Rendering Avanzado

Path Tracing

Álvaro Muñoz Fernández
Iván Velasco González

1. Path Tracing

En esta parte de la práctica se debían implementar varios algoritmos de *Path Tracing* que se detallarán en las secciones a continuación.

1.1. Naive Path Tracing

En esta sección se pedía implementar un método de *Path Tracing* básico, teniendo en cuenta únicamente la BRDF de los materiales. Por lo tanto, el algoritmo es muy similar al realizado en la práctica anterior (*direct_mats*), con la salvedad de que no se limitará la profundidad del camino a un único rayo, sino que podrán tener más profundidad, consiguiendo más rebotes por la escena y, por tanto, iluminación global.

Debido a las similitudes con el algoritmo de *direct_mats* se decidió utilizar este como base. En primer lugar se añadieron una serie de variables: L, la cual almacena la iluminación total calculada para un punto en concreto y que es inicializada a 0; W, que almacena todos los multiplicadores, como la brdf, pdfs, etc. que se irán acumulando tras los sucesivos rebotes, siendo inicializada a 1; y por último, una variable que representa el rayo que se está tratando, *mRay*, y que se inicializa con el primer rayo trazado desde la cámara.

Seguidamente se ha implementado un bucle que se encargara de computar la iluminación en sí. En primer lugar se comprueba si el rayo que se esta trazando en ese momento interseca con la geometría de la escena. De no hacerlo, se suma a la iluminación total del punto la iluminación proporcionada por el ambiente multiplicada por todas las interacciones de la luz (W) y se devuelve la iluminación total calculada. En el caso de que el rayo sí interseque con geometría de la escena, se comprueba si ha intersecado con una fuente de luz y, de hacerlo, se suma la iluminación de esa fuente de luz multiplicada por W a la iluminación total en el punto.

Posteriormente se comprueba si se debe de seguir trazando rayos por el método de ruleta rusa, comprobando si un número aleatorio es menor que la probabilidad de que continue el rayo, definida como el maximo valor de color que tenga actualemente la variable W, actuando como una medida de la aportación que realizarán los siguientes rayos. En este caso se termina de iterar el bucle de iluminación y se devuelve la iluminación calculada hasta el momento. En caso de que se deba seguir trazando rayos, se muestrea la BRDF del material con el que se ha intersecado y se calcula el nuevo rayo a utilizar ($mRay$). Además, se actualiza el valor de W con las características de este nuevo rebote de la siguiente forma:

$$W* = \frac{brdf * cos\theta}{pdf_{dir} * pdf_{surv}}$$

Donde θ es el ángulo entre la normal del punto a tratar y el ángulo de salida del rayo, y pdf_{surv} es la probabilidad de supervivencia de un rebote. Este último término es necesario añadirlo debido a la utilización del método de ruleta rusa para terminar con los rebotes de la luz.

Además del metodo de terminación por ruleta rusa tambien se ha implementado la terminación por un número fijo de rebotes, el cual es configurable y deberá ser un valor elevado para que en la mayoria de los casos actúe la ruleta rusa, siendo su finalidad la de evitar bucles casi infinitos si la probabilidad de terminación por ruleta rusa es excesivamente baja.

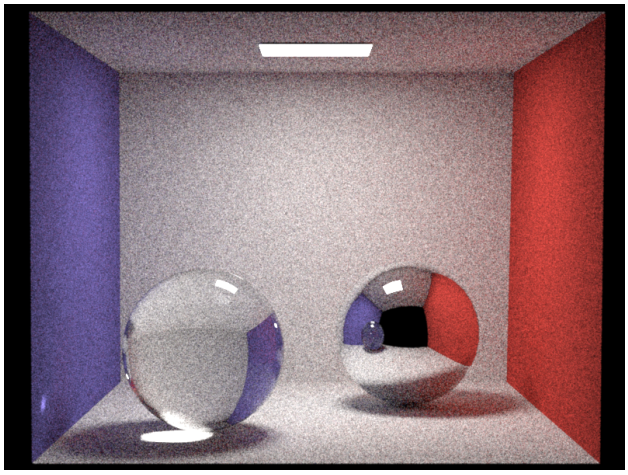

Figura 1: Imagen generada utilizando *Path Tracing* simple (512 muestras)

Como puede verse en la imagen se han conseguido varios efectos de iluminación global, como son los reflejos en la esfera de espejo (derecha) o la cáustica en la esfera que representa un cristal (izquierda), además de un ligero *color bleeding* en el techo proveniente de las paredes. Sin embargo, se aprecia una gran cantidad de ruido tanto en las paredes como en los reflejos.

Por otro lado, la convergencia de la imagen no es muy buena debido a que solamente los caminos que intersequen con una fuente de luz contribuirán a la iluminación final. Por lo tanto, una gran cantidad de muestras se desperdician al no encontrar la luz.

1.2. Path Tracing with Next-Event Estimation

En esta parte de la práctica se va a implementar un *Path tracer* algo más sofisticado que resuelva algunos de los problemas de la versión anterior. En concreto, trata de resolver el problema de convergencia que provocan los caminos que no intersecan con ninguna luz y que, por tanto, no aportan nada a la escena. Para resolver esto se va a calcular en cada rebote la iluminación directa que recibe ese punto desde una fuente de luz, consiguiendo con ello que cada rebote tenga, en la mayoría de los casos, un mínimo de aporte de luz garantizado haciendo que todos los rebotes contribuyan a la iluminación, lo que mejora la convergencia. Además se siguen trazando los rayos impuestos por la BRDF para conservar la iluminación global.

Para realizar esta tarea se ha partido del código implementado en la sección anterior, pero se ha añadido un nuevo bloque de código que para cada rebote muestrea una fuente de luz aleatoria y calcula su aportación a la luz del punto a tratar como:

$$L+ = \frac{Li * brdf_{nee} * V * W * \cos\Theta}{pdf_{dir} * pdf_{light}}$$

Donde Li es la iluminación directa que llega al punto desde la fuente de luz muestreada; $brdf_{nee}$ es el término BRDF teniendo en cuenta las direcciones del rayo de entrada y del rayo de salida (el que va hacia la luz); V es el termino de visibilidad, ya que como hemos muestreado la luz esta podría estar ocluida; y por último, pdf_{light} y pdf_{dir} son las PDF de la luz muestreada y del punto en concreto muestreado, respectivamente.

Adicionalmente hay que tener en cuenta que al incluir un rayo directo a la luz puede darse el caso de que para un mismo punto se sume la aportación de una fuente de luz dos

veces, una al muerstrear esa fuente de luz y la otra si el rayo de la BRDF interseca con esa misma fuente de luz. Para evitar estos casos se ha añadido un *flag* que indica si se debe contar la iluminación si la BRDF cae en una fuente de luz o no, mientras que la aportación de *Nee* siempre se tiene en cuenta. Debiendo contar con la iluminación de la BRDF si es el primer rebote, para evitar que las fuentes de luz se vean negras o si el material es completamente especular, es decir, de tipo *EDISCRETE*, ya que este tipo de materiales no pueden muestrearse utilizando iluminación directa (su PDF devuelve siempre 0).

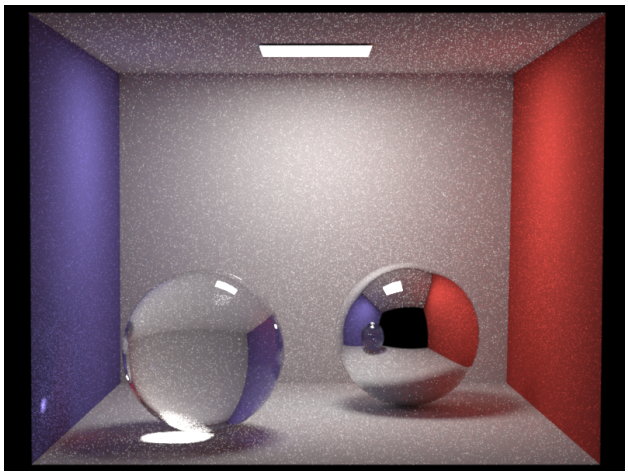

Figura 2: Imagen generada utilizando *Path Tracing* con *Nee* (128 muestras)

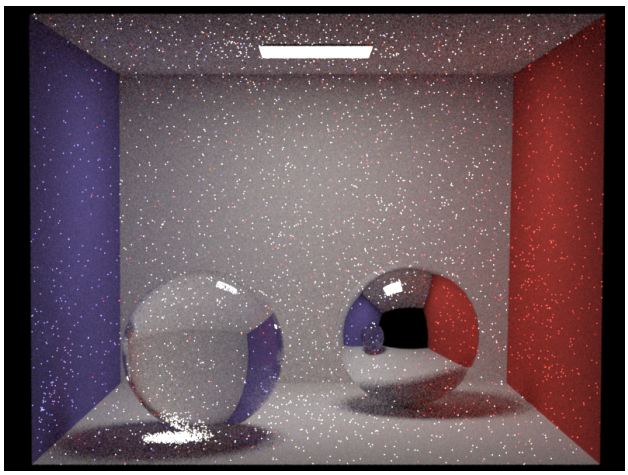

Figura 3: Imagen generada utilizando *Path Tracing* con *Nee* (8 muestras)

Como puede observarse en la imagen, se ha conseguido el resultado esperado obteniendo muchísima mejor convergencia con un numero muy inferior de muestras para esta escena, donde ha mejorado tanto la iluminación de las paredes como los reflejos. Sin embargo, sigue habiendo problemas con el ruido de alta frecuencia ocasionado por los rebotes especulares y las refracciones de los materiales, que puede observarse claramente al disminuir el numero de muestras. Este problema podría atenuarse en gran medida utilizando *Multiple Importance Sampling*.

1.3. Interesting Image

En esta imagen de ejemplo se pueden observar la mayoría de efectos que se peuden conseguir utilizando *Path Tracing* con *Next-Event Estimation* utilizando nuestra implementación.

Figura 4: Imagen generada utilizando *Path Tracing* con *Nee* (512 muestras)

En esta imagen pueden observarse varios efectos de iluminación global bastante complejos, como una triple interfaz de dieléctrico, que dificulta enormemente el encontrar un camino de luz con la cámara. Al igual que cáusticas muy dispersas producidas por la copa. Por otro lado, se puede observar cómo con un número de muestras conservativo el resultado no presenta apenas ruido, y aun con la dificultad de encontrar caminos, el agua de la copa presenta un aspecto muy plausible. Sin embargo, para conseguir esto se ha aumentado el numero maximo de rebotes del valor previo de 5 para las imagenes anteriores, a 50 para la generación de las actuales.

Por otro lado, se ha notado una reducción del tiempo de renderizado de en torno a un cincuenta por ciento al incluir la ruleta rusa en el algoritmo, con unos resultados visualemente casi idénticos. Además, gracias a este método el algoritmo computa un número de rebotes más alto en los dieléctricos (ya que su brdf es 1) que en las superficies lambertianas, siendo un resultado ideal al lanzar más rebotes donde se necesitan.