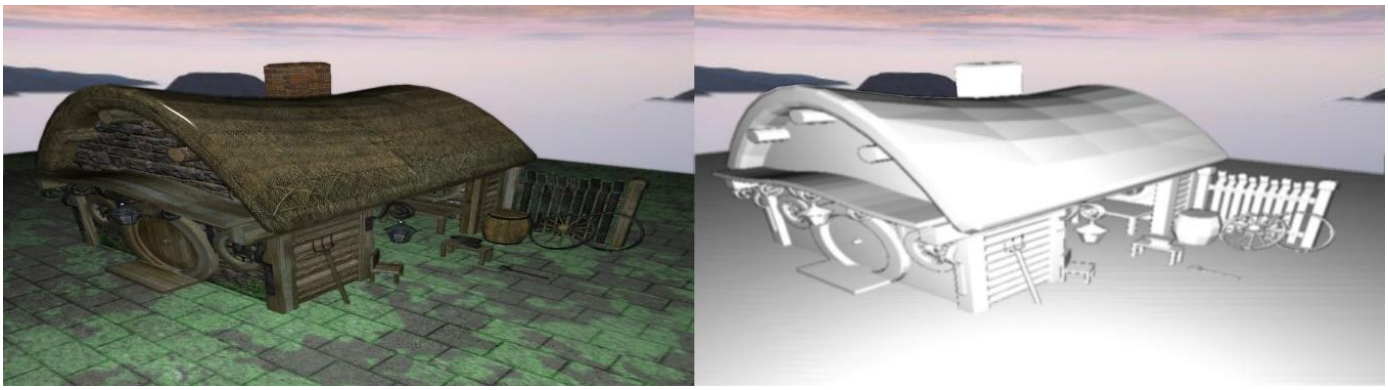


Figura 38: Escena 1 punto de vista 1 comparación visual Unreal Engine 4 AO.

- **Multi-Scale Ambient Occlusion**

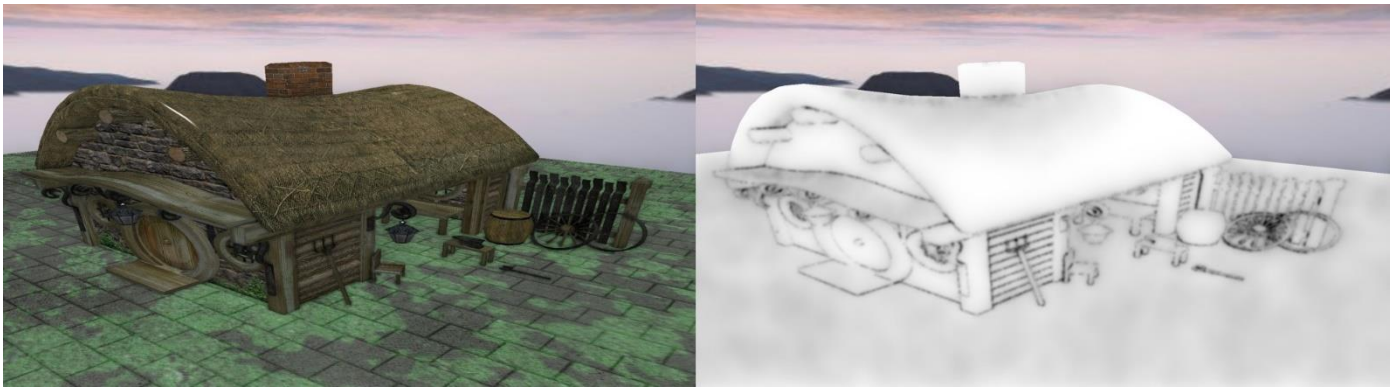


Figura 39: Escena 1 punto de vista 1 comparación visual Multi Scale AO.

En la Tabla 4 se observa que en la escena 1 para el punto de vista 1 la técnica de *Scalable AO* es la que provee mejor rendimiento cumpliendo con la premisa de que es una optimización de la técnica de *Alchemy AO*. Por otra parte la técnica de *Multi Scale AO* es la que menor rendimiento ofrece. Sin embargo, el incremento no es tan significativo. Con respecto al resultado visual, es difícil de ver a simple vista el resultado de *Alchemy AO* y *Scalable AO* en la escena con texturas. Sin Embargo, Para la técnica de *Unreal Engine 4 AO* queda claro donde se encuentran la oclusión en la escena.

2. Escena 1: herrería, punto de vista 2.

   En la Figura 40 se muestra el segundo punto de vista de la escena 1 sin *Ambient Occlusion.*

Figura 40: Escena 1 punto de vista 2.

Pruebas	Tiempo Promedio (ms)
Render Objetos	0,5354
Alchemy AO	1,2879
Scalable AO	0,9587
Unreal Engine AO	1,4352
Multi Scale AO	1,7432
Blur	2,6524
Blur Multi Scale AO	3,4856
Final Blending	0,1768
Final Blending Multi Scale AO	0,3269

Tabla 5: Resultados del tiempo promedio de ejecución para cada técnica *Ambient Occlusion*.

- **Alchemy Ambient Obscurance**

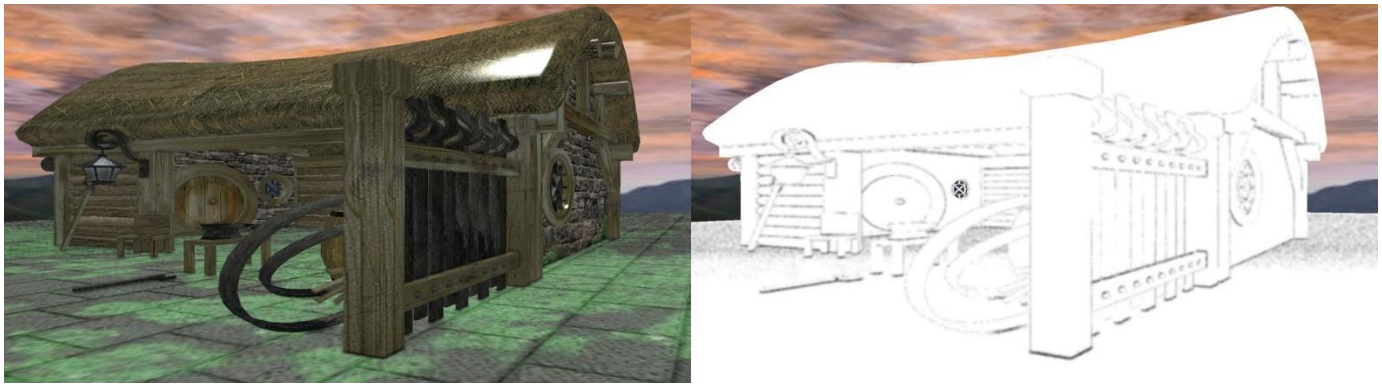


Figura 41: Escena 1 punto de vista 2 comparación visual Alchemy AO.

- **Scalable Ambient Obscurance**

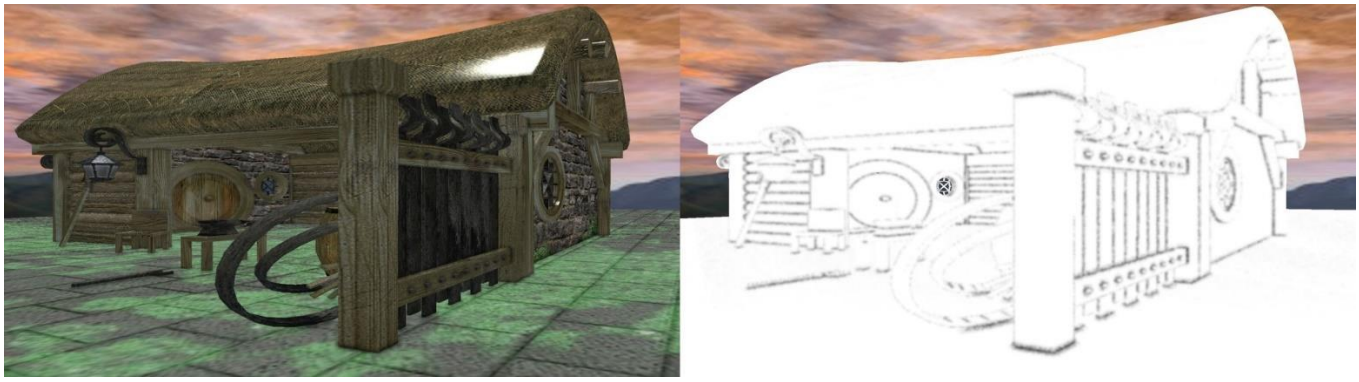


Figura 42: Escena 1 punto de vista 2 comparación visual Scalable AO.

- **Unreal Engine 4 Ambient Occlusion**

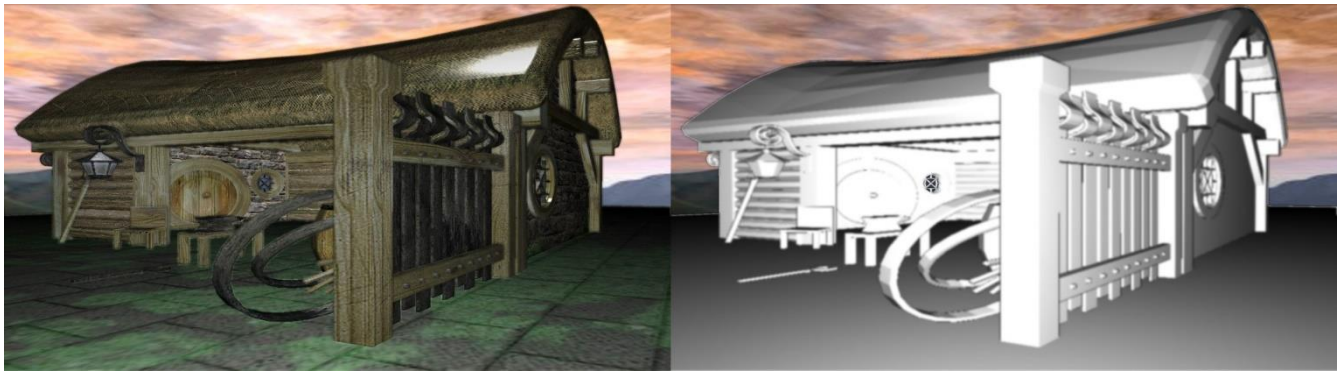


Figura 43: Escena 1 punto de vista 2 comparación visual Unreal Engine 4 AO.

- **Multi-Scale Ambient Occlusion**

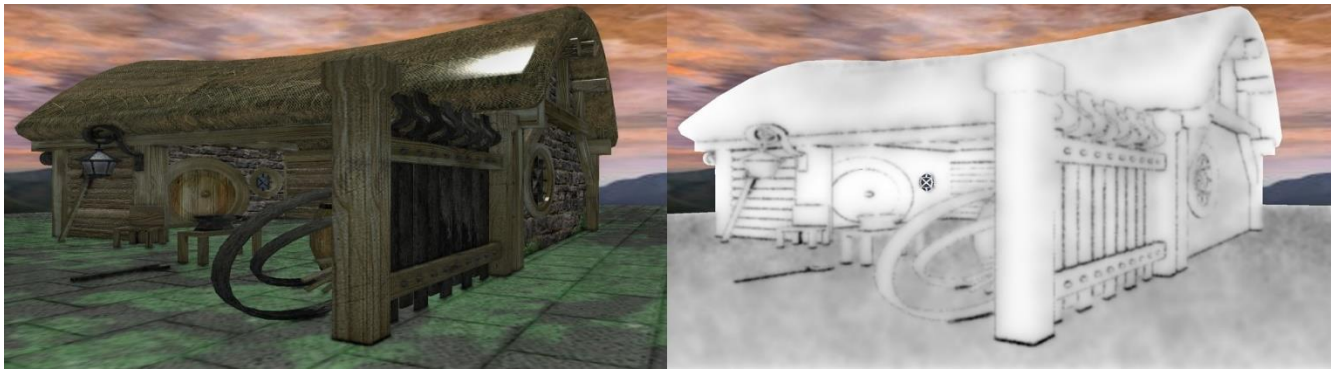


Figura 44: Escena 1 punto de vista 2 comparación visual Multi Scale AO.

Para el segundo punto de vista de la primera escena, lo primero que se observa en la Tabla 5 es como el hecho de cambiar el punto de vista tiene efecto en los valores de rendimiento. Sin embargo, la técnica de *Scalable AO* permanece como la técnica de mejor rendimiento aumentando su tiempo de ejecución en aproximadamente 0,07 milisegundos, mientras que las otras técnicas aumentan más de 0,15 milisegundos. Debido a la cercanía con el objeto resulta más fácil observar los cambios visuales, de igual manera las técnicas de *Unreal Engine 4 AO* y *Multi Scale AO* son las que se ve mejor la oclusión con las texturas aplicadas.

3. Escena 2: Dragón, punto de vista 1

   En la Figura 45 se muestra el primer punto de vista de la escena 2 sin *Ambient Occlusion.*

Figura 45: Escena 2 punto de vista 1 sin Ambient Occlusion.

Pruebas	Tiempo Promedio (ms)
Render Objetos	0,4272
Alchemy AO	1,0900
Scalable AO	0,8082
Unreal Engine AO	1,2072
Multi Scale AO	1,4487
Blur	2,6907
Blur Multi Scale AO	3,3123
Final Blending	0,1903
Final Blending Multi Scale AO	0,3199

Figura 46: Resultados del tiempo promedio de ejecución para cada técnica *Ambient Occlusion*.

- **Alchemy Ambient Obscurance**

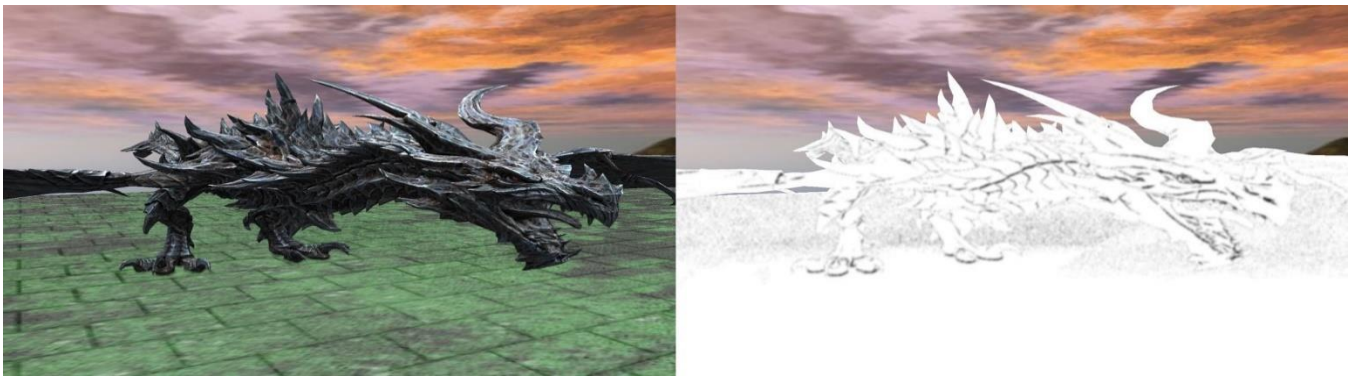


Figura 47: Escena 2 punto de vista 1 comparación visual Alchemy AO.

- **Scalable Ambient Obscurance**

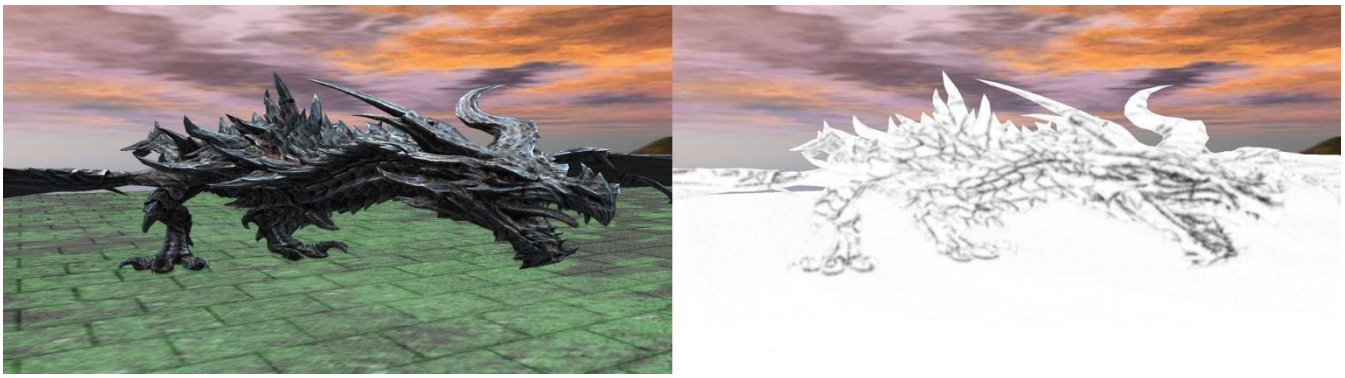


Figura 48: Escena 2 punto de vista 1 comparación visual Scalable AO.

- **Unreal Engine 4 Ambient Occlusion**

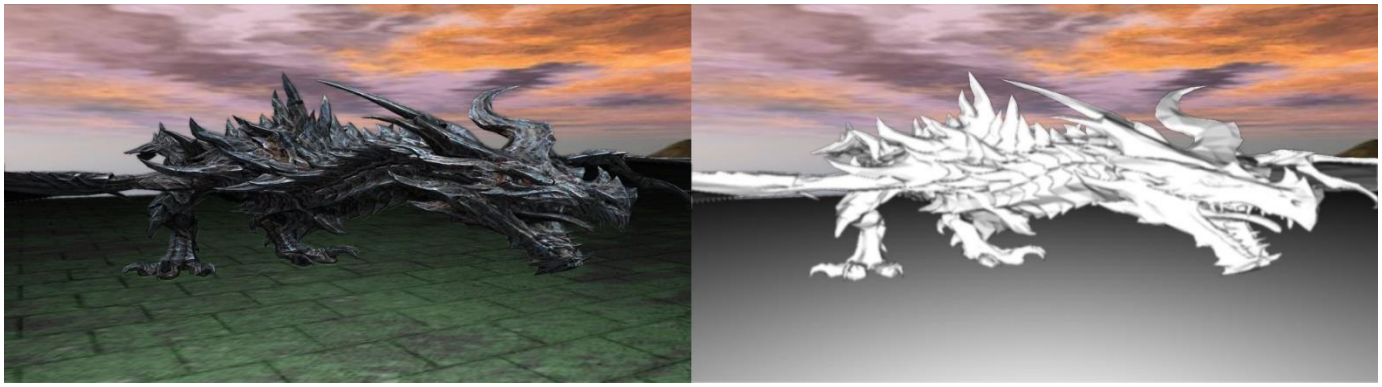


Figura 49: Escena 2 punto de vista 1 comparación visual Unreal Engine 4 AO.

- **Multi-Scale Ambient Occlusion**

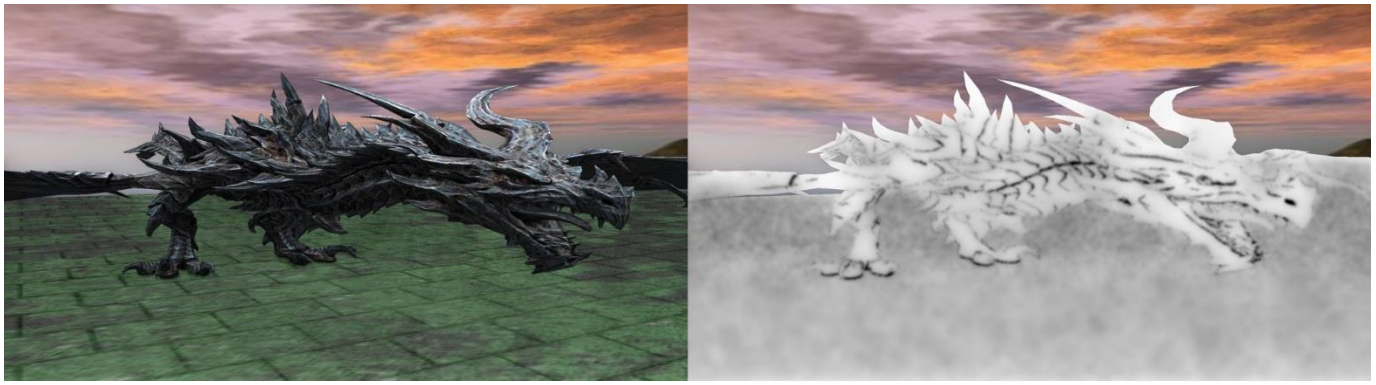


Figura 50: Escena 2 punto de vista 1 comparación visual Multi Scale AO.

En esta escena se observa un objeto con mucha oclusión debido a las escamas del dragón. Se observa que las técnicas que resaltan mejor los detalles de este objeto son *Scalable AO* en la Figura 48 y *Multi Scale AO* en la Figura 50, aunque todas realizan una buena representación. En términos de rendimiento *Scalable AO* sigue siendo la más eficiente de todas las técnicas como se observa en la Figura 46.

4. Escena 2: dragón, punto de vista 2.

   En la Figura 51 se muestra el segundo punto de vista de la escena 2 sin *Ambient Occlusion.*

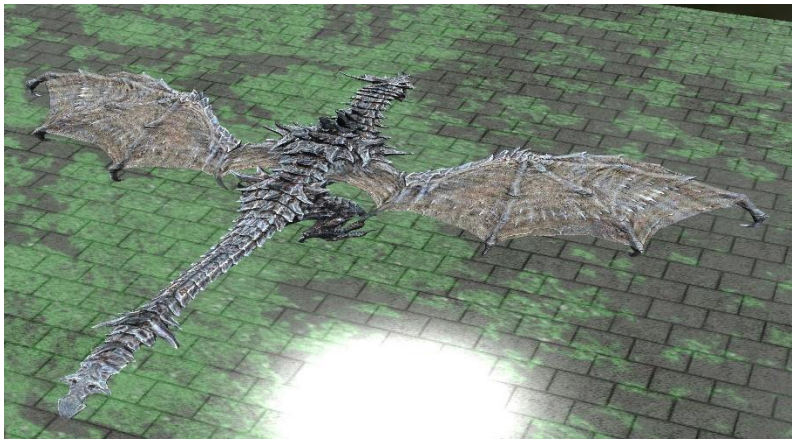


Figura 51: Escena 2 punto de vista 2 sin ambient occlusion.

Pruebas	Tiempo Promedio (ms)
Render Objetos	0,5094
Alchemy AO	1,0899
Scalable AO	1,1114
Unreal Engine AO	1,6250
Multi Scale AO	1,8597
Blur	2,7919
Blur Multi Scale AO	3,4573
Final Blending	0,1454
Final Blending Multi Scale AO	0,3087

Tabla 6: Resultados del tiempo promedio de ejecución para cada técnica *Ambient Occlusion*.

- **Alchemy Ambient Obscurance**

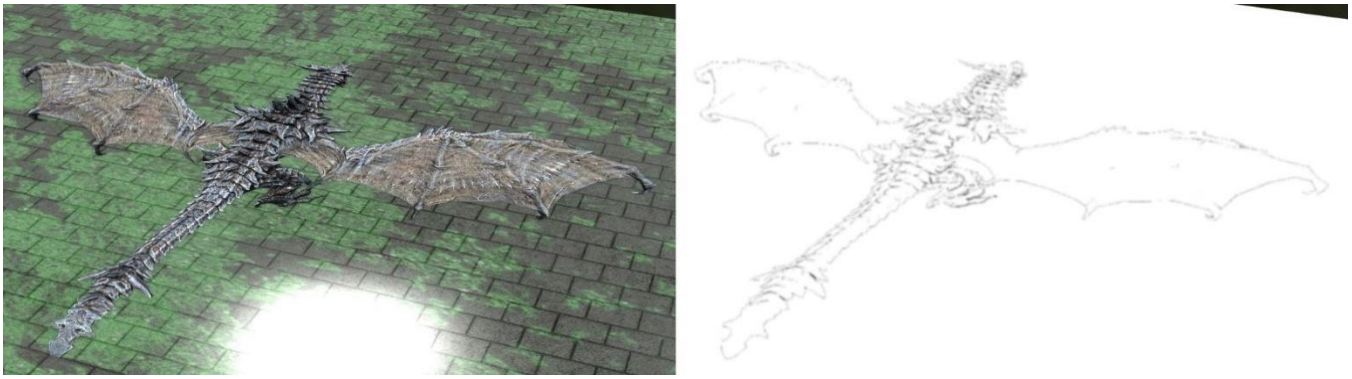


Figura 52: Escena 2 punto de vista 2 comparación visual Alchemy AO.

- **Scalable Ambient Obscurance**

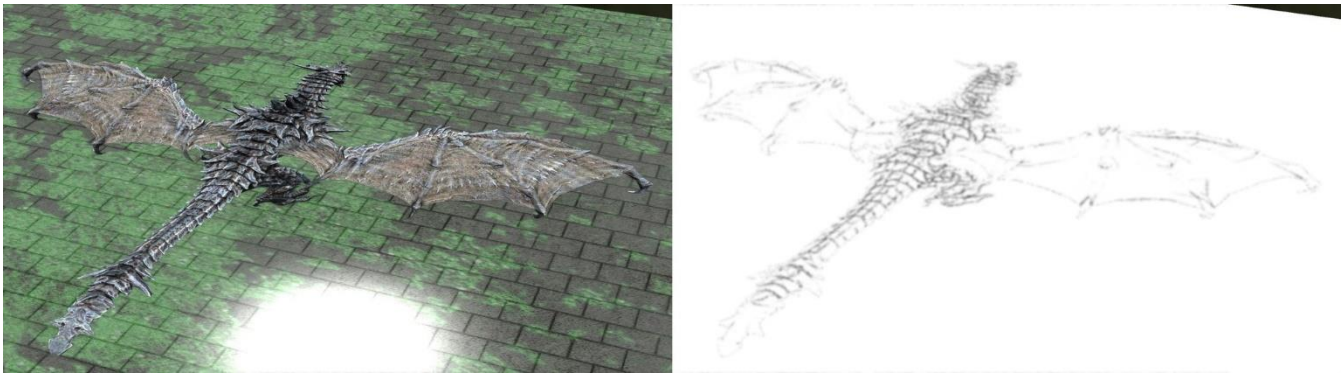


Figura 53: Escena 2 punto de vista 2 comparación visual Scalable AO.

- **Unreal Engine 4 Ambient Occlusion**

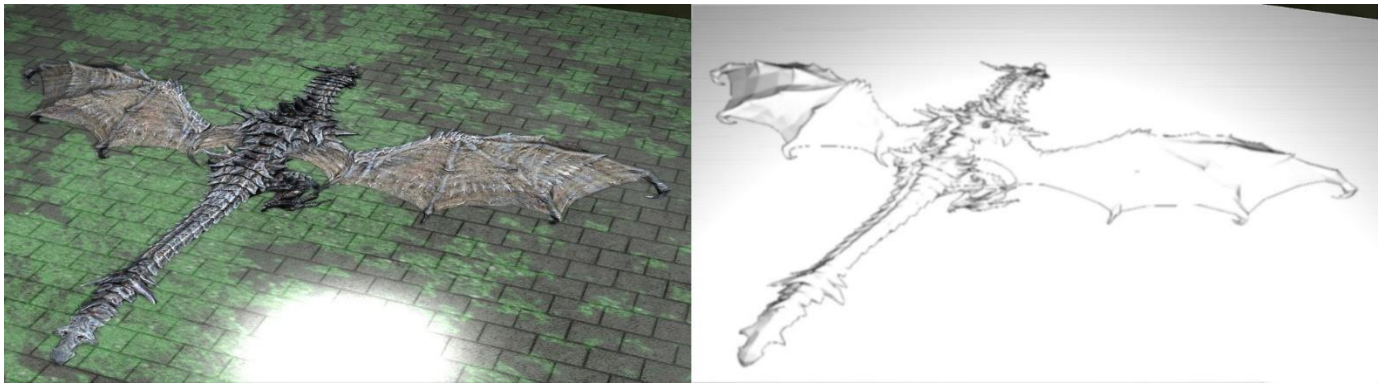


Figura 54: Escena 2 punto de vista 2 comparación visual Unreal Engine 4 AO.

- **Multi-Scale Ambient Occlusion**

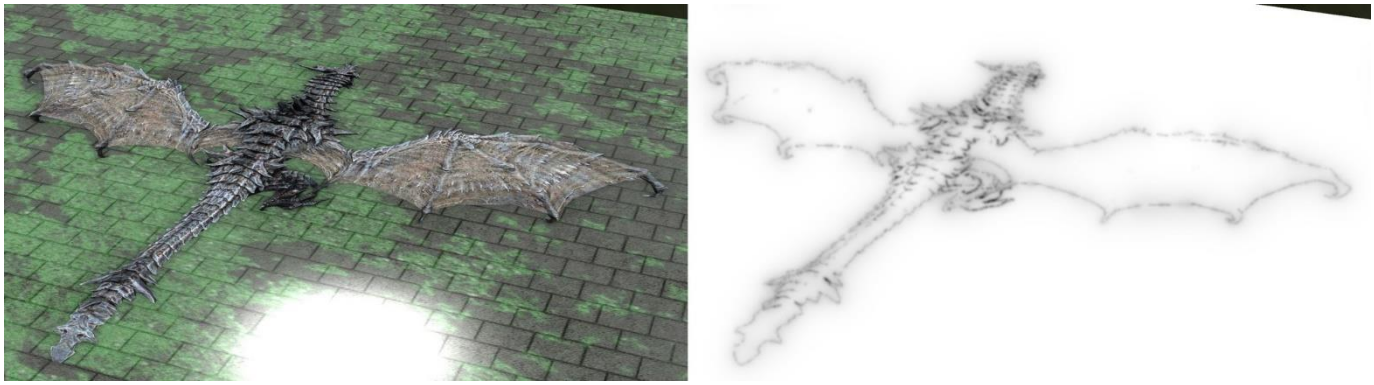


Figura 55: Escena 2 punto de vista 2 comparación visual Multi Scale AO.

Debido a la distancia a la que se encuentra el objeto, es muy difícil observar la oclusión con texturas. Sin embargo, al mostrar solamente la oclusión, las técnicas de *Scalable AO y* Unreal Engine 4 AO que se observan en las Figuras 53 y 54 respectivamente son las que mejor representan los detalles del objeto, siendo *Scalable AO* capaz de resaltar la mayoría de las escamas del dragón. Con respecto al rendimiento en este caso la técnica de *Alchemy AO* es la que mejor resultados provee superando incluso la técnica de *Scalable AO,* demostrando que no es superior en todos los casos.

5. Escena 3: Pueblo, punto de vista 1

   En la Figura 56 se muestra el primer punto de vista de la escena 3 sin *Ambient Occlusion.*

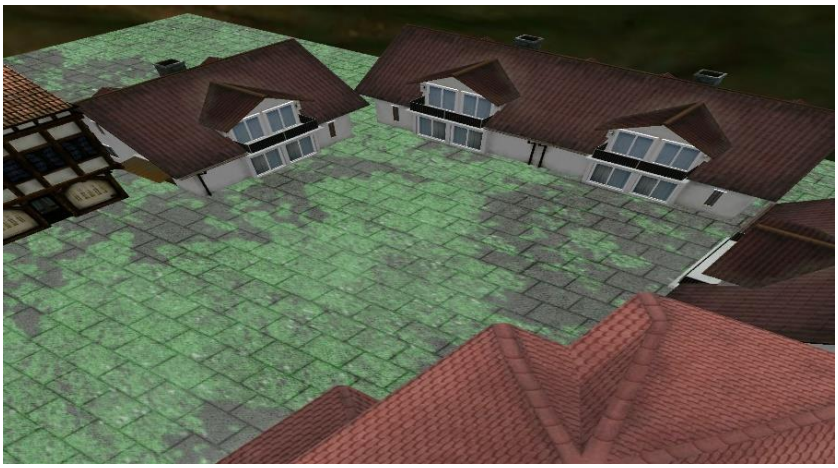


Figura 56: Escena 3 punto de vista 1.

Pruebas	Tiempo Promedio (ms)
Render Objetos	0,5434
Alchemy AO	1,0081
Scalable AO	0,9868
Unreal Engine AO	1,4794
Multi Scale AO	1,6958
Blur	2,6934
Blur Multi Scale AO	3,5597
Final Blending	0,1578
Final Blending Multi Scale AO	0,3102

Tabla 7: Resultados del tiempo promedio de ejecución para cada técnica *Ambient Occlusion*.

- **Alchemy Ambient Obscurance**

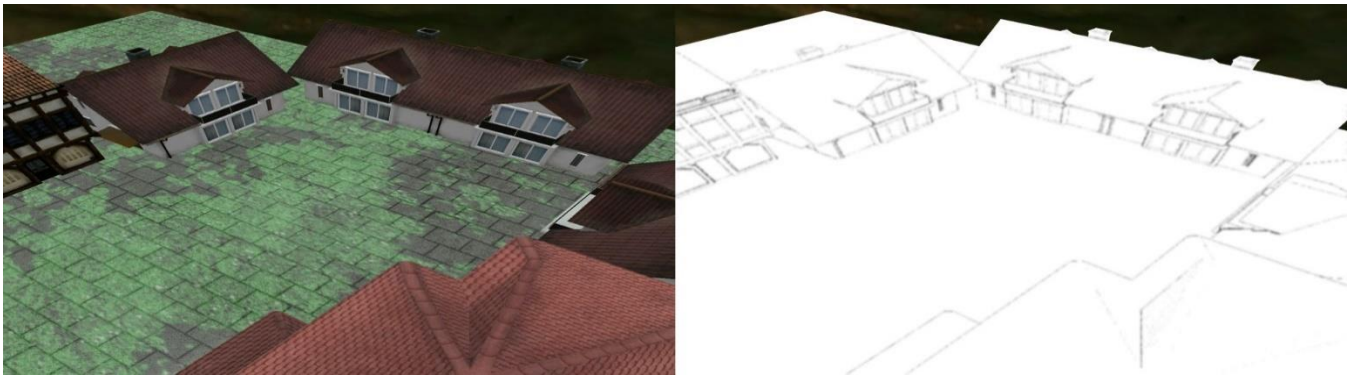


Figura 57: Escena 3 punto de vista 1 comparación visual Alchemy AO.

- **Scalable Ambient Obscurance**

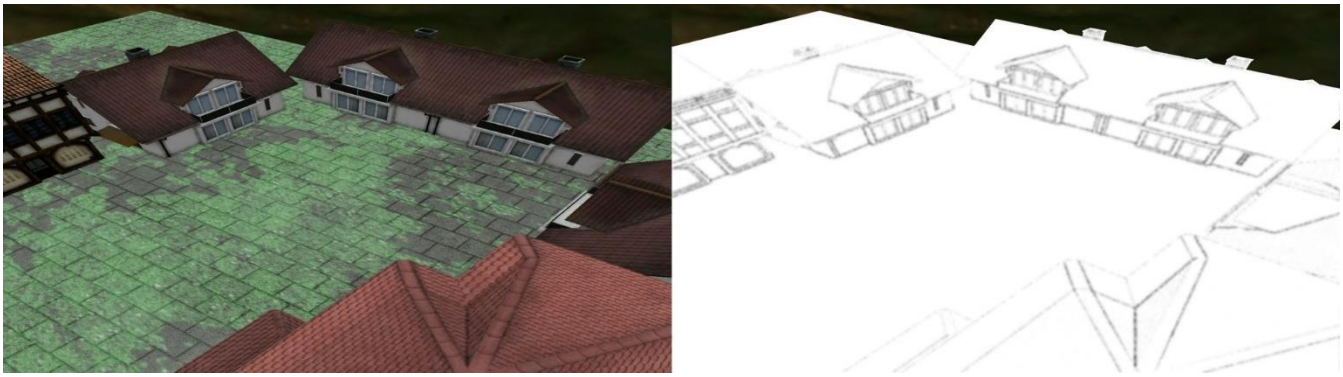


Figura 58: Escena 3 punto de vista 1 comparación visual Scalable AO.

- **Unreal Engine 4 Ambient Occlusion**

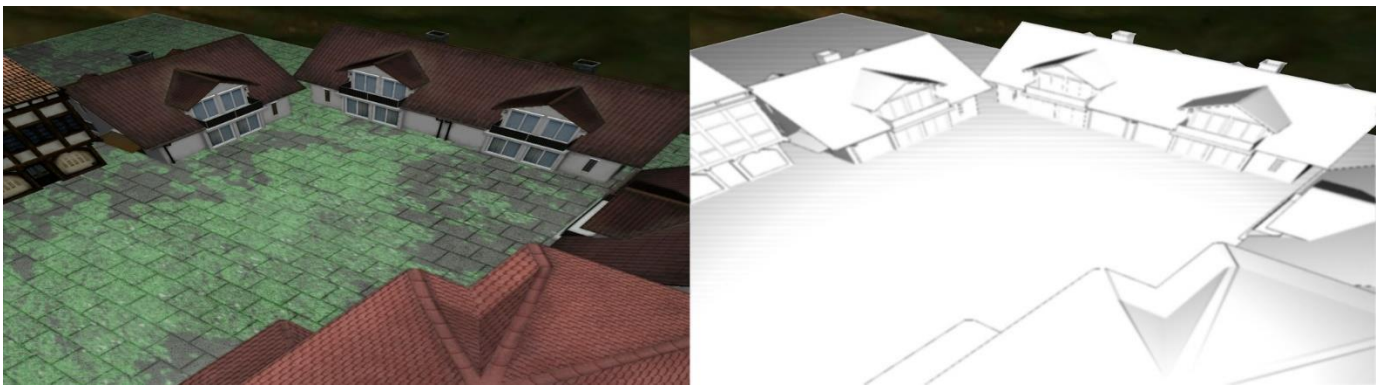


Figura 59: Escena 3 punto de vista 1 comparación visual Unreal Engine 4 AO.

- **Multi-Scale Ambient Occlusion**

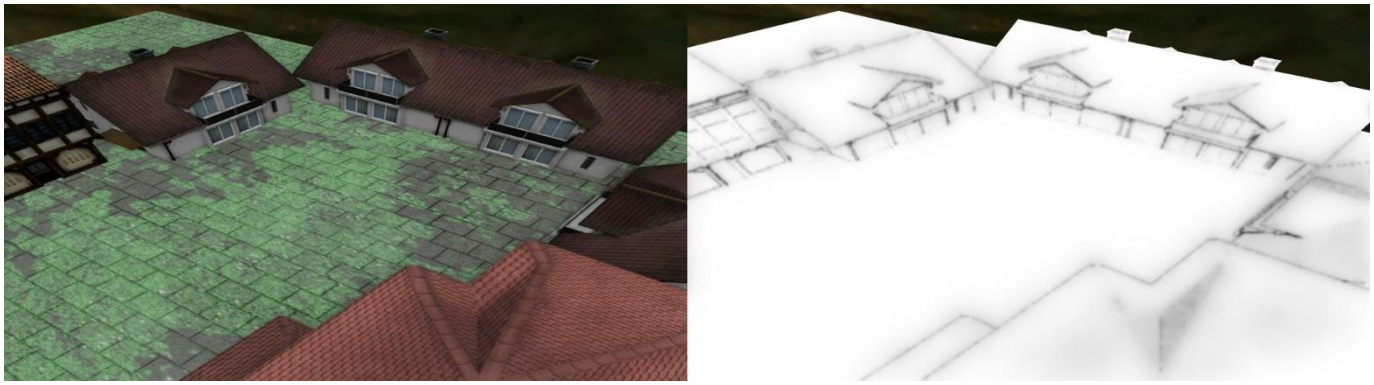


Figura 60: Escena 3 punto de vista 1 comparación visual Multi Scale AO.

Para el primer punto de vista de la tercera escena se observa que una vez más la distancia de los objetos dificulta su visualización, siendo la técnica de Unreal Engine 4 AO en la Figura 59 y en menor medida la técnica de *Scalable AO* en la Figura 58 las más fáciles de observar. Por otra parte, al mostrar sólo la oclusión en la técnica de *Unreal Engine 4 AO* en la Figura 59, se observa uno de los errores de las técnicas de SSAO, se le conoce como "*banding",* produciendo las líneas de sombra que se observa en el suelo tal como se observa en la Figura 61. En cuanto al rendimiento observado en la Tabla 7, *Scalable AO* ofrece el mejor resultado seguido muy de cerca por *Alchemy AO*. Por otra parte, las técnicas de Unreal Engine 4 AO y *Multi Scale AO* se tardan un tiempo considerable en comparación con *Alchemy AO* y *Scalable AO.*

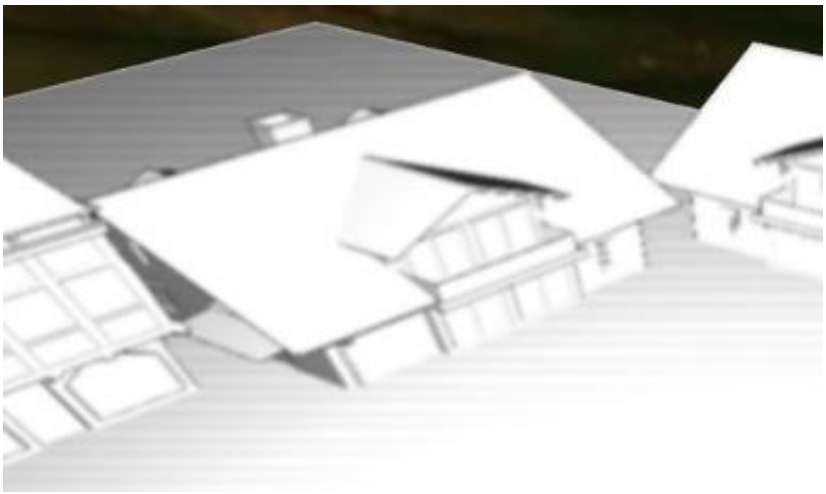


Figura 61: Error de banding en la técnica de Unreal Engine 4 AO.
Líneas horizontales en el suelo de la escena con diferentes todos de grises.

6. Escena 3: pueblo, punto de vista 2.

   En la Figura 62 se muestra el segundo punto de vista de la escena 3 sin *Ambient Occlusion.*

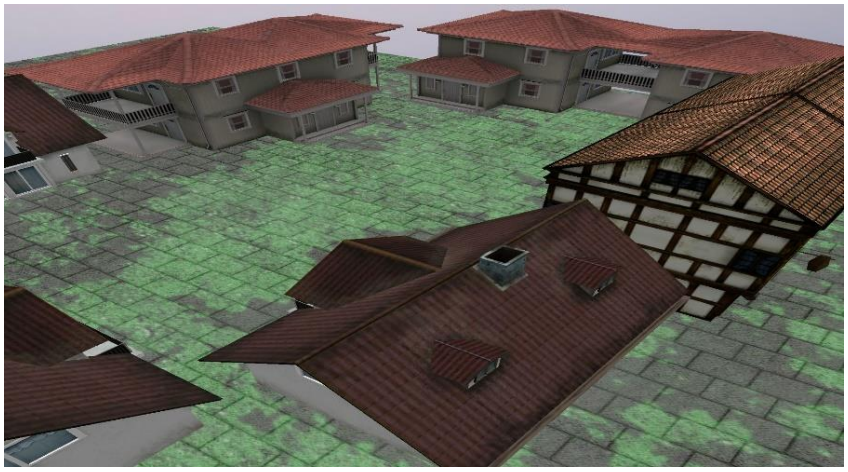


Figura 62: Escena 3 punto de vista 2 sin Ambient Occlusion.

Pruebas	Tiempo Promedio (ms)
Render Objetos	0,6869
Alchemy AO	1,0530
Scalable AO	0,9664
Unreal Engine AO	1,5062
Multi Scale AO	1,7762
Blur	2,6934
Blur Multi Scale AO	3,5597
Final Blending	0,1887
Final Blending Multi Scale AO	0,3196

Tabla 8: Resultados del tiempo promedio de ejecución para cada técnica Ambient Occlusion.

- **Alchemy Ambient Obscurance**

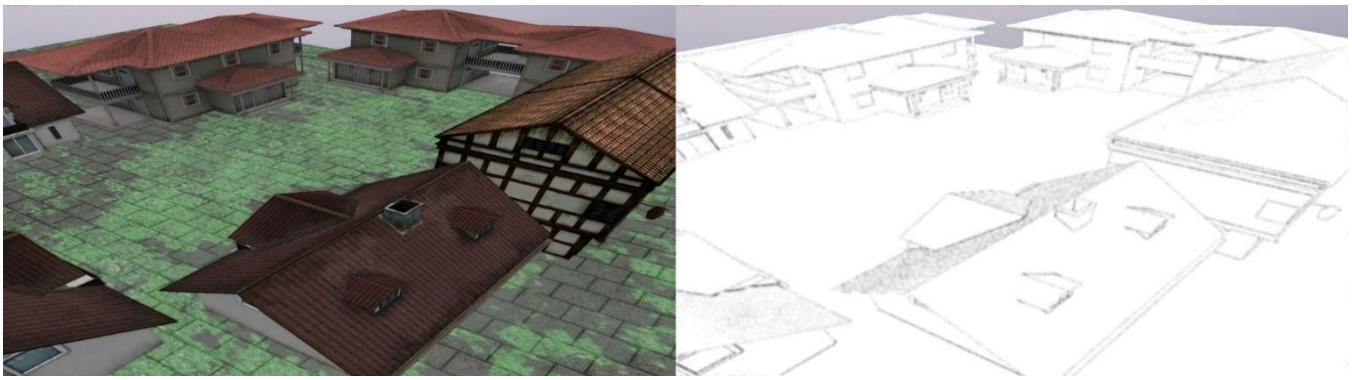


Figura 63: Escena 3 punto de vista 2 comparación visual Alchemy AO.

- **Scalable Ambient Obscurance**

Figura 64: Escena 3 punto de vista 2 comparación visual Scalable AO.

- **Unreal Engine 4 Ambient Occlusion**

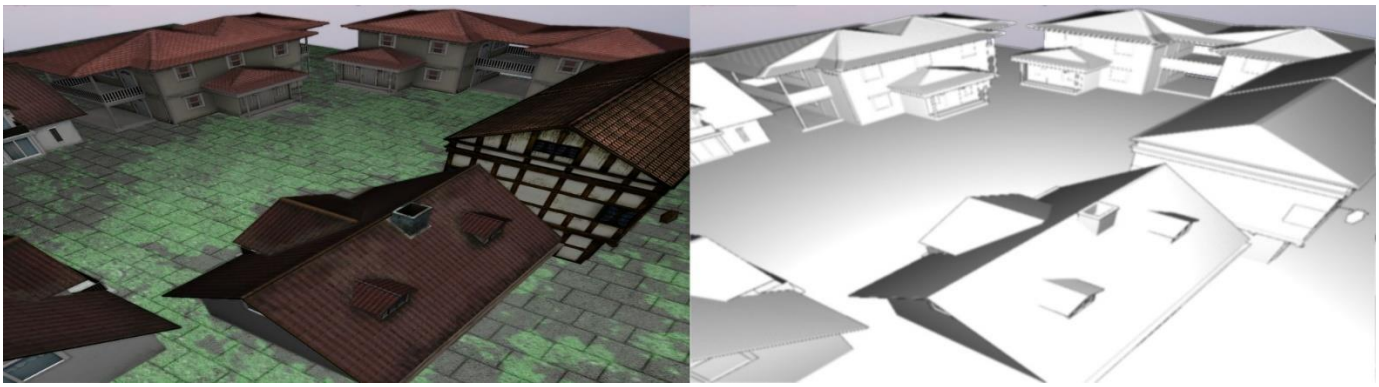


Figura 65: Escena 3 punto de vista 2 comparación visual Unreal Engine 4 AO.

- **Multi-Scale Ambient Occlusion**

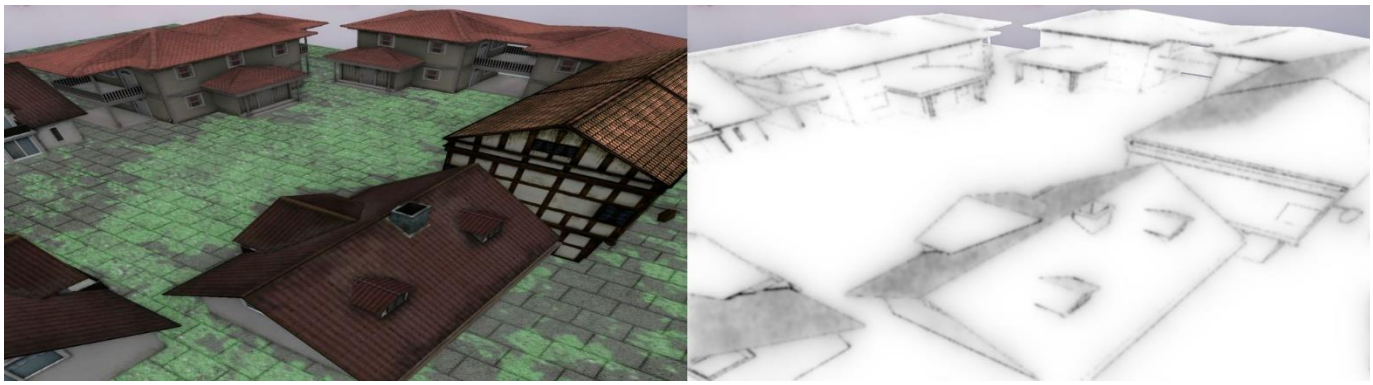


Figura 66: Escena 3 punto de vista 2 comparación visual Multi Scale AO.

A diferencia de las escenas anteriores, en esta escena no se alteran considerablemente el rendimiento entre los diferentes punto de vista, obteniendo así resultados similares, esto es debido a que ambas tomas fueron realizadas a distancia similares. En el resultado visual, se observa que todas las técnicas en general ofrecen una buena representación de los detalles de la escena, siendo

Scalable AO y *Unreal Engine 4 AO* en la Figura 64 65 respectivamente las que mejor representan los detalles de la escena, por ejemplo, los detalles en los techos.

5.2.1 Resumen y Observaciones

A continuación se presenta un resumen de las tablas presentadas anteriormente con la finalidad de exponer observaciones referentes a los resultados expresados en las mismas.

Técnica	Radio (m)	Muestras	Escena 1 Punto de vista 1 (ms)	Escena 1 Punto de vista 2 (ms)	Escena 2 Punto de vista 1(ms)
Render Objetos	--	--	0,5538	0,5354	0,4272
Alchemy AO	1	16	0,9839	1,2879	1,0900
Scalable AO	10	8	0,8840	0,9587	0,8082
Unreal Engine AO	0,05	8	1,2942	1,4352	1,2072
Multi Scale AO	2	16	1,4198	1,7432	1,4487
Blur	--	--	2,7808	2,6524	2,6907
Blur Multi Scale AO	--	--	3,4103	3,4856	3,3123
Final Blending	--	--	0,1546	0,1768	0,1903
Final Blending Multi Scale AO	--	--	0,3096	0,3269	0,3199
Técnica	**Radio (m)**	**Muestras**	**Escena 2 Punto de vista 2 (ms)**	**Escena 3 Punto de vista 1 (ms)**	**Escena 3 Punto de vista 2(ms)**
Render Objetos	--	--	0,5094	0,5434	0,6869
Alchemy AO	1	16	1,0899	1,0081	1,0530
Scalable AO	10	8	1,1114	0,9868	0,9664
Unreal Engine AO	0,05	8	1,6250	1,4794	1,5062
Multi Scale AO	2	16	1,8597	1,6958	1,7762
Blur	--	--	2,7919	2,6934	2,6934
Blur Multi Scale AO	--	--	3,4573	3,5597	3,5597
Final Blending	--	--	0,1454	0,1578	0,1887
Final Blending Multi Scale AO	--	--	0,3087	0,3102	0,3196

Tabla 9: Resumen tabla comparativa.

La Tabla 10 presenta el resultado total en cuanto a tiempo que se tarda cada técnica en ejecutarse considerando también el resto de los elementos presentes en la escena, es decir, *render* de objetos, *blur* y *blending* final.

Técnica	Escena 1 Punto de vista 1 (ms)	Escena 1 Punto de vista 2 (ms)	Escena 2 Punto de vista 1 (ms)
Alchemy AO	4,4731	4,6525	4,3982
Scalable AO	4,3732	4,3233	4,1164
Unreal Engine AO	4,7834	4,7998	4,5154
Multi Scale AO	5,6935	6,0911	5,5081
Técnica	**Escena 2 Punto de vista 2 (ms)**	**Escena 3 Punto de vista 1 (ms)**	**Escena 3 Punto de vista 2 (ms)**
Alchemy AO	4,5366	4,4027	4,6220
Scalable AO	4,5581	4,3814	4,5354
Unreal Engine AO	5,0717	4,8740	5,0752
Multi Scale AO	6,1351	6,1091	6,3424

Tabla 10: Tiempos totales de render.

Como primera observación se tiene que *Alchemy AO* y *Scalable AO* son las técnicas que ofrecen menor tiempo de ejecución, de hecho, se puede observar que *Scalable AO* cumple en la mayoría de las veces con lo que promete. Recordando que *Scalable AO* se presenta como una optimización de *Alchemy AO* con mejor calidad visual a radios mayores y una menor necesidad de muestras. Por otra parte, se observa que a pesar de que el desarrollo entre *Alchemy, Scalable y Unreal Engine 4* no es tan diferente entre ellos, los resultados de rendimiento obtenidos difieren bastante entre ellos.

Unreal Engine 4 AO presenta un tiempo de ejecución significativamente mayor que *Alchemy* y *Scalable*, esto puede ser debido a que tiene sus origen en la técnica de HBAO la cual se enfoca más en calidad visual que en rendimiento. Por otra parte, hay que recordar que la técnica incluye el *normal buffer* que influye también en el resultado final.

Por último se tiene la técnica de *Multi Scale AO* la cual con solo a partir de la teoría se puede predecir que es una técnica mucho más costosa que las demás debido a que aplica la técnica de *Ambient Occlusion* para 5 resoluciones diferentes. Además *Multi Scale AO* trabaja con cinco diferentes texturas a las cuales se les hace un paso de *blur* y presentan un paso extra en el *blending* final debido a que deben ser combinadas, debido a esto se calcula *blur* y *blending* final en una sección aparte.

Para finalizar observamos que todas las técnicas estudiadas se ejecutan en pocos milisegundos cumpliendo con la capacidad de producir *Ambient Occlusion* en tiempo real. A continuación se presenta la Tabla 11 con la cantidad de cuadros por segundo o *Frames Per Seconds* (FPS) promedio para cada escena.

Técnica	Escena 1 Punto de vista 1 (FPS)	Escena 1 Punto de vista 2 (FPS)	Escena 2 Punto de vista 1 (FPS)
Alchemy AO	137	127	133
Scalable AO	135	128	137
Unreal Engine AO	136	130	131
Multi Scale AO	81	76	77
Técnica	**Escena 2 Punto de vista 2 (FPS)**	**Escena 3 Punto de vista 1 (FPS)**	**Escena 3 Punto de vista 2 (FPS)**
Alchemy AO	125	121	120
Scalable AO	127	119	118
Unreal Engine AO	122	121	113
Multi Scale AO	70	70	67

Tabla 11: Tabla de FPS por escena.

En la Tabla 11 se observa que entre las técnicas de *Alchemy AO, Scalable AO y Unreal Engine 4 AO* la diferencia de *frames* no es tan pronunciada. Sin embargo, si se nota la gran diferencia que tienen al compararse con la técnica de *Multi Scale AO,* lo que sorprende más aún, recordando que solo hay una diferencia de casi dos milisegundos entre su ejecución comparándola con las demás técnicas.

Desde la Figura 67 a la Figura 72 se muestra un resumen de los resultados finales de cada técnica de *Ambient Occlusion* sin texturas para cada una de las escenas, estas imágenes se presentan en el siguiente orden: arriba en la izquierda *Alchemy AO,* arriba a la derecha *Scalable AO*, abajo a la izquierda *Unreal Engine 4 AO* y abajo a la derecha *Multi Scale AO*.

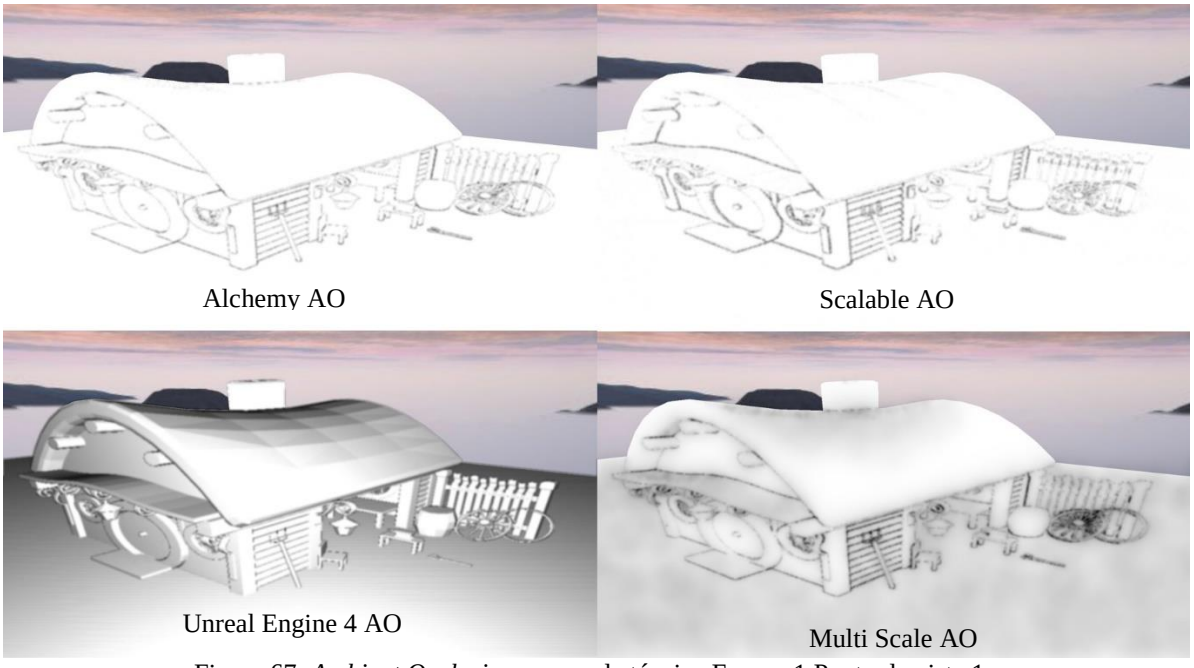


Figura 67: *Ambient Occlusion* para cada técnica Escena 1 Punto de vista 1.

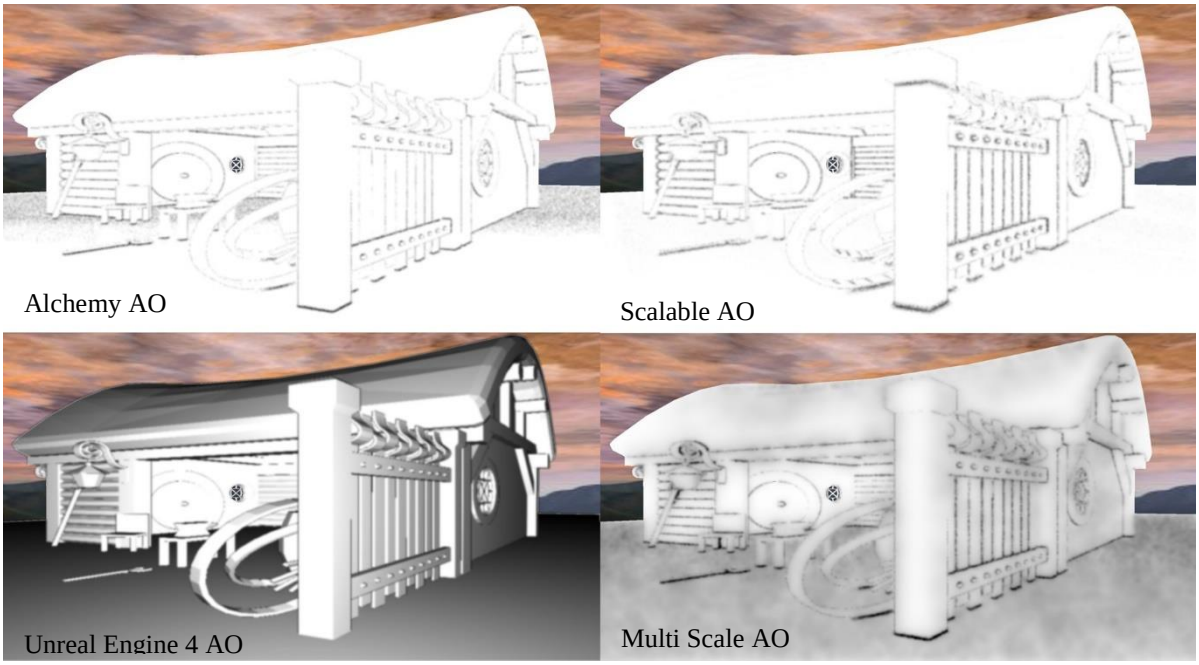


Figura 68: *Ambient Occlusion* para cada técnica Escena 1 Toma 2.

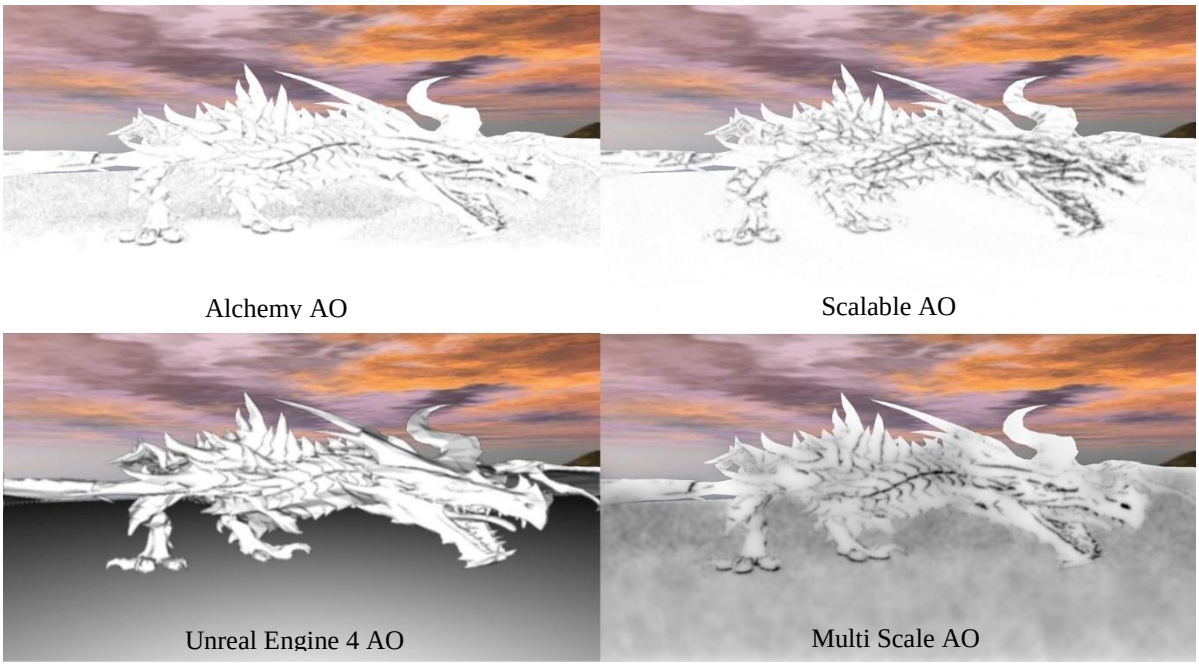


Figura 69: *Ambient Occlusion* para cada técnica Escena 2 Punto de vista 1.

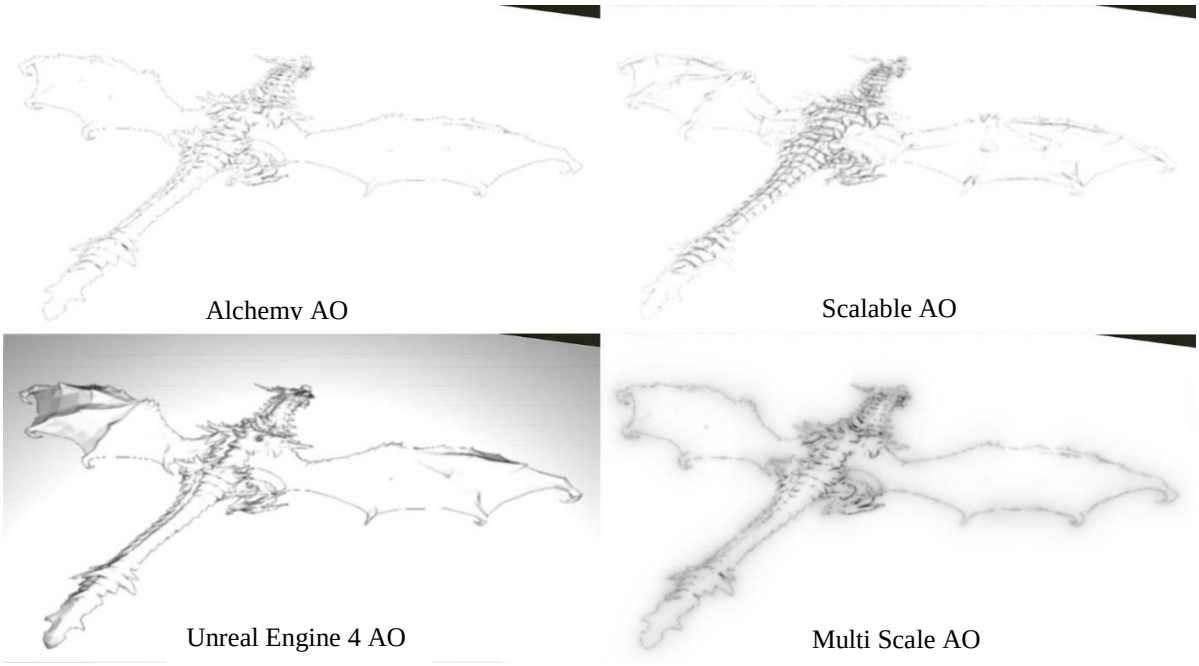


Figura 70: *Ambient Occlusion* para cada técnica Escena 2 Punto de vista 2.

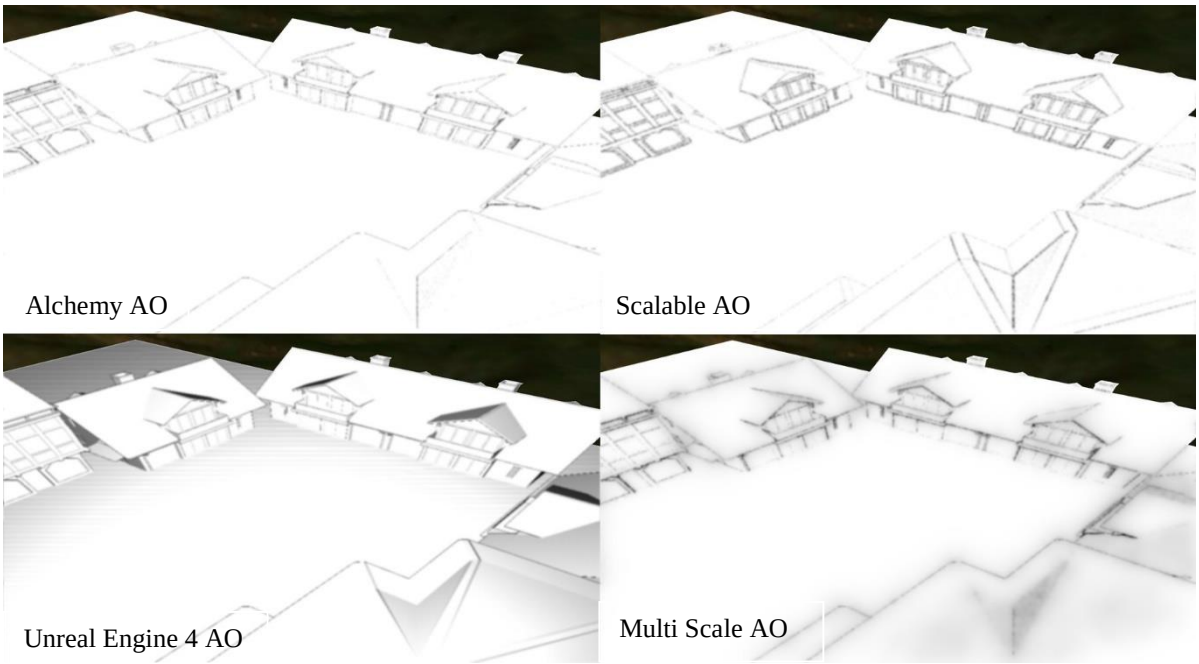


Figura 71: *Ambient occlusion* para cada técnica Escena 3 Punto de vista 1.

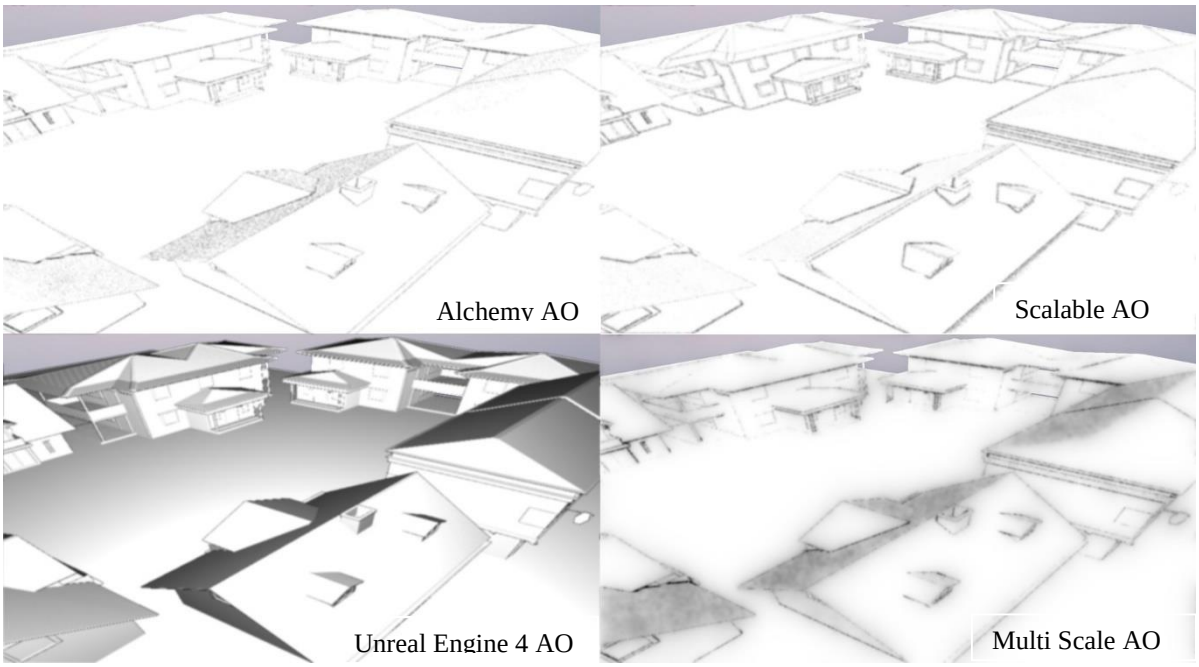


Figura 72: *Ambient occlusion* para cada técnica Escena 3 Punto de vista 2.

Con respecto a los resultados visuales, se observa que *Alchemy AO* desde la Figura 67 a la Figura 72 ofrece un muy buen resultado sobre la escena. Sin embargo, al comparar con las otras técnicas observamos que hay detalles que no se muestran. Cambiar los parámetros puede mejorar los resultados, pero los utilizados son los recomendados por los desarrolladores creadores de las técnicas [3], pudiendo concluir que son los que mejor se adaptan para la mayoría de las escenas. Por otra parte, *Scalable AO* ofrece una mayor representación de los detalles de la escena en comparación con *Alchemy AO*. *Unreal Engine 4 AO* ofrece un resultado visual algo diferente a lo esperado por las técnicas de *Ambient Occlusion* inspiradas en SSAO. De hecho, las sombras detallan muy bien la geometría de la escena, ofrece poco ruido, además de una ambientación más oscura que las demás, ofrece un mejor resultado al agregar texturas a la escena que sin ellas. Por último *Multi Scale AO* permite utilizar cualquier técnica inspirada en SSAO, en este caso se utilizó la técnica de *Alchemy AO* lo que permite observar que la calidad visual de los resultados aumentan considerablemente, debido a que *Multi Scale AO* aplica la técnica de *Ambient Occlusion* a una mayor parte de la escena, es decir, muchos más detalles de la escena son representados por esta técnica al considerar oclusores lejanos al punto de interés. Al observar la mejora visual ofrecida por *Multi Scale* se puede concluir que posee la capacidad de aumentar la calidad visual de los resultados de la mayoría de las técnicas SSAO por no decir que a todas.

5.2.2 Comparación Perceptual

En algunos casos resulta complicado percatarse de las diferencias visuales que existen entre dos imágenes resultantes, entre un par de imágenes con o sin *Ambient Occlusion*. A continuación se presentan una serie de imágenes donde se realizan comparación

perceptual entre una imagen sin *Ambient Occlusion* y otra con *Ambient Occlusion* para cada técnica. La aplicación Resemble.js [12] utilizada para mostrar el porcentaje de diferencia entre imágenes ofrece dos opciones llamadas "*Ignore Nothing*" que toma en cuenta diferencias imposibles de ver con el ojo humano y otra "*Ignore Less*" que son los cambios que se pueden observar con un poco de esfuerzo. Esta prueba se realiza solo sobre el punto de vista uno de la escena uno para cada técnica de estudio. Cada Figura presenta en primer lugar la escena sin *Ambient Occlusion*, seguido de la imagen con la respectiva técnica de *Ambient Occlusion* aplicada, después la imagen *Ignore Less* y por último *Ignore Nothing.*

En la Figura 73 se observa la comparación perceptual de *Alchemy AO* la cual es un buen ejemplo de una técnica difícil de observar a primera vista.

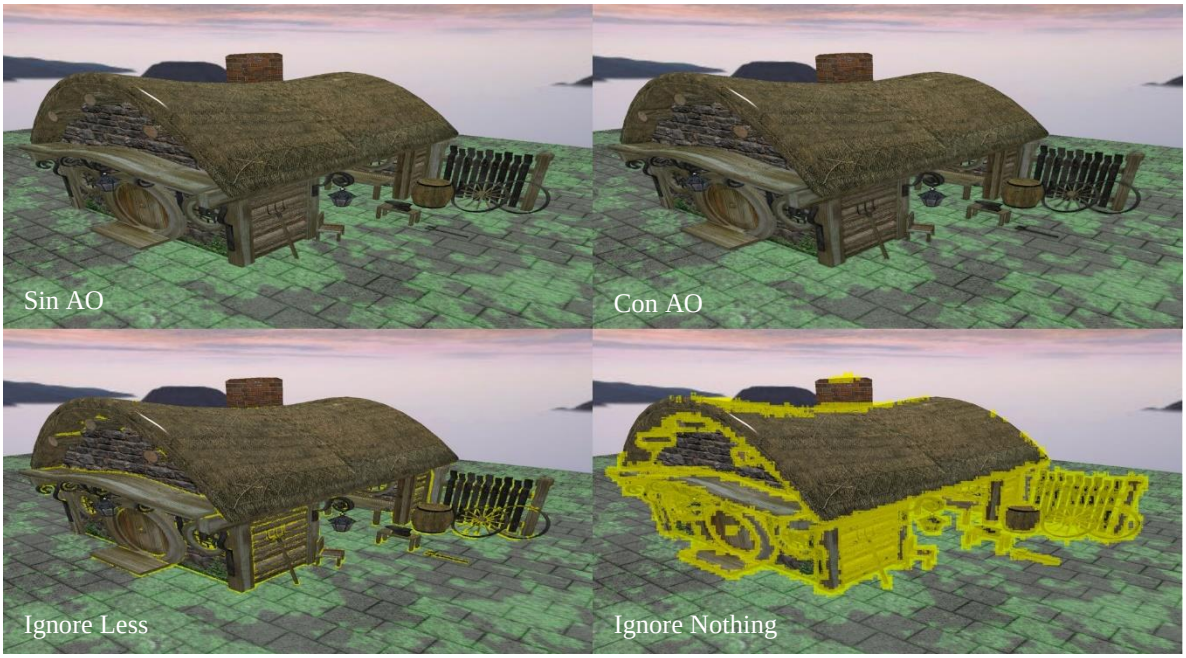


Figura 73: Comparación perceptual Alchemy AO.

Tipo de comparación	Porcentaje de diferencia
Ignore Less	1,80%
Ignore Nothing	17,95%

Tabla 11: Porcentaje de diferencia Alchemy AO entre la imagen sin Ambient Occlusion y con Ambient Occlusion.

De la Figura 73 se entiende que el color amarillo representa las áreas que son diferentes entre cada imagen, para este caso representa las áreas donde se aplica la técnica de *Ambient Occlusion*, en la imagen de *Ignore Nothing* se puede interpretar como un área donde la técnica de *Ambient Occlusion* generó sombras, sin embargo, la sombra producida es mínima por lo que es difícil de ver naturalmente.

En la Figura 74 se muestra la comparación perceptual de la técnica *Scalable AO*, donde cabe destacar el resultado de la imagen *Ignore Nothing* la cual muestra recuadros amarillos esparcidos en la escena, esto puede ser interpretado como resultado que *Scalable AO* trabaja en radios más amplios que *Alchemy AO*, por lo que considera posibles oclusores que se encuentran más alejados del punto de interés, debido a esto la oclusión es mínima.

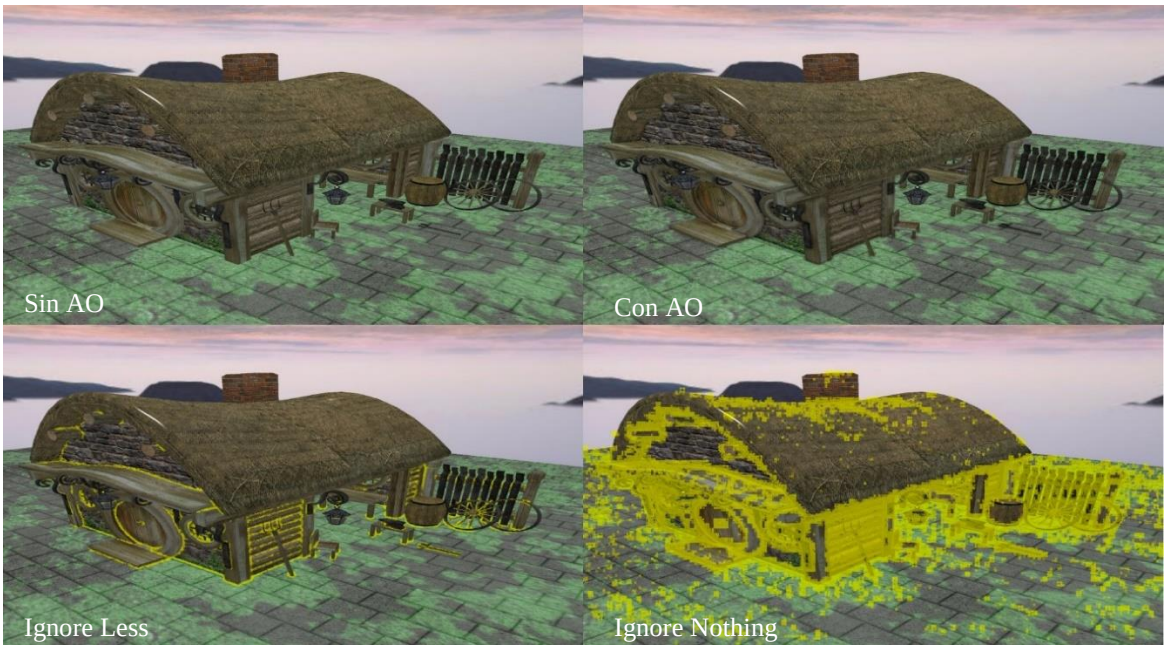


Figura 74: Comparación perceptual Scalable AO.

Tipo de comparación	Porcentaje de diferencia
Ignore Less	2,33%
Ignore Nothing	25,78%

Tabla 12: Porcentaje de diferencia Scalable AO entre la imagen sin Ambient Occlusion y con Ambient Occlusion.

La Figura 75 de *Unreal Engine 4 AO* se puede observar como la escena se hace más oscura mientras más se aleja de la cámara, además de que se observa que la técnica aplica sombras en gran parte de la escena.

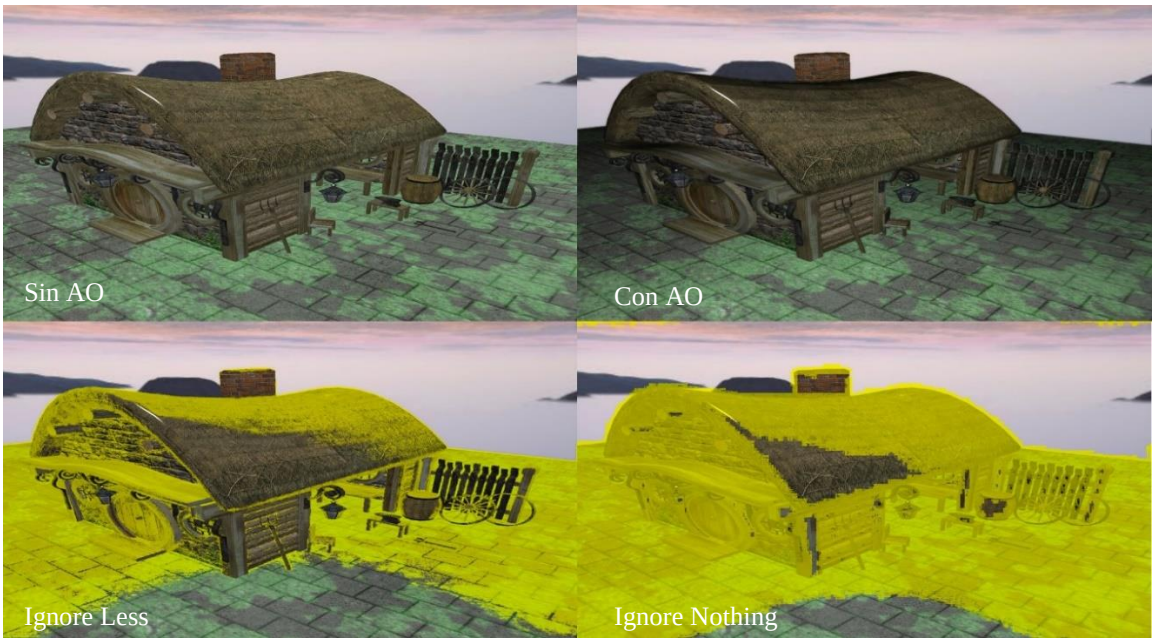


Figura 75: Comparación perceptual Unreal Engine 4 AO.

Tipo de comparación	Porcentaje de diferencia
Ignore Less	38,39%
Ignore Nothing	63,05%

Tabla 13: Porcentaje de diferencia Unreal Engine 4 AO entre la imagen sin Ambient Occlusion y con Ambient Occlusion.

En la Figura 76 se puede observar que la aplicación de la técnica de Multi Scale AO es la que más información abarca de todas las técnicas, esto se debe a que mediante la aplicación de *Ambient Occlusion* a diferentes resoluciones es capaz de obtener mayor información de la escena, debido a que se obtienen oclusores que se encuentran lejos de un punto de interés, debido a esto se obtiene una representación más completa al aplicar un porcentaje de oclusión en la totalidad de la escena.

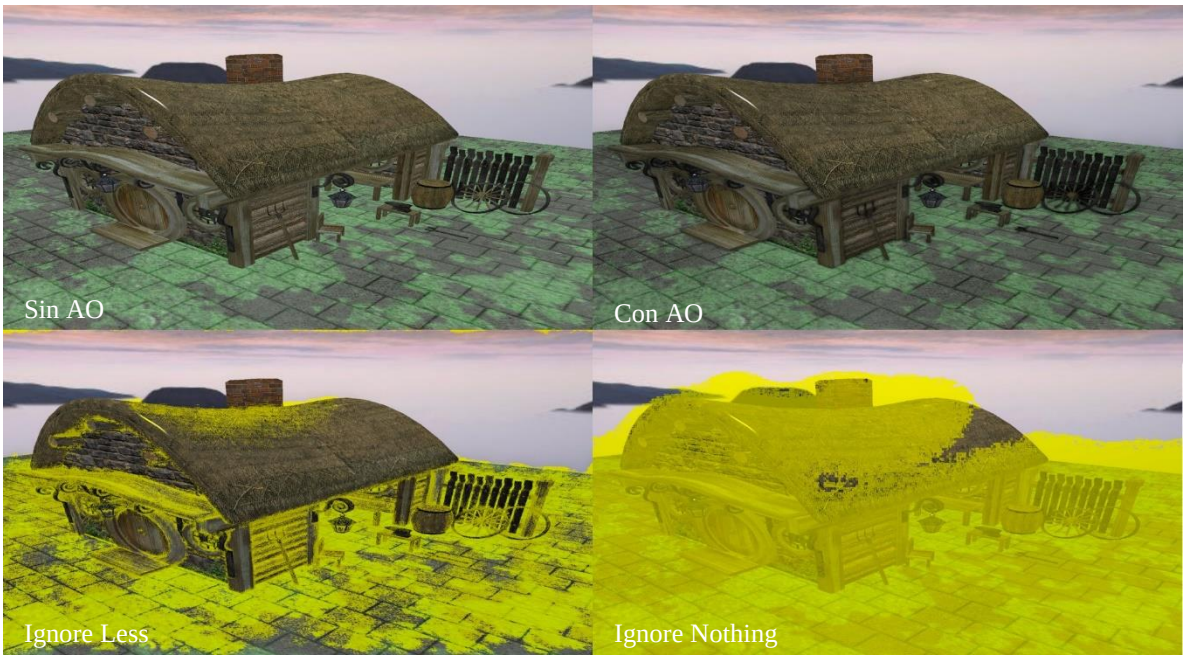


Figura 76: Comparación perceptual Multi Scale AO.

Tipo de comparación	Porcentaje de diferencia
Ignore Less	38,09%
Ignore Nothing	84,04%

Tabla 14: Porcentaje de diferencia Multi Scale AO entre la imagen sin Ambient Occlusion y con Ambient Occlusion.

Con estos resultados se puede observar que en técnicas como *Alchemy AO* y *Scalable AO* en la Figuras 73 y 74 respectivamente, la oclusión está presente en la escena, aunque no siempre es fácil de reconocer. Cada técnica de *Ambient Occlusion* realiza una representación diferente, por ejemplo, la técnica de *Alchemy AO,* se concentra en su totalidad en el objeto de interés, la técnica de *Scalable AO* debido a que trabaja con radios amplios toma en cuenta un poco más de la escena lo cual también influye en el resultado final, *Unreal Engine 4 AO* en la Figura 75 provee mayor oclusión conforme más lejos se encuentra de la cámara, y por último la técnica de *Multi Scale AO* en la Figura 76 debido a que trabaja a diferentes resoluciones es la que representa la escena en casi su totalidad.

5.2.3 Alteración de parámetros

En esta sección se presentan tablas, imágenes y observaciones de los resultados obtenidos al cambiar algunos parámetros de cada técnica para una escena.

5.2.3.1 Alchemy Ambient Obscurance

En primer lugar, se muestra representados en milisegundos, los resultados de rendimiento obtenidos al realizar cambios de parámetros utilizando la técnica de Alchemy AO, en la Tabla 15 se muestran estos resultados donde "s" representa la cantidad de muestras (*Samples*) y "r" representa el radio de muestreo en metros seguido de la comparación de imágenes en la Figura 77.

	s = 8	**s = 16**	**s = 32**
r = 0,50	0,4086	0,6418	1,0567
r = 1,00	0,6392	0,9839	2,1937
r = 1,50	0,6622	1,0075	2,2627
r = 2,00	0,6995	1,1060	2,3769

Tabla 15: Resultados del rendimiento en milisegundos para la parametrización de Alchemy AO.

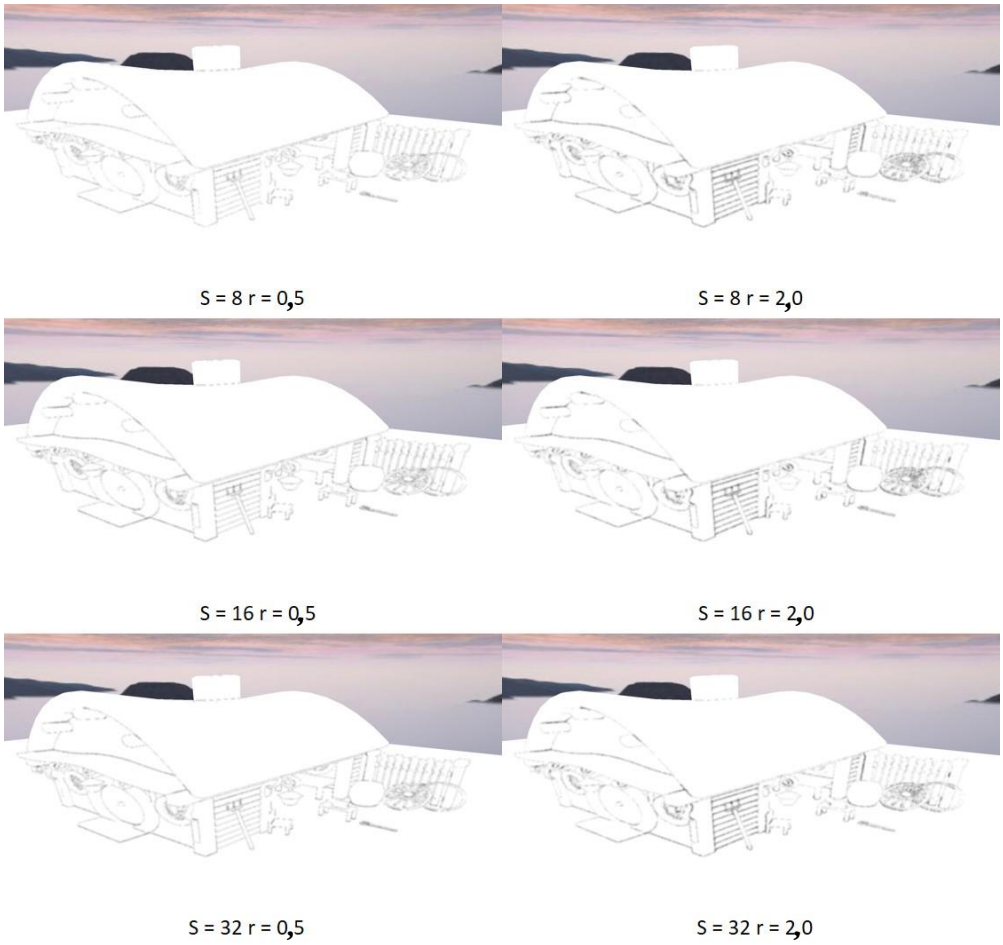


Figura 77: Resultados visuales para la parametrización de Alchemy AO.

De este resultado se puede concluir que menor radio significa menor detalle mientras que mayor radio mayor detalle, sin embargo, ambos resultados son bastante aceptables y cumplen con su función de representar detalles de escena. Así la decisión final viene a ser por parte de rendimiento donde no sólo la cantidad de muestras influyen en el rendimiento sino que también el radio tiene un efecto importante. Por otra parte *Alchemy*

AO no es una técnica que funcione para amplios radios ni muestra cambios significativos al aumentar la cantidad de muestras, estos resultados se muestran en la siguiente sección.

5.2.3.1 Comparación entre extremos de Alchemy AO

En esta sección se mostrará los cambios que suceden entre muy pocas y muchas muestras, un radio pequeño o muy grande, el tiempo que tarda en renderizar y cómo afectan los *frames per seconds (FPS)* en cada caso.

- Comparación entre 8, 500 y 1000 cantidad de muestras, a un radio de 1,0 metro.

Escenas	# Muestras	FPS	Tiempo Promedio (ms)
A	8	142	0,3912
B	500	34	21,1606
C	1000	22	37,3699

Tabla 16: Resultados del rendimiento en milisegundos para la parametrización de Alchemy AO, Comparación entre extremos.

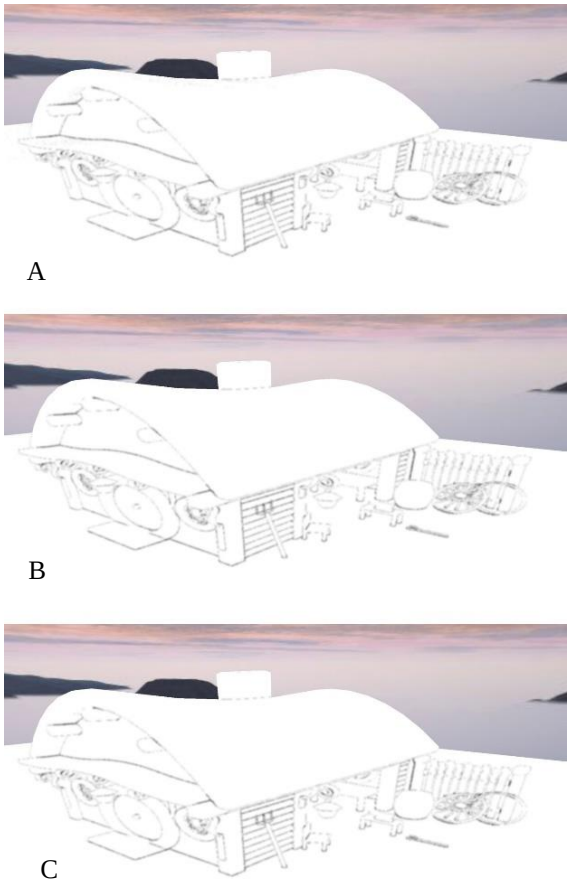


Figura 78: Resultados visuales para la parametrización de Alchemy AO, Comparación entre extremos.

De esta comparación se puede observar inmediatamente que la diferencia entre estos resultados es visualmente no perceptible, por lo que se puede concluir que obtener

mejor calidad de los resultados utilizando una mayor cantidad de muestras tiene un límite, que una vez superado no presenta cambios importantes.

- Comparación entre un radio de 0,5 y 10 metros, ambas con una cantidad de muestras de 16.

Escenas	Radio(m)	FPS	Tiempo Promedio (ms)
A	0,5	138	0,6419
B	10,0	119	1,4889

Tabla 17: Resultados del rendimiento en milisegundos para la parametrización de Alchemy AO, Comparación entre extremos.

A B

Figura 79: Resultados visuales para la parametrización de Alchemy AO, Comparación entre extremos.

De esta comparación se puede concluir que *Alchemy AO* no funciona bien para amplios radios, después de un cierto punto se empieza a perder definición de detalles. Sin embargo, el algoritmo sigue trabajando, de ahí que aunque no se observe algo, afecta el rendimiento.

5.2.3.2 Scalable Ambient Obscurance

En esta sección se muestra representados en milisegundos, los resultados de rendimiento obtenidos al realizar variación de parámetros utilizando la técnica de *Scalable AO*, en la Tabla 18 se muestran estos resultados donde "s" representa la cantidad de muestras (*Samples*) y "r" representa el radio de muestreo en metros, seguido de la comparación de imágenes.

	s = 4	s = 8	s = 16
r = 2,50	0,3921	0,5298	0,9288
r = 5,00	0,4064	0,7423	1,4100
r = 10,00	0,4760	0,8840	1,6300
r = 20,00	0,6341	1,0890	1,7915

Tabla 18: Resultados del rendimiento en milisegundos para la parametrización de Scalable AO.

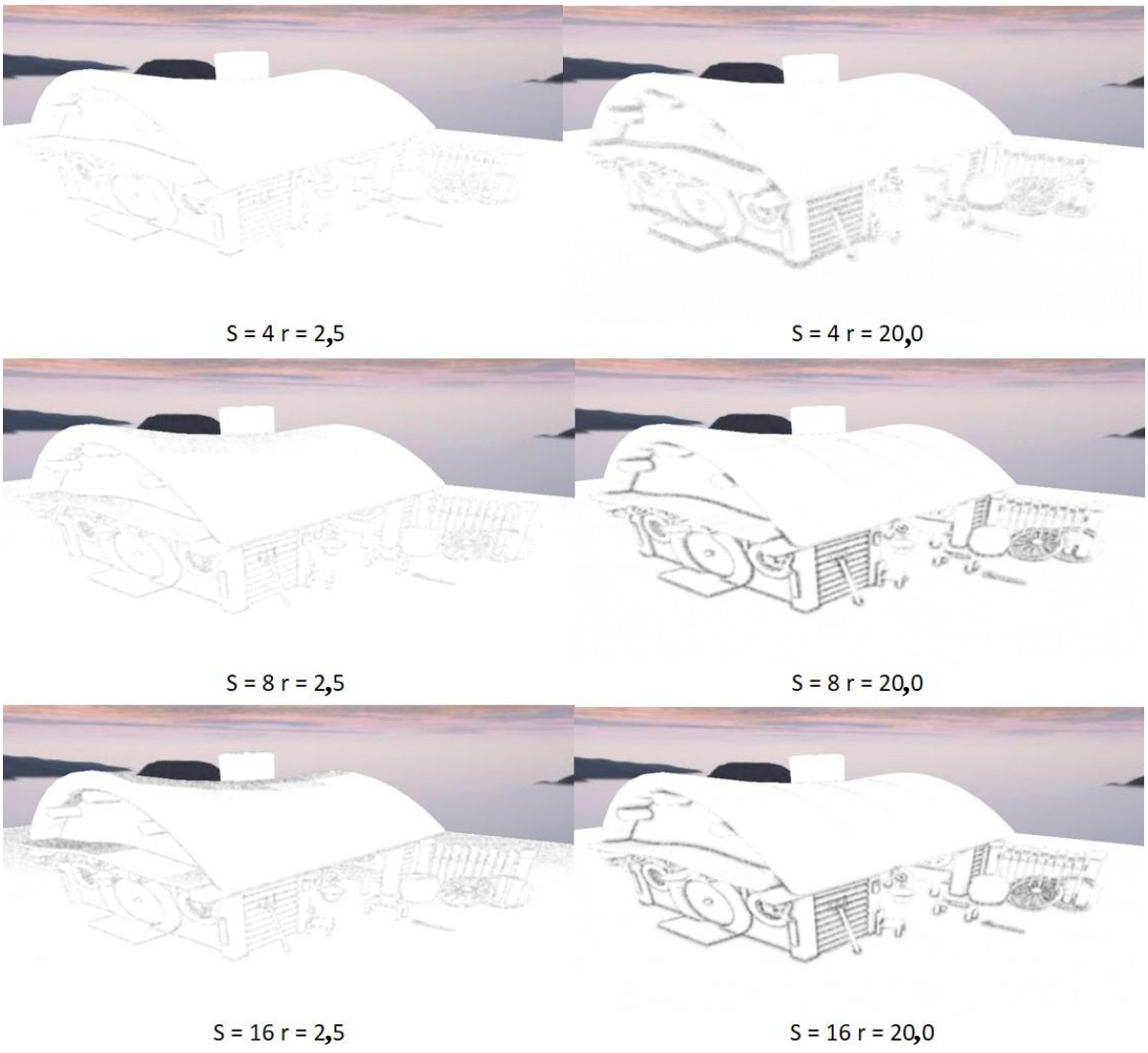


Figura 80: Resultados visuales para la parametrización de Scalable AO.

De este resultado se puede observar que *Scalable AO* no genera buenos resultados a pequeños radios, pero si muestra resultados bastante aceptables a amplios radios. Por otra parte, se debe encontrar un equilibrio entre la cantidad de muestras y el tamaño del radio debido a que amplios radios y pocas muestras generan ruido. También se debe tener en cuenta que amplios radios generan largas sombras que posiblemente no sean las adecuadas para la aplicación a realizar en un determinado momento.

5.2.3.2 Comparación entre extremos de Scalable AO

En esta sección se mostrarán los cambios que suceden entre muy pocas y muchas muestras, un radio pequeño o muy grande, el tiempo que tarda en realizar un render y cómo afectan los *frames per seconds (FPS)* en cada caso.

- Comparación entre 4, 100, 500 y 1000 cantidad de muestras, a un radio de 10,0 metros.

Escenas	Muestras	FPS	Tiempo Promedio (ms)
A	4	131	0,3987
B	100	64	8,4781
C	500	20	42,3924
D	1000	11	79,9984

Tabla 19: Resultados del rendimiento en milisegundos para la parametrización de Scalable AO, Comparación entre extremos.

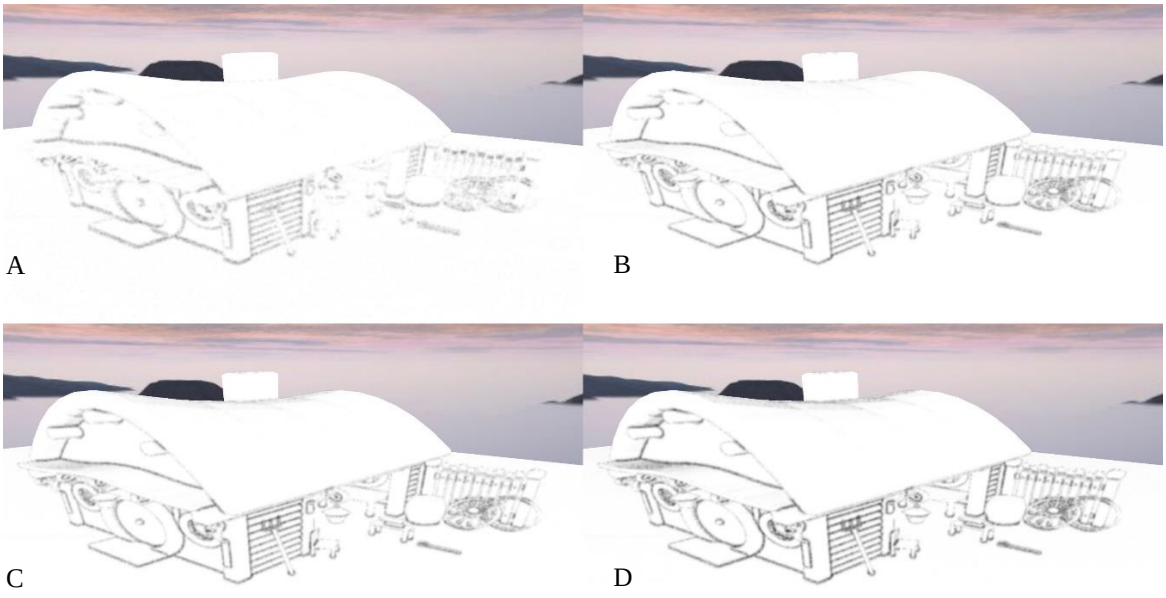


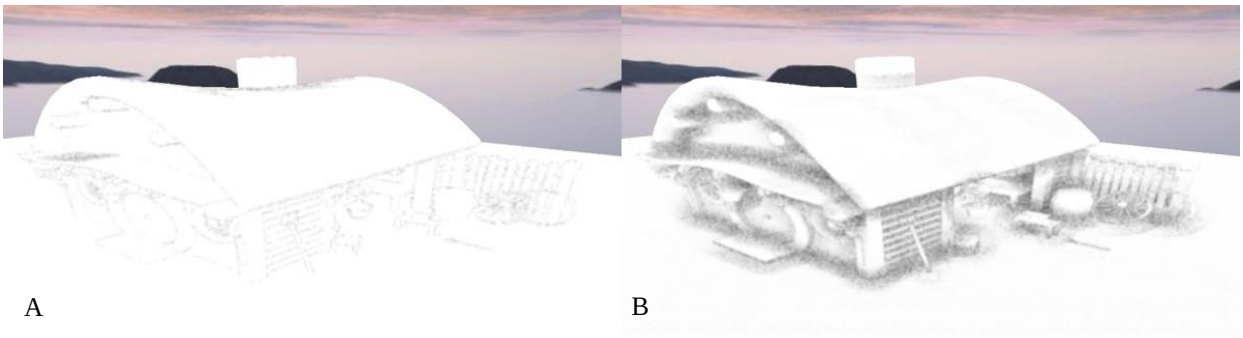
Figura 81: Resultados visuales para la parametrización de Scalable AO, Comparación entre extremos.

De esta comparación se puede observar que al tener pocas muestras en un amplio radio se obtiene mucho ruido, donde la técnica de *blur* es incapaz de reducir. Sin embargo, es un resultado aceptable debido a que cumple con resaltar los detalles de la escena, además al incluir texturas se nota aún menos. Por otra parte, a diferencia de *Alchemy AO*, en este caso sí se observa una mejora considerable con respecto a la calidad visual al aumentar la cantidad de muestras, pero a un costo mucho mayor. Cabe también destacar que esta técnica trabaja con grandes radios, lo cual tiene impacto en el rendimiento.

- Comparación entre un radio de 1,0 y 100,0 metros, ambas con una cantidad de muestras de 8.

Escenas	Radio(mts)	FPS	Tiempo Promedio (ms)
A	1,0	129	0,4429
B	100,0	100	2,4319

Tabla 20: Resultados del rendimiento en milisegundos para la parametrización de Scalable AO, Comparación entre extremos.



A B

Figura 82: Resultados visuales para la parametrización de Scalable AO, Comparación entre extremos.

En esta comparación se observa en la Figura 82 como *Scalable AO* no funciona muy bien con radios pequeños, pero ofrece un mejor resultado con radios amplios. Sin embargo, se observa que poseen mucho ruido debido a la poca cantidad de muestras utilizadas, debido a esto es necesario encontrar un equilibrio entre el radio y la cantidad de muestras. Por otra parte, incluso con ruido es un resultado bastante aceptable, todo depende del tipo de sombras se necesitan para una aplicación.

5.2.3.2.2 Comparación entre Alchemy AO y Scalable AO

Las técnicas de *Alchemy AO* y *Scalable AO* están relacionadas debido a que *Scalable AO* se presenta como una optimización de *Alchemy AO*, sin embargo, esto no siempre es verdad, de hecho, a partir de los resultados obtenidos de las pruebas que se han realizado en secciones anteriores, se puede concluir que *Alchemy AO* es mejor para radios pequeños y *Scalable AO* para radios grandes, aun así es necesario comparar de cerca estas dos técnicas y observar los resultados.

En la Tabla 21 se expresan diferentes resultados de pruebas de rendimiento realizadas para las técnicas de *Alchemy AO* y *Scalable AO*.

# Muestra	Radio(m)	Alchemy AO(ms)	Scalable AO(ms)
4	0,5	0,2172	0,2805
8	0,5	0,3263	0,4385
16	0,5	0,5383	0,7491
100	0,5	2,7501	4,0134
4	1,0	0,2327	0,2805
8	1,0	0,3893	0,4379
16	1,0	0,7900	0,7503
100	1,0	4,7482	4,0381
4	10,0	0,4225	0,3983
8	10,0	0,7768	0,7198
16	10,0	1,4930	1,3671
100	10,0	9,0017	8,0849

Tabla 21: Resultado de pruebas de rendimiento

Cabe destacar que las pruebas se realizaron entre los radios 0,5 y 10,0 esto se debe a que el resultado visual de *Scalable AO* en radios muy pequeños es desfavorable, es decir, se ve muy poco y muy claro. Por otra parte, *Alchemy AO* desde un radio de 10,0 comienza a aclararse siendo muy difícil ver resultado alguno como se ha observado en pruebas anteriores.

De la Tabla 21 se puede observar que el rendimiento de *Alchemy AO* es mejor que el de *Scalable AO* a radios pequeños mientras que a radios grandes el resultado de *Scalable AO* es mejor que el de *Alchemy AO*. Sin embargo, suceden casos particulares como los observados en las filas marcadas de azul, donde a pesar de ser un radio pequeño al aumentar la cantidad de muestras se observa que *Scalable AO* ofrece un mejor rendimiento, a partir de esto se puede concluir que es más eficiente en el uso de una mayor cantidad de muestras.

5.2.3.3 Unreal Engine 4 Ambient Occlusion

En primer lugar, se muestra representados en milisegundos, los resultados de rendimiento obtenidos al realizar cambios de parámetros utilizando la técnica de *Unreal Engine 4*, en la Tabla 22 se muestran estos resultados donde "s" representa la cantidad de muestras (*Samples*) y "r" representa el radio de muestreo en metros, seguido de la comparación de imágenes.

	s = 4	**s = 8**	**s = 16**
r = 0,05	0,7056	1,2942	2,3628
r = 0,25	0,7622	1,3019	2,3860
r = 0,50	0,8147	1,3258	2,4127
r = 1,00	0,8367	1,3905	2,4359

Tabla 22: Resultados del rendimiento en milisegundos para la parametrización de Unreal Engine 4 AO.

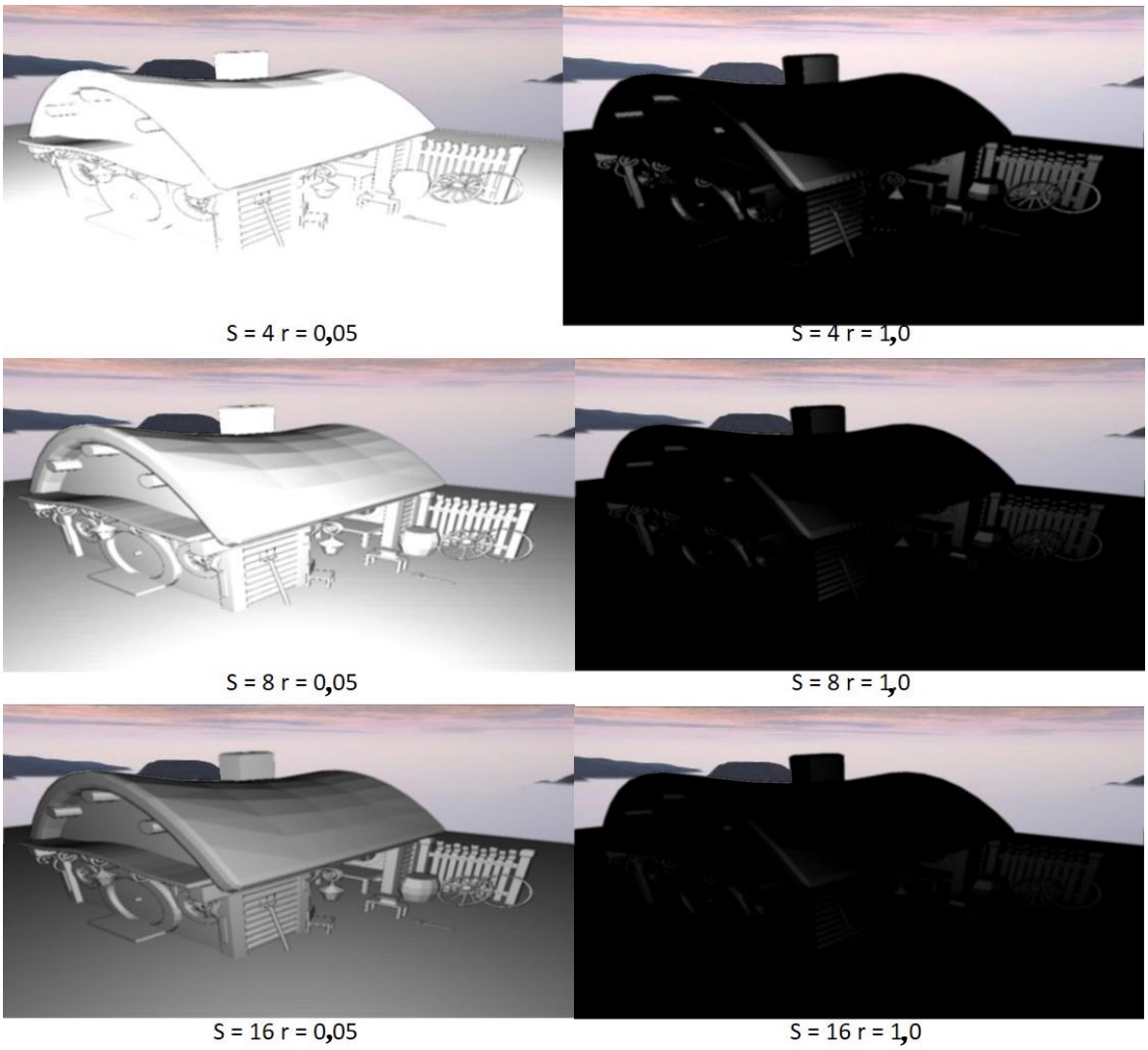


Figura 83: Resultados visuales para la parametrización de Unreal Engine 4.

De este resultado se puede observar en la Figura 83 que la técnica de *Unreal Engine 4 AO* obtiene un mejor resultado en radios pequeños, en radios grandes se obtiene un resultado muy oscuro, que se debe a algún horizonte lejano que afecta en gran medida los resultados locales. Por otra parte, al aumentar la cantidad de muestras se observa una representación cada vez más completa de la escena además de que la hace cada vez más oscura, pero a un paso más lento que al aumentar el tamaño del radio.

En la técnica de *Unreal Engine 4 AO*, mientras más muestras utilicen, ofrece una representación cada vez más detallada de la escena. Por otra parte, incluso con pocas muestras y en el pequeño radio que necesita, obtiene resultados bastante favorables acompañados de buen rendimiento, recordando que mayor muestra y mayor radio afectan rendimiento tal como se observa en la Tabla 22.

5.2.3.4 Multi-Scale Ambient Occlusion

En primer lugar, se muestra representados en milisegundos, los resultados de rendimiento obtenidos al realizar cambios de parámetros utilizando la técnica de *Multi-Scale AO*, en la Tabla 23 se muestran estos resultados donde "s" representa la cantidad de muestras (*Samples*) y "r" representa el radio de muestreo en metros, seguido de la comparación de imágenes.

	s = 8	**s = 16**	**s = 32**
r = 0,50	0,4761	0,7774	1,3665
r = 1,00	0,5604	1,0558	1,9972
r = 1,50	0,7644	1,2773	2,4299
r = 2,00	0,7972	1,4198	2,6944

Tabla 23: Resultados del rendimiento en milisegundos para la parametrización de Multi-Scale AO.

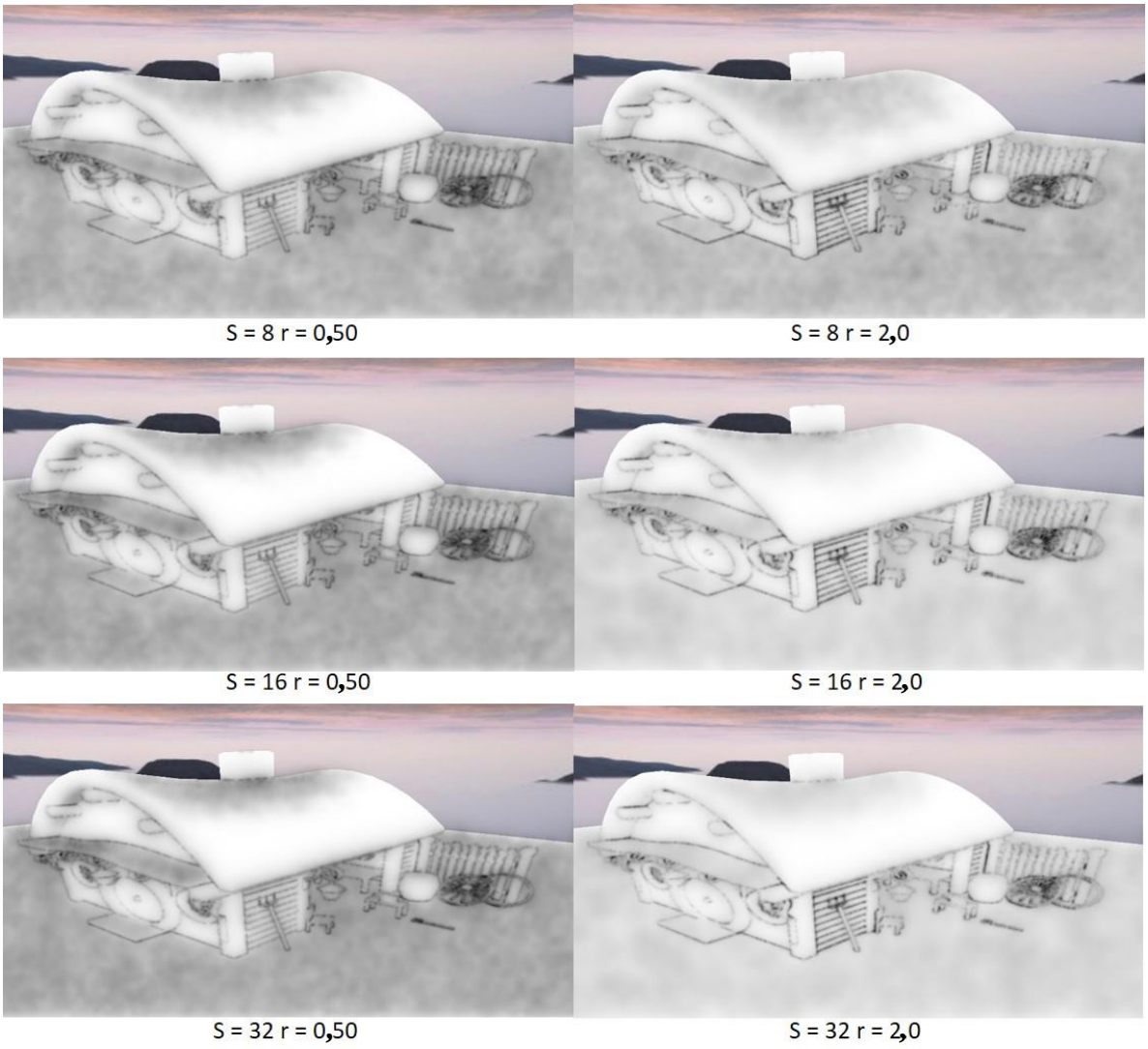


Figura 84: Resultados visuales para la parametrización de Multi-Scale AO.

De este resultado se observa en la Figura 84 que tomando en cuenta sólo el resultado de la técnica de *Multi Scale AO*, se obtiene un buen rendimiento que incluso es mucho mejor en varios casos que los obtenidos por la técnica de *Unreal Engine 4 AO*, y no está muy alejado del resultado obtenido por las técnicas de *Alchemy AO* y *Scalable AO*. Con respecto al resultado visual, se logra obtener una gran representación de los objetos en escena, la capacidad de aplicar *Ambient Occlusion* a diferentes resoluciones permite obtener más información de los detalles de escena, obteniendo así muy buenos resultados visuales debido a una representación más completa de la escena. Sin embargo, se debe recordar que para este caso *Multi Scale AO* está empleando como base la técnica de *Alchemy AO*, por lo que acarrea los problemas de esta técnica, tal como que no es buena para amplios radios.

5.2.3.4 Texturas de Multi-Scale AO

A continuación se muestran por separado las texturas que al mezclarse conforman la técnica de *Multi Scale AO*. Los resultados obtenidos son a resolución 1920x1080 píxeles, las texturas que componen esta técnica son la división entre dos de esta resolución cinco veces. Así, se calculó *Ambient Occlusion* para las siguientes resoluciones: 1920x1080, 960x540, 480x270, 240x137 y 120x68 píxeles. En la Tabla 24 se muestra el tiempo promedio que tarda en aplicar *Ambient Occlusion* para cada resolución.

Resolución(píxeles)	Tiempo Promedio (ms)
1920x1080	2,1299
960x540	0,3632
480x270	0,0729
240x137	0,0299
120x68	0,0170
Total	2,6129

Tabla 24: Resultados de rendimiento al aplicar Multi-Scale AO en cada resolución.

Como es de esperarse mientras mayor es la resolución más tarda en calcular. A continuación se presenta una serie de imágenes donde se observa cada textura por separado seguido de la combinación de todas ellas.

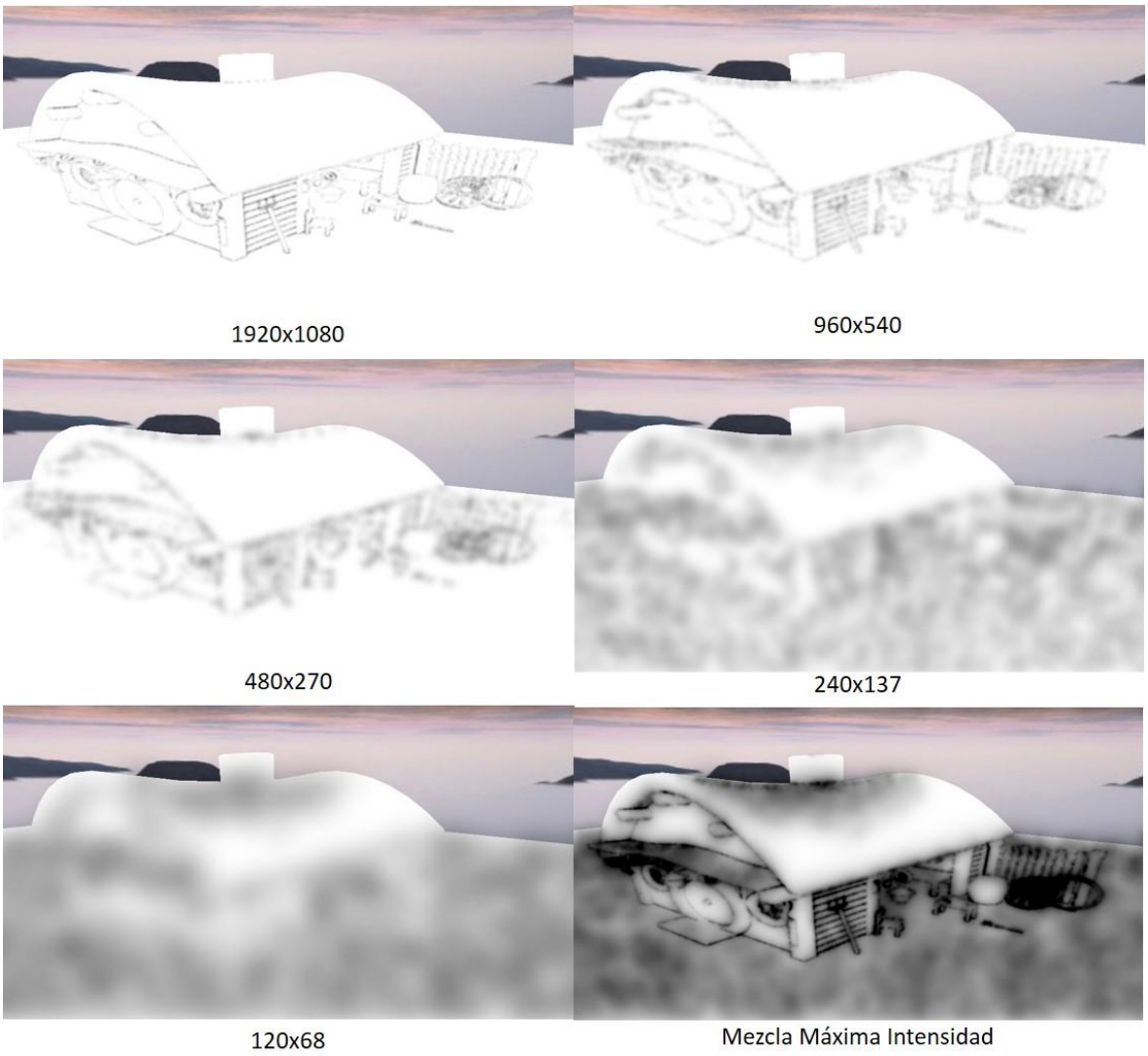


Figura 85: Resultados Multi-Scale AO Separado por texturas.

En la Figura 86 se puede observar como al reducir la resolución aumenta el área donde se aplica *Ambient Occlusion*, además pierde definición de la escena. Por otra parte, en la imagen final se observa una combinación de las texturas en caso de combinarlas a su máxima intensidad, este resultado no siempre es el deseado, por lo que permite disminuir la intensidad de cada textura hasta conseguir el requerido. A continuación se presenta una comparación entre el resultado obtenido al combinar las texturas a máxima intensidad, y un resultado con las intensidades para cada textura mostrada en la Tabla 25**.**

Resolución (píxeles)	Intensidad
1920x1080	1,0
960x540	0,5
480x270	0,3
240x137	0,3
120x68	0,3

Tabla 25: Multi-Scale AO intensidades por texturas.

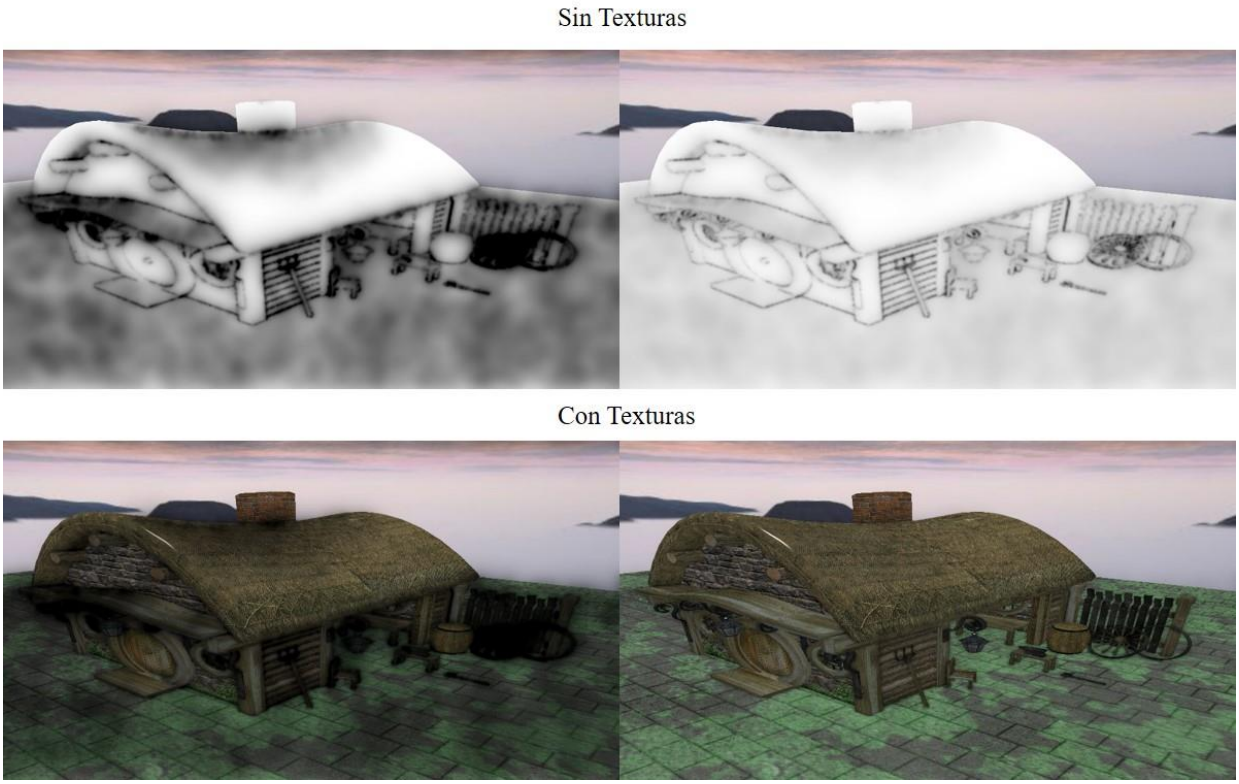


Figura 86: Multi-Scale AO Comparación entre resultados finales.

Esta es solo una de las diferentes posibilidades que se pueden obtener al combinar las texturas a diferentes intensidades, dependiendo de la aplicación a realizar puede ser preferible un resultado a máxima intensidad o no. En el ejemplo de la Figura 86 se observa partes de la escena donde hay mucha oclusión, hay mucha oscuridad concentrada, además de manchas negras lo cual podría ser un resultado no deseable.

5.2.4 Comparación a diferentes resoluciones

Por último, es necesario realizar una última prueba sobre las diferentes técnicas de *Ambient Occlusion* que consiste en comparar cada técnica a diferentes resoluciones. La resolución a la que se aplica *Ambient Occlusion* no necesariamente es la misma que la resolución que se muestra al final en pantalla, para estas pruebas se aplica *Ambient Occlusion* a resolución de 3840x2160 píxeles conocida como 4K o Ultra HD, en 1920x1080 píxeles o Full HD y en 1280x720 píxeles o HD. Todos estos resultados se muestran a una resolución de pantalla de 1920x1080 píxeles.

A continuación se presenta la Tabla 26 donde se observa el tiempo promedio y los *frames per seconds (FPS)* de cada técnica de *Ambient Occlusion*, seguido por una serie de imágenes donde se observan las diferencias entre cada técnica en cada resolución.

Técnicas	3840x2160(ms)	FPS	1920x1080(ms)	FPS	1280x720(ms)	FPS
Alchemy AO	4,3935	48	0,7774	125	0,2514	179
Scalable AO	3,1515	50	0,7287	124	0,2974	176
Unreal Engine AO	4,0332	48	1,0094	118	0,4638	163
Multi Resolution AO	6,6507	32	1,3729	71	0,5499	91

Tabla 26: Resultados rendimiento a diferentes resoluciones.

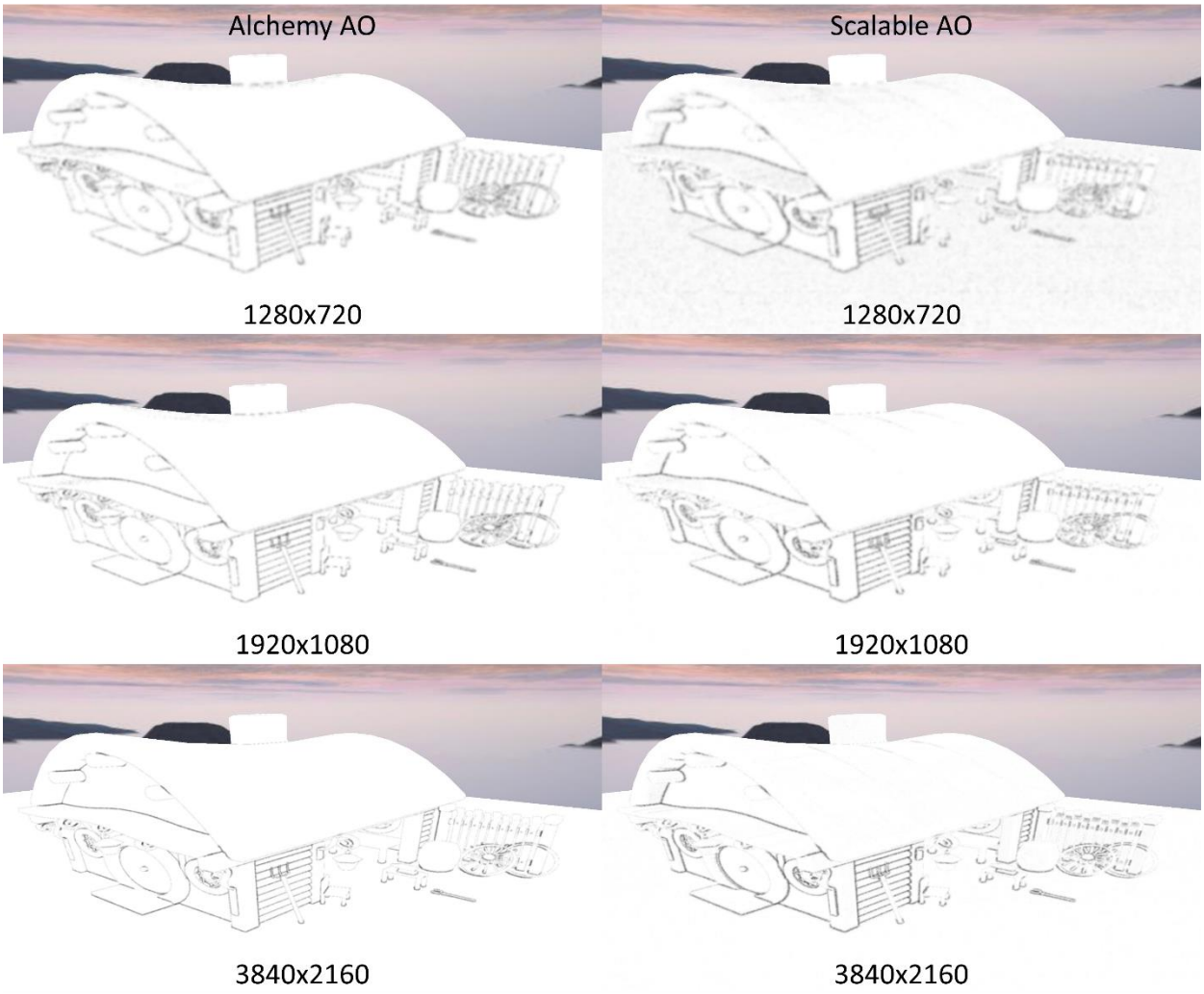


Figura 87: Comparación entre resoluciones medidas en píxeles. En la izquierda Alchemy AO y en la derecha Scalable AO

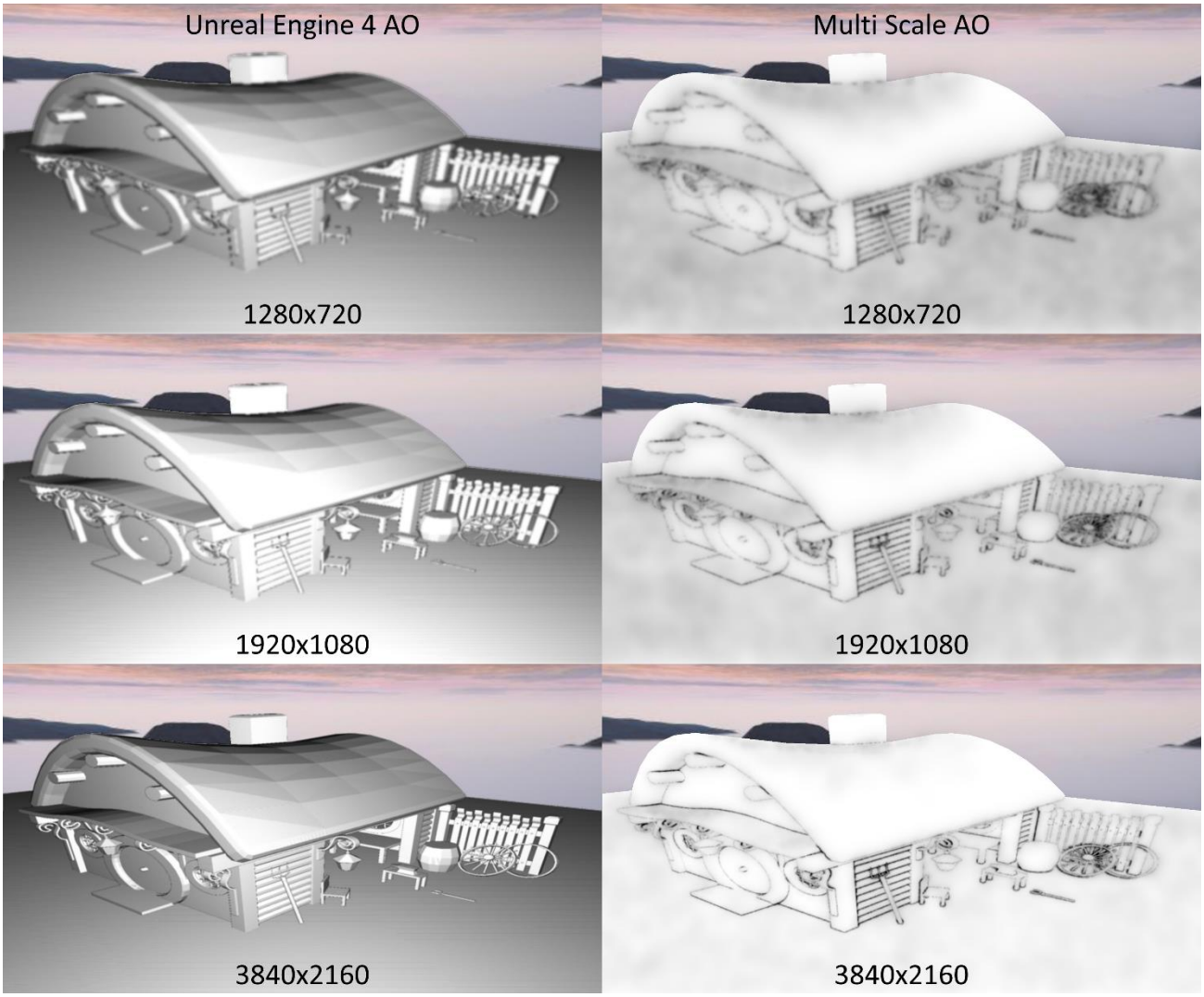


Figura 88: Comparación entre resoluciones medidas en píxeles. En la izquierda Unreal Engine 4 AO y en la derecha Multi Scale AO

A partir de estos resultados, se puede inferir que a mayor resolución existe un mayor detalle y definición de la escena, sin embargo, a mayor resolución menor rendimiento. También se puede observar que a pesar de aplicar *Ambient Occlusion* a una resolución significativamente menor a la resolución de pantalla, el resultado final es bastante bueno. De hecho, en la mayoría de guías y libros donde se explican estas técnicas asumen que se aplicará *Ambient Occlusion* a una resolución menor a la de pantalla [14]. Por otra parte, las aplicaciones que utilizan estas técnicas de *Ambient Occlusion* no permiten modificar la resolución a la que se calcula, es decir, se asume que se calcula a una resolución menor recordando que el objetivo principal es aplicar *Ambient Occlusion* en tiempo real.

Capítulo 6

Conclusiones y trabajo futuros

En este trabajo se presentó una aplicación, la cual permite incluir cuatro diferentes técnicas de *Ambient Occlusion* en tiempo real basadas en la técnica de *Screen-Spaced Ambient Occlusion* (SSAO) desarrollada por Vladimir Kajalin a una escena determinada. Esta implementación fue desarrollada para realizar pruebas y comparaciones sobre las técnicas de *Alchemy Screen-Space Ambient Obscurance*, *Scalable Ambient Obscurance*, *Unreal Engine 4 Ambient Occlusion* y la técnica de *Multi-Scale Ambient Occlusion.* Con la finalidad de conocer sus ventajas, desventajas, rendimiento, *frames per seconds* (FPS), y calidad visual de cada técnica en diferentes escenas, al utilizar diferentes parámetros, y a diferentes resoluciones. También se realizó comparaciones perceptuales, debido a que en ocasiones es difícil notar las diferencias a simple vista.

Desde su creación la técnica de *Alchemy AO* representó un gran avance en el desarrollo de *Ambient Occlusion* en tiempo real, contando con ventajas que permiten que pueda ser considerada incluso hoy en día, debido a su capacidad de representar los detalles de la escena a un muy buen rendimiento, recordando que esta técnica fue desarrollada en el año 2011. Por otra parte, la técnica de *Scalable AO* que se desarrolló como una optimización de *Alchemy AO*, en la mayoría de las veces mejora la calidad visual al resaltar aún más detalles de la escena, siendo más eficiente para trabajar con grandes cantidades de muestras a grandes radios y altas resoluciones. Por otro lado, la técnica de *Unreal Engine 4* a pesar de trabajar con pocas muestras presenta poco ruido en la mayor parte de la escena, sin embargo, a elevados radios se observa una mayor oscuridad en general sobre toda la escena, lo cual no siempre es adecuado para cada aplicación. Finalmente la técnica de *Multi-Scale Ambient Occlusion* es la técnica que mejor representa la oclusión presente, debido a que al trabajar a diferentes resoluciones y mezclar resultados obtiene una mayor y más completa información sobre la escena, permitiendo obtener una mejor representación de la misma. Por otra parte, este incremento de calidad viene con un costo asociado en rendimiento, sin embargo, a pesar del costo sigue siendo una técnica que puede ser utilizada en aplicaciones en tiempo real, debido a que ofrece más de setenta *frames per seconds* a una resolución Full HD de 1920x1080 píxeles.

Al realizar comparaciones entre las diferentes técnicas se concluye que en la mayoría de los casos *Scalable AO* cumple en ser una optimización de la técnica de *Alchemy AO*, sin embargo, existen casos particulares donde *Alchemy AO* resulta ser más eficiente. Por otra parte, la técnica de *Alchemy AO* trabaja mucho mejor a radios pequeños, mientras que *Scalable AO* es mejor para radios grandes. La técnica de *Unreal Engine 4 AO* debido al uso del *normal buffer* representa en gran medida la geometría de la escena, también se observó que dependiendo del punto de vista posibles errores aparecen en el resultado los cuales al realizar el pase de *blur* se resuelven considerablemente, sin embargo, siguen presente en la escena. *Unreal Engine 4 AO* es una técnica que provee muy buenos resultados con muy pocas muestras y trabaja mucho mejor con radios muy pequeños. *Multi-Scale AO* es la técnica que ofrece mayor flexibilidad al permitir combinarse con cualquier técnica basada en SSAO

aumentando en gran medida el resultado visual de la técnica seleccionada, debido a que al aplicar la técnica en 5 diferentes resoluciones se obtiene mucha más información sobre los oclusores presente en la escena para cada punto de interés, obteniendo una mayor y más completa representación de la misma.

Se puede intuir que a mayor cantidad de muestras mejor es la calidad de los resultados, sin embargo, una vez superado un límite no presenta cambios significativos. De igual manera aumentar el tamaño del radio permite obtener más información de la escena, sin embargo, dependiendo de la técnica, el tamaño del radio también tiene un límite que una vez superado presenta resultados que puede desfavorables para una aplicación en particular. Por otra parte, al realizar pruebas sobre diferentes resoluciones se observó que a menor resolución menor calidad visual, pero mejor rendimiento y a mayor resolución mayor calidad visual, pero menor rendimiento. Debido a esto es necesario definir los requerimientos de la aplicación a realizar, de tal forma que se pueda escoger la resolución para aplicar la técnica seleccionada de *Ambient Occlusion* que ofrezca los mejores resultados.

Actualmente existen muchas optimizaciones para las diferentes técnicas de *Ambient Occlusion.* Sin embargo, las técnicas implementadas en este trabajo contienen solo las optimizaciones que forman parte de la técnica en sí, esto se debe a que algunas optimizaciones fácilmente pueden ser tema de un trabajo aparte. Una de las optimizaciones tocadas en este trabajo, es que se puede calcular la técnica de *Ambient Occlusion* a una resolución menor a la mostrada en el render final y otra es el uso de la función de atenuación, la cual reduce la cantidad de muestras utilizadas dependiendo de qué tan lejos se encuentra el punto de interés en la escena.

Se recomienda en un trabajo posterior la implementación de las siguientes optimizaciones: el uso de bandas de guardia (*Guard Band*) que permite tomar en cuenta la geometría que aporta oclusión encontrada fuera de pantalla, esto a su vez significa aumentar la resolución de cálculo de *Ambient Occlusion*, usualmente en un cinco por ciento. Otra optimización es *Temporal Coherence* que consiste en reutilizar información de *frames* anteriores y combinarlas con el frame actual, lo que mejora rendimiento y calidad de imagen. Otra técnica es concentrar el cálculo de *Ambient Occlusion* en el centro de la escena y a medida que se alejan del centro utilizar menos muestras, esto alegando que el usuario enfoca su mirada en el centro de la pantalla la mayor parte del tiempo que se usa una aplicación, de esta forma se obtiene una gran calidad visual en el centro y al reducir las muestras alrededor aumenta el rendimiento. Y otra técnica es calcular *Ambient Occlusion* a partir de diferentes puntos de vista, lo que permite reconocer oclusores que no se observan desde el punto de vista principal.

Bibliografía

[1] De Vries, J. Screen Space Ambient Occlusion (SSAO). [En línea]. https://learnopengl.com/#!Advanced-Lighting/SSAO

[2] Bavoil, L. y Sainz, M. Image-Space. Horizon-Based Ambient Occlusion. [En línea]. https://www.geforce.com/hardware/technology/hbao-plus/technology

[3] McGuire, M., Osman, B., Bukowski, M. y Hennessy, P. "The Alchemy Screen-Space Ambient Obscurance Algorithm". High-Performance Graphics. 2011.

[4] McGuire, M., Mara, M. y Luebke, D. "Scalable Ambient Obscurance". High-Performance Graphics. 2012.

[5] Hoang, T.-D., Low, K.-L. "Efficient Screen-Space Approach to High-Quality Multi-Scale Ambient Occlusion". National University of Singapore. 2011.

[6] Mittring, M. "The technology behind the unreal engine 4 Elemental demo". ACM SIGGRAPH 2012 Computer Animation Festival, pp. 86-86. 2012.

[7] Snow, B. ILM, Industrial Light & Magic. 2001. [En Linea] https://www.fxguide.com/featured/ben-snow-the-evolution-of-ilm-lighting-tools/

[8] Alan H. Baxr. "Ray Tracing Deformed Surfaces". ACM SIGGRAPH Computer Graphics, vol. 20, nro 4, pp. 287-296. 2011. 1986

[9] Donald, P. Greenberg, Michael, F. Cohen, y Kenneth, E. Torrance. "Radiosity: A method for computing global illumination". The Visual Computer, Vol. 2, nro 5, pp 291-297. 1986.

[10] James, T. Kajiya. "The rendering equation (1986)". SIGGRAPH '86 Proceedings of the 13th annual conference on Computer graphics and interactive techniques, pp 143-150. 1986.

[11] Jensen, H. "Global Illumination using Photon Maps". Proceedings of the eurographics workshop on Rendering techniques '96, pp 21-30. 1996.

[12] Cryer, J. "Resemble.js: Image analysis and comparison". [En línea]. http://huddle.github.io/Resemble.js/

[13] Decaudin, P. AntTweakBar. http://anttweakbar.sourceforge.net/doc/

[14] Rost, R. Licea-Kane, B. "OpenGL Shading Language". 3rd Edition. 2010.

[15] Snow, B. “Ben Snow: the evolution of ILM's lighting tools”. [En Linea] https://www.fxguide.com/featured/ben-snow-the-evolution-of-ilm-lighting-tools/