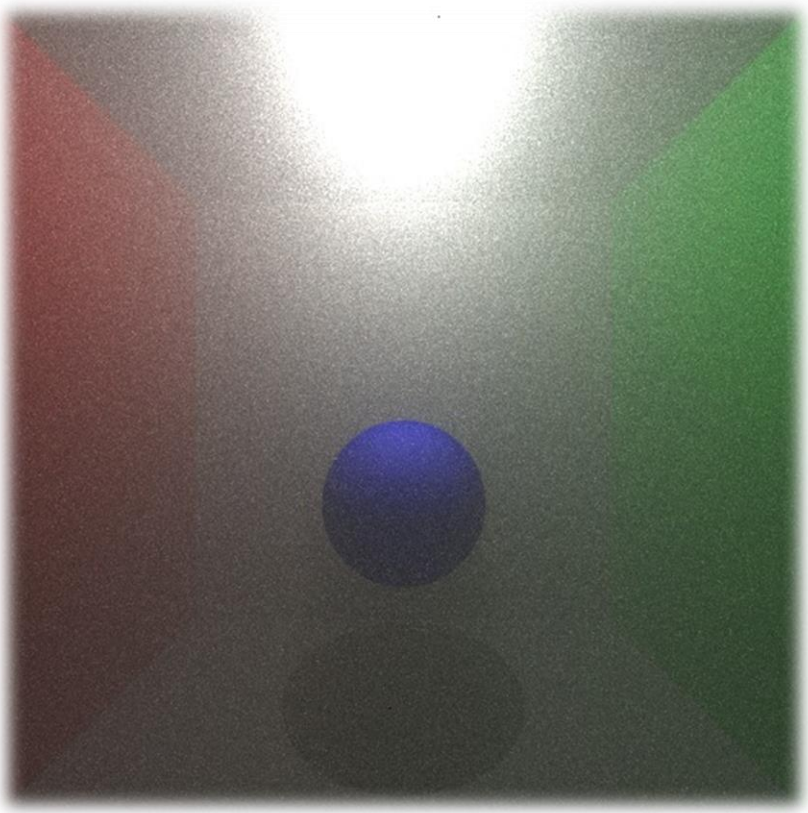
PATH TRACING

Informática Gráfica



1 DE FEBRERO DE 2019

Álvaro Fuentemilla Martínez, 699678
David Márquez Calavia, 700940
Víctor Labrador Ortiga, 701658

Ray tracing

Para la implementación del path tracing primero se ha necesitado implementar el ray tracing. Este trazador de rayos **se compone** de un conjunto de objetos y de la cámara desde donde se lanzan los rayos inicialmente. El trazador de rayos funciona de forma que, las escenas se generan lanzando rayos desde la cámara, uno por cada píxel de la imagen a generar. Una vez lanzado el rayo, si colisiona con un objeto, dicho píxel tendrá el color del objeto con el que ha colisionado, si no, el pixel será de color negro. De esta forma se obtienen escenas sin profundidad (escenas planas, ya que, al tener iluminación, no se generan sombras que den esa profundidad) tal y como se muestra en el ejemplo de la figura 1 donde no se consiguen apreciar las esquinas del paralelepípedo.

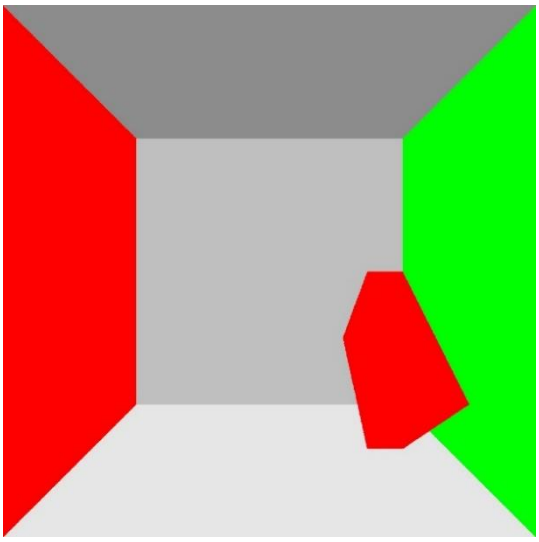


Figura 1. Escena generada con Ray Tracing.

La **cámara** se compone de su eje x, y, z junto al punto p que forma el punto de coordenadas donde se sitúa la cámara. Además, están los valores x e y que marcan el tamaño de la escena.

Por otro lado, están **las geometrías** que se han añadido entre las cuales se destacan las dos obligatorias que son las esferas y los planos. Para ello, se ha decidido crear una clase que es "forma" de la cual se heredan el resto de los atributos. A partir de aquí, se explican las geometrías creadas una por una:

- **Esfera**: La esfera se compone de un tipo de dato punto (clase creada con sus atributos x, y, z) donde se encuentra su centro y el radio (r) de la esfera. A partir de estos puntos y el radio se tiene la ecuación de la esfera:

$$x^2 + y^2 + z^2 = r^2$$

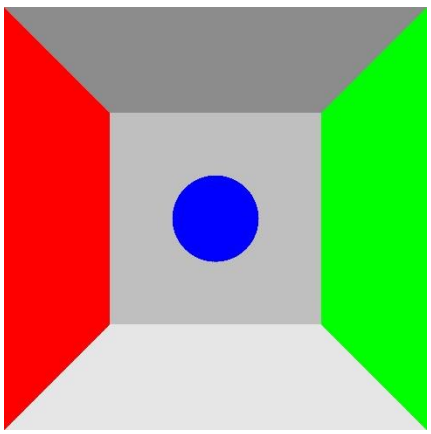


Figura 2. Esfera generada con Ray Tracing.

Ahora se reconvierte la ecuación anterior donde los puntos x, y, z son el punto P y se obtiene:

$$P^2 - R^2 = 0$$

Si tenemos el punto origen (o) del rayo, su dirección (d) y la distancia recorrida por el rayo (r) se tiene la ecuación del rayo donde q es el punto final del rayo:

$$q = o + d \times r$$

Ahora, sustituimos el punto P por la ecuación del rayo y se obtiene una ecuación de segundo grado:

$$\boldsymbol{o^2 + (d \times r)^2 + 2 \times o \times d \times r - R^2 = 0}$$

Si para esta ecuación no se obtiene ninguna solución para el valor r, entonces el rayo no corta a la esfera. Si se obtiene una solución, el rayo es tangente a la esfera y finalmente, si se obtienen dos soluciones, el rayo corta a la esfera (las dos soluciones significan por donde entra y por donde sale el rayo al atravesar la esfera, pero solamente nos interesa el más cercano a la cámara ya que será el que sea vea). Se puede ver el resultado de una esfera en la figura 2.

- **Plano**: El plano se compone de un vector que es su normal (n) y de su punto de origen (p). Entonces si un nuevo punto q pertenece al plano se cumple:

$$\boldsymbol{(q - p) \times n = 0}$$

Se sustituye la ecuación del rayo obteniendo:

$$\boldsymbol{(o + d \times r) \times n = 0}$$

Ahora, en lugar de trabajar con un punto q, se sustituye por la ecuación del rayo y de esta forma se puede encontrar el punto de colisión entre el rayo y el plano.

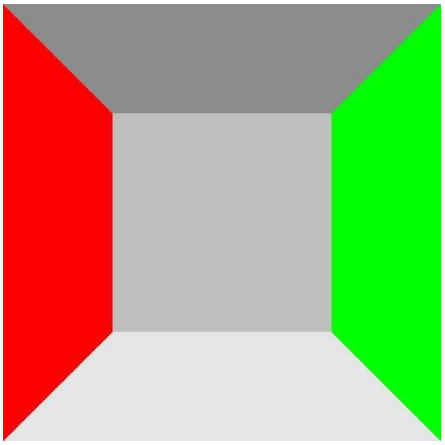


Figura 2. Cornell Box con Ray Tracing.

Ahora se despeja la r y se obtiene la distancia desde el origen del rayo hasta el plano:

$$\boldsymbol{t = (p - o) \times n \div (d \times n)}$$

El único caso donde el rayo no corta al plano ocurre cuando el rayo es paralelo donde:

$$\boldsymbol{r \times n = 0}$$

Como los planos son infinitos, se ha montado la “Cornell box” que se muestran en la figura 3.

Además de estas geometrías obligatorias, se han decidido añadir las siguientes geometrías para poder así obtener unas escenas más complejas:

- **Disco**: Para este caso, un disco se compone de un plano al cual se le añade un radio para fijar así su tamaño. Como ya se ha explicado la colisión que ocurre al trazar el rayo para el plano, para este caso se hace igual, pero una vez que se ha obtenido la colisión (si es que ocurre y el rayo no es paralelo al disco) se obtiene

la distancia que hay desde el punto de corte con el plano hasta el centro del disco y si ésta es menor que el radio que se ha fijado al inicio para delimitar su tamaño, el punto pertenece al disco, en caso contrario, no.

La explicación gráfica se ve en las figuras 4 y 5 donde están el plano y el disco respectivamente. La forma de crearlos es igual, sólo que al disco se le añade el radio "6" y, por lo tanto, los puntos sólo contarán como colisión si están a una distancia máxima de 6 del centro.

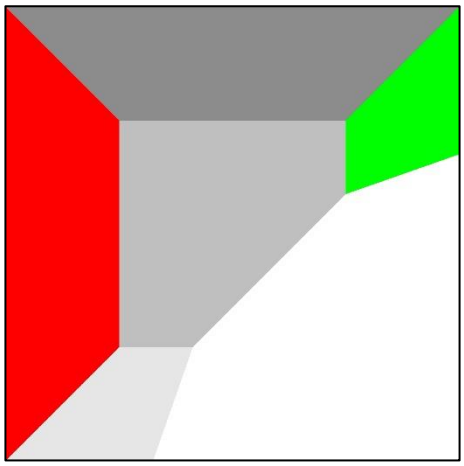


Figura 4. Plano con Ray Tracing.

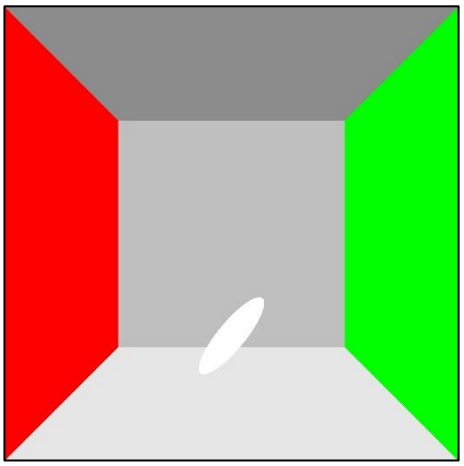


Figura 5. Disco con Ray Tracing.

- **Cilindro infinito y cilindro**: Para la creación del cilindro ha sido necesario implementar primero el concepto de cilindro infinito. La ecuación del cilindro infinito teniendo un punto cualquiera (p), un punto de origen del cilindro (o), la dirección del cilindro (d) y el radio (r) es la siguiente:

$$\left(\boldsymbol{p}-\boldsymbol{o}-\left(\boldsymbol{d}\times(\boldsymbol{p}-\boldsymbol{o})\right)^{2}\right)-\boldsymbol{r}^{2}=\mathbf{0}$$

Ahora, como para el resto de las geometrías, se sustituye esta ecuación el punto cualquiera (p) por la ecuación del rayo y se obtiene como para la esfera, una ecuación de segundo grado donde la variable que se despeja (si hablamos de la fórmula de la esfera es r) representa la distancia del origen del rayo al cilindro. Ocurre lo mismo que para la esfera: cuando no se obtiene ninguna solución, el rayo no corta al cilindro, cuando se obtiene una única solución es que el rayo es tangente al cilindro y cuando se obtienen dos, son los dos puntos por los que atraviesa al cilindro infinito obteniendo así el resultado que se muestra en la figura 6.

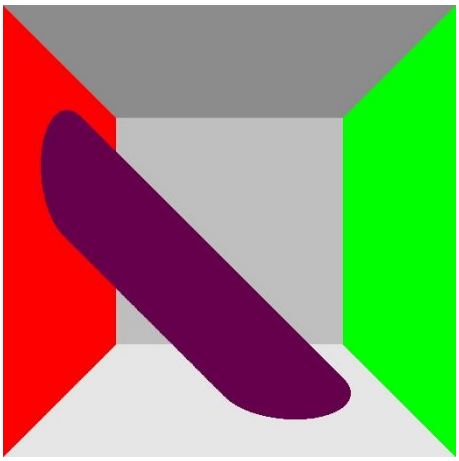


Figura 6. Cilindro infinito con Ray Tracing.

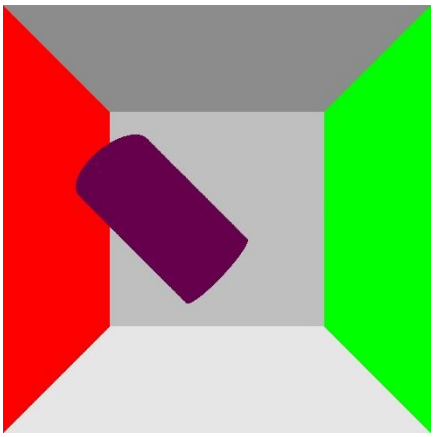


Figura 6. Cilindro con Ray Tracing.

Ahora, teniendo el cilindro infinito, se crea el cilindro de la misma manera que con un cilindro infinito, pero añadiendo dos discos que son los que se encargan de delimitar la altura de éste. De esta forma, se consigue el resultado de la figura 7.

Al no tener aún ninguna iluminación, la base que es un disco se ve de forma curvada que ahora mismo no queda muy estética, pero, una vez aparecen las sombras, se distingue el contorno.

- **Triángulo**: El caso del triángulo se compone de la misma manera que el disco. Es decir, es un plano al cual se le añaden unas condiciones de contorno las cuales se encargan de delimitar la superficie.

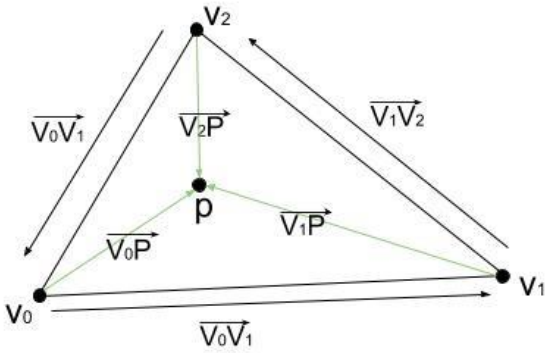


Figura 7. Colisión con un triángulo.

Sin embargo, para este caso, no sirve con calcular una distancia ya que un triángulo se forma por tres puntos por lo que, para comprobar si el rayo colisiona con el triángulo, se crean tres vectores, uno por cada arista del triángulo. A partir de estos vectores, se calcula mediante el producto escalar si el punto queda a un lado u otro de cada uno de los vectores. Para que el punto esté dentro del triángulo tiene que darse que esté a la izquierda de los tres vectores tal y como se ejemplifica en la figura 7.

Como ejemplo usando la figura 7, se utiliza el producto escalar para hacer:

$$\left(\boldsymbol{n} \times (\overrightarrow{\boldsymbol{V_0 V_1}} \times \overrightarrow{\boldsymbol{V_0 P}}\right)$$

Donde n es la normal del punto y si este resultado da un valor mayor que 0, significa que está a la izquierda. Si esto se hace para los tres vectores correspondientes a las aristas del triángulo y se obtiene un valor mayor que 0 para los tres, el punto está dentro del triángulo.

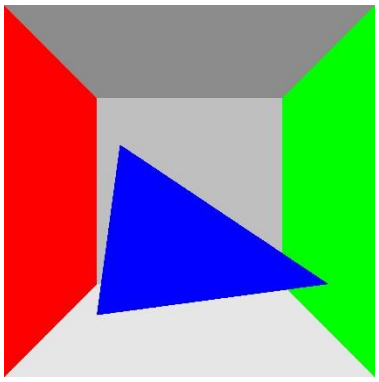


Figura 8. Triángulo en Ray Tracing.

Con esto se obtienen resultados como el de la figura 8.

- **Paralelepípedo**: Para el caso del paralelepípedo se ha decidido trabajar con una estructura de doce triángulos. Cada par de triángulos forma un lado del paralelepípedo. Para construir el paralelepípedo, se le pasan dos triángulos que tienen que estar en el mismo plano y forman lo que se considera la base y su altura. A partir de aquí, se calculan los vectores de cada lado de la base y se hace el producto escalar de la normal por la altura obteniendo así el vector que se consideraría como la altura.

  Ahora, teniendo la altura y los cuatro puntos de la base del paralelepípedo, se le suma la altura a cada uno los puntos iniciales obteniendo así lo que correspondería con los ocho vértices del paralelepípedo. Teniendo los ocho vértices, se crean los triángulos correspondientes para formar el paralelepípedo.

  Para trabajar con la colisión del paralelepípedo se recorren los doce triángulos que lo conforman y se calcula la colisión para cada uno de ellos quedándose con los que tienen una menor distancia a la cámara (ya que en un paralelepípedo siempre habrá una pared que esté detrás de la que se en la cámara). Esto hace que se obtenga un resultado como el de la figura 9.

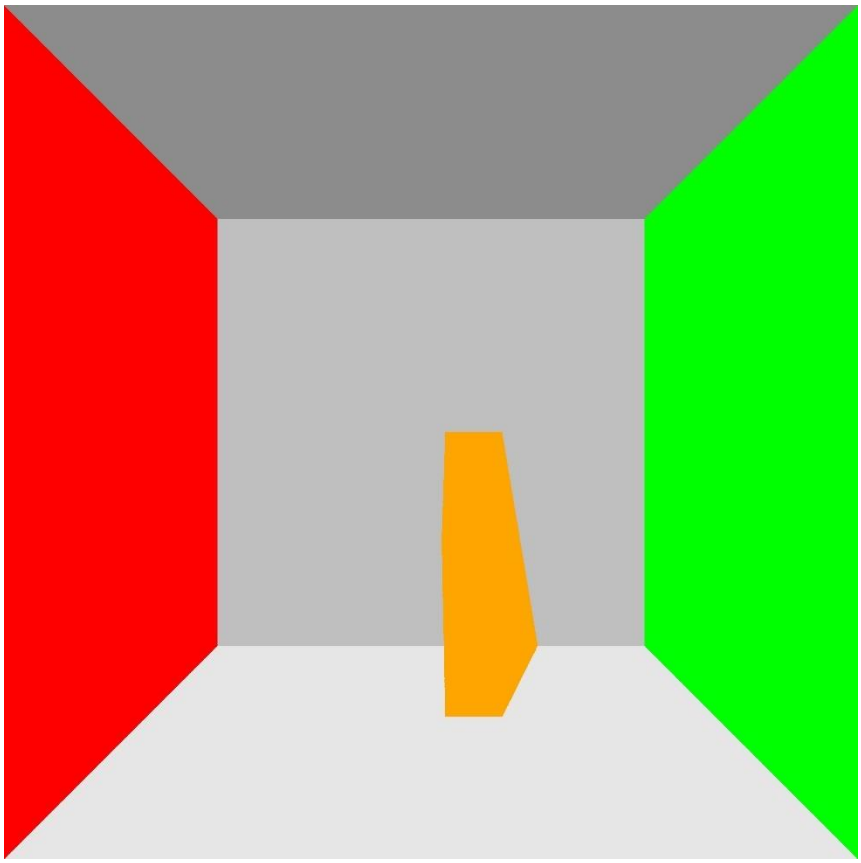


Figura 9. Paralelepípedo en Ray Tracing.

Path Tracing

Una vez que se ha creado la cámara y las geometrías que se han considerado convenientes para esta práctica en Ray Tracing, se ha pasado a implementar Path Tracing. Path Tracing es un algoritmo para renderizar imágenes que trabaja con el método de Monte Carlo.

Path Tracing destaca frente a Ray Tracing ya que se añade la iluminación y BRDF (entre otras cosas) lo que permite obtener sombras entre otros. Otra gran diferencia frente a Ray Tracing es que, en este caso, no se lanza un solo rayo al centro del píxel como se hacía antes si no que se lanzan múltiples rayos por píxel.

Iluminación y BRDF

Tal y como se ha explicado en Ray Tracing, al no tener iluminación, se obtienen escenas planas lo que hace que, por ejemplo, para la figura 9 no se aprecian las esquinas y cada lado del paralelepípedo. Por eso, se ha introducido la iluminación y los materiales. Esto se debe gracias a la ecuación de render:

$$\boldsymbol{L_0(x, \omega_0) = L_e(x, \omega_0) + \int_{\Omega} L_i(x, \omega_i) f_r(x, \omega_i, \omega_0) |n \cdot \omega_i| d\omega_i}$$

Entonces ahora, para cada colisión que ocurre cuando se lanza el rayo, se estima la ecuación del render de forma que como computacionalmente no es viable calcular esta integral, se traduce a un sumatorio de múltiples rayos por píxel y cada rayo que se lanza entre todos los que sea lanzan por píxel genere un único camino de forma que al alcanzar una superficie no rebote ya que eso crece exponencialmente y no sería viable.

De esta forma, se lanzan n rayos por píxel (en nuestro programa se pasa como primer parámetro al ejecutarlo) y el color final del píxel será le media de color de todos los rayos que se han lanzado para ese píxel. Para decidir dentro de un píxel hacia dónde se lanzan los rayos, se hace de forma aleatoria por el área que ocupa ese píxel. Esto implica, como es obvio, que cuanto mayor sea el número de rayos que se lanzan, mayor será el número de caminos que se generarán para ese píxel por lo que cuando se hayan obtenido el valor del color para cada uno de los rayos lanzados, al hacer la media de todos los colores obtenidos, la imagen final será mucho más precisa.

Hay un problema a la hora de lanzar todos estos rayos y es que los rayos lanzados pierden energía en cada rebote que absorbe el material, pero esa energía que se pierde está relacionada con la distancia que recorre, pero nunca llega a ser cero. Por eso mismo, es necesario “matar” el rayo en algún momento ya que, si no, el algoritmo nunca finalizaría.

Para ello, se ha implementado la ruleta rusa de forma que cuando se obtiene el color de una BRDF, se genera un número aleatorio (llamado rnd) comprendido entre 0 y 1. Esto se complementa con el método de Monte Carlo de forma que este número aleatorio puede dar tres opciones diferentes. La primera de ellas es cuando se compara con el valor de "kd". El valor de kd representa el coeficiente de difusión que tiene el material. Si el número aleatorio obtenido es menor que dicho coeficiente, entonces se trabaja con ese rayo y material como si fuera difuso. Si el número aleatorio está comprendido entre el coeficiente kd y la suma de kd y ks, implica que el rayo con el que se va a trabajar es especular. Sin embargo, si el número aleatorio obtenido está por encima de la suma de kd y ks, se devuelve el color negro, es decir, se "mata" el rayo para que así acaben los rebotes. Gracias a esto de aquí, se está implementando la ruleta rusa de forma que la probabilidad de matar el rayo es de 1 – (kd +ks). La ruleta rusa implica que el algoritmo de Monte Carlo converja ya que existe la probabilidad de que, en algún rebote, el rayo se acabe. Además, para que el algoritmo acabe, existe una variable que limita el número de rebotes siendo ésta diez.

Respecto a la pregunta relacionada con la convergencia de path tracing según el número de samples por píxel que tenga, tarda más en converger si el número de samples es menor como se ilustra en la figura 10 con 2 samples que ha tardado 7,53 segundos que en la figura 11 con 1000 samples que ha tardado 3446,73 segundos.

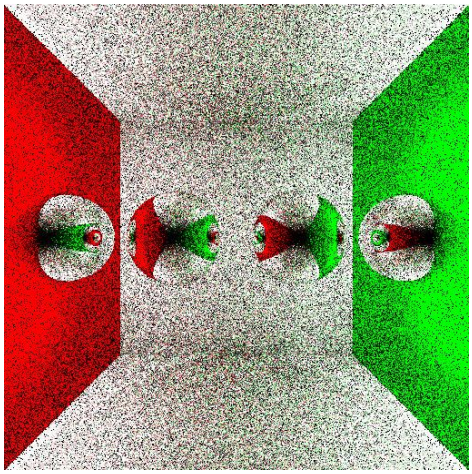


Figura 10. Imagen con 2 samples.

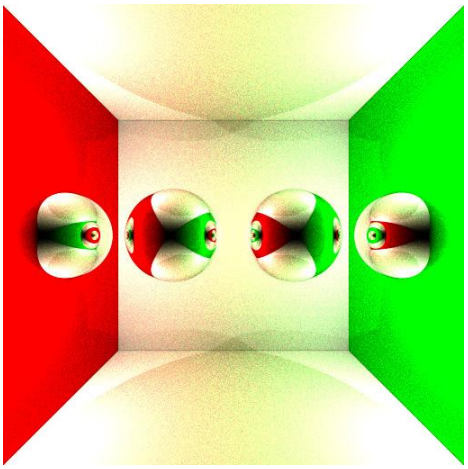


Figura 11. Imagen con 1000 samples.

Respecto a la segunda pregunta relacionada con la convergencia, se ha observado que para el caso de geometrías con volumen (esferas, paralelepípedos, etc.) el material que más suele tardar es un material con refracción perfecta ya que al rebotar el rayo tiene que lanzar un rayo que choca con la geometría (tiene que calcularse la ley de Snell) y luego hay que lanzar otro hasta que se choca con el propio material otra vez y se vuelve a refractar ese rayo.

Contestando a la tercera pregunta, teóricamente converge más despacio con las luces de área. Esto ocurre porque es necesario muestrear las luces de área calculando la probabilidad con la que el rayo puede chocar con la ella. Sin embargo, para luces puntuales se busca directamente el punto de luz haciendo que el coste sea mucho menor.

Otro de los materiales creado ha sido el material refractivo. Para este caso, se trabaja con la ley de Snell donde la nueva dirección después del rebote depende de dos factores. El ángulo de incidencia y el índice de refracción del medio con el que rebota. Por ejemplo, el índice de refracción del cristal es más o menos de 1,5 y el del agua es de 1,3.

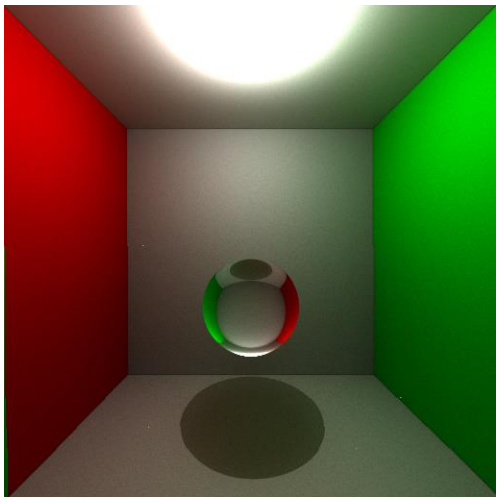


Figura 12. Refracción en path tracer.

Para implementar este material se ha tenido en cuenta una variable que determina si el material tiene volumen o no. Dependiendo de esto, si el material tiene volumen, es necesario calcular el rayo refractado que volverá a chocar consigo mismo teniendo que calcular otra vez la refracción del rayo. Si no tiene volumen, solamente refracta una vez. Este resultado se puede observar en la figura 12.

Otro material implementado ha sido el material reflexivo. Para este material ocurre que si la superficie es reflexiva como puede ser un espejo, hay un rayo que sale de la superficie con una dirección simétrica al rayo incidente, es decir, el ángulo con el que el rayo entra es el mismo con el que sale como se puede observar en la figura 13. El resultado para nuestro path tracer se puede ver en la figura 14.

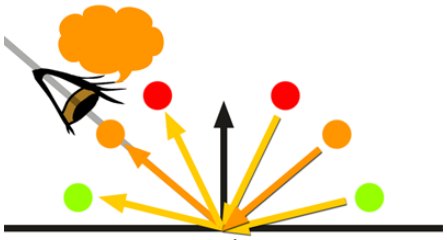


Figura 13. Ángulo de reflexión.

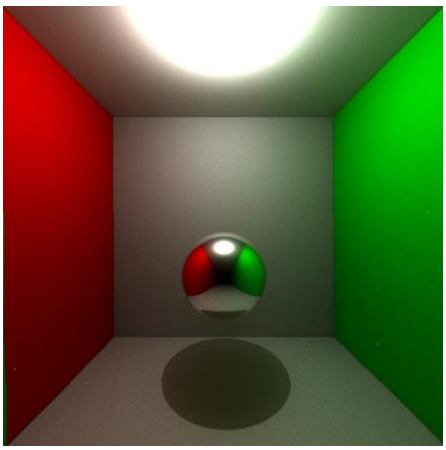


Figura 14. Reflexión en path tracer.

Para el caso de **luces puntuales**, como no se puede aproximar mediante la colisión de un rayo con un objeto, ya que un rayo no tiene probabilidad de pasar por la posición de una fuente de luz, es necesario calcular el aporte de este tipo de fuentes. Por eso, cuando se encuentra un objeto en la escena, se va a buscar las luces puntuales para ver qué aporte tienen en ese punto de la escena.

Para calcular el aporte de las luces directas sobre una BRDF de Phong, es necesario calcular el sumatorio ya que, como se ha explicado antes, no es viable calcular la integral que obtiene toda esta luz. Como la luz tiene la propiedad de que es aditiva, a partir de la ecuación de render explicada previamente, se calcula la potencia de la luz dividiendo por la distancia al cuadrado por el término del coseno de la ecuación de render.

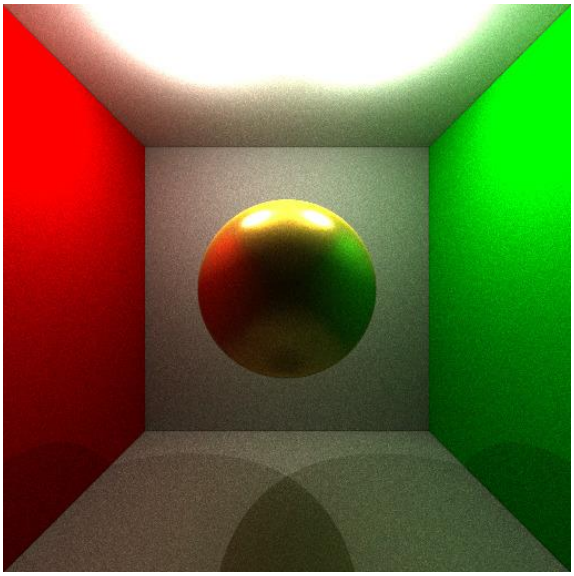


Figura 15. Dos luces puntuales.

Para el caso de luces puntuales, cuando hay dos luces puntuales, se pueden apreciar las sombras duras que genera cada una de ellas en la figura 15. Sin embargo, como son dos luces y la parte más oscura de la sombra es la parte a la que no llega la luz puntual directamente pero el resto de sombra no es tan fuerte ya que solo es iluminada por una de las dos.

Para **luces de área** se ha creado un tipo de material llamado emisivo que se trata de la siguiente manera. Cuando un rayo llega a este material, se obtiene el color que emite este material con una función que tiene en cuenta la atenuación de la luz según la distancia a la que está del objeto y se devuelve este valor multiplicado por el color que emite el objeto. Estas luces de área se pueden ver muy bien en la figura 16.

Figura 16. Cuatro luces de área.

Contestando a la primera pregunta sobre **iluminación global**, el efecto que más fácil se obtiene es el de “hard shadows” porque se calcula la luz buscando fuentes puntuales por lo que un punto está o no está iluminado por esa luz. Sin embargo, para luces de área, un objeto puede estar iluminado por una parte de la luz y por otra no.

Para el caso de la segunda pregunta dos, para obtener “hard shadows” es necesario tener una luz puntual, para “soft shadows” es necesario tener luces de área ya que se atenúa la sombra de los objetos, para “color bleeding” es suficiente con tener dos paredes que tengan colores diferentes y para el caso de causticas es necesario tener cualquier objeto refractivo como, por ejemplo, una esfera de cristal.

Finalmente, para la última pregunta el efecto que no se ha podido conseguir ha sido causticas ya que se tendría que haber implementado path tracing bidireccional.

Medios participativos

A la hora de trazar un rayo se calcula la probabilidad de que ocurra un evento de interacción con el medio y la distancia a la que ocurre con la siguiente fórmula (ξ representa un número aleatorio entre 0 y 1):

$$distancia = \frac{-log(1-\xi)}{\sigma_t}$$

$$L_i(x,\varpi) \approx \sum_{t=0}^{S-1} T_r(x \leftrightarrow x_t)\sigma_s(x_t)L_i(x,\varpi)\triangle_t + T_r(x \leftrightarrow x_t)L_i(x_s,\varpi)$$

$$\xi < \frac{\sigma_s}{\sigma_t}$$

$$tr = e^{||x-x_t||(-\sigma_t)}$$

Siendo $||x - x_t||$ la distancia entre el rayo y la ocurrencia del evento de interacción con el medio.

Se comprueba que la distancia es menor que la distancia del rayo al objeto con el cual se produce la primera colisión, si se cumple esta condición pueden suceder dos cosas que ocurra un evento de scatter o de absorción. Esta probabilidad de que ocurra un evento u otro se calcula generando un número aleatorio entre 0 y 1 si este es mayor que el coeficiente de scattering entre el coeficiente de extinción ocurre un evento de absorción en caso contrario ocurre un evento de scatter.

Cuando ocurre un evento de scatter se busca la aportación de todas las fuentes de luz en ese punto y se suman, multiplicándolas por la transmitancia, el albedo del medio y el coeficiente de scatter.

Como el medio es homogéneo implica que la función de fase es isótropa, por lo que para calcular la nueva dirección del rayo se muestrea la esfera uniformemente para obtener la dirección del rayo refractado, trazándose con dicha dirección.

Si el evento que ocurre es de absorción se vuelve buscar la aportación de todas las fuentes de luz a ese punto y se suman, multiplicándolas por la transmitancia, el albedo del medio y el coeficiente de absorción. Pero no se vuelve a trazar un nuevo rayo ya que cuando hay absorción el rayo continúa con la misma dirección.

Para el caso de que no ocurra ningún evento de scattering el color obtenido se sigue multiplicando por el albedo del medio y la transmitancia.

Se ve el efecto conseguido con medios participativos para el caso de la niebla en la figura 17 (como si fuera Londres). También se puede conseguir niebla de colores como en la figura 18 (tormenta en el Sahara).

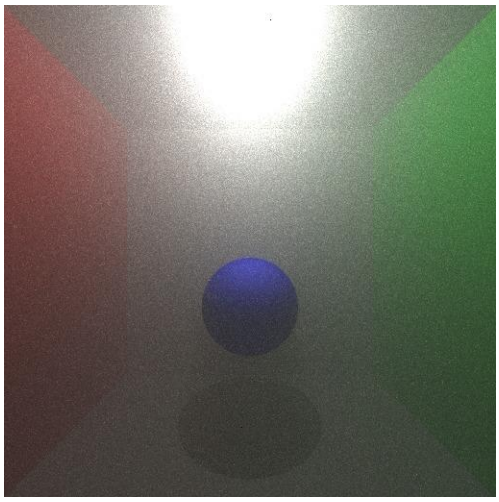


Figura 17. Niebla.

Figura 18. Niebla amarilla.

Bibliografía

https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-shading/reflection-refraction-fresnel

https://graphics.pixar.com/library/ProductionVolumeRendering/paper.pdf

http://www.diva-portal.org/smash/get/diva2:796154/FULLTEXT01.pdf

http://corysimon.github.io/articles/uniformdistn-on-sphere/

http://www.cs.cornell.edu/courses/cs6630/2012sp/notes/09volpath.pdf

http://www.cs.cornell.edu/courses/cs6630/2012sp/notes/08radiative-transfer.pdf

https://ccrma.stanford.edu/~jos/pasp/Mean_Free_Path.html

https://computergraphics.stackexchange.com/questions/4412/volume-rendering-path-tracing-real-time