Graphics and Imaging Lab

Path tracing

Informática Gráfica

Grado en Ingeniería Informática

Objetivo

El objetivo de esta tarea es implementar un *path tracer* desde cero [Kaj86, FHF+17][1]. Puede realizarse de forma individual o en grupos de dos estudiantes.

1 Tarea anterior

El *path tracer* se basa en tareas anteriores, de las cuales se utilizará lo siguiente:

- El modelo de cámara pinhole.
- El formato de imagen y el procedimiento de guardado.
- Las primitivas geométricas de esferas y planos infinitos.
- La propiedad de emisión (tripleta RGB) para cada primitiva.
- Cualquier extensión opcional implementada.

Esta tarea consistirá en añadir propiedades de reflectancia, y se usará el algoritmo de *render* para tratarlas adecuadamente. Como en tareas anteriores, el *path tracer* tendrá solo un parámetro específico (definido por línea de comandos): el **número de muestras por píxel**.

Los parámetros de la imagen de salida también se indicarán por línea de comandos: ancho, alto, nombre de archivo y resolución de color (para guardar la imagen como HDR, podéis consultar la tarea de *tone mapping*). El parámetro de resolución de color puede tener un valor predeterminado y debe ser lo suficientemente alto, $> 2^{24}$, para evitar tanto problemas de resolución como artefactos al aplicar *tone mapping*.

2 Modelo de materiales

De la tarea anterior, la propiedad *color* de cada objeto debe tratarse ahora como la **emisión** del objeto (L_e).

[1] `https://jo.dreggn.org/path-tracing-in-production/`

En cuanto a la reflectancia, la BSDF básica del objeto debe definirse como la combinación de:

- Una BRDF lambertiana difusa.
- Una reflectancia especular perfecta de acuerdo con la ley de reflexión total (BSDF delta).
- Una refracción perfecta de acuerdo con la ley de Snell (BSDF delta).

La reflectancia de cada lóbulo se modula mediante un coeficiente espectral específico (RGB): difuso (k_d), especular (k_s) y de transmitancia (refracción, k_t). Estos coeficientes se convertirán en las nuevas propiedades de cada objeto, lo que llevará a la siguiente BSDF completa f_r:

$$f_r(\mathbf{x}, \omega_\mathbf{i}, \omega_\mathbf{o}) = k_d \frac{1}{\pi} + k_s \frac{\delta_{\omega_\mathbf{r}}(\omega_\mathbf{i})}{\mathbf{n} \cdot \omega_\mathbf{i}} + k_t \frac{\delta_{\omega_\mathbf{t}}(\omega_\mathbf{i})}{\mathbf{n} \cdot \omega_\mathbf{i}} \tag{1}$$

donde $\mathbf{x}$ es el punto de intersección, $\omega_\mathbf{i}$ es la dirección de radiancia entrante, $\omega_\mathbf{o}$ es la dirección de radiancia saliente, $\omega_\mathbf{r}$ es la dirección de reflexión perfecta de $\omega_\mathbf{o}$ y $\omega_\mathbf{t}$ es la dirección de refracción perfecta de $\omega_\mathbf{o}$ de acuerdo con la ley de Snell.

Para garantizar la exactitud física, la suma de todos los coeficientes debe ser inferior a 1 en todos los canales RGB. No todos los objetos tendrán las tres propiedades activadas (un objeto con las tres propiedades sería extraño), por lo que algunos de los coeficientes RGB deben ser 0. Algunos ejemplos de materiales pueden ser:

- Un material puramente difuso (sin reflexión especular ni refracción).
- Un material plástico, con reflexión difusa y especular (sin refracción). Los componentes espectrales del coeficiente especular (k_s) deben ser iguales (reflejo especular gris/blanco).
- Materiales dieléctricos, con reflexión especular y refracción (sin difusión). Los coeficientes de reflexión y transmisión deben tener valores iguales para sus componentes espectrales (gris/blanco). Los coeficientes podrían estar mejor modulados por las ecuaciones de Fresnel, pero esto es opcional.

Recuerda que los objetos también pueden emitir luz. Se puede suponer que un objeto emisor no refleja luz y viceversa (un objeto reflectante no emite luz).

3 Luces puntuales e iluminación directa

Primero agregaremos luces puntuales a la simulación. Cada luz puntual tendrá una potencia (tripleta RGB) y una posición específica en el espacio. La escena puede tener tantas fuentes de luz como sea necesario.

Para cada rayo trazado desde la cámara, el algoritmo estimará la iluminación directa lanzando rayos de sombra (*next-event estimation*) hacia todas las fuentes de luz puntuales. Ten en cuenta que la cantidad de fuentes de luz afectará directamente al tiempo de renderizado, a menos que se haga algún tipo de *Ruleta Rusa* entre las fuentes de luz (En este caso, la *Ruleta Rusa* es opcional).

Ten en cuenta que las posibilidades de encontrar una fuente de luz puntual (delta) a partir del BSDF delta son 0. Dada la naturaleza delta de los lóbulos especulares y de transmisión del modelo de material, estos nunca contribuirían a la iluminación directa de las fuentes de luz puntuales. Por lo tanto, (a menos que se implementen BSDF opcionales adicionales), solo el componente difuso de los materiales contribuirá a la iluminación directa desde las luces puntuales.

No es necesario que la geometría se cargue desde un archivo. Por lo tanto, las escenas pueden codificarse directamente (definidas en el código fuente). Una de las escenas (la escena "predeterminada") debería ser una **Cornell Box**: cinco paredes formando una caja (con un lado abierto para que la cámara observe), todas grises excepto la izquierda (roja) y la derecha (verde). Además, debe tener dos esferas en el suelo. La fuente de luz debe ser una luz puntual en la mitad superior de la caja. Puede (y debe) definir más escenas, pero debe usar la Cornell Box para el resto de la tarea.

4 Path tracer

Hasta ahora, el algoritmo solo simula la iluminación directa de las fuentes de luz puntuales. El *path tracer* permitirá una simulación completa del transporte de luz. Las interacciones de los caminos seguirán un muestreo de Monte Carlo con *Ruleta Rusa*. Cada camino tendrá las condiciones de terminación estándar (no chocar con nada, chocar con un objeto sin propiedades de reflectancia o por una condición de terminación de *Ruleta Rusa*). No recomendamos la implementación iterativa del *path tracer*: no importa si es más eficiente, su implementación es más compleja que su versión recursiva.

Debes probar el *path tracer* con la escena de la **Cornell Box**. En este caso, las paredes deben tener materiales completamente difusos, una de las dos esferas debe tener un material similar al plástico y la otra debe tener un material similar a un dieléctrico. La iluminación debe ser una fuente de luz de área en la parte superior de la caja (puede ser toda la pared superior).

Luego, el *path tracer* debe incluir *next-event estimation* desde fuentes de luz puntuales (como se discutió en la sección anterior). Debido a la dificultad de controlar la iluminación, no se recomienda incluir escenas con luces de área y luces puntuales. Sin embargo, deberás comparar el efecto de la estimación del próximo evento con las dos **Cornell Boxes** (luces de área frente a luces puntuales).

Si se logran correctamente todas estas características requeridas (*path tracer* + *next-*

event estimation con luces puntuales), los estudiantes obtendrán una calificación de 7. Para obtener calificaciones más altas, los estudiantes deben realizar cualquiera de las extensiones opcionales.

5 Extensiones opcionales

Nota para nosotros: dejar claro que no se flipen. Hay muchos trabajos con muchas cosas opcionales que acaban sacando un notable por no responder correctamente a las preguntas

Los estudiantes pueden elegir una (o más) de las siguientes extensiones (incluyendo las de tareas anteriores):

Otras geometrías. Agrega diferentes primitivas geométricas definidas por su ecuación implícita, como conos, cilindros, elipsoides, discos o triángulos. Puedes ser creativo aquí.

Estructuras de aceleración. Para escenas con muchos objetos, implementa Bounding Volume Hierarchies (BVHs) o Kd-trees para acelerar el renderizado. Deberás evaluar la mejora en el tiempo de renderizado.

Paralelización. Paraleliza el algoritmo entre varios hilos de ejecución. Usa una estrategia de paralelización a nivel de imagen: divide la imagen en regiones (líneas /rectángulos / columnas /píxeles) y usa una cola de tareas segura para los hilos para configurar las regiones de renderizado. Puedes encolar todas las tareas de renderizado (un único productor) y tener múltiples hilos que retiren tareas y las ejecuten (múltiples consumidores). Prueba diferentes estrategias: diferentes regiones de imagen y tamaños de regiones de imagen, diferentes implementaciones de colas seguras para hilos y diferentes combinaciones entre paralelización y estructuras de aceleración. **Además:** explora estrategias de paralelización más sofisticadas, incluida la utilización de diferentes tipos de consumidores para tareas de alto consumo de tiempo (como separar cálculos de intersección de cálculos de transporte de luz). Las estrategias de paralelización estáticas sencillas no se valorarán en este caso.

Texturas. Agrega texturas que modulen la emisión de un objeto o cualquiera de los coeficientes que modelan su apariencia. También se pueden agregar otras texturas (bump maps o normal maps, por ejemplo), pero son más difíciles de modelar porque requieren un campo tangente. Esto equivale a agregar dependencia espacial a uno (o a todos) los coeficientes del material: $L_e(\mathbf{x})$, $k_d(\mathbf{x})$, $k_s(\mathbf{x})$ y $k_t(\mathbf{x})$.

Renderizado espectral. En lugar de tener propiedades de emisión RGB para cada objeto, incluye una representación espectral del color que se convierte posteriormente en RGB uti-

lizando las curvas RGB CIE estándar. La representación espectral puede ser un vector de 8, 16 o 32 valores, o incluso una función espectral (una función que depende de la longitud de onda) que se muestrea en los valores correspondientes de 8, 16 o 32.

Efectos de Fresnel. Modula el efecto de los coeficientes según la dirección de la vista ω_o, siguiendo las ecuaciones de Fresnel. Esto es especialmente útil para los dieléctricos, pero también puede definir la relación entre especular y difuso en el caso de los plásticos. Esto equivale a agregar dependencia direccional a uno (o a todos) los coeficientes de reflectancia: $k_d(\omega_\mathbf{o})$, $k_s(\omega_\mathbf{o})$ y $k_t(\omega_\mathbf{o})$.

BSDFs. Agrega nuevos modelos de BSDF (BRDF o BTDF) que traten otros fenómenos. Esto debe construirse sobre los efectos de Fresnel e incluir otros lóbulos más sofisticados: lóbulos especulares Phong o Ward, modelos de microfacetas... También se pueden agregar BSDF anisotrópicos, pero requieren más trabajo porque necesitan un campo tangente.

Desenfoque de movimiento. Agrega animaciones al modelo de geometría: las animaciones simples, como las traslaciones a lo largo de una línea recta, serán suficientes. Luego, agrega la dimensión temporal al muestreo e integra a lo largo de un pequeño intervalo temporal (tiempo de exposición).

Profundidad de campo. Amplía el modelo de cámara pinhole a un modelo de cámara que incluye una apertura (agregando dos dimensiones adicionales) y una distancia focal que permita efectos de profundidad de campo (desenfoque debido a la distancia al punto focal).

Muestreo por importancia en *next-event estimation*. En lugar de todas las fuentes de luz puntuales, encuentra una forma de muestrear por importancia su contribución en cada interacción, y tal vez incluir luces de área (objetos emisores) en el *next-event estimation*.

Trazado de caminos bidireccional. Implementa una versión simple del *path tracer* bidireccional en la que los caminos también se generan desde la fuente de luz con las correspondientes sombras.

Medios participativos. Extiende el *path tracer* para que tenga en cuenta medios participativos como la niebla, el humo o el agua, incluyendo eventos de absorción y dispersión.

Otras extensiones. Cualquier otra extensión considerada por los estudiantes también se puede agregar, pero primero deben discutirla con el profesor. En general, antes de emprender cualquier extensión opcional, los estudiantes deben ponerse en contacto con el profesor para que sean aconsejados sobre la mejor manera de abordar la extensión específica, y esta se

puede reducir si supera la carga de trabajo esperada.

6 Detalles de implementación

Lenguaje de programación. Los estudiantes pueden elegir cualquier lenguaje de programación que consideren. Sin embargo, recomendamos C++ debido a lo siguiente:

- El código compilado será más eficiente que en otros lenguajes.
- El código que se proporcionará para el segundo trabajo práctico está en C++ (a menos que elija utilizar su propio código para ambos).
- C++ permite definir operadores como + o * para cualquier nuevo tipo de datos. Esto hace que el código sea más legible para tipos de datos como vectores.

Formatos de archivo. Como se ha utilizado anteriormente en tareas anteriores, recomendamos el formato de imagen `.ppm` para escribir imágenes (en formato similar al HDR, con una gran resolución de espacio de color, como se discutió en tareas anteriores). Puede encontrar los detalles en `http://netpbm.sourceforge.net/doc/ppm.html`. También se puede usar para cargar imágenes como entrada si se incluyen las extensiones opcionales de texturas o mapas de entorno.

Si los estudiantes deciden incluir la extensión opcional de mallas de triángulos y desean cargarlas, recomendamos el formato de archivo `.ply`. Nuevamente, es un archivo de texto muy simple. La descripción de dicho formato, junto con algunos modelos gratuitos en formato `.ply`, se puede encontrar en `https://people.sc.fsu.edu/~jburkardt/data/ply/ply.html`.

Bibliotecas externas. El uso de cualquier tipo de biblioteca externa o código fuente descargado está estrictamente prohibido. A los estudiantes solo se les permite usar la biblioteca estándar que viene con el lenguaje de programación elegido. La única excepción a esta regla es incluir código que sea su propio trabajo anterior (quizás procedente de trabajos anteriores en materias de informática anteriores). Si incluye código que es su propio trabajo anterior, debe documentarlo correctamente (cuándo lo generó, para qué tarea y qué materia).

7 Informe

Nota para nosotros, profesores en el futuro: hacer más hincapié en la importancia de la memoria. Añadir las siguientes "recomendaciones" que yo casi las pondría obligatorias para que no nos entreguen según qué memorias desagradables de corregir:

- Estructura la memoria siguiendo el esquema que te planteamos, dejando claro qué estás respondiendo en cada apartado y por qué. Si es necesario, copia la pregunta que te hacemos. No esperes que busquemos en toda la memoria a ver si has respondido una pregunta concreta.
- Sigue el orden que te planteamos de forma estricta.
- Cuando pongas varias imágenes para comparar, no las pongas a lo loco y ocupes varias páginas. Existe una cosa que se llaman tablas, y en las tablas se pueden poner imágenes.
- Cuando pongas una imagen (o una tabla con imágenes) pon un pie de imagen que la describa.

Como entrega de este trabajo, cada grupo debe escribir un informe sobre el diseño y la implementación del *path tracer*. El contenido de su informe debe estar estructurado con los siguientes puntos:

1. Escribe la **Ecuación de Render** y describe brevemente cada una de las partes. Discute en un par de frases cómo tu *path tracer* tiene en cuenta cada una de esas partes.
2. Muestra ejemplos de una o dos escenas diferentes (por ejemplo, la Cornell Box) con diferentes cantidades de muestras por píxel (por ejemplo, 2, 8, 32, 128 y 512). En cuanto a la **convergencia** de *path tracing*, responde a las siguientes preguntas en uno o dos párrafos:
   a) ¿Con qué rapidez converge *path tracing* con respecto al número de caminos por píxel? Puntos extra si incluyes un análisis numérico.
   b) De los materiales implementados, ¿cuáles hacen que la convergencia sea más lenta? ¿en qué circunstancias? ¿Por qué?
   c) ¿Qué fuentes de luz hacen que *path tracing* sea más lento? ¿Luces de área o luces puntuales? ¿en qué circunstancias? ¿Por qué? Ilústralo con renders.
3. Hay cuatro **efectos de iluminación global** discutidos en clase: sombras duras, sombras suaves, color bleeding y caústicas. En cuanto a estos efectos, responde en uno o dos párrafos:
   a) ¿Cuáles de estos efectos de iluminación global se obtienen fácilmente con *path tracing*? ¿Cuáles de estos efectos de iluminación global no se pueden obtener (o son muy difíciles de obtener en cuanto al tiempo de convergencia)? ¿Por qué? Ilústralos con renders y haz zoom (aumentando la imagen) en áreas específicas.
   b) ¿Qué escenas necesitas para obtener cada uno de estos efectos? Discute los requisitos de las escenas para cada uno de los efectos: propiedades de los materiales, propiedades de las fuentes de luz, distribución geométrica de los objetos...

Además, debes agregar uno o dos párrafos **para cada extensión implementada**:

1. Describe la extensión implementada, incluyendo decisiones de diseño específicas.
2. Relaciona la extensión implementada con los materiales de enseñanza y la bibliografía: ¿se discutió en clase? ¿Cuándo? Si no, agrega referencias bibliográficas de los materiales que te ayudaron a implementar la extensión.
3. ¿Cómo mejora tu extensión la versión básica del *path tracer*? Ilustra con renders lado a lado, compara los tiempos de renderizado...

Además del contenido requerido, el informe debe incluir:

- Un análisis de carga de trabajo: ¿cómo habéis gestionado vuestro propio trabajo, qué tareas ha emprendido cada miembro del grupo, cómo habéis compartido y mejorado el código, qué metodología habéis utilizado...
- Un conjunto de renders bonitos. Al menos dos de ellos deben ser una Cornell Box (una con una luz de área y la otra con una luz puntual) con iluminación indirecta.

No hay requisitos con respecto al número de páginas del informe, pero todos los informes deben seguir la estructura dada. Además, tenemos las siguientes recomendaciones para escribir:

- Sé **conciso** en sus explicaciones. Evita redefinir términos conocidos o añadir redundancias en el texto. Recuerda que divagar normalmente suele confundir al lector.
- Al mostrar renders, **discútelos**. ¿Qué ilustran los renders? ¿Qué fenómeno es relevante? Esto se puede hacer en la leyenda de la figura.
- Puede ser útil **comparar renders** uno al lado de otro para ilustrar una característica específica (con y sin ella).
- **No incluyas código fuente**. Ya estás enviando tu 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 línea delgada entre "inspirarse" y "copiar y pegar código".

La calidad del informe también se tendrá en cuenta en la evaluación. Un informe más largo no necesariamente equivale a un mejor informe.

8 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 *path tracing* debe estar comprimido en un archivo `.zip`, con el siguiente nombre de archivo:

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

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

References

[FHF+17] Luca Fascione, Johannes Hanika, Marcos Fajardo, Per Christensen, Brent Burley, and Brian Green. Path tracing in production - part 1: Production renderers. In *ACM SIGGRAPH 2017 Courses*, SIGGRAPH '17, pages 13:1–13:39, 2017.

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