

Informática Gráfica

Photon Mapping

José Daniel Subías Sarrato (NIA: 759533)
Fernando Peña Bes (NIA: 756012)

2 de febrero de 2021

Índice

1. Introducción	1
2. Objetivos	2
3. Creación del mapa de fotones	3
3.1. Muestreo de la luz	3
3.2. Conservación de la energía	4
3.3. Selección de fuentes de luz	5
4. Estimación de la radiancia	6
5. Propiedades de los materiales	8
5.1. Luz directa y sombras	8
5.2. Materiales Delta	9
6. Análisis de los parámetros	10
6.1. Número de fotones	11
6.2. Numero de vecinos	11
6.3. Kernels	12
6.3.1. Gaussiano	12
6.3.2. Cono	13
6.4. Medios participativos	14
6.4.1. Volumetric Photon Mapping	15
6.4.2. Emisión de los fotones	15
6.4.3. Estimación de la radiancia	16
7. Conclusiones	19
8. Control de esfuerzos	20

1. Introducción

Dentro del gran conjunto de disciplinas que abarca la informática, la informática gráfica desempeña la tarea de utilizar los ordenadores para generar imágenes. Esta rama de las ciencias de la computación se basa la aplicación de conceptos matemáticos como: geometría, estadística y álgebra así como de las leyes físicas. El principal objetivo es intentar resolver el problema de: dado un modelo que intenta representar la realidad, ¿de qué color pinto los píxeles de una imagen para representar dicho modelo de la forma más real posible? Es aquí donde surgen diversos algoritmos que intentan resolver este problema de la manera más eficiente posible. A lo largo de la asignatura de *Informática Gráfica*, se han descrito los algoritmos más utilizados en el entorno empresarial.

En el primer trabajo se implementó el algoritmo de *Path Tracing* para el renderizado de imágenes. Este algoritmo recibe como entrada una lista de geometrías en \mathbb{R}^3 y un cámara representada como un sistema de coordenadas contenido en \mathbb{R}^3 . Como salida, el algoritmo genera una imagen en \mathbb{R}^2 que representa fielmente la geometría recibida vista desde el punto de enfoque de la cámara. Este algoritmo resulta ser muy potente y eficiente en comparación con otros algoritmos como *Ray Tracing*. Sin embargo, este algoritmo presenta ciertas limitaciones a la hora de generar efectos visuales como por ejemplo cáusticas. Para lograr esto sería necesario que en el algoritmo se lanzara rayos desde las fuentes de luz. Es sobre esta idea sobre la que se base el algoritmo de *Photon Mapping* a implementar en este proyecto.

Photon Mapping parte de la idea de emitir energía desde la fuentes de luz, consiste en emitir un número N de fotones cargados de energía por toda la escena. Posteriormente, se lanzan rayos desde la cámara y se estima la radiancia cuando un rayo impacta con un material difuso.

Como punto de partida se cuenta con la implementación de un *Ray Tracer* creado por los profesores de la asignatura. El objetivo será diseñar e implementar los procesos de emisión de fotones y estimación de la radiancia, para conseguir un algoritmo de *Photon Mapping* que logra capturar efectos que eran imposibles con *Path Tracing*.

En esta memoria se recoge una explicación detallada de la metodología seguida por el equipo de trabajo. A su vez, contiene explicaciones de los principios matemáticos y las decisiones tomadas para su correcta implementación. Puesto que las salidas del algoritmo son resultado gráficos, se han llevado a la práctica numerosos experimentos para mostrar el potencial de la solución final.

2. Objetivos

Los objetivos de este proyecto comprenden varios aspectos, entre los que destacan:

- Aprender los concepto básicos del campo de la *Informática Gráfica*, para el renderizado de imágenes dada un determinada geometría y un punto de vista.
- Comprender los principios matemáticos básicos sobre los que se desarrolla el campo de la *Informática Gráfica*.
- Estudiar el principio de *conservación de la energía* para implementar un proceso de emisión de fotones, que permita la correcta creación de un mapa de fotones.
- Conocer el proceso de estimación de la radiancia sobre una superficie y implementar de manera opcional distintos kernels para evitar la sobrestimación.
- Entender el comportamiento de la luz en la naturaleza y su interacción con los distintos materiales, intentando comprender sus propiedades para estudiar los efectos de profundidad, color bleeding e iluminación global.
- Ver la aplicación de algoritmos probabilistas como *Ruleta Rusa* en *Informática Gráfica* para el muestreo de fuentes de luz.

3. Creación del mapa de fotones

A diferencia del algoritmo de *Path Tracing*, en *Photon Mapping* la ejecución del algoritmo no parte desde la cámara. Como primer paso, se debe generar un número N de fotones desde las fuentes de luz presentes en la escena. El objetivo intentar distribuir la energía de las fuentes por la escena, para posteriormente estimar la radiancia de los píxeles de la imagen mediante *Ray Tracing*. Este último proceso se explica de forma más amplia en el Apartado 2. En esta sección se procede a hacer una explicación detallada de las decisiones tomadas para hacer una correcta distribución de la energía por la imagen.

3.1. Muestreo de la luz

La emisión de la energía desde las fuentes de luz supone distribuir toda la energía de las mismas entre un número N de fotones. Dicha emisión es dependiente de la geometría de la fuente pues deben ser emitidas en direcciones perpendiculares a la superficie. En la implementación de este trabajo solamente se ha realizado la inclusión de luces puntuales. Por esta razón en este apartado solamente se explica el muestreo implementado para este tipo de geometrías.

Puesto que las luces puntuales son representadas como puntos en espacios \mathbb{R}^3 , estas pueden ser consideradas como esferas y se puede aplicar sobre ellas *Muestreo por el ángulo sólido*. Esto requiere la generación de dos ángulos θ y ϕ en los rangos $[0, 2\pi]$ y $[-\pi, \pi]$ respectivamente. Este tipo de muestreo por importancia requería de la generación de dos números aleatorios, ξ_θ y ξ_ϕ para muestrear las correspondientes funciones de probabilidad acumulada inversa c^{-1} en el intervalo correcto. Para el ángulo θ , al utilizar como función de probabilidad la función $\cos \theta$, el ángulo resultante correspondía con $\text{arc cos } \xi_\theta$, donde $\xi_\theta \in [-1, 1]$. Por otro lado, para el ángulo ϕ al utilizar el muestreo por ángulo sólido, la función probabilidad correspondía con $\frac{1}{2\pi}$. Esto implicaba que el ángulo resultante fuese generado por $2\pi\xi_\phi$, donde $\xi_\phi \in [0, 1]$. Una vez generados los ángulos θ y ϕ , se podía calcular la dirección de emisión del fotón mediante la siguiente expresión.

$$d_i = \begin{pmatrix} \sin \theta \cdot \cos \phi \\ \sin \theta \cdot \sin \phi \\ \sin \phi \cdot \cos \theta \end{pmatrix} \quad (1)$$

En la siguiente figura puede apreciarse un ejemplo de *Muestreo por ángulo sólido*, donde puede verse que las muestras se distribuyen uniformemente a lo largo de la esfera.

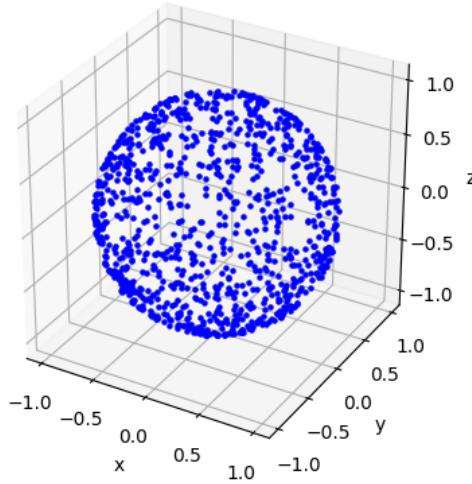


Figura 1: Ejemplo de muestreo con ángulo sólido con 1000 muestras

La imagen de la figura anterior puede ser generada ejecutando el script en Python entregado llamado `sphere.py`.

3.2. Conservación de la energía

Una vez resuelto el problema de la distribución de la energía de las fuentes de luz puntuales de forma uniforme, había que asignarle a cada fotón emitido una cantidad de energía de total emitido por la fuente de luz. Si consideramos la siguiente ecuación para definir la distribución de la energía a lo largo de toda la esfera:

$$\int_0^{2\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos \theta \sin \theta d\theta d\phi \quad (2)$$

Aplicando las correspondientes simplificaciones por realizar un muestreo por ángulo sólido, el resultado es la siguiente integral:

$$\int_0^{2\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos \theta d\theta d\phi = 4\pi \quad (3)$$

De este modo, puede expresarse la energía emitida en un fotón como

$$E_p = \frac{4\pi E_l}{N_l} \quad (4)$$

donde E_l corresponde con la energía total de la fuente de luz y N_l con el número de fotones emitidos por la fuente de luz. De este modo se consiguió que toda la energía de la fuente de luz fuese emitida por toda la escena.

Una vez explicado tanto la forma de muestreo de la dirección de emisión de los fotones y la distribución de la energía. Puede responderse la pregunta 1.1 presentada en el enunciado.

Question 1.1: *Note that in the provided ray tracer the photon random walk is already implemented (function PhotonMapping::trace_ray in PhotonMapping.cpp, together with a KD-Tree to accelerate photons' look-ups. If you choose to use the provided code, you will need to explain thoroughly what the function trace ray does and why. The more descriptive, the more you will show that know what's going on. On the other hand, if you use your own ray tracer, it is expected that you also explain thoroughly the random walk you implemented.*

Puesto que el grupo de trabajo a decidió utilizar el algoritmo de *Ray Tracer* proporcionado por los profesores de la asignatura, hubo que realizar un estudio detallado del método `PhotonMapping::trace_ray`. En primer lugar, se pudo ver que el método recibía como parámetros, un rayo en la dirección de emisión, una energía y varias listas de fotones donde posteriormente se guardarían los fotones del *KD-tree* global y el *KD-tree* de cáusticas, así como un flag para indicar si hay cálculo de iluminación directa por mapa de fotones. Con estos parámetros la función emitía un rayo en la dirección y calculaba la intersección de dicho rayo en la escena. A partir de aquí podían darse varias situaciones:

Si el rayo no intersecta con ninguna geometría el método retorna la correspondiente tupla *RGB* que representa la transferencia. En caso contrario si el rayo intersecta con un material, pueden darse dos situaciones. La primera se debe a que el rayo choca con una geometría con un material delta, en tal caso se calcula la dirección de salida y no ha perdido de energía. En caso de que el material no sea delta si el flag de iluminación directa toma valor `true`, se guarda el primer impacto del fotón y se remite con una dirección y una energía dependientes del material. En caso de que el flag de iluminación directa sea igual a `false`, el primer impacto del fotón no es almacenado y el fotón es remitido en función del material. En dicha función puede verse que hay un límite de rebotes, es decir si el fotón rebota un número N de veces, o lo es absorbido por el material en función del algoritmo de *Ruleta Rusa*.

3.3. Selección de fuentes de luz

En los dos apartados anteriores se ha descrito cómo determinar la dirección de emisión de los fotones y la cantidad de energía que transportan. No obstante, se ha hecho desde un punto de vista en el que se asume que hay una única fuente de luz en el entorno. Es aquí donde surge el problema de cómo distribuir los N fotones a lanzar entre las L luces. Si estudiamos el problema desde un punto de vista lógico, puede considerarse la heurística de que aquella fuente de luz que tenga más energía debe emitir más fotones. Es por esta razón que se consideró aplicar el algoritmo de *Ruleta Rusa* para abordar el problema.

Utilizar este algoritmo probabilista para seleccionar la luz emisora de los fotones, permite realizar un muestreo en el que la luz con más energía tiene más probabilidad de ser elegida.

Si definimos E como la energía total,

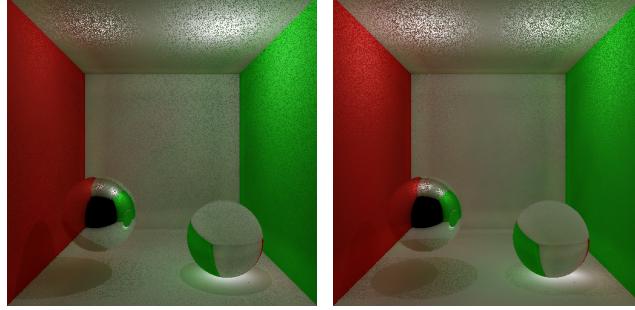
$$E = \sum_{i=0}^{L-1} E_i \quad (5)$$

donde E_i representa la energía de la i -ésima luz de la escena.

Podemos asignar a cada luz una probabilidad de evento realizando una normalización por E de la energía de la luz, del siguiente modo:

$$p_i = \frac{E_i}{E} \quad (6)$$

Una vez echo esto, se pudo utilizar la generación de un número aleatorio ξ_l y seleccionar el evento mediante *Ruleta Rusa*. A continuación, pueden verse varias escenas en las que hay más de una fuente de luz puntual.



(a) Escena con dos luces puntuales presentes haciendo muestreo por *Ruleta Rusa* donde la luz de la izquierda tiene una energía de 5W y la de la derecha de 15W por lo que la luz de la derecha tiene más probabilidad de ser muestreada.
(b) Escena con dos luces puntuales presentes haciendo muestreo por *Ruleta Rusa* donde la luz de la izquierda tiene una energía de 10W y la de la derecha de 10W por lo que ambas tienen la misma probabilidad de ser muestreadas.

Figura 2: Escena número 1 generada con varias luces puntuales.

4. Estimación de la radiancia

En el apartado anterior se ha explicado de forma detallada el proceso implementado para la creación del mapa de fotones. No obstante, el algoritmo de *Photon Mapping* consta de dos fases, siendo la segunda la estimación de la radiancia de los píxeles. En este apartado se va a explicar de forma amplia la implementación realizada para este proceso.

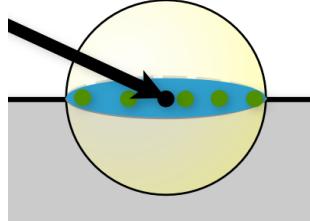
En el algoritmo de *Path Tracing*, la radiancia se calculaba en función del número de rebotes que realizaban los rayos. En cada uno de estos rebotes se realizaba el cómputo de

la *Ecuación de Render*, aplicando los algoritmos probabilistas de *Monte Carlo* y *Ruleta Rusa*. Dicha ecuación calcula la radiancia que llega a la cámara en un determinado punto de una superficie, en función de la normal y el ángulo de incidencia de la luz tal y como se muestra a continuación:

$$L_0(x, w_0) = \int_{\Omega} L_i(x, w_i) f_r(x, w_i, w_o) |n \cdot w_i| dw_i \quad (7)$$

La ecuación está compuesta por tres términos, que definen la radiancia que llega a la cámara desde el punto x en la dirección w_o . El primer término $L_i(x, w_i)$ determina, la radiancia incidente recibida desde una dirección w_i . Esta puede ser emitida tanto por luces de área como lucen puntuales en el espacio. En apartados posteriores se realizará un análisis más detallado de cómo se calcula la radiancia incidente cuando un camino impacta con la superficie de un material. El segundo de los términos $f_r(x, w_i, w_o)$ corresponde con el material de la superficie sobre la que se estima la radiancia. Dicho término es la denominada *Bidirectional Reflectance Distribution Function (BRDF)* del material. Esta función modela cómo la luz es reflejada y distribuida a lo largo de la semiesfera al impactar con una superficie opaca. Por último, es de gran importancia considerar el término $|n \cdot w_i|$, dependiente del coseno que forman n (normal de la superficie) y w_i (rayo emitido desde x hasta la fuente de luz). Esto tiene una gran importancia en el cálculo de la radiancia incidente, pues cuanto más perpendicular es la dirección de la luz incidente mayor es la energía recibida.

En *Photon Mapping*, la ecuación de *Ecuación de Render* no es calculada cuando un rayo de la cámara impacta con una determinada superficie. Esto se debe a que la radiancia es una estimación en función el flujo de luz que llega a la superficie. Esta se calcula haciendo una expansión del punto de impacto a una esfera, y calculando el flujo en función de los fotones próximos a dicho punto. De este modo que la radiancia puede aproximarse de la siguiente manera.



$$L_r(x, w) \approx \frac{1}{\pi r^2} \sum_{p=1}^N f_r(x, w_p, w) \Phi_p \quad (8)$$

donde N corresponde con el número de fotones que se selecciona alrededor del punto de impacto, r el radio de búsqueda que corresponde con la distancia al fotón más lejano, $f_r(x, w_i, w_o)$ la *BRDF* del material y Φ_p la energía del fotón p .

Esta estimación permite obtener un valor aproximado de la radiancia, pero como se ha dicho anteriormente, se trata de una mera aproximación. Por esta razón resulta difícil

encontrar el número óptimo de fotones pues seleccionar poco fotones vecinos implica una subestimación de la radiancia y seleccionar demasiado provoca una sobreestimación.

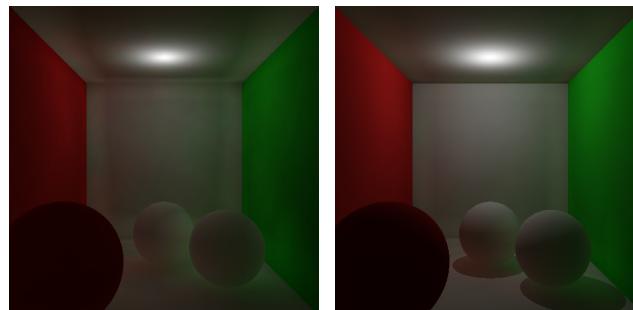
5. Propiedades de los materiales

Una vez realizada la implementación del algoritmo de *Photon Mapping* explicada en los anteriores, se procedió a generar un conjunto de imágenes para analizar las prestaciones del algoritmo en función de sus parámetros. En este apartado se va a proceder a mostrar los resultados generados, intentando responder a las cuestiones planteadas en el enunciado del proyecto.

5.1. Luz directa y sombras

La generación de sombras en las imágenes resultaba sencilla en el algoritmo de *Path Tracing*, al ser generadas por la oclusión de la luz entre geometrías. El cálculo de la iluminación directa en un punto podía hacerse mediante el trazado de rayos de sombra aplicando *Next Event Estimation*. En *Photon Mapping* el cálculo de la luz directa se puede realizar de dos formas: análogamente a como se realizaba en *Path Tracing*, o almacenando el primer rebote de los fotones en el *KD-tree*. Dicho esto, en esta sección se procede a mostrar resultados para intentar responder a las siguientes preguntas:

Question 2.1: *Render two images of Scene2 (using -scene 2) with only direct illumination (DI), each image where DI is computed with ray tracing and with photon mapping respectively. You can compute direct illumination with photon mapping setting the parameter direct=true in PhotonMapping::trace_ray. What are the differences between the direct illumination computed using ray tracing photon mapping? Which one do you think is more accurate and why?*



(a) Escena número 1 renderizada con luz directa por mapa de fotones. (b) Escena número 1 renderizada con luz directa por ray tracing.

Figura 3: Escena renderizada con luz directa por ray trace y mapa de fotones.

En respuesta a la pregunta 2.1 pude verse como en la imagen de la derecha, en la cual el cálculo de luz directa se hace mediante trazados de rayos, las sombras que aparecen están más remarcadas que la imagen de la izquierda. En esta última el cálculo de la luz directa

se hace mediante el mapa de fotones habiendo guardado el primer rebote de cada fotón. Que el cálculo de luz directa se haga mediante mapa de fotones no asegura que en el radio de búsqueda a la hora de estimar la radiancia se incluyan los fotones correspondientes al primer rebote, por lo que su energía no se tendría en consideración. Por otro lado aplicar trazado de rayos para el cálculo de luz directa hace en cada estimación, se tenga siempre en cuenta la contribución de las luces puntuales y esta sea más precisa, representando las sombras duras que producen las luces puntuales.

Question 2.4: *In the scene shown in Figure 1 you can see a diffuse torus inside glass, illuminated by a point light source. What would be the difference (in a converged scene) between rendering the direct illumination using ray tracing and using photon mapping? Render the Scene4 (using -scene 4) with both ray traced and photon-based direct illumination.*

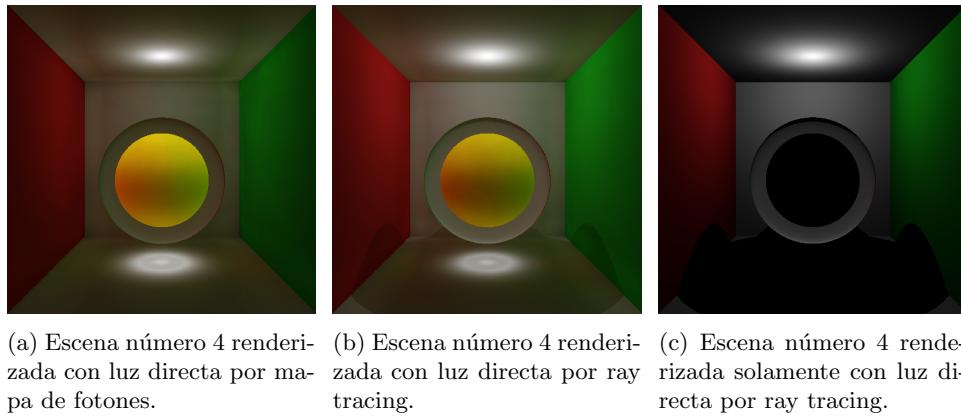


Figura 4: Misma escena generada con luz directa por mapa de fotones, ray tracing, únicamente con luz directa por ray tracing

En este caso, la radiancia de la bola naranja se estima mejor con renderizando la luz directa con el mapa de fotones, ya que la iluminación que se calcula mediante rayos de sombra permite la creación de sombras mas marcadas. No obstante como puede verse en la tercera imagen si el cálculo es realizado únicamente por luz directa con trazado de rayos, los que impactan con el toroide no tiene contribución. Esto se debe a que al trazar el rayo de sombra este choca con el cristal y la luz no contribuye.

5.2. Materiales Delta

Como se ha mencionado en el Apartado 3.2, los fotones son almacenados en los *KD-trees* cada vez que impactan con una superficie. El almacenamiento del primer rebote es determinado en función de si se hace cálculo de luz directa con mapa de fotones o con *Next Event Estimation*. Por otro lado, durante la creación del mapa de fotones también se crea un segundo mapa para almacenar los fotones correspondientes a los rebotes en materiales delta, que no producen pérdida de energía. Esto permite aislar el cálculo de la radiancia de la superficie de la estimación de la energía correspondiente a cáusticas. A

continuación, se muestra un conjunto imágenes para dar respuesta la siguiente pregunta:

Question 2.2: Render Scene1 (using -scene 1) using photon mapping. This scene has two spheres with delta BSDF. Describe how do you shade each of these spheres and why.

La escena 1 se muestra en la Figura 5. La esfera de la izquierda tiene material perfecto especular, y la de la izquierda un material transmisor perfecto.

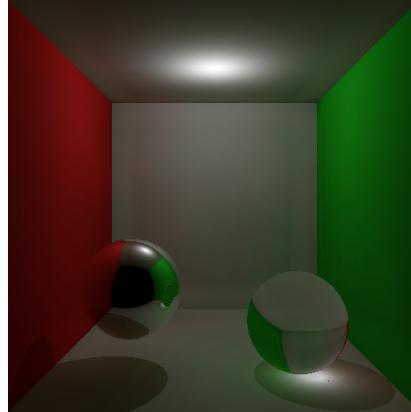


Figura 5: Renderizado de la escena 1 con 100.000 fotones lanzados y 1000 vecinos utilizados para realizar la estimación de la radiancia y luz directa por trazado de rayos.

Como las dos esferas son delta, se sombrean de forma similar con respecto a la luz que reciben de las fuentes. Como sobre los materiales delta no se almacenan fotones, cuando el rayo choca con estos, se utiliza *Ray Tracing* para seguir la dirección de reflexión, en el caso del especular, o transmisión, en el caso del transmisor. El rayo va rebotando por la escena hasta llegar a un material difuso.

Cuando se llega al material difuso se estima la radiancia usando tanto el mapa de fotones global como el mapa de cáusticas. Sin embargo, como las cáusticas se concentran sobre los materiales dieléctricos, cuando se estima la radiancia en un punto alejado de estos el radio de estimación es muy grande, lo que implica que los fotones cáusticos tengan una contribución despreciable en ese punto.

6. Análisis de los parámetros

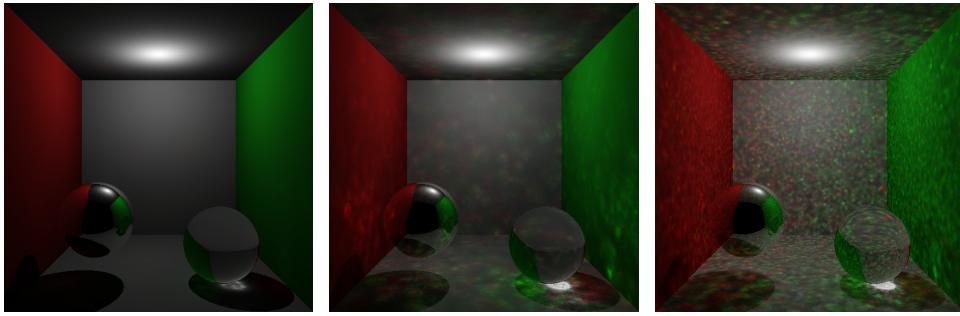
En este apartado se pretende mostrar un análisis detallado de los resultados producidos por el algoritmo de *Photon Mapping* implementado, variando el número de fotones emitido, el número de vecinos utilizado para estimar la radiancia y el número de rayos por píxel. Con las imágenes renderizadas se pretende también responder de forma detallada a la siguiente pregunta:

Question 2.3 Render Scene1 using an increasing number of photons (e.g. n = 1K, 10K, 100K), fixing the number of nearest neighbors in radiance esti-

mation to $k = 10$. Now, render the same scene with $n = 100K$ photons, and increasing the nearest neighbors density estimation kernel with $k = 1, 10, 50, 100$. Discuss the results obtained, in terms of cost and image quality.

6.1. Número de fotones

Este parámetro corresponde al número de fotones emitidos desde la fuentes de luz puntuales presentes en la escena. Variar este parámetro implica distribución más uniforme de la imagen, pues como puede verse en la figura 1 si cuantas más muestras más se asemeja la distribución de la energía a una esfera ideal. En las siguientes imágenes pueden verse ejemplos de escenas renderizadas variando el número de fotones.



(a) Imagen generada lanzando 1000 fotones.

(b) Imagen generada lanzando 10000 fotones.

(c) Imagen generada lanzando 100000 fotones.

Figura 6: Misma escena generada variando el número de fotones emitidos con un número de vecinos fijo $k = 10$.

Como puede apreciarse en la imágenes de la Figura 7 conforme aumenta el número de fotones para realizar, la estimación de la radiancia es más precisa. Cuando se usan pocos fotones se pierde detalle, debido a la mayor varianza en la estimación. Además, desaparecen efectos como el color bleeding.

Como el número de vecinos se mantiene fijo, el cambio en el número de fotones también implica que el área de estimación cambie de tamaño. En la primera imagen, además de usar muy pocos fotones, el área es muy grande y se pierde casi toda la información sobre iluminación global. En la tercera imagen se usa un número de fotones suficiente para mostrar efectos de iluminación global, pero el número de vecinos es demasiado bajo, provocando que el área de estimación sea muy pequeña y la imagen aparezca llena de pequeños círculos.

6.2. Número de vecinos

En el Apartado 4 se ha comentado que el número de vecinos seleccionados para la estimación de la radiancia, es difícil de ajustar. Esto se debe a que seleccionar el valor de este parámetro debe hacerse de manera manual, aunque existen extensiones del algoritmo original como *Progressive Photon Mapping*.

Seleccionar un número bajo de fotones implica que la radiancia es subestimada, pues no se tiene el flujo suficiente como estimarlas. Por otro lado, utilizar un número excesivo de fotones provoca una excesiva presencia de energía, implicando así que se sobreestima la radiancia calculada. En las siguientes figuras pueden verse ejemplos de imágenes renderizadas variando el número de vecinos utilizados para la estimación.

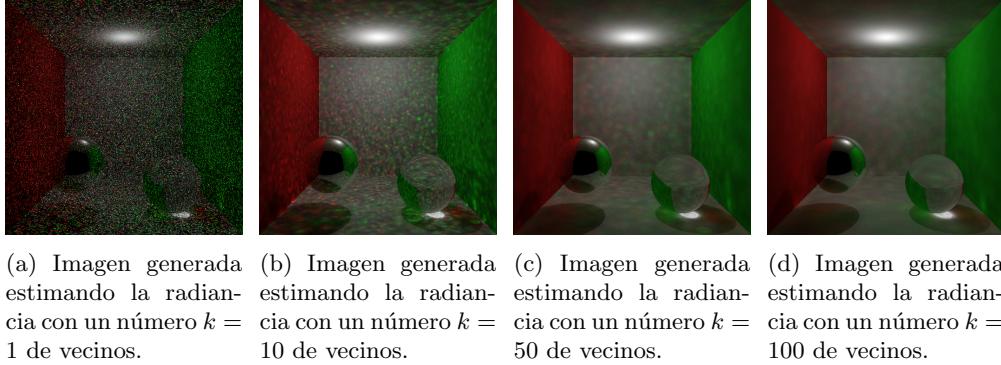


Figura 7: Misma escena generada variando el número de fotones emitidos con un número de fotones emitidos $n = 100.000$

6.3. Kernels

En repetidas ocasiones a lo largo del documento se ha mencionado que el número de fotones vecinos para estimar la radiancia, debe seleccionarse evitando que se produzca una sobreestimación o subestimación. Esto se debe a que el cálculo se hace de forma aproximada y puede existir con facilidad cierto error de aproximación. A pesar de esto existen distintos kernels que pueden aplicarse a la estimación para que el impacto de elegir mal los vecinos sea menor. Por esta razón se decidió implementar varios kernels en algoritmo de *Photon Mapping*.

6.3.1. Gaussiano

Este primer filtro asigna un peso w_{pg} a cada fotón durante el cómputo de la radiancia. Dicho peso viene dado por la siguiente expresión:

$$w_{pg} = \alpha \left[1 - \frac{1 - e^{-\beta \frac{d_p^2}{2r^2}}}{1 - e^{-\beta}} \right] \quad (9)$$

Con este peso, en el que r es el radio de búsqueda de fotones y d_p la distancia al fotón, puede realizarse la estimación de la radiancia.

$$L_r(x, w) \approx \frac{1}{\pi r^2} \sum_{p=1}^N f_r(x, w_p, w) \Phi_p w_{pg} \quad (10)$$

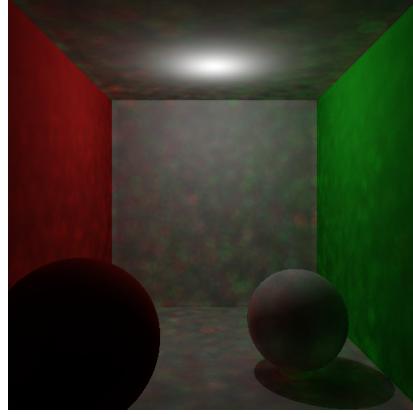


Figura 8: Renderizado de la escena por defecto con un número de vecinos $k = 50$ aplicando el kernel de Gauss con $\alpha = 0,918$ y $\beta = 1,953$.

6.3.2. Cono

Este de forma análoga al filtro anterior, este filtro asigna un peso w_{pc} a cada fotón en función de la distancia.

$$w_{pc} = \frac{d_p}{kr} \quad (11)$$

Donde r es el radio de búsqueda de fotones, d_p la distancia al fotón y k una constante mayor o igual que 1. Con estos parámetros y aplicando una distribución $1 - \frac{2}{3k}$ la estimación de la radiancia queda de la siguiente manera:

$$L_r(x, w) \approx \frac{1}{\pi r^2 \left(1 - \frac{2}{3k}\right)} \sum_{p=1}^N f_r(x, w_p, w) \Phi_p w_{pc} \quad (12)$$

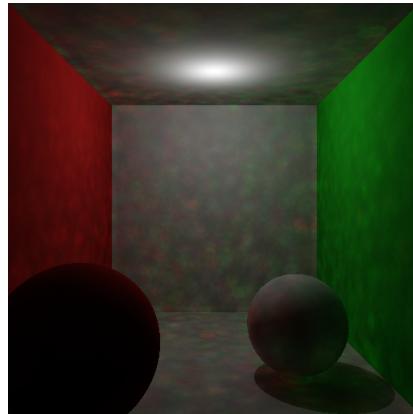


Figura 9: Renderizado de la escena por defecto con un número de vecinos $k = 50$ aplicando el kernel de Cono con $k = 2,0$.

6.4. Medios participativos

Durante la realización tanto de este trabajo como de la implementación del algoritmo de *Path Tracer*, se ha considerado que luz viajaba sin ser alterada por nada. Esto produce que las imágenes renderizadas sean ideales y los únicos efectos que aparecen en ellas son a causa de la interacción de la luz con los materiales. Sin embargo en la naturaleza esto no es así casi en ninguna situación, pues la luz suele interactuar con el medio por el que se propaga. Durante dicha propagación la luz puede estar sometida a diferentes efectos como: *Absorción*, *Emisión*, *Out-Scattering* y *In-Scattering*. Durante estos eventos la energía que viaja por un rayo es modificada a causa de la interacción con el medio.

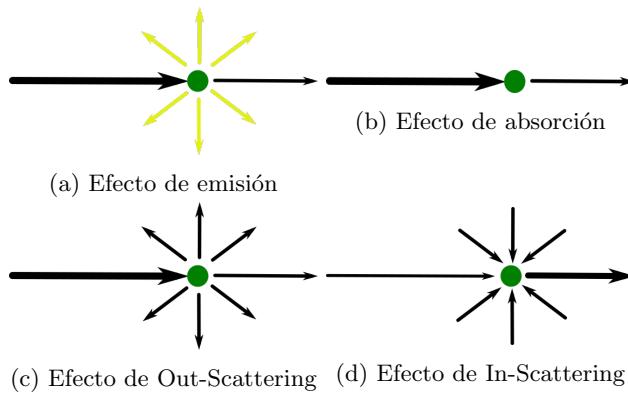


Figura 10: Posibles efectos que puede sufrir la luz al atravesar un medio.

Vistos estos cuatro efectos puede construirse la ecuación de transporte de luz, la cual estará pormenorizada en función de los coeficientes del medio. Dicha ecuación es la que se ve a continuación.

$$(\vec{w} \cdot \Delta) L(x \rightarrow \vec{w}) = \underbrace{-\sigma_a(x)L(x \rightarrow \vec{w})}_{\text{absorción}} - \underbrace{\sigma_s(x)L(x \rightarrow \vec{w})}_{\text{out-scattering}} + \underbrace{\sigma_a(x)L_e(x \rightarrow \vec{w})}_{\text{emisión}} + \underbrace{\sigma_s(x)L_i(x \rightarrow \vec{w})}_{\text{in-scattering}} \quad (13)$$

Donde $\sigma_a(x)$ corresponde con el coeficiente de absorción del medio y $\sigma_s(x)$ el coeficiente *Scattering*. No obstante para este trabajo meramente académico puede obviarse el suceso de emisión, puesto que puede complicar la implementación final. Descartando este tipo de evento y viendo que en la absorción y el *in-scattering* se pierde energía, estos dos eventos pueden agruparse en un solo término denotado como extinción. Haciendo estos cambios en la ecuación 13, el resultado es el que se muestra a continuación:

$$(\vec{w} \cdot \Delta) L(x \rightarrow \vec{w}) = \underbrace{-\sigma_t(x)L(x \rightarrow \vec{w})}_{\text{extinción}} - \underbrace{\sigma_s(x)L(x \rightarrow \vec{w})}_{\text{in-scattering}} \quad (14)$$

donde $\sigma_t(x)$ corresponde con $\sigma_a(x) + \sigma_s(x)$.

Una vez vista la ecuación 14, esta puede ser reescrita como la ecuación volumétrica de

render de la siguiente manera.

$$L(x \leftarrow \vec{w}) = T_r(x \leftrightarrow x_s) L(x \rightarrow -\vec{w}) + \int_0^s T_r(x \leftrightarrow x_t) \sigma_s(x_t) L_i(x \rightarrow -\vec{w}) dt \quad (15)$$

Siendo T_r la transmitancia del medido que representa la atenuación de la energía a lo largo del medio. Una vez explicada la ecuación que modela el comportamiento de la luz en un medio, es de importancia destacar que si este es homogéneo los coeficientes $\sigma_t(x)$ y $\sigma_s(x)$ son conocidos, y la transmitancia puede ser definida de la siguiente manera.

$$T_r(x' \leftrightarrow x) = e^{-||x' - x||\sigma_t} \quad (16)$$

Es por esto que solamente hubo que buscar una manera eficiente de computar esta la ecuación 15, de manera eficiente en el *Photon Mapping* desarrollado.

6.4.1. Volumetric Photon Mapping

Tras la explicación teórica anterior, pueden estudiarse diversos métodos para poder realizar el cómputo de la ecuación 15. Una primera aproximación podría ser asumir que la radiancia de *in-scattering* es una constante ambiental. Si se hace esta suposición la radiancia puede ser aproximada de las siguiente manera para medios homogéneos.

$$L(x \leftarrow \vec{w}) = \sigma_s L_i \frac{1 - e^{-s\sigma_t}}{\sigma_t} + e^{-s\sigma_t} L(x_s, \vec{w}) \quad (17)$$

No obstante esta aproximación se aleja bastante del comportamiento real de la luz en el medio. Es por este motivo que se decidió hacer una implementación un poco más sofisticada para poder simular medios participativos en la imágenes, de modo que se extendió el algoritmo de *Photon Mapping* a *Volumetric Photon Mapping*.

6.4.2. Emisión de los fotones

Partiendo de la consideración de que el medio participativo presente en las escenas, ocupaba todo el espacio. Hubo que hacer diversas modificaciones en los pasos del algoritmo explicado en los Apartados 3 y 4. Es por esto que en este apartado se procede a hacer una explicación de los cambios en el proceso de emisión de fotones.

En primer lugar es de importancia destacar que la implementación desarrollada asume que los medios son *isotrópicos*, es decir la distribución de scattering viene dada por una función de fase $p(x, \vec{w}', \vec{w})$ que distribuye la energía de forma uniforme dada la siguiente expresión.

$$p(x, \vec{w}', \vec{w}) = \frac{1}{4\pi} \quad (18)$$

Una vez decidida la distribución de la energía de los medios para los eventos de scattering, se hubo que hacer las correspondientes modificaciones para poder muestrear los eventos de *scattering* que sufre la luz durante el trayecto por el medio. Esto implicó la creación de luces puntuales virtuales para la simulación de los evento, implicando la inclusión de una estructura *KD-tree* adicional para almacenar los fotones que se encuentran “flotando” en

el ambiente. Con esto sólo faltaba implementar las modificaciones necesarias para generar los eventos. Siguiendo el estudio de [2], se sabe que la probabilidad de que se produzca un evento de scattering en un rayo de luz de una determinada distancia es de:

$$p = 1 - e^{-\int_{x_s}^x \sigma_t(\xi) d\xi} \quad (19)$$

Si calculamos la función de probabilidad acumulada inversa para muestrear a lo largo de la distancia del rayo se obtiene que la distancia puede ser muestreada por la siguiente expresión:

$$\text{distancia} = -\frac{\log(\xi)}{\sigma_t} \quad (20)$$

Con esta probabilidad puede aplicarse el algoritmo de *Ruleta Rusa* para determinar si la luz interactúa con el medio o pasa sin ser alterada. En caso de que se seleccione el evento de interacción el fotón sera remitido en una dirección seleccionada mediante *Muestreo por Ángulo Solido* con la misma energía. Sin embargo se decidirá nuevamente por *Ruleta Rusa* si dicho fotón es almacenado y remitido con la misma energía o absorbido. Esto último se hace en función del albedo del medio, el cual se puede expresar como $\frac{\sigma_s}{\sigma_t}$.

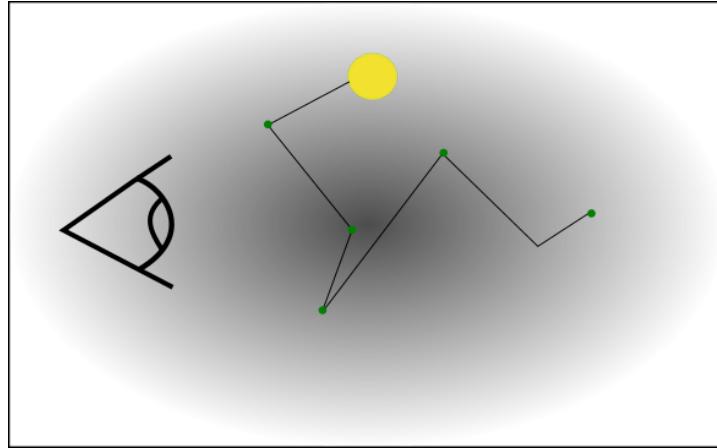


Figura 11: Esquema de la interacción de la luz en un medio participativo.

Dicho todo esto es de importancia destacar que a nivel de implementación, se que se guarda el primer rebote del fotón en el medio, para no tener que hacer cálculo de la iluminación directa trazando rayos.

6.4.3. Estimación de la radiancia

Implementando un método para guardar la información de los evento de scattering producidos durante la emisión de los fotones en el medio. Se tuvo que considerar la implementación de un método de cálculo eficiente para la expresión 15. Es por esto que siguiendo las explicaciones de las clases de teoría, se decidió utilizar el algoritmo de *Ray*

Marching. Dicho algoritmo permite realizar un cálculo eficiente, al transformar la integral de la ecuación 15 en un sumatorio resultando la siguiente expresión.

$$L(x \leftarrow \vec{w}) = T_r(x \leftrightarrow x_s) L(x \rightarrow -\vec{w}) + \sum_{t=0}^s T_r(x \leftrightarrow x_t) \sigma_s(x_t) L_i(x \rightarrow \vec{w}) \Delta t \quad (21)$$

Con esta discretización de la ecuación, puede hacerse una estimación de la radiancia en cada punto x_t del rayo. El problema surge al realizar el cálculo del término L_i , pues representa la radiancia incidente en el punto analizado según la siguiente ecuación.

$$L_i(x \rightarrow \vec{w}) = \int_{\Omega_{4\pi}} p(x, \vec{w}', \vec{w}) L(x_t, \vec{w}_t) dw_t \quad (22)$$

Nuevamente para poder resolver el problema hubo que discretizar la ecuación, resultando la siguiente expresión.

$$L_i(x \rightarrow \vec{w}) = \sum_{p=1}^k p(x, \vec{w}', \vec{w}) L(x_t, \vec{w}_t) \frac{\Phi_p}{\frac{3}{4}\pi r^3} \quad (23)$$

Finalmente para aplicar el cálculo de la radiancia se evaluaban diferentes puntos del rayo calculando la distancia entre estos mediante la siguiente ecuación:

$$distancia = -\frac{\log(\xi)}{\sigma_t} \quad (24)$$

Donde $\xi \in [0, 1]$. En la siguiente figura se muestra un esquema gráfico del muestreo del rayo.

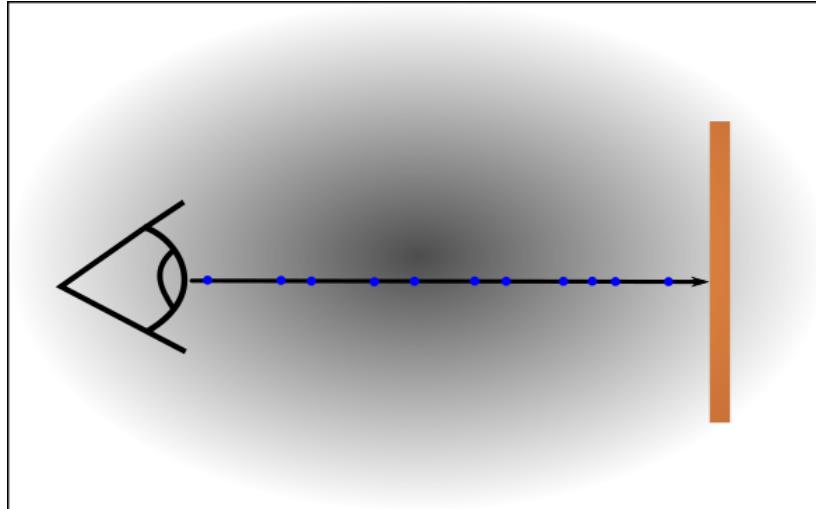


Figura 12: Esquema de muestreo del rayo en *Ray Marching*

Finalmente tras aplicar *Ray Marching* para estimar la radiancia que llaga a la cámara, la cual ha sido modificada por el medio. A esta radiancia había que sumarle la radiancia que llega al punto de impacto del rayo ya sea directa o indirecta, atenuada por transmitancia en función la longitud del rayo. Tras esto ya se podía ejecutar el algoritmo para lograr generar efectos como por ejemplo niebla en la imagen. Por desgracia para el equipo, tras haber realizado la implementación de los pasos descritos en los apartados anteriores, no se logró una correcta implementación de los medios participativos. Esto puede verse reflejado en la imágenes que se muestran continuación con $\sigma_t = 8$ y $\sigma_s = 0,5$, pues pude verse que hay presente un fallo en la estimación de la radiancia.

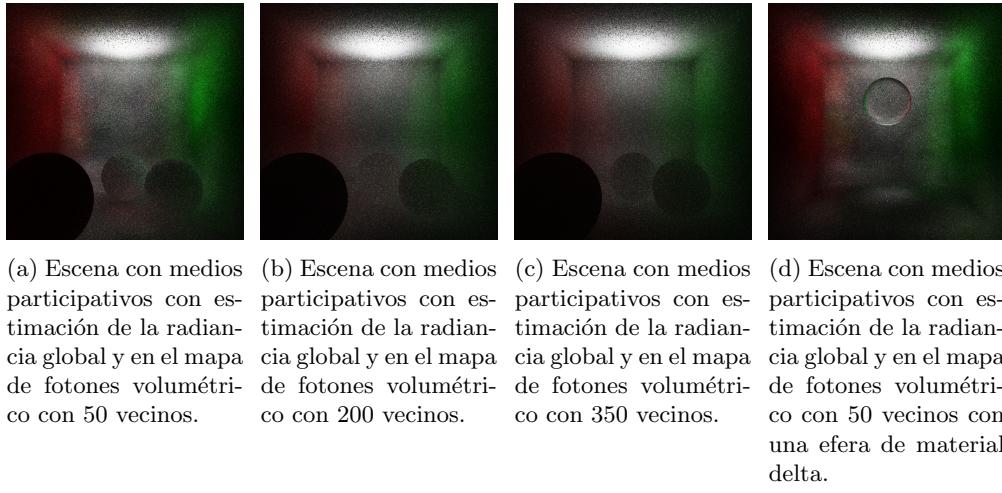


Figura 13: Imagenes generadas con la presencia de medios participativos en la escena.

7. Conclusiones

- Se han comprendido a la perfección los conceptos los elementos básicos de la *Informática Gráfica* para resolver el problema de representación de modelos 3D en 2D.
- Se ha entendido por completo la base matemática sobre la que se sustenta la informática gráfica, más concretamente el algoritmo de *Photon Mapping*.
- El estudio del principio de conservación de la energía, ha permitido tener una correcta compresión del mismo para poder implementar una proceso de creación de mapas de fotones para el algoritmo de *Photon Mapping*.
- Realizar la implementación de la estimación de la radiancia en superficies difusas, supuso al equipo tener que realizar el esfuerzo de entender este proceso del algoritmo de *Photon Mapping*.
- Se han entendido las diversas propiedades de la luz en la naturaleza para poder tenerlas en cuenta en la generación artificial de gráficos por ordenador.
- La necesidad de implementar algoritmos probabilistas como *Ruleta Rusa* y *Monte-carlo* ha permitido comprender comprender el funcionamiento de ambos algoritmos y la necesidad de los mismos para la optimización del algoritmo de *Photon Mapping*.

8. Control de esfuerzos

José Daniel Subías Sarrato

Horas invertidas: 64

Tareas realizadas:

- Compresión del código proporcionado. 6h
- Implementación de la creación del mapa de fotones. 7h
- Implementación de la estimación de la radiancia. 5h
- Implementación de kernels opcionales. 3h
- Redacción de la memoria. 22 h
- Medios participativos. 16h
- Diseño de imágenes para la memoria. 5h

Fernando Peña Bes

Horas invertidas: 48

Tareas realizadas:

- Compresión del código proporcionado. 6h
- Implementación de la creación del mapa de fotones. 15h
- Implementación de la estimación de la radiancia. 20h
- Visualización del mapa de fotones. 2h
- Redacción de la memoria. 5h

Referencias

- [1] Link la documentacion de la STL de C++. https://www.cppreference.com/Cpp_STL_ReferenceManual.pdf.
- [2] Link al paper de Volumetric Photon Mapping <http://www.cs.otago.ac.nz/cosc455/p311-jensen.pdf>