UNIVERSIDAD CENTRAL DE VENEZUELA

FACULTAD DE CIENCIAS

ESCUELA DE COMPUTACIÓN

IMPLEMENTACIÓN DE LA TÉCNICA DE SCATTERING EN DATOS VOLUMÉTRICOS Y MALLAS POLIGONALES

Trabajo especial de grado presentado ante la ilustre

Universidad Central de Venezuela por

Br. José Manuel Alvarez García

Br. Jesús Alberto Cibeira Castiblanco

Tutor:

Prof. Rhadamés Carmona

Caracas, Julio 2018

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

ACTA DE VEREDICTO

Quienes suscriben, miembros del jurado designado por el Consejo de la Escuela de Computación de la Facultad de Ciencias de la Universidad Central de Venezuela, para examinar el Trabajo Especial de Grado titulado **"Implementación de la técnica de Scattering en datos volumétricos y mallas poligonales"** y presentado por los Bachilleres: José Manuel Alvarez García C.I: 25.038.805 y Jesús Alberto Cibeira Castiblanco C.I: 24.203.933, 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 26 de julio de 2018, a las 1:00PM, para que los autores lo defendieran en forma pública en el Centro de Computación Gráfica, lo cual realizaron mediante una presentación oral de su contenido y luego respondieron 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 aprobar con la nota de 20 puntos.

En fe de lo cual se levanta la presente Acta, en Caracas el día de 26 de julio de 2018, dejándose también constancia de que actuó como Coordinador de Jurado el Profesor Rhadamés Carmona.

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

Prof. Carmona Rhadamés

(Tutor)

Prof. Hernandez Walter

(Jurado)

Prof. Navarro Héctor

(Jurado)

RESUMEN

El Centro de Computación Gráfica de la Facultad de Ciencias de la Universidad Central de Venezuela está formado por un grupo de investigadores expertos en el desarrollo e implementación de *software* gráfico. El objetivo de este grupo consiste en promover la experticia en conocimientos de sistemas de visualización de datos por computador. El presente trabajo especial de grado consiste en el desarrollo e implementación de la técnica de la dispersión de la luz en modelos mallados y en volúmenes de datos. El fenómeno de dispersión de la luz o *scattering* consiste en el desvío de la luz causado por partículas encontradas en su trayectoria. Esta técnica de iluminación no ha sido implementada en el Centro de Computación Gráfica de la Universidad Central de Venezuela, por lo cual puede complementar otros trabajos realizados recientemente en iluminación global, e incluso crear una posible línea de investigación para trabajos futuros. Debido a los motivos descritos, la finalidad del presente trabajo especial de grado consiste en la implementación de una aplicación de escritorio que simule el fenómeno de *scattering*. Para el desarrollo de la misma se utilizaron diversas tecnologías, tales como el lenguaje de programación C++ y OpenGL® para el despliegue de gráficos 3D, y a su vez, se utilizó una adaptación de la metodología de desarrollo de *software* denominada proceso unificado ágil (AUP).

Palabras clave: aplicación de escritorio, dispersión de la luz, iluminación global, gráficos 3D.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO I - INTRODUCCIÓN

En la actualidad, los investigadores y científicos utilizan herramientas computacionales para visualizar los datos con los que trabajan, y así poder realizar un análisis de los mismos. Uno de los enfoques que ha tenido el mayor auge en los últimos años es la visualización de volúmenes, que consiste en el despliegue de uno o varios conjuntos de datos tridimensionales en la pantalla, de forma que el usuario pueda entenderlos, estudiarlos e interpretarlos satisfactoriamente.

Muchas técnicas se han desarrollado en los últimos tiempos para disminuir el problema de espacio y procesamiento de los datos, dado que el poder de cómputo del *hardware* gráfico, del procesador central y el ancho de banda de los buses del sistema se han incrementado con el tiempo. Diversos investigadores han propuesto métodos para poder visualizar esta gran cantidad de volúmenes datos. A su vez, también han propuesto el despliegue de conjunto de datos de mallados realizando una triangulación de todos sus vértices y mostrar finalmente una superficie en tres dimensiones.

La mayoría de estas investigaciones están enfocadas sólo en el despliegue volúmenes o mallas poligonales, pero muy pocas consideran una escena mixta, compuesta por conjuntos de datos volumétricos y mallados. La cirugía virtual y la edición de volúmenes son sólo algunos ejemplos de aplicaciones que requieren la interacción de conjuntos de datos volumétricos con objetos virtuales representados comúnmente por mallas poligonales. Un cuarto de cirugía virtual fue diseñado en la Universidad Central de Venezuela **[1]** con el objetivo de simular cirugías de rodillas. Este tipo de aplicación requiere de datos mixtos debido a que los huesos y los músculos son datos que provienen de tomografías, mientras que los instrumentos quirúrgicos son representados con mallados.

La síntesis realística de las imágenes ha sido un componente fundamental en la computación gráfica. Con el auge de aplicaciones que varían desde el entretenimiento (películas, efectos especiales o videojuegos) hasta el diseño de iluminación y arquitectura, la necesidad de modelar realísticamente la propagación y la dispersión de la luz en entornos complejos continúa siendo un desafío para los recursos computacionales. El cálculo preciso de las características de reflectancia de un objeto es importante en muchos aspectos: además de determinar el color y los atributos de los mismos, su información también puede ser utilizada para diversos propósitos tales como la restauración y conservación de detalles en las escenas a representar.

A continuación se presenta la problemática identificada en los distintos métodos investigados sobre la aplicación de la técnica de *scattering* en el despliegue de volúmenes y de mallas poligonales, así como el objetivo general, los objetivos específicos, justificación, el alcance, antecedentes y la metodología utilizada.

1 PROBLEMA

La técnica de *scattering* incrementa el realismo en la visualización, como se puede notar en la **Fig. 1.1**. Esta técnica de iluminación global no ha sido implementada en el Centro de Computación Gráfica de la Universidad Central de Venezuela, por lo cual puede complementar otros trabajos realizados recientemente en iluminación global, e incluso crear una posible línea de investigación para trabajos futuros. En datos volumétricos sólo se han trabajado escenas limitadas a iluminación local, mientras que en mallas poligonales, a pesar de que se han construido escenas con iluminación global, no se ha desarrollado una implementación para simular la dispersión de la luz, y a su vez, tampoco se ha implementado una escena híbrida donde se desplieguen mallas poligonales y volúmenes de datos a la vez aplicando dicho efecto. Las técnicas que se han aplicado a diferentes modelos pueden calcular efectos difusos de manera precisa, pero no la dispersión de la luz. Estudiando las diferentes implementaciones de esta técnica en datos volumétricos y en mallados

poligonales, se observa que se pueden incluir efectos de sombras y translucidez para lograr resultados más realistas (*ver **Fig. 1.1***).

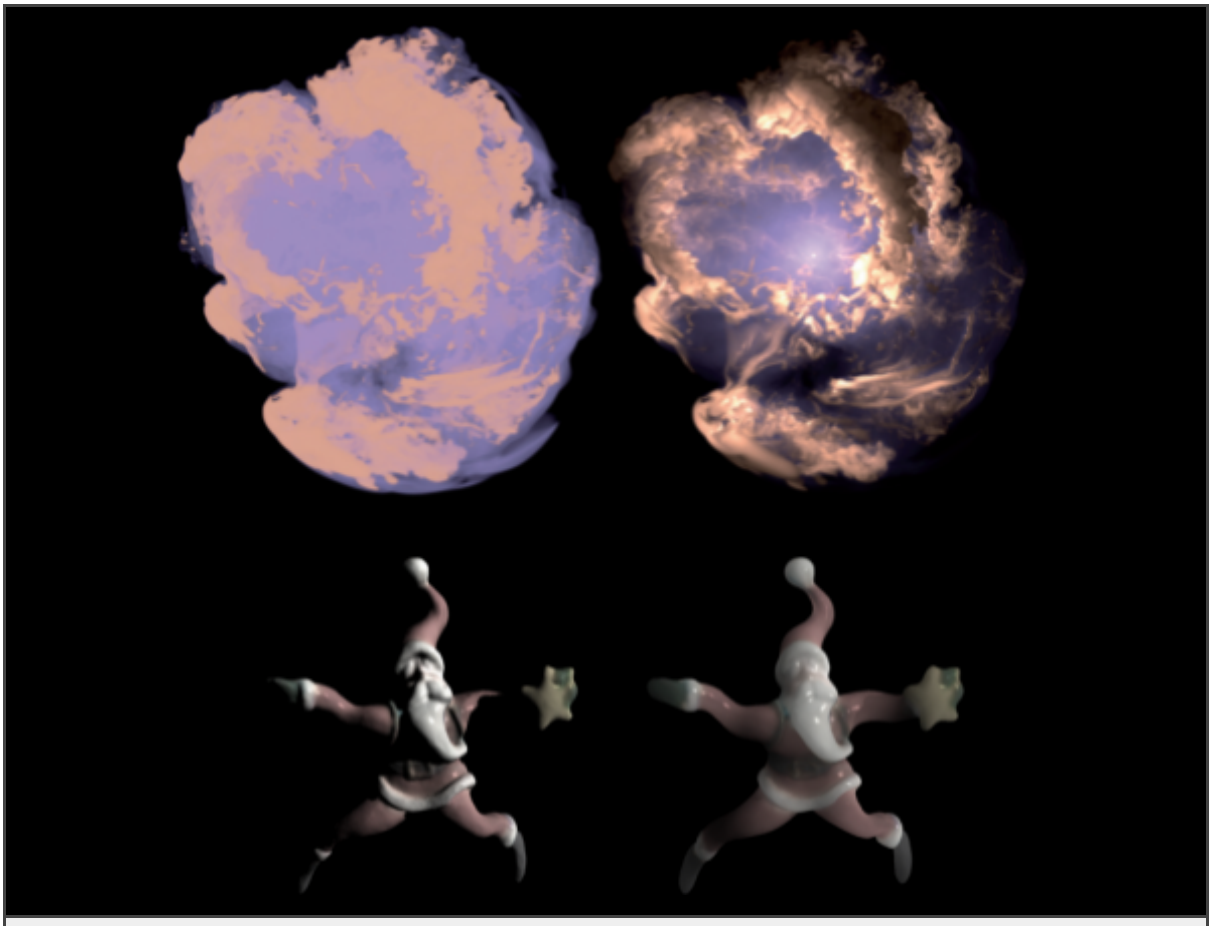

Figura 1.1: *Scattering en ambos tipos de dato*. En la fila superior se puede observar el volumen de una supernova con un modelo óptico de absorción-emisión clásico (izquierda), y el mismo volumen aplicando la técnica de *scattering* (derecha). En la fila inferior se puede observar el modelo mallado del personaje de Santa Claus con el modelo de iluminación *Blinn-Phong*[1] (izquierda), y el mismo mallado aplicando la técnica de *scattering* (derecha).

2 ANTECEDENTES

A partir de las implementaciones en el ámbito de la iluminación global, son muchas las investigaciones que se han llevado a cabo con el propósito de desarrollar nuevas técnicas para la simulación de efectos de dispersión de la luz, así como para incluir mejoras en dichas técnicas. Cabe destacar que estas técnicas poseen diferentes salidas para sus resultados finales, tales como despliegue en tiempo real en GPU y despliegue *offline*, donde los cálculos son realizados en CPU en vez de utilizar el *hardware* gráfico.

La técnica de *scattering* ha sido ampliamente aplicada en diversos proyectos. En 1993, Hanrahan y Krueger **[67]** aproximaron la técnica de *scattering* simple con una función BRDF[2], y utilizaron una solución aplicando un algoritmo de Monte-Carlo para implementar *scattering* múltiple.

[1]**Blinn-Phong:** es una extensión del modelo de reflexión *Phong*, donde, para hacer más eficiente el cálculo de la iluminación, se utiliza un vector medio en lugar de utilizar un vector reflejado.
[2]**BRDF**: consiste en la función de distribución de reflectancia bidireccional y describe la tasa diferencial entre la luz saliente y la entrante.

En el 2001, Jensen **[70]** simplificó la técnica de *scattering* múltiple utilizando una aproximación BRDF y rayos de sombra que no toman en cuenta la refracción en el *scattering* simple **[71][72]**.

Diversos investigadores han utilizado el principio de Fermat[3] para lograr reflexiones en superficies curvas. En 1992, Mitchell y Hanrahan **[73]** aplicaron geometría diferencial y análisis de intervalos para calcular los puntos exactos de reflexión, además de discutir cómo conseguir los puntos de refracción en una superficie. En el 2000, Chen y Arvo **[74]**, calculan una rápida aproximación de rayos reflejados aplicando una serie de Taylor[4]. Ambos enfoques están restringidos a superficies implícitas con ecuaciones conocidas, y son usualmente aplicados en mallas triangulares.

Un derivado del algoritmo de *ray tracing* consiste en la técnica de *beam tracing*, la cual reemplaza rayos que no tienen grosor con haces de luz. La luz refractada a través de la superficie puede ser representada por un conjunto de rayos volumétricos. En este enfoque, las aproximaciones de cáusticas[5] pueden ser construidas muestreando la intensidad en estos rayos, tal como lo hicieron Nishita y Nakamae **[75]** en 1994, Iwasaki **[76]** en el 2003 y Ernst **[77]** en el 2005. Cabe destacar que este método sólo genera aproximaciones y tiene problemas representando sombras en las cáusticas.

Ament y Weiskopf **[4]** presentan un algoritmo simple de trazado de rayos que emplea una tabla de integración previa, que es combinable con muchas otras técnicas de *rendering* de volúmenes, tales como sombreado basado en gradientes y oclusión ambiental. El beneficio de este enfoque es la visualización interactiva de estructuras volumétricas que emplean la dispersión de la iluminación, donde se pueden simular efectos de translucidez en la región ambiental de cada muestra.

3 ANTECEDENTES PRINCIPALES

A continuación se mostrarán en detalle dos trabajos que servirán de base en el desarrollo del trabajo especial de grado: el trabajo de Dal Corso, Frisvad, Mosegaard y Bærentze **[2]** para la técnica de dispersión de la luz en modelos mallados, y el trabajo de Ropinski, Döring y Rezk-Salama **[3]** para la técnica de dispersión de la luz en volúmenes de datos. Se han seleccionado estas investigaciones para el desarrollo del trabajo especial de grado, ya que son técnicas modernas y eficientes en los cálculos. Ambas implementaciones comparten la característica de simular *scattering* simple. Éstas permiten un despliegue en tiempo real, gracias a que se pre-calculan los datos de la iluminación y sólo son modificados si se produce un cambio en la escena, como por ejemplo la posición de la luz, transformaciones al modelo ó modificaciones en los parámetros de la dispersión (parámetro de asimetría, radio de dispersión, material difuso, entre otros).

3.1 SUBSURFACE SCATTERING INTERACTIVO Y TRANSPORTE EMERGENTE DE LA LUZ

La técnica de *subsurface scattering* (dispersión subsuperficial de la luz) es un fenómeno físico que ocurre en materiales translúcidos, por ejemplo leche, miel, piel, mármol y cera de vela. Es posible producir la apariencia cualitativa de la translucidez utilizando técnicas de despliegue interactivo de volúmenes **[8]**, pero dichas técnicas no son siempre precisas. Con el auge de los modelos analíticos para el *subsurface scattering* **[9]**, es más factible construir técnicas más precisas para el despliegue interactivo de objetos translúcidos.

---

[3] **Principio de Fermat**: consiste en un fundamento que establece que la trayectoria que sigue un rayo de luz entre dos puntos cualesquiera es aquella donde se emplea un tiempo mínimo para recorrerla.

[4] **Serie de Taylor**: consiste en la representación de una función como la suma infinita de elementos que son calculados a partir de las derivadas de la función sobre un punto.

[5] **Cáusticas**: consiste en la forma donde se presentan los fenómenos de reflexión y refracción de la luz que incide sobre una superficie curva. Las cáusticas pueden verse como trozos de luz o bordes brillantes sobre una superficie.

Dal Corso, Frisvad, Mosegaard y Bærentze **[2]** presentan una técnica interactiva que soporta *subsurface scattering* direccional sin basarse en el precálculo o en una cuadrícula para la propagación volumétrica de la luz. Este método funciona para geometría deformable o generada proceduralmente. Debido a la reciprocidad del transporte de luz, se tratarán idealmente las direcciones de la luz incidente y emergente por igual. Sin embargo, esto sería muy costoso para una técnica interactiva. Para lograr la interactividad, se necesita almacenar los cálculos del *subsurface scattering*. Las técnicas existentes almacenan la luminancia transmitida (luz entrante en un punto de una superficie) y utilizan un filtro pre-calculado para evaluar la dispersión. Estas técnicas requieren que el método dependa sólo de la distancia, mientras que se necesita utilizar la dirección de la luz entrante.

En algunos modelos **[8]**, la dispersión de la luz se almacena por vértice. Para obtener efectos direccionales más detallados, se usan mapas de dispersión. Estos mapas se calculan sin requerir de la parametrización de la textura del objeto translúcido, mediante la representación del objeto desde múltiples vistas utilizando cámaras ortográficas. Para cada una de estas vistas, se calcula un mapa de dispersión (*scattered radiosity map*). Luego, se despliega de manera eficiente el objeto translúcido desde cualquier punto de vista mediante búsquedas en los *scattered radiosity maps*. Siempre que la fuente de luz y el objeto sean estacionarios, se pueden mezclar los mapas y así mejorar progresivamente el despliegue, además de calcular el transporte de luz emergente en la escena. Debido a que se incluye el transporte de luz emergente, este método es muy útil para la representación interactiva de escenas con la fuente de luz oculta detrás de un objeto translúcido.

3.2 ILUMINACIÓN DE VOLÚMENES SIMULANDO SCATTERING Y SOMBRAS

Hoy en día, el despliegue de volúmenes puede ser realizado en tiempo real en *hardware* gráfico, gracias a la variedad de técnicas de aceleración desarrolladas en el pasado. En la práctica, el *shading* en el mayor de los casos es simple, por ejemplo, iluminación local basada en gradientes, la cual contiene dos pasos. Inicialmente, dado que el modelo de iluminación *Phong*[6] ha sido originalmente desarrollado para iluminación en superficies, se basa en un gradiente bien definido que es considerado como un sustituto de la normal al ser aplicada a los datos volumétricos. Seguidamente, a excepción de los vóxeles utilizados para la estimación del gradiente, la información de las vecindades no es tomada en cuenta para los cálculos. Ésto restringe al modelo la simulación del fenómeno de iluminación local, aunque diversos modelos avanzados de iluminación mejorarían significativamente la comprensión espacial del medio a representar. Por lo general, el cálculo de gradientes normalizados conlleva usualmente a una buena aproximación de los vectores normales de isosuperficies en datos volumétricos.

Ropinski, Döring y Rezk-Salama **[3]**, presentan un modelo de iluminación volumétrica, que simula *scattering* y sombras para generar un despliegue de volúmenes realista. Aproximando el transporte de la luz en un medio participante no homogéneo, es posible obtener una implementación eficiente en GPU, para lograr los efectos deseados en tasas de *frames* interactivas. Además, en muchos casos las tasas de *frames* son incluso más altas que las logradas con *shading* basado en gradientes. Con el objetivo de generar una representación realística del modelo volumétrico, esta técnica soporta efectos de dispersión de luz interactiva aplicados en GPU utilizando la técnica de despliegue *ray casting*. El desvanecimiento de las sombras duras es un problema perceptivo de los efectos de la dispersión en un objeto, y cabe destacar que estas sombras son importantes para la comprensión espacial de una escena. En este modelo, el cálculo de las sombras no se encuentra

[6]**Phong:** es un modelo de iluminación local que describe la manera en que una superficie refleja la luz como la combinación de la reflexión difusa de superficies rugosas con la reflexión especular de superficies brillantes.

dentro del *scattering*; éste será realizado posteriormente con el fin de lograr bordes de sombras duras en la imagen final.

A pesar de que diversos investigadores **[2][3][4][5]** han propuesto métodos que implementan la técnica de *scattering*, desplegar en tiempo real los detalles de la escena generados con ésta ha sido un reto con respecto al *hardware* gráfico que se posee, ya que son algoritmos complejos y se tiene que recalcular la iluminación cada vez que ocurra un cambio en la escena. En este trabajo se lleva a cabo la implementación de la técnica del *scattering* en datos volumétricos y en mallas poligonales, basada en los trabajos investigados previamente.

Se plantea el desarrollo de un algoritmo para el despliegue de volúmenes y mallas poligonales en una escena híbrida aplicando la técnica de la dispersión de la luz. La visualización de modelos volumétricos será a partir de *ray casting* **[6][7]**. Además, se implementará la técnica de *scattering* basada en la utilización de un volumen de luz y aplicando sombras suaves. Adicionalmente, en los modelos mallados se aplicará la técnica de *scattering* utilizando el método multipaso **[2]** basado en el cálculo de *scattered radiosity maps* y la mezcla de los mismos en GPU.

4 OBJETIVO GENERAL

Implementar la técnica de dispersión de la luz (*scattering*) en volúmenes y mallados en una escena para que la imagen final presente efectos físicos realistas que se adecúen de la mejor manera a los resultados esperados de translucidez y sombras.

5 OBJETIVOS ESPECÍFICOS

- Implementar una aplicación interactiva que permita cargar modelos de volúmenes y mallas poligonales.
- Implementar una interfaz interactiva que permita el control de la función de transferencia, para visualizar los diferentes materiales de los volúmenes.
- Implementar la técnica de *scattering* y aplicarla a los modelos volumétricos.
- Implementar la técnica de *scattering* y aplicarla a los modelos mallados.
- Evaluar los resultados obtenidos en cuanto a rendimiento y tiempo de respuesta del programa con y sin la aplicación de la técnica de *scattering* en los diferentes modelos. Adicionalmente determinar el cambio de calidad perceptible por el ojo humano en la imagen final con respecto a otras técnicas de iluminación.

6 ALCANCE

La aplicación a desarrollar en este trabajo especial de grado permitirá desplegar en una escena híbrida un modelo volumétrico de 8 bits por muestra en formato *.raw* y un modelo mallado en formato *.obj*. A ambos modelos se les aplicará la técnica de dispersión de la luz (*scattering*) dependiendo de su tipo de dato. A su vez, La manipulación interactiva de la función de transferencia podrá ser realizada mediante una interfaz en la aplicación, con la posibilidad de agregar, editar y eliminar puntos de control.

En cualquier momento, la técnica de *scattering* podrá ser activada o desactivada interactivamente desde una interfaz en la aplicación. En caso de estar desactivada, al modelo volumétrico se aplicará el modelo de iluminación basado en gradientes de *Phong*, el cual se encarga

de aproximar vectores normales de isosuperficies en volúmenes; y para el modelo mallado se aplicará el modelo difuso *Lambert*[7] junto con el modelo especular *Blinn-Phong*.

6.1 PLATAFORMA DE DESARROLLO

El desarrollo del trabajo especial de grado consta en la utilización de un ambiente con las siguientes características:

El entorno de *hardware* consiste en un computador con las siguientes especificaciones: 12.0 GB de memoria RAM, procesador Intel Core i7-3770 con 3.40 GHz, tarjeta gráfica EVGA GeForce GTX 660 con una memoria de video dedicada de 2.0 GB. El sistema fue implementado y evaluado en la plataforma del sistema operativo Microsoft Windows 7. Además, se utilizó Microsoft Visual Studio 2015 como entorno de desarrollo.

6.1.1 LENGUAJE

El lenguaje de programación a utilizar es C++, el cual es recomendado para trabajar con el API de gráficos 3D OpenGL®. C++ es un lenguaje de programación para propósitos generales, orientado a objetos que provee facilidades de manipulación de memoria a bajo nivel.

6.1.2 HERRAMIENTAS

- **OpenGL®**: *Open Graphics Library* es un lenguaje de programación que define un API multiplataforma para desplegar gráficos 2D y 3D. Este API es generalmente utilizado para interactuar con la unidad de procesamiento gráfico (GPU), y así lograr el *rendering* acelerado por hardware. La versión utilizada en el desarrollo fue OpenGL® 4.5 **[10]**.

- **GLSL**: *OpenGL Shading Language* es el lenguaje principal de sombreado para OpenGL®. Es un lenguaje para crear *shaders* y está basado en el lenguaje de programación C. La versión utilizada en el desarrollo fue GLSL 4.5 **[11]**.

- **GLFW**: Es una biblioteca multiplataforma de código abierto desarrollada para OpenGL. Ésta provee una API simple para crear ventanas, contextos y superficies, eventos, entre otros. La versión utilizada en el desarrollo fue GLFW 2.5 **[12]**.

- **AntTweakBar**: Es una biblioteca sencilla disponible para los lenguajes C y C++ que le permite a los programadores añadir una interfaz gráfica de usuario intuitiva a aplicaciones basadas en OpenGL, y modificar parámetros interactivamente por pantalla. La versión utilizada en el desarrollo fue AntTweakBar 1.16 **[13]**.

- **GLM**: *OpenGL Mathematics* es una biblioteca matemática para aplicaciones gráficas basadas en OpenGL, la cual provee clases y funciones diseñadas e implementadas con las mismas convenciones y funcionalidades que provee la biblioteca GLSL, para su uso en el lenguaje de programación C++**.** La versión utilizada en el desarrollo fue GLM 0.9.8.5 **[14]**.

- **GLEW**: *OpenGL Extension Wrangler* es una biblioteca de carga de extensiones de código abierto y multiplataforma desarrollada en C y C ++. Ésta proporciona mecanismos eficientes

[7] **Lambert:** es un modelo de iluminación local que determina que la iluminación producida por una fuente luminosa sobre una superficie es directamente proporcional a la intensidad de la fuente y al coseno del ángulo que forma la normal a la superficie con la dirección de los rayos de luz, y es inversamente proporcional al cuadrado de la distancia a dicha fuente.

en tiempo de ejecución para determinar qué extensiones de OpenGL son compatibles con la plataforma de destino. La versión utilizada en el desarrollo fue GLEW 7.0 **[15]**.

6.2 METODOLOGÍA DE DESARROLLO

La metodología en la que se basa el desarrollo del trabajo especial de grado consiste en el proceso unificado ágil (*agile unified process*), el cual es una versión simplificada del proceso unificado racional (RUP) (*ver **Fig. 1.2***). Éste describe un enfoque simple y fácil de entender para desarrollar aplicaciones de negocio utilizando técnicas ágiles. La naturaleza del proceso unificado ágil consiste en cuatro fases:

- **Inicio**: Consiste en identificar el ambiente inicial del proyecto, una arquitectura potencial para el sistema y obtener la aceptación del mismo por parte del cliente.
- **Elaboración**: Consiste en realizar pruebas a la arquitectura del sistema para verificar si es la más adecuada al desarrollo del proyecto.
- **Construcción**: Consiste en desarrollar un *software* funcional en una base incremental que captura las necesidades más prioritarias del proyecto.
- **Transición**: Consiste en validar y publicar la aplicación final en un entorno de producción.

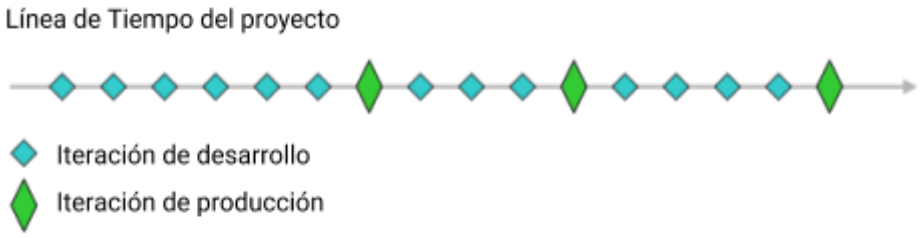

Figura 1.2: *Proceso unificado ágil*. El proceso es denotado en dos tipos de iteraciones. Una iteración de desarrollo se basa en el despliegue a un entorno de pruebas y de control de calidad. Una iteración de producción se basa en el despliegue a un entorno de producción.

La aplicación desarrollada en el trabajo especial de grado está basada en los trabajos de Dal Corso, Frisvad, Mosegaard y Bærentze **[2]** para la técnica de dispersión de la luz en modelos mallados, y en los trabajos de Ropinski, Döring y Rezk-Salama **[3]** para la técnica de dispersión de la luz en volúmenes de datos. Este trabajo está organizado de la siguiente manera: En el Capítulo I se presenta la propuesta de Trabajo Especial de Grado, donde se desarrolla el planteamiento del problema, los objetivos alcanzados y la metodología de trabajo utilizada en el desarrollo de la aplicación. En el Capítulo II se describen las características de los volúmenes de datos manejados en diversas ramas de la ciencia, así como los algoritmos más comunes para su despliegue. Además, se describen las principales representaciones de superficies en mallados. A su vez, se explica la técnica de la dispersión de la luz (*scattering*), donde se indican las interacciones de objetos con la luz del entorno y métodos de despliegue. El Capítulo III abarca el proceso de desarrollo a la metodología ágil AUP, describiendo cada una de las etapas realizadas durante la implementación. En el Capítulo IV se detallan los resultados obtenidos luego de finalizar la implementación del trabajo especial de grado, describiendo todas las funcionalidades de la aplicación. Posteriormente, se describen las conclusiones obtenidas, y finalmente, se proponen recomendaciones para futuras implementaciones.

CAPÍTULO II – CONCEPTOS BÁSICOS

En el presente capítulo, se introducen los conceptos y teoría básica necesaria para la comprensión de los capítulos posteriores. Esta teoría incluye la visualización y obtención de volúmenes de datos, así como la representación de las superficies y sus propiedades.

1 VISUALIZACIÓN DE VOLÚMENES DE DATOS

Un volumen se define como un conjunto de datos ubicados en R^3, los cuales por lo general son definidos como un grupo de muestras de una función escalar continua, representada por una malla regular y almacenada en un arreglo tridimensional de escalares **[16]**. El proceso de visualización de volúmenes de datos comprende a la serie de pasos llevados a cabo para proyectar un volumen de datos hacia un plano imagen bidimensional, con el propósito de entender la estructura del mismo **[17]**.

La visualización de volúmenes de datos utiliza técnicas de gráficos por computadora para ayudar a los científicos a comprender sus datos **[18] [19]**. La visión generalmente se logra utilizando las imágenes obtenidas para adquirir información y conocimiento de los datos producidos por experimentos para compartir experiencias a instituciones que brindan apoyo a investigaciones y al público en general. Para poder alcanzar este objetivo, las técnicas de visualización deben ofrecer una representación entendible de los datos, y a su vez, una rápida manipulación y despliegue de los mismos que permita una interacción fluida con el usuario.

En la actualidad, la visualización de volúmenes se utiliza ampliamente en la medicina, astrofísica, química, microscopía, ingeniería mecánica, pruebas no destructivas y otras áreas de la ciencia y la ingeniería. Entre los datos que los científicos e ingenieros almacenan como volúmenes se encuentran densidad, presión, temperatura, carga electrostática, calor, velocidad, entre otros. Se debe destacar que los datos almacenados tienen características muy diferentes, por lo cual algunos métodos para la visualización de volúmenes proporcionan buenos resultados para ciertos tipos de datos pero no para otros **[17]**.

1.1 FUENTES DE LOS VOLÚMENES DE DATOS

Los conjuntos de volúmenes de datos a menudo se adquieren a través del muestreo del material de interés mediante imágenes por resonancia magnética (*RMI*), tomografías computarizadas (*CT*) (*ver **Fig. 2.1***), tomografías de emisión de positrones (*PET*) y/o máquinas de sonografía, entre otras. El escaneo con láser confocal y otros microscopios de alta potencia también se usan para adquirir dichos datos.

Los datos volumétricos también pueden ser generados mediante la voxelización de objetos geométricos, el uso de herramientas de edición de volúmenes o de programas para la generación de volúmenes mediante métodos estocásticos. No obstante, todos los conjuntos de datos pueden tratarse de manera similar, incluso aunque se generen por diversos medios **[17]**.

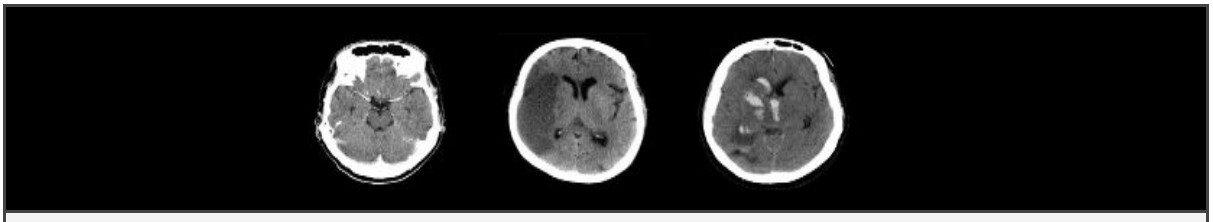

Figura 2.1: *Tomografía computarizada*. Visualización de diferentes cortes pertenecientes a una cabeza humana a partir de una tomografía computarizada.

1.2 CARACTERÍSTICAS DE LOS VOLÚMENES DE DATOS

Los volúmenes de datos generalmente se tratan como un conjunto de elementos volumétricos (*vóxeles*[8]) o una matriz de celdas. Estos dos enfoques provienen de la necesidad de volver a muestrear el volumen entre los puntos de conexión durante el proceso de *rendering*. El remuestreo, que requiere interpolación, ocurre en casi todos los algoritmos de visualización de volumen.

El enfoque de vóxeles indica que el área alrededor de un punto de cuadrícula tiene el mismo valor que el punto de cuadrícula. El enfoque de vóxeles tiene la ventaja de que no se hacen suposiciones sobre el comportamiento de los datos entre puntos de cuadrícula, es decir, sólo se usan valores de datos conocidos para generar una imagen (*ver* ***Fig. 2.2***).

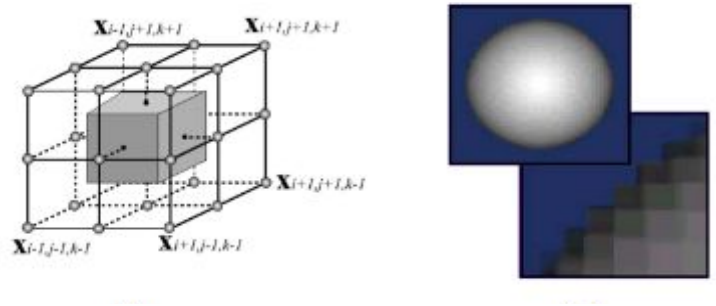

Figura 2.2: *Enfoque de vóxeles*. Representación de un vóxel, donde es definido como el área hexaédrica de valor no variable que rodea al punto central de una cuadrícula. Se puede observar una imagen resultante generada a través del método mencionado.

El enfoque de celdas representa a un volumen como una colección de hexaedros (*cubos*) cuyas esquinas son puntos de red y su valor varía entre los puntos de la cuadrícula. Esta técnica intenta estimar los valores dentro de la celda mediante la interpolación de los valores en las esquinas de la misma. Las funciones de interpolación más utilizadas son trilinear y tri-cúbica. Las imágenes generadas con este enfoque parecen más suaves que aquellas generadas con el enfoque de vóxeles (*ver* ***Fig. 2.3***) **[17]**.

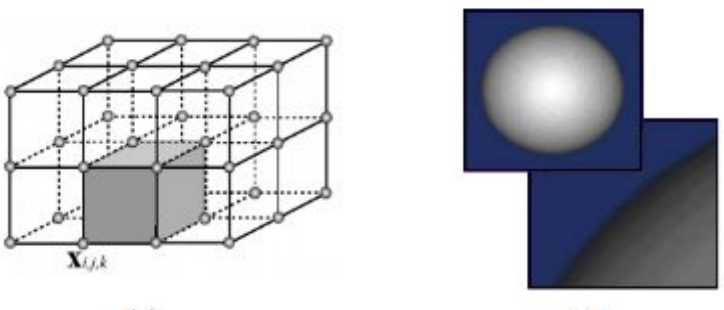

Figura 2.3: *Enfoque de celdas*. Representación de una celda, donde es definida como una colección de hexaedros de valor variable, donde mediante la interpolación entre los valores en las esquinas se obtiene el valor real. Se puede observar una imagen resultante generada a través del método mencionado, donde se nota la diferencia de suavidad con respecto al método anterior.

[8]**Voxel**: Es la unidad básica de volumen y hace referencia a un píxel volumétrico, es decir, un píxel en tres dimensiones.

1.3 ECUACIÓN DE DESPLIEGUE DE VOLÚMENES

La ecuación básicamente pretende lograr un modelo físico para el despliegue de volúmenes, donde cada volumen es representado a través de una integral que no tiene anti-derivada para funciones lineales a trozos o de orden superior. A continuación se detalla cómo la misma es aproximada para su posterior evaluación.

1.3.1 MODELO ÓPTICO

Para visualizar el volumen se requiere de un modelo óptico para representar la interacción de la luz con el volumen, y determinar la cantidad de luz alcanzada en cada píxel de la imagen. La modelación de esta interacción es relativamente compleja, y requiere el uso de la *Teoría del transporte radiativo*[9] **[20][21]**. Para derivar el modelo óptico utilizado, es necesario primero comprender y relacionar algunos conceptos asociados con el fenómeno del transporte de luz como lo son: dispersión, absorción, emisión, reflexión, refracción y reducción. Gran parte de la luz que llega a nuestros ojos es luz indirecta, proveniente del reflejo de la luz en objetos. La energía absorbida se transforma en otra energía; el resto de ella es dispersada (*scattered*), y parte de esta llega hasta nuestros ojos. El fenómeno de dispersión o *scattering* consiste en el desvío de la luz causado por partículas encontradas en su trayectoria **[7]**.

Si se consideran los casos donde las partículas son más grandes que la longitud de onda de la luz, el estudio del *scattering* de la luz puede ser comprendido a través de los conceptos de óptica geométrica **[22]**, y puede utilizarse *ray casting* o *ray tracing* para obtener resultados numéricos aceptables. El *scattering* puede ser estudiado mediante los fenómenos de reflexión y refracción. Cuando la luz colisiona con una partícula, parte de la luz se refleja, otra se refracta, y otra se absorbe. La reflexión cambia la dirección de la luz, pero la devuelve al medio de donde provenía; la refracción también cambia la dirección de la luz, pero en este caso la luz viaja dentro de la partícula. El ángulo de refracción depende de las densidades de ambos medios, y puede calcularse por la Ley de Snell **[23]**. El rayo refractado viaja dentro de la partícula, hasta volver a colisionar con la superficie de la misma, en donde el rayo puede volver al medio original por refracción, o puede reflejarse, en cuyo caso sigue su trayectoria de vuelta al interior de partícula (*reflexión interna*). Este proceso continúa, y si se considera a la partícula como una caja negra colisionada por un rayo de luz, se puede notar que la luz se dispersa (*scattering*) al incidir en la misma (*ver **Fig. 2.4***) **[7]**.

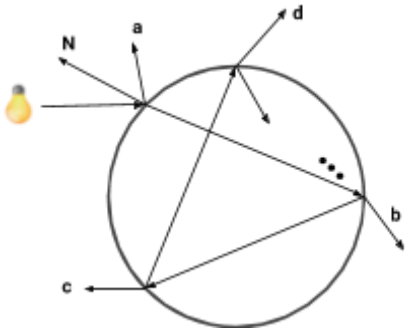

Figura 2.4: *Scattering*. Considerando que las partículas son más grandes que la longitud de onda de la luz. Los rayos resultantes de la interacción de la luz con la partícula mostrada en la figura son producto de: **(a)** Reflexión externa, **(b)** Dos refracciones, **(c)** Reflexión interna, **(d)** Dos refracciones internas.

[9]**Transferencia Radiativa**: el término de transferencia radiactiva se refiere al fenómeno físico de la transferencia de energía en la forma de radiación electromagnética. La propagación de la radiación a través de un medio es afectado por los procesos de absorción, emisión y *scattering*.

Tanto el *scattering* como la absorción reducen la energía del rayo de luz, que atraviesa un medio o un volumen colmado de partículas. Esta atenuación de la luz se denomina extinción **[22]**.

Para modelar la interacción de la luz con el volumen, también se pueden considerar partículas infinitesimalmente pequeñas, estudiando la interacción de la luz con la nube a nivel atómico o molecular. En este caso, el *scattering* puede verse como la dispersión de la luz en todas las direcciones, cuando esta colisiona con la partícula. Uno de los primeros modelos ópticos en esta área describe un método para sintetizar imágenes de los anillos del planeta Saturno, los cuales consisten de nubes de partículas de hielo reflectivo **[24]**. Este modelo considera el *scattering* simple, *shadowing* [10] y propagación de la luz a través de la nube. El *scattering* múltiple es posteriormente considerado por, Kajiya y Von Herzen **[25]**. Estas interacciones de la luz con volumen pueden resumirse en la ***Fig. 2.5*** **[7]**.

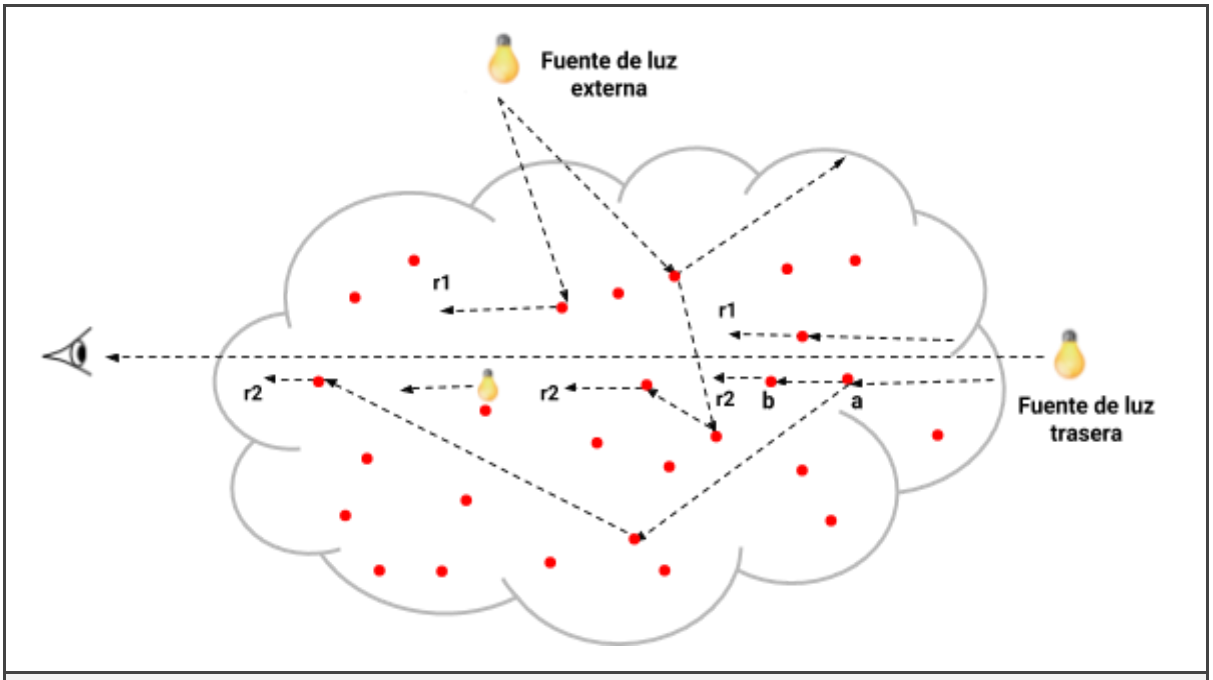

Figura 2.5: *Interacciones de la luz con el volumen*. En el modelo se asume que hay una fuente de luz trasera que emite energía en dirección al visor. Un rayo de luz que viaja hacia el visor puede ser atenuado (por absorción y *scattering*) o incluso obstruido totalmente por partículas encontradas en la travesía del rayo, según el modelo óptico utilizado. Dentro de la nube pueden existir fuentes de luz, que suelen ser las mismas partículas emanando energía en todas las direcciones, y particularmente en la dirección al visor. También pueden existir una o más fuentes de luz externas. Mediante el *scattering*, la luz puede llegar al visor de manera indirecta, a través de la interacción de un rayo de luz con una única partícula (*scattering* simple, ver rayos r_1), o con más de una partícula (*scattering* múltiple, ver rayos r_2), ya sea considerando la fuente de luz trasera o la externa. Adicionalmente, una partícula a puede producir *shadowing* a otra partícula b, al interponerse en la trayectoria de la luz.

[10] **Shadowing**: opacamiento o sombra. En el contexto de modelos ópticos, significa la atenuación de la luz por parte de una partícula previa a la partícula en estudio.

1.4 PASOS GENERALES PARA LA VISUALIZACIÓN DE VOLÚMENES

A pesar de la variedad de algoritmos existentes para la visualización de volúmenes, la mayoría de los pasos involucrados son comunes entre ellos. Los algoritmos más comunes para la visualización son el despliegue directo de volúmenes o *direct volume rendering*[11] (*DVR*) y algoritmos de extracción de isosuperficies o *surface fitting*[12] (*SF*). Generalmente, los algoritmos difieren en la implementación de cada uno de los pasos, los cuales se explican a continuación.

1.4.1 ADQUISICIÓN DE LOS DATOS

El primer paso en cualquier procedimiento para la visualización de volúmenes consiste en la obtención y preprocesamiento de los datos, de forma que se puedan obtener mejores resultados visuales. Este preprocesamiento consiste en modificar los valores para que se logre una buena distribución, tengan altos contrastes, estén libres de ruido y dentro de un rango **[26]**.

Así, en algunos casos es necesario que el conjunto de datos sea reconstruido para que éstos tengan la misma proporción espacial que el material de interés, de manera que las imágenes desplegadas no estén deformadas a la hora de la visualización **[27]**. Cuando la proporción del material y de los datos no es la misma, puede ser necesario interpolar dos cortes para obtener uno nuevo, interpolar muestras para obtener datos faltantes, o convertir un mallado irregular a un mallado uniforme **[26]**.

1.4.2 CLASIFICACIÓN DE LOS DATOS

Consiste en elegir la forma en que los datos deben ser desplegados en base a sus valores. Este paso es llevado a cabo por el usuario, y el procedimiento a realizar depende del algoritmo utilizado para visualizar los datos. Si el algoritmo está basado en *SF*, el usuario debe elegir el umbral a extraer, el cual consiste en un valor real que representa el valor a ajustar por la *isosuperficie*[13] extraída (*ver **Fig. 2.6***). Cuando el algoritmo está basado en *DVR*, el usuario debe configurar la función de transferencia **[26]**.

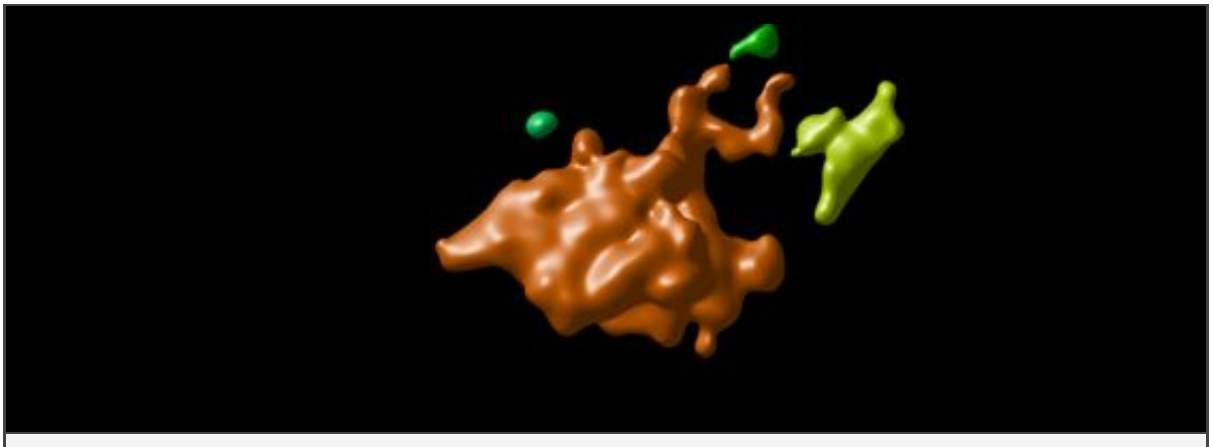

Figura 2.6: *Isosuperficie*. Imagen de una isosuperficie en un volumen microscópico con un umbral definido. Los objetos no conectados están representados por un color diferente.

[11] ***DVR***: se encarga de obtener una representación 3D del volumen de datos de manera directa, se considera que los datos representan un medio emisor de luz semitransparente. Por lo tanto, también se pueden simular fenómenos gaseosos.
[12] ***SF***: consiste en obtener un mallado poligonal del volumen de datos por medio de un umbral, siguiendo una serie de reglas.
[13] **Isosuperficie**: es una superficie que representa puntos de un valor constante dentro de un volumen. Estos valores pueden representar presión, temperatura, velocidad, densidad, entre otros.

(a) FUNCIÓN DE TRANSFERENCIA

Es una función que relaciona un color y una opacidad a cada valor posible que puedan tener los atributos del volumen de forma que el algoritmo muestre los datos que le interesen. Normalmente el *dataset* no contiene la intensidad de luz por vóxel. Sólo incluyen escalares que están representados comúnmente en 8 o 16 bits. Al desplegar un volumen, el usuario decide qué materiales o estructuras visualizar. Para ello, se le asignan las propiedades ópticas de emisión $c(s)$ y absorción $t(s)$ a cada vóxel del volumen.

Una forma de aplicar una clasificación a un volumen es utilizando funciones, de manera que al ser manipuladas se logre observar las diferentes capas que lo componen. En el área de visualización de volúmenes se les denomina función de transferencia, que pueden ser unidimensionales o multidimensionales (*ver* ***Fig. 2.7***) **[28]**.

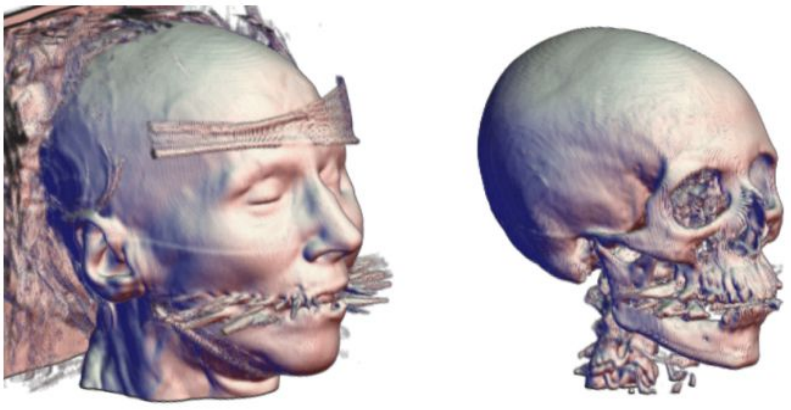

Figura 2.7 [28]: *Materiales de volúmenes*. Mediante dos funciones de transferencia distintas aplicadas al mismo volumen (*CT head*), se puede observar las distintos materiales que lo componen.

Como se puede notar en la ***Fig. 2.7***, dependiendo de la función de transferencia el volumen se va a poder observar de una manera diferente; en la ***Fig. 2.8*** se encuentran las funciones de transferencia aplicadas al volumen de la figura anterior.

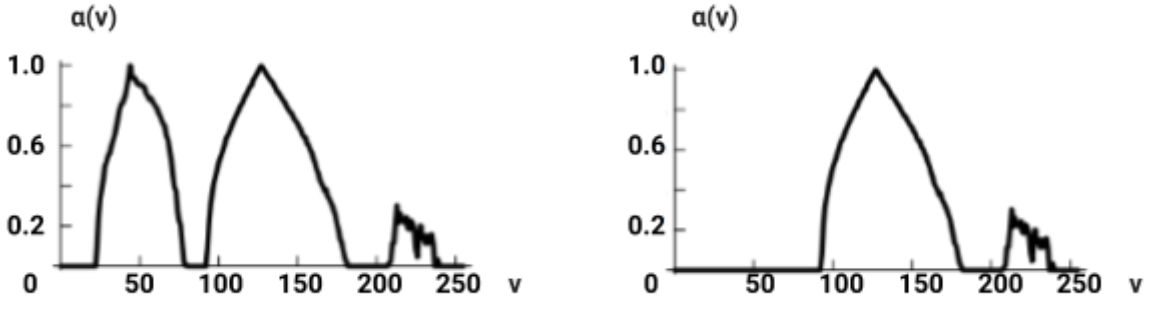

Figura 2.8: *Función de transferencia*. Dependiendo de la opacidad en la función de transferencia el volumen puede observarse de manera diferente. En este caso nótese que para poder visualizar los huesos del volumen de la ***Fig. 2.7*** se aplicó una función con los primeros valores con la opacidad en 0.

La clasificación del volumen puede ser realizada de tres maneras. La diferencia entre cada una radica más que todo en el orden en que se realiza el proceso de clasificación e interpolación de las muestras.

- **Pre-Clasificación**: este método aplica la función de transferencia antes del proceso de interpolación, es decir, todos los valores de los vóxeles del volumen van a ser sustituidos por el color y opacidad de la función de transferencia antes de aplicar la interpolación (*ver **Fig. 2.9***) **[29]**.

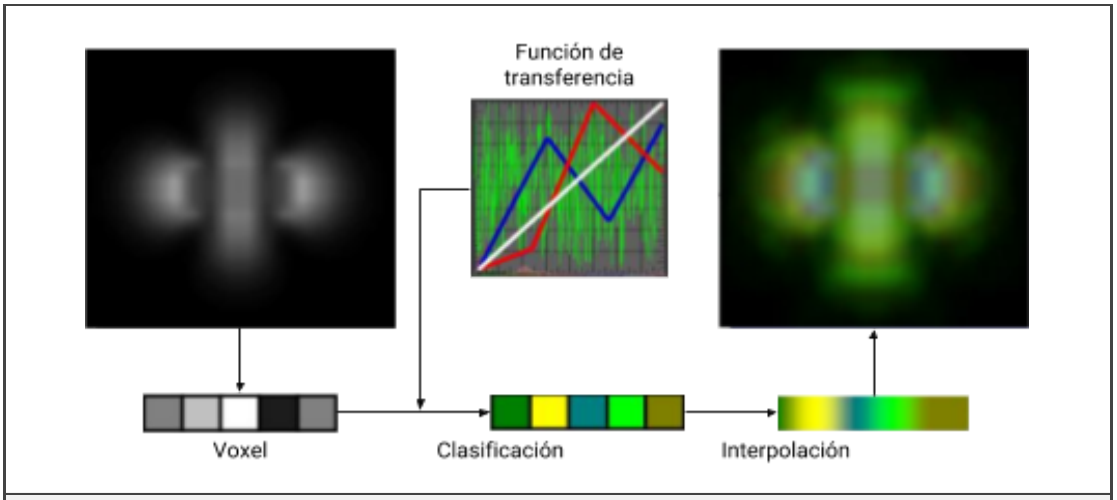

Figura 2.9: *Pre-Clasificación*. El primer paso sería realizar la clasificación a partir de la función de transferencia utilizando las muestras del volumen, para luego interpolar dichos valores y lograr el resultado esperado.

- **Post-Clasificación**: es el proceso de aplicar la función de transferencia luego de haberse realizado el proceso de interpolación de las muestras del volumen, para que luego estas sean sustituidas por los valores de emisión y absorción de la función de transferencia (*ver **Fig. 2.10***) **[29]**.

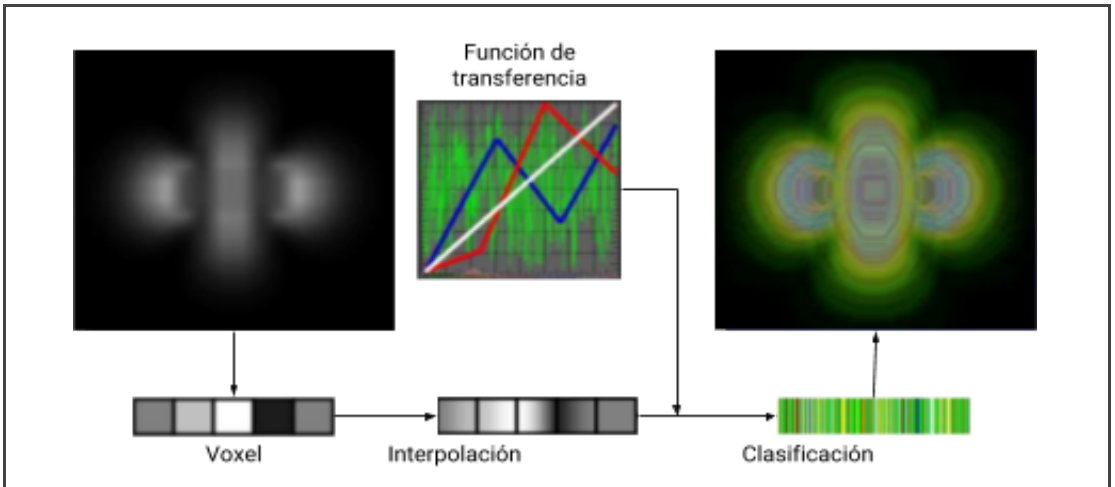

Figura 2.10: *Post-Clasificación*. Se aplica la interpolación de las muestras del volumen, luego se procede a la sustitución de estos por los valores de emisión y absorción de la función de transferencia respectiva.

Los procesos de pre-clasificación y post-clasificación producen resultados diferentes si las funciones de transferencia no son constantes o la identidad **[29]**.

- **Clasificación Pre-Integrada**: Acorde al teorema de muestreo, una reconstrucción correcta de un campo escalar se logra si esta es muestreada a una tasa superior o igual a la frecuencia de *Nyquist*[14]. Sin embargo, al utilizar post-clasificación, primero se interpolan las muestras escalares antes de aplicar la función de transferencia, donde se sabe que dicha función puede estar definida a trozos, por lo tanto se requiere una tasa de muestreo elevada para capturar todos los detalles. Normalmente este factor no se tiene en cuenta ya que el cómputo requerido sería muy elevado. Si se considera una función de transferencia definida por un pulso delgado, si el espaciado entre las muestras del rayo es superior a la longitud de este pulso, algunas muestras interpoladas pueden capturar este detalle, mientras otras no, resultando en una imagen compuesta de bandas y puntos, y no una superficie continua **[7]**. Estos artefactos visuales pueden ser reducidos si se aumenta el muestreo a una tasa muy alta (reducir la distancia entre muestras).

Si se utiliza una clasificación pre-integrada, en lugar de clasificar cada muestra individualmente, esta se realiza por segmentos definidos entre cada par de muestras h_i y h_{i+1}. Para esto se calcula previamente la integral de la función de transferencia entre cada par de posibles muestras. Este cálculo puede ser almacenado en una textura 2D, donde la coordenada (x, y) representaría la muestra (s_f, s_b), es decir, el segmento que comienza en s_f y termina en s_b, y el valor almacenado sería el color y opacidad obtenido de la integral (*ver **Fig. 2.11***). Si se realiza un muestreo adaptativo se requerirá de una textura 3D para almacenar la tabla de pre-integración.

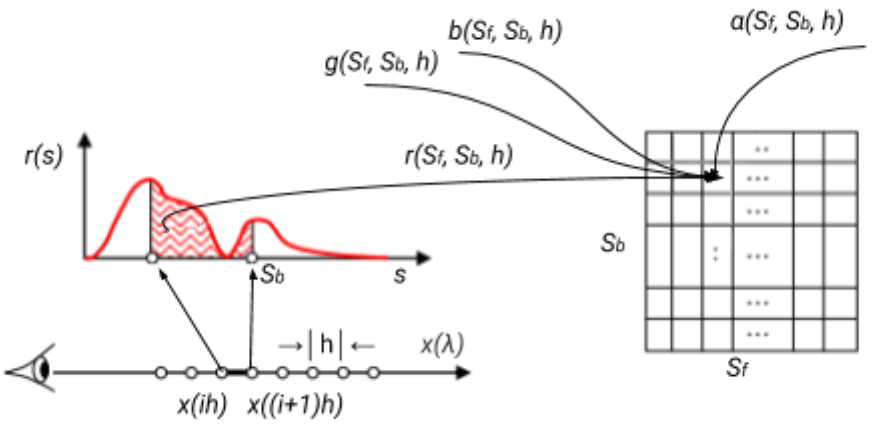

Figura 2.11: *Clasificación Pre-integrada*. Construcción de la tabla 2D de integrales. Se precalcula la integral entre pares de muestras (s_f, s_b), por cada canal de color $r(s)$, $g(s)$ y $b(s)$, y opacidad $\alpha(s)$. En la gráfica se muestra el segmento considerado para el canal $r(s)$, donde $S_f = s(x(ih))$ y $S_b = s(x((i+1)h))$.

La pre-integración tiene sus orígenes en el *rendering* de volúmenes y superficies sin realizar reconstrucción poligonal a partir de mallas tetraédricas. Seguidamente fue utilizada para visualizar volúmenes representados en mallas regulares o cuadrículas **[29]**.

[14]**Frecuencia de *Nyquist***: consiste en la frecuencia mínima capaz de muestrear una señal analógica de manera precisa. Ésta debe ser al menos el doble de la frecuencia máxima de la señal que se desea muestrear.

La clasificación de los datos es uno de los pasos más difíciles que debe llevar a cabo el usuario, porque es necesario que éste tenga experiencia en este proceso y que el sistema proporcione una respuesta rápida, debido a que este procedimiento está basado en intento y error. Una solución para dar respuesta rápida al usuario consiste en ofrecer una vista con menor resolución mientras se configura la clasificación, y generar una imagen refinada después que el usuario confirma que ha finalizado.

1.4.3 RECORRIDO DE LOS DATOS

Después de configurar la clasificación del volumen de datos, se deben generar las imágenes recorriendo los datos. Existen dos formas de recorrer el volumen: en orden de objeto (*object-order*), que consiste en calcular la contribución de cada elemento del volumen a los píxeles de la imagen; o en orden de imagen (*image-order*), que consiste en determinar el color en cada píxel de la imagen, buscando los elementos del volumen que contribuyen por cada uno de estos (*ver **Fig. 2.12***).

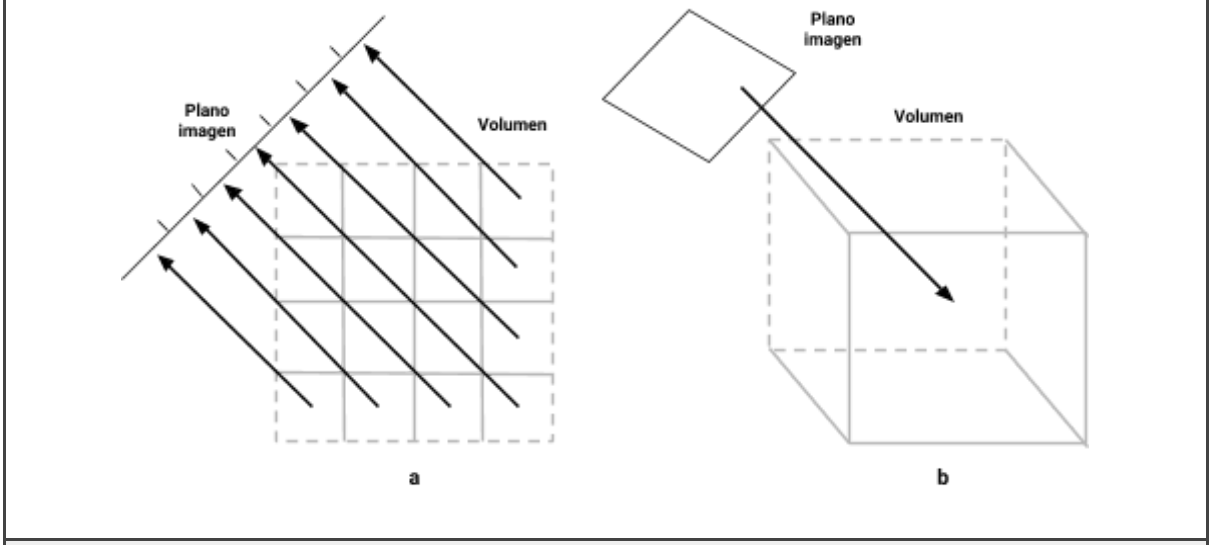

Figura 2.12: *Recorridos en orden de imagen y en orden de objeto*. **(a)** Recorrido en orden de objeto, los rayos parten del volumen hacia el plano de vista. **(b)** Recorrido en orden de imagen, los rayos de vista parten de un pixel en la imagen hacia el volumen.

Los recorridos en orden de objeto pueden ir de adelante hacia atrás (*front-to-back*) o de atrás hacia adelante (*back-to-front*). Recorrer el modelo de adelante hacia atrás tiene la ventaja de que los elementos en la parte de atrás no deben ser visitados si los de adelante ya han creado una imagen lo suficientemente opaca. La ventaja del recorrido de atrás hacia delante es que permite el uso de las operaciones de mezcla o *blending* provistos por el *hardware*.

El algoritmo en orden de imagen generalmente proceden de arriba hacia abajo, de izquierda a derecha. También se pueden calcular los píxeles de forma aleatoria, de forma que el usuario observe como la imagen es refinada mientras los pixeles faltantes son calculados **[26]**.

1.4.4 VISUALIZACIÓN Y SOMBREADO

Para visualizar un volumen de datos se puede utilizar tanto proyección ortogonal como proyección perspectiva, sin importar si el algoritmo está basado en *DVR* o en *SF*. Sin embargo, el uso de proyección ortogonal asegura que el usuario no se confunda al observar los datos deformados por la transformación perspectiva. No obstante, cuando no se usa perspectiva, se deben incluir otras

características que le permitan al usuario percibir la profundidad de los elementos, tal como niebla por profundidad, atenuación por distancia o estereoscopía **[17]**.

Para realizar el sombreado en los algoritmos basados en *DVR* y en *SF* generalmente se usa sombreado por gradiente (*gradient shading*), el cual consiste en utilizar el gradiente normalizado de los datos como vector normal en un modelo de iluminación local, como *Phong* o *Blinn-Phong*. Para calcular los gradientes dentro de una celda, se utiliza una interpolación donde el gradiente de un punto se calcula por diferencias finitas entre los puntos adyacentes en cada dirección (*ver* ***Fig. 2.13***).

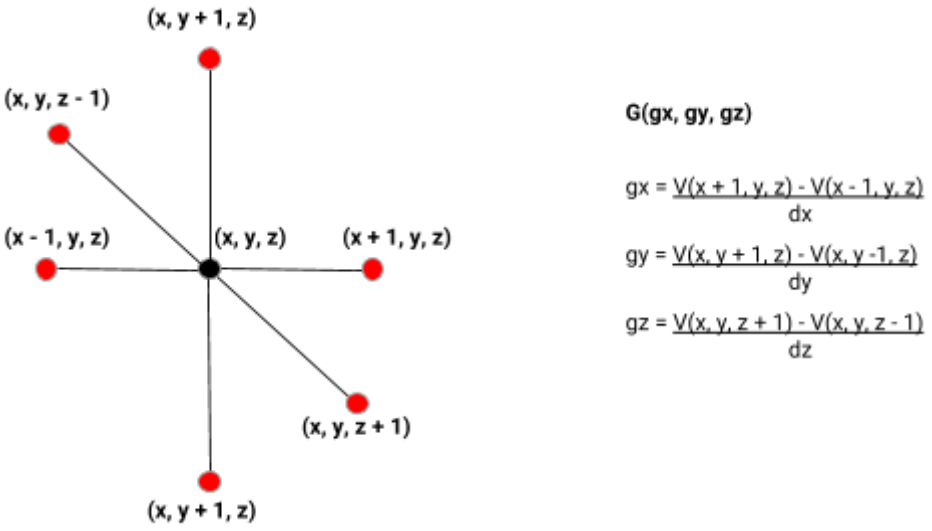

Figura 2.13: *Cálculo del gradiente*. Como se puede observar, se determina la normal en cada vértice vía diferencias centrales, realizando los cálculos con las fórmulas planteadas.

1.5 MÉTODOS DE VISUALIZACIÓN DE VOLÚMENES DE DATOS

Los algoritmos fundamentales de visualización de volúmenes se dividen en dos categorías, algoritmos de despliegue directo de volúmenes o *direct volume rendering* (*DVR*) y algoritmos de extracción de isosuperficies o *surface fitting* (*SF*).

Aquellos basados en *DVR* incluyen el *ray casting*, métodos de pre integración, *splatting* y despliegue de *V-Buffer*, y están caracterizados por aplicar elementos directamente en la imagen sin utilizar primitivas geométricas como representación intermedia **[17]**. Estos métodos son apropiados para crear imágenes a partir de volúmenes de datos que contengan características amorfas como nubes, fluidos y gases.

El despliegue directo de volúmenes es una simulación aproximada de la propagación de la luz a través de un medio participante representado por el volumen **[30]**. Con esta técnica, se produce una imagen mediante el cómputo de la cantidad de luz alcanzada en cada píxel. Físicamente, cuando la luz fluye a través del volumen, ésta puede ser absorbida, dispersada (*scattering*) y emitida; adicionalmente, se pueden producir otros tipos de interacción tales como fosforescencia (absorción y reemisión de energía luego de un pequeño retardo), y fluorescencia (absorción y reemisión de la energía en una frecuencia distinta). Sin embargo, la representación de volúmenes no tiene como objetivo simular todos estos fenómenos. Sólo debe importar el resultado óptico del proceso de propagación y avance de la luz a través del volumen, quedando así un modelo óptico basado únicamente en emisión y absorción **[30] [31] [32]**.

Los algoritmos basados en *SF* usualmente ajustan primitivas como polígonos o parches a superficies de contorno con valor constante en volúmenes de datos. El primer paso consiste en la elección de un umbral por parte del usuario, el cual es usado para ajustar primitivas geométricas a los contornos en el volumen que sean iguales al umbral. Este enfoque incluye algoritmos como conexión de contornos (*contour connecting*), cubos marchantes (*marching cubes*), tetraedros marchantes (*marching tetrahedra*), entre otros.

1.5.1 RAY CASTING

El algoritmo de *ray casting* está basado en *DVR*, lleva a cabo un recorrido en orden de imagen, donde el color y opacidad de cada píxel se calcula emitiendo un rayo desde el píxel hacia el volumen de datos, acumulando las opacidades y colores encontrados durante la trayectoria del mismo (*ver **Fig. 2.14***) **[6][33][34]**. En general, el rayo atraviesa el volumen en una dirección arbitraria, y las muestras requeridas durante la travesía del rayo no coinciden con las muestras originales del volumen. Por lo tanto, se suele utilizar un filtro para el remuestreo. Comúnmente se emplea un filtro tri-lineal entre las 8 muestras más cercanas a la muestra a reconstruir. Sin embargo pueden aplicarse filtros más sofisticados que involucran más cantidad de muestras, como filtros gaussianos y bi-cúbicos **[35]**.

Por lo general, la implementación del *ray casting* es realizada por *software*. Dado que el algoritmo requiere de mucho cómputo, y aunado a problemas de localidad espacial, se dificulta la respuesta en tiempo real. El despliegue se puede acelerar mediante la utilización de las técnicas de terminación temprana del rayo y salto de espacios vacíos **[34] [36]**. La terminación temprana del rayo consiste en truncar la travesía del rayo cuando se acumule un umbral de opacidad definido por el usuario (comúnmente de 0.95 a 0.99). El resto del rayo puede ignorarse, puesto que el aporte de las muestras remanentes es insignificante en el color final del píxel.

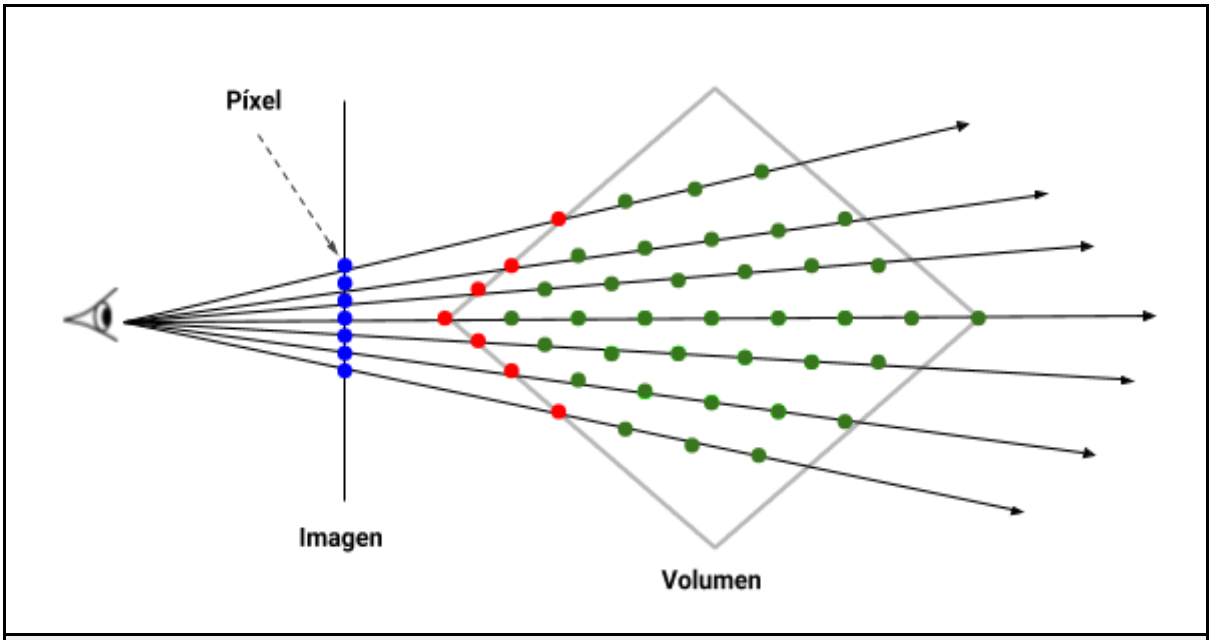

Figura 2.14: *Ray Casting*. Por cada centro de píxel de la imagen se lanza un rayo. Aquellos rayos que atraviesan el volumen acumulan un color y opacidad. El paso entre muestras es constante. Los puntos rojos son los puntos de entrada de cada rayo en el volumen.

Este tipo de algoritmos recorre la imagen en el mismo orden en que los píxeles están almacenados en memoria principal (*image-order*) **[30]**. Sin embargo, el problema principal es que no se accede al volumen en el orden en que está almacenado, ya que los rayos de visualización lo

atraviesan en cualquier dirección. Como resultado, los algoritmos de *ray casting* ocupan mucho tiempo en cálculos de posición de las muestras, y no explotan la localidad espacial de datos volumétricos, limitando el rendimiento en computadores convencionales.

Hoy en día, con la flexibilidad de hacer programas que se ejecutan en los procesadores de fragmentos, el *ray casting* puede ser implementado con aceleración en hardware en tiempo real **[37]**. En este caso, en cada programa de fragmento se evalúa la ecuación de composición volumétrica para un píxel de la imagen. Para disparar los programas de fragmentos, se considera el volumen como un cubo (6 cuadriláteros). Rasterizando las caras visibles del cubo, e interpolando las coordenadas de los vértices de dichas caras, se disparan los programas de fragmentos con el punto de entrada respectivo. El punto de salida del rayo puede hallarse de manera análoga al punto de entrada, pero esta vez rasterizando las caras no visibles del cubo. El punto de salida también puede determinarse durante la travesía del rayo, cuando el punto del rayo a considerar se encuentre fuera del volumen.

Además de la técnica de *ray casting* existen otros métodos de visualización de volúmenes, tales como *splatting* **[38][26]**, conchas esféricas **[40][39]**, planos alineados al plano de visión **[7][39]**, *shear-warp* **[41]**, entre otros.

Como ya se mencionó anteriormente, se han implementado diversas técnicas para desplegar modelos mallados realizando una triangulación de sus vértices para finalmente mostrar en pantalla una superficie. En la siguiente sección se explicarán en detalle las diferentes maneras de representar superficies poligonales.

2 REPRESENTACIÓN DE SUPERFICIES

El procesamiento eficiente de objetos geométricos, requiere el diseño de estructuras de datos adecuadas. Para cada problema específico en el procesamiento de la geometría se puede identificar un conjunto característico de operaciones donde el cálculo es sencillo y por lo tanto, se debe elegir una estructura de datos apropiada que respalde la implementación eficiente de estos operadores. Desde un punto de vista de alto nivel, existen dos clases principales de representaciones de superficies: representaciones paramétricas y representaciones implícitas **[42]**.

2.1 DEFINICIÓN DE SUPERFICIES Y PROPIEDADES

La definición común de una superficie en el contexto de aplicaciones gráficas es la de una variedad de puntos bidimensionales continuos en R^3. Las superficies paramétricas se definen mediante una función de parametrización con valores vectoriales $f: \Omega \rightarrow S$ que mapea un dominio de parámetros bidimensionales $\Omega \subset R^2$ a la superficie $S = f(\Omega) \subset R^3$. Por el contrario, una superficie implícita (volumétrica) se define como el conjunto cero de una función de valores escalares $F: R^3 \rightarrow R$, es decir, $S = \{x \in R^3 \mid F(x) = 0\}$ **[42]**.

2.2 REPRESENTACIÓN DE SUPERFICIES PARAMÉTRICAS

Este tipo de superficies tienen la ventaja que la función $f: \Omega \rightarrow S$ habilita la reducción de diversos problemas tridimensionales en la superficie S para transformarlos en problemas del dominio Ω. Los puntos de la misma pueden ser generados fácilmente por simples evaluaciones de f, las cuales van a permitir que se realicen operaciones de evaluación de manera eficiente. A su vez, las adyacencias de la superficie S pueden ser halladas fácilmente considerando sus puntos vecinos en el dominio Ω. Una composición simple de f con una función de deformación $d: R^3 \rightarrow R^3$ puede generar una modificación eficiente de la geometría de S.

Por otra parte, generar una superficie paramétrica con la parametrización f puede llegar a ser muy compleja, debido a que el dominio Ω debe corresponderse con la estructura métrica y topológica de S. Al momento de cambiar la forma de S, puede que sea necesario actualizar la función de parametrización con el fin de reflejar los cambios respectivos en la geometría subyacente. Una parametrización de baja distorsión requiere que las métricas en S y Ω sean similares y, por lo tanto, debemos evitar o adaptar el estiramiento excesivo.

2.2.1 SUPERFICIES SPLINE

Las superficies *spline* son curvas diferenciables definidas en porciones mediante polinomios. Estas son la representación estándar de los sistemas *CAD*[15] (*Computer-aided design*) actuales. Se utilizan para construir superficies de alta calidad desde cero y para las tareas posteriores de deformación de las mismas. Las superficies *spline* pueden describirse convenientemente por las funciones de base *B-spline*[16] $N_i^n(.)$ **[43] [44] [45]**. Un tipo de superficie *spline* que se comenzó a utilizar con éxito en los sistemas *CAD*, fue la curva de Bézier, la cual consiste en una línea curva entre dos o más puntos de control.

Debido a que las formas de una estructura topológica complicada deben representarse mediante superficies *spline*, el modelo debe descomponerse en una gran cantidad de parches de Bézier[17] (posiblemente recortados). Como consecuencia de estas restricciones, los modelos *CAD* típicos consisten en una gran colección de parches superficiales. Para representar una superficie de alta calidad, globalmente suave, estos parches deben estar conectados de una manera delicada, lo que genera restricciones geométricas adicionales que deben ser atendidas a lo largo de todas las fases de procesamiento de la superficie.

2.2.2 SUPERFICIES DE SUBDIVISIÓN

Las superficies de subdivisión **[46]** se pueden considerar como una generalización de las superficies *spline*, pero a su vez pueden representar superficies de topología arbitraria, ya que las superficies de subdivisión se generan mediante el refinamiento repetido de su mallado: después de cada paso de refinamiento topológico, las posiciones de los vértices (viejo y nuevo) se ajustan en función de un conjunto de reglas de promediado local. Un análisis cuidadoso de estas reglas revela que en el límite, este proceso da como resultado una superficie suaves (*ver **Fig. 2.15***).

Como consecuencia, las superficies de subdivisión no están restringidas por condiciones topológicas y geométricas como las superficies *spline*, y su estructura jerárquica inherente permite algoritmos altamente eficientes.

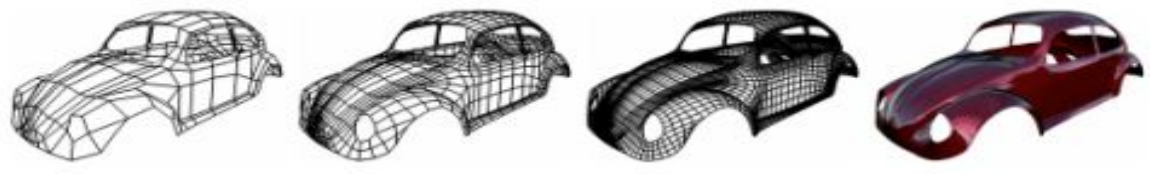

Figura 2.15 [42]: *Superficie de subdivisión*. Las superficies de subdivisión se generan mediante un refinamiento iterativo de una malla de control.

[15]**Sistema CAD:** conocido también como el diseño asistido por computadoras. Consiste en el uso de un amplio rango de herramientas computacionales que asisten a ingenieros, arquitectos y diseñadores.
[16]***B-spline**:* es una función spline que tiene el mínimo soporte con respecto a un determinado grado, suavidad y partición del dominio.
[17]**Parches de Bézier**: es la generalización de una curva de Bézier a su forma bidimensional. Son construidos a partir de dominios triangulares, usando coordenadas baricéntricas.

2.2.3 MALLAS TRIANGULARES

En muchos algoritmos de procesamiento de geometría, las mallas triangulares se consideran como una colección de triángulos sin ninguna estructura matemática particular. En principio, cada triángulo define, a través de su parametrización baricéntrica, un segmento de una superficie lineal a trozos **[42]**.

Cada punto p en el interior de un triángulo $[a, b, c]$ se puede escribir como una combinación baricéntrica de los puntos de las esquinas: $p = \alpha a + \beta b + \gamma c$ con $\alpha + \beta + \gamma = 1$. Al elegir un triángulo arbitrario $[u, v, w]$ en el dominio del parámetro, se puede definir un mapeo lineal $f: R^2 \rightarrow R^3$ con:

$$\alpha u + \beta v + \gamma w \Rightarrow \alpha a + \beta b + \gamma c \qquad \textbf{Ec. 2.1}$$

Una malla triangular M está formada por un componente geométrico y un componente topológico, donde este último se puede representar mediante una estructura gráfica a través de un conjunto de vértices $V = \{v_1, \ldots, v_V\}$ y un conjunto de caras triangulares que los conectan $F = \{f_1, \ldots, f_F\}$, $f_i \in V \times V \times V$. A veces es más eficiente representar la conectividad de una malla triangular en términos de las aristas del grafo respectivo $E = \{e_1, \ldots, e_E\}$, $e_i \in V \times V$, de modo que cada cara $f \in F$ representa un triángulo en el espacio. Aún así, si los encajes geométricos se definen asignando posiciones 3D a los vértices (discretos), la superficie poligonal resultante sigue siendo una superficie continua que consta de piezas triangulares con funciones de parametrización lineal (*ver **Ec. 2.1***).

Si una superficie es lo suficientemente suave, esta se aproxima a través de una función lineal por partes, tomando en cuenta que el error de aproximación es de orden $O(h^2)$, con h indicando la longitud máxima de la arista. Debido a esto, el error disminuye por un factor de ¼ al reducir a la mitad la longitud de las aristas. A medida que este refinamiento divide cada triángulo en cuatro sub-triángulos, aumenta el número de triángulos de F a $4F$ (*ver **Fig. 2.16***).

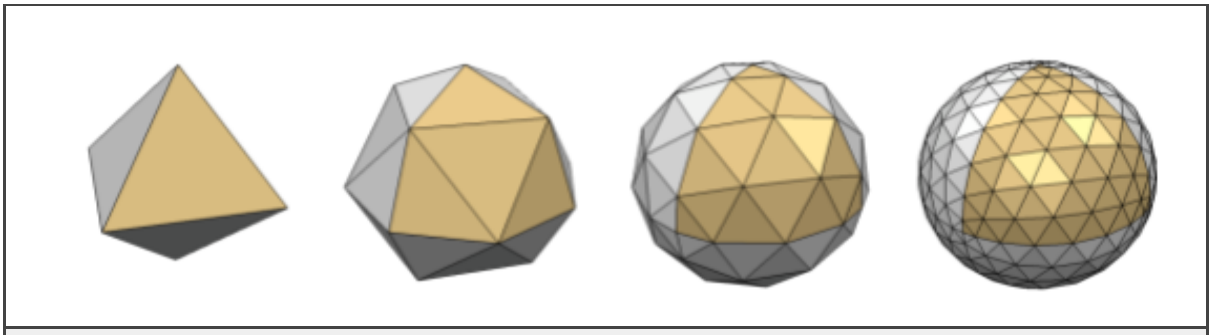

Figura 2.16 [42]: *Subdivisión de un prisma*.Cada paso de subdivisión divide a la mitad las longitudes de las aristas de cada triángulo, y por ende, aumenta el número de caras por un factor de 4 y reduce el error por un factor de ¼.

2.3 REPRESENTACIÓN DE SUPERFICIES IMPLÍCITAS

El concepto básico de representaciones implícitas o volumétricas de modelos geométricos consiste en caracterizar todo el espacio de un objeto, clasificando cada punto 3D que se encuentre dentro, fuera o exactamente en la superficie S.

Existen diferentes representaciones para funciones implícitas, como lo son las superficies algebraicas continuas, funciones de base radial o vóxelización discreta. En cualquier caso, la superficie S se define como la isosuperficie de nivel cero de una función escalar $F : R^3 \rightarrow R$. Por definición, si los valores que arroja la función F son negativos designan puntos dentro del objeto, y valores positivos representan puntos fuera del objeto, de modo que la isosuperficie de nivel cero separa el interior del exterior.

Como consecuencia, las consultas geométricas *dentro/fuera* simplifican las evaluaciones de la función F y verifican el signo del valor resultante. Esto hace que las representaciones implícitas sean adecuadas para la geometría sólida constructiva (*constructive solid geometry o CSG*), donde los objetos complejos se construyen mediante operaciones booleanas (*ver* ***Fig. 2.17***).

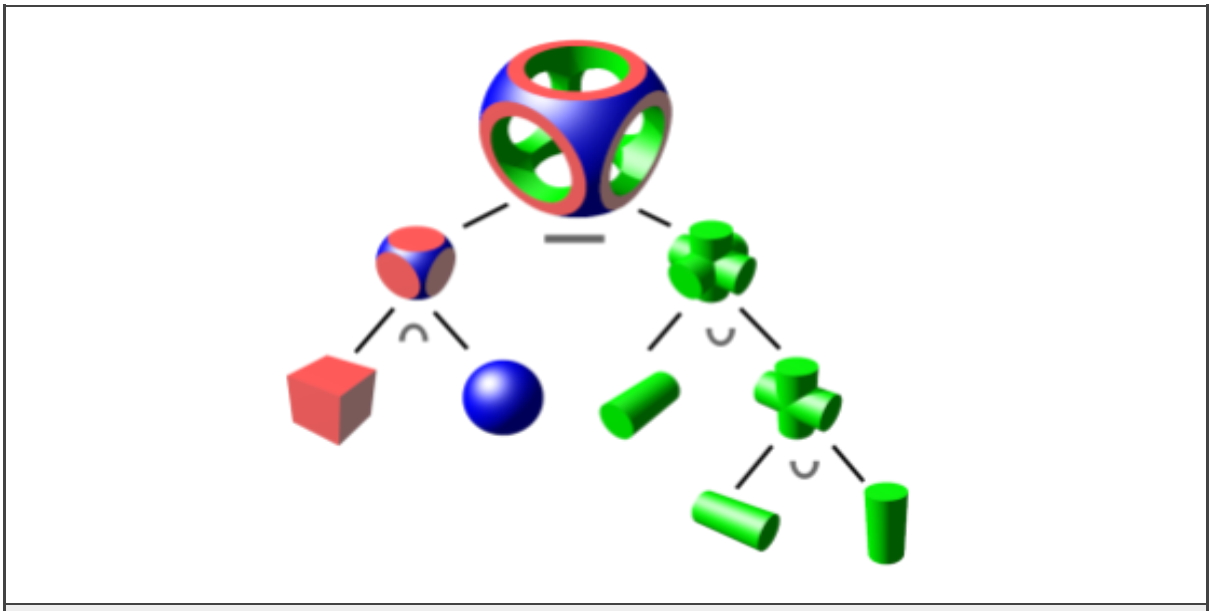

Figura 2.17 [42]: *Geometría sólida constructiva.* Un objeto complejo construido mediante operaciones booleanas, donde son utilizadas las operaciones de unión, intersección y diferencia.

La función implícita F para una superficie S no es determinada de manera única, pero la representación más común y más natural es la llamada función distancia con signo[18]. Además de las consultas *dentro/fuera*, esta representación también simplifica los cálculos de distancia para evaluaciones de funciones simples, que se pueden usar para calcular y controlar el error global para algoritmos de procesamiento de mallas.

2.3.1 CUADRÍCULAS REGULARES

Para procesar de manera eficiente las representaciones implícitas, el campo escalar continuo F se suele discretizar en un cuadro delimitador (*bounding box*) alrededor del objeto utilizando una cuadrícula suficientemente densa con nodos $g_{ijk} \in R^3$. Por lo tanto, la representación más básica es una cuadrícula escalar uniforme de los valores muestreados $F_{ijk} := F(g_{ijk})$, y los valores de la función dentro de los vóxeles se derivan por interpolación trilineal, proporcionando así un orden de aproximación cuadrática. Sin embargo, el consumo de memoria de esta estructura de datos crece de forma cúbica si se aumenta la precisión al reducir la longitud de las aristas de los vóxeles en la cuadrícula.

[18]**Función distancia con signo**: mapea la distancia más cercana de cada punto 3D en una superficie S.

2.3.2 ESTRUCTURAS DE DATOS AJUSTABLES

Para una mejor eficiencia en memoria con respecto a las cuadrículas regulares **[42]**, la densidad de muestreo de estas estructuras se adapta a menudo a la importancia geométrica local en el campo escalar F : dado que los valores de la función distancia con signo son más importantes en las proximidades de la superficie, sólo se puede utilizar la frecuencia de muestreo más alta en estas regiones. Por ejemplo, se puede utilizar la estructura de un *octree* jerárquico[19] para almacenar los valores muestreados **[47]**. A su vez, existen otras estructuras adaptables como el *quadtree* de 3-colores, ADF **[48]**, árboles BSP[20], entre otras. Sin embargo, el refinamiento adicional de una celda del *octree* que yace completamente dentro o fuera del objeto no mejora la aproximación de la superficie S (*ver* ***Fig. 2.18***).

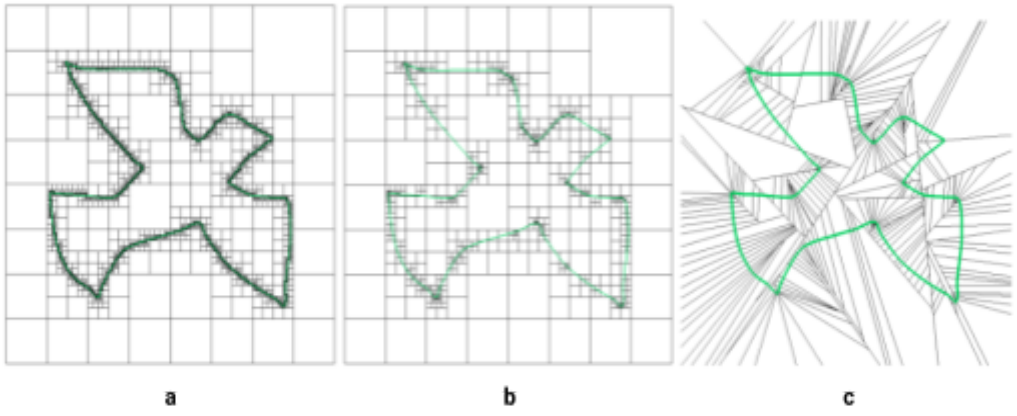

Figura 2.18 [42]: *Estructuras de datos ajustables*. Diferentes aproximaciones ajustables en un campo de distancia con signo con la misma precisión: **(a)** *Quadtree* de 3-colores (12040 celdas). **(b)** ADF (895 celdas). **(c)** *Árbol BSP* (254 celdas).

2.4 MÉTODOS DE CONVERSIÓN DE REPRESENTACIONES DE SUPERFICIES

Para aprovechar las ventajas específicas de las representaciones superficiales paramétricas e implícitas, se necesitan métodos de conversión eficientes entre ellas. Sin embargo, se puede observar que ambos tipos de representaciones son usualmente muestreos finitos (mallas triangulares en el caso explícito, y cuadrículas uniformes/ajustables en el caso implícito) y que cada conversión corresponde a un paso de remuestreo. Por lo tanto, se debe tener especial cuidado para minimizar la pérdida de información durante estas rutinas de conversión.

2.4.1 PARAMÉTRICA A IMPLÍCITA

La conversión de una representación de superficie paramétrica a una implícita equivale al cálculo o aproximación de su campo de distancia con signo. Esto se puede hacer de manera eficiente mediante la voxelización o las técnicas de conversión de escaneo 3D **[49]**. Como el campo de distancia de una superficie en general no es del todo uniforme, una aproximación lineal por partes o trilineal a trozos es el mejor equilibrio entre la precisión de la aproximación y la eficiencia

[19]**Octree jerárquico:** es una estructura en "árbol" de datos en la cual cada nodo interno tiene exactamente 8 "hijos". Las estructuras octree se usan mayormente para particionar un espacio tridimensional, dividiéndolo recursivamente en ocho octantes.
[20]**Árbol BSP:** es un método para subdividir recursivamente un espacio en elementos convexos empleando hiperplanos. Esta subdivisión da lugar a una representación de la escena por medio de una estructura de datos del árbol.

computacional. La conversión de una representación paramétrica como lo es el enfoque de mallas triangulares a una representación implícita, requiere básicamente del cálculo de la función distancia con signo a la malla triangular en los nodos de una cuadrícula 3D (uniforme o ajustable).

Calcular la distancia exacta de un nodo de la cuadrícula a una malla determinada equivale a calcular la distancia al triángulo más cercano. Esta distancia se puede encontrar de manera eficiente mediante estructuras de datos espaciales. Para procesar un campo de distancia con signo, se debe determinar adicionalmente si un nodo de la cuadrícula se encuentra dentro o fuera del objeto **[42]**.

2.4.2 IMPLÍCITA A PARAMÉTRICA

La conversión de una representación implícita o volumétrica a una malla triangular paramétrica, llamada extracción de isosuperficies, ocurre mayormente en el modelado *CSG* y en aplicaciones médicas. El algoritmo estándar de facto para la extracción de la isosuperficie es *Marching Cubes* (Cubos Marchantes) **[50]**. Este método consiste en reconstruir cada celda de la malla independientemente mediante el uso de una tabla de conectividad. Cada celda se encuentra delimitada por ocho vóxeles y doce aristas enumeradas (*ver* ***Fig. 2.19***).

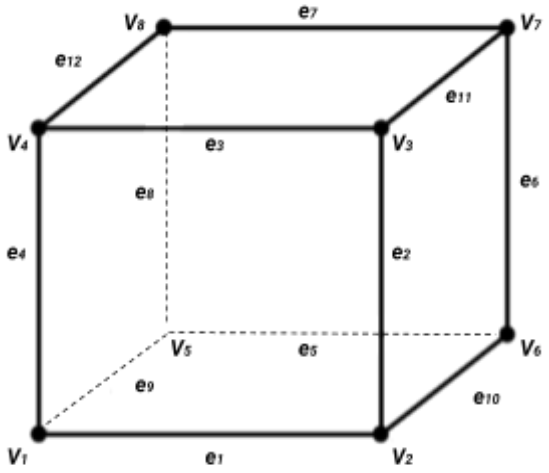

Figura 2.19: *Representación de una celda*. Enumeración de los vértices y aristas que delimitan cada celda.

Para triangular cada celda, se determina inicialmente cuáles de los vóxeles que la delimitan se encuentran dentro de la superficie y cuáles no. Para determinar esto, se construye una máscara de ocho bits, donde el *k-ésimo* bit se encuentra encendido si y sólo si el *k-ésimo* vóxel se encuentra dentro de la superficie (*ver* ***Fig. 2.20***). Un vóxel se encuentra dentro de la superficie si y sólo si su *isovalor* es mayor o igual que el umbral a extraer. De lo contrario, se encuentra fuera.

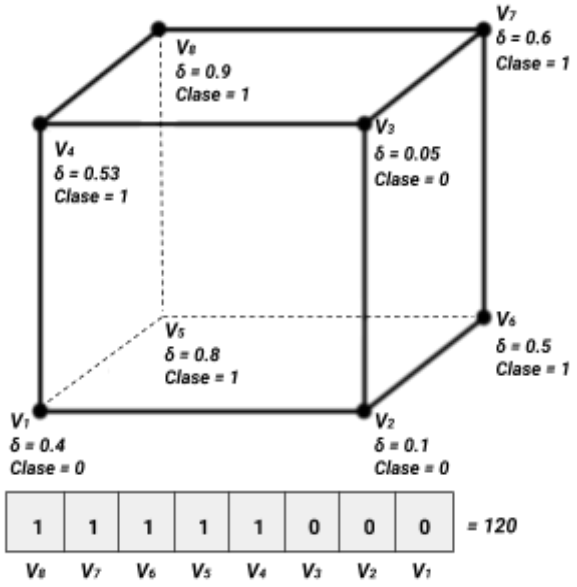

Figura 2.20: *Máscara de bits.* Construcción de la máscara a partir de los valores en los vértices con $\delta = 0.5$.

Debido a que cada uno de los vóxeles tiene dos posibles estados, existe un total de 2^8 formas diferentes en que la superficie a extraer interseca la celda. Sin embargo, la topología de la superficie no cambia si todos los vóxeles cambian a otro estado, por lo que se pueden reducir los 128 casos superiores a los 128 casos inferiores invirtiendo el sentido de los triángulos generados. Luego, por medio de reflexiones y rotaciones se pueden reducir los 128 casos a 15 casos topológicamente diferentes *(ver **Fig. 2.21**)*.

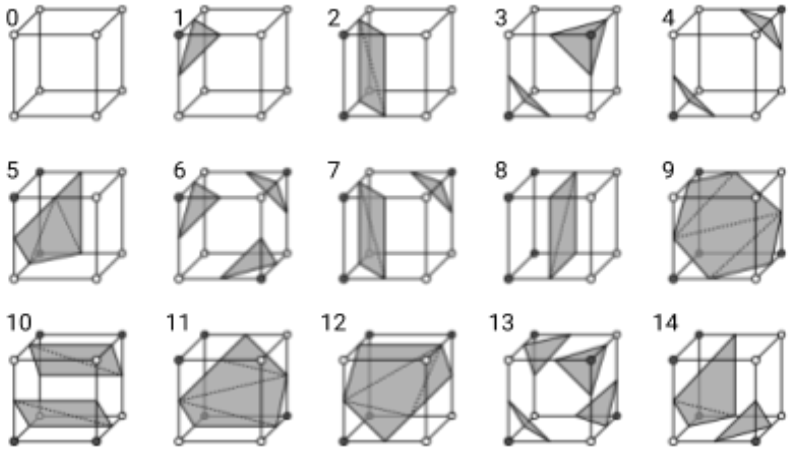

Figura 2.21: *Configuraciones de Cubos Marchantes*. Los puntos negros se consideran marcados, y los puntos blancos se consideran no marcados.

El siguiente paso consiste en determinar cuáles aristas de la celda son intersecadas por la superficie y dónde se encuentra el punto de intersección. Una arista es intersecada por la superficie si y sólo si uno de sus extremos se encuentra marcado y el otro no. Para determinar el punto P_x donde

la arista es intersecada por la superficie se puede utilizar interpolación lineal sobre la misma de la siguiente manera:

$$t_x = (\alpha - v_0) / (v_1 - v_0) \qquad \textbf{Ec. 2.2}$$

$$P_x = P_0 + (P_1 - P_0)\, t_x \qquad \textbf{Ec. 2.3}$$

En la **Ec. 2.2** y **Ec. 2.3**, v_0 y v_1 son los isovalores de los vóxeles que delimitan la arista, α es el umbral de la superficie que se desea extraer, P_0 y P_1 son las posiciones de los extremos de la arista, y P_x es la posición del punto de intersección en ésta. Después de calcular todos los puntos de intersección de la superficie con la celda se generan los triángulos utilizando la tabla de casos, la cual se indexa usando la máscara con la clasificación de los nodos.

Mediante el algoritmo de Cubos Marchantes también se pueden calcular otros atributos de los vértices del mallado final, por medio de la interpolación lineal de los atributos en los extremos de cada arista intersecada por el modelo. Por ejemplo, se puede calcular el gradiente por vértice G_x interpolando los gradientes en los extremos de cada arista intersecada de la forma $G_x = G_0 + (G_1 - G_0)\, t_x$, donde G_0 y G_1 son los gradientes en los extremos de la arista y t_x es el gradiente en el punto de intersección en ésta.

El algoritmo de Cubos Marchantes calcula los puntos de intersección en las aristas de una cuadrícula regular, lo cual hace que las aristas o las esquinas agudas se *"corten"*. En cambio, una reconstrucción con respecto a características agudas requeriría puntos de muestra adicionales dentro de las celdas que los contienen. Por lo tanto, la técnica de Cubos Marchantes extendidos **[51]** examina el gradiente G de la función de distancia **[42]** para detectar esas celdas y encontrar puntos de muestra adicionales al intersecar los planos tangentes en los puntos de intersección de la arista (*ver **Fig. 2.22***), en la **Fig. 2.23** se puede observar un ejemplo de la representación del modelo de una turbina mediante el algoritmo estándar de Cubos Marchantes y la técnica extendida.

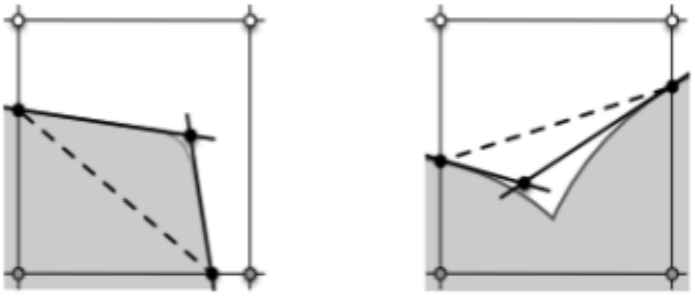

Figura 2.22 [42]: *Extensión de Cubos Marchantes*. Mediante el uso de información del punto y la normal en ambos lados del rasgo agudo de la superficie, se puede encontrar una buena estimación para el punto en la intersección de los elementos tangentes. La línea punteada es el resultado que produciría el algoritmo estándar de Cubos Marchantes, mientras que la formada por las líneas y puntos negros representa el resultado con la técnica extendida.

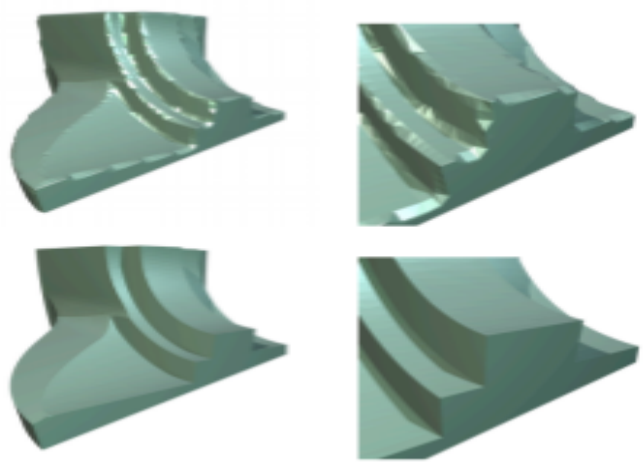

Figura 2.23 [42]: *Comparación entre Cubos Marchantes y extracción iso-superficial*. Dos reconstrucciones del conjunto de datos del modelo de una turbina de tamaño 65 × 65 × 65. El algoritmo estándar de Cubos Marchantes conduce a que las aristas o esquinas agudas se distorsionen (*arriba*), mientras que la extracción iso-superficial sensible a las características reconstruye detalladamente el volumen (*abajo*).

2.5 FORMATOS DE ARCHIVO

Existen muchos formatos de archivo diferentes para almacenar datos de mallas poligonales. Cada formato es más eficaz cuando se utiliza para el propósito previsto por su creador. Algunos de estos formatos se describen en la **Tab. 2.1**.

Extensión del archivo	Nombre del formato	Descripción del archivo
.raw	*Raw mesh*	Formato ASCII. Cada línea contiene 3 vértices, separados por espacios, para formar un triángulo. Ejemplo: $x_1\ y_1\ z_1\ \ x_2\ y_2\ z_2\ \ x_3\ y_3\ z_3$.
.obj	*Wavefront*	Los vértices son representados en cada línea por el caracter **v**, seguido de las coordenadas x,y,z de cada uno. Las coordenadas de textura s,t serán precedidas por la cadena **vt**. Las coordenadas de las normales x,y,z serán precedidas por la cadena **vn**. Cada cara será precedida por el caracter **f** y tendrá el vértice, coordenadas de textura y normales correspondientes separadas por una barra diagonal, es decir, *v1/vt1/vn1*. En este tipo de archivo también se pueden incluir referencias a otros objetos y materiales.
.off	*Object File Format*	La primera línea contiene al número de vértices, número de caras y número de aristas. Para cada **N** vértices habrán **N** líneas con las coordenadas x,y,z de los mismos. Para cada E aristas habrán **E** líneas conteniendo el número número V de vértices y los índices de los **V** vértices que forman una cara.

Tab. 2.1: Formatos de archivo para almacenar mallados.

CAPÍTULO III - DISPERSIÓN DE LA LUZ

El despliegue directo de volúmenes (*DVR*) a menudo se basa en un modelo óptico de emisión y absorción **[52]**. Sin embargo, la percepción espacial de la forma del volumen es compleja debido a la ausencia de variaciones de luminosidad y sombras. A su vez, la iluminación natural proporciona rasgos visuales importantes para percibir de mejor manera la profundidad espacial. Un estudio realizado por Lindemann y Ropinski **[53]** demostró que las técnicas de despliegue de volúmenes basadas en sombras son superiores a aquellas que usan iluminación local en cuanto a la determinación del tamaño relativo de las características volumétricas de un objeto. Una alternativa común a las sombras direccionales es la oclusión ambiental volumétrica, esta técnica busca regiones donde se atenúe la luz ambiental mediante el uso de una región esférica finita alrededor de cada punto para crear sombras suaves locales. De esta manera, las concavidades en un objeto se muestran más oscuras que las regiones no cubiertas. Sin embargo, el estudio de Langer y Bulthoff **[54]** demostró que la percepción humana no se basa únicamente en que "la oscuridad significa profundidad". En particular, las inter-reflexiones influyen en la percepción de la forma de acuerdo a diferentes condiciones de iluminación. El objetivo es desarrollar un modelo de iluminación que explique estos aspectos.

Los modelos de iluminación son importantes para la síntesis de imágenes fotorealistas. Modelar correctamente la interacción de la luz con objetos geométricos es una tarea compleja. A través de los años, diversos modelos de iluminación han sido desarrollados para la síntesis de las imágenes. Muchos investigadores han simulado con éxito los efectos que logran estos resultados, tales como Hanrahan y Krueger **[67]**.

Muchos modelos de sombreado **[55]** se fundamentan en vectores normales, que generalmente se derivan de los gradientes. Sin embargo, en sub-volúmenes homogéneos o en áreas con baja relación señal-ruido, la dirección del gradiente no es numéricamente estable y el sombreado es susceptible a elementos externos. Una alternativa común es un modelo de iluminación de dispersión **[52]** basado en rayos de sombra y una función de fase. El inconveniente de este enfoque es que sólo una pequeña fracción de luz interactúa con el medio, lo que puede conducir a un oscurecimiento fuerte y sombras duras. Estos problemas pueden evitarse con técnicas de oclusión ambiental que integran la reducción de la intensidad de la luz sobre una pequeña región esférica o un cono direccional **[56]** alrededor de cada punto y así calcular un factor de oclusión local para el sombreado de la luz ambiental.

Las inter-reflexiones de luz más complejas conducen a una iluminación más natural, pero el cálculo de dispersión múltiple requiere la integración del transporte de la luz de alta dimensión en la proximidad de cada punto. La integración estocástica de Monte-Carlo se adapta adecuadamente a la resolución de la transferencia radiativa en los medios participantes **[57]**. Sin embargo, para obtener resultados sin ruido, se requieren altas tasas de muestreo, lo cual es computacionalmente costoso para una visualización interactiva del volumen. No obstante, las altas tasas de muestreo son un problema bien conocido en el contexto del modelo clásico de absorción-emisión, especialmente si la función de transferencia contiene altas frecuencias como picos agudos para visualizar isosuperficies.

Una técnica establecida para acelerar la representación es la integración previa de la función de transferencia. Sin embargo, la adaptación de la integración previa a la dispersión volumétrica no es directa debido a que la dispersión requiere la solución de la iluminación global. Para conjuntos de datos no homogéneos, la dimensión de cualquier técnica de integración previa excede los límites prácticos en términos de cálculo y almacenamiento. Por lo tanto, es vital simplificar el cálculo del

transporte de la luz teniendo en cuenta las demandas específicas de la visualización interactiva del volumen.

1 FUNCIONES DE REFLECTANCIA

Es importante describir la manera en que la luz interactúa con un material, más aún, se requiere conocer cómo se relacionan la luz saliente y entrante en un punto de una superficie.

1.1 FUNCIONES BRDF

Una de las posibles maneras para describir la interacción entre un material y la luz, es mediante una función BRDF **[58]**, acrónimo de función de distribución de reflectancia bidireccional. La función BRDF $f(x, \omega_i, \omega_o)$ es definida sobre un punto x de la superficie como la tasa diferencial entre la luz saliente y la entrante:

$$f(x, \omega_i, \omega_o) = \frac{dL_o(x,\omega_o)}{dE_i(x,\omega_i)} = \frac{dL_o(x,\omega_o)}{L_i(x,\omega_i)cos\theta_i d\omega_i}$$

donde el término L_o representa la luz saliente, E_i la irradiancia, L_i la luz incidente, ω_o la dirección saliente, ω_i la dirección incidente y $cos\theta_i$ representa el producto punto entre la normal de la superficie y la dirección de la fuente de luz. La BRDF indica que la luz entrante y saliente son proporcionales, así que la energía que colisiona con el material en el punto x es proporcional a la energía que sale del mismo punto. Las funciones BRDF contienen generalmente las siguientes propiedades:

- **Reciprocidad**: La función BRDF será la misma en caso de intercambiar la dirección saliente y entrante:

$$f(x, \omega_i, \omega_o) = f(x, \omega_o, \omega_i)$$

- **Anisotropía**: Si la superficie cambia de orientación y los términos ω_i y ω_o se mantienen iguales, las funciones BRDF son diferentes:

$$f(x, \omega_i, \omega_o) \neq f(x, R\omega_o, R\omega_i)$$

donde R es una matriz de rotación con un eje arbitrario alrededor del punto x.

- **Positividad**: Dado que las funciones BRDF regulan el transporte entre dos cantidades positivas, es decir, la luz entrante y saliente, entonces:

$$f(x, \omega_o, \omega_i) \geq 0$$

- **Conservación de la energía**: La energía del rayo de la luz saliente no es mayor a la del rayo de la luz entrante, por lo tanto:

$$\int_{2\pi} f(x, \omega_o, \omega_i)cos\theta_o d\omega_o \leq 1$$

Las funciones BRDF tienen algunas limitaciones, y no son capaces de calcular todos los efectos de dispersión deseados. Por ejemplo, con una BRDF no es posible calcular efectos de

subsurface scattering, dado que ésta asume que la luz entra y sale del material en un mismo punto (*ver* ***Fig. 3.1***).

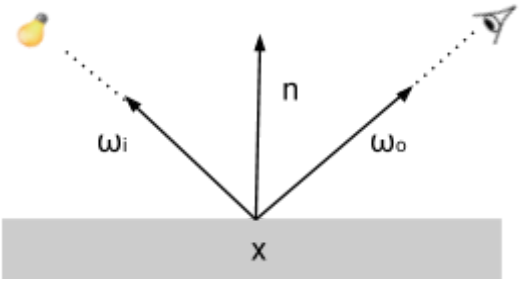

Figura 3.1: *Configuración de una función BRDF*. Nótese que la luz entra y sale de la superficie en el mismo punto.

1.2 EJEMPLOS DE FUNCIONES BRDF

En esta sección se ilustran algunos ejemplos de funciones BRDF como lo son la función BRDF Lambertiana o difusa, la función BRDF especular o de espejo y la BRDF brillante.

- **BRDF Lambertiana**: En este tipo de BRDF, la luz entrante es distribuida equitativamente en todas las direcciones, sin importar la dirección entrante. Para lograr esto, la BRDF debe permanecer constante (*ver* ***Fig. 3.2***).

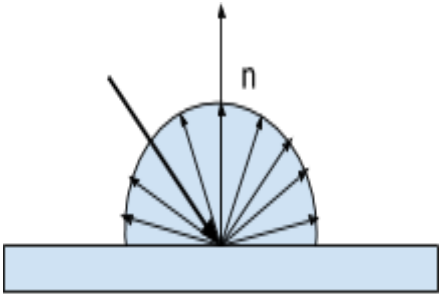

Figura 3.2: *Configuración de una BRDF Lambertiana*.

- **BRDF de espejo**: En este tipo de BRDF, toda la luz entrante de una dirección ω_i es transferida hacia una dirección reflejada ω_r, definida como $\omega_r = \omega_i - 2(\omega_i \cdot n)\, n$ (ver **Fig. 3.3**).

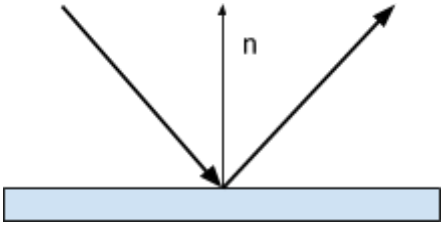

Figura 3.3: *Configuración de una BRDF de espejo*.

- **BRDF brillante**: En la vida real, rara vez los objetos son completamente difusos o completamente especulares. Estos dos modelos representan un caso ideal. Para crear un modelo BRDF realista, a menudo se necesitan combinar los dos términos además de agregar uno adicional, denominado reflexión brillante n (*ver **Fig. 3.4***).

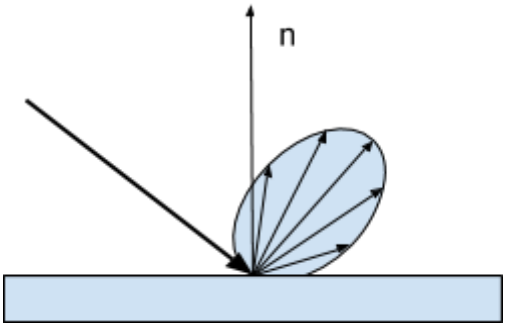

Figura 3.4: *Configuración de una BRDF brillante*.

1.3 ECUACIÓN DE RENDERING

La ecuación de *rendering* propuesta por Kajiya **[66]** es una instancia particular de la ecuación de transferencia que contiene condiciones de bordes adicionales. Esta ecuación aplicada a la técnica de *scattering* en superficies indica que, asumiendo que todas las superficies son opacas, el resplandor emitido por una superficie desde un punto en una dirección determinada es el resultado de la suma del brillo emitido y el brillo dispersado por la superficie *(ver **Ec. 3.1**)*.

$$L_o(x,\omega) = L_e(x,\omega) + \int_{\Omega} f_r(\omega' \rightarrow \omega) L_i(x,\omega') d\omega' \qquad \textbf{Ec. 3.1}$$

donde f_r es la función de distribución de reflectancia bidireccional (BRDF) que describe el *scattering* en las superficies de una manera similar a la función de fase. La formulación original de Kajiya sobre esta ecuación consiste en una integral sobre todas las superficies de la escena. En este caso, se tiene la integral de la ecuación sobre el brillo incidente en la superficie en el punto x.

El segundo operador de suma en la ecuación anterior es conocido comúnmente como la ecuación de reflectancia, la cual describe la distribución del brillo reflejado sobre una superficie resultado de una distribución de brillo incidente L_i (*ver **Ec. 3.2***).

$$L_o(x,\omega) = \int_{\Omega} f_r(\omega' \rightarrow \omega) L_i(x,\omega') d\omega' \qquad \textbf{Ec. 3.2}$$

1.4 ECUACIONES DE FRESNEL

En los modelos BRDF descritos hasta ahora, se considera solamente la parte reflectiva de la luz. Cuando un haz de luz proveniente de una dirección ω_i choca contra una superficie, sólo una parte de la luz entrante es reflectada, mientras que la otra parte es refractada dentro del material.

Como se puede observar en la ***Fig. 3.5***, se obtienen dos vectores ω_r y ω_t, el vector reflejado y el vector refractado **[59]**, definidos como:

$$\omega_r = \omega_i - (\omega_i \cdot n)\, n$$

$$\omega_t = \eta\,((\omega_i \cdot n)\, n - \omega_i) - n\sqrt{1-\eta^2\left(1-(\omega_i \cdot n)^2\right)}$$

donde $\eta = \frac{n_1}{n_2}$ es el índice de refracción relativo entre dos materiales. En particular, se puede decir qué parte de la potencia se propaga en la dirección reflejada y refractada. Los coeficientes que describen esta subdivisión de la potencia se llaman coeficientes de Fresnel. Éstos son diferentes según la polarización de la luz entrante (paralela o perpendicular), de modo que hay dos para la reflexión $(R_s,\ R_p)$ y dos para la transmisión $(T_s,\ T_p)$.

$$R_s(\eta, \omega_i) = \left|\frac{\eta cos\theta_i - cos\theta_t}{\eta cos\theta_i + cos\theta_t}\right|^2 \qquad R_p(\eta, \omega_i) = \left|\frac{\eta cos\theta_t - cos\theta_i}{\eta cos\theta_t + cos\theta_i}\right|^2$$

$$T_s(\eta, \omega_i) = \eta\frac{cos\theta_t}{cos\theta_i}\left|\frac{2cos\theta_i}{\eta cos\theta_i + cos\theta_t}\right|^2 \qquad T_p(\eta, \omega_i) = \eta\frac{cos\theta_t}{cos\theta_i}\left|\frac{2cos\theta_i}{\eta cos\theta_t + cos\theta_i}\right|^2$$

En la mayoría de las aplicaciones de gráficos por computador, se supone que las dos polarizaciones están igualmente mezcladas. Así que, se utilizan los coeficientes $R = \frac{R_s + R_p}{2}$ y $T = \frac{T_s + T_p}{2}$ en los cálculos. Cabe destacar que $R + T = 1$, por lo que la energía total se conserva.

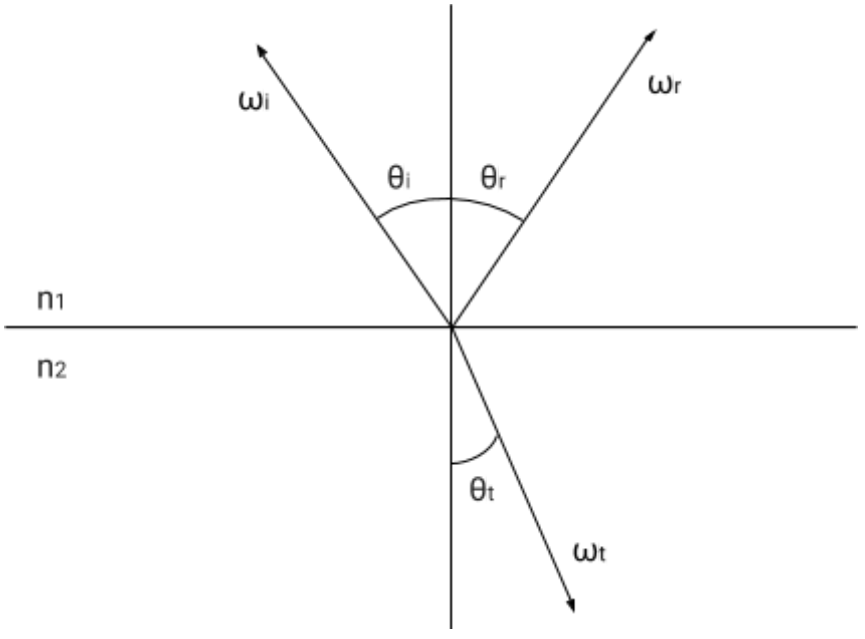

Figura 3.5: *Vector reflejado* ω_r *y vector refractado* ω_t *con índices de refracción diferentes.* Los coeficientes de Fresnel son denotados por n_1 y n_2.

2 TRANSPORTE Y DISPERSIÓN DE LA LUZ

A continuación se mostrarán los pasos principales que afectan la distribución de brillo en un entorno. El primero es la absorción, que describe la reducción del brillo emitido por un rayo debido a la absorción de energía en un medio participante, por ejemplo, su conversión a alguna otra forma de energía como el calor. El siguiente paso es la emisión, la cual describe la distribución de energía que será finalmente añadida al entorno. El paso final es la dispersión (*scattering*), que describe cuánta luz a través de un rayo es esparcida en diferentes direcciones debido a colisiones con partículas en el medio. En este caso se tratarán dos fenómenos de dispersión como lo son la dispersión entrante (*in-scattering*) y la dispersión saliente (*out-scattering*). Además, se presenta el modelo iluminación utilizado para el desarrollo.

2.1 ABSORCIÓN

La absorción de la iluminación es el proceso donde el brillo es captado por la materia. Cuando la absorción se produce dentro de un rango de luz visible, recibe el nombre de absorción óptica. Esta iluminación, al ser absorbida, puede ser reemitida o transformarse en otro tipo de energía, como calor o energía eléctrica (*ver **Fig. 3.6***). Según Pharr y Hanrahan **[68]**, se considera T_r una cantidad fraccionaria de brillo transmitida en un rayo desde un punto a otro. Esta cantidad sólo tomará en cuenta el brillo absorbido en el medio. Este hecho es algunas veces descrito como absorción real. Dado un rayo que parte de una posición x, en una dirección ω y con una longitud r, se definirá la función de atenuación en un punto x a través del rayo como sigue:

$$a_r(x, \omega) = \frac{1 - T_r(x, x + r\omega)}{r}$$

donde $T_r(x, x + r\omega)$ es la fracción de brillo original en $x + r\omega$ que ha sido transmitido a x. En la notación de límite, el término r tiende a 0 y se tendrá el coeficiente de absorción en x en la dirección ω.

$$\sigma_a(x, \omega) = \lim_{r \to 0} a_r(x, \omega)$$

En la mayoría de las aplicaciones, σ_a es solo una función de posición y tiene el mismo valor para todas las direcciones.

Se puede demostrar que:

$$\frac{dTr}{dr} = -\sigma_a T_r ,$$

de lo cual se deduce:

$$T_r = e^{-\int_0^r \sigma_a(x)\, dx} ,$$

Para σ_a constante, aplicando la ley de Beer[21],

$$T_r = e^{-\sigma_a r}$$

[21]**Ley de Beer**: consiste en la relación que describe la atenuación a las propiedades del material a través de donde viaja la luz.

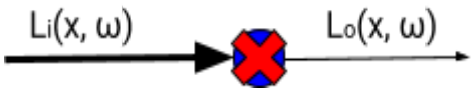

Figura 3.6: *Absorción de luz en un punto* x.

2.2 EMISIÓN

La emisión consiste el proceso donde se irradia la luz desde los electrones de átomos que han recibido energía en algún momento determinado. Los procesos de emisión son necesarios para que exista algún tipo de energía en un entorno y así sea visible cualquier objeto en él. Diversos procesos químicos y térmicos, transforman la energía en ondas de luz visibles capaces de iluminar un medio. Se denotará al brillo emitido en un punto x de una superficie, con una dirección ω como $L_o(x, \omega)$ (*ver* ***Fig. 3.7***).

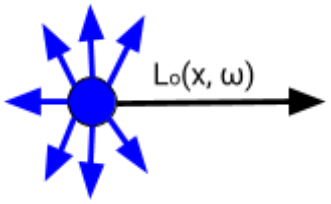

Figura 3.7: *Emisión de luz en un punto* x.

2.3 SCATTERING EN UN PUNTO

Debido a que un rayo de luz se propaga a través de un medio, puede que éste colisione con partículas y sea dispersado en diferentes direcciones. Este tipo de dispersión es comúnmente denominado *scattering* simple a diferencia del *scattering* múltiple que comprende la combinación de múltiples eventos de *scattering* simple. La principal diferencia entre los efectos de la dispersión simple y múltiple es que la dispersión simple generalmente se puede tratar como un fenómeno aleatorio, mientras que la dispersión múltiple, de manera un tanto intuitiva, se puede modelar como un proceso más determinista, ya que los resultados son la combinación de una gran cantidad de eventos de dispersión simple que se tienden a promediar. Se han implementado una gran cantidad de funciones que describen la variedad de medios de dispersión, desde modelos parametrizados, los cuales son utilizados para ajustar una función con pocos parámetros para visualizar datos, hasta modelos analíticos, que surgen de la derivación directa de la distribución de brillo resultante de algún tipo de dispersión, por ejemplo, las partículas esféricas.

Para lograr describir la distribución de la iluminación dispersada en un punto, es necesaria la utilización de una función fase. En la mayoría de los medios naturales, la función de fase es una función del ángulo entre dos direcciones ω y ω'; tal medio se llama isótropo[22] y estas funciones de fase a menudo se escriben como $p(cos\theta)$. En un medio anisótropo[23], tal como uno de estructura

[22]**Medio isótropo**: comprende a un medio donde todas sus propiedades físicas son iguales en todas las direcciones y no dependen de alguna dirección en específico.
[23]**Medio anisótropo**: comprende a un medio donde sus propiedades físicas varían según la dirección en la que es estudiado.

cristalina, la función de fase es una función que representa los valores de cada uno de los dos ángulos. Una propiedad importante de las funciones de fase de origen natural es que son recíprocas: las dos direcciones pueden intercambiarse y el valor de la función de fase permanece inalterado.

Las funciones de fase se definen generalmente para que estén normalizadas, en todas las direcciones de ω,

$$\frac{1}{4\Pi}\int_{S^2} p(\omega \rightarrow \omega')\, d\omega' = 1 \qquad \textbf{Ec. 3.3}$$

Al igual que los medios, las funciones de fase pueden ser isotrópicas y anisotrópicas. La función de fase isotrópica es independiente de cualquiera de los dos ángulos y siempre tiene un valor de 1 (para satisfacer la **Ec. 3.3**). Una función de fase anisotrópica depende del ángulo entre las dos direcciones, y de si el medio es isótropo o anisótropo, respectivamente.

Las funciones de fase a menudo se describen mediante un parámetro de asimetría g, que es el valor promedio del producto de la función de fase con el coseno del ángulo entre ω' y ω. El rango de g está comprendido entre -1 y 1, y corresponde a la dispersión total de la luz de atrás hacia adelante (*ver **Fig. 3.8***). Para cualquier ω,

$$g = \frac{1}{2}\int_{S^2} p(\omega \rightarrow \omega')(\omega \cdot \omega')\, d\omega' \qquad \textbf{Ec. 3.4}$$

Por lo tanto, la dispersión isotrópica corresponde a un g nulo. Cualquier número de funciones de fase puede satisfacer la **Ec. 3.5**; el valor de g solo no es suficiente para describir de manera única una distribución de dispersión. No obstante, la facilidad de convertir una distribución de dispersión compleja en un modelo parametrizado simple compensa la pérdida de precisión. La función de fase de Henyey-Greenstein ha sido ampliamente utilizada y es computacionalmente eficiente.

$$p(\omega \rightarrow \omega') = \frac{1-g^2}{(1+g^2-2g(-\omega\cdot\omega'))^{3/2}} \qquad \textbf{Ec. 3.5}$$

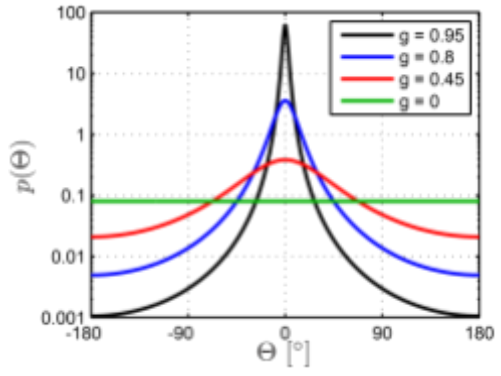

Figura 3.8: *Función de fase*. La función de fase describe la distribución de brillo disperso en la dirección ω' en un punto, dado el brillo incidente a lo largo de la dirección ω. Se puede observar la función de fase de *Henyey-Greenstein*[24] con un parámetro de asimetría denotado por g igual a 0, 0.45, 0.8 y 0.95.

[24]**Función de fase de Henyey-Greenstein**: es una función de fase analítica de un sólo parámetro utilizada en *scattering*.

2.3.1 COEFICIENTE DE SCATTERING

El coeficiente de dispersión σ_s describe la probabilidad de que ocurra un evento de dispersión por unidad de distancia en el medio. Cuando éste es añadido al coeficiente de absorción, da como resultado el coeficiente de atenuación:

$$\sigma_t = \sigma_a + \sigma_s \qquad \textbf{Ec. 3.6}$$

el cual indica la reducción total de energía provocada por ambos fenómenos.

Un valor útil basado en el coeficiente de absorción es el espesor óptico[25] entre dos puntos. Este último es representado mediante la integral sobre la línea entre los dos puntos del coeficiente de absorción:

$$\int_{x}^{x'} \sigma_t(x'') \, dx''$$

El albedo es otro valor importante basado en los coeficientes de dispersión y atenuación. Éste indica la fracción de resplandor incidente que es dispersado nuevamente:

$$\alpha = \frac{\sigma_s}{\sigma_t}$$

Los medios que contienen un alto albedo, con un α cercano a 1, redistribuyen la mayor parte de la luz incidente en cada interacción de dispersión, mientras que los medios de bajo albedo, con α cercano a 0, absorben la mayor parte.

2.3.2 SCATTERING SALIENTE

El *scattering* saliente consiste en la luz atenuada debido a la dispersión de la luz. El fenómeno de *scattering* ocurre cuando los fotones son desviados de la dirección actual ω (*ver* ***Fig. 3.9***). La atenuación debido al *scattering* saliente se modela como:

$$(\nabla \cdot \omega) \, L(x, \omega) = -\sigma_s(x, \omega) \, L(x, \omega) \qquad \textbf{Ec. 3.7}$$

Cabe destacar que en este caso no importa en qué dirección están dirigidos los fotones. Las direcciones de los mismos se tomarán en cuenta en el término de *scattering* entrante de otro punto en el material.

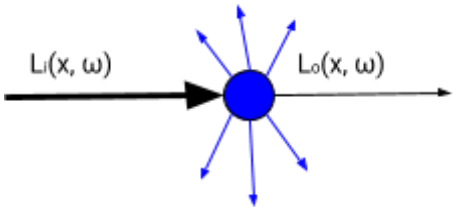

Figura 3.9: *Scattering saliente en un punto* x.

[25]**Espesor óptico**: es la capacidad para atenuar la radiación térmica de longitud de onda de una longitud dada con material semi-transparente.

2.3.3 SCATTERING ENTRANTE

Dada la pérdida de algunos de los fotones de la luz al cambiar su dirección, otros eventos de dispersión cambiarán a su vez la dirección ω. Se requiere conocer el número de fotones que provienen de todas las demás direcciones. Para realizar esto, se integra el transporte de la luz entrante desde todas las direcciones en el punto x (ver **Fig. 3.10**). Esta cantidad, similar a luz saliente en un medio infinito es denominada fluencia y se denota como ϕ:

$$\phi(x) = \int_{4\Pi} L(x, \omega') \, d\omega' \qquad \textbf{Ec. 3.8}$$

La fluencia debe ser promediada sobre una esfera de direcciones, tomando en cuenta $\frac{\Phi}{4\Pi}$ como factor de normalización. Esta cantidad es multiplicada por el coeficiente de *scattering*, dado que sólo algunos fotones se dispersan en el punto actual. Esto resulta como:

$$(\nabla \cdot \omega) \, L(x, \omega') = \sigma_s(x) \, \frac{1}{4\Pi} \int_{4\Pi} L(x, \omega') \, d\omega' \qquad \textbf{Ec. 3.9}$$

Sin embargo, la **Ec. 3.9** asume que la luz se dispersa equitativamente en todas las direcciones. Esto usualmente no ocurre, y el término $\frac{1}{4\Pi}$ debe ser reemplazado por una función de distribución de probabilidad que describe cuánto se dispersan los fotones en el medio. Esta última es la función de fase p descrita anteriormente.

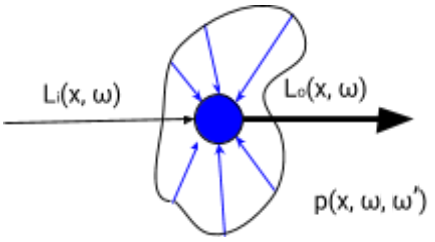

Figura 3.10: *Scattering entrante en un punto* x.

2.4 ECUACIÓN DE TRANSFERENCIA RADIATIVA

La ecuación de transferencia radiativa (ETR) es la fórmula fundamental que muestra el comportamiento de la luz en algunos medios que absorben, emiten y dispersan brillo. Ésta describe la distribución equilibrada del brillo en términos de resplandor incidente en el medio y sus propiedades de dispersión. La mayoría de los algoritmos de despliegue en computación gráfica se enfocan en resolver esta ecuación para calcular el resplandor en un medio.

Debido a que la luz viaja a través de un rayo, diversos procesos contribuyen a cambiar su distribución. El resplandor de la luz puede ser incrementado producto de la emisión y *scattering* entrante. A su vez, el resplandor puede ser disminuido producto de la absorción y *scattering* saliente. En su forma más básica, esta ecuación describe cuánto cambia el resplandor de la luz de un rayo en un punto. Puede derivarse sustrayendo los efectos de los procesos de dispersión que reducen la energía a lo largo del rayo (absorción y dispersión saliente) de los procesos que aumentan la energía

a lo largo del mismo (emisión y dispersión entrante). Se asume que el medio tiene un índice de refracción constante, es decir, un rayo sigue un camino de línea recta.

3 DISPERSIÓN SUBSUPERFICIAL DE LA LUZ EN MODELOS MALLADOS

La dispersión de la luz en modelos se basa en la aproximación de una ETR definida aproximación de la difusión, la cual asume a su vez la utilización de materiales altamente dispersores. En este caso, se puede obtener una nueva ecuación denotada como BSSRDF[26] para una superficie planar en un medio semi-infinito.

3.1 MODELO DE DIPOLO ESTÁNDAR

El primer modelo a describir se trata en el trabajo de Jensen en el 2001 **[60]**. Generalmente, es referido como el modelo de dipolo de Jensen o modelo de dipolo estándar. En la investigación original, los autores utilizan aproximaciones difusas para la luz en un medio infinito. Empezando en este punto, derivan una aproximación que mantiene la luz en un medio semi-infinito, es decir, la luz viaja en el vacío hasta colisionar con una superficie translúcida. Como condición de borde, se toma en cuenta la luz saliente del material. Esta última tiene una fluencia inicial φ_0. Se asume que la fluencia se mantiene constante hasta una distancia $z = 2AD$ desde la superficie, donde se vuelve 0. D corresponde al coeficiente de difusión, mientras que A es el término correctivo que calcula los índices de refracción:

$$A = \frac{1+F_{dr}}{1-F_{dr}}$$

$$F_{dr} = \int_{2\pi} R(\eta, n \cdot \omega)(n \cdot \omega) d\omega$$

donde R es el término de Fresnel descrito anteriormente, y $\eta = n_1 / n_2$ es el índice de refracción relativa. La integral de reflectancia de Fresnel F_{dr} es usualmente aproximada con una expresión analítica:

$$F_{dr} = -\frac{1.440}{\eta^2} + \frac{0.710}{\eta} + 0.668 + 0.0036\eta$$

Dada la condición borde, se puede modelar el *subsurface scattering* en un punto x_o con dos fuentes en el mismo, y llevar a cabo una configuración denominada dipolo. Una fuente es colocada debajo de la superficie, denominada fuente real, mientras que la otra es una copia exacta de la anterior ubicada por encima de la superficie, y es denominada fuente virtual. La primera fuente modela el efecto del *subsurface scattering*, mientras que la segunda reduce los efectos de la primera (*ver **Fig. 3.11**)*.

[26]**BSSRDF:** consiste en la función de distribución de reflectancia bidireccional para dispersión subsuperficial de la iluminación que describe la relación entre la luz saliente y entrante incluyendo el fenómeno de dispersión subsuperficial, y a su vez, especifica cuánta luz es transportada entre dos rayos que colisionan una superficie.

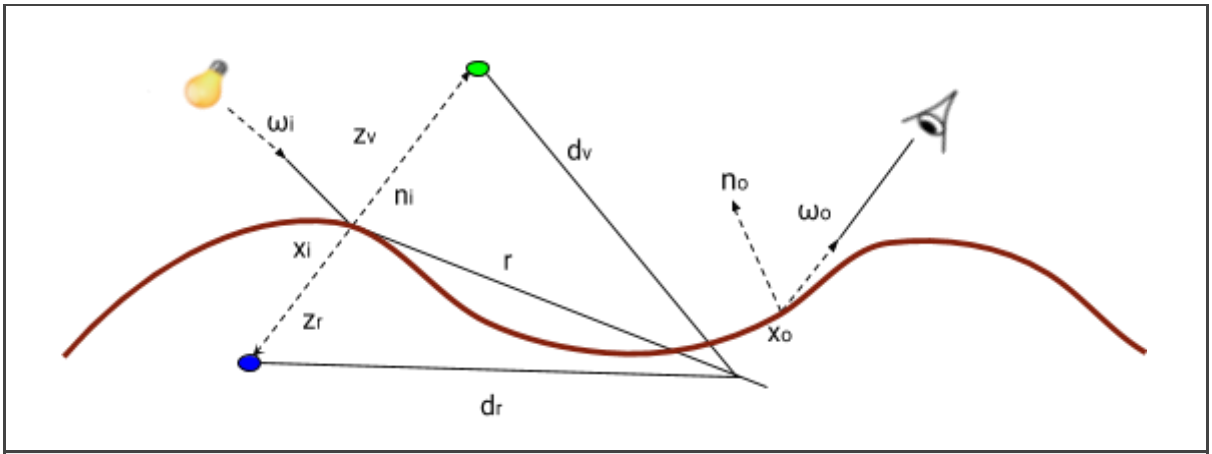

Figura 3.11: *Dipolo estándar*. Configuración para el modelo de dipolo estándar donde el punto azul representa a la fuente real, el punto verde representa a la fuente virtual y la curva marrón representa la superficie translúcida.

3.2 MODELO DE DIPOLO DIRECCIONAL

Diversas variaciones del modelo de dipolo estándar han sido propuestas a través de los años. Frisvad en el 2014 **[61]**, propone una aproximación de la función BSSRDF denominada dipolo direccional (*ver **Fig. 3.12***). En el modelo de dipolo estándar, el término difuso de la ecuación BSSRDF depende únicamente de la distancia entre el punto de incidencia y el punto de emergencia, el cual es denotado como:

$$S_d(x_i, \omega_i, x_o, \omega_o) = S_d(\| x_o - x_i \|)$$

El modelo de dipolo direccional, basado en la aproximación difusa, toma en cuenta la dirección de la luz entrante en sus cálculos, para modelar los efectos de *scattering* de manera más precisa. Cabe destacar que este modelo segmenta la ecuación BSSRDF en un término difuso S_d y en un término $S_{\delta E}$, llamado intensidad reducida, que puede ser calculada utilizando una aproximación denominada Delta-Eddington **[62]**. La BSSRDF final resulta de la siguiente forma:

$$S(x_i, \omega_i, x_o, \omega_o) = T(\eta, \omega_i)(S_d(x_i, \omega_i, x_o) + S_{\delta E}(x_i, \omega_i, x_o, \omega_o))T(\eta, \omega_o)$$

donde T corresponde a los coeficientes de transmisión de Fresnel para las direcciones entrantes y salientes. Es importante mencionar que la parte difusa de la BSSRDF no depende de la dirección saliente ω_o.

3.2.1 BSSRDF DIFUSA

La parte difusa del modelo de dipolo direccional utiliza una aproximación de primer orden de la ETR, que para un punto de luz en un medio infinito resulta la siguiente fluencia:

$$\phi(x_o, \theta) = \frac{\Phi}{4\pi D} \frac{e^{\sigma_{t_r} r}}{r} \left(1 + 3D\frac{1+\sigma_{t_r} r}{r} cos\theta\right) \qquad \textbf{Ec. 3.10}$$

donde D y σ_{t_r} son los dos coeficientes de dispersión definidos anteriormente. $r = ||x_o||$ y:

$$cos\theta = \frac{x \cdot \omega_{12}}{r}$$

donde ω_{12} es el vector refractado definido en la **Sec. 1.4** de este capítulo. La Ec. 3.10 introduce un término que depende del ángulo θ entre el vector refractado de luz entrante (ω_i) y el vector director de luz saliente (ω_o).

Usando la aproximación difusa, se puede establecer una relación entre la luz emitida $M(x_o)$ y la BSSRDF difusa S_d' en un medio infinito:

$$\frac{dM(x_o)}{d\Phi_i(x,\omega_i)} = T(\eta, \omega_i) S_d'(x_i, \omega_i, x_o) 4\Pi C_\phi(1/\eta) \qquad \textbf{Ec. 3.11}$$

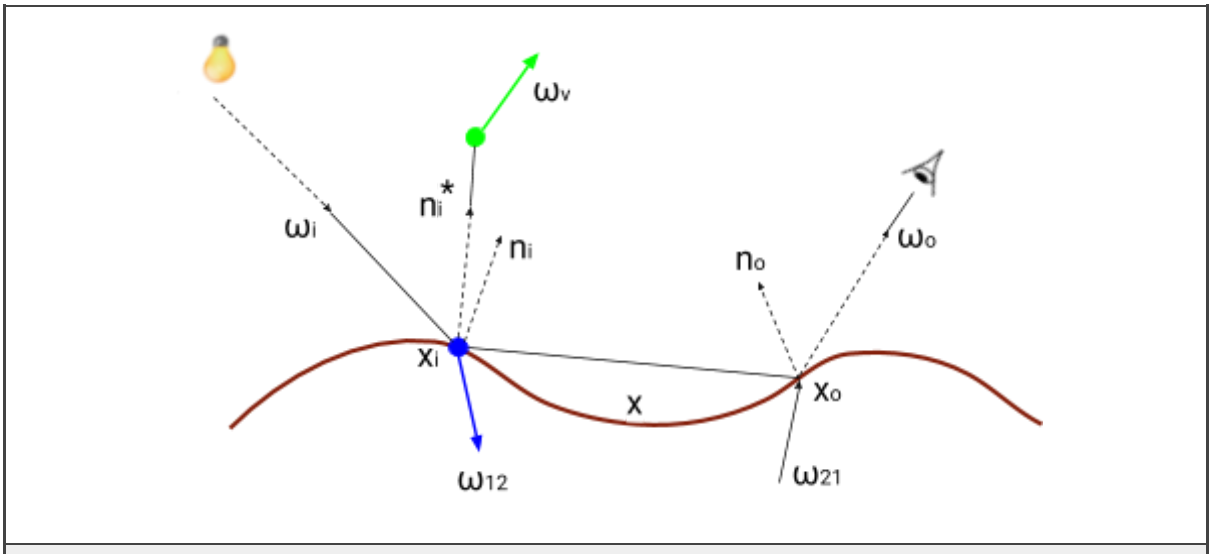

Figura 3.12: *Dipolo direccional*. Configuración para el modelo de dipolo direccional.

donde $C_\phi(\frac{1}{\eta})$ está relacionado con la integral de la hemi-esfera de los coeficientes de Fresnel. Utilizando la definición de la luz emitida e insertando la aproximación clásica de difusión, se denota la formulación de difusión de la radiancia emitida como sigue:

$$M(x_o) = C_\phi(\eta)\phi(x_o) + C_E(\eta) D n_o \cdot \nabla\phi(x_o) \qquad \textbf{Ec. 3.12}$$

Nuevamente, $C_\phi(\eta)$ y $C_E(\eta)$ son dos términos que están relacionados a la integración de los coeficientes del Fresnel. Combinando las 3 ecuaciones **2.14**, **2.15** y **2.16**, se llega a la forma final de la ecuación BSSRDF en un medio infinito:

$$S'_d(x, \omega_{12}, r) = \frac{1}{4C_\phi(1/\eta)} \frac{1}{4\Pi^2} \frac{e^{-\sigma_{tr} r}}{r^3} + [C_\phi(\eta)\left(\frac{r^2}{D} + 3(1+\sigma_{tr} r) x \cdot \omega_{12}\right) -$$
$$C_E(\eta)\,(3D(1+\sigma_{tr} r)\,\omega_{12} \cdot n_0 - \left((1+\sigma_{tr} r) + 3D\frac{3(1+\sigma_{tr} r)+(\sigma_{tr} r)^2}{r^2} x \cdot \omega_{12}\right) x \cdot n_o)]$$

3.2.2 INTEGRALES DE FRESNEL

Los dos términos $C_\phi(\eta)$ y $C_E(\eta)$ provienen originalmente de la integración de la transmitancia de Fresnel saliente a través de la hemi-esfera, promediada con un término de coseno. Éstos son definidos como:

$$C_\phi(\eta) = \tfrac{1}{4\pi} \int_{2\pi} T(\eta, \omega)(\eta_o \cdot \omega) d\omega$$

$$C_E(\eta) = \tfrac{3}{4\pi} \int_{2\pi} T(\eta, \omega)(\eta_o \cdot \omega)^2 d\omega$$

Estas dos integrales pueden ser ajustadas para expresarlas en términos de reflectancia en vez de transmitancia, tomando en cuenta $R = 1 - T$:

$$C_\phi(\eta) = \tfrac{1}{4\pi}(\pi - \int_{2\pi} R(\eta, \omega)(\eta_o \cdot \omega) d\omega) = \tfrac{1}{4}(1 - 2C_1)$$

$$C_E(\eta) = \tfrac{3}{4\pi}(\tfrac{2\pi}{3} - \int_{2\pi} R(\eta, \omega)(\eta_o \cdot \omega) d\omega) = \tfrac{1}{4}(1 - 3C_2)$$

D'Eon e Irving **[63]** utilizan una aproximación polinomial para los coeficientes C_1 y C_2, expresados de la siguiente forma:

$$2C_1 \approx 0.919317 - 3.4793\eta + 6.75335\eta^2 - 7.80989\eta^3 + 4.98554\eta^4 - 1.36881\eta^5, \text{ si } \eta < 1$$

$$2C_1 \approx -9.23372 + 22.2272\eta - 20.9292\eta^2 + 10.2291\eta^3 - 2.54396\eta^4 + 0.254913\eta^5, \text{ si } \eta \geq 1$$

$$3C_2 \approx 0.828421 - 2.62051\eta + 3.36231\eta^2 - 1.95284\eta^3 + 0.236494\eta^4 - 0.145787\eta^5, \text{ si } \eta < 1$$

$$3C_2 \approx -1641.1 + \frac{135.926}{\eta^3} - \frac{656.175}{\eta^2} + \frac{1376.53}{\eta} + 1213.67\eta - 568.556\eta^2 + 164.798\eta^3 - 27.0181\eta^4 + 1.91826\eta^5, \text{ si } \eta \geq 1$$

3.2.3 CONDICIONES DE BORDE

Para el dipolo direccional se modelan las condiciones de borde sobre la superficie del material utilizando un dipolo. Sin embargo, en este caso en vez de utilizar dos fuentes de luz puntuales, se utilizan dos fuentes de rayo, una real y una virtual. Al igual que en el modelo de dipolo estándar, la fuente es desplazada hacia la normal en una distancia d_e, que a su vez, puede ser tomada en cuenta como $2D$. Y para el caso de los índices de refracción no coincidentes, esta distancia puede convertirse posteriormente en $2AD$. Para el modelo de dipolo direccional, se utiliza:

$$d_e = \frac{2{,}131D}{\sqrt{\alpha'}}$$

donde se denota $\alpha' = \alpha'_s/\alpha'_t$ como el albedo reducido. A su vez, el término A es modificado utilizando las integrales hemi-esféricas de Fresnel:

$$A = \frac{1 - C_E(\eta)}{2C_\phi(\eta)}$$

Al igual que en el dipolo estándar, el dipolo direccional asume un medio semi-infinito dadas las condiciones de borde anteriores. Para asegurar estas suposiciones, se necesita extender el modelo para reducir efectos no deseados. Una primera modificación propuesta por Frisvad, consiste en utilizar un plano tangente modificado por la normal n_i^* para replicar la fuente real hacia la fuente

de luz duplicada, en vez de utilizar la anteriormente definida por n_i. Se define la normal modificada de la forma:

$$n_i^* = n_i \text{ , si } x_o = x_i$$

$$n_i^* = \frac{x_o - x_i}{\| x_o - x_i \|} \times \frac{n_i \times (x_o - x_i)}{\| \overline{n_i} \times (x_o - x_i) \|} \text{ , en otro caso}$$

Otra modificación importante consiste en la distancia a la fuente real. En el dipolo estándar se considera $d_r = \sqrt{z_r^2 + r^2}$, con $z_r = 1/\sigma'_t$, el cual consiste en la distancia promedio donde un fotón viaja a través del material antes de ser absorbido o dispersado. El problema de esta definición recae en que pueden ocurrir disparidades en $r = 0$. Más aún, el modelo de dipolo estándar puede ser ligeramente impreciso cuando r tiene un valor pequeño, subestimando el efecto en general. Para evitar estos problemas, Frisvad propuso una definición más compleja de d_r que concuerda con las simulaciones de la teoría de transporte. En su investigación, d_r se define de la forma:

$$d_r^2 = r^2 + D\mu_0(D\mu_0 - 2d_e cos\beta) \text{ , si } \mu_0 \geq 0 \text{ (iluminado por delante)}$$

$$d_r^2 = r^2 + \frac{1}{(3\sigma_t)^2} \text{ , si } \mu_0 < 0 \text{ (iluminado por detrás)}$$

donde $\sigma_t = \sigma_s + \sigma_a$ consiste en el coeficiente de atenuación, y $\mu_0 = -\omega_{12} \cdot n_o$ indica si el punto x_o está iluminado por delante o por detrás. β es un término geométrico que es evaluado de la forma:

$$cos\beta = -\sqrt{\frac{r^2 - (x \cdot \omega_{12})^2}{r^2 + d_e^2}}$$

Considerando todas estas correcciones se puede escribir la forma final del modelo BSSRDF, la cual consiste en la combinación del término de la fuente real menos el término de la fuente virtual:

$$S_d(x_i, \omega_i, x_o) = S'_d(x_o - x_i, \omega_{12}, d_r) - S'_d(x_o - x_v, \omega_v, d_v)$$

donde los coeficientes adicionales para la fuente virtual se definen de la forma:

$$x_v = x_i + 2Ad_e n_i^*$$

$$\omega_v = \omega_{12} - 2(\omega_{12} \cdot n_i^*)\, n_i^*$$

$$d_v = \| x_o - x_i \|$$

El modelo del dipolo direccional descrito anteriormente, da mejores resultados visuales que el dipolo estándar, a cambio de unos pocos cálculos adicionales. En particular, este modelo mejora al anterior mayormente para materiales dispersores, donde sus resultados son muy cercanos a los logrados con la técnica de *path tracing*[27] (ver **Fig. 3.13**).

[27] ***Path tracing***: consiste en una técnica de computación gráfica que genera imágenes de escenas tridimensionales tales que los efectos de iluminación global sean realistas.

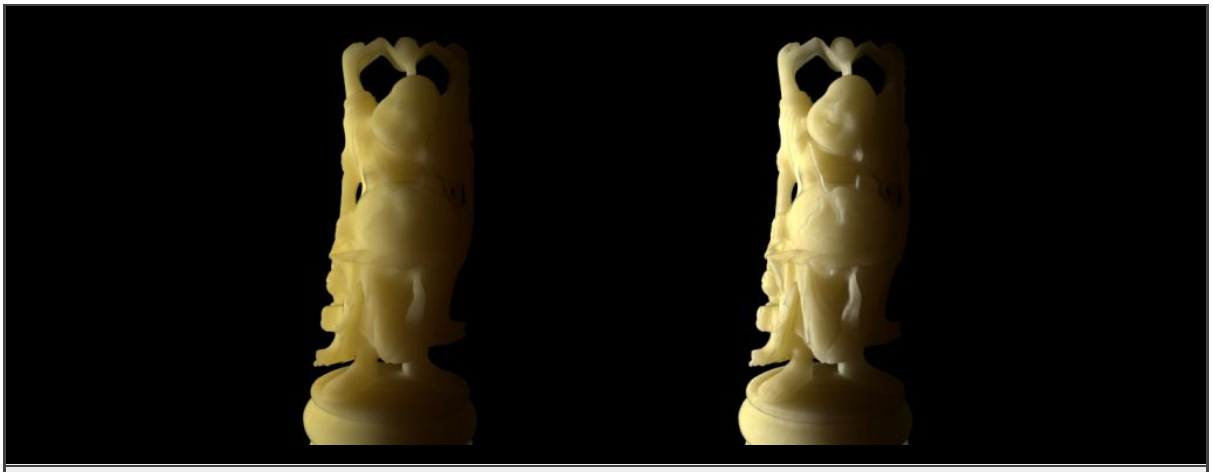

Figura 3.13 [2]: *Comparación del modelo de dipolo estándar* (izquierda) *y direccional* (derecha) para el modelo del Buda de Stanford hecho de patata, con una luz direccional. Se puede observar que el dipolo direccional es capaz de capturar detalles más finos y proveer un aspecto menos planar que el dipolo estándar.

4 DISPERSIÓN DE LA LUZ EN DATOS VOLUMÉTRICOS

La dispersión de la luz y las sombras tienen una gran influencia en la calidad de las imágenes finales generadas por la técnica de *volume rendering* **[3]**. La luz $L_s(x, \omega_o)$ dispersada desde la posición x dentro del volumen en la dirección ω_o es definida como sigue:

$$L_s(x, \omega_o) = s(x, \omega_i, \omega_o) \cdot L_i(x, \omega_i) + L_e(x)$$

donde $L_i(x, \omega_i)$ representa a la luz incidente alcanzando el punto x desde la dirección ω_i, $L_e(x)$ es la radiancia emisiva y $s(x, \omega_i, \omega_o)$ describe a una función que requiere tanto a la dirección de la luz incidente ω_i como a la dirección de la luz saliente ω_o. La función $s(x, \omega_i, \omega_o)$ depende de diversos parámetros, los cuales pueden variar basados en el punto x, al igual que las propiedades ópticas asignadas a través de la función de transferencia. En el contexto de *volume rendering*, $s(x, \omega_i, \omega_o)$ es comúnmente descrita como:

$$s(x, \omega_i, \omega_o) = \tau(x) \cdot p(x, \omega_i, \omega_o)$$

donde $\tau(x)$ representa el coeficiente de extinción en la posición x, y $p(x, \omega_i, \omega_o)$ es la función de fase que describe las características dispersoras de la luz en el medio participante en la posición x. Cuando se incorpora la atenuación de una luz externa que viaja a través del volumen, la luz incidente puede ser definida como:

$$L_i(x, \omega_i) = L_l \cdot T(x_l, x)$$

Esta definición está basada en la integral estándar de *volume rendering*, donde la fuente de luz que está posicionada en el punto x_l contiene a la radiancia L_l, y $T(x_l, x)$ es la transparencia entre x_l y x. De acuerdo a Max **[64]**, la integral estándar de *volume rendering* que simula la absorción y emisión puede ser extendida con estas definiciones para agregar efectos de *scattering* simple y sombras de la siguiente forma:

$$L(x, \omega_o) = L_0 \cdot T(x_0, x) + \int_{x_0}^{x} (L_e(x) + (s(x, \omega_i, \omega_o) \cdot L_i(x, \omega_i))) \cdot T(x', x)\, dx'$$

donde $\omega_i = x_l - x'$, L_0 corresponde a la intensidad de fondo, y x_0 es un punto detrás del volumen. Al extender esta ecuación para soportar *scattering* múltiple, es necesario integrar la dispersión de la luz proveniente de todas las posibles direcciones ω_i sobre la esfera. Al igual que en el trabajo de Kniss **[65]**, se debe incorporar iluminación indirecta difuminando la luz incidente con un cono centrado en la dirección de la misma. En contraste a su técnica, este enfoque puede ser fácilmente extendido, para también calcular los efectos de dispersión que ocurren cuando la luz viaja a través del ojo. Para lograr esto, se realiza un pase adicional de *scattering* a un volumen de luz (*ver* ***Sec. 2*** del siguiente capítulo), el cual difumina la luz saliente utilizando un cono ubicado de espaldas a la dirección de la vista. Por otra parte, para garantizar bordes de sombras duras, se omite el difuminado de la intensidad y se aplica interpolación lineal estándar (*ver* ***Fig. 3.14***). Así, el modelo de iluminación puede ser especificado de la forma:

$$L(x, \omega_o) = L_0 \cdot T(x_0, x) + \int_{x_0}^{x} (L_e(x) + q(x, \omega_i, \omega_o)) \cdot T(x', x)\, dx' \qquad \textbf{Ec. 3.13}$$

donde la radiancia $q(x, \omega_i, \omega_o)$ es calculada por el producto del transporte de color $t(x)$ asignado a través de la función de transferencia, la cromaticidad de la luz dispersada entrante $c(x, \omega_o)$ y la luz atenuada $l_i(x, \omega_i)$ de la forma:

$$q(x, \omega_i, \omega_o) = t(x) \cdot c(x, \omega_o) \cdot l_i(x, \omega_i)$$

El transporte de color $t(x)$ al igual que la cromaticidad $c(x, \omega_o)$ son dependientes de la longitud de onda, mientras que el término $l_i(x, \omega_i)$ describe la luz incidente. La cromaticidad $c(x, \omega_o)$ es calculada desde la luz dispersada entrante de la forma:

$$c(x, \omega_o) = \int_{\Omega} \tau(x) \cdot p(x, \omega_i', \omega_o) \cdot c(x, \omega_i')\, d\omega_i'$$

donde Ω corresponde a la esfera unitaria centrada en el punto x. En este modelo, se considera como función de fase la función de Henyey-Greenstein explicada en la **Sec. 2.3** de este capítulo.

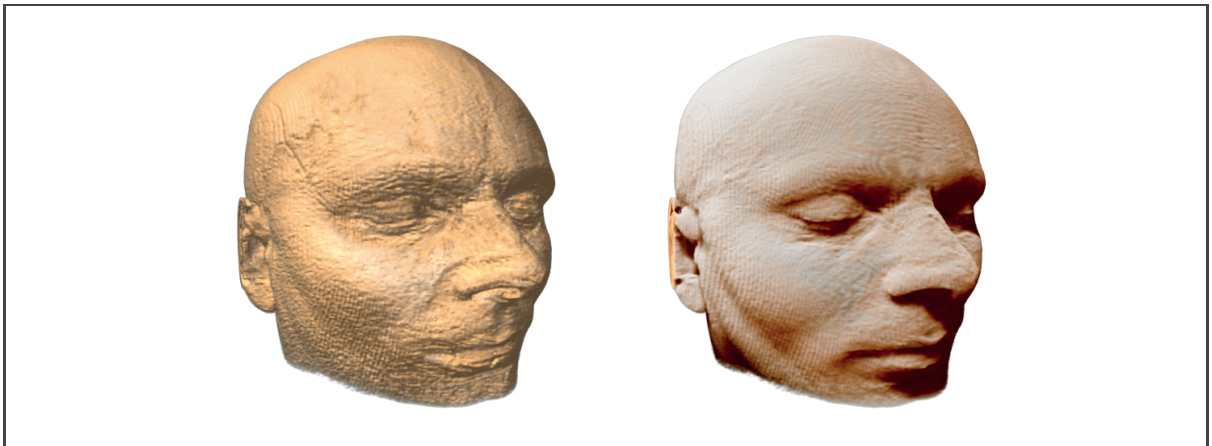

Figura 3.14 [3]: *Comparación en modelos de iluminación en volúmenes*. Comparación del modelo de iluminación por gradientes en despliegue directo de volúmenes y la técnica de dispersión de la luz y sombras en el modelo volumétrico de una cabeza humana.

CAPÍTULO IV – DETALLES DE IMPLEMENTACIÓN

En el siguiente capítulo se describen los pasos que se siguieron para la implementación de la técnica de *scattering* en ambos tipos de datos. El desarrollo del modelo de dispersión de la luz en modelos mallados está basado en la técnica propuesta por Dal Corso, Frisvad, Mosegaard y Bærentze **[2]**, mientras que la técnica en datos volumétricos se basa en el trabajo de Ropinski, Döring y Rezk-Salama **[3]**.

1 TÉCNICA DE DISPERSIÓN DE LA LUZ EN MALLAS POLIGONALES

En este capítulo, se introduce el método para simular materiales translúcidos en modelos mallados de manera eficiente utilizando el modelo de dipolo direccional. La idea general consiste en desplegar un objeto desde el punto de vista de la luz, y posteriormente, colocar una estructura de disco en la textura generada, almacenando un mapa de dispersión. Para ello, se utilizan cámaras con diversas direcciones apuntando al centro del objeto en la escena. Por último, se realiza una combinación de todos los mapas de dispersión generados para visualizar el resultado final de la técnica.

1.1 PASOS GENERALES DEL ALGORITMO

Se despliegan objetos translúcidos utilizando una función de distribución de reflectancia bidireccional de *subsurface scattering* (BSSRDF), la cual describe la relación entre la luz saliente e incidente, además de determinar cuánta es transportada entre dos rayos cualesquiera que colisionan una superficie.

1.1.1 BÚFER DE LUZ

En el primer paso, las posiciones, normales y profundidad del objeto en la escena son desplegadas en una textura desde el punto de vista de la luz (*ver **Fig. 4.1***), representada mediante un G-Buffer[28]. Cada luz direccional está asociada a una cámara ortográfica y un G-Buffer almacenado en un arreglo de texturas 2D. Estos últimos se calculan en una sola pasada.

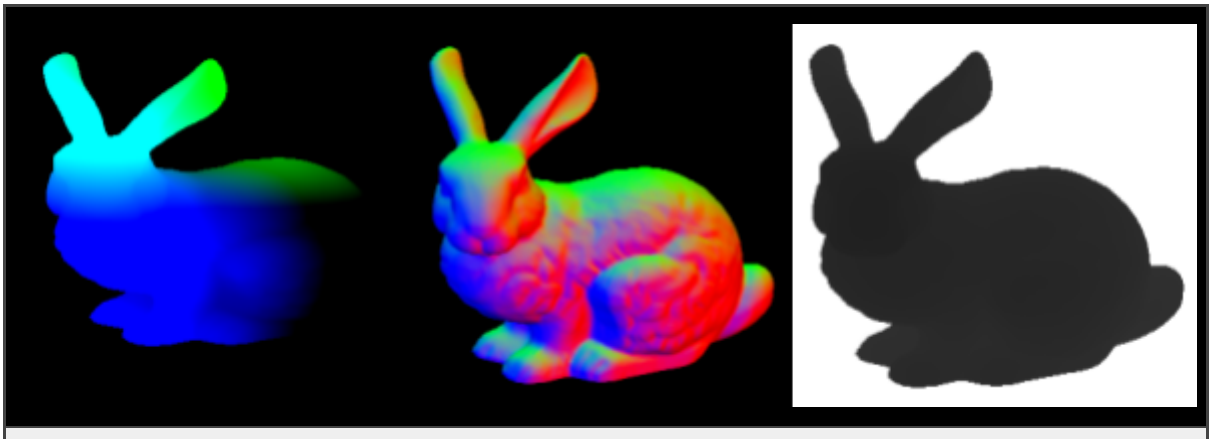

Figura 4.1: *Configuración de G-Buffer*. G-Buffer de posiciones, normales y profundidad luego de realizar el *rendering* desde una luz direccional. El modelo utilizado es el conejo de Stanford.

[28]**G-Buffer**: es el término colectivo que abarca a todas las texturas utilizadas para almacenar datos relevantes de la iluminación.

1.1.2 RENDERING A MAPA DE DISPERSIÓN

En el segundo paso, se despliega el objeto translúcido desde *K* direcciones en una textura denominada mapa de dispersión. El número de direcciones es elegido de modo que la superficie del modelo a representar esté lo más cubierta posible. El mapa de dispersión es organizado a través de una textura en capas, donde cada capa representa una dirección (*ver* ***Sec. 1.2.2***). Los puntos donde se ubican las cámaras ortográficas son elegidos mediante un algoritmo generador de números semi-aleatorios, y se distribuyen en una esfera (*ver* ***Sec. 1.3***). Luego, se configuran las cámaras para mirar hacia el centro de la *bounding box* del modelo. En este paso, para cada fragmento del objeto translúcido observado por una cámara, se calcula la dispersión generando *N* muestras a partir de la textura desplegada en el paso anterior. Si el punto a muestrear es válido, se utilizará para calcular la BSSRDF y se acumula el resultado en el mapa de dispersión como sigue:

$$R^k(x_o) = L_d \sum_{i=0}^{N} S\left(x_i^k, \omega_l, x_o, \omega_o\right) e^{(\sigma_{tr} r_i^k)},\ k\ \varepsilon\ [0, k-1] \qquad \textbf{Ec. 4.1}$$

Se introduce un parámetro k, que representa la dirección actual en la que se está desplegando la escena. Se puede observar cómo se calcula un punto de una de las direcciones consideradas en la ***Fig. 4.2***. También en este caso, las matrices de conversión de espacio de textura y de espacio de mundo se almacenan para ser reutilizadas en el paso 3, donde se combinan los resultados. Algorítmicamente, la **Ec. 4.1** se traduce en el producto de la luz difusa y la sumatoria de todas las muestras aleatorias generadas mediante la ecuación final del modelo BSSRDF.

Cabe destacar que representar la luz desde el punto de vista de la cámara tiene dos ventajas importantes:

- Si el disco es colocado en el espacio de textura, éste se orienta automáticamente hacia la dirección de la luz, es decir: $\omega_d = \omega_l$.

- La luz en la textura muestra sólo los puntos que son visibles desde la misma, donde éstos son los únicos que se encuentran iluminados. Además, si se muestrea el mapa de luz en cualquier punto, se obtiene el vértice correspondiente más cercano a la luz.

Estos dos factores permiten muestrear el punto y la dirección óptimos para ubicar el disco.

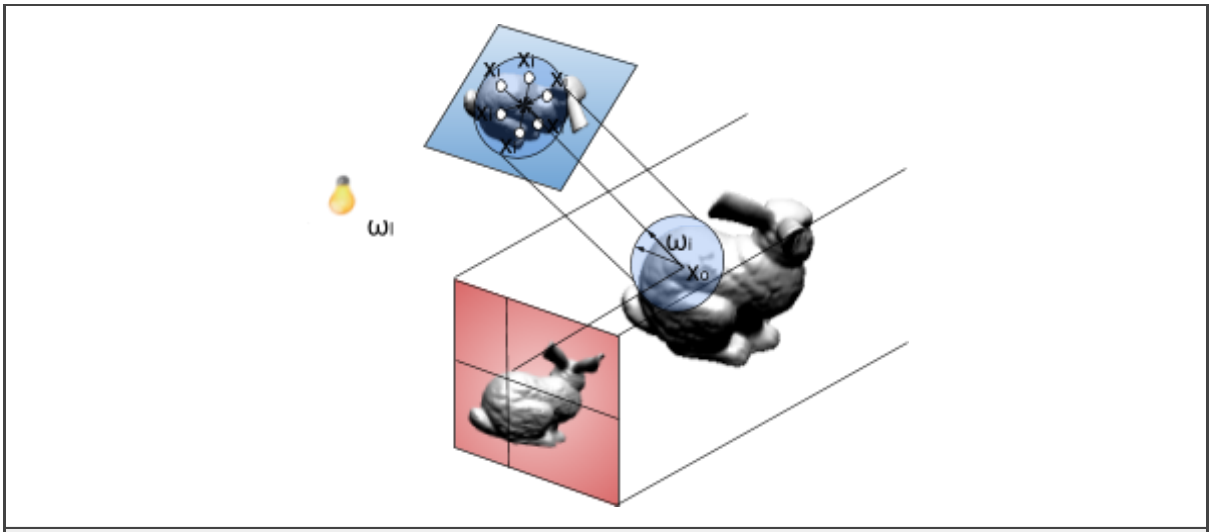

Figura 4.2: *Rendering al mapa de dispersión*. Cuando se despliega el punto x_o, la posición en el mapa de luz se calcula y los valores x_i en las muestras se calculan y se acumulan al resultado final.

1.1.3 COMBINACIÓN

En el tercer y último paso, utilizando las matrices y las texturas calculadas en el paso anterior, para cada fragmento de la superficie se tomarán muestras de los mapas de dispersión. Para realizar esto, se requiere muestrear el mapa de profundidad generado previamente, con la ayuda de una función de visibilidad que determina si un punto x de la capa k es visible (*ver* ***Fig. 4.3***). Esta función es calculada muestreando la capa k del mapa de profundidad utilizando un algoritmo similar al de mapeo de sombras (*ver* ***Sec. 1.4***). Dada esta función, se puede representar la luz saliente promediando la sumatoria de las K capas.

Posteriormente, se realiza una corrección *gamma* para lograr el resultado final. Dado un coeficiente γ, se realiza la corrección como sigue:

$$L^t_{gamma}(x, \omega_o, \gamma) = L^t(x, \omega_o)^{1/\gamma}$$

Finalmente, se envía la combinación de mapas de dispersión final al dispositivo de salida. El resultado obtenido se puede observar en la ***Fig. 4.4***.

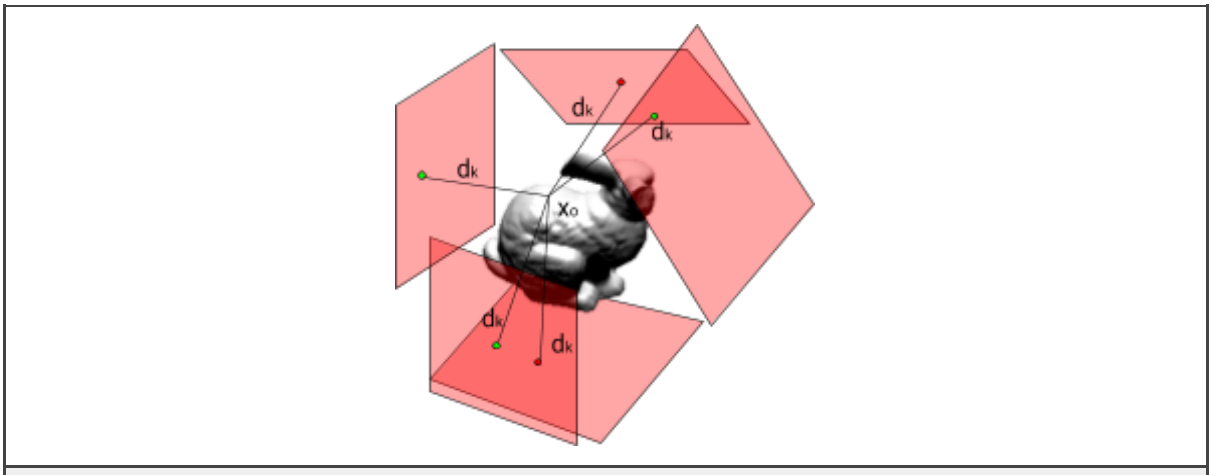

Figura 4.3: *Blending de mapas de dispersión*. Los círculos verdes y rojos representan puntos visibles y no visibles respectivamente al momento de combinar todos los mapas de dispersión.

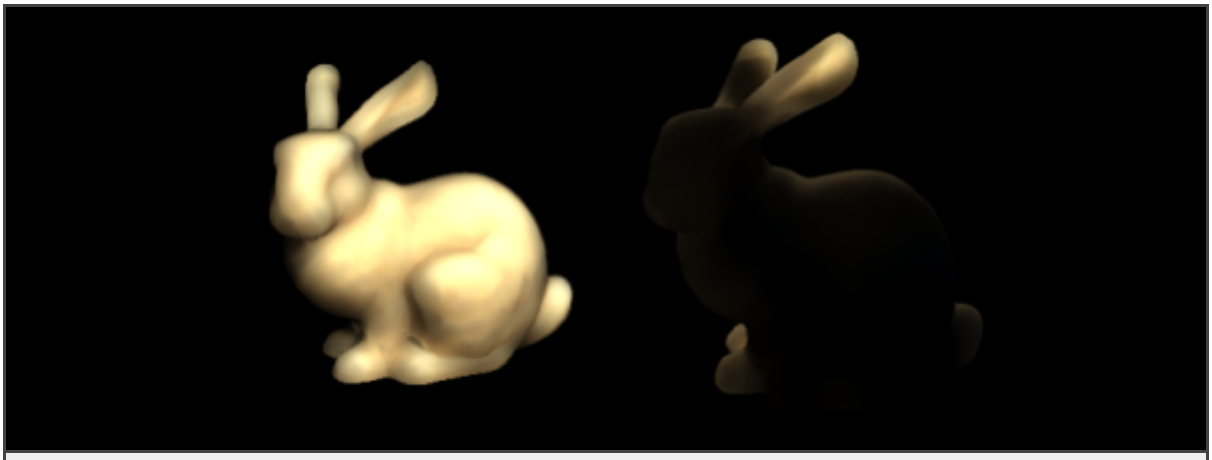

Figura 4.4: *Subsurface scattering*. Resultado final de la técnica aplicada al modelo del conejo de Stanford utilizando el material crema, considerando una luz direccional por delante (izquierda) y detrás (derecha) con 16 cámaras ortográficas, 48 muestras por fragmento y $\gamma = 1,25$ para aplicar la corrección *gamma*.

1.2 PASOS DE LA IMPLEMENTACIÓN

A continuación se explican en detalle el resumen y los pasos generales descritos en la sección anterior, divididos por tópicos utilizados en un área en particular del algoritmo implementado.

1.2.1 DESPLIEGUE EN TEXTURAS

En OpenGL, es posible redireccionar la salida del color del *fragment shader* a otra área de la memoria del GPU y re-utilizarla para cálculos futuros, lo cual permite crear técnicas de despliegue complejas. En el caso del desarrollo del trabajo especial de grado, se utilizará un *framebuffer object* (FBO) para mostrar la salida de la textura. Un FBO es una colección de objetos capaz de realizar un despliegue redireccionado, donde se pueden agregar texturas. Una textura puede ser vinculada a uno de los canales de color del *fragment shader* (*GL_COLOR_ATTACHMENT0*, *GL_COLOR_ATTACHMENT1*, *GL_COLOR_ATTACHMENT2,* ...), a la salida del búfer de profundidad (*GL_DEPTH_ATTACHMENT*), o a la salida del búfer de *stencil* (*GL_STENCIL_ATTACHMENT*) (*ver* ***Fig. 4.5***).

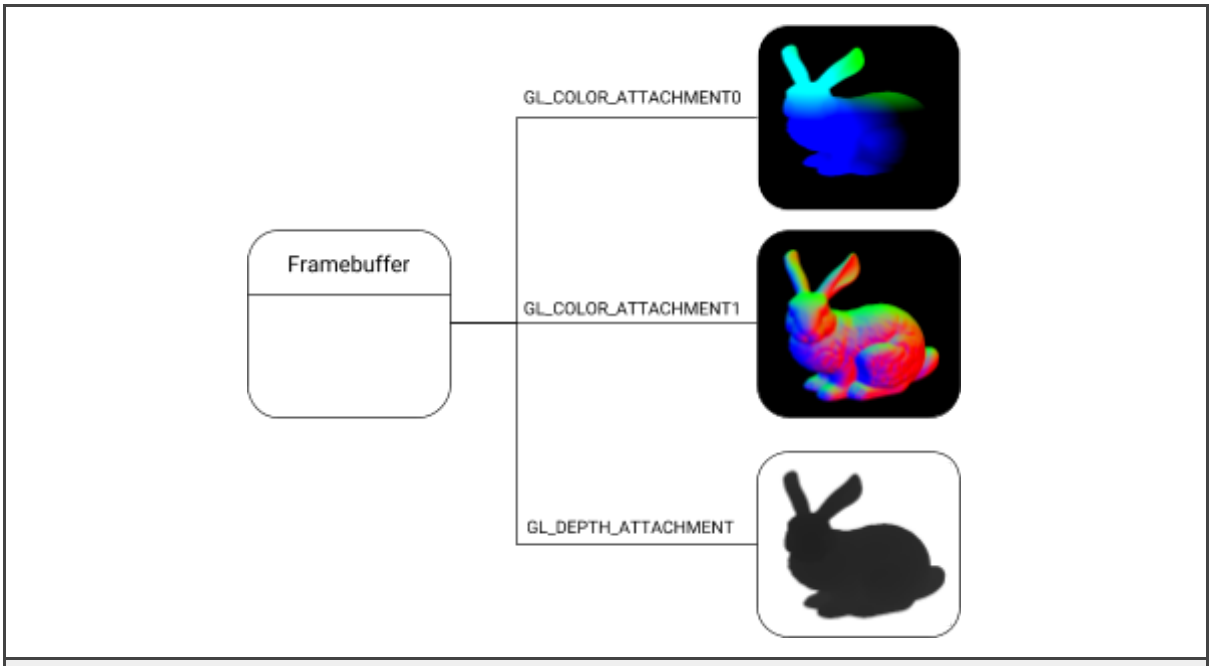

Figura 4.5: *Despliegue en texturas*. Se configura el FBO para almacenar la posición, normal y profundidad mediante los búferes *GL_COLOR_ATTACHMENT0*, *GL_COLOR_ATTACHMENT1 y GL_DEPTH_ATTACHMENT*.

1.2.2 DESPLIEGUE POR CAPAS

El despliegue por capas es una técnica que se utiliza para representar un tipo especial de textura llamada *layered texture*. Tomando como ejemplo el paso 2 del algoritmo implementado, donde se necesita representar el objeto desde k diferentes direcciones, se pueden crear k texturas 2D del tipo *GL_TEXTURE_2D*, y realizar k llamadas a dibujar vinculando cada vez la textura en el FBO actual. Sin embargo, OpenGL proporciona una forma de hacerlo con una única llamada, lo cual puede reducir costos de procesamiento.

En esta técnica, se usa el tipo de textura *GL_TEXTURE_2D_ARRAY* provisto por OpenGL, que se inicializa de la forma habitual, teniendo en cuenta que un arreglo de texturas (*array texture*) debe

asignarse como una textura 3D (*ver **Fig. 4.6***). Cuando éste se vincula a un FBO, se utiliza la llamada genérica *glBindFramebufferTexture*.

```
glBindTexture(GL_TEXTURE_2D_ARRAY, this->array_texture);
glTexParameteri(GL_TEXTURE_2D_ARRAY, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D_ARRAY, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D_ARRAY, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D_ARRAY, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexImage3D(GL_TEXTURE_2D_ARRAY, 0, GL_RGBA32F, g_width, g_height, layers, 0, GL_RGBA, GL_UNSIGNED_BYTE,
NULL);

glBindTexture(GL_TEXTURE_2D_ARRAY, this->depth_texture);
glTexParameteri(GL_TEXTURE_2D_ARRAY, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D_ARRAY, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D_ARRAY, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D_ARRAY, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexImage3D(GL_TEXTURE_2D_ARRAY, 0, GL_DEPTH_COMPONENT32, g_width, g_height, layers, 0,
GL_DEPTH_COMPONENT, GL_FLOAT, NULL);

glBindFramebuffer(GL_DRAW_FRAMEBUFFER, this->buffer);
glFramebufferTexture(GL_DRAW_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, this->array_texture, 0);
glFramebufferTexture(GL_DRAW_FRAMEBUFFER, GL_DEPTH_ATTACHMENT, this->depth_texture, 0);
glBindFramebuffer(GL_FRAMEBUFFER, 0);
```

Figura 4.6: *Inicialización de texturas del tipo GL_TEXTURE_2D_ARRAY*. La variable *layers* indica la cantidad de capas.

Para lograr este tipo de despliegue, se utiliza un *geometry shader*, donde la diferencia entre cada capa es básicamente una matriz de vista. Por lo tanto, en lugar de realizar los cálculos de la posición en el *vertex shader*, éstos serán realizados en el *geometry shader* (*ver **Fig. 4.7***).

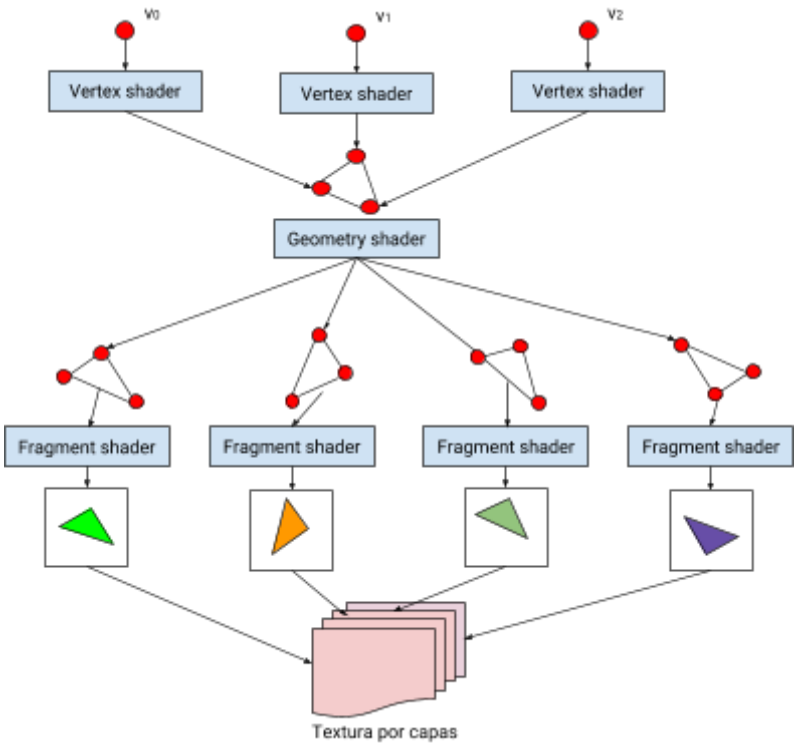

Figura 4.7: *Despliegue de textura por capas*. En el diagrama se muestra que cada vértice correspondiente a una cara del modelo es enviado al *geometry shader*, donde por cada dirección, este vértice es multiplicado por la matriz de vista respectiva a la capa y esta posición será enviada k veces al fragment shader (en el caso de la figura, $k = 4$).

1.2.3 GENERACIÓN DE PUNTOS UNIFORMEMENTE DISTRIBUIDOS

Para una correcta distribución de las muestras utilizadas en la técnica, se generan puntos uniformemente distribuidos en una esfera. Para realizar esto, se emplea una secuencia de números pseudo-aleatorios denominada puntos de Halton, descrita en detalle en **[2]**.

Inicialmente, dado un número primo p y un número entero positivo n, se puede expresar este último en base a p de la siguiente forma:

$$n = a_0 + a_1\, p + a_2\, p^2 + ... + a_r\, p^{\,p}$$

donde $a_i \in [0, p-1]$. Se define una secuencia Van der Corput $\Phi_p(n)$ como sigue:

$$\Phi_p(n) = \sum_{i=0}^{r} \frac{a_i}{p^{i+1}} = \frac{a_0}{p} + ... + \frac{a_r}{p^r}$$

Dado el hecho de que esta secuencia está basada en números primos, automáticamente asume buenas cualidades de aleatoriedad, además de que ya está normalizada en el rango [0, 1).

Se define un punto de Halton como la combinación de dos secuencias Van der Corput:

$$H_{p_1,p_2}(n) = \left(\Phi_{p_1}(n), \Phi_{p_2}(n)\right)$$

donde p_1 y p_2 son dos números primos, siendo $p_1 < p_2$. Usualmente, con (p_1, p_2) = (2, 3) se obtienen buenos resultados. Todos los puntos de Halton pertenecen al espacio de región [0, 1] x [0, 1]. Para obtener una muestra de los puntos de Halton en una esfera, se convierten utilizando la siguiente fórmula de coordenadas cartesianas a esféricas:

$$H_{p_1p_2}(n) = \left(\Phi_{p_1}(n), \Phi_{p_2}(n)\right) \rightarrow (s, t) \Rightarrow$$

$$\Rightarrow H^{esfera}_{p_1,p_2}(n) = \left(\sqrt{1 - (2t-1)^2}\, cos\,(2\Pi s),\, \sqrt{1 - (2t-1)^2}\, sen\,(2\Pi s),\; 2t-1 \right)$$

(a) PUNTOS EXPONENCIALMENTE SESGADOS

Para obtener un mejor muestreo, se necesita tener una buena distribución de puntos. Para ello, se emplea una técnica llamada muestreo de rechazo. La idea consiste en generar una secuencia de puntos de Halton, luego calcular su radio r y calcular su función de distribución de probabilidad utilizando un valor σ^*. Posteriormente, se utiliza el siguiente criterio de aceptación:

$$e^{-\sigma^* r} > \zeta$$

donde $\zeta \in [0,\ 1)$ es un número generado pseudo aleatoriamente. Se puede observar que si el punto está cerca del centro $(r \rightarrow 0)$, si $e^{-\sigma^* r} \approx 1$ entonces es más probable que que el punto sea aceptado. En otro caso, si el punto se encuentra lejos del centro del disco $(r \rightarrow \infty)$, si $e^{-\sigma^* r} \approx 0$ entonces es menos probable que el punto sea aceptado.

Se genera un número de muestras M, con un radio ajustable por el usuario. Luego, en el cálculo sólo se utilizan los N puntos más cercanos a x_o, lo cual permite ajustar el rendimiento y el resultado final. El valor del término σ^* está relacionado con σ_{tr} (el cual es un vector, ya que las tasas de transmisión son espectrales), y permite calcular las propiedades de dispersión en el material. σ^* se obtiene de la forma:

$$\sigma^* = min(\sigma_{tr,x}, \sigma_{tr,y}, \sigma_{tr,z})$$

1.2.4 MAPEO DE SOMBRAS

En esta técnica, se utiliza el algoritmo de mapeo de sombras (*shadow mapping*) para obtener los puntos de la textura generada en el paso 1, posteriormente calcular una función de visibilidad y muestrear los mapas de dispersión para generar la imagen final. El mapeo de sombras es una técnica comúnmente utilizada en el área de la computación gráfica, y pretende representar un objeto desde el punto de vista de la luz, para luego, al usar la información de profundidad generada por éste se decida si un punto está sombreado o no (*ver **Fig. 4.8***). En primer lugar, se convierte cada punto a espacio de textura, utilizando una matriz de conversión. Posteriormente, en un punto p=(p_x, p_y, p_z), se compara p_z con la textura T muestreada en el punto (p_x, p_y):

$$p_z > T(p_x, p_y)$$

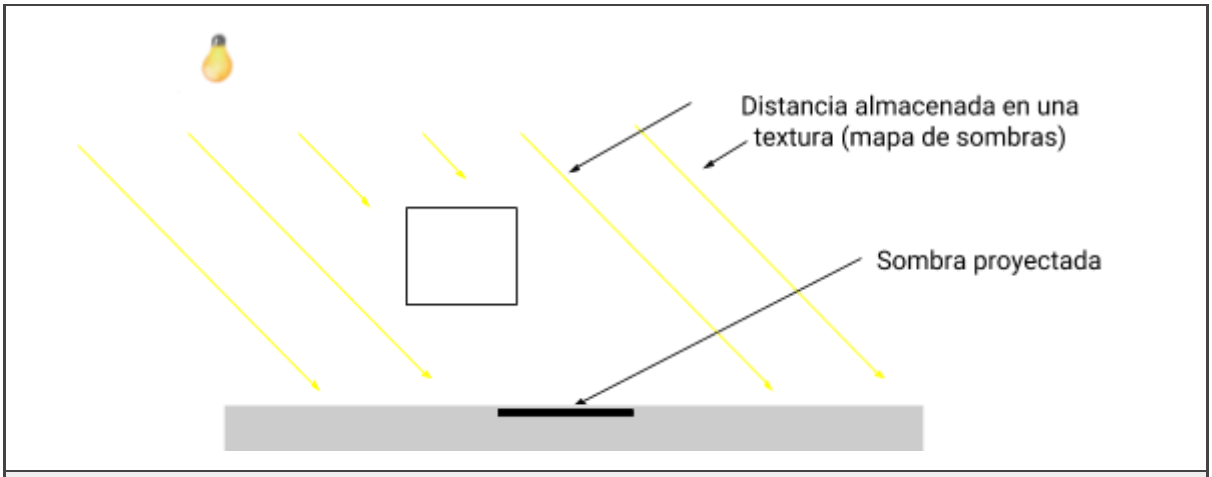

Figura 4.8: *Proyección de una sombra en un objeto iluminado*.

Si se cumple la condición anterior, significa que el punto actual no es visible desde el punto de vista de la luz y por ende, debe sombrearse. La matriz L utilizada para convertir un punto del espacio de mundo a espacio de textura es la siguiente:

$$L = T(\tfrac{1}{2}) \cdot S(\tfrac{1}{2}) \cdot P \cdot V$$

donde P y V son las matrices de proyección y de vista que se utilizan para desplegar la luz. Estas matrices son necesarias para realizar la conversión de coordenadas de *clipping* a coordenadas de textura; ya que las coordenadas de *clipping* están en el rango [-1, 1] × [-1, 1], mientras que las coordenadas de textura están en el rango [0, 1] × [0 , 1].

Como se mencionó anteriormente, en este algoritmo se utiliza la técnica de *shadow mapping* en dos ocasiones. La primera, ocurre en el paso 2 para obtener los puntos x_i de la textura generada en el paso 1, donde se utiliza la matriz L para convertir el punto del espacio de mundo x_o a x_d. En el paso de combinación final, también se utiliza el mapeo de sombras para calcular la función de visibilidad $V^k(x)$ y para muestrear el mapa de dispersión (*ver* ***Fig. 4.9***).

```
visibility = 1.0f;

for (int k = -1; k <= 1; k++) {
      for (int j = -1; j <= 1; j++) {
            texel_size = 1.0f / vec2(g_width, g_height);
            pcf_depth = texture(g_depth, vec3(offset.xy + vec2(k, j) * texel_size, i)).r;
            if (pcf_depth < offset.z - bias)
                  visibility -= 1.0f / 9.0f;
      }
}
```

Figura 4.9: *Función de visibilidad*. Algoritmo que determina la visibilidad de un fragmento en los mapas de dispersión por medio de un *kernel* 3x3 donde se utilizan las dimensiones de la textura para tomar muestras en el mapa de profundidad.

(a) BIAS DE SOMBRA

El mapeo de sombras tiene un inconveniente llamado acné de sombra. La discretización de la profundidad presenta un problema: dependiendo del ángulo de incidencia de la luz, la superficie comienza a generar un patrón alterno de oscuridad e iluminación en las áreas visibles. La solución en este caso consiste en introducir un factor constante al momento de comparar la posición del espacio de textura y la profundidad de un punto, llamado *bias* de sombra ε_b. Cuando se aplica este valor, todas las muestras obtienen una profundidad más pequeña que la de la superficie, y así, esta última estará iluminada correctamente. El *bias* de sombra es expresado de la siguiente manera:

$$p_z - \varepsilon_b \; < \; T(p_x, \, p_y)$$

2 TÉCNICA DE DISPERSIÓN DE LA LUZ EN DATOS VOLUMÉTRICOS

La implementación de esta técnica está basada en un volumen de luz. Alternativamente, se podría utilizar un enfoque de mapeo de sombras. Sin embargo, dado que los mapas de sombras permiten una impresión realista de las sombras y no requieren el almacenamiento de un conjunto de datos volumétricos, no permiten una integración fácil de la dispersión de la luz y tienen la desventaja de que pueden incorporar valores erróneos que dependen del usuario, como la especificación del valor del *bias*. En contraste, esta técnica no necesita ningún parámetro adicional asignado por el usuario.

2.1 PROPAGACIÓN DE LA LUZ

Con la intención de explotar las características de las GPUs actuales, se aproxima el modelo descrito procesando el conjunto de datos del volumen por rebanadas, es decir, capa por capa. Así, se calcula un volumen de luz el cual es actualizado dinámicamente cuando la función de transferencia y/o la posición de la luz x_l cambia. Dado que el sombreado y los efectos de *scattering* en cada posición x dependen sólo de aquellos vóxeles que se encuentran entre x y x_l, es suficiente procesar el conjunto de datos del volumen comenzando cerca de x_l y prosiguiendo en una dirección

ubicada de espaldas a la fuente de luz para calcular $c(x, \omega_o)$ y $l_i(x, \omega_i)$. Se empieza a realizar el recorrido desde la cara F_0 del cubo del volumen (siendo ésta la más cercana a la fuente de luz) y se propaga la información de la luz a través de uno de los ejes x, y o z máximos del volumen. Como se ilustra en la ***Fig. 4.10***, el eje utilizado se determina por un término denominado dir_{max}. La obtención de este último puede ser complicada debido a que se debe escoger la cara F_j de todas las caras del cubo F, la cual debe tener la mayor proyección vista desde el punto x_l.

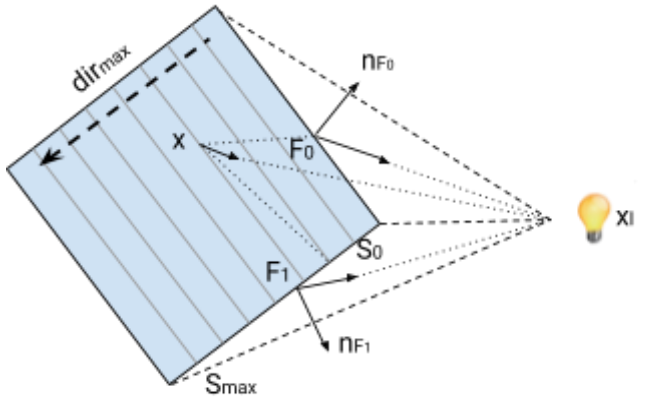

Figura 4.10: *Propagación de la luz*. La luz es propagada a través del volumen capa por capa, empezando desde la cara F_0 del cubo del volumen siendo ésta la más cercana a la fuente de luz posicionada en x_l.

Debido a que se desea propagar la información de la luz desde x_l a través del volumen, no todas las caras F_j van a ser consideradas; sólo se van a tomar en cuenta aquellas que son visibles desde la posición de la luz y que cumplen con la siguiente ecuación:

$$n_{F_j} \cdot (x_l - c_{F_j}) < 0$$

donde n_{F_j} es la normal de la cara F_j, y c_{F_j} corresponde al centro de la cara F_j. Basándose en esta observación, se ordenan las tres caras del cubo que son visibles desde x_l, tal que F_0 minimiza a la ecuación $n_{F_j} \cdot (x_l - c_{F_j})$ (*ver* ***Fig. 4.10***). Así, F_0 es la cara del cubo con la mayor proyección y F_1 es la cara del cubo que contiene la menor proyección vistas desde la posición de la luz. Dado que n_{F_0} es la normal de la cara F_0, el principio de la dirección de la propagación de la luz denotado como dir_{max} puede ser representado como $dir_{max} = -n_{F_0}$.

2.2 CÁLCULO DE VOLUMEN DE LUZ

Ahora que se conoce en qué dirección se va a propagar la información de la luz a través del volumen, se puede empezar a recorrer desde la capa S_0 la cual está más cerca del punto x_l a lo largo del eje dir_{max}. Durante la propagación de la luz se recorren todas las capas S_i hasta llegar a la última capa S_{max}. Esta propagación se realiza en tiempo real utilizando *framebuffer objects, d*ado que el *rendering* en una textura 3D sólo permite desplegar texturas a través del eje Z, se aplicará una función propia de GLSL denotada *imageStore* que permite almacenar una textura en la estructura de una imagen 3D.

Para cada capa excepto S_0, se requiere acceder a la capa calculada en el paso anterior. Así, en el *rendering* cada fragmento corresponde a un vóxel en el conjunto de datos original y se puede utilizar un programa de fragmentos para calcular la información de luz como se denota en la **Ec. 3.13**. Esto último se realiza basándose en el ángulo del cono descrito en la **Sec. 4** del capítulo anterior. Para calcular tanto el color de la dispersión como la intensidad de la sombra, se aplica una composición desde atrás hacia el frente para S_i y S_{i-1}, al igual que en la definición estándar de despliegue directo de volúmenes. Cuando se asume que el color del vóxel actual y la opacidad son almacenados como *voxel.rgb* y *voxel.a* en el *fragment shader*, el color de dispersión $c(x_i, \omega)$ y la intensidad atenuada $l_i(x_i, \omega)$ en x_i pueden ser calculadas en el mismo de la siguiente forma:

$$c(x_i, \omega) = (1.0 - voxel.a) \times c(x_{i-1}, \omega) + voxel.a \times voxel.rgb$$

$$l_i(x_i, \omega) = (1.0 - voxel.a) \times l_i(x_{i-1}, \omega) + voxel.a$$

donde $c(x_{i-1}, \omega)$ y $l_i(x_{i-1}, \omega)$ se calculan alrededor de la intersección p_i entre vector de luz del vóxel actual $dir_l = x_l - x_i$ con la capa anterior. En este paso, se desea simular el transporte de la luz desde la fuente a través del volumen, y cabe destacar que la dirección incidente $l_i(x_{i-1}, \omega)$ y la dirección saliente $c(x_i, \omega)$ son la misma. Por lo tanto, se puede aproximar el transporte de la luz considerando el modelo de emisión-absorción. Si se desean garantizar sombras duras, se puede calcular $l_i(x_{i-1}, \omega)$ realizando una única obtención de textura en p_i, y $c(x_{i-1}, \omega)$ se obtiene combinando vóxeles en los alrededores de p_i. La cantidad de vóxeles y los factores de la combinación dependen del ángulo del cono q.

Para incorporar la dispersión de la luz de saliente, se procede de manera similar en una segunda pasada. Pero en lugar de usar x_l para determinar F_0, se usa la posición actual de la cámara y se establece dir_{max} en n_{F_0} antes de propagar la luz de atrás hacia adelante. Durante esta propagación, se combina la cromaticidad saliente con la cromaticidad entrante calculada anteriormente.

La implementación de la propagación de la luz descrita previamente puede conllevar algunos problemas técnicos. Al aplicar el algoritmo, si x_l está ubicado dentro del volumen, la información de la luz no se puede propagar en una sola dirección. Para solucionar este problema se aplica lo siguiente: la información de la luz puede propagarse en ambas direcciones a partir de x_l, sin embargo, este enfoque puede conducir a discontinuidades notables dentro del volumen de luz y por lo tanto se proyecta x_l en F_0.

2.3 DIRECCIÓN ADICIONAL DE PROPAGACIÓN DE LA LUZ

La técnica de propagación de la luz descrita anteriormente, podría funcionar bien para posiciones fijas de la luz x_l, pero un cambio en la misma podría introducir problemas. Esto se debe al hecho de que el término dir_{max} siempre se encuentra a lo largo del eje máximo del volumen. Cuando x_l cambia de tal manera de que dir_{max} también, la luz es propagada a lo largo de otra dirección completamente distinta a través del volumen, y esto puede conllevar a efectos visuales no deseados.

Para resolver este problema, se introduce una dirección adicional de propagación de luz denotada como dir_{blend}. Mientras que dir_{max} consiste en la dirección de propagación de la luz

basada en la cara del cubo F_0 con la mayor proyección vista desde la posición x_l, el término $dir_{blend} = -n_{F_1}$ está basado en F_1, la cara del cubo con la segunda mayor proyección (*ver **Fig. 4.11***). Cuando se realiza el proceso de combinación de propagación de la luz a lo largo de estos dos ejes principales, se pueden evitar efectos no deseados, debido a que la contribución de la posiblemente nueva dirección ya ha sido tomada en cuenta antes de que la dirección de la propagación de la luz cambie. Seguidamente, se procesa el volumen a través de dir_{blend} y se calculan los efectos de dispersión y sombras a lo largo de este eje.

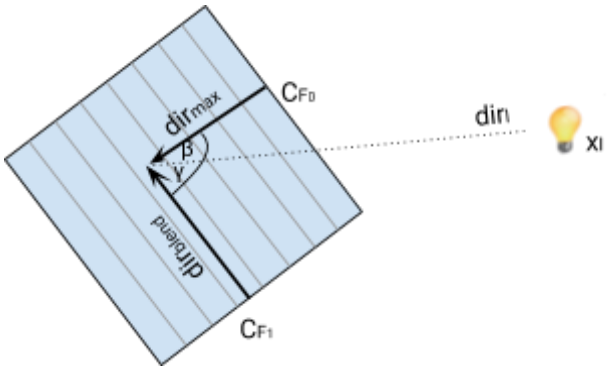

Figura 4.11: *Dirección adicional en propagación de la luz*. La luz es propagada a través del volumen capa por capa, empezando desde la cara F_0 del cubo del volumen siendo ésta la más cercana a la fuente de luz posicionada en x_l.

2.4 DETALLES ADICIONALES DE DESPLIEGUE

A continuación, se asume que el volumen de luz es considerado como un conjunto de datos RGBA, donde los canales RGB representan la cromaticidad $c(x,\omega)$ y el canal *alpha* representa la intensidad de la luz atenuada $l_i(x,\omega)$, ambos términos en la posición del vóxel x. Basándose en los valores de los conjuntos de datos volumétricos a utilizar, se resuelve la integral de *volumen rendering* recorriendo los volúmenes del frente hacia atrás. Así, el término x consiste en la multiplicación del color de transporte $t(x)$ asignado a través de la función de transferencia, y $c(x,\omega)$, al cual se le aplica una adaptación de intensidad basada en $l_i(x,\omega)$. Para soportar efectos de emisión, se puede agregar el color emisivo $L_e(x)$.

A pesar de que esta técnica de despliegue puede generar resultados realistas para materiales difusos y dispersores, no es capaz de representar materiales que tienen un alto grado de especularidad. Sin embargo, la técnica puede extenderse añadiendo una intensidad especular $c_s(x)$ que es calculada utilizando el modelo basado en gradientes de *Phong*. Con este término, las superficies pueden lucir muy realistas ya que presentan elementos difusos y contienen un componente especular. Para lograr la intensidad deseada, es posible multiplicar los términos $c_s(x)$ y $l_i(x,\omega)$ de la siguiente forma:

$$I = |\nabla\tau(x)| \cdot c_s(x) \cdot l_i(x,\omega)$$

CAPÍTULO V – RESULTADOS

Una vez terminada la etapa de diseño e implementación de la técnica de *scattering* en volúmenes y mallados, es necesario poner a prueba la aplicación para evaluar el rendimiento, la eficiencia y precisión de la misma, y así determinar si se cumplieron o no los objetivos anteriormente establecidos. En este capítulo se presentan las pruebas realizadas y los resultados obtenidos.

1 CONJUNTOS DE DATOS

A continuación se especifican los modelos utilizados para cada una de las pruebas realizadas a la aplicación.

1.1 MODELOS MALLADOS

Para las pruebas de *scattering* en mallados se utilizaron cinco modelos: tres mallados tomados del repositorio de la Universidad de Stanford (*Stanford Bunny* **[78]**, *Stanford Buddha* **[78]** y *Stanford Dragon* **[78]**), el modelo de la estatua de Hebe **[79]** y una esfera estándar **[80]** (*ver* ***Fig. 5.1***). Las siguientes imágenes fueron generadas utilizando el modelo de iluminación *Blinn-Phong*, con un *viewport* de 1200 × 680. Además se muestran las especificaciones de cada uno de estos mallados (*ver* ***Tab. 5.1***).

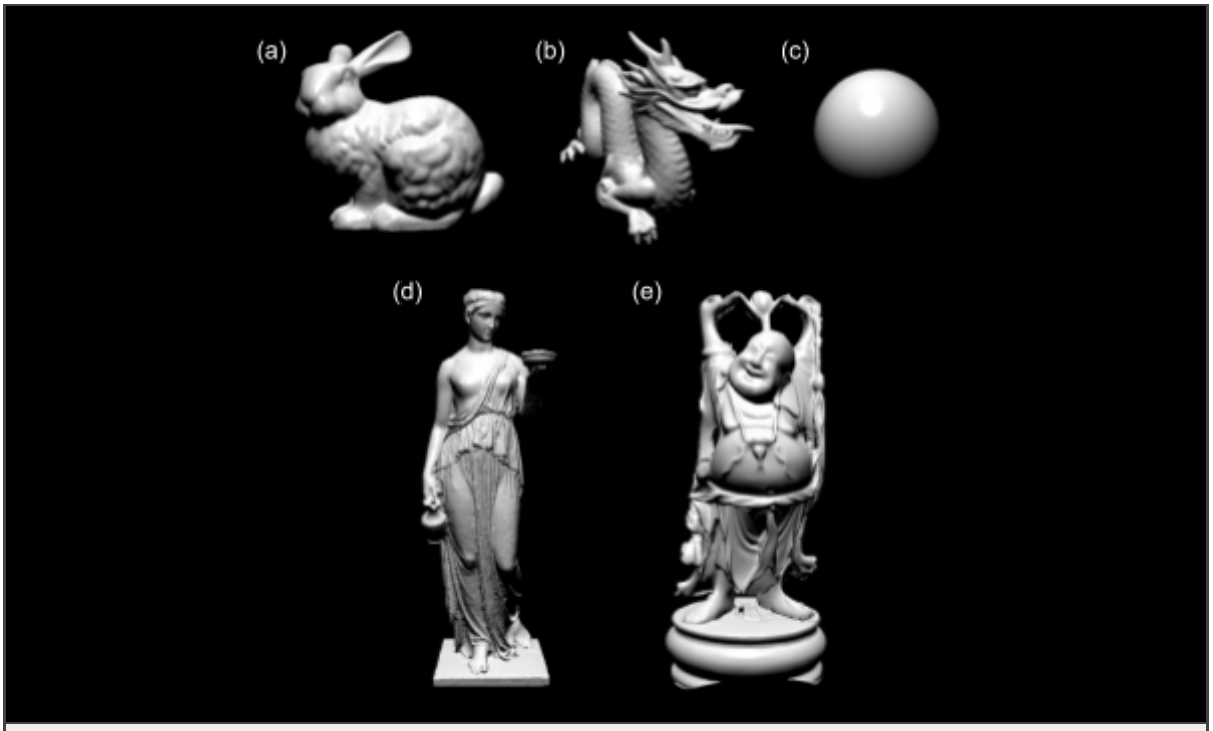

Figura 5.1: *Modelos mallados.* (a) *Stanford Bunny*, (b) *Stanford Dragon*, (c) Esfera estándar, (d) Estatua de Hebe, (e) *Stanford Buddha*.

Modelo	Número de vértices	Número de caras
Stanford Bunny	34817	69630
Stanford Dragon	50000	100000
Estatua de Hebe	206173	412346
Stanford Buddha	49990	100000
Esfera estándar	382	800

Tab. 5.1: Especificaciones de los modelos mallados.

1.2 DATOS VOLUMÉTRICOS

Para las pruebas de *scattering* en volúmenes se utilizaron tres volúmenes: Una molécula de futboleno (*Bucky*), el modelo de un árbol bonsai y una pieza de motor **[81]** (*ver **Fig. 5.2***). Las siguientes imágenes fueron generadas con el algoritmo de *ray casting* de una pasada, con un *viewport* de 1200 × 680. Además se muestran las especificaciones de cada uno de estos volúmenes (*ver **Tab. 5.2***).

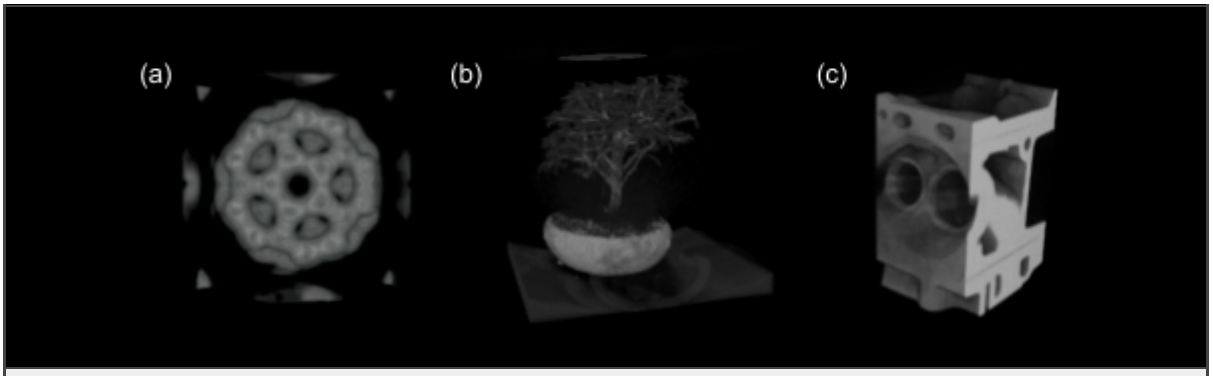

Figura 5.2: *Modelos volumétricos.* (a) *Bucky*, (b) Bonsai, (c) Pieza de motor.

Modelo	Dimensiones	Tamaño en memoria (KB)
Molécula de futboleno	32 × 32 × 32	32
Árbol bonsai	256 × 256 × 256	16000
Pieza de motor	256 × 256 × 256	16000

Tab. 5.2: Especificaciones de los modelos volumétricos.

2 RESULTADOS CUANTITATIVOS

En esta sección se muestran una serie de mediciones de tiempo de procesamiento para los tres volúmenes y los cinco modelos mallados vistos anteriormente. Cada prueba se ejecutó ocho veces para obtener resultados más precisos. Como se mencionó en el ***Cap. I***, el entorno de *hardware* consta de las siguientes características: 12.0 GB de memoria RAM, procesador Intel Core i7-3770 con 3.40 GHz, tarjeta gráfica EVGA GeForce GTX 660 con una memoria de video dedicada de 2.0 GB.

2.1 MODELOS MALLADOS

A continuación se detalla para cada uno de los modelos mallados el tiempo en milisegundos para generar los mapas de dispersión y combinarlos en tiempo real (Cambios en la escena), la combinación de los mapas previamente calculados (Escena fija), y el cálculo del modelo de iluminación *Blinn-Phong*. El material utilizado fue crema y los parámetros asignados para realizar estas pruebas fueron los siguientes: parámetro de asimetría $g = -1.00$, radio de dispersión $r = 0.025$, número de muestras a tomar $M = 12$ (4 cámaras), $M = 24$ (8 cámaras), $M = 48$ (16 cámaras), tamaño de las texturas para los mapas de luz y dispersión $W = 1200 \times 680$.

	N° de cámaras	Cambios en la escena (ms)	Escena fija (ms)	*Blinn-Phong* (ms)
Stanford Bunny	4	117,80	0,90	0,87
	8	246,60	1,36	
	16	543,48	2,39	
Stanford Dragon	4	135,69	1,03	0,93
	8	266,66	1,98	
	16	584,79	3,28	
Estatua de Hebe	4	157,23	1,44	0,92
	8	320,51	1,59	
	16	729,92	2,41	
Stanford Buddha	4	102,77	1,44	1,08
	8	206,61	1,64	
	16	444,44	2,14	
Esfera estándar	4	24,81	0,89	0,79
	8	50,13	0,97	
	16	102,04	1,07	

Tab. 5.3: Tiempos promedio para desplegar los modelos mallados.

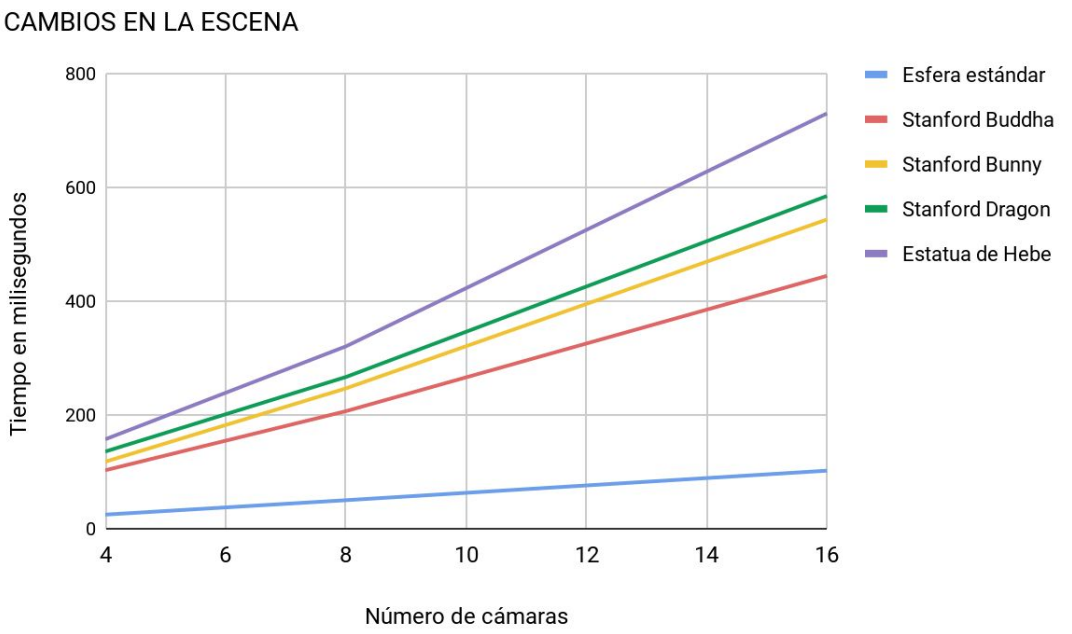

Figura 5.3: *Gráfico de líneas para modelos mallados al realizar cambios en la escena.*

Como se puede observar en la **Tab. 5.3** y la **Fig. 5.3**, se obtienen resultados de tiempo superiores en el *rendering* de los modelos con más caras, a excepción del despliegue del *Stanford Buddha*, ya que este factor implica un mayor cálculo en GPU. A su vez, cuando la escena tiene más cámaras se obtiene un tiempo mayor al realizar cambios en la misma, resultando en un comportamiento aproximadamente lineal.

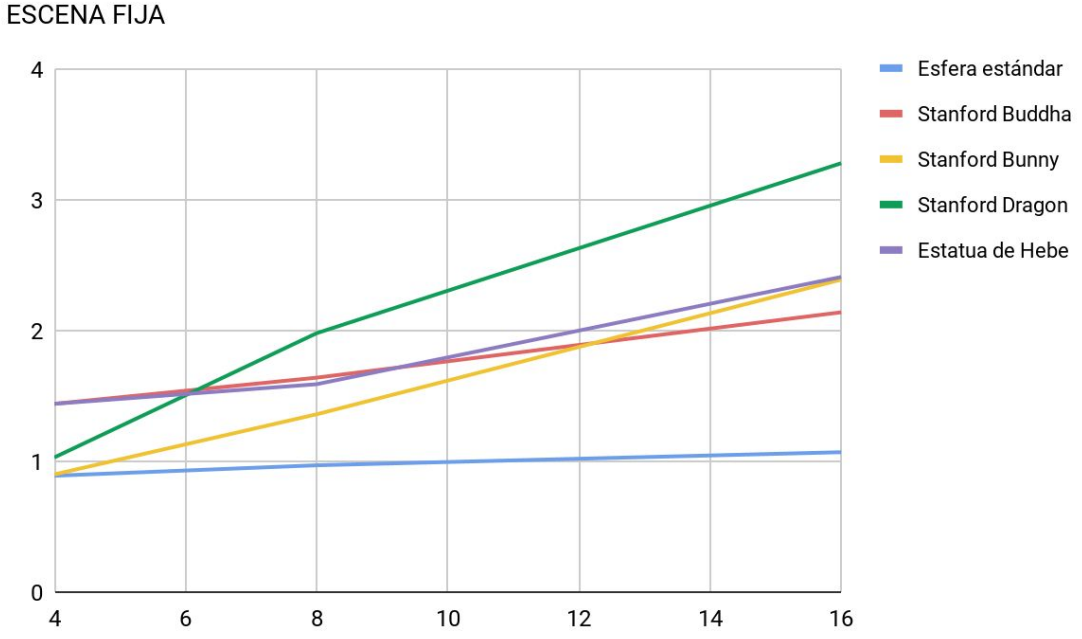

Figura 5.4: *Gráfico de líneas para modelos mallados al mantener la escena fija.*

Debido a que la técnica implementada es independiente de la vista, los resultados de tiempo de despliegue de la escena final utilizando la mezcla de los mapas de dispersión (Escena fija) son bastante bajos y no tienen una diferencia notable si se agregan más cámaras ortográficas, resultando en un comportamiento aproximadamente lineal (*ver* ***Fig. 5.4*** *y* ***Tab. 5.3***). Esta mezcla presenta resultados similares de tiempo comparado al despliegue aplicando el modelo de iluminación local *Blinn-Phong* (*ver* ***Tab. 5.3***).

2.2 MODELOS VOLUMÉTRICOS

A continuación se detalla para cada uno de los modelos volumétricos el tiempo en milisegundos para generar el volumen de luz y desplegar el mismo mediante el algoritmo de *ray casting* (Cambios en la escena), la técnica de *scattering* implementada con el volumen de luz ya generado (Escena fija), desplegar el volumen con el modelo de emisión-absorción, y el cálculo del modelo de iluminación por gradientes adicionado a los anteriores. Todos los volúmenes utilizan 8 muestras por fragmento y el tamaño del volumen de luz corresponderá al tamaño del volumen original.

	Cambios en la escena (ms)	Escena fija (ms)	Emisión-Absorción (ms)	Emisión-Absorción + Gradiente (ms)	Escena fija + Gradiente (ms)
Molécula de futboleno	6,04	1,41	1,31	1,61	1,73
Bonsai	34,99	8,86	3,05	9,49	16,47
Pieza de Motor	34,79	7,98	2,81	8,88	14,55

Tab. 5.4: Tiempos promedio para desplegar los modelos volumétricos.

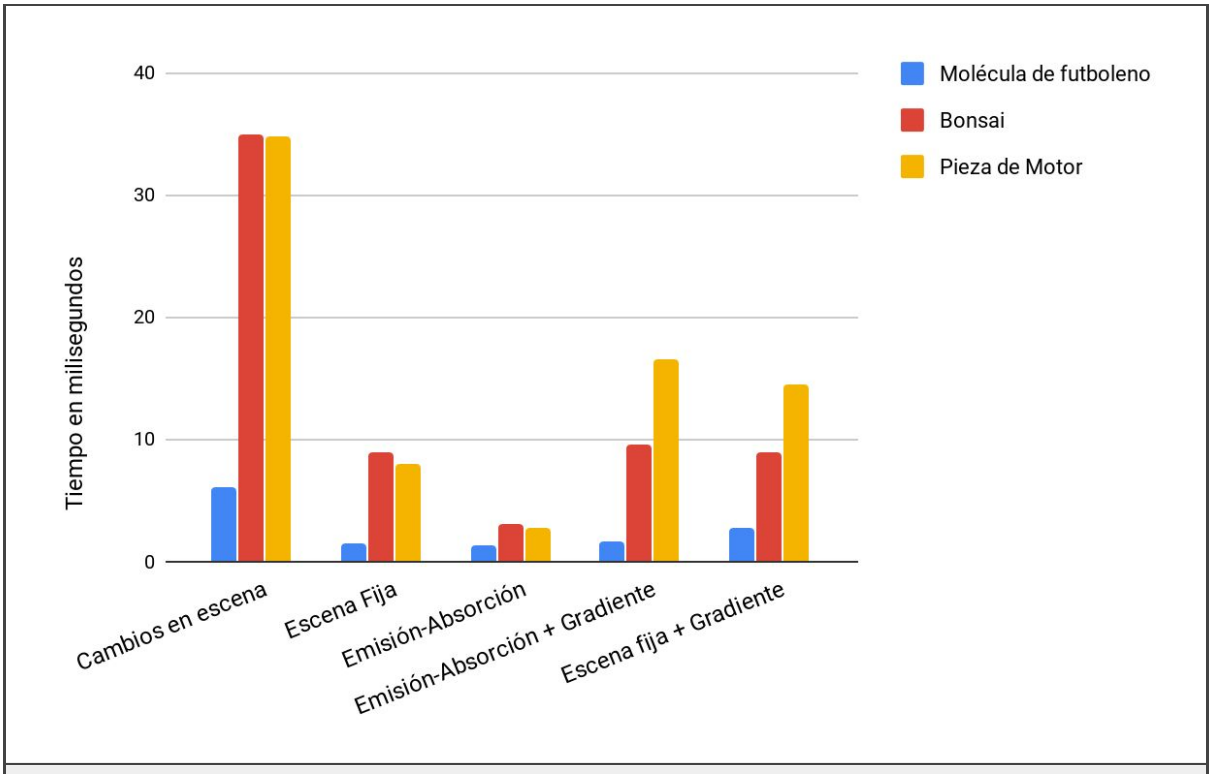

Figura 5.5: *Gráfico de barras para modelos volumétricos.*

Como se puede observar en la **Tab. 5.4** y la **Fig. 5.4**, se obtienen resultados de tiempo superiores en el *rendering* de los modelos con mayor número de vóxeles, ya que al igual que en el caso de los modelos mallados, este factor implica la realización de más cálculos en GPU, esto ocurre también para la generación del volumen de luz. Debido a que esta técnica es independiente de la vista, los resultados para el despliegue utilizando el volumen de luz (Escena fija) son menores a los de la generación del mismo (Cambios en la escena). Sin embargo, los tiempos para desplegar el volumen aplicando el modelo de emisión-absorción clásico, son menores. El agregado de iluminación por gradientes en la técnica implementada y en el modelo de emisión-absorción, presenta tiempos superiores con respecto a los obtenidos en la escena fija, debido a que se debe calcular un vector gradiente por cada muestra del volumen.

2.3 ESCENA HÍBRIDA

A continuación se tiene una escena híbrida donde se van a mostrar un modelo mallado y un modelo volumétrico a la vez dentro de la caja de Cornell. En las pruebas realizadas fueron utilizados los modelos *Bucky* y Bonsai (volúmenes), Estatua de Hebe y *Stanford Dragon* (mallados). Las características del *scattering* en modelos mallados son las mismas presentadas en la ***Sec. 2.1***, y en el caso de los modelos volumétricos se tienen las mismas características presentadas en la ***Sec. 2.2***. La siguiente tabla detalla el tiempo en milisegundos para el *rendering* de la imagen final donde se han aplicado las técnicas presentadas anteriormente.

	Rendering de la escena (ms)
Molécula de futboleno y Estatua de Hebe	12,63
Molécula de futboleno y *Stanford Dragon*	5,73
Bonsai y Estatua de Hebe	13,62
Bonsai y *Stanford Dragon*	5,07

Tab. 5.5: Tiempos promedio para el *rendering* final de ambos tipos de dato.

En la **Tab. 5.5** se indica que el *rendering* de los volúmenes y mallados con mayor dimensión implican un tiempo superior, ya que se realiza un mayor cálculo al momento de aplicar las técnicas implementadas, como ya se ha explicado anteriormente. A pesar de esto, los tiempos obtenidos desplegando ambos tipos de dato son bajos y la escena puede ser recorrida con fluidez.

3 RESULTADOS CUALITATIVOS

A continuación se realizaron una serie de mediciones para determinar el cambio de calidad perceptible por el ojo humano en la imagen final en los tres volúmenes y los cinco modelos mallados presentados anteriormente, realizando una comparación con otras técnicas de iluminación clásicas como *Blinn-Phong* para el despliegue de mallados y el modelo de emisión-absorción para el despliegue de volúmenes.

3.1 MODELOS MALLADOS

En las pruebas realizadas se puede observar cómo a medida que se aumenta número de cámaras ortográficas, el modelo se torna más suave y se cubre toda su extensión de tal forma que no exista ningún punto que no sea visible. Si el modelo utilizado tiene más partes convexas que cóncavas no es necesario utilizar un número de cámaras muy alto para lograr cubrirlo en su totalidad, pero aun así su calidad visual mejora si este número es mayor, ya que se mezclan más mapas de

dispersión y el resultado es más acorde a los diferentes puntos de vista que elija el usuario con la cámara interactiva.

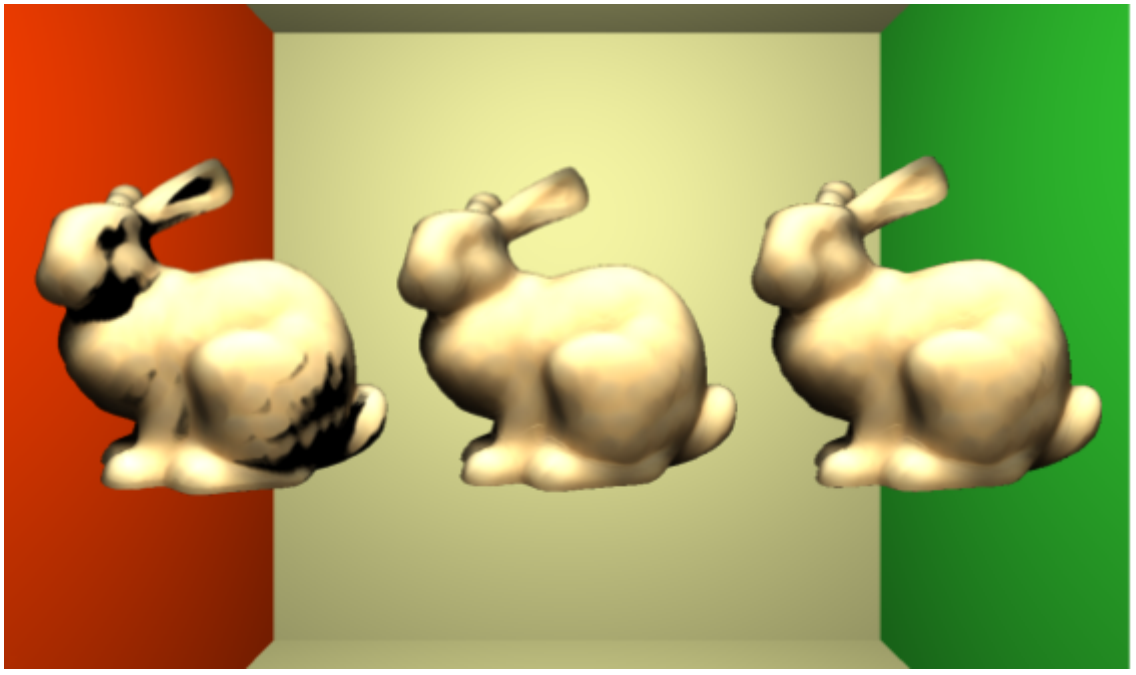

Figura 5.6: *Stanford Bunny hecho de crema*. Utilizando 4 (izquierda), 8 (centro) y 16 (derecha) cámaras.

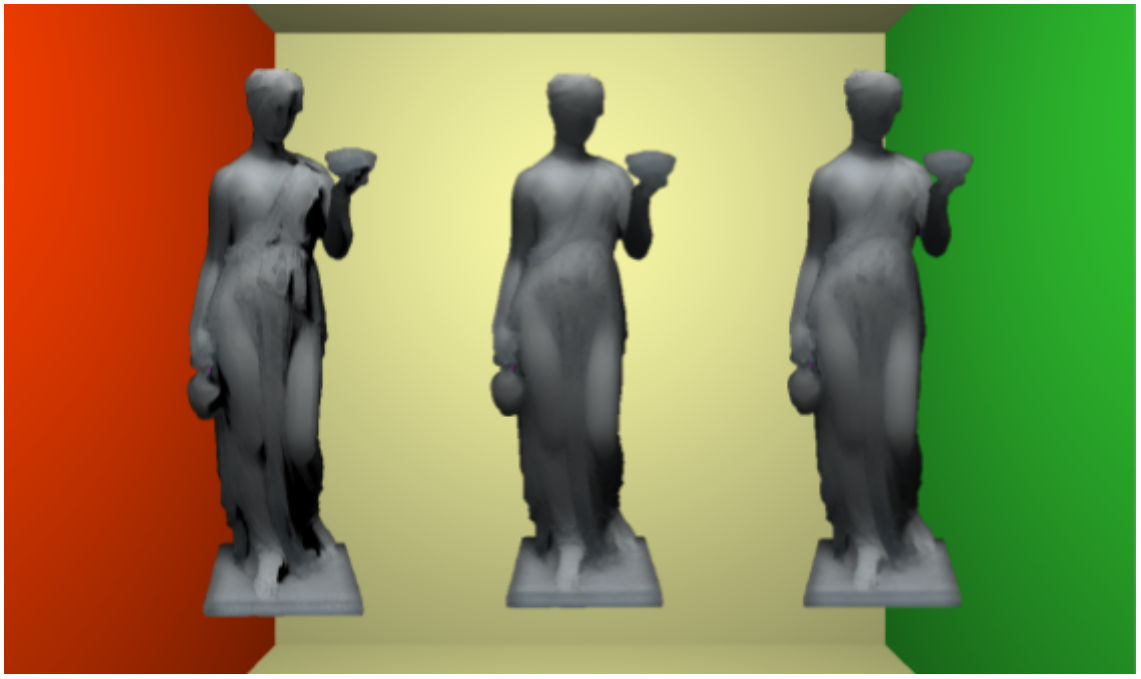

Figura 5.7: *Estatua de Hebe hecha de mármol*. Utilizando 4 (izquierda), 8 (centro) y 16 (derecha) cámaras.

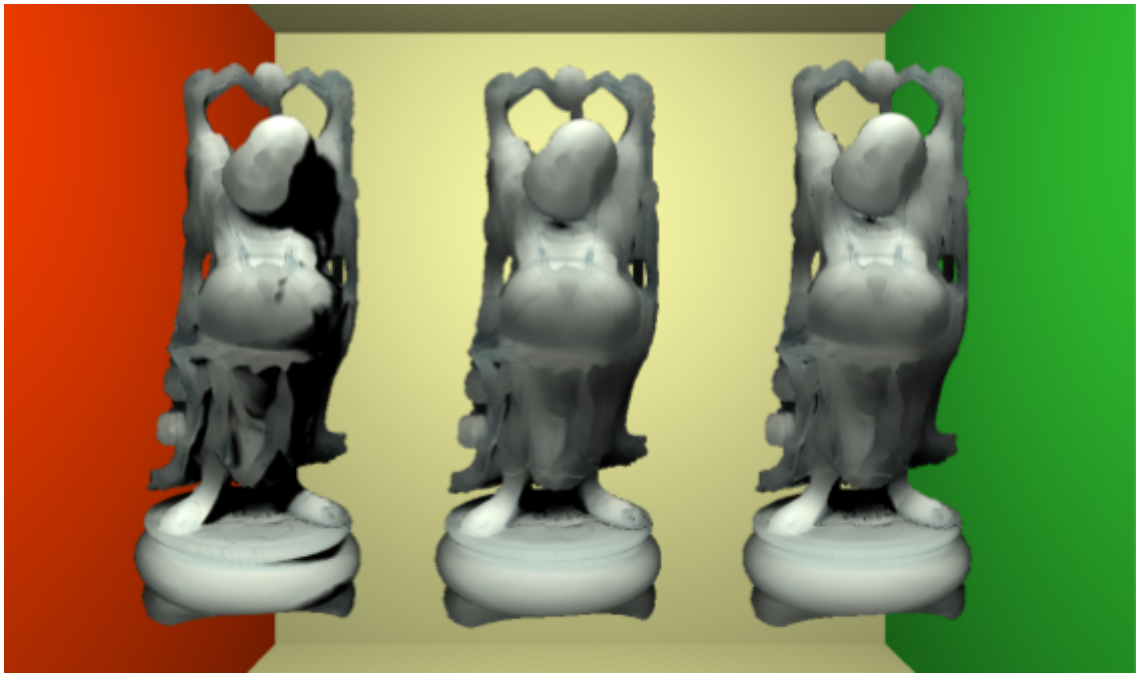

Figura 5.8: *Stanford Buddha hecho de leche*. Utilizando 4 (izquierda), 8 (centro) y 16 (derecha) cámaras.

Figura 5.9: *Stanford Dragon hecho de piel*. Utilizando 4 (izquierda), 8 (centro) y 16 (derecha) cámaras.

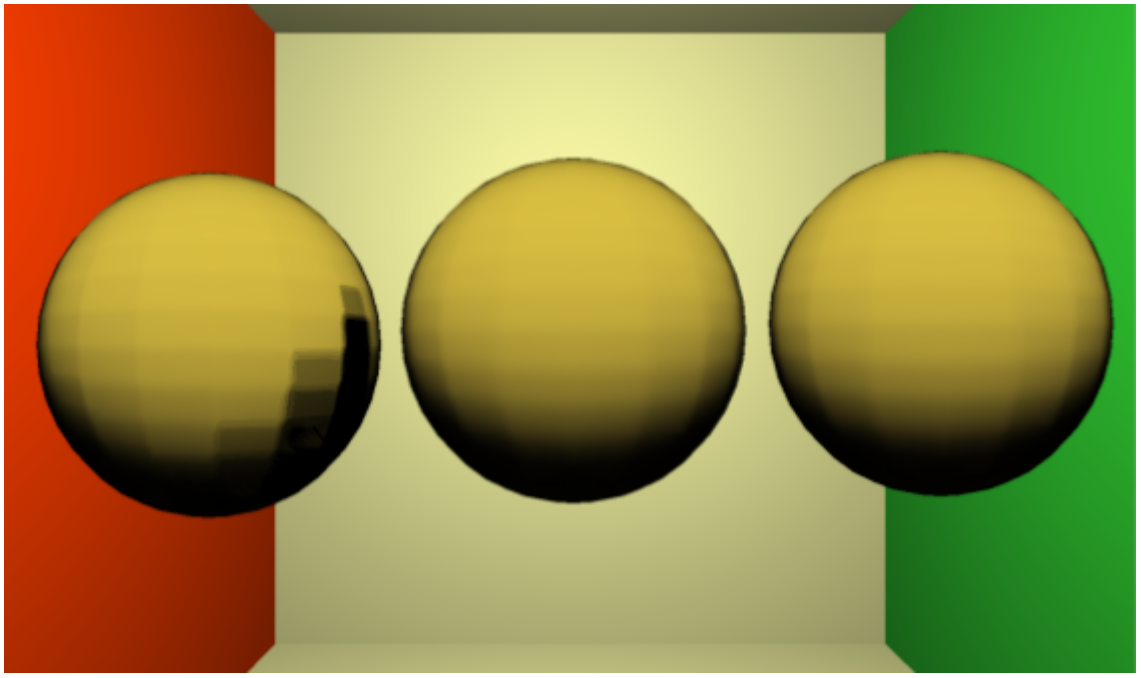

Figura 5.10: *Esfera estándar hecha de patata*. Utilizando 4 (izquierda), 8 (centro) y 16 (derecha) cámaras.

Para un resultado más completo, la aplicación tiene la posibilidad de activar el componente especular y agregarlo al resultado del *scattering*. Este componente especular está basado en los cálculos de la reflexión especular presentada en el modelo de iluminación *Blinn-Phong*.

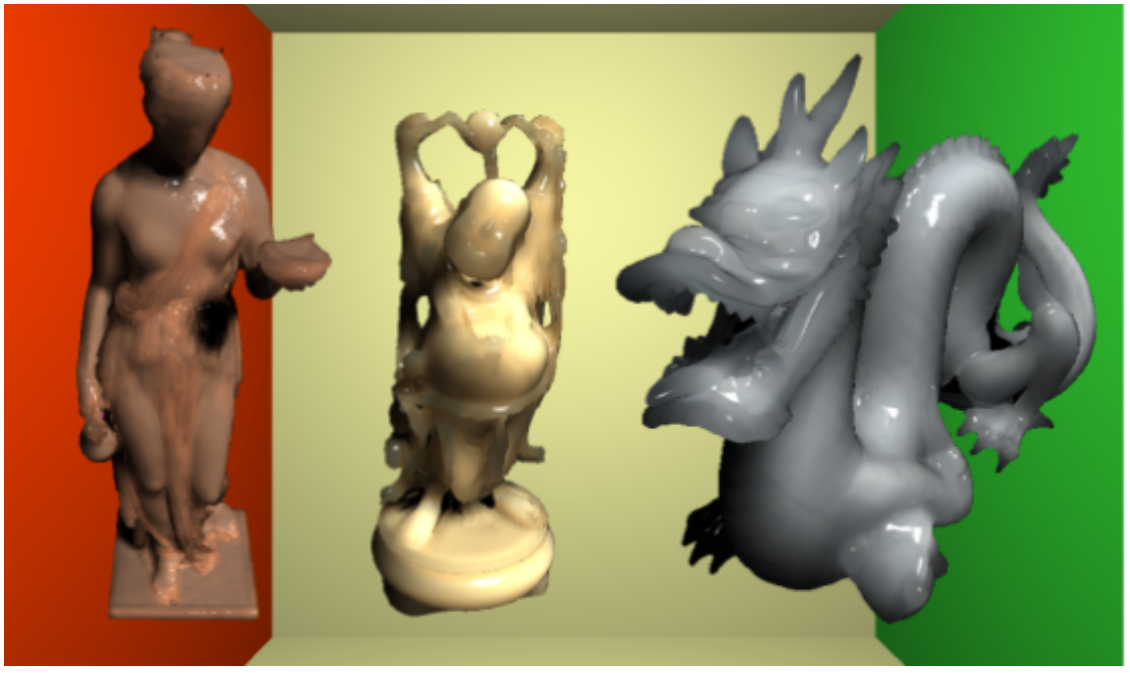

Figura 5.11: *Extensión de scattering*. Estatua de Hebe hecha de piel, *Stanford Buddha* hecho de crema y *Stanford Dragon* hecho de mármol utilizando 16 cámaras ortográficas y aplicando el componente especular.

La técnica implementada puede ser comparada con el modelo clásico de iluminación *Blinn-Phong* para observar los cambios visuales que se presentan. Para la técnica de *scattering* se pueden notar efectos de sombras y translucidez en el modelo mallado (*ver* ***Fig. 5.9***).

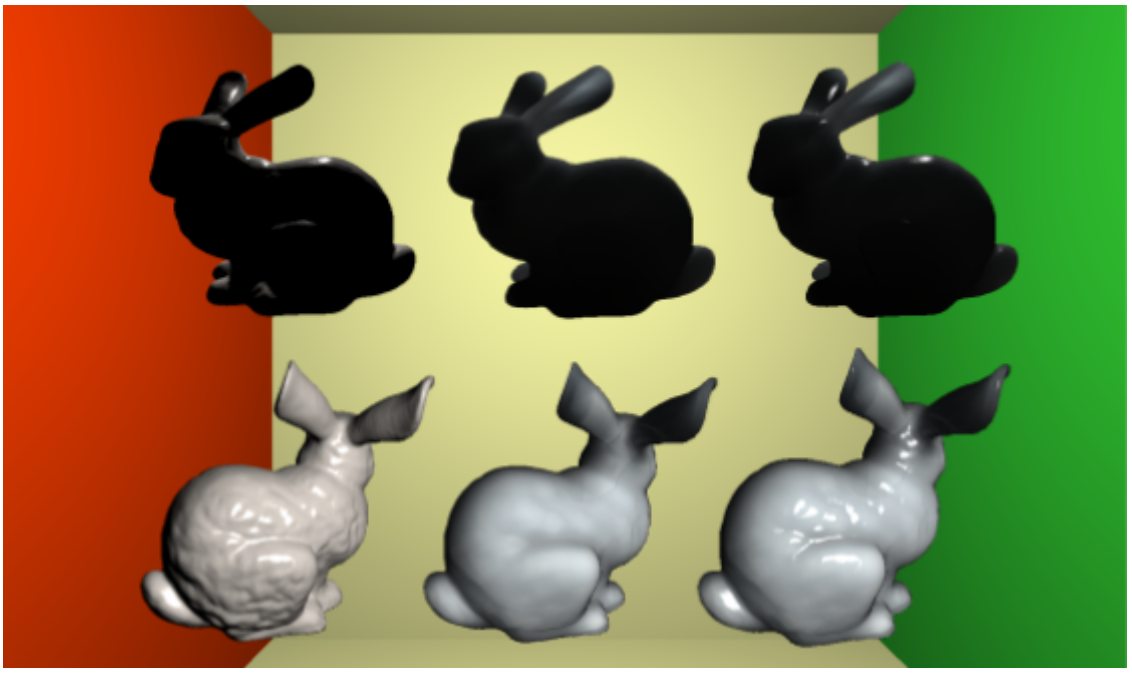

Figura 5.12: *Comparación de modelos de iluminación*. *Stanford Bunny* hecho de mármol aplicando el modelo de iluminación *Blinn-Phong* (izquierda), la técnica de *scattering* implementada (centro) y con el componente especular adicionado (derecha). En la primera fila, la luz en la escena se encuentra detrás del modelo, mientras que en la segunda fila la luz se encuentra delante.

Como se puede observar en la figura anterior, la técnica implementada produce resultados más realistas agregándole efectos de sombras y translucidez al modelo, los cuales no son provistos al utilizar el modelo de iluminación *Blinn-Phong*.

3.2 MODELOS VOLUMÉTRICOS

En las pruebas realizadas se puede observar cómo los volúmenes que poseen la técnica de *scattering* implementada presentan efectos de sombras y translucidez en comparación al modelo de emisión-absorción llevado a cabo en el trazado de rayos. Más aún, los volúmenes presentan más realismo adicionando el modelo de iluminación con gradientes al resultado final. Para cada una de las pruebas que se encuentran a continuación, se muestran las funciones de transferencia utilizadas.

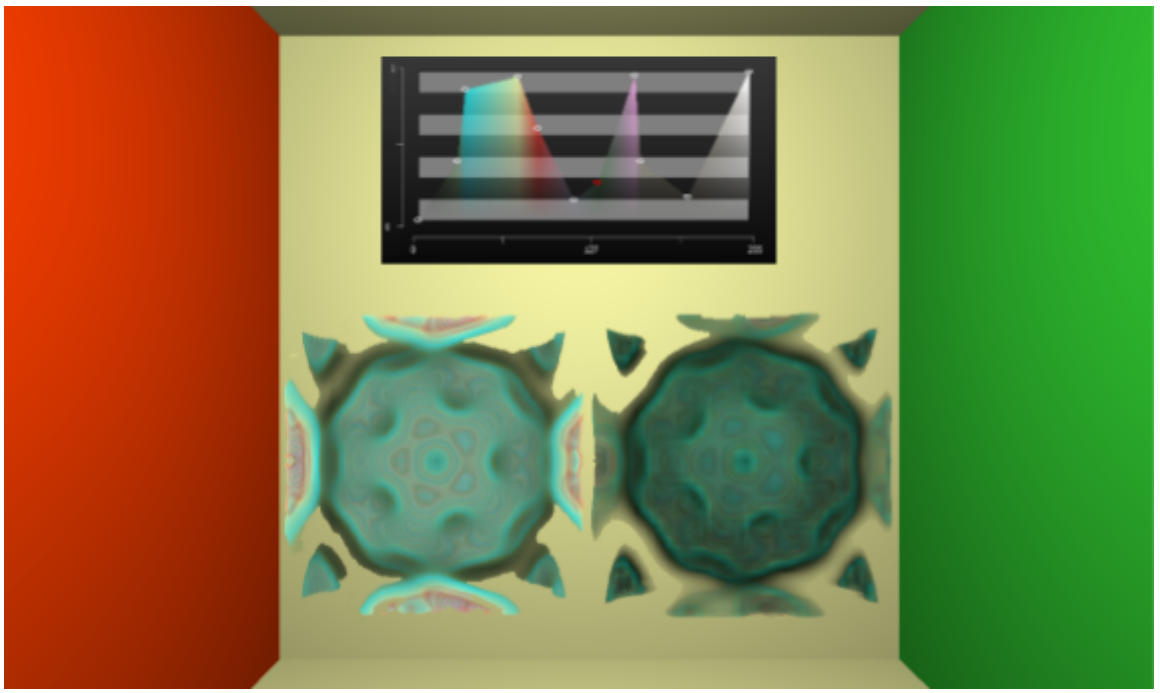

Figura 5.13: *Comparación de emisión-absorción y scattering en Bucky.* Molécula de futboleno desplegada con el modelo de emisión-absorción clásico (izquierda) y la técnica de *scattering* implementada (derecha).

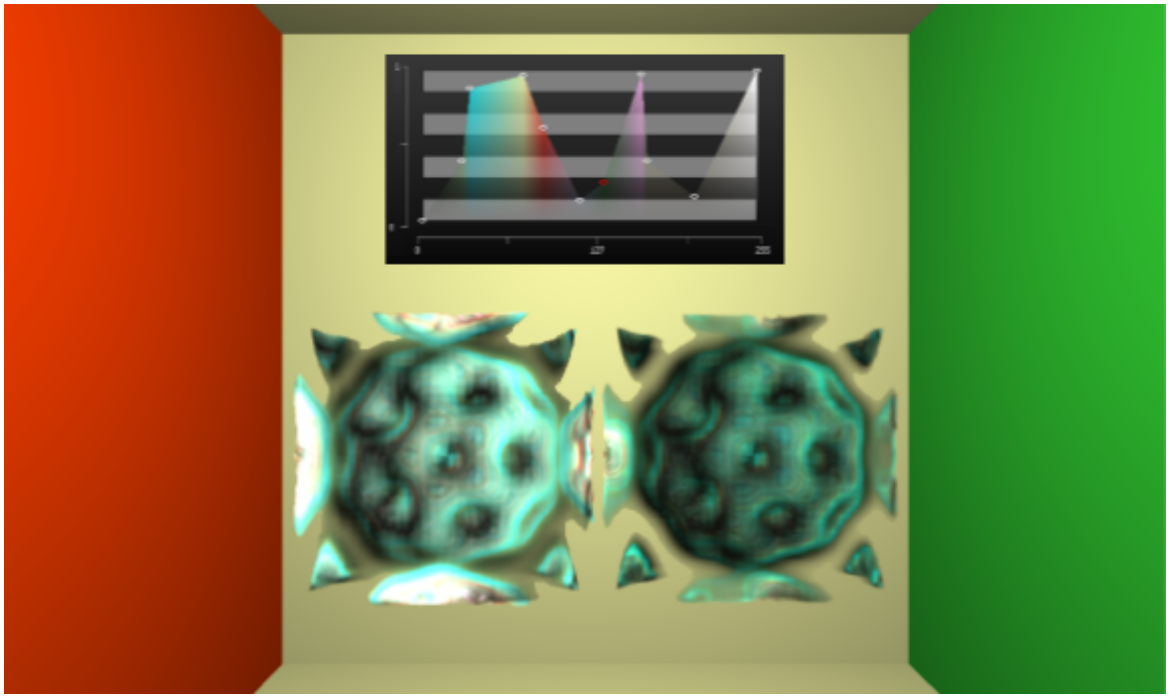

Figura 5.14: *Comparación de emisión-absorción y scattering con gradientes en Bucky.* Molécula de futboleno desplegada con el modelo de emisión-absorción clásico (izquierda) y la técnica de *scattering* implementada (derecha) junto con el modelo de iluminación por gradientes.

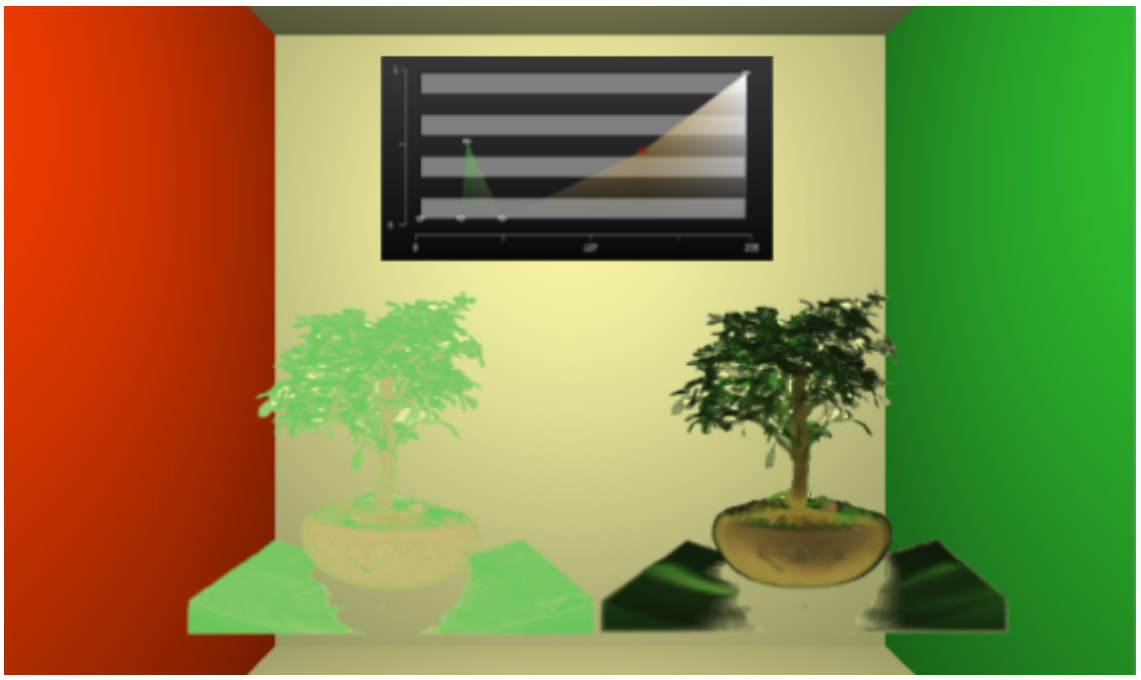

Figura 5.15: *Comparación de emisión-absorción y scattering en Bonsai.* Árbol bonsai *rendering* con el modelo de emisión-absorción clásico (izquierda) y la técnica de *scattering* implementada (derecha).

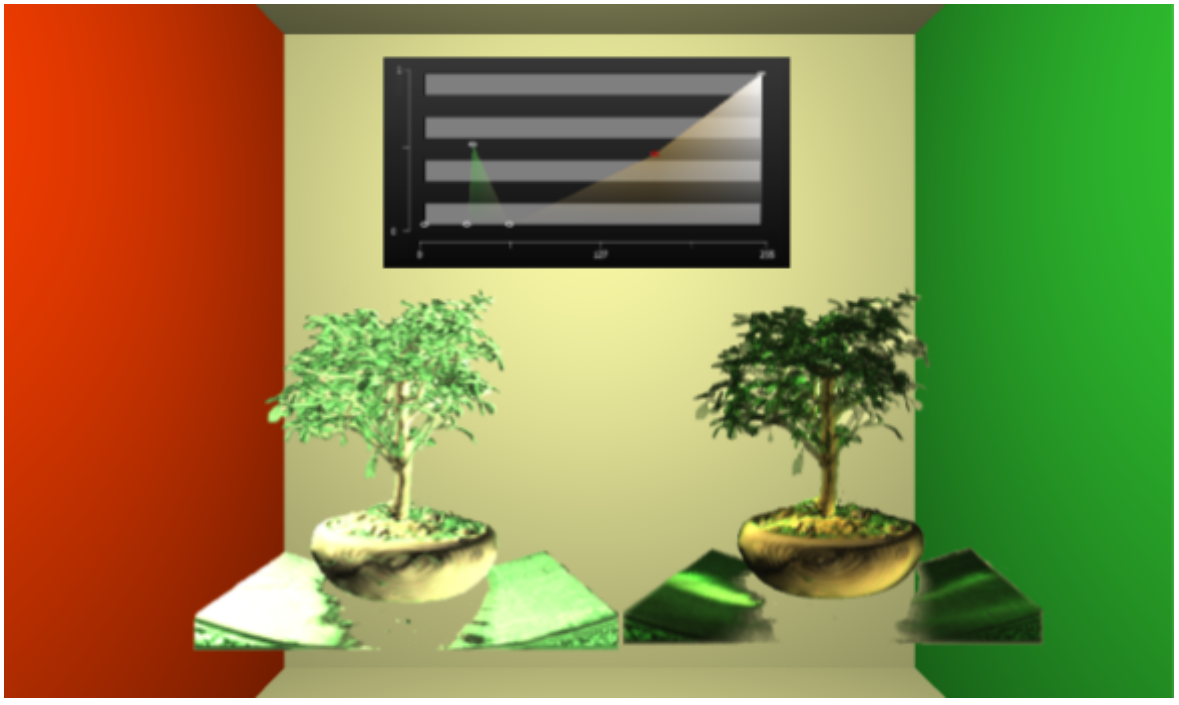

Figura 5.16: *Comparación de emisión-absorción y scattering con gradientes en Bonsai.* Árbol bonsai *rendering* con el modelo de emisión-absorción clásico (izquierda) y la técnica de *scattering* implementada (derecha) junto con el modelo de iluminación por gradientes.

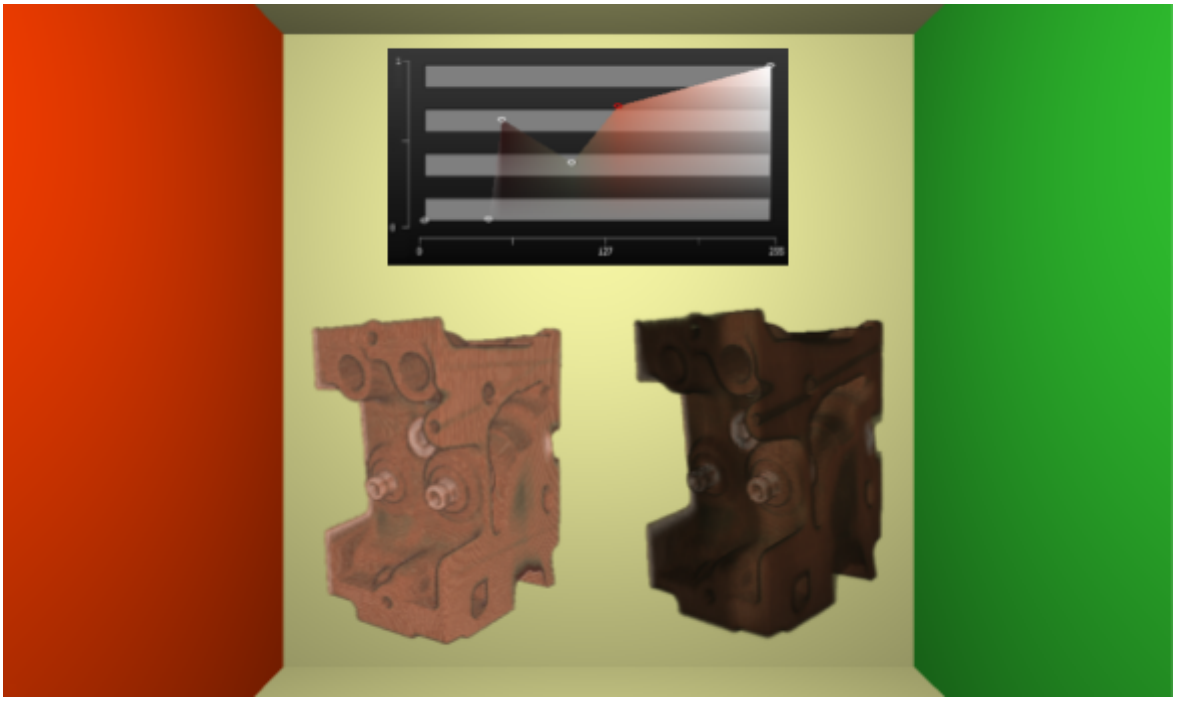

Figura 5.17: *Comparación de emisión-absorción y scattering en Motor.* Pieza de motor desplegada con el modelo de emisión-absorción clásico (izquierda) y la técnica de *scattering* implementada (derecha).

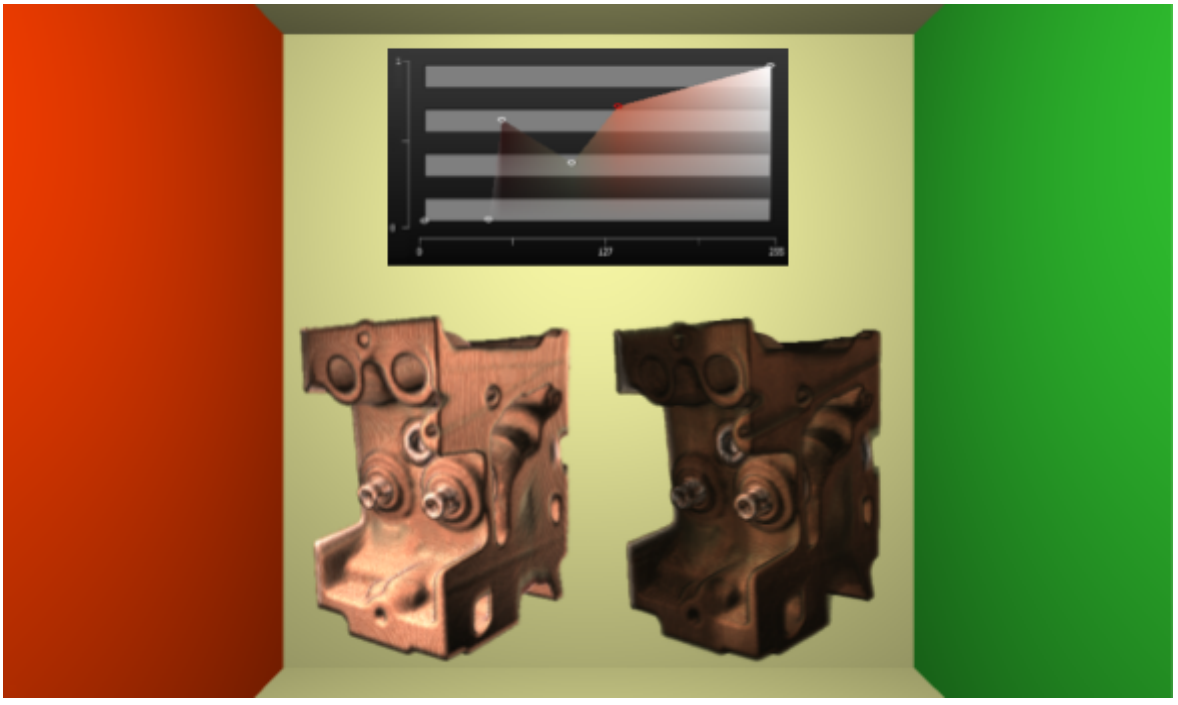

Figura 5.18: *Comparación de emisión-absorción y scattering con gradientes en Motor.* Pieza de motor desplegada con el modelo de emisión-absorción clásico (izquierda) y la técnica de *scattering* implementada (derecha) junto con el modelo de iluminación por gradientes.

4 COMPARACIONES CON TRABAJOS PREVIOS

Los resultados de la aplicación final fueron comparados con los trabajos presentados en el ***Cap. I*** del trabajo especial de grado. Las pruebas se realizaron aplicando una configuración de parámetros equivalentes tanto la presentada en datos volumétricos como en modelos mallados.

4.1 MODELOS MALLADOS

Se realizaron comparaciones con el trabajo realizado por Dal Corso **[2]** utilizando el modelo *Stanford Buddha*. Como se mencionó anteriormente, la técnica implementada se basa en un algoritmo de tres pasos, donde se lleva a cabo el cálculo de mapas de luz, mapas de dispersión y finalmente la combinación de estos últimos. En la ***Fig. 5.16*** se muestra una comparativa aplicando una configuración equivalente en ambas implementaciones.

Figura 5.19: *Comparación con trabajo previo en mallados*. Comparación del modelo *Stanford Buddha* hecho de patata aplicando la técnica de *scattering* implementada (izquierda) y el trabajo previo descrito (derecha). La diferencia visual en ambas imágenes se debe a la cantidad de luces utilizadas para representar el *scattering*: en el caso del trabajo de Dal Corso, se utilizaron 16 luces direccionales de las cuales no se especifican las posiciones; en este trabajo, se utilizó una sola luz direccional frente al modelo. A su vez, otros parámetros de dispersión (radio de dispersión y parámetro de asimetría) no son especificados por el autor.

4.2 MODELOS VOLUMÉTRICOS

La técnica implementada está basada en el cálculo de un volumen de luz para representar efectos de translucidez y sombras. Debido a que en la investigación de Ropinski **[3]** no se provee la función de transferencia utilizada para mostrar los resultados finales, no se pudo realizar una correcta comparativa entre ambas implementaciones (*ver* ***Fig. 5.17***). Sin embargo, se presentan resultados esperados de *scattering* en el volumen final.

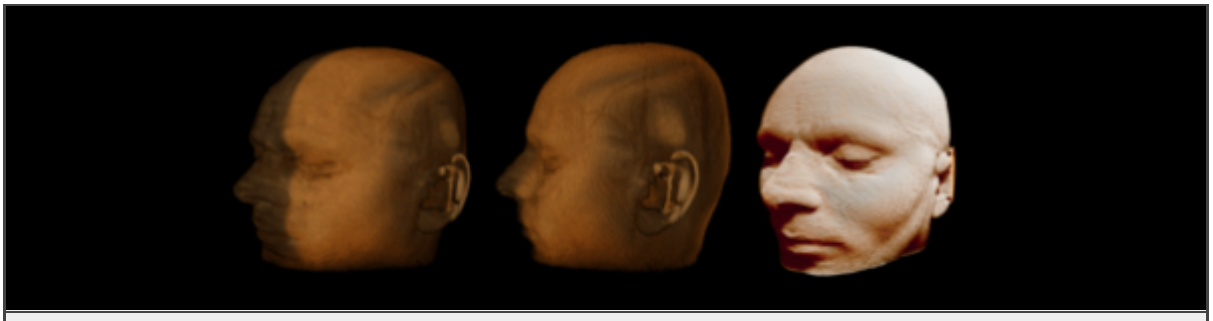

Figura 5.20: *Comparación con trabajo previo modelos volumétricos*. Comparación del modelo volumétrico de una cabeza humana aplicando la técnica de *scattering* implementada (izquierda y centro) y el trabajo previo descrito (derecha).

CONCLUSIONES

En el trabajo especial de grado explicado a lo largo de este documento, se puede concluir que se alcanzó de forma exitosa el objetivo general planteado, de implementar una aplicación donde se aplique la técnica de dispersión de la luz (*scattering*) en volúmenes y mallados en una escena híbrida con la finalidad de que las imágenes desplegadas presenten efectos físicos realistas de translucidez y sombras.

En base a las pruebas realizadas, se puede concluir que en el algoritmo propuesto para *scattering* en mallados, el número de cámaras ortográficas utilizadas para desplegar la escena es influyente en el resultado final, ya que se presentan mejores resultados visuales en modelos con características cóncavas al momento de establecer una gran cantidad de éstas, como por ejemplo el buda de *Stanford* con 16 cámaras, por el simple hecho de que se puede cubrir la mayor parte del modelo, mientras que en mallados con características convexas el aumento del número de cámaras solo suavizará el resultado final. A pesar de que un mayor número de cámaras ralentiza la generación de mapas de dispersión, se debe destacar que este algoritmo es independiente de la vista, por ende, el tiempo de cálculo de estos mapas es mayor al tiempo de *rendering* de la imagen final, y el recálculo de los mismos sólo estará atado a alguna transformación afin aplicada al modelo, cambios en los parámetros de la técnica y cambios en la posición de la luz.

Comparando los resultados finales obtenidos en esta técnica, se observa que no hay mucha diferencia al considerar 8 y 16 cámaras en cuanto a los resultados visuales, mientras que el tiempo de cálculo entre ambos números es considerablemente mayor. En estos casos, si se logra cubrir por completo el modelo mallado, y a su vez, se obtienen los resultados de *scattering* esperados, no es necesario incluir una gran cantidad de cámaras en la escena. Además, se pudo observar que los tiempos de despliegue del modelo cuando la escena es fija y cuando ocurren cambios en la misma, presentan un comportamiento aproximadamente lineal. Este comportamiento depende del número de cámaras ortográficas en la escena.

Comparando el algoritmo de *scattering* en mallados con la técnica de Dal Corso **[2]**, se obtuvieron resultados similares en cuanto a la calidad visual de las imágenes finales. El tiempo de cálculo de los mapas de dispersión no puede ser comparado en ambas implementaciones debido a la desigualdad del *hardware* utilizado y los parámetros utilizados en ambas técnicas. Cabe destacar que esta técnica fue extendida agregando el componente especular en el resultado final de la iluminación, y se mejoró la función de visibilidad (originalmente binaria, es decir, si el fragmento es visible, entonces la función es igual a 1; en caso contrario, la función es igual a 0) tomada en cuenta para el cálculo de los mapas de dispersión como un promedio de vecindad entre las muestras generadas.

Realizando una comparativa del algoritmo de *scattering* en volúmenes con la técnica propuesta por Ropinski **[3]**, se obtuvieron resultados similares en cuanto a la translucidez y sombras de las imágenes finales. Nuevamente, el tiempo de cálculo del volumen de luz inicial no puede ser comparado en ambas implementaciones debido a la desigualdad del *hardware* utilizado. Sin embargo, el cálculo de este volumen de luz es independiente de la vista, y el recálculo del mismo sólo viene dado por algún cambio en la escena que involucre al volumen final. A diferencia de la técnica de *scattering* en mallados, la implementación del algoritmo de *scattering* en volúmenes no obliga al usuario a cambiar parámetros desde una interfaz interactiva. Éste sólo va a intervenir al momento de realizar una configuración en la función de transferencia como ocurre en el algoritmo de trazado de rayos original. Cabe destacar que esta técnica fue extendida eliminando el uso de permutaciones para el cálculo del volumen de luz, mediante una función propia de GLSL denotada *imageStore* la cual permite almacenar una textura en la estructura de una imagen 3D.

La aplicación desarrollada fue llevada a cabo siguiendo buenas prácticas de desarrollo, cumpliendo con propiedades como la escalabilidad, usabilidad y portabilidad, y por tanto, puede ser extendida por otros desarrolladores de manera sencilla. La técnica implementada presenta resultados más realistas, donde se pueden observar efectos de translucidez y sombras, en comparación a los obtenidos con un modelo de iluminación local clásico como *Blinn-Phong*, el cual no refleja estos efectos. Finalmente, se destaca que la implementación de este trabajo especial de grado permitió dar un aporte importante al Laboratorio de Computación Gráfica de la Universidad Central de Venezuela, logrando iniciar línea de investigación a futuros trabajos que apliquen modelos de iluminación relacionados con *scattering*.

TRABAJOS FUTUROS

Aunque la aplicación desarrollada en este trabajo especial de grado complementa las investigaciones y trabajos realizados en el Laboratorio de Computación Gráfica de la Universidad Central de Venezuela, no queda exenta de algunas recomendaciones en pro de su evolución:

- Implementar la técnica de *scattering* múltiple en modelos mallados y modelos volumétricos.
- Agregar a la escena híbrida algún tipo de detección de colisiones entre mallados y volúmenes, implementado previamente por Carmona **[1]**.
- Agregar más luces. Esto conlleva a que la técnica de *scattering* en mallados produzca resultados más suaves e iluminados de los modelos.
- Implementar el algoritmo de *Marching Cubes* para transformar los modelos volumétricos a mallados y aplicarles la técnica de *subsurface scattering* para así comparar los resultados finales entre ambos tipos de datos.
- Utilizar diferentes tipos de luces en la escena, tales como una luz puntual o reflector; ya que actualmente sólo se consideran luces direccionales.
- Texturizar los modelos mallados y comparar los resultados de la técnica de *scattering* con los materiales por defecto que se muestran en la aplicación.

REFERENCIAS

[1] J. Ortegano, H. Navarro, R. Carmona. "Volume-Surface Collision Detection". En Proc. V Iberoamerican Symposium in Computer Graphics (SIACG 2011). F. Silva et al. (Eds.), Portugal, pp. 175-182, Junio 2011.

[2] A. Dal Corso, J. R. Frisvad, J. Mosegaard y J. A Bærentze. "Interactive Directional Subsurface Scattering and Transport of Emergent Light". The Visual Computer, pp. 1–8, 2016.

[3] T. Ropinski, C. Döring y C. Rezk-Salama. "Interactive Volumetric Lighting Simulating Scattering and Shadowing". IEEE Pacific Visualization Symposium (PacificVis), pp. 1–8, 2010.

[4] M. Ament y D. Weiskopf. "Ambient Volume Scattering". IEEE Transactions on Visualization and Computer Graphics, pp. 1–10, 2013.

[5] M. Pharr y P. Hanrahan. "Monte Carlo Evaluation Of Non-Linear Scattering Equations For Subsurface Reflection". En Computer Graphics, SIGGRAPH 2000 Proceedings, 2005.

[6] M. Levoy. "Display of Surfaces from Volume Data". IEEE Computer Graphics and Applications, vol. 8, num. 3, pp. 29–37, Julio 1988.

[7] R. Carmona. “Introducción al rendering de volúmenes”. Reporte técnico No. RT-2015-01, Escuela de Computación, Universidad Central de Venezuela. Febrero 2015.

[8] J. Kniss, S. Premoze, C. Hansen y D. Ebert. "Interactive translucent volume rendering and procedural modeling". Proceedings of IEEE Visualization 2002, pp. 109–116, 2002.

[9] H. W. Jensen, S. R. Marschner, M. Levoy, P. Hanrahan. "A practical model for subsurface light transport". Proceedings of ACM SIGGRAPH 2001, pp. 511–518, 2001.

[10] K. Group. OpenGL® - The Industry Standard for High Performance Graphics. [En línea] https://www.opengl.org/.

[11] K. Group. Current OpenGL API, OpenGL Shading Language and GLX Specifications. Khronos OpenGL® Registry. [En línea] https://www.khronos.org/registry/OpenGL/index_gl.php/.

[12] GLFW . GLFW - An OpenGL® library. [En línea] http://www.glfw.org/.

[13] AntTweakBar . AntTweakBar GUI library to tweak parameters of your OpenGL® and DirectX programs. [En línea] http://anttweakbar.sourceforge.net/doc/.

[14] GLM. OpenGL® Mathematics. [En línea] https://glm.g-truc.net/0.9.8/index.html.

[15] GLEW. GLEW: The OpenGL® Extension Wrangler Library. [En línea] http://glew.sourceforge.net/.

[16] K. Moreland. "Fast High Accuracy Volume Rendering". Tesis doctoral, Universidad de New Mexico, Julio 2004.

[17] T.T. Elvins. "A Survey of Algorithms for Volume Visualization". Computer Graphics, Vol. 26, No. 3, pp. 194–201, 1992.

[18] B. McCormick, T. DeFanti, y M. Brown. "Visualization in Scientific Computing". Computer Graphics, vol. 21, num. 6, pp. 21–37, Noviembre 1987.

[19] K. Brodlie, L. Carpenter, R. Earnshaw, J. Gallop, R. Hubbold, C. Mumford, C. Osland y P. Quarendon. "Scientific Visualization-Techniques and Applications". Reporte técnico, Institute for Biological and Medical Imaging, Alemania y Tomsk Polytechnic University, Rusia, pp. 130–133, 1991.

[20] S. Chandrasekhar. "Radiative Transfer". Dover, New York, 1960.

[21] W. Krueger. "The application of Transport Theory to Visualization of 3D Scalar Data Fields". En Proc. IEEE Visualization '90, pp. 272–280, Octubre 1990.

[22] J. Hansen y L. Travis. "Light Scattering in Planetary Atmospheres". Space Science Reviews 16, pp. 527–610, 1974.

[23] J. Foley, A. Van Dam, S. Feiner y J. Hughes. "Computer Graphics: Principles and Practice". Addison-Wesley, 2da. edición, 1990.

[24] J. Blinn. "Light reflection functions for simulation of clouds and dusty surfaces". Computer Graphics ACM SIGGRAPH 82, vol. 16, pp. 21–29, Julio 1982.

[25] J. Kajiya y B. Von Herzen. "Ray tracing volume densities". Computer Graphics ACM SIGGRAPH '84, vol. 18, num. 3, pp. 165–174, Julio 1984.

[26] L. Westover. "Footprint Evaluation for Volume Rendering". Computer Graphics, vol. 24, num. 4, pp. 367–376, 1990.

[27] B. Guo. "Interval Set: A Volume Rendering Technique Generalizing Isosurface Extraction". Proceedings of the 6th Conference on Visualization ’95, p. 3, 1995.

[28] J. Kniss, G. Kindlmann y C. Hansen. "Multidimensional Transfer Functions for Interactive Volume Rendering". IEEE Transactions on Visualization and Computer Graphics, vol. 8, pp. 270–285, 2002.

[29] K. Engel, M. Kraus y T. Ertl. "High Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading". Siggraph Eurographics Workshop on Graphics Hardware, pp. 9–16, California-USA, 2001.

[30] P.G. Lacroute. "Fast Volume Rendering Using Shear-Warp Factorization of the Viewing Transformation". Reporte Técnico CSL-TR-95-678, Universidad de Stanford, USA, 1995.

[31] P. Sabella. "A rendering algorithm for visualizing 3D scalar fields". En. Proc. ACM SIGGRAPH '88, Computer Graphics, vol. 22, pp. 51–58, Agosto 1988.

[32] P. Williams y M. Nelson. "A volume density optical model". En Proc. Workshop on Volume Visualization '92, Computer Graphics, pp. 61–68, Octubre 1992.

[33] M. Levoy. "Volume Rendering: A Hybrid Ray Tracer for Rendering Polygon and Volume Data". IEEE Computer Graphics and Applications, vol. 10, num. 2, pp. 33–40, Julio 1990.

[34] M. Levoy. "Efficient Ray Tracing from Volume Data". ACM Transactions on Graphics, vol. 9, num. 3, pp. 245–261, Julio 1990.

[35] R. Drebin, L. Carpenter y P. Hanrahan. "Volume rendering". Computer Graphics ACM SIGGRAPH 88, vol. 22, pp. 65–74, Agosto 1988.

[36] J. Danskin y P. Narran. "Fast Algorithms for Volume Ray Tracing". En Proc. Workshop on Volume Visualization '02, pp. 91–98, Boston-Massachusetts-USA, 1992.

[37] J. Krüger y R. Westermann. "Acceleration techniques for GPU-based Volume Rendering". En Proc. Visualization '03, pp. 287–292, Washington-USA, 2003.

[38] M. Meißner, H. Pfister, R. Westermann y C.M. Wittenbrink. "Volume Visualization and Volume Rendering Techniques", p. 13. Eurographics Association, 2000.

[39] A. Madero. "Visualización Volumétrica Estereoscópica en Tiempo Real". Proyecto de Grado, Biblioteca Alonso Gamero, Universidad Central de Venezuela, 2001.

[40] E. LaMar, B. Hamann y K. Joy. "Multiresolution Techniques for Interactive Texture-based Volume Visualization". En Proc. Visualization '99, pp. 355–361, California-USA, 1999.

[41] Anónimo. "Direct Volume Rendering". Stuttgart Visualization Course. Universidad de Stuttgart, Alemania, pp. 13–14, 2006.

[42] M. Botsch, M. Pauly, L. Kobbelt, P. Alliez, B. Lévy, S. Bischoff y C. Rössl. "Geometric Modeling Based on Polygonal Meshes". Notas del curso de ACM SIGGRAPH 2007, pp.13–33, Noviembre 2007.

[43] G. Farin. "Curves and Surfaces for Computer Aided Geometric Design". Academic Press, 4th edition, 1997.

[44] L. A. Piegl y W. Tiller. "The NURBS Book". Springer, 2nd edition, 1997.

[45] H. Prautzsch, W. Boehm, y M. Paluszny. "Bézier and B-Spline Techniques". Springer Verlag, 2002.

[46] D. Zorin, P. Schr ¨oder, T. DeRose, L. Kobbelt, A. Levin, y W. Sweldens. "Subdivision for modeling and animation". Notas del curso de ACM SIGGRAPH 00, 2000.

[47] H. Samet. "The Design and Analysis of Spatial Data Structures". Addison–Wesley, 1994.

[48] S. Frisken, R. Perry, A. Rockwood y T. Jones. "Adaptively sampled distance fields: A general representation of shape for computer graphics". En Proc. ACM SIGGRAPH 00, pp. 249–254, 2000.

[49] A. Kaufman. "Efficient algorithms for 3D scan-conversion of parametric curves, surfaces, and volumes". En Proc. ACM SIGGRAPH 87, pp. 171–179, 1987.

[50] W. E. Lorensen y H. E. Cline. "Marching cubes: a high resolution 3D surface construction algorithm". En Proc. ACM SIGGRAPH 87, pp. 163–170, 1987.

[51] L. Kobbelt, M. Botsch, U. Schwanecke, y H. P. Seidel. "Feature sensitive surface extraction from volume data". En Proc. ACM SIGGRAPH 01, pp. 57–66, 2001.

[52] N. Max. "Optical models for direct volume rendering". IEEE Transactions on Visualization and Computer Graphics, pp. 99–108, 1995.

[53] F. Lindemann y T. Ropinski. "About the influence of illumination models on image comprehension in direct volume rendering". IEEE Transactions on Visualization and Computer Graphics, pp. 1922–1931, 2011.

[54] M. Langer y H. Bulthoff. "Depth discrimination from shading under diffuse lighting". Perception, pp. 649–660, 2000.

[55] B. T. Phong. "Illumination for computer generated pictures". Communications of the ACM, pp. 311–317, 1975.

[56] M. Schott, V. Pegoraro, C. Hansen, K. Boulanger y K. Bouatouch. "A directional occlusion shading model for interactive direct volume rendering". Computer Graphics Forum, pp. 855–862, 2009.

[57] E. Cerezo, F. Perez, X. Pueyo, F. J. Seron y F. X. Sillion. "A survey on participating media rendering techniques". The Visual Computer, pp. 303–328, 2005.

[58] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, y T. Limperis. "Geometrical considerations and nomenclature for reflectance". Jones and Bartlett Publishers, Inc., pp. 94–145, 1992.

[59] D. S. Kay y D. Greenberg. "Transparency for computer synthesized images". SIGGRAPH Comput. Graph., pp. 158–164, 1979.

[60] H. W. Jensen, S. R. Marschner, M. Levoy y P. Hanrahan. "A practical model for subsurface light transport". In Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '01, pp. 511–518, 2001.

[61] J. R. Frisvad, T. Hachisuka y T. K. Kjeldsen. "Directional dipole for subsurface scattering in translucent materials". ACM Transactions on Graphics, 2014.

[62] J. H. Joseph, W. J. Wiscombe y J. A. Weinman. "The delta-Eddington approximation for radiative flux transfer". Journal of Atmospheric Sciences, pp. 2452–2459, 1976.

[63] E. D'Eon y G. Irving "A quantized-diffusion model for rendering translucent materials". ACM Transactions on Graphics, pp. 1–14, 2011.

[64] N. Max. "Optical models for direct volume rendering". IEEE Transactions on Visualization and Computer Graphics, pp. 99–108, 1995.

[65] J. Kniss, S. Premoze, C. Hansen, P. Shirley y A. McPherson. "A model for volume lighting and modeling". IEEE Transactions on Visualization and Computer Graphics, pp. 150–162, 2003.

[66] J. Kajiya. "The rendering equation". Computer Graphics, pp 143–150, 1986.

[67] P. Hanrahan y W. Krueger. "Reflection from layered surfaces due to subsurface scattering". In Computer Graphics Proceedings, ACM SIGGRAPH, Annual Conference Series, pp. 165–174, 1993.

[68] M. Pharr y P. Hanrahan. "Monte Carlo Evaluation Of Non-Linear Scattering Equations For Subsurface Reflection". Stanford University, p. 14–29, 2005.

[69] P. Hanrahan y W. Krueger. "Reflection from layered surfaces due to subsurface scattering". In Computer Graphics Proceedings, ACM SIGGRAPH, Annual Conference Series, pp. 165–174, 1993.

[70] H. W. Jensen. "Realistic Image Synthesis Using Photon Mapping". AK Peters. 2001.

[71] B. Sun, R. Ramamoorthi, S. G. Narasimhan y S. K. Nayar. "A practical analytic single scattering model for real time rendering". ACM Transactions on Graphics 24, 3, pp. 1040–1049, 2005.

[72] T. Nishita, Y. Miyawaki, y E. Nakamae. "A shading model for atmospheric scattering considering luminous intensity distribution of light sources". Computer Graphics (Proc. of Siggraph), pp. 21, 4, pp. 303–310, 1987.

[73] D. Mitchell y P. Hanrahan. "Illumination from curved reflectors". Computer Graphics Proc. of Siggraph, pp. 26, 3, 283–291, 1992.

[74] M. Chen y J. Arvo "Theory and application of specular path perturbation". ACM Transactions on Graphics, pp. 19, 4, 246–278, 2000.

[75] T. Nishita y E. Nakamae. "Method of displaying optical effects within water using accumulation buffer". En Computer Graphics Proceedings, ACM SIGGRAPH, Annual Conference Series, pp. 373–379 1994.

[76] K. Iwasaki, Y. Dobashi y T. Nishita. "A fast rendering method for refractive and reflective caustics due to water surfaces". Computer Graphics Forum, pp. 601–610, Septiembre 2003.

[77] M. Ernst, T. Akenine-Moller y H. W. Jensen. "Interactive rendering of caustics using interpolated warped volumes". En Graphics Interface 2005, pp. 87–96, 2005.

[78] Stanford University. The Stanford 3D Scanning Repository. [En línea] http://graphics.stanford.edu/data/3Dscanrep/

[79] Thingiverse. Hebe by Geoffrey Marchal. [En línea]
https://www.thingiverse.com/thing:2455081/files

[80] GAMMA Research Group. Public Models. [En línea]
http://gamma.cs.unc.edu/public_models/Models/MiscOBJ/sphere.obj

[81] S. Röttger. The volume library. [En línea]
http://www9.informatik.uni-erlangen.de/External/vollib/