PHOTON MAPPING
INFORMÁTICA GRÁFICA

David Solanas (738630)

Introducción

En este trabajo se pide realizar la implementación del algoritmo de Photon Mapping estudiado en la asignatura, para ello se ha partido del código proporcionado por los profesores como base, teniendo que completar varias funciones además de añadir las funcionalidades que se consideren oportunas.

Implementación

El algoritmo clásico de Photon Mapping se divide en 2 fases, la primera la generación del mapa de fotones; la segunda, la estimación de la radiancia. Para ambas fases, el código proporcionado ya tenía las bases implementadas y simplemente había que completar las funciones necesarias para obtener así el algoritmo completo.

Generación del mapa de fotones

Esta fase consiste en generar los fotones desde las distintas fuentes de luz de la escena y guardar las interacciones de los mismos con los objetos de la escena [5].

A la hora de generar fotones, se ha tenido en cuenta si la fuente de luz era una fuente de luz puntual o de área (un plano) ya que la normalización de los fotones varía dependiendo del tipo de luz. La normalización de los fotones se ha realizado de la siguiente forma:

- Fuentes de luz puntuales: El punto de origen del fotón siempre es el origen de la fuente de luz, y la dirección de dicho fotón se calcula de forma aleatoria en todo el espacio de una esfera en 3 dimensiones (<x, y, z> vector unitario con x, y, z ∈ [-1.0 , 1.0]). Una vez calculado el origen y la dirección del fotón hay que normalizarlo, para ello hay que tener en cuenta la probabilidad de elegir esa fuente de luz *p(Li)* (calculado directamente dividiendo el número de fotones a muestrear entre el número de fuentes de luz ya que se hace una división de fotones equitativa y determinista) y la probabilidad de elegir una dirección de salida del fotón *p(w)* ($\frac{1}{4}$ ya que el ángulo sólido de una esfera de radio 1 medido desde cualquier punto de su interior es 4π [1]) de modo que la intensidad de un fotón viene dada por la fórmula $\frac{intensidad\ luz}{p(Li) * p(w)}$.
- Fuentes de luz de área (planos): La generación de los fotones es muy parecida a la de las luces puntuales pero con ciertos matices. El punto

de origen del fotón ya no es siempre el mismo sino que se elige aleatoriamente un punto de la superficie del plano (se ha implementado una función para ello en la clase *BoundedPlane*) de modo que ahora para la normalización de los fotones habrá que tener en cuenta además la probabilidad de elegir un punto en la superficie de la luz como origen del fotón *p(S)* (no se tenía en cuenta en las puntuales ya que al ser siempre el mismo origen la probabilidad es 1; esta probabilidad se calcula dividiendo 1 entre el área de la fuente de luz). La dirección del fotón se calcula igual sin embargo ahora hay que tener en cuenta que el producto escalar entre la dirección escogida y la normal de la fuente de luz sea mayor que 0, por lo que la *p(w)* pasa a ser $\frac{1}{2}$ ya que ahora se muestrea en la hemiesfera de radio 1. La probabilidad de elegir esa fuente de luz *p(Li)* se calcula igual que con las fuentes de luz puntuales. Con todo esto, la intensidad de un fotón se calcula con la fórmula $\frac{intensidad\ luz}{p(Li) * p(w) * p(S)}$.

Una vez generados los fotones, hay que ver cómo interactúan con la escena y guardar dichas interacciones. Para ello se hace uso de la función *trace_ray* proporcionada con el código.

Esta función recibe como entrada 6 parámetros, el primero el fotón (origen + dirección), el segundo su intensidad, el tercero y cuarto son dos listas para almacenar las interacciones de los fotones (los fotones globales y aquellos destinados al mapa de cáusticas) , por último los parámetros *direct* y *direct_only* que determinan si guardar también la luz directa (primer rebote del fotón) o si únicamente guardar la luz directa. *Trace_ray* devuelve cierto si quedan fotones por lanzar y false cuando ya se hayan registrado todas las interacciones de todos los fotones, realiza lo siguiente:

1. Comprueba si se han lanzado todos los fotones, si es así termina la ejecución y devuelve *false*.
2. Comienza a procesar la interacción del fotón por la escena y guardar los resultados en los respectivos mapas.
    a. Calcula la primera intersección del fotón con los objetos de la escena.
    b. Comprueba si el material del objeto es delta o no. Si es delta no se guarda ningún fotón en esa interacción, se deja marcado que es una cáustica para la siguiente iteración y guardarlo en su respectivo mapa de fotones (cáusticas).
    c. Si el material del objeto no es delta, se comprueba si la interacción que está procesando es la primera o no (luz directa o no). Si es el primer rebote (nivel del fotón igual a 0) y los

parámetros *direct* y *direct_only* son false no se guarda esa interacción en el mapa de fotones. En caso contrario se guarda ese rebote del fotón en su respectivo mapa de fotones (dependerá del parámetro que indica si es una cáustica o no).

d. Una vez guardado (o no) el rebote en el mapa de fotones, se prepara la siguiente iteración (el siguiente rebote de ese fotón). Se hace ruleta rusa para determinar si "matar" al fotón o no (el fotón se "mata" si supera un máximo de interacciones, 50. También se mata si el parámetro *direct_only* es *true* y el material del objeto interceptado no es delta y además el nivel del fotón es mayor que uno, es decir ya se había guardado la interacción de la luz directa). Si el fotón no ha sido "matado" se sigue con su interacción, para ello se calcula la nueva dirección del fotón, y se reajusta su energía dependiendo del material intersectado.
e. Finalmente, una vez calculadas todas las interacciones del fotón, se comprueba nuevamente si quedan fotones por lanzar o ya se han calculado todos. Dependiendo de si quedan por procesar o no devolverá *true* o *false*.

Finalmente, falta almacenar todas las interacciones de los fotones globales y cáusticas en sus respectivos KD-Trees para permitir la búsqueda de los mismos y sus vecinos de forma eficiente, para ello se almacenan en ambos árboles y se balancean.

Estimación de la radiancia

Una vez generado los mapas de fotones, falta realizar la estimación de la radiancia en la escena para generar finalmente la imagen, esta parte se divide en 2 pasos principalmente y el resultado final será la suma de los dos resultados obtenidos debido a que la luz es aditiva:

1. <u>Cálculo de la luz directa en un punto</u>

   Se presentan dos casos en este paso, si el material del punto intersectado es delta o no. Si no es delta se calcula la contribución de todas las fuentes de luz de la escena en ese punto como sigue:

   - Para cada luz de la escena:
     - Se comprueba si es visible desde ese punto, si no es visible se considera que no llega luz y el resultado será 0.
     - Si la luz es visible, se calcula la luz que llega a ese punto con la fórmula *Li(x,wi)* = $\frac{p}{|c-x|^2}$ [4] donde p es la intensidad de la luz, c es el punto de la fuente de luz y x es el punto de intersección en la escena. Esa *Li*

calculada se multiplica por la BRDF del material, como el material del objeto sólo puede ser Lambertiano o Phong (se realiza una comprobación para descartar los deltas), se comprueba el parámetro de *shininess* del material, si es 0 ó ∞ es de tipo Lambertiano por lo que la brdf será $\frac{albedo}{kd}$, si es Phong se aplica la ecuación de la parte especular de la BRDF Phong. Por último, se multiplica por el término del coseno que será el producto escalar entre la normal en la intersección de la escena y la dirección desde el punto de la fuente de luz y la intersección.

- ➢ El resultado obtenido se va acumulando para cada fuente de luz.

- ❖ En este cálculo hay que tener en cuenta que las luces pueden ser puntuales o de área, en el caso de las puntuales el punto de la luz (*c*) será el punto especificado por el usuario. Sin embargo para las luces de área el cálculo cambia, ya que lo ideal es muestrear múltiples rayos de sombra desde la intersección en busca de la fuente de luz [3]. Esto se consigue muestreando un punto aleatorio en la superficie de la luz y tratarlo como una fuente de luz puntual, este proceso se repite *n* veces (especificado por el usuario, cuanto mayor *n* el resultado se aproxima más a la realidad) acumulando la luz obtenida y finalmente dicha luz se divide entre *n*, esto se conoce como una técnica de aproximación probabilista de Monte Carlo. Este método permite generar lo que se conoce como soft-shadows generando escenas más realistas a costa de un mayor tiempo de cálculo (depende de cuántas muestras se generen en la fuente de luz), también genera ruido debido a la aleatoriedad de las muestras. Con las luces puntuales es imposible conseguir este efecto (Figura 1 y 2).

Si el material del objeto intersectado es delta lo que se hace es propagar el rayo por la escena (cuando intersectan con materiales delta) calculando el rayo reflejado (en el caso de espejos) o refractado (en el caso de bolas de cristal por ejemplo) y acumular sus BRDF (multiplicándolas, la BRDF será directamente el albedo del objeto) hasta encontrar un material difuso, en ese momento se calcula la luz directa en ese punto como se ha explicado anteriormente y se estima la radiancia con los mapas de fotones (explicado abajo en el punto 2). De forma que la luz en la primera intersección de la escena (en el

espejo, bola de crista, etc.) será la luz que recibe tras haber rebotado por los materiales deltas desde un punto difuso en la escena.

2. Estimación de la radiancia en un punto con los mapas de fotones

   En este paso se calcula si el material del objeto intersectado no es delta ya que no se guarda ninguna información de los materiales delta en los mapas de fotones, para ello se han implementado dos funciones *global_photons_radiance* y *caustic_photons_radiance* que calculan la radiancia en ese punto con el mapa de fotones globales y de cáusticas respectivamente. El comportamiento de las funciones es idéntico (el único cambio es el mapa de fotones a utilizar) y es el siguiente:

   a. Encontrar los *n* vecinos (parámetro introducido por el usuario) más cercanos al punto de intersección con la escena. Para ello se hace uso de la función *find* del KD-tree correspondiente.
   b. Aplicar esta fórmula [2]:

$$L_r(\mathbf{x},\omega_r) \approx \frac{1}{\Delta A}\sum_{p=1}^{n} f_r(\omega_p,\omega_r)\Delta\Phi_p(\mathbf{x},\omega_p)$$

   Donde *fr* es la BRDF del material del objeto intersectado, $\Delta\boldsymbol{\Phi}_p$ es el flujo del fotón y ΔA es el área del kernel (círculo de radio variable, el radio del kernel es devuelto en la función *find*).

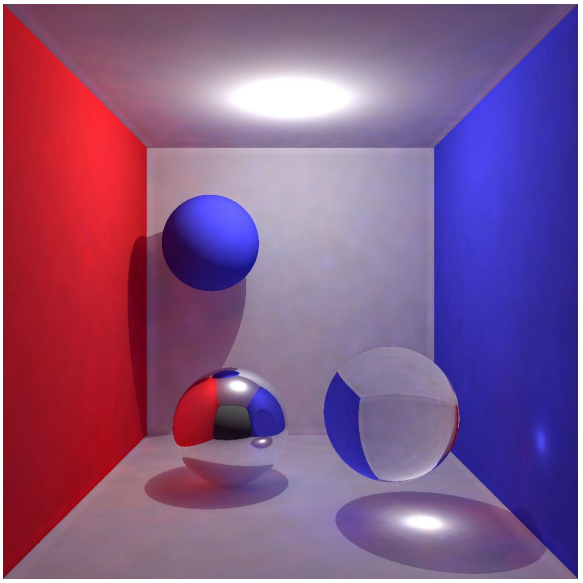

Figura 1: *Fuente de luz puntual*

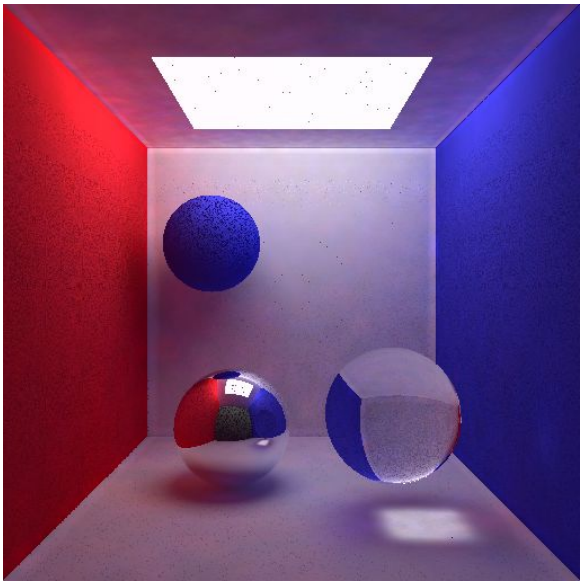

Figura 2: *Soft-shadows con fuente de luz de área (2500 rayos de sombra)*

Cuestiones adicionales

Cuestión 2.1:

Las imágenes renderizadas corresponden a las figuras 3 y 4. La principal diferencia entre *Ray tracing* y *Photon Mapping* para el cálculo de la luz directa es que con *Ray tracing* se calcula la contribución de las fuentes de luz en ese punto sin que puntos cercanos al mismo ejerzan ninguna influencia, esto genera escenas con mucho contraste en las sombras ya que si la luz no es visible desde ese punto se considera que la contribución de la misma es 0. Sin embargo, con *Photon Mapping* este efecto de las sombras no es tan brusco (generando escenas más acordes a la realidad) debido a la fase de estimación de la radiancia, ya que al estimar la radiancia en un punto se buscan los k fotones más cercanos a ese punto y se aproxima la contribución de la luz. La solución con *Photon Mapping* produce resultados más realistas a cambio de introducir un cierto sesgo.

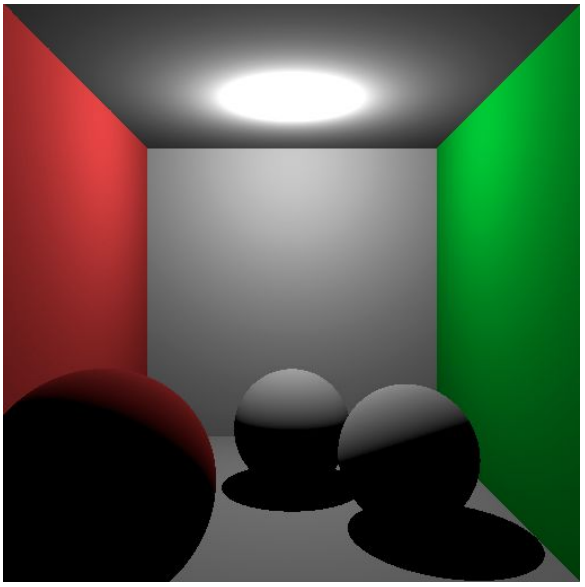

Figura 3: *Luz directa Ray tracing*

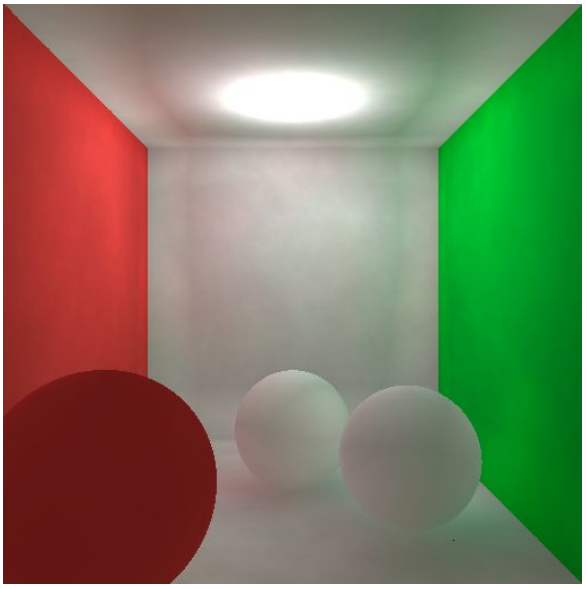

Figura 4: *Luz directa Photon Mapping (100K fotones y 500 vecinos)*

Cuestión 2.2:

Los materiales de las dos esferas son delta, por lo que a la hora de obtener la luz que llega a las mismas se realiza lo siguiente. En primer lugar no se calcula la luz directa ni tampoco se estima la radiancia con los mapas de fotones en ese punto. Cuando intersecta con un material delta lo que se hace es calcular el albedo del mismo (para ir acumulándolo para el cálculo de las BRDF de los rebotes delta) y obtener el rayo de salida (en el caso del espejo con la ley de reflexión y en el caso del cristal con la ley de Snell [6]), con el rayo de salida obtenido se calcula el nuevo punto de intersección más cercano con la escena y se comprueba que tipo de material es. Si el material es delta se repite el mismo proceso hasta encontrar un

material difuso. Si el material de la nueva intersección es difuso se para el proceso de rebotes y se calcula la contribución de la luz directa en ese punto más la radiancia estimada con los mapas de fotones de cáusticas y de fotones globales, de modo que al final se tiene que la luz en un objeto con material delta será ($L_{directa\_difuso}$ + $L_{fotones\_globales\_difuso}$ + $L_{causticas\_difuso}$) * $BRDF_{rebotes\_delta}$. El resultado final puede verse en la figura 5

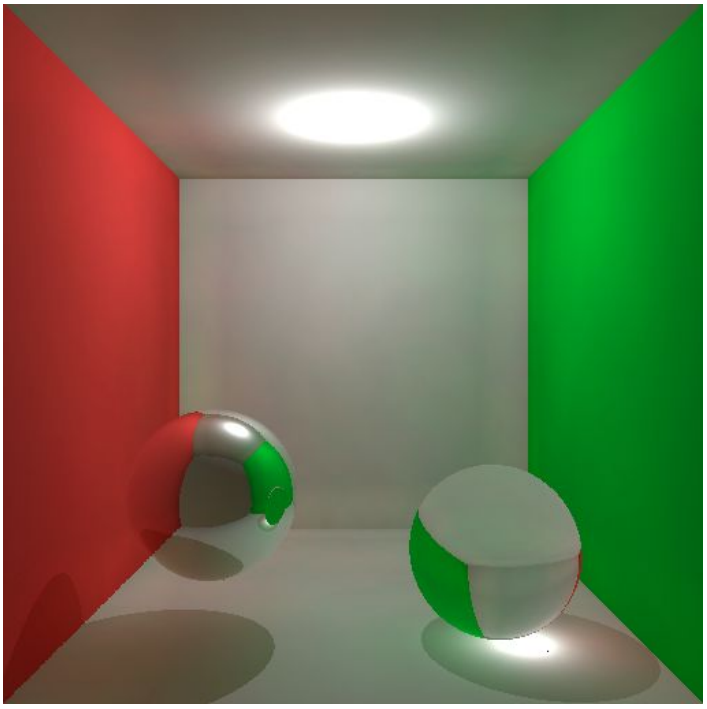

Figura 5: *Escena 1 con 100K fotones y 500 vecinos para la estimación de la radiancia.*

Cuestión 2.3:

Con *Photon Mapping* el número de fotones que se especifican para ser lanzados y el número de vecinos a encontrar para estimar la radiancia en un punto de la escena repercute directamente a la calidad de las imágenes obtenidas pero también al tiempo de cálculo. Conforme mayor es el número de fotones lanzados, más se aproxima la solución a la realidad y habrá una mejor iluminación en la escena, pero aumenta el tiempo de ejecución de la generación del mapa de fotones, ya que hay más fotones que guardar. Por otro lado a la hora de estimar la radiancia, el parámetro del número de vecinos a encontrar influye de la misma forma que el número de fotones al hacer el mapa de fotones, cuanto mayor es el parámetro (usualmente se eligen 500 fotones) más realismo se consigue aunque añadiendo cierto sesgo, sin embargo mayor tiempo de cálculo a la hora de la estimación de la radiancia. A continuación, en las figuras 6, 7, 8, 9, 10, 11, se muestra la escena 1 renderizada con distintas cantidades de fotones lanzados (*n*) y distintos números de vecinos (*k*).

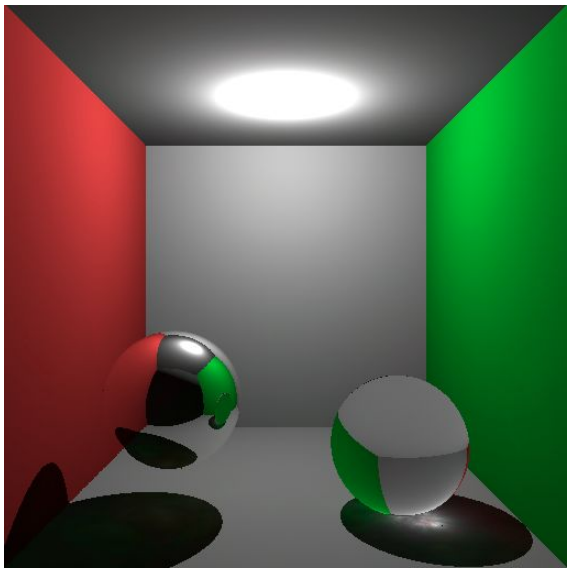

Figura 6: *Escena 1. n=1K, k=10*

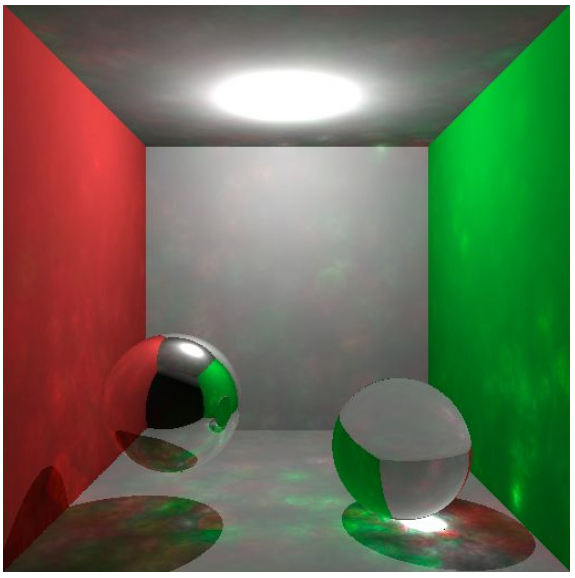

Figura 7: *Escena 1. n=10K, k=10*

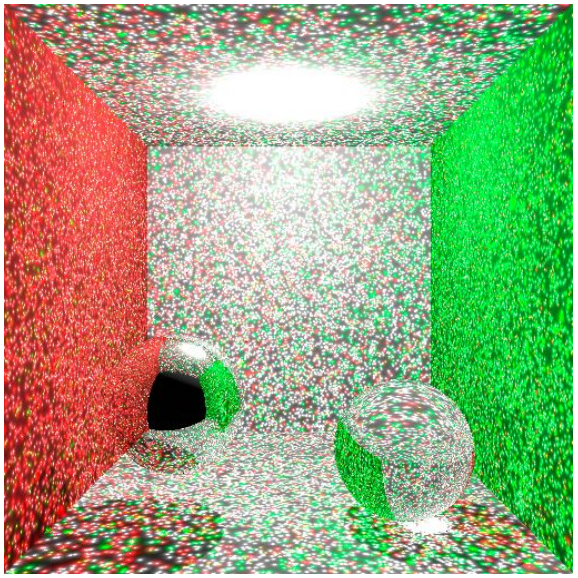

Figura 8: *Escena 1. n=100K, k=1*

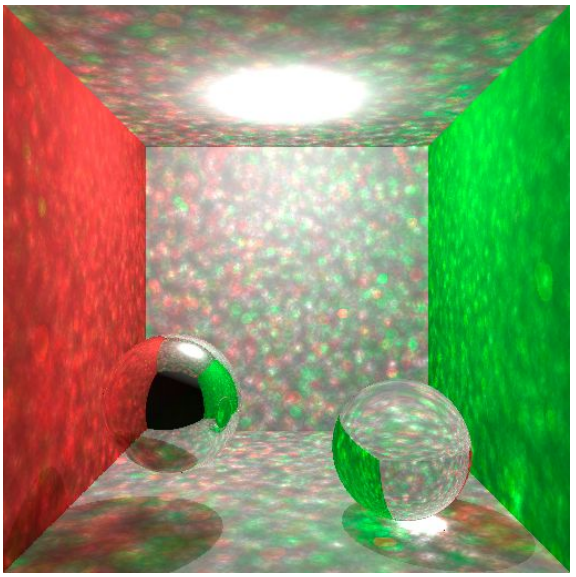

Figura 9: *Escena 1. n=100K, k=10*

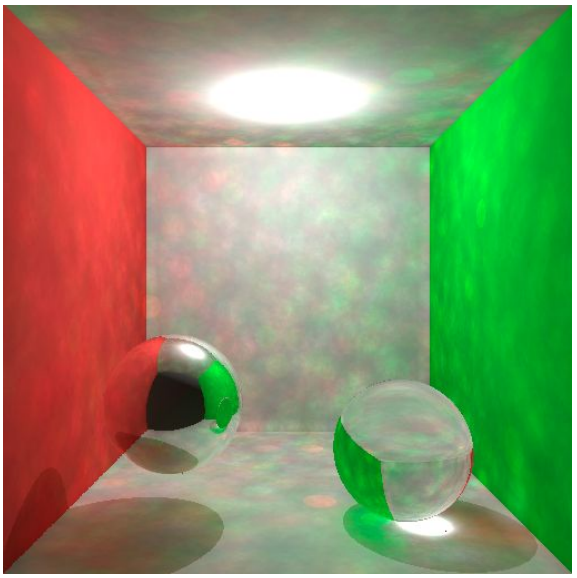

Figura 10: *Escena 1. n=100K, k=50*

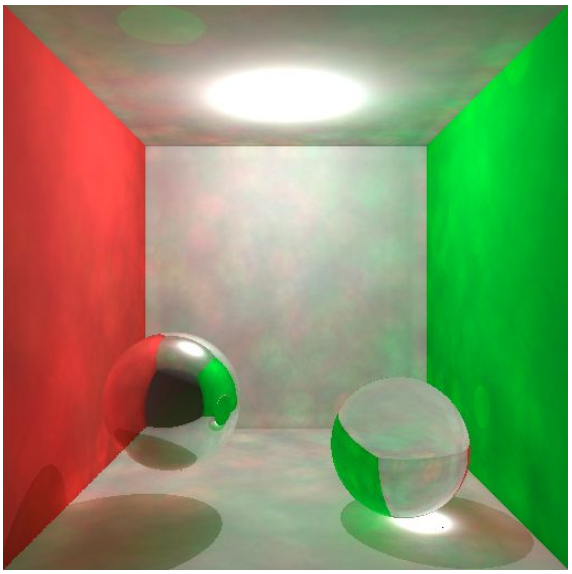

Figura 11: *Escena 1. n=100K, k=100*

En la figura 12 se puede ver el incremento del coste temporal conforme se aumenta *n* y *k.*

```
david@pc-ubuntu:~/Documentos/informatica_grafica-master/SmallPM_v1/SmallPM-linux/Ux$ ./smallpm -scene 1 -film-name q2_3_1k_10 -pm-total-photons 500
Prepared scene to render: ered..[0:0:2.9e-05]
Photons Shot: ns[0:0:1.0842]
Rendering ...[DONE]:    [0:0:4.2307]
david@pc-ubuntu:~/Documentos/informatica_grafica-master/SmallPM_v1/SmallPM-linux/Ux$ ./smallpm -scene 1 -film-name q2_3_10k_10 -pm-total-photons 5000
Prepared scene to render: ered..[0:0:3.4e-05]
Photons Shot: ns[0:0:10.7247]
Rendering ...[DONE]:    [0:0:4.8136]
david@pc-ubuntu:~/Documentos/informatica_grafica-master/SmallPM_v1/SmallPM-linux/Ux$ ./smallpm -scene 1 -film-name q2_3_100k_10 -pm-total-photons 50000
Prepared scene to render: ered..[0:0:2.9e-05]
Photons Shot: ns[0:0:21.4844]
Rendering ...[DONE]:    [0:0:6.11401]
david@pc-ubuntu:~/Documentos/informatica_grafica-master/SmallPM_v1/SmallPM-linux/Ux$ ./smallpm -scene 1 -film-name q2_3_100k_1 -pm-total-photons 50000 -pm-nb-nearest-neighbor 1
Prepared scene to render: ered..[0:0:2.5e-05]
Photons Shot: ns[0:0:21.554]
Rendering ...[DONE]:    [0:0:3.34154]
david@pc-ubuntu:~/Documentos/informatica_grafica-master/SmallPM_v1/SmallPM-linux/Ux$ ./smallpm -scene 1 -film-name q2_3_100k_50 -pm-total-photons 50000 -pm-nb-nearest-neighbor 50
Prepared scene to render: ered..[0:0:3.4e-05]
Photons Shot: ns[0:0:21.0748]
Rendering ...[DONE]:    [0:0:14.9859]
david@pc-ubuntu:~/Documentos/informatica_grafica-master/SmallPM_v1/SmallPM-linux/Ux$ ./smallpm -scene 1 -film-name q2_3_100k_100 -pm-total-photons 50000 -pm-nb-nearest-neighbor 100
Prepared scene to render: ered..[0:0:3.4e-05]
Photons Shot: ns[0:0:21.057]
Rendering ...[DONE]:    [0:0:24.4828]
```

Figura 12: *Costes temporales de la generación de las figuras 6, 7, 8, 9, 10 y 11*

Cuestión 2.4:

La principal diferencia entre una solución calculada con *Ray tracing* y *Photon Mapping*, una vez han convergido ambos algoritmos tras el tiempo de cálculo necesario es las cáusticas. Con *Photon Mapping* al tener un mapa de fotones únicamente dedicado a las cáusticas se pueden obtener muy buenos resultados a cambio de introducir cierto sesgo, ya que en el mapa de fotones de cáusticas habrá una gran concentración de los mismos en los puntos correspondientes en la escena por lo que la estimación de la radiancia de las cáusticas será más precisa. En el caso de *Ray tracing* resulta más difícil reconstruir el camino de una cáustica, y para luces puntuales resulta imposible debido a que nunca se podrá intersectar con la luz puntual.

Se pueden observar en las figuras 13 y 14 que en el cálculo de la luz directa con *Ray tracing* todo lo que está debajo de la esfera de cristal sale en negro (debido a que la luz no es “visible”) y como la fuente de luz es puntual sería imposible

generar cáusticas, sin embargo con *Photon Mapping* se puede observar que se encuentran las cáusticas. La figura 15 sería la escena con todos los componentes.

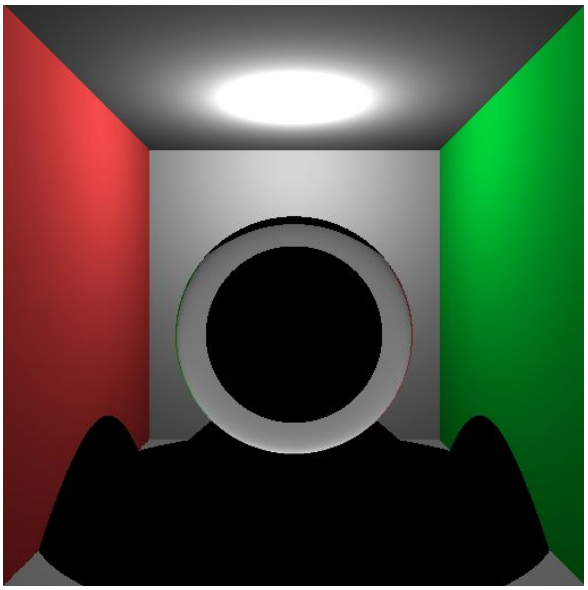

Figura 13: *Luz directa con Ray tracing*

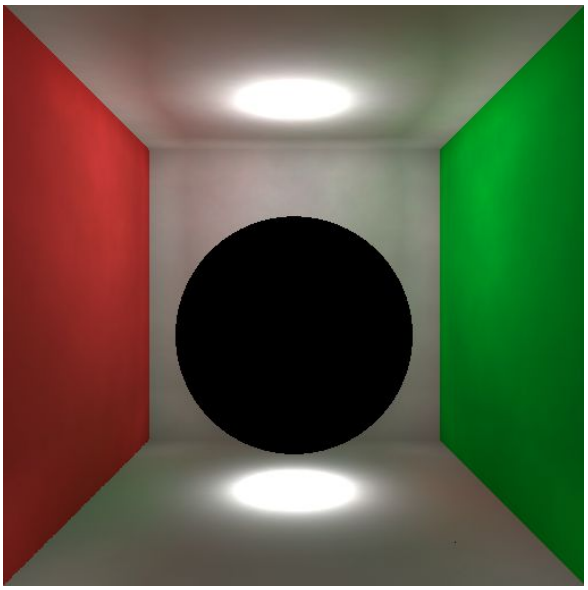

Figura 14: *Luz directa con Photon Mapping*

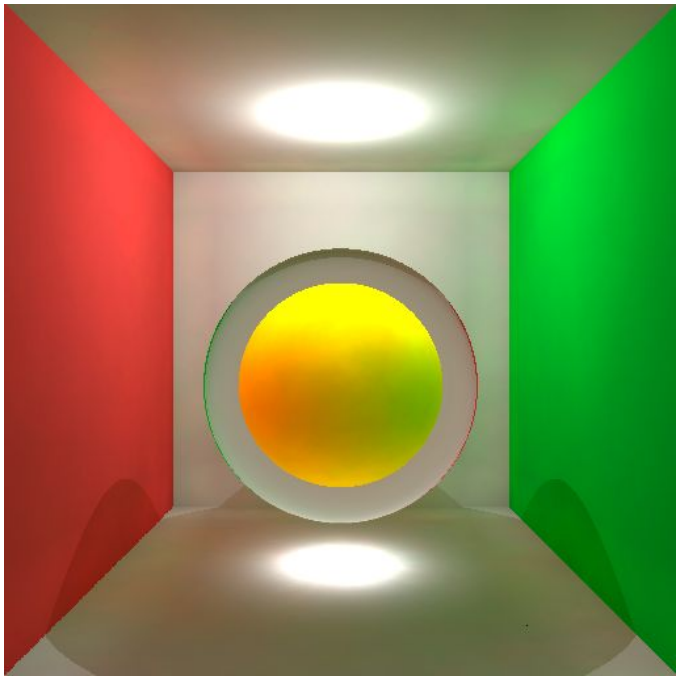

Figura 15: *Escena 4. Luz directa + mapa de fotones, 100K fotones emitidos y 500 fotones en la estimación de radiancia.*

A continuación se muestran dos escenas desarrolladas con el algoritmo, figuras 16 y 17. En la escena 16 se muestra un conejo de cristal con una única fuente de luz puntual y en la escena 17 hay tres fuentes de luz putnuales.

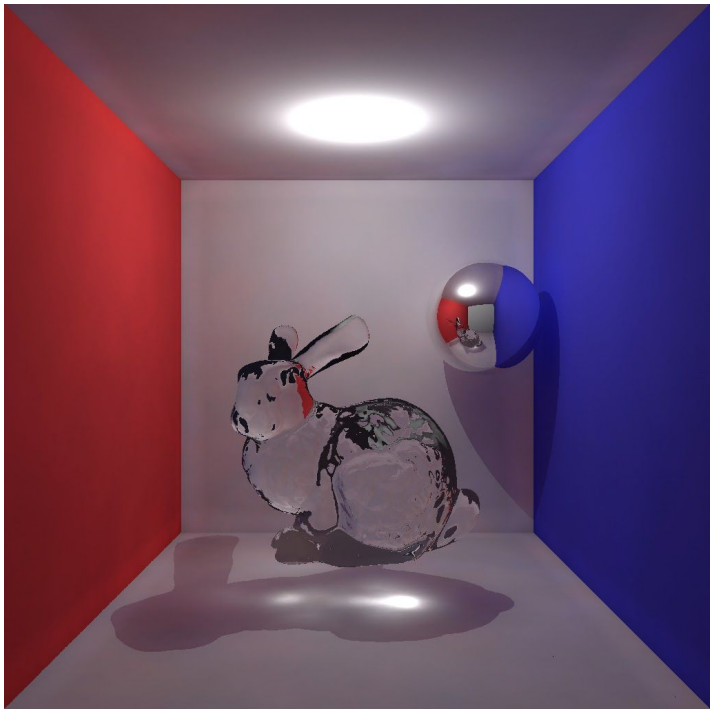

Figura 16: *400K fotones emitidos y 500 fotones utilizados para la estimación de la radiancia.*

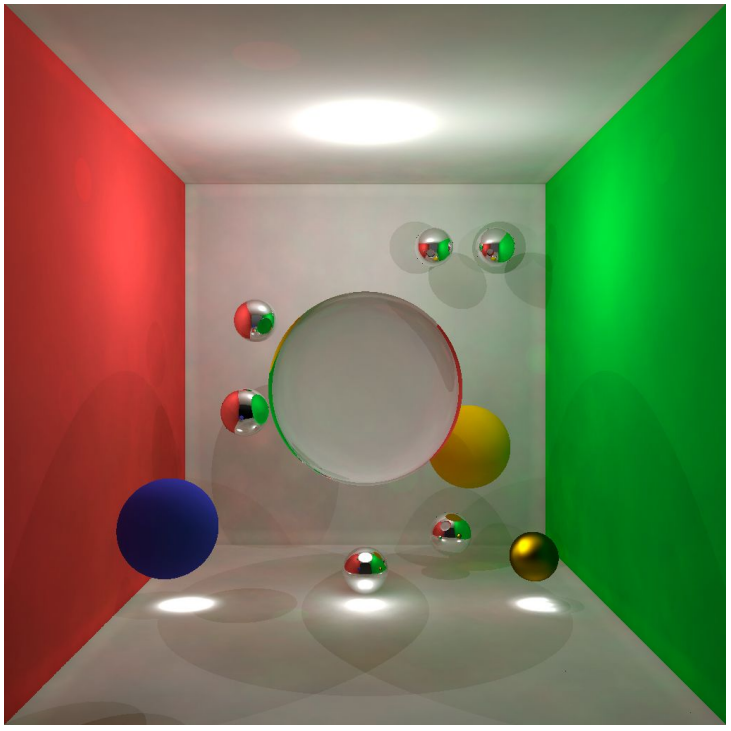

Figura 17: *1M fotones emitidos y 500 fotones utilizados para la estimación de la radiancia.*

Bibliografía

[1] - *Wikipedia (9 January 2020, at 18:48). Solid angle.* https://en.wikipedia.org/wiki/Solid_angle

[2] - *Graphics and Imaging Lab. Photon Mapping.* https://moodle.unizar.es/add/pluginfile.php/2303601/course/section/356530/40_photon_mapping.pdf?time=1574916009098

[3] - *Graphics and Imaging Lab. Ray tracing.* https://moodle.unizar.es/add/pluginfile.php/2303601/course/section/356530/06-raytracing.pdf?time=1572262300716

[4] - *Graphics and Imaging Lab. Lighting.* https://moodle.unizar.es/add/pluginfile.php/2303601/course/section/356530/03-lighting-handout.pdf

[5] - *Henrik Wann Jensen (Sunday, July 23, 2000). A Practical Guide to Global Illumination using Photon Maps.* https://graphics.stanford.edu/courses/cs348b-00/course8.pdf

[6] - *Wikipedia (10 January 2020, at 13:55). Snell's Law.* https://en.wikipedia.org/wiki/Snell%27s_law