Graphics and Imaging Lab

Photon Mapping

Informática Gráfica

Grado en Ingeniería Informática

Objetivo

El objetivo de esta tarea consiste en extender un trazador de rayos simple para soportar iluminación global usando el algoritmo de **Photon Mapping**, en particular la versión original de Jensen [Jen96, Jen01]. Para ello, se utilizará el *path tracer* implementado en la práctica anterior.

1 Tarea anterior

El trabajo obligatorio está dividido en dos partes principiales, una para cada paso del algoritmo del *photon mapping*. Para la primera parte, los fotones son enviados desde las fuentes de luz a la escena en un proceso llamado *random walk*, y son almacenados en una estructura de datos intermedia llamada **mapa de fotones**. La segunda parte de la tarea requiere trazar rayos desde la cámara y usar el mapa de fotones para computar la ecuación de renderizado [Kaj86] en el punto de intersección.

El algoritmo de *photon mapping* puede reutilizar una gran parte del código que ya habéis escrito. Algunas partes las tendréis que implementar desde cero, pero se recomienda buscar en vuestra implementación existente antes de reinventar la rueda. También proporcionamos una implementación de un k-d tree, explicado en la Sección 4, que podéis usar para vuestros mapas de fotones.

2 Generación del mapa de fotones

En la primera parte de la tarea, tendréis que generar un mapa de fotones para la escena. Para ello, tendréis que muestrear una fuente de luz en la escena para comenzar el *random walk* de fotones. Esto significa seleccionar una luz (puntual) de la escena y muestrear un rayo desde la posición de la luz en una dirección aleatoria. Este rayo es el origen del *random walk*, cuya energía inicial es la energía de la fuente de luz. Para cada intersección del *random walk* tendréis que almacenar un fotón en el mapa de fotones, incluyendo su posición, dirección incidente y flujo. Repetid este proceso múltiples veces para obtener vuestro el mapa de fotones como resultado. Recordad que el flujo de cada fotón debe ser normalizado para asegurar la **conservación de la energía**.

La generación del *random walk* desde la fuente de luz tiene muchas similitudes con el trabajo anterior en cuanto al muestreo de intersecciones se refiere, incluyendo la Ruleta Rusa. Deberíais considerar generar el *random walk* de fotones con las funciones que ya implementasteis para el *path tracing*.

No todas las interacciones generan un fotón que debe almacenarse en el mapa de fotones. Las excepciones y sus razones se detallan en la siguiente sección.

3 Estimación de radiancia

En la segunda parte del ejercicio, tendréis que utilizar el mapa de fotones para computar la Ecuación de Render en múltiples puntos para generar una imagen. La implementación de esta parte también debería basarse en vuestro algoritmo de *path tracing* existente, trazando rayos desde la cámara e intersectando con la escena múltiples veces. Sin embargo, en este caso, la Ecuación de Render requerirá encontrar los k fotones vecinos (los más cercanos) a cada punto que se está iluminando, y usarlos para computar la radiancia reflejada en ese punto. Las contribuciones de los k fotones recuperados pueden sumarse (es decir, usando un kernel de filtro de caja). Notad que también debéis realizar **normalización del kernel** para asegurar la conservación de la energía.

Hay multiples casos donde no deberíais guardar los fotones, ni buscar los fotones vecinos en el mapa de fotones. En primer lugar, las superficies perfectamente especulares y transmisivas. Para estos materiales, evaluar su BSDF delta dará un resultado cero. Para ambas partes del algoritmo, deberían generar un nuevo rayo en la dirección correspondiente y seguir intersectando hasta encontrar una superficie difusa (o parcialmente difusa, como el plástico). Segundo, la luz directa puede ser computada analíticamente desde las luces puntuales sin la ayuda de un mapa de fotones en cada rebote utlizando vuestro *next-event estimation*. Notad que, para este segundo caso, no deberíais guardar los fotones del primer rebote durante el *random walk* de los fotones explicado en la sección anterior. Deberíais guardar los fotones del primer rebote **o** usar *next-event estimation*, pero no ambas. Para comprobar que realizáis este paso correctamente, podéis generar una imagen con cada enfoque, y los resultados deberían tener valores muy similares sin aplicar ningún *tone mapping*.

4 Detalles de implementación

Código proporcionado. Proporcionamos la implementación de una estructura de datos k-d tree que podéis utilizar para vuestro mapa de fotones. Como habéis visto, *photon mapping* consiste en encontrar los k fotones más cercanos a un punto para computar la radiancia. Realizar algún tipo de partición del espacio mejora enormemente la velocidad de este proceso, y el k-d tree hace un gran trabajo en este sentido.

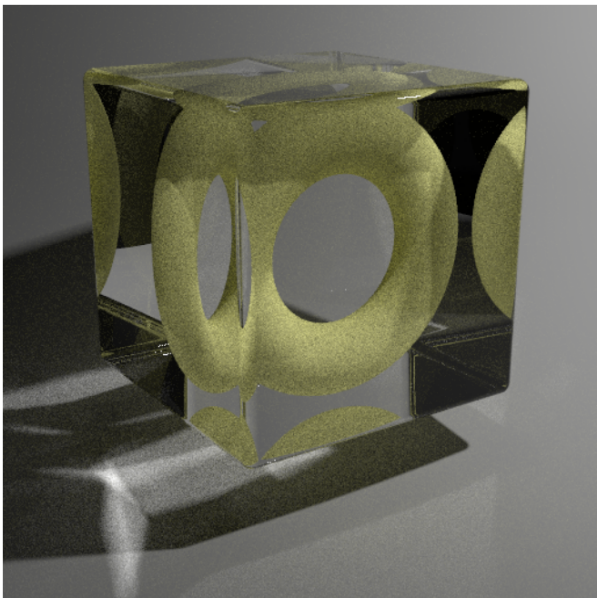

Figura 1: *Toro lambertiano dentro de un cubo de cristal, renderizado mediante iluminación global robusta usando* photon mapping. *Imagen de [HOJ08].*

Podéis descargaros los fuentes de Moodle, implementados en *C++*. La clase `KDTree` está implementada en el fichero *kdtree.h*. Podéis encontrar un ejemplo de uso en el fichero *kd_tree_example.cpp*, que os puede ayudar a integrar la clase proporcionada en vuestro código.

Lenguaje de programación. Los utilizar cualquier lenguaje de programación que consideren. Sin embargo, recomendamos C++ debido a su eficiencia comparada con otros lenguages, y también porque es el lenguaje de programación de los fuentes proporcionados.

Librerías externas. El uso de cualquier librería o código descargado esta estríctamente prohibido, con la excepción del código fuente de vuestro propio trabajo previo. Si incluís código de vuestro trabajo previo, deberéis documentarlo adecuadamente (cuando lo generasteis, para qué práctica, y qué parte del código es).

5 Extensiones opcionales

Adicionalmente a las partes obligatorias (Secciones 2 and 3), podéis implementar partes opcionales. A continuación se proponen varias extensiones opcionales, aunque sois libres de proponer cualquier otra extensión que consideréis interesante. En cualquier caso, antes de empezar cualquier implementación opcional, por favor preguntadnos primero para acordar

qué puede tener sentido. Extensiones no relacionadas o triviales (por ejemplo, paralelizar el código en múltiples hilos) no serán consideradas como una extensión.

- Utilizar un **mapa de fotones separado para fotones caústicos** que ocurren justo después de interacciones delta BSDF. Para la estimación de la radiancia, sumar las contribuciones de ambos mapas, usando **parámetros diferentes para cada mapa de fotones** para hacer las cáusticas más claramente definidas.
- Extender el *photon mapping* para que soporte **medios participativos** [JC98].
- Implementar un renderizador de iluminación global basado en **Virtual Point Lights (VPL)** [Kel97]. Estos métodos están estrechamente relacionados con el *photon mapping*, especialmente con su generalización a las llamadas *Virtual Spherical Lights* [HKWB09].
- **Mapa de fotones progresivo** [HOJ08], para obtener una solución consistente sin necesidad de almacenar infinitos fotones. Una versión más simple de implementar (aunque más difícil de entender) del método se puede encontrar en [KZ11].
- Implementar un **kernel adaptativo del ancho de banda (es decir, el radio) para la estimación de la densidad de fotones** [FSES14]. Un enfoque relativamente simple para ello son los **diferenciales de fotones** [FSES14], aunque la implementación puede ser compleja. Un enfoque más formal y matemáticamente complejo (aunque más simple de implementar) es el trabajo de Kaplanyan y Dachsbacher [KD13].
- Desarrollar un ***photon mapping* transitorio**, siguiendo nuestro trabajo [JMMn+14].
- Analizar el **comportamiento de varios kernels sofisticados** (por ejemplo, Perlin, Epanechnikov...) en la estimación de la radiancia [Sil86]. Notad que tendréis que analizar filtros avanzados más allá de los kernels de caja y triángulo, de lo contrario no contará como una extensión adecuada.

6 Informe

Como entrega de este trabajo, cada grupo deberá escribir un informe acerca del diseño e implementación del *photon mapping*. El informe puede estar escrito en español o inglés, y su contenido debe responder a los siguientes puntos:

1. Escribid la **Ecuación de Render** y discutid brevemente (en un par de frases) como vuestro *photon mapper* tiene en cuenta cada uno de sus términos. No es necesario que describais la Ecuación, ya lo hicisteis en el informe anterior.
2. **Comparad la iluminación directa** obtenida con *next-event estimation* (rayos de sombra a las luces puntuales) y con los fotones almacenados en el mapa de fotones. Discutid los resultados en uno o dos párrafos. ¿Qué efectos se obtienen fácilmente con el mapa de fotones? ¿Qué problemas presenta el mapa de fotones cuando se usa para

estimar la iluminación directa? Vuestra discusión debe estar apoyada por los dos renders correspondientes de una *Cornell Box* con dos esferas de material completamente difuso Lambertiano.

3. Analiza los efectos de las **BSDFs delta** en el *Photon Mapping*. Para ello, renderizad una *Cornell Box* con dos esferas: una de material plástico (difuso + especular) y otra de material de cristal (transmisivo + especular). ¿Qué efectos de iluminación y apariencia provienen del *photon mapping*? ¿Qué efectos provienen de la parte de trazado de rayos del algoritmo?

4. Analizad y mostrad resultados renderizados utilizando un número creciente de fotones (por ejemplo, $N = \{1K, 10K, 100K\}$) fijando el número máximo de vecinos cercanos en la estimación de radianzia a $k = 10$. Ahora, renderiza la misma escena con $N = 100K$ fotones, incrementando el número de vecinos cercanos en la estimación de radiancia (por ejemplo, $k = \{1, 10, 50, 100\}$). Discute los resultados obtenidos en uno o dos párrafos, en términos de **coste**, **calidad de imagen** y **convergencia**.

Adicionalmente, deberíais añadir uno o dos párrafos para **cada extensión implementada** cuyo contenido sea:

1. Una descripción de la extensión implementada, incluyendo decisiones de diseño específicas.

2. Relacionad la extensión implementada con los materiales de clase y bibliografía: ¿se ha discutido en clase? ¿Cuándo? Si no, añadid referencias bibliográficas de los materiales que os han ayudado a implementar la extensión.

3. ¿Cómo mejora la extensión la versión básica del *path tracer*? Ilustrad con renders lado a lado, comparando tiempos de renderizado...

El informe debe seguir la estructura mencionada, respondiendo a las preguntas planteadas anteriormente en el mismo orden. Un informe más largo no implica necesariamente un mejor informe. Recordad que **la calidad del informe** también será tenida en cuenta para la evaluación. Para ello, tened en cuenta las siguientes recomendaciones:

- Sed **concisos** en vuestras explicaciones. Evitad redefinir términos ya conocidos o redundancias en el texto. Recordad que divagar suele llevar a malentendidos por parte del lector, y, en este caso, el lector decide vuestra nota.

- Al mostrar renders, **discutidlos**. ¿Qué ilustran los renders? ¿Qué fenómenos son relevantes? Esto se puede hacer en la propia leyenda de la figura.

- Puede ser útil **comparar renders** uno al lado de otro para ilustrar una característica específica (con y sin ella).

- **No incluyais código fuente.** Ya estás entregando el código fuente.

- Es normal inspirarse en fuentes externas (artículos, información en línea, Wikipedia o

incluso código fuente). Cuando esto suceda, **agrega referencias bibliográficas** a los artículos, libros o materiales de los que te has inspirado. Ten en cuenta que hay una delgada línea entre "inspirarse" y "copiar y pegar código".

7 Entrega

La entrega debe realizarse a través de Moodle (`https://moodle2.unizar.es/add`) por uno (y solo uno) de los estudiantes.

Código fuente. Todo el código fuente del *photon mapping* debe estar comprimido en un archivo `.zip`, con el siguiente nombre de archivo:

- `photonmapping_<nip1>_<nip2>.zip` (donde `<nip1>` y `<nip2>` son los ID de estudiante de 6 dígitos) si el trabajo se ha realizado entre dos estudiantes.
- `photonmapping_<nip>.zip` (donde `<nip>` es el ID de estudiante de 6 dígitos) si el trabajo se ha realizado de forma individual.

También debe incluir un archivo *readme* con instrucciones claras de compilación y ejecución.

Informe. El informe debe entregarse en formato `.pdf`, con el siguiente nombre de archivo:

- `photonmapping_<nip1>_<nip2>.pdf` (donde `<nip1>` y `<nip2>` son los ID de estudiante de 6 dígitos) si el trabajo se ha realizado entre dos estudiantes.
- `photonmapping_<nip>.pdf` (donde `<nip>` es el ID de estudiante de 6 dígitos) si el trabajo se ha realizado de forma individual.

8 Evaluación

La evaluación de la parte obligatoria (Secciones 2 and 3) se realizará hasta 7. Para una mayor nota, necesitaréis implementar extensiones opcionales (ver Sección 5 para detalles y sugestiones).

La mayor parte de la evaluación se realizará basandose en el **informe**. Esto significa que debéis explicar lo que habéis hecho y demostrar que entendéis la técnica en el informe. Además. el código entregado debe **compilar y generar** las escenas que mostráis en el render. No hay puntos extra por usar vuestro propio código en lugar del proporcionado.

Preguntas & comentarios

Para cualquier pregunta sobre las partes opcionales, o dudas sobre la parte obligatoria, puedes acudir a los profesores durante la sesión de prácticas, o enviarnos un e-mail:

- Diego Royo – droyo@unizar.es
- Pablo Luesia – pluesia@unizar.es
- J. Daniel Subías – dsubias@unizar.es

Para cuestiones cara a cara (es decir, *tutorías*), envíanos un email. Es más facil responder a una pregunta cuando se sabe exactamente la duda. Así que, por favor, considera utilizar **pseudo-código** para cualquier duda acerca de la implementación, y utiliza **imágenes** cuando menciones algún efecto específico del renderizado (*una imagen vale más que mil palabras*). Para dudas respecto a teoría, por favor, contacta con los profesores de teoría: Adolfo (adolfo@unizar.es) y Julio (juliom@unizar.es).

References

[FSES14] Jeppe Revall Frisvad, Lars Schjøth, Kenny Erleben, and Jon Sporring. Photon differential splatting for rendering caustics. In *Computer Graphics Forum*, volume 33, pages 252–263. Wiley Online Library, 2014.

[HKWB09] Miloš Hašan, Jaroslav Křivánek, Bruce Walter, and Kavita Bala. Virtual spherical lights for many-light rendering of glossy scenes. *ACM Trans. Graph.*, 28(5), 2009.

[HOJ08] Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. Progressive photon mapping. *ACM Trans. Graph.*, 27(5), 2008.

[JC98] Henrik Wann Jensen and Per H Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 311–320. ACM, 1998.

[Jen96] Henrik Wann Jensen. Global illumination using photon maps. In *Rendering Techniques' 96*, pages 21–30. Springer, 1996.

[Jen01] Henrik Wann Jensen. *Realistic Image Synthesis Using Photon Mapping*. AK Peters, 2001.

[JMMn+14] Adrian Jarabo, Julio Marco, Adolfo Muñoz, Raul Buisan, Wojciech Jarosz, and Diego Gutierrez. A framework for transient rendering. *ACM Trans. Graph.*, 33(6), 2014.

[Kaj86] James T. Kajiya. The rendering equation. In *Computer Graphics (Proc. of SIGGRAPH)*, 1986.

[KD13] Anton S. Kaplanyan and Carsten Dachsbacher. Adaptive progressive photon mapping. *ACM Trans. Graph.*, 32(2), 2013.

[Kel97] Alexander Keller. Instant radiosity. In *SIGGRAPH '97*, 1997.

[KZ11] Claude Knaus and Matthias Zwicker. Progressive photon mapping: A probabilistic approach. *ACM Trans. Graph.*, 30(3), 2011.

[Sil86] Bernard W. Silverman. *Density Estimation for Statistics and Data Analysis*. Taylor & Francis, 1986.