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 verán mas adelante es subsiguientes secciones.

1.1. Naive Path Tracing

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

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

Seguidamente, se ha implementado un bucle que se encargara de computar la iluminación en si. En primer lugar, se comprueba si el rayo que se esta trazando en ese momento interseca con la geometría de la escena, si esto no es así 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 si interseque con geometría de la escena, se comprueba si ha intersecado con una fuente de luz, si esto es así se suma a la iluminación total en el punto la iluminación de esa fuente de luz multiplicada por W. Seguidamente se samplea la BRDF del material con el que se ha intersecado, y se calcula el nuevo rayo a utilizar ($mRay$). Ademas, se actualiza el valor de W con las características de este nuevo rebote de la siguiente forma:

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

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

Como condición de parada del algoritmo se ha optado por un metodo basado en profundidad, en el cual se indica al integrador cuantos rebotes se deben tener en cuenta para el calciulo de la iluminación.

Como puede verse en la imagen, se han conseguido varios efectos de iluminación global, como los reflejos en la esfera de espejo o la caustica en la esfera que representa un cristal o su transparencia en si, ademas de un poco de *color bleding* 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.

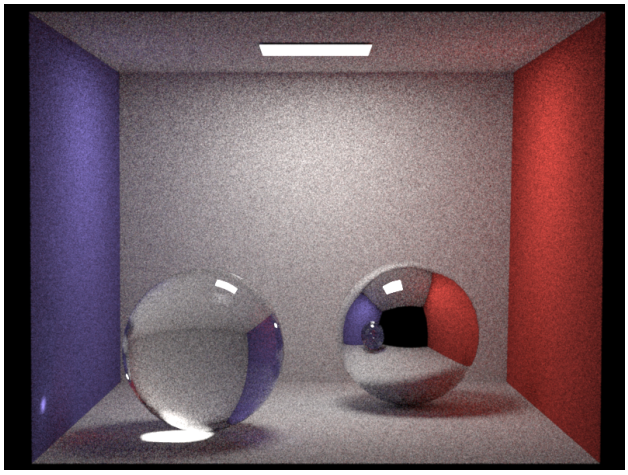

Figura 1: Imagen generada utilizando *Path Tracing* simple

1.2. Path Tracing with Next-Event Estimation

En esta parte de la práctica se va a implementar un *Path tracer* algo mas 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, con esto se consigue que cada rebote tenga, en la mayoría de los casos, algo de aporte de luz garantizando haciendo que todos los rebotes contribuyan a la iluminación , lo que mejora la convergencia. Ademas, 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 *samplea* 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 sampleada; $brdf_{nee}$ es el termino brdf teniendo en cuenta la dirección del rayo de entrada, y al dirección del rayo de salida, el que va hacia la luz; V es el termino de visibilidad, ya que, como hemos sampleado al luz esta podría estar ocluida y , por ultimo pdf_{light} y pdf_{dir} son las pdf de la luz sampleada y del punto en concreto sampleado respectivamente.

Ademas, 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 samplear 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; y si el material es completamente especular, es decir de tipo *EDISCRETE*, ya que este tipo de materiales no pueden samplearse utilizando iluminación directa (tanto su brdf como pdf devuelven siempre 0).

Como puede observarse en la imagen, se ha conseguido el resultado esperado obteniendo muchísima mejor convergencia a igual numero de muestras para esta escena, donde

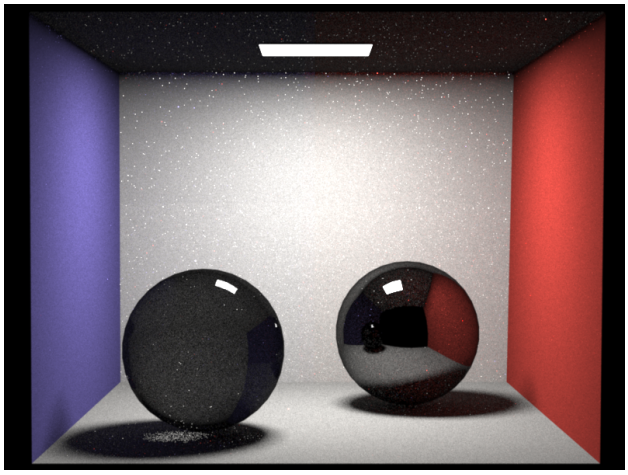

Figura 2: Imagen generada utilizando *Path Tracing* con *Nee*

ha mejorado tanto la iluminación de las paredes como los reflejos. Sin embargo, se sigue pudiendo observar ruido de alta frecuencia provocado por los rebotes especulares y las refracciones del dielectrico. Estos problemas podrian solucionarse utilizando *Multiple Importance Sampling*.

1.3. Interesting Image

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

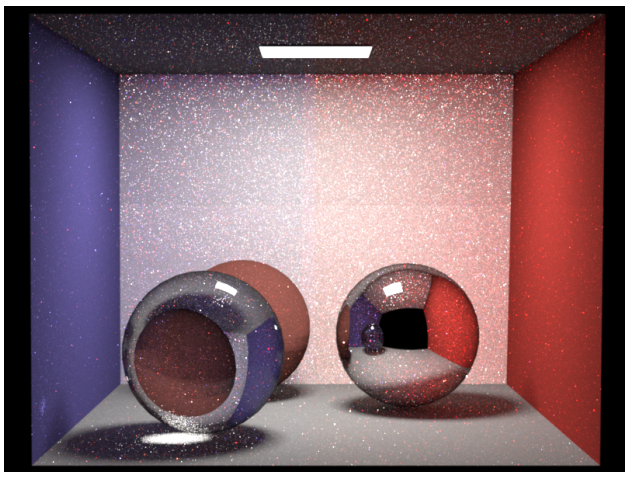

Figura 3: Imagen generada utilizando *Path Tracing* con *Nee* y un valor bajo de absorción

Para la generación de esta imagen se ha utilizado un nivel mas bajo de absorción (modelado por la probabilidad de sobrevivir a la ruleta rusa), para conseguir que la esfera del dielectrico se viera mas clara y ademas la caustica fuese mas acentuada. Sin embargo, al bajar la absorción nos encontramos con un exceso de *color bleeding* en la pared del fondo. Esto es debido a que en nuestra implementación se utiliza la misma probabilidad de sobrevivir a un rebote para todos los materiales de la escena. Sin embargo, esto en la realidad no es así y cada material tienen un nivel de absorción distinto. Por ejemplo, el nivel

de absorción del cristal seria muy inferior al de las paredes, y por tanto si obtuviésemos la probabilidad de sobrevivir a un rebote del material en cuestión , en lugar de una constante global, se podrían conseguir escenas mas realistas, pudiendo eliminar el *color bleeding*, sin oscurecer las esferas.