



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

**Trazado de Conos para el Cálculo de Iluminación Global empleando
Sombreado de Vóxeles**

Trabajo Especial de Grado presentado ante la Ilustre
Universidad Central de Venezuela
por el Br. José Gabriel Villegas Guedez
para optar al título de Licenciado en Computación.

Tutor:
Prof. Esmitt Ramírez

Caracas, abril del 2016

Resumen

La iluminación de escenas es fundamental para la generación imágenes de alta calidad, esta provee inmersión, sensación de profundidad y realismo. La producción de iluminación realista comprende, entre muchas otras cosas, la inclusión de iluminación indirecta para la composición de iluminación global.

Existen varios algoritmos para el cálculo de la iluminación global de forma analítica, sin embargo el costo computacional de estos es alto y dependiente de la complejidad de la escena. Esto hace estas soluciones poco flexibles o simplemente inadecuadas para aplicaciones en tiempo real, altamente interactivas o de considerable complejidad geométrica.

Con el incremento de la capacidad de cómputo en las unidades de procesamiento gráfico modernas también ha aumentado el interés por la inclusión de fenómenos de iluminación global en aplicaciones en tiempo real. De esto han surgido una variada cantidad de aproximaciones explotando características del hardware de procesamiento gráfico y los recursos disponibles en el pipeline de renderizado. Estos algoritmos mantienen cierto grado de coherencia o convergen con la solución analítica al problema de iluminación global.

Este trabajo tiene como enfoque principal el renderizado de iluminación global en tiempo real, por esto se realiza una investigación de estos algoritmos y se presenta el desarrollo de una aplicación con iluminación global inspirada principalmente en el trabajo de trazado de véxeles y conos introducido por Cyril Crassin y otros en 2011.

Palabras Clave: Iluminación global, véxeles, trazado de conos, iluminación indirecta, unidad de cómputo gráfico, renderizado en tiempo real.

Índice general

Introducción	v
1. Marco Teórico	1
1.1. Iluminación Global	1
1.2. Radiometría	2
1.2.1. Unidades en Radiometría	2
1.2.1.1. Flujo Radiante	2
1.2.1.2. Irradiancia	2
1.2.1.3. Emitancia Radiante o Radiosidad	2
1.2.1.4. Radiancia	3
1.3. Iluminación directa e indirecta	3
1.4. Representación de Superficies	4
1.4.1. Función de Distribución de Reflectancia Bidireccional	4
1.4.1.1. Propiedades de la Función BRDF	5
1.4.1.2. Ejemplos de BRDF	5
1.4.1.3. Modelos de sombreado.	6
1.4.2. Función de Distribución Normal	8
1.5. Ecuación de Renderizado	10
1.5.1. Formulación Hemisférica	10
1.5.2. Procedimientos	11
1.5.2.1. Radiosidad	12
1.5.2.2. Trazado de Rayos e Integración Monte Carlo	12
1.6. Técnicas Comunes de Renderizado	13
1.6.1. Mapeado de Sombras	13
1.6.2. Sombreado Diferido	14
1.7. Iluminación Global en Tiempo Real.	15
1.7.1. Luces Puntuales Virtuales	15

1.7.2. Mapas de Sombras Reflectivo	16
1.7.3. Volúmenes de Propagación de Luz en Cascada	17
1.7.4. Iluminación Indirecta con Trazado de Conos y Vóxeles	19
1.7.4.1. Construcción del Octree de Vóxeles	20
1.7.4.2. Contenido de un Vóxel	20
1.7.4.3. Filtrado Mip-mapping	21
1.7.4.4. Trazado de Conos y Vóxeles	21
1.7.4.5. Filtrado Anisotrópico de Vóxeles.	22
1.7.4.6. Captura de Iluminación Directa	23
2. Solución Propuesta	24
3. Implementación	25
4. Pruebas y Resultados	26
4.1. Entorno de Pruebas	26
4.1.1. Configuración de la Aplicación	26
4.2. Escenarios de Estudio	27
4.2.1. Escenas de Prueba y Objetos	27
4.2.1.1. Escenas Completas	27
4.2.1.2. Escenas Sandbox	30
4.2.1.3. Objetos Precargados	30
4.2.1.4. Criterios de Complejidad Geométrica e Iluminación	31
4.3. Estudio de Rendimiento	32
4.3.1. Prueba Base	32
4.3.1.1. Densidad Geometrica y Velocidad de Voxelizacion.	34
4.3.1.2. Espaciado Interior y Velocidad de Trazado.	34
4.3.2. Trazado de Sombras y Volumen de Visibilidad	34
4.3.3. Apertura del Cono Especular	34
4.3.4. Apertura del Cono de Sombreado	34
4.3.5. Estudio de Calidad de Imagen	34
4.3.6. Estudio de Memoria	34
5. Conclusión	35
Bibliografía	36

Introducción

Capítulo 1

Marco Teórico

1.1. Iluminación Global

Iluminación global se le llama al proceso de calcular la distribución de la energía de la luz sobre escenas 3-dimensionales desplegadas por computadora. Los efectos de la iluminación global incluyen suave sombreado debajo de objetos y cerca de esquinas, rebotes de luz, mezcla de colores, cáusticas, transluminiscencia, entre otros. Estos efectos son muy sutiles pero afectan el realismo de la imagen final de forma substancial [1].

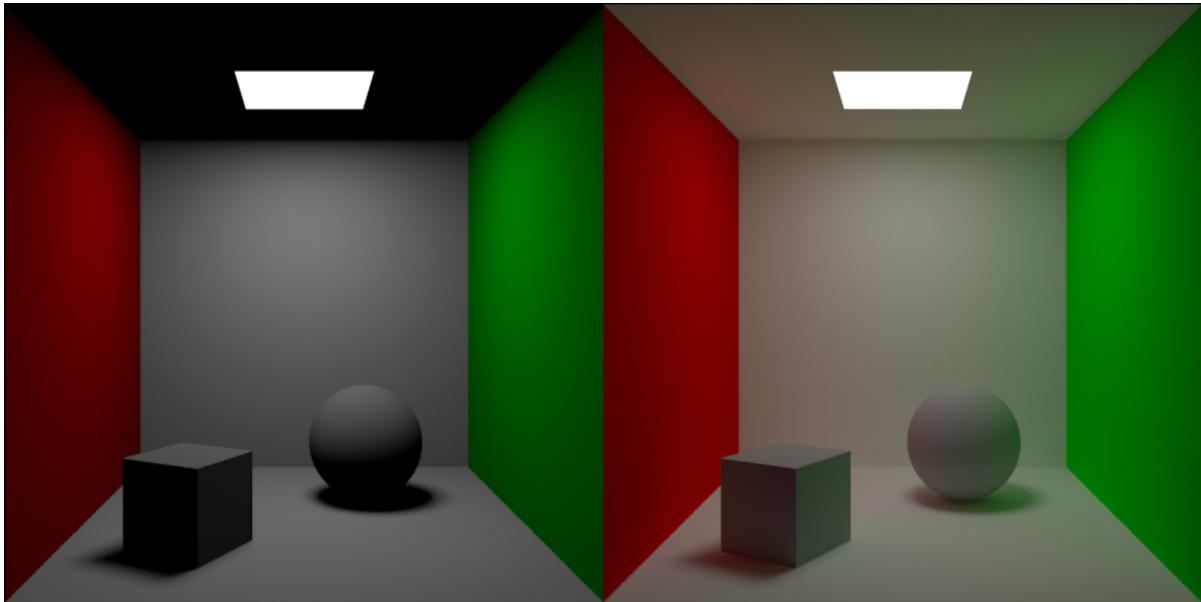


Figura 1.1: Solo iluminación directa (izquierda). Iluminación global (derecha)

El cómputo preciso y completo de iluminación global es considerado poco flexible, costoso y lento para ser utilizable en ciertos medios visuales que requieren producción de imágenes en tiempos interactivos o en escenas de gran complejidad, por ejemplo videojuegos o simulaciones. Es por esto que el desarrollo de aproximaciones y algoritmos para simular iluminación global con mejores factores de rendimiento y flexibilidad es un constante tema de investigación.

1.2. Radiometría

Radiometría es el campo dedicado al estudio y medición de la radiación electromagnética. Para el computo de la distribución de la luz es necesario entender algunas unidades importantes [2].

1.2.1. Unidades en Radiometría

1.2.1.1. Flujo Radiante

La unidad fundamental en radiometría, usualmente denotada como Φ es expresada en watts o vatios. Esta cantidad expresa cuanta energía total fluye desde, hasta y a través de una superficie por unidad de tiempo.

1.2.1.2. Irradiancia

La irradiancia E es el flujo radiante entrante o incidente por unidad sobre el área de una superficie. Esta es expresada en $watts/m^2$:

$$E = \frac{d\Phi}{dA} \quad (1.1)$$

1.2.1.3. Emitancia Radiante o Radiosidad

La emitancia radiante M es el flujo radiante saliente o emitido por unidad sobre el área de una superficie. También es expresada en $watts/m^2$:

$$M = \frac{d\Phi}{dA} \quad (1.2)$$

1.2.1.4. Radiancia

La radiancia L es el flujo radiante emitido por unidad de ángulo sólido y por unidad de área proyectada, expresada en *watts/estereorradián · m²*. De forma intuitiva la radiancia expresa cuanta potencia llega (o sale) de un punto x por unidad de ángulo sólido y por unidad de área proyectada. La radiancia varia con la posición x y el vector dirección Θ , es expresada como $L(x, \Theta)$.

$$L = \frac{d^2\Phi}{dwdA^\perp} = \frac{d^2\Phi}{dwA \cos \theta} \quad (1.3)$$

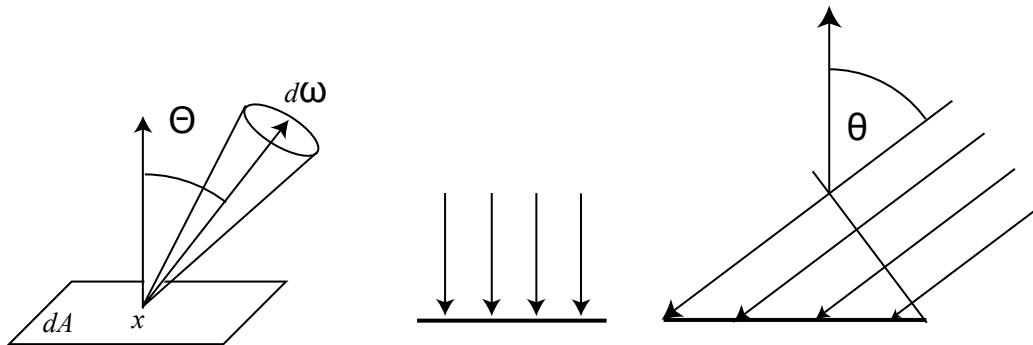


Figura 1.2: Definicion de radiancia $L(x, \Theta)$. Flujo radiante emitido por unidad de ángulo sólido $d\omega$ y por unidad de área proyectada A^\perp .

La radiancia es el término más importantes para los propósitos de este trabajo y probablemente lo es también para una cantidad importante de algoritmos y aproximaciones para el cálculo de iluminación global, es esta unidad la que captura la apariencia de los objetos en escena.

1.3. Iluminación directa e indirecta

La iluminación directa es aquella que se proyecta sobre la superficie de algún objeto directamente desde las fuentes de luz en la escena. Iluminación indirecta es la luz que proviene de los subsecuentes rebotes de luz originados desde las superficies iluminadas, ya sea esta superficie reflectiva o no [2] (Figura 1.1). La composición de ambas resulta en iluminación global.

1.4. Representación de Superficies

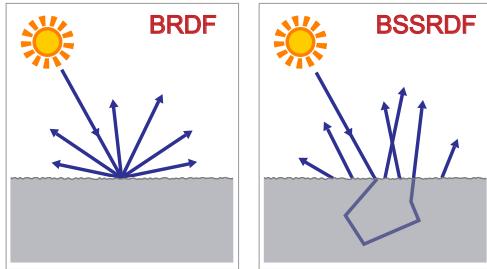


Figura 1.3: BRDF izquierda, punto de salida es igual al punto de entrada.
BSSRDF derecha, se observa como el punto de salida es distinto al de entrada.

Los materiales interactúan con la luz de distintas maneras. Esto hace que la apariencia de ciertos materiales difiera según las condiciones de la luz en una escena. Algunos materiales parecen espejos mientras que otros son totalmente difusos; son visualmente distinguibles materiales como vidrio, madera o metales. Las propiedades de reflectancia de una superficie afectan la apariencia del objeto [2], estos objetos se distinguen por la cantidad de luz reflectada en ciertas direcciones.

En el caso más general, un rayo de luz entra en algún punto p sobre una superficie en una escena, este rayo de luz tiene una dirección incidente Ψ y puede salir de esta superficie sobre otro punto q con dirección saliente Θ . La función que define esta relación entre la radiancia incidente y la radiancia reflectada se llama BSSRDF.

1.4.1. Función de Distribución de Reflectancia Bidireccional

Al asumir que la luz incidente en algún punto x sale del mismo punto x (ignorando la transluminiscencia) las propiedades de reflectancia de una superficie son entonces descritas por una BRDF.

La BRDF en un punto x se define entonces como la distribución de la radiancia reflectada diferencial en una dirección saliente Θ y la irradiancia incidente diferencial a través de un ángulo sólido $d\omega_\Psi$. La función BRDF puede ser escrita de la siguiente forma [2]:

$$f_r(x, \Psi \rightarrow \Theta) = \frac{dL(x \rightarrow \Theta)}{dE(x \leftarrow \Psi)} = \frac{dL(x \rightarrow \Theta)}{L(x \leftarrow \Psi) \cos(N_x, \Psi) d\omega_\Psi} \quad (1.4)$$

Donde $\cos(N_x, \Psi)$ es el coseno del ángulo formado entre la normal en el punto x y el vector dirección Ψ

1.4.1.1. Propiedades de la Función BRDF

La función BRDF tiene una variada cantidad de importantes propiedades:

1. Dimensión: La función BRDF es una función 4-dimensional definida sobre cada punto de una superficie, dos dimensiones corresponden a la dirección entrante y dos a la dirección saliente.
2. Reciprocidad: El resultado de la función BRDF es el mismo si se intercambian la dirección entrante y la dirección saliente:

$$f_r(x, \Psi \rightarrow \Theta) = f_r(x, \Theta \rightarrow \Psi) \quad (1.5)$$

3. Conservación de la energía: La ley de conservación de la energía dicta que la cantidad total de energía reflectada en todas las direcciones debe ser menor o igual a la cantidad de total de energía incidente sobre las superficies:

$$\int_{\Omega^+} f_r(x, \Psi \rightarrow \Theta) \cos(N_x, \Theta) dw_\Theta \leq 1 \quad (1.6)$$

1.4.1.2. Ejemplos de BRDF

Dependiendo del comportamiento de la BRDF, el material se verá como una superficie difusa, como un espejo o como una mezcla de ambos (lustroso o *glossy*). Los tipos de BRDF relevantes para este trabajo serán listados aquí.

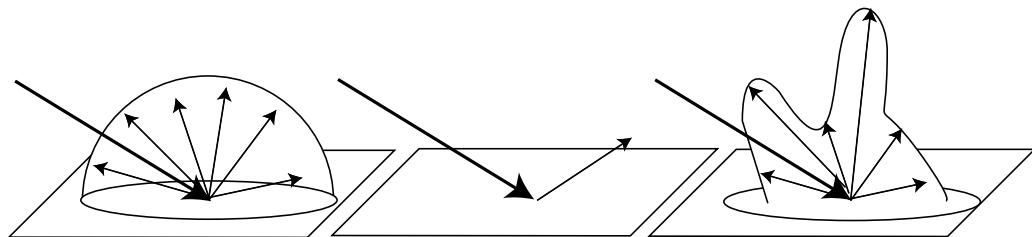


Figura 1.4: Ejemplos de superficies: totalmente difusa izquierda, totalmente especular centro, lustrosa (*glossy*) derecha. Fuente: Phillip Dutre y otros, *Advanced Global Illumination* [2]

Superficies Difusas: Algunos materiales reflectan la luz de forma uniforme sobre la totalidad de la semiesfera de reflectancia. Esto quiere decir que según la distribución de la irradiancia, la radiancia reflectada es independiente de la dirección de salida. Estos materiales son llamados

reflectores difusos y el valor de su BRDF es constante para todos los valores Θ y Ψ . Para un observador en un punto sobre una superficie difusa se ve igual desde todas las direcciones [2]. Para una superficie difusa ideal, la reflexión difusa puede ser representada como:

$$f_r(x, \Psi \leftrightarrow \Theta) = \frac{\rho_d}{\pi} \quad (1.7)$$

La reflectancia ρ_d representa la fracción de la energía incidente que es reflectada en la superficie. Para materiales físicamente correctos, ρ_d varía entre 0 y 1.

Superficies Especulares: Superficies especulares perfectas solo reflectan o refractan luz en una dirección específica.

Reflexión Especular: La dirección de reflexión puede ser obtenida utilizando la ley de reflexión, esta indica que la dirección de la luz incidente y saliente tienen un ángulo equivalente con la normal de la superficie. Dado que la luz es incidente con respecto a la superficie con vector dirección Ψ , y la normal de la superficie es N , la luz incidente es reflectada en la dirección R :

$$R = 2(N \cdot \Psi)N - \Psi \quad (1.8)$$

Un reflector espectral perfecto tiene solo una dirección de salida donde la BRDF es diferente de 0, esto implica que el valor de la BRDF en esa dirección es infinito.

Superficies Lustrosas: La mayoría de las superficies no son ni idealmente difusas ni idealmente especulares, sino que demuestran una combinación de ambas características de reflectancia. Estas superficies son llamadas superficies lustrosas o superficies *glossy*. La BRDF que describe esta clase de materiales es usualmente difícil de modelar de forma analítica [2].

1.4.1.3. Modelos de sombreado.

Materiales reales pueden tener BRDFs particularmente complejas. En computación gráfica existen varios modelos que intentan capturar la complejidad de las BRDFs. En la siguiente sección se expande sobre ciertos modelos relevantes a este trabajo. Nótese que Ψ es la direc-

ción de la luz (dirección incidente), Θ es la dirección del observador (dirección saliente) y N la normal de la superficie.

Modelo de Lambert: Uno de los modelos más simples, este modelo es ideal para superficies difusas, en este modelo la BRDF es una constante como ya fue descrito anteriormente en la ecuación 1.7.

$$f_r(x, \Psi \leftrightarrow \Theta) = k_d = \frac{\rho_d}{\pi} \quad (1.9)$$

Donde ρ_d es la reflexión difusa.

Modelo de Phong: La BRDF del modelo de Phong es:

$$f_r(x, \Psi \leftrightarrow \Theta) = k_s \frac{(R \cdot \Theta)^n}{N \cdot \Psi} + k_d \quad (1.10)$$

Donde el vector reflectado R es calculado con la ecuación 1.8.

Modelo de Blinn-Phong: El modelo Blinn-Phong utiliza el vector medio H entre Ψ y Θ de la siguiente manera:

$$f_r(x, \Psi \leftrightarrow \Theta) = k_s \frac{(N \cdot H)^n}{N \cdot \Psi} + k_d \quad (1.11)$$

El valor n varía según las propiedades del material. Un mayor valor provee un lóbulo especular más pequeño, simulando superficies lisas y pulidas.

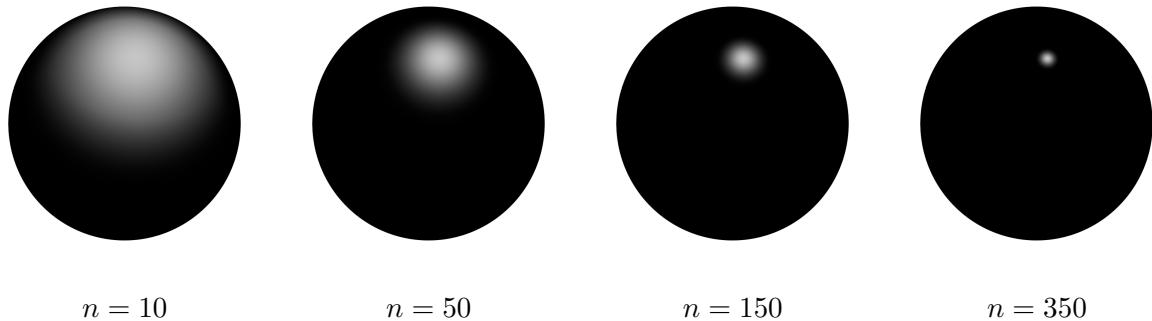


Figura 1.5: Distribución especular para distintos valores de n en el modelo Blinn-Phong.

Modelo de Blinn-Phong Modificado: El modelo de Phong a pesar de ser simple este tiene ciertas limitaciones, no es ni conservador de energía ni reciproco y tiene problemas para simular materiales reales. El modelo modificado toma en cuenta algunos de estos problemas:

$$f_r(x, \Psi \leftrightarrow \Theta) = k_s(N \cdot H)^n + k_d \quad (1.12)$$



Figura 1.6: Ejemplo de BRDFs. Se observa como estas afectan la apariencia final de la superficie.

1.4.2. Función de Distribución Normal

La función de distribución normal (NDF) introducida por Alain Fournier [3] describe la densidad de las normales como una función de dirección. Funciones gaussianas como la siguiente es una de las posibles formas de una NDF.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (1.13)$$

En la función gaussiana el término σ^2 es llamado variancia y el término μ es llamado media. Tanto el producto como la convolución de dos distribuciones gaussianas es también una distribución gaussiana [4].

La función gaussiana puede ser utilizada como una representación direccional. En este caso la media es un vector promedio D del lóbulo gaussiano. Como es descrito en el trabajo de Toksvig para el mipmapping de mapas de normales [5], el valor de σ puede ser calculado por la

longitud del vector promedio D utilizando la siguiente ecuación.

$$\sigma^2 = \frac{1 - |D|}{|D|} \quad (1.14)$$

Esto permite obtener lóbulos gaussianos a partir de dos o más vectores para representar su dirección en común.

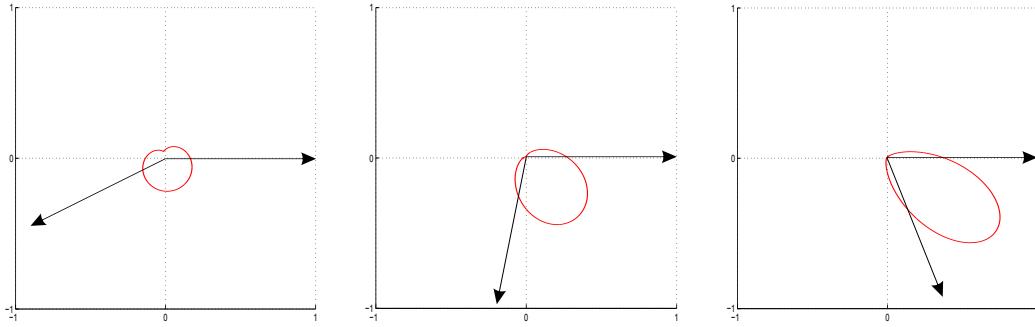


Figura 1.7: Ejemplo de lóbulos gaussianos según dos vectores, se puede observar como la forma del lóbulo cambia según la dirección promedio.

Algunas BRDF también pueden describirse como distribuciones gaussianas, las propiedades de producto y convolución se mantienen.

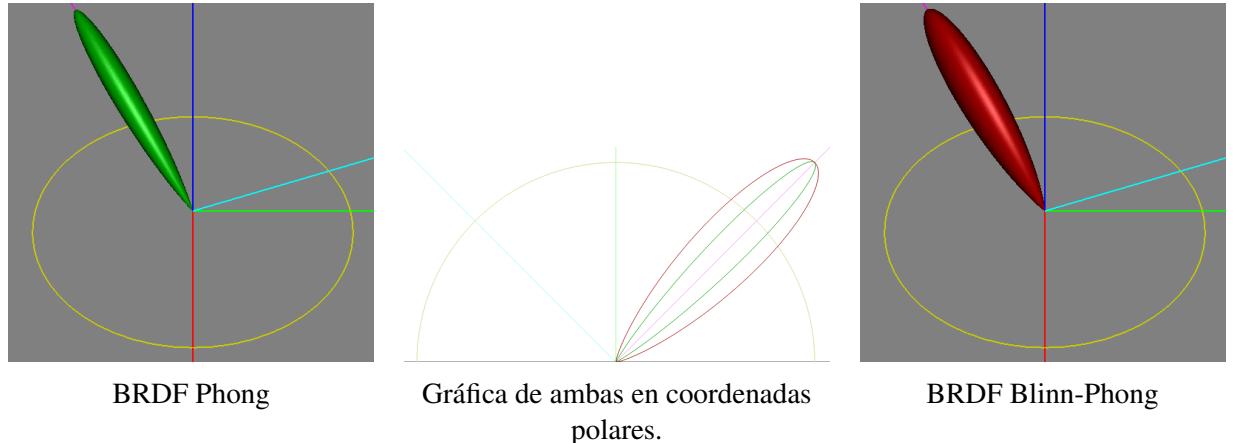


Figura 1.8: Lóbulos especulares para BRDF Phong y Blinn-Phong para un Θ y Ψ de $\angle 45$, se puede observar cómo estas describen la distribución de la dirección de reflectancia. Como se explica en 1.4.1.3 el valor de n afecta la forma del lóbulo mientras mayor es este número más fino y largo es el lóbulo especular. Imágenes renderizadas en Disney's BRDF Explorer [6].

1.5. Ecuación de Renderizado

La ecuación de renderizado recursiva, introducida por Kajiya [7]. La ecuación de renderizado describe en cada punto x de una superficie y en cada dirección Θ , la radiancia saliente $L(x \rightarrow \Theta)$ en ese punto y esa dirección.

El objetivo de un algoritmo para el cálculo de iluminación global es aproximar el resultado de esta ecuación. En esta ecuación asumimos que no existen medios participantes como objetos translúcidos como ya fue explicado en la sección 1.4. También asumimos que la luz se propaga de forma inmediata por tanto la distribución de la luz, ya en un estado estacionario, se obtiene inmediatamente.

1.5.1. Formulación Hemisférica

La formulación hemisférica de la ecuación de renderizado es una de las más utilizadas [2]. Esta formulación se obtiene utilizando la propiedad de conservación de energía en el punto x . Asumiendo que $L_e(x \rightarrow \Theta)$ representa la radiancia emitida por la superficie en el punto x con dirección saliente Θ y $L_r(x \rightarrow \Theta)$ representa la radiancia reflectada por la superficie en el punto x con dirección Θ .

Por conservación de energía, el total de la radiancia saliente en un punto y dirección particular es la suma de la radiancia emitida y la radiancia reflectada en este punto de la superficie y dirección. La radiancia saliente $L(x \rightarrow \Theta)$ es expresada en términos de $L_e(x \rightarrow \Theta)$ y $L_r(x \rightarrow \Theta)$ de la siguiente forma:

$$L(x \rightarrow \Theta) = L_e(x \rightarrow \Theta) + L_r(x \rightarrow \Theta) \quad (1.15)$$

Por la definición de BRDF en la ecuación 1.4 tenemos que:

$$\begin{aligned} f_r(x, \Psi \rightarrow \Theta) &= \frac{dL(x \rightarrow \Theta)}{dE(x \leftarrow \Psi)} \\ L_r(x \rightarrow \Theta) &= \int_{\Omega_x} f_r(x, \Psi \rightarrow \Theta) L(x \leftarrow \Psi) \cos(N_x, \Psi) dw_\Psi \end{aligned} \quad (1.16)$$

Colocando estas ecuaciones juntas obtenemos la ecuación de renderizado:

$$L(x \rightarrow \Theta) = L_e(x \rightarrow \Theta) + \int_{\Omega_x} f_r(x, \Psi \rightarrow \Theta) L(x \leftarrow \Psi) \cos(N_x, \Psi) dw_\Psi \quad (1.17)$$

1.5.2. Procedimientos

En esta sección se explica dos populares procedimientos clásicos para obtener una aproximación a la ecuación de renderizado, esto es una aproximación de la propagación de la luz en una escena. Estas soluciones no están pensadas para tiempos interactivos y su enfoque principal es precisión.

Métodos como elementos finitos y Monte Carlo son los grupos de algoritmos más utilizados para aproximar la ecuación de renderizado. El método de elementos finitos utiliza alguna forma de discretización para reducir la ecuación de renderizado a una ecuación de matrices. Los métodos Monte Carlo muestran los caminos que siguen los rayos de luz en una escena, generando un estimado estadístico de la apariencia real de la escena. *Radiosity* es una popular aproximación que utiliza el método de elementos finitos. Trazado de rayos y caminos (*ray tracing* y *path tracing*) son aproximaciones comunes que utilizan el método Monte Carlo [8].

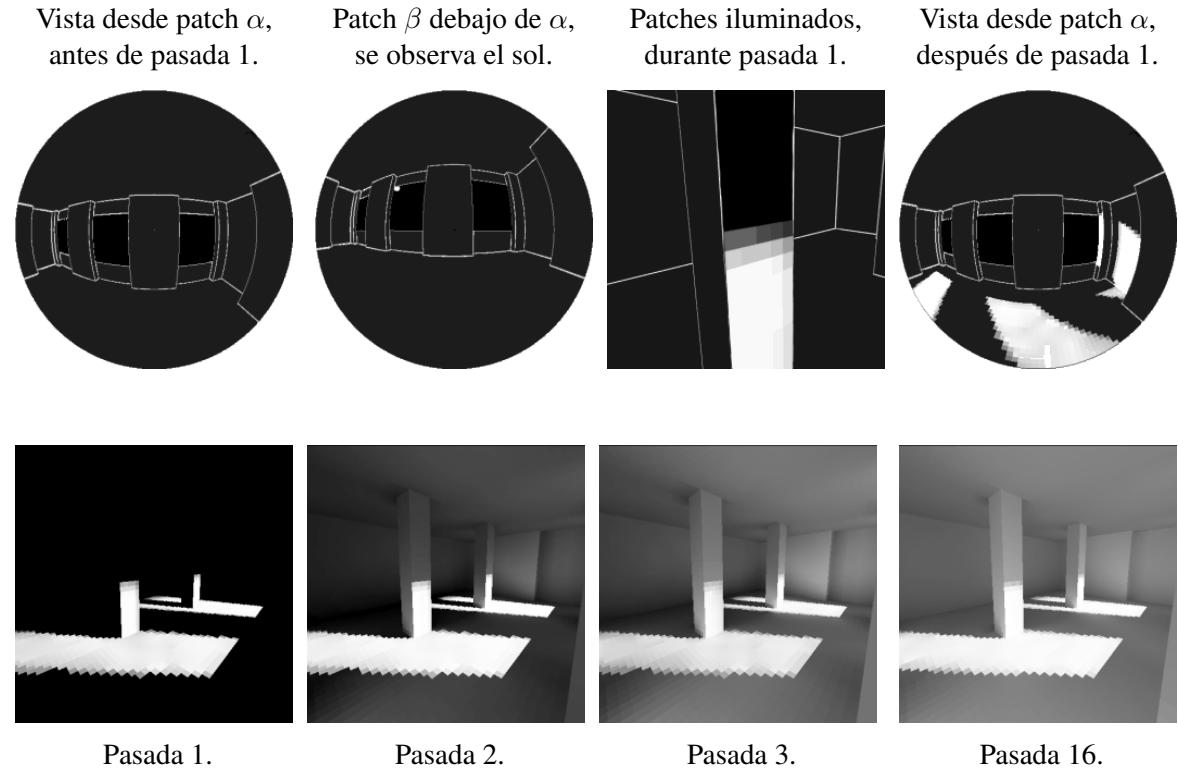


Figura 1.10: Ejemplo de varias pasadas de radiosidad sobre una escena. Fuente: Hugo Elias, *The Workings of a Radiosity Renderer* [9].

1.5.2.1. Radiosidad

Radiosidad es una aproximación con elementos finitos común para el cómputo del transporte de luz global. Esta técnica fue introducida por Goral y otros en 1984 [10]. La idea general es discretizar las superficies de la escena en elementos finitos de éstas, estos elementos son usualmente llamados *patches* (parches o trozos) los cuales son utilizados para calcular el transporte de luz entre ellos como se observa en la figura 1.10. Esto conlleva a ciertas implicaciones; de cada *patch* se necesita guardar el valor de radiosidad para las superficies difusas, o la distribución direccional de la luz saliente y entrante para superficies no difusas.

1.5.2.2. Trazado de Rayos e Integración Monte Carlo

La ecuación de renderizado puede ser aproximada utilizando el algoritmo de trazado de rayos o *ray tracing*, esta es una técnica basada en integración Monte Carlo. Para aproximar la propagación de la luz se crea un número considerable de muestras en variadas direcciones, luego por cada muestra se evalúa la ecuación de renderizado y el promedio de todos los resultados converge hacia la solución analítica de la ecuación de renderizado. Para evaluar una muestra la luz incidente desde una dirección tiene que ser calculada, para esto un rayo de luz es enviado en una dirección y la luz emitida desde el primer punto de colisión es calculada evaluando la ecuación de renderizado en este punto.

Ray tracing está dividido en dos categorías: forward y backward. Forward ray tracing consiste en lanzar las trazas/rayos de luz desde las fuentes de luz y usar aquellos que llegan a la cámara. Backward ray tracing por el contrario lanza las trazas/rayos de luz desde la cámara y traza el camino de estos a través de la escena [11].



Figura 1.11: Ray tracing sobre una escena, se puede observar mayor calidad y reducción de ruido al aumentar la cantidad de muestras. Fuente: Loc Do, HW6: *Ray Tracing Extension* [12].

1.6. Técnicas Comunes de Renderizado

En esta sección se explican técnicas comunes utilizadas en síntesis o renderizado de imágenes que son relevantes para este trabajo.

1.6.1. Mapeado de Sombras

Con la luz es representada en forma de rayo las superficies sombreadas reciben menos rayos de luz ya que estas están ocluidas por otras superficies que se encuentran entre ellas y los emisores de luz. En un pipeline de renderizado general, donde las superficies en escenas son representadas en geometría poligonal, trazar rayos por cada fragmento para comprobar la visibilidad del mismo no es una operación trivial.

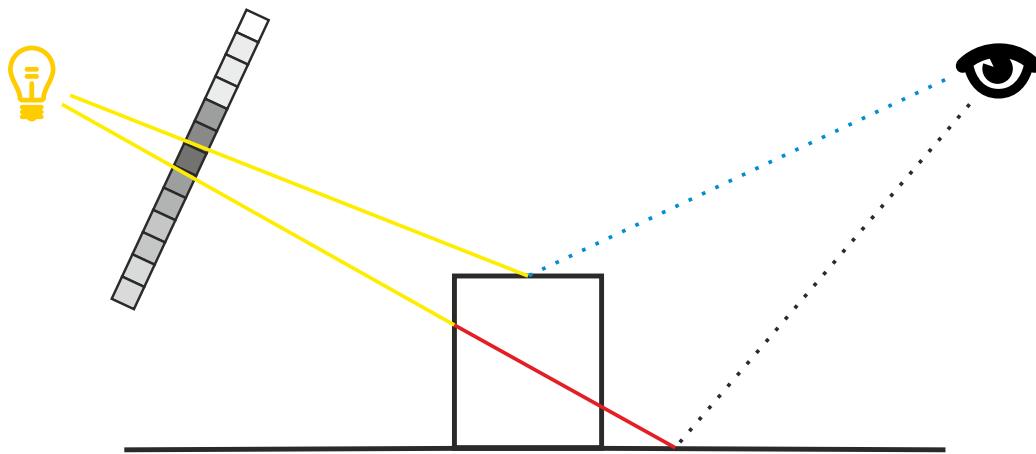


Figura 1.12: La profundidad almacenada en el mapa de sombra (amarillo) es comparada con la profundidad del punto en la superficie desde la luz (rojo).

Los mapas de sombras, presentados inicialmente por Lance Williams en 1978 [13] presentan una solución simple para el cálculo de visibilidad de un fragmento.

La técnica consiste en proyectar la escena en una textura bidimensional desde la posición y con la dirección de una fuente de luz. La proyección es calculada utilizando una matriz de proyección P_l . Por cada píxel de esta textura la profundidad de cada fragmento sobre una superficie es almacenada. Esta textura es llamada mapa de sombra. Una que se pasa a renderizar la escena desde el punto de vista del observador, por cada fragmento con posición p_{ws} en espacio de mundo, la posición en el mapa de sombra p_{sh} es calculada utilizando la siguiente ecuación:

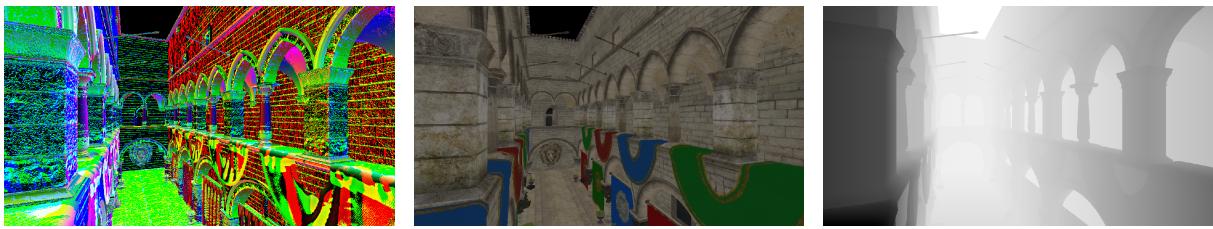
$$p_{sh} = P_l * p_{ws} \quad (1.18)$$

Como se muestra en la figura 1.12 si la profundidad del fragmento desde la fuente de luz es mayor que el valor almacenado en el mapa de sombra entonces este punto esta sombreado.

El mapeado de sombras es una solución sencilla y efectiva al problema de pruebas de visibilidad pero este tiene dos mayores problemas. El mapa de sombras está limitado a la resolución de la textura y además este representa una discretización de la profundidad de la escena desde la fuente de luz y esto introduce una variedad de anomalías visuales. A partir de este concepto existe una variedad de algoritmos para el cálculo de sombras que intentan solventar estos problemas.

1.6.2. Sombreado Diferido

En sombreado directo la representación poligonal de la escena es rasterizada y operaciones por píxel como iluminación y sombreado son realizadas por cada fragmento generado por el proceso de rasterización. Esto es poco efectivo cuando consideramos que gran parte de los fragmentos no forman parte de la imagen final. Con sombreado diferido se pueden realizar estas operaciones por pixel solo sobre los fragmentos visibles. Este concepto está basado en el trabajo de Deering en y otros en 1998 [14]. La escena es renderizada solo una vez y varios atributos de la escena son almacenados en buffers. Este buffer es llamado buffer de geometría (G-Buffer) y fue introducido por Saito y otros en 1990 [15]. El contenido general de un G-Buffer es profundidad, albedo y normal, esto puede cambiar según las necesidades de la aplicación. El propósito de almacenar esta información es separar las operaciones que solo son necesarias sobre los fragmentos visibles de la rasterización de toda la escena, de manera que cálculos como iluminación ahora son realizados en otro paso solo sobre cada píxel almacenado en el G-Buffer.



Normales.

Albedo.

Profundidad.

Figura 1.13: El contenido de un buffer de geometría.

1.7. Iluminación Global en Tiempo Real.

En esta sección se examinan algunos algoritmos para el cálculo de iluminación global en tiempos interactivos o *real-time*. Iluminación indirecta con trazado de conos y vóxeles es revisada con detalle ya que esta técnica es de particular interés para este trabajo.

1.7.1. Luces Puntuales Virtuales

Una variedad de algoritmos para el cálculo de iluminación global se inspiran o hacen uso del concepto de luz puntual virtual (VPL). Este trabajo fue presentado por Keller en 1997 [16].

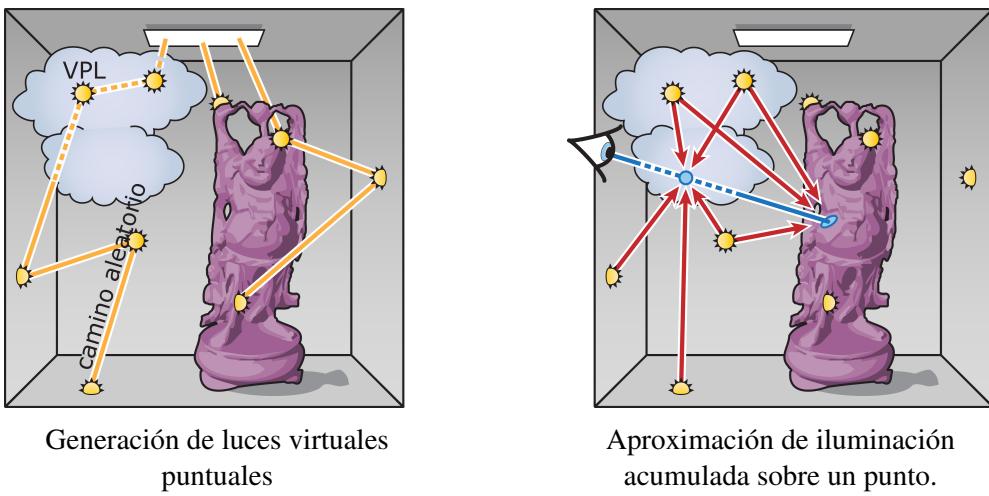


Figura 1.14: Pasos del algoritmo VPL para la aproximación de iluminación indirecta. Fuente: Dachsbacher y otros, *Scalable Realistic Rendering with Many-Light Methods* [17].

En este algoritmo se aproxima la radiancia reflectada en escena utilizando un conjunto de luces virtuales. La radiancia que llega a un punto x es aproximada por la radiancia que proviene de las luces virtuales. Pruebas de visibilidad para cada una de estas luces son realizadas utilizando técnicas de sombreado estándar y la radiancia proveniente de cada una de estas es almacenada en un buffer de acumulación. Las luces virtuales son generadas a partir de partículas lanzadas por las fuentes de luz principales utilizando la secuencia de Halton para el muestreo. En un principio, un numero n de partículas son generadas, como no toda la radiancia es absorbida algunas de estas partículas son reflejadas. Luego del primer rebote, uno numero $p'n$ de partículas son reflejadas. Luego de $j - 1$ reflexiones p'^j son reflejadas. El numero p' es descrito por la

siguiente ecuación.

$$p' = \frac{\sum_{k=1}^K p_{d,k} |A_k|}{\sum_{k=1}^K |A_k|} \quad (1.19)$$

Donde la escena es compuesta por K elementos de superficie A_k con una reflectividad promedio de $p_{d,k}$.

1.7.2. Mapas de Sombras Reflectivo

Otra técnica utilizada en varios algoritmos de iluminación global es mapas de sombras reflectivo (RSM) o *reflective shadow maps*. La técnica es RSM presentada por Dachsbacher y otros en 2005 [18]. Esta tecica está inspirada en mapeado de sombras como ya fue explicado anteriormente en 1.6.1, se utiliza proyección desde la fuente de luz para determinar el primer rebote de luz. Al renderizar la escena desde el punto de vista de la fuente de luz, se entiende que todos los fragmentos en el mapa de sombras son los únicos fragmentos involucrados en el primer rebote de luz. En RSM cada uno de los pixeles en el mapa de sombras es considerado una fuente de luz. Por cada pixel p además de la profundidad d_p , se necesita almacenar posición x_p , normal n_p , y el flujo de radiancia reflectada Θ_p .

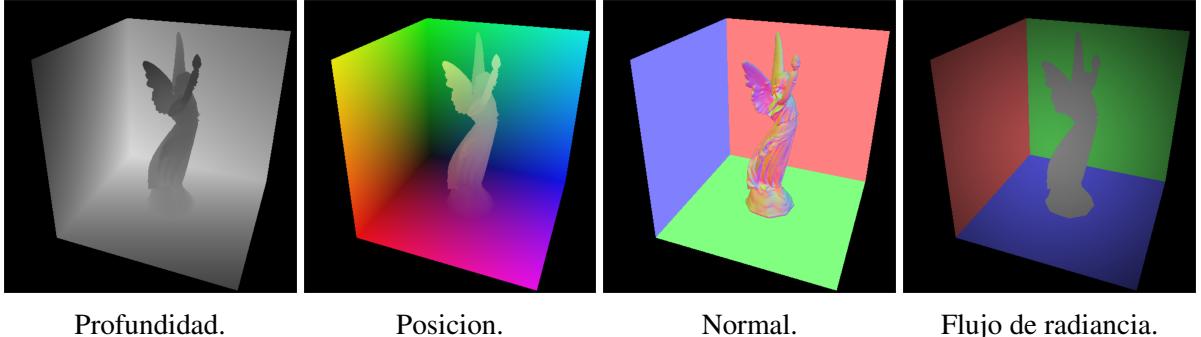


Figura 1.15: Mapas utilizados por RSM. Fuente: Dachsbacher y otros, *Reflective Shadow Maps* [18].

Si asumimos que todas las superficies son reflectores difusos, la intensidad de la radiancia emitida en una dirección ω desde un pixel del RSM es descrita por la siguiente ecuación.

$$I_p(\omega) = \Theta_p \max(0, n_p \cdot w) \quad (1.20)$$

La iluminación indirecta de un punto se calcula sumando todas las intensidades de todos los pixeles (considerados ahora como luces) en el RSM visibles. Calcular esto es costoso, por tanto en vez sumar todos los pixeles visibles al punto se toma cierta cantidad de muestras

del RSM. La posición del punto iluminado x_p es proyectado sobre el RSM y las muestras son seleccionadas alrededor de esta posición proyectada. La densidad de las muestras decrece con la distancia cuadrada de la posición proyectada al punto iluminado. Esto asume que dos superficies cercanas proyectadas al RSM también son cercanas en el RSM. También se asume que la muestra es directamente visible desde la superficie iluminada.

1.7.3. Volúmenes de Propagación de Luz en Cascada

Volúmenes de propagación de luz en cascada (CLPV) o *Cascaded Light Propagation Volumes* presentando por Kaplanyan y otros en 2010 [19] es un algoritmo para el cálculo de iluminación indirecta en tiempo real.



Figura 1.16: Iluminación global para la escena *Sponza* utilizando volúmenes de propagación de luz. Composición final arriba y solo iluminación indirecta abajo. Fuente: Kaplanyan y otros, *Cascaded Light Propagation Volumes for Real-time Indirect Illumination* [19]

Este método simula el transporte de luz utilizando técnicas similares en algoritmos para simulación de fluidos basados en cuadriculas [20] tridimensionales. La intensidad de luz es almacenada en una cuadricula y de forma iterativa cada celda transfiere la intensidad de la luz a sus vecinos. Esta cuadricula es llamada volumen de propagación de luz (LPV). Esta luz puede ser bloqueada por la geometría de la escena, la cual es obtenida de otra cuadricula llamada volumen de geometría (GV). Para mejorar el rendimiento del algoritmo y reducir el consumo de memoria se utiliza un conjunto de cuadriculas anidadas. Para los objetos cercanos al observador la iluminación indirecta es calculada utilizando una cuadricula mucho más fina.

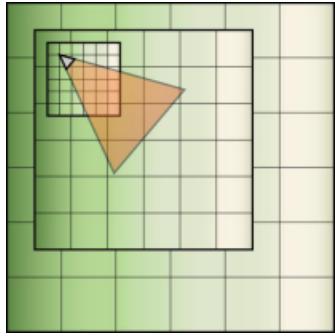


Figura 1.17: Cuadriculas de propagación anidadas, las cuadriculas se solapan.

Primero por cada fuente de luz, es necesario renderizar un RSM. Cada texel del RSM es considerado una VPL. La intensidad del VPL es acumulada y almacenada como un harmónico esférico dentro de las celdas de la cuadricula.

Para una correcta propagación de la luz el algoritmo necesita conocer la geometría de la escena. Por esto de la misma manera en la que se almacena la intensidad de la luz también se almacena una representación de la escena en una cuadricula. Esta representación es guardada sobre el GV. Esta cuadricula es trasladada por mitad de tamaño de celda unidades con respecto al LPV, esto asegura que el centro de todas las celdas del GV queden en las esquinas del LPV. Para conocer segmentos de geometría se utiliza un G-Buffer y las fuentes de luz de los RSMs.

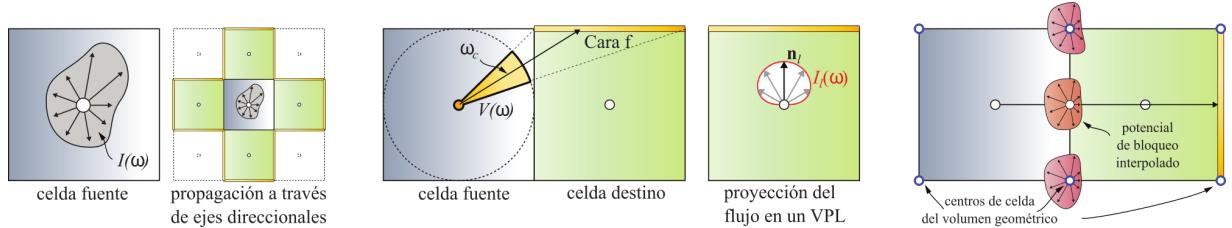


Figura 1.18: Izquierda: Cada celda del LPV almacena la intensidad de luz que es propagada desde la celda fuente. Centro: El flujo es calculado sobre cada cara de la celda destino para preservar información direccional. Derecha: Oclusión borrosa o *fuzzy* almacenando una representación volumétrica de la escena [19].

La propagación de luz se realiza de forma iterativa. La intensidad de luz en cada iteración es propagada a 6 vecinos por celda según eje direccional principal. Primero por cada celda adyacente el flujo de radiancia incidente por cada una de las celdas es calculado. Luego el flujo incidente de cada celda es transformado en emitancia radiante. Esto se logra creando VPLs, cada una de estas luces virtuales colocadas sobre una de las caras de la celda y emitiendo un flujo radiancia similar el flujo de radiancia de la celda. Estas VPL son acumuladas dentro del LPV de nuevo y almacenadas como esféricos harmónicos utilizando el mismo proceso de inyección del primer paso.

1.7.4. Iluminación Indirecta con Trazado de Conos y Vóxeles

Este algoritmo es presentado por Crassin y otros en 2011 [21] para el cálculo de iluminación indirecta utilizando trazado de conos contra vóxeles o *Indirect Illumination Using Voxel Cone tracing*. En este técnica se utiliza una estructura de árbol disperso o *sparse* para almacenar ya filtrados los valores necesarios para el cálculo de iluminación indirecta en vóxeles. Este árbol es una representación tridimensional de la escena por tanto cada nodo tiene ocho hijos que representan las ocho particiones de un cubo en partes más pequeñas de forma uniforme. Esta clase de estructuras son llamadas *octrees*. La estructura dispersa requiere menor consumo de memoria ya que solo los vóxeles necesarios son almacenados.

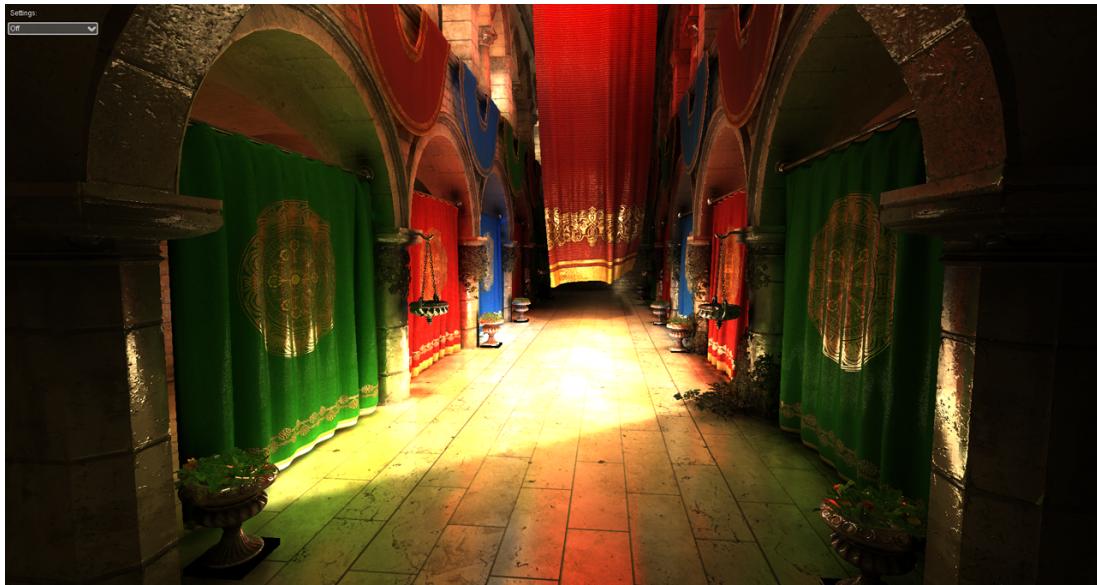


Figura 1.19: Escena *Sponza* renderizada utilizando con trazado vóxeles con conos para la iluminación indirecta. Fuente: Cyril Crassin y otros, *Interactive Indirect Illumination Using Voxel Cone Tracing* [21].

El algoritmo comprende varios pasos. Primero la información de la luz y escena son almacenados en las hojas de la estructura de árbol, este es el nivel más fino de la jerarquía. Luego estos valores son filtrados dentro del árbol disperso hacia todos los niveles de la jerarquía hasta llegar a la raíz. En un último paso para el cálculo de iluminación indirecta por cada fragmento, los valores dentro de esta jerarquía son recolectados sobre una semiesfera. En algoritmos como ray tracing esta recolección de valores es lenta y es realizada por muchos rayos. Es de notar que todos estos rayos trazados sobre la semiesfera son direccional y espacialmente coherentes. VCT hace uso de este concepto para discretizar muchos rayos en simples conos.

1.7.4.1. Construcción del Octree de Vóxeles

El algoritmo está pensando para funcionar con escenas dinámicas. Sin embargo las escenas son divididas entre partes dinámicas y estáticas para acelerar el proceso de voxelización con objetos dinámicos.

Primero la escena es renderizada utilizando proyección ortogonal. Cada triángulo es proyectado sobre uno de los ejes principales, este eje es seleccionado según la normal del triángulo, esto se hace para maximizar el área visible del triángulo según con respecto al eje. Cada fragmento producto de esta proyección es almacenado en una lista de vóxeles-fragmentos junto a parámetros como posición en escena, normal y color. Para saber la cantidad de elementos que esta lista debe almacenar se debe realizar una pasada antes contando el número de fragmentos con un contador atómico.

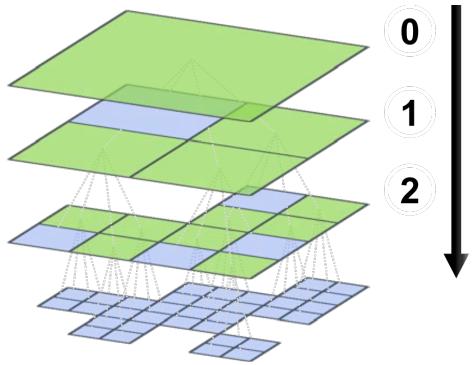


Figura 1.20: Descripción gráfica del proceso de subdivisión del octree.

Una vez llena la lista de vóxeles-fragmentos.

Se genera un hilo de procesamiento por cada fragmento en la lista. Los fragmentos son introducidos en el árbol disperso que inicialmente solo tiene un nodo raíz. Cada vez que un nodo de este octree necesita ser dividido un nuevo nodo es creado y se almacena sobre memoria ya reservada en GPU. La posición del nodo en memoria es determinada por un contador atómico, el cual se incrementa con cada nuevo nodo. Al inicio del proceso de subdivisión de los nodos es de esperar una gran cantidad de colisiones entre hilos, por esto cada nodo tiene asociado un símbolo mutex.

1.7.4.2. Contenido de un Vóxel

El algoritmo está diseñado para hacer uso de filtrado tri-linear por hardware. Sin embargo dos vóxeles vecinos no necesariamente están posicionados de forma subsecuente en memoria. Por esto cada nodo contiene un bloque o *brick*. Este bloque representa el entramado 3^3 de la celda, donde estas celdas se encuentran en las esquinas de los hijos del nodo.

Cada vóxel representa varios parámetros. Entre ellos color, opacidad, normal, intensidad, etc.

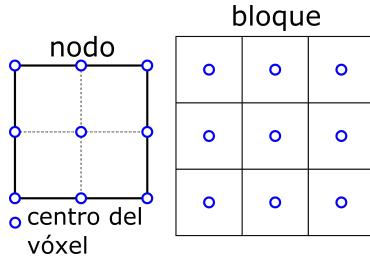


Figura 1.21: Bloques y nodos, los valores de los bloques se encuentran en las esquinas para rápido acceso y filtrado [21].

1.7.4.3. Filtrado Mip-mapping

Inicialmente cada uno de estos parámetros es almacenado dentro de las hojas del árbol disperso. Luego de forma iterativa estos valores son filtrados desde los niveles más bajos a los niveles más altos de la jerarquía, este proceso es llamado *mip-mapping*. Cada nodo de un bloque es filtrado a partir de las 3^3 celdas del nivel anterior en jerarquía. Para calcular el valor filtrado sobre el actual nodo el algoritmo promedia los valores de los nodos en el nivel anterior. Al calcular el valor filtrado cada voxel debe ser pesado con la inversa de su multiplicidad, resultando en un kernel gaussiano de 3^3 .

1.7.4.4. Trazado de Conos y Vóxeles

Una vez que el árbol disperso esta filtrado y completo este es utilizado para el cálculo de iluminación indirecta. Por cada fragmento un conjunto de conos es generado. La dirección y apertura de cada cono es determina por la BRDF del material en ese fragmento. Por ejemplo la BRDF Blinn-Phong vista en 1.4.1.3 puede ser descompuesta como un lóbulo ancho para la parte difusa y un lóbulo especular. Para el lóbulo difuso varios conos son generados orientados por la semiesfera con apertura y dirección maximizada a tal manera que los conos cubran parte de la misma. Para el lóbulo especular se genera un solo cono con una apertura que varía según el termino n de la BRDF Blinn-Phong. El cono especular tiene como dirección la dirección de la luz incidente reflectada R visto en 1.4.1.2.

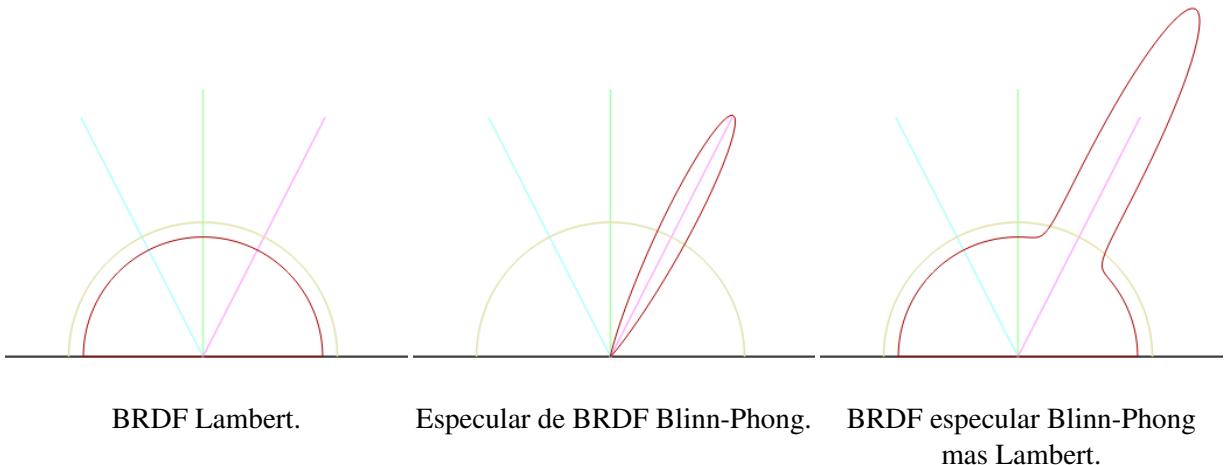


Figura 1.22: Graficas en coordenadas polares de la BRDF Lambert, specular Blinn-Phong y composición de ambas.

El algoritmo utiliza trazado de conos aproximados para recolectar la intensidad de la luz incidente sobre el fragmento. Los valores son recolectados de varias muestras a través del recorrido del cono. Por cada muestra se examina el árbol disperso de vóxeles. El nivel del árbol a examinar es determinado por el diámetro del cono en esa posición.

1.7.4.5. Filtrado Anisotrópico de Vóxeles.

A pesar de que el filtrado gaussiano es suficiente para proveer resultados visuales coherentes, algunos problemas de calidad visual pueden ocurrir bajo ciertas condiciones. El primer problema se conoce como el problema de la pared rojo-verde. Al promediar valores dentro del octree si dos vóxeles opacos con diferentes colores provenientes de por ejemplo, dos paredes planas, el color resultante del voxel describe ambas paredes como si estas fueran transparentes. Otro problema resulta también de promediar opacidad entre vóxeles totalmente transparentes y vóxeles totalmente opacos, resultando un valor filtrado semitransparente. Esto puede resultar en fugas de luz a través de la geometría en la escena. Para solventar este problema se realiza una representación anisotrópica de los vóxeles durante el proceso de mip-mapping. En vez de tener un solo canal de valores filtrados sin dirección, ahora los valores serán filtrados de forma direccional, almacenando 6 canales por cada eje positivo y negativo. Un valor direccional es calculado realizando un paso de integración volumétrica en profundidad y promediando los cuatro valores direccionales para obtener el valor resultante según una dirección.

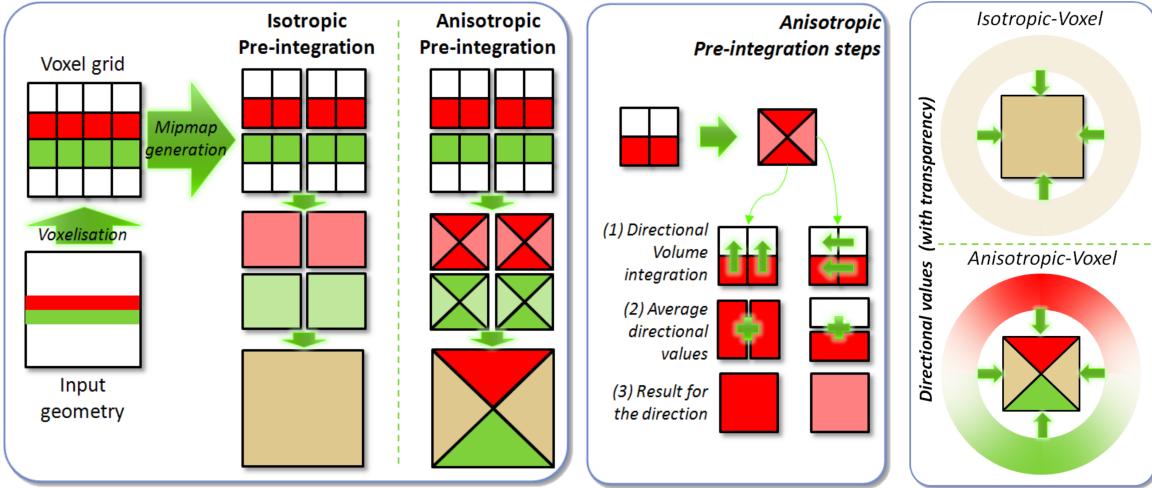


Figura 1.23: La caja izquierda describe el proceso de mipmaping de véxeles sin (izquierda) y con (derecha) filtrado anisotrópico. Los pasos para la integración direccional en la caja central. La caja derecha muestra la diferencia entre un véxel anisotrópico como resultado final versus un véxel isotrópico filtrado por kernel gaussiano [21].

1.7.4.6. Captura de Iluminación Directa

Para el cálculo de iluminación indirecta es necesario describir como la radiancia incidente es almacenada en los nodos del árbol. Este proceso está inspirado en RSM, donde la escena es renderizada desde el punto de vista de la fuente de luz y se utiliza rasterización estándar para almacenar las posiciones de los fragmentos en una textura. Cada pixel en esta textura representa un fotón que rebota en escena. Esta textura se le llama mapa de luz-vista o *light-view map*. Luego de generar este mapa es necesario almacenar los fotones en el árbol octree. Los fotones son almacenados como una distribución direccional y energía proporcional al casquete esférico del ángulo sólido del pixel visto desde la luz.

El procesamiento de la textura de fotones sobre el árbol se realiza por fragmento con un *fragment shader*. Como usualmente la dimensión del *light-view map* es mayor a la resolución de la cuadricula de véxeles se puede asumir que los fotones serán almacenados directamente en las hojas del árbol octree. Además, los fotones siempre pueden ser almacenados en el nivel más fino de detalle en la representación con véxeles porque estos describen información de la superficie geométrica. Es posible que muchos fotones terminen sobre un mismo véxel, por esto es necesario utilizar adicción atómica para garantizar coherencia entre los hilos generados por cada fragmento.

Capítulo 2

Solución Propuesta

Capítulo 3

Implementación

Capítulo 4

Pruebas y Resultados

4.1. Entorno de Pruebas

Todos los experimentos realizados en esta sección fueron ejecutados en un computador de escritorio con las siguientes características de hardware:

1. Procesador AMD Phenom II X6 1055T 2.8 Ghz
2. 8 GB de Memoria RAM DD3
3. Disco duro de 1TB
4. Tarjeta gráfica AMD R9 380
5. Sistema Operativo Windows 7 de 64 bits.

4.1.1. Configuración de la Aplicación

Con respecto a la representación en véxeles distintos pasos del algoritmo solo se realizan dependiendo de ciertos eventos en escena. Sin embargo cada paso de este algoritmo es dependiente de pasos anteriores. El sombreado se véxeles solo necesita volver a realizarse bajo algún cambio en los parámetros de iluminación, al actualizarse el sombreado también deben realizarse todos los pasos siguientes. Igualmente sucede con la voxelización dinámica bajo algún cambio sobre un objeto dinámico. La aplicación también permite el cambio de parámetros en la escena estática, esto implica realizar todos los pasos del algoritmo. En contraste el trazado de conos se realiza constantemente por frame durante el paso de iluminación del sombreado diferido.

Para ejecutar pruebas que comprendan todos los aspectos del algoritmo es necesario que luces y objetos se encuentren registrando cambios constantemente. Para simplificar este proceso la aplicación provee un modo de actualización forzosa por frame. Esto permite simular situaciones de estrés donde tanto objetos como luces en escena se encuentran bajo constantes cambios. Para ciertos experimentos este modo será desactivado de manera que solo se obtengan datos relevantes a ese ambiente de prueba.

4.2. Escenarios de Estudio

En esta sección se describe las distintas escenas y configuraciones a realizar para las distintas pruebas sobre la aplicación.

4.2.1. Escenas de Prueba y Objetos

Para la realización de pruebas de precisión y rendimiento se utilizaran siete escenas las cuales clasificaremos en dos categorías, escenas completas y ambientes de pruebas que llamaremos escenas *sandbox*. Todas las escenas comprenden varios niveles de complejidad con respecto a geometría e iluminación. La aplicación también incluye una serie de modelos precargados los cuales serán agregados a las escenas, estos objetos son considerados dinámicos.

4.2.1.1. Escenas Completas

Como escenas completas consideramos cuatro escenas comunes para pruebas de iluminación global. Estas escenas presentan varios niveles de complejidad geométrica y de propagación de luz. En la siguiente tabla listamos sus nombres y atributos.

Nombre	Vértices	Triángulos	Texturas	Geometría	Iluminación
Sponza	153.635	278.163	Si	Compleja	Compleja
Conference	194.399	331.179	No	Compleja	Compleja
Sibenik	40.479	75.283	Si	Media	Media
Cornell Box	72	36	No	Simple	Simple

Tabla 4.1: Escenas completas y sus atributos.

Sponza: Modelo del atrio del palacio Sponza en Dubrovnik, este modelo originalmente realizado por Marko Dabrovic y luego remodelado por Frank Meinl de Crytek con nuevos elementos como cortinas, vasos y plantas, además de mapas especulares, albedo y normales. Esta es una escena de dimensión considerable con propagación de luz compleja, especialmente en las áreas ocluidas por las cortinas y los pasillos superiores donde la luz difusa rebota luego de pasar a través de varias columnas.



Figura 4.1: Sponza con solo iluminación directa desde una luz direccional más luz ambiental para visualizar las áreas sombreadas.

Conference: Un modelo 3D basado en una sala de conferencias real del *Lawrence Berkeley National Laboratory*. Esa es una escena pequeña pero de gran complejidad geométrica con muchos objetos repetidos. El transporte de luz es particularmente complejo en partes ocluidas como debajo de sillas o la mesa central, la escena es un interior cerrado y la única luz directa exterior entra a través de unas pequeñas rendijas cerca de las cortinas.

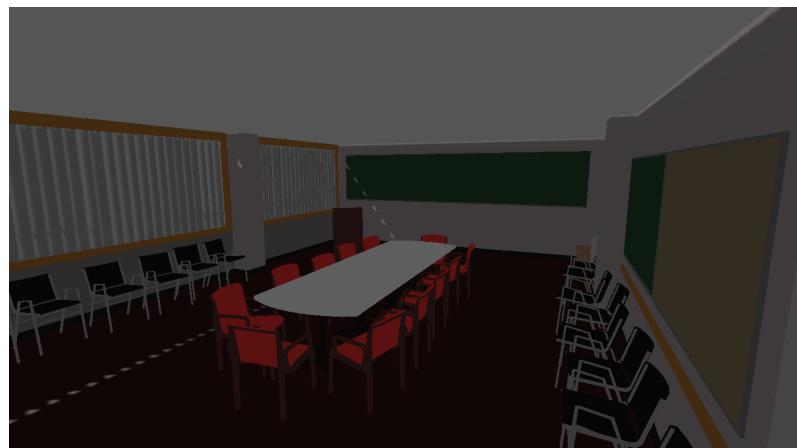


Figura 4.2: Conference con solo iluminación directa desde una luz direccional más luz ambiental para visualizar las áreas sombreadas.

Sibenik: El interior de una catedral. La luz entra a ella a través de ventanas y se propaga por toda la escena. La escena tiene una sección con columnas ideal para probar sombras. En esta misma sección de columnas hay un piso de mármol que permite apreciar reflexión especular. Una alfombra roja cubre gran parte de la catedral esta permite la visualización de reflexión difusa.



Figura 4.3: Sibenik con solo iluminación directa desde una luz direccional más luz ambiental para visualizar las áreas sombreadas.

Cornell Box: Este es un modelo popular para la visualización de iluminación global. Creado por Donald Greenberg y estudiantes de *Cornell University*. El Cornell Box está compuesto por una caja con dos paredes de colores, a la izquierda una roja y a la derecha una verde, estas permiten observar reflexión difusa y mezclada de colores. El resto de la caja es blanca. En la pared superior se observa un pequeño plano pensando para simular una fuente de luz de área. Dentro de esta caja se encuentran dos cajas más, una pequeña y otra más alta

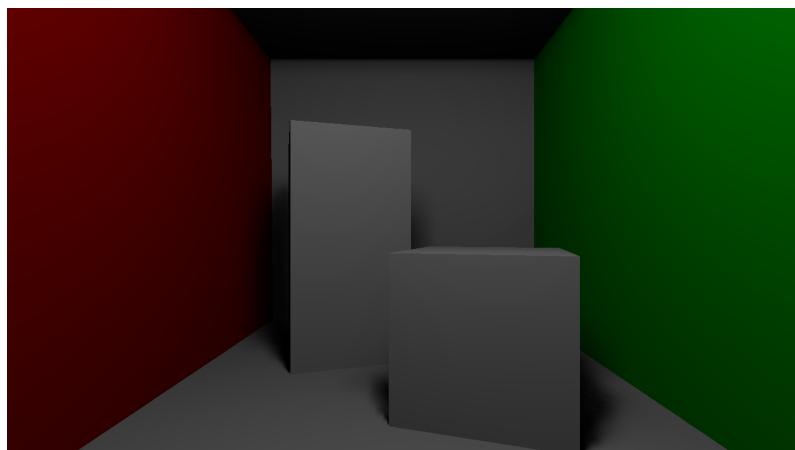


Figura 4.4: Cornell Box con solo iluminación directa desde una luz puntual con trazado de sombras.

4.2.1.2. Escenas Sandbox

Las escenas sandbox son entornos simples y vacíos donde se realizaran pruebas de aspectos específicos de la aplicación. Estas escenas no presentan mayor complejidad geométrica o complejidad de iluminación. En la siguiente tabla listamos sus nombres y atributos.

Nombre	Vértices	Triángulos	Texturas	Geometría	Iluminación
Light Box	4.098	8.192	No	Simple	Media
Plane Test	16	24	No	Simple	Simple
Cornell Box Vacío	24	12	No	Simple	Simple

Tabla 4.2: Escenas Sandbox y sus atributos.

Light Box es un interior donde los bordes son tan suaves que no hay oclusión ambiental, el interior es totalmente blanco y no hay cambios bruscos en geometría, esto maximiza la propagación de luz en esta escena. Plane Test es sencillamente un plano, en una esquina se encuentra un cuboide fino pero de gran altura para permitir la voxelización de objetos sobre el plano. Cornell Box Vacío es la escena completa Cornell Box sin las cajas internas.



Escena Light Box con solo iluminación directa desde una luz puntual de baja intensidad.

Escena Plane Test con solo iluminación directa desde una luz direccional.

Escena Cornell Box Vacío con iluminación indirecta y solo una luz puntual.

Figura 4.5: Escenas sandbox.

4.2.1.3. Objetos Precargados

La aplicación cuenta con una serie de objetos tridimensionales que pueden ser agregados a las escenas. Cada uno de estos objetos puede tener su propio material y matriz espacial. Estos objetos son considerados dinámicos por la aplicación. Los objetos están divididos en dos categorías: primitivas y modelos. Las primitivas son: Icosaedro, Cubo, Esfera, Cono, Toro, Plano y Cilindro. Los modelos son: Stanford Happy Buddha, Stanford Bunny, Stanford Dragon y Utah Teapot.

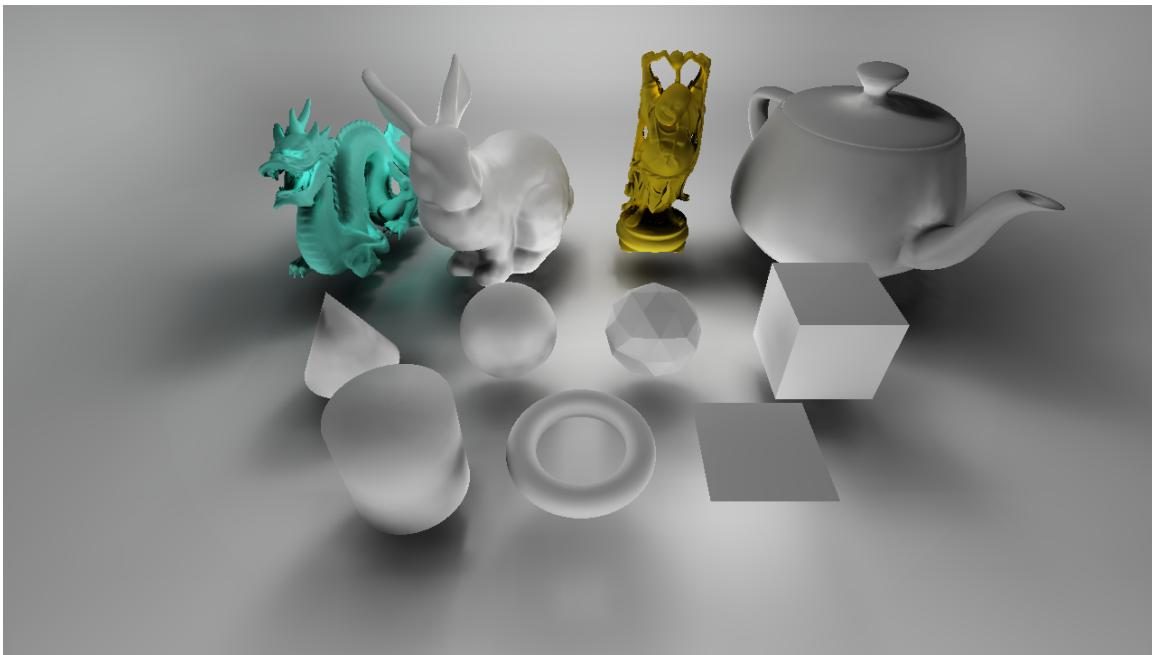


Figura 4.6: Objetos precargados. Arriba se encuentra los modelos Stanford Dragon, Stanford Bunny, Stanford Happy Buddha y Utah Teapot. El resto son primitivas, de izquierda a derecha: Cono, Esfera, Icosaedro, Cubo, Cilindro, Toro y Plano. Todos los objetos se encuentran en la escena Light Box utilizando una luz focal con trazado de sombras e iluminación indirecta.

4.2.1.4. Criterios de Complejidad Geométrica e Iluminación

Para determinar la complejidad geométrica de cada escena se consideró la cantidad de triángulos que estas poseen y no la cantidad de objetos a renderizar ya que este es el factor que mayor afecta el proceso de voxelización.

Para determinar la complejidad de iluminación se consideraron los siguientes aspectos:

1. El nivel de detalle de la representación en vértices de la escena. En la escena Cornell Box la representación en vértices es muy similar a la geometría original. En contraste la escena Sponza difiere considerablemente de la geometría original en lugares con detalles finos como los vasos y plantas de esta escena.
2. Secciones ocluidas y facilidad de propagación de luz sobre distintas áreas de la escena. En la escena Cornell Box la luz se propaga fácilmente desde cualquier superficie iluminada a otras partes de la escena. En contraste escenas como Conference o Sponza tienen complicadas secciones de geometría como las sillas en Conference y las cortinas inferiores en Sponza que reducen la luz a pequeños haces.

4.3. Estudio de Rendimiento

Para el estudio de rendimiento utilizo la cantidad de tiempo entre rutinas relevantes a cada prueba en milisegundos. Para obtener tiempos precisos en GPU se utilizó el software GPU PerfStudio y análisis de frame. Este software nos permite observar los detalles en el pipeline de renderizado y tiempo total de cada llamada de dibujo o *draw call* al API gráfico OpenGL.

4.3.1. Prueba Base

La prueba base de rendimiento comprende todos los pasos del algoritmo sin trazado de sombras o modificaciones sobre la escena estática. Para esta prueba se colocó la aplicación en actualización forzosa de tal manera que todos los pasos del algoritmo se realicen por frame. También se varían tres aspectos de la aplicación: resolución de pantalla, dimensión del volumen de véxeles y factor de longitud de marcha del cono.

Esta prueba utiliza todas las escenas completas. La configuración del escenario comprende una luz direccional con mapeado de sombras, y tres modelos precargados. Los modelos utilizados son Stanford Happy Buddha, Stanford Dragon y Stanford Bunny, estos modelos fueron seleccionados por su complejidad geométrica. Cada modelo tiene su propio material con un cono especular con apertura de 45, 27 y 9 grados respectivamente. La cámara en escena es colocada de tal forma que todos los fragmentos visibles formen parte del trazado de conos.

Escena	Voxelización Estática	Limpieza de Vóxeles Dinámicos	Voxelización Dinámica	Sombreado de Vóxeles	Mipmapping Direccional	Illuminación Global de Vóxeles	Mipmapping Direccional	Trazado de Conos con Vóxeles	Tiempo Dinámico
Sibenik	1.80	0.58	2.11	0.95	1.39	3.88	1.38	7.31	17.60
Cornell Box	0.51	0.78	1.30	1.33	1.38	8.41	1.37	7.23	21.81
Conference	46.04	0.56	1.52	0.86	1.38	3.23	1.37	7.50	16.42
Sponza	11.29	0.60	2.03	1.13	1.37	5.44	1.38	7.01	18.97

Tabla 4.3: Rendimiento de todas las partes del algoritmo en distintas escenas utilizando un volumen de con resolución 256^3 , factor de longitud de marcha de 0,5 y resolución de pantalla de 1280×720 . Todos los tiempos en milisegundos.

		Tiempo Dinámico									
		Trazado de Conos con Vóxeles									
		Mipmapping Direccional									
		Illuminación Global de Vóxeles									
		Sombreado de Vóxeles									
		Voxelización Dinámica									
		Limpieza de Vóxeles Dinámicos									
		Voxelización Estática									
		Resolución Volumen Escena									
64^3	64^3	Sibenik	3.00	0.01	9.17	0.04	0.11	0.12	0.10	5.17	14.74
		Cornell Box	0.07	0.02	2.13	0.06	0.11	0.29	0.11	4.79	7.51
		Conference	39.68	0.01	5.98	0.03	0.11	0.10	0.11	5.38	11.72
		Sponza	22.87	0.01	13.93	0.05	0.17	0.13	0.17	5.49	19.97
128^3	128^3	Sibenik	2.17	0.08	3.93	0.17	0.26	0.62	0.26	2.17	3.93
		Cornell Box	0.16	0.10	1.38	0.28	0.26	1.62	0.26	0.16	1.38
		Conference	39.19	0.07	2.47	0.15	0.26	0.54	0.26	39.19	2.47
		Sponza	15.15	0.08	4.00	0.20	0.26	0.82	0.26	15.15	4.00
512^3	512^3	Sibenik	2.36	4.51	1.35	6.06	10.92	23.23	10.91	8.57	65.55
		Cornell Box	2.17	6.08	1.80	7.54	10.70	41.14	10.87	7.87	86.00
		Conference	47.44	4.46	1.28	5.63	11.57	18.70	11.35	8.56	61.56
		Sponza	9.05	4.55	1.39	6.87	11.16	28.09	10.79	7.64	70.49

Tabla 4.4: Rendimiento en contraste con la tabla 4.3 considerando distintas resoluciones para la representación en vóxeles, todos los pasos del algoritmo se ven afectados por este parámetro.

Resolución Pantalla	1920x1080	1280x720
Escena	Trazado de Conos con Vóxeles	Trazado de Conos con Vóxeles
Sibenik	14.30	7.31
Cornell Box	15.17	7.23
Conference	16.27	7.50
Sponza	16.69	7.01

Tabla 4.5: Rendimiento con una mayor resolución de pantalla. Esto solo afecta el trazado de conos con vóxeles.

Considerando como tiempos interactivos todo resultado por debajo de $33.3ms$ o aproximadamente 30 cuadros por segundo. Nuestra implementación logra colocarse por debajo de estos tiempos en todos los casos excepto los casos que comprenden resoluciones de volumenes mayores a 256^3 .

Longitud de Marcha	0.1	0.25	0.5	2.5
Escena	Trazado de Conos con Vóxeles			
Sibenik	29.55	12.73	7.31	2.67
Cornell Box	26.80	11.57	7.23	2.68
Conference	29.73	12.93	7.50	2.85
Sponza	29.81	12.92	7.01	2.81

Tabla 4.6: Rendimiento con distintas configuraciones de la longitud de marcha del cono. Esto solo afecta el trazado de conos con vóxeles.

4.3.1.1. Densidad Geometrica y Velocidad de Voxelizacion.

La cantidad de triangulos dentro del volumen que representa un voxel afecta la velocidad de voxelizacion, esto debido a que este proceso requiere sincronizacion entre distintos hilos por fragmento. Si hay mucha geometria dentro de un voxel esto hace que aumente la cantidad de hilos en espera por la culminacion de otros.

Podemos observar este efecto especialmente en la escena Conference. Esta es una escena pequena en escala, sin embargo de todas las escenas completas es la que posee mayor cantidad de triangulos. Otro caso notable es la voxelizacion dinamica de la escena Sponza, se puede observar que medida que la resolucion del voxel disminuye el tiempo de voxelizacion dinamica aumenta.

4.3.1.2. Espaciado Interior y Velocidad de Trazado.

4.3.2. Trazado de Sombras y Volumen de Visibilidad

4.3.3. Apertura del Cono Especular

4.3.4. Apertura del Cono de Sombreado

4.3.5. Estudio de Calidad de Imagen

4.3.6. Estudio de Memoria

Capítulo 5

Conclusión

Bibliografía

- [1] P. Christensen. (2003) Global Illumination and All That: Pixar Animation Studios. <http://renderman.pixar.com/view/global-illumination-and-all-that>.
- [2] P. Dutre, K. Bala, P. Bekaert, y P. Shirley, *Advanced Global Illumination*, 2da. ed. AK Peters Ltd, 2006.
- [3] A. Fournier, “Normal distribution functions and multiple surfaces,” en *Graphics Interface '92 Workshop on Local Illumination*, Vancouver, BC, Canada, 11 Mayo 1992, pp. 45–52.
- [4] P. Bromiley, “Products and convolutions of gaussian distributions,” TINA Vision, Internal Report 2003-003, 2003.
- [5] M. Toksvig, “Mipmapping normal maps,” *J. Graphics Tools*, vol. 10, núm. 3, pp. 65–71, 2005. [En línea]. Disponible en: <http://dx.doi.org/10.1080/2151237X.2005.10129203>
- [6] Disney. (2012) Disney’s BRDF Explorer. <https://github.com/wdas/brdf>.
- [7] J. T. Kajiya, “The Rendering Equation,” en *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH ’86. New York, NY, USA: ACM, 1986, pp. 143–150. [En línea]. Disponible en: <http://doi.acm.org/10.1145/15922.15902>
- [8] D. Ritchie y R. Ramamoorthi, “Global Illumination and the Rendering Equation,” <https://inst.eecs.berkeley.edu/~cs294-13/fa09/lectures/scribe-lecture3.pdf>, 2009, university of California, Berkeley.
- [9] H. Elias, “Radiosity: The Workings of a Radiosity Renderer,” <http://freespace.virgin.net/hugo.elias/radiosity/radiosity.htm>, 2000.
- [10] C. M. Goral, K. E. Torrance, D. P. Greenberg, y B. Battaile, “Modeling the Interaction of Light Between Diffuse Surfaces,” *SIGGRAPH Comput. Graph.*, vol. 18, núm. 3, pp. 213–222, Ene. 1984. [En línea]. Disponible en: <http://doi.acm.org/10.1145/964965.808601>

- [11] J. Arvo, “Backward Ray Tracing,” en *In ACM SIGGRAPH '86 Course Notes - Developments in Ray Tracing*, 1986, pp. 259–263.
- [12] L. Do, “HW6: Ray Tracing Extension,” <https://inst.eecs.berkeley.edu/~cs184/sp12/assignments/Archive/HW6/>, 2013.
- [13] L. Williams, “Casting curved shadows on curved surfaces,” en *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '78. New York, NY, USA: ACM, 1978, pp. 270–274. [En línea]. Disponible en: <http://doi.acm.org/10.1145/800248.807402>
- [14] M. Deering, S. Winner, B. Schediwy, C. Duffy, y N. Hunt, “The triangle processor and normal vector shader: A vlsi system for high performance graphics,” en *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '88. New York, NY, USA: ACM, 1988, pp. 21–30. [En línea]. Disponible en: <http://doi.acm.org/10.1145/54852.378468>
- [15] T. Saito y T. Takahashi, “Comprehensible rendering of 3-d shapes,” *SIGGRAPH Comput. Graph.*, vol. 24, núm. 4, pp. 197–206, Sep. 1990. [En línea]. Disponible en: <http://doi.acm.org/10.1145/97880.97901>
- [16] A. Keller, “Instant radiosity,” en *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997, pp. 49–56. [En línea]. Disponible en: <http://dx.doi.org/10.1145/258734.258769>
- [17] C. Dachsbacher, J. Křivánek, M. Hašan, A. Arbree, B. Walter, y J. Novák, “Scalable realistic rendering with many-light methods,” *Computer Graphics Forum*, vol. 33, núm. 1, pp. 88–104, 2014.
- [18] C. Dachsbacher y M. Stamminger, “Reflective shadow maps,” en *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, I3D '05. New York, NY, USA: ACM, 2005, pp. 203–231. [En línea]. Disponible en: <http://doi.acm.org/10.1145/1053427.1053460>
- [19] A. Kaplanyan y C. Dachsbacher, “Cascaded light propagation volumes for real-time indirect illumination,” en *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '10. New York, NY, USA: ACM, 2010, pp. 99–107. [En línea]. Disponible en: <http://doi.acm.org/10.1145/1730804.1730821>

- [20] K. Crane, I. Llamas, y S. Tariq, *Real Time Simulation and Rendering of 3D Fluids*. Addison-Wesley, 2007, cap. 30.
- [21] C. Crassin, F. Neyret, M. Sainz, S. Green, y E. Eisemann, “Interactive indirect illumination using voxel cone tracing,” *Computer Graphics Forum (Proceedings of Pacific Graphics 2011)*, vol. 30, núm. 7, sep 2011. [En línea]. Disponible en: <http://maverick.inria.fr/Publications/2011/CNSGE11b>