

Práctica 3: Path tracer

Ana Márquez Moncada

Índice

1. Explicación del código	2
2. Resultados y pruebas	3

En esta práctica se implementa un nuevo integrador, el path tracer, que resuelva la iluminación global. El integrador *pathtracer* realiza solo iluminación indirecta utilizando la ruleta rusa como criterio para evitar la recursividad infinita. El integrador *pathtracer_nee* genera 2 rayos en cada rebote, uno para samplear un emisor, y el otro recursivo sampleando al BSDF. El objetivo es crear un path tracer con luz directa e indirecta.

1. Explicación del código

Primero comentaré un poco el funcionamiento del path tracer. En el archivo *path.cpp* está implementado un path tracer que se encarga de la luz indirecta. Lo que se hace es trazar un rayo desde la cámara y samplear la BSDF del objeto colisionado. Después, desde el punto de impacto en el objeto se samplea una nueva dirección y se traza un rayo en esa dirección. El proceso se repite hasta que el path se salga de la escena o impacte con un material emisor.

Para los rebotes indirectos, utilizo una función recursiva. La información que se quiere conservar del path, la guardo en un struct llamado *PathInfo*, que guarda la profundidad del path y la energía que le queda. Para evitar que la recursividad no sea infinita, se pone una condición de terminación que depende de la profundidad del path (*pathInfo.depth*).

También se ha implementado la técnica de la ruleta rusa, que elimina rayos al azar, con más probabilidad de eliminar los rayos que menos iban a aportar a la escena con un nuevo rebote. Es decir, elimina rayos al azar, siendo más probable eliminar rayos que hayan recorrido más distancia. Se utiliza para ello una variable que guarda la energía que le queda al path, *pathInfo.pathThroughput*. Esta variable se actualiza en caso de que el rayo no se haya eliminado, dividiendo por la probabilidad.

Para calcular la probabilidad, utilizo este *pathThroughput*, inicializado a 1, que voy multiplicando por el material en cada rebote, además de tener en cuenta las probabilidades. De esta forma se tiene en cuenta el aporte que un rayo iba a tener en la escena final.

En resumen, 4 casos base en esta función recursiva, que finalizan la función y devuelven el color. El primero es si el rayo no choca con nada. El segundo es que la depth del pathThroughput sea mayor que un *MAX_PATH_LENGTH*. En ambos casos se devuelve color negro (0). El tercero es si el rayo choca con un emitter, que se devuelve el color del emitter. El último es el de la ruleta rusa, que se devuelve el color del rayo.

Ahora hablaré del funcionamiento del path tracer con luz directa e indirecta, *pathTracer_nee*. En este caso, el integrador generará en cada rebote un rayo recursivo como en el anterior integrador, que corresponde con la iluminación indirecta, y otro para samplear el emisor como en la práctica 2, que corresponde con la directa.

Para ello calculo 4 pdfs, 2 relacionadas con la luz directa y 2 con la indirecta, y a partir de ellas calculo los pesos y el color final en la función *Li*.

Primero, a esta función recursiva le añado el cálculo de la luz directa, que se añadirá a la luz final multiplicada por su peso. Ya internamente en esta función se calcula el peso. También se multiplica por el *pathThroughput*, para ponderarla según su influencia en la escena.

Para esta luz directa, el cálculo de las pdf lo hago a partir de lo que implementé en la práctica 2 para el cálculo de la pdf, utilizando *its2.mesh->getEmitter()->pdf(lRec)*. y para la del material, utilizo *its.mesh->getBSDF()->pdf(brecM)*. Una vez tengo las pdf, calculo el peso como *weight = pdfLuz/(pdfMat + pdfLuz)* en caso de que *(pdfMat + pdfLuz) > 0*, en otro caso la pdf es 0.

Para la luz indirecta, hay que tener en cuenta el rayo recursivo. Para calcular las pdf lo hago de la misma forma. Para sacar la pdf del material, utilizo *its.mesh->getBSDF()->pdf(bRec)*. Para sacar la pdf de la luz, creo una nueva intersección a partir del rayo recursivo y, si choca con un emitter creo un nuevo *EmitterQueryRecord* con los nuevos valores del rayo y calculo la pdf mediante *its2.mesh->getEmitter()->pdf(lRec2)*. Si no choca con un emitter, la pdf del emisor vale 0. Una vez tengo las pdf, calculo el peso como *weight = pdfMat/(pdfMat + pdfLuz)* en caso de que *(pdfMat + pdfLuz) > 0*,

en otro caso la pdf es 0. También hay que tener en cuenta que si la BSDF en la que ha rebotado el rayo es discreta, el peso es 1. Esto es por el material espejo.

Una vez tengo los pesos, multiplico cada peso por su correspondiente color (luz directa o indirecta), y sumo ambos para hallar el color final.

2. Resultados y pruebas

En esta sección incluiré los resultados tras realizar varias pruebas con distintas características. Las imágenes obtenidas son con 64 samples salvo las que se indica otra cosa.

1) Sólo iluminación directa:

En esta primera escena se utiliza sólo iluminación directa, que equivale al resultado de la práctica 2 (figura 1).

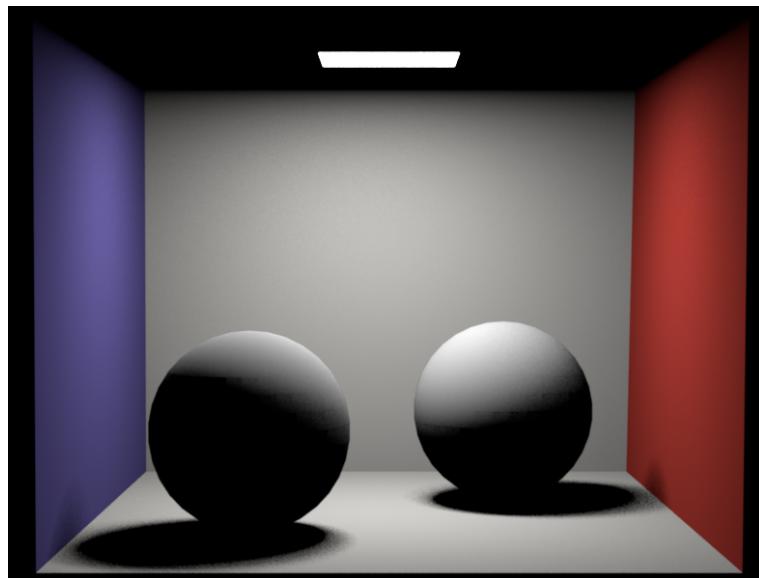


Figura 1: Sólo iluminación directa

2) Sólo iluminación indirecta:

En esta escena se utiliza sólo iluminación indirecta, resultado del apartado A de la práctica. Como se observa en las esferas, hay sangrado de color (figura 2).

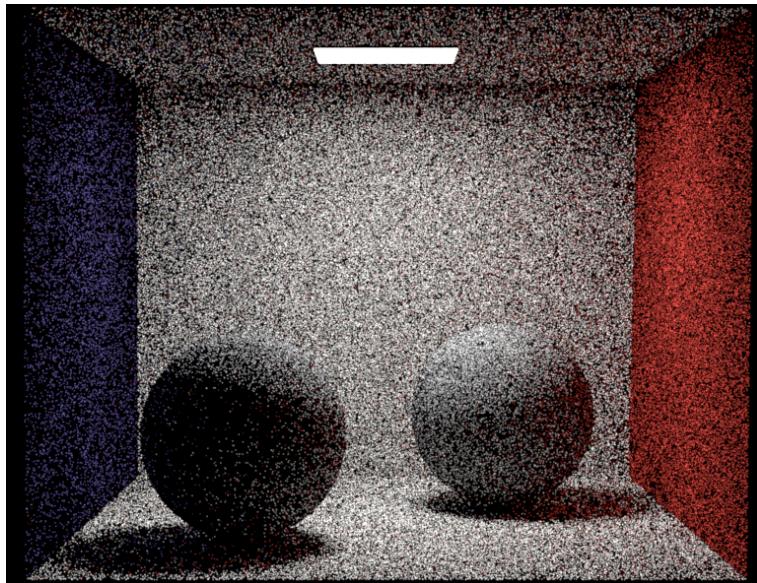


Figura 2: Sólo iluminación indirecta

3) Luz directa e indirecta:

En esta escena se utiliza luz directa e indirecta, ponderadas ambas por su peso. Se puede observar cómo afectan ambas iluminaciones, se nota la presencia de la luz directa además de haber sangrado de color por influencia de la indirecta, y sombras de la influencia de ambas (figura 3).

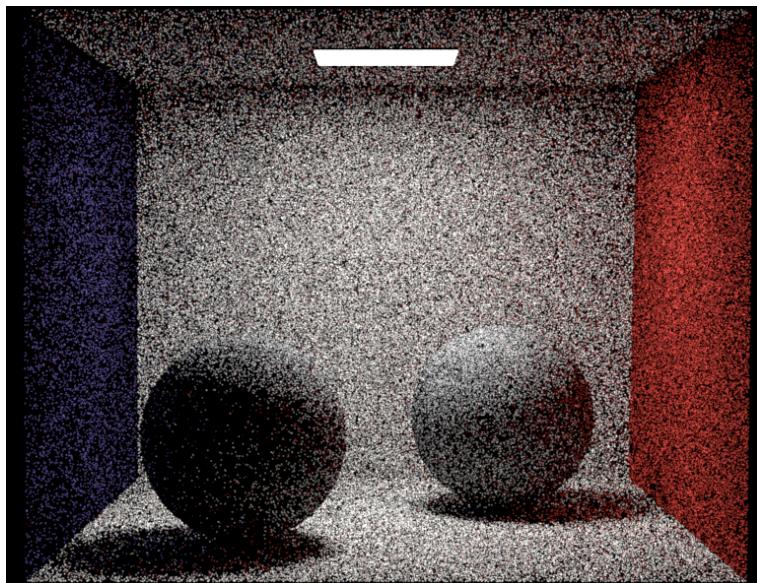


Figura 3: Iluminación directa e indirecta

4) Material espejo:

En esta escena se ha cambiado el material de las esferas a espejo. Con este path tracer se puede observar los efectos del espejo, ya que tiene en cuenta los rebotes gracias a la iluminación global (figura 4) .

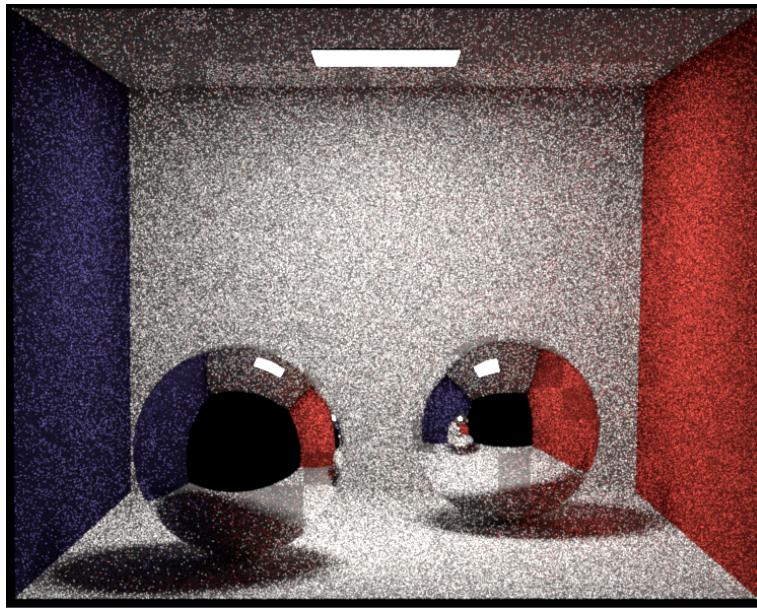


Figura 4: Esferas con material espejo

5) Utilizando más samples:

En esta escena se ha renderizado la imagen con 256 samples. Ha tardado bastante más en renderizarse, y debería verse mejor, sin embargo, se puede observar que se producen algunos artefactos. Esto no se por qué se está produciendo, ya que en las anteriores imágenes que rendericé no ocurría. (figura 5).

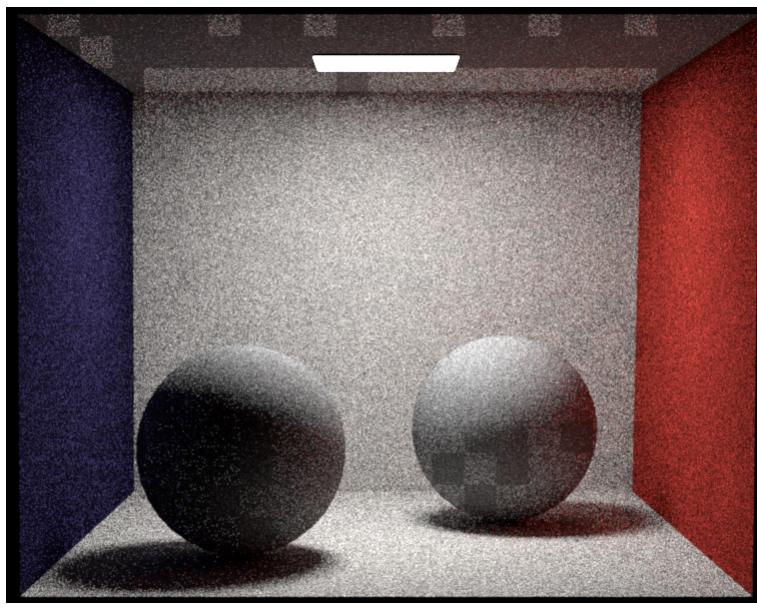


Figura 5: Imagen con 256 samples

6) Probabilidad fija de la ruleta rusa:

En esta escena se ha probado utilizar un valor de probabilidad fijo de la ruleta rusa (0.05), que no dependa del camino del rayo. Como se observa el resultado es bastante peor. Al no tener en cuenta la importancia del rayo en la escena, elimina rayos al azar y puede que elimine alguno más importante (figura 6).

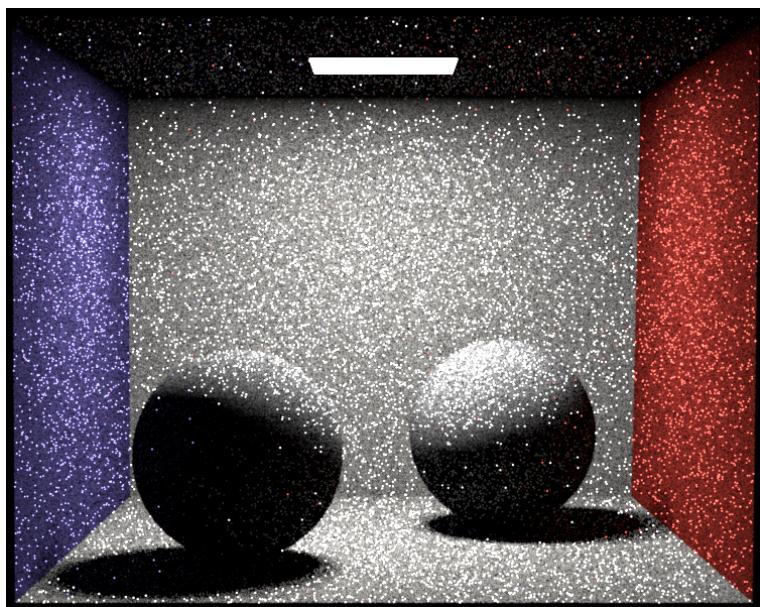


Figura 6: Ruleta rusa fija