Informática gráfica

Path tracing

Patricia Briones Yus - 735576
Diego Caballé Casanova - 738712

Report

Para definir las figuras hemos creado una clase base llamada geometryRGBfigures que contiene todo lo necesario para conocer el tipo de material de la figura, el color y si es foco o no. A esta clase le heredan otras clases como Plane y Sphere, que definen las formas de plano y esfera e implementan funciones virtuales de geometryRGBfigures como por ejemplo si se da la intersección de esa figura con un rayo.

Hablando de los campos, geometryRGBfigures tiene una tupla kdTuple con los porcentajes de cada uno de sus colores, además de las probabilidades kd ks y kt según el tipo de material del objeto, el tipo de material del objeto se define usando las funciones esDifuso, esDielectrico y esEspecular. Para el caso de dieléctrico, las probabilidades están predefinidas a ks=0.4 y kt = 0.6, por último, se añadío un campo foco para definir las figuras que son focos y emiten luz.

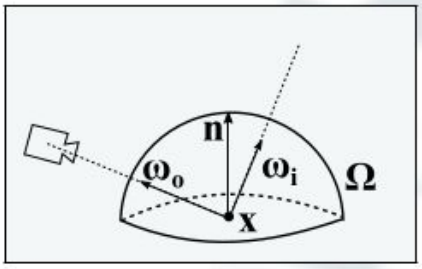
Para calcular los colores finales de un píxel, se hace uso de la función reboteCamino. Esta devuelve el valor del color de la figura, y cambia la dirección del camino. En caso de que el rayo generado se haya elegido como absorbido o el rayo inicial no haya intersectado con ningún objeto, el color del pixel final será negro.
Haciendo más incidencia en la función reboteCamino, se ha implementado la ruleta rusa, que calcula de manera aleatoria si se va a devolver el color de la parte del objeto difusa, especular o dieléctrica, además, se calcula la dirección del rayo rebote dependiendo de la parte elegida, esto es, usando la función cosineSampling, respetando la ley de reflexión o usando la ley de Snell.
Para la ruleta rusa, se ha añadido un nuevo atributo a la clase rayo, prAbs que lleva la cuenta del porcentaje para un no evento como criterio de parada. Cada rebote aumenta un 0.05 esa probabilidad.

En cuanto a la opinión y análisis del trabajo realizado, se considera muy interesante saber todo el proceso y lógica que hay detrás de un pathtracer de este estilo. Para el avance del trabajo, además de haber investigado por separado algunos puntos, se han realizado reuniones de equipo mediante la aplicación Discord e implementado código compartiendo pantalla. La organización del código se ha realizado mediante un repositorio compartido en github, con commits bien definidos y explicando cada avance del código. En cuanto al reparto del trabajo, esta ha sido bastante equitativa, debido a las reuniones entre ambos miembros del equipo comentadas anteriormente, aunque haciendo Diego más incidencia en la parte de cómo calcular el color final de un objeto mediante pathtracing y Patricia de cómo calcular las direcciones de los rebotes.
Comentar que para poder transformar el fichero ppm generado por el algoritmo y poder visualizar la imagen creada, se ha hecho uso de la página web https://jinaconvert.com/es/convert-to-jpg.php.

Preguntas pathtracer

Write and describe the Render Equation and discuss how does your path tracer accout for it.

Esta es la ecuación de render:



$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_\Omega L_i(\mathbf{x}, \omega_i) f_r(\mathbf{x}, \omega_i, \omega_o) |\mathbf{n} \cdot \omega_i| d\omega_i$$

En ella vemos como la radiancia espectral que recibimos del rayo inicial es igual a la radiancia que nos devuelve el emisor más la integral de la radiancia de todos los rayos de la semiesfera.
La implementación de esta ecuación nos obliga a que, en cada intersección, tenemos que generar todos los rayos de la semiesfera para conocer, realmente, el color del punto x.

El funcionamiento del algoritmo path tracer está basado en esta idea de la ecuación de render, pero simplificando la integral. En vez de generar todos los rayos de la semiesfera, se genera un camino, un camino que colisiona con el punto x y rebota con una dirección wi, elegida de forma uniforme sobre todas las posibles direcciones que deja la semiesfera.
De esta manera, conseguimos que la integración de todos los rayos para conocer el color de x se reduzca a el valor que nos va a dar un rayo en ese punto, que continuará rebotando, hasta llegar a un emisor.

3. Regarding the global illumination effects (hard shadows, soft shadows, color bleeding,caustics):
a) Which of these global illumination effects are easily obtained with path tracing?Why?
Color bleeding y soft shadows deben de ser sencillos de obtener. En las partes de soft shadows, el número de caminos absorbidos aumenta, oscureciendo la imagen ligeramente. Color bleeding se consigue fácilmente por la acumulación del throughput de cada rebote de los caminos.
c) Which of these global illumination effects cannot be obtained (or are very hard to obtain regarding convergence time)? Why?
Cáusticas y hard shadows será complicado, con una buena iluminación y el next event estimation, es complicado que la mayoría de caminos acaben absorbidos como para obtener una sombra oscura. Por otro lado, las cáusticas, como hemos visto en clase, al rebotar entre espejos, muchas veces dejaban puntos negros de rayos absorbidos que no han rebotado hacia la luz, generando huecos en dos espejos colocados uno en frente del otro.