Metropolis Light Transport

Miguel Angel Astor Romero

5 de diciembre de 2016

Agenda

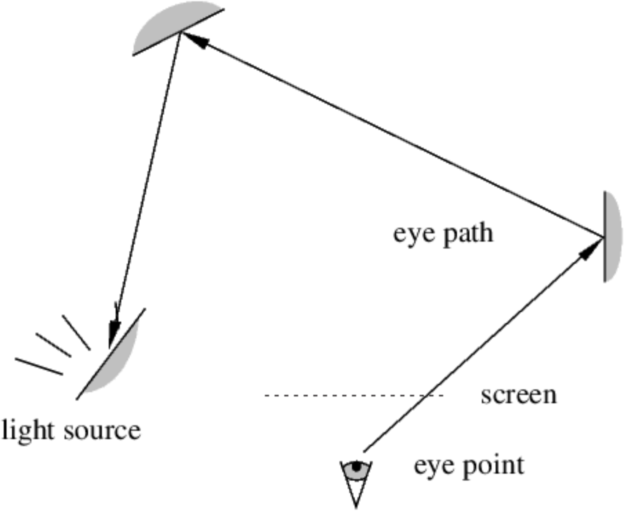
Introducción

- *Path Tracing* es una de las técnicas utilizadas para calcular iluminación global en una escena:
  - Aproxima una solución de la *Rendering Equation*
  - Es no sesgado
- Sin embargo, *Path Tracing* no es infalibre:
  - Funciona muy bien para ciertos tipos de escenas
  - Deja mucho que desear en otros casos

Path Tracing - Definición

- También se conoce como Montecarlo *Ray Tracing*
- Por cada pixel de la imagen final se lanzan múltiples rayos a la escena
  - Cada rayo es llamado "muestra"
  - Estas muestras intersectar un objeto o intersectar una fuente de luz
    - Cuando una muestra intersecta un objeto esta se refleja según la BRDF del objeto
    - Las muestras pueden reflejarse una cantidad finita de veces
  - Si una muestra nunca intersecta una fuente de luz, entonces no contribuye al color final del pixel
- Al trazar todas las muestras de un pixel estas se promedian y se obtiene el color final del pixel

Path Tracing - Definición



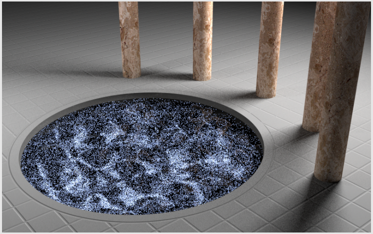
Path Tracing - Problemas

- *Path Tracing* presenta problemas para calcular ciertos caminos de la luz:
  - Cáusticas
  - Medios participantes
  - Luz a través de aperturas pequeñas
- Estos casos suelen tener mucho ruido en las imágenes generadas con *Path Tracing*
  - Es poco probable que caminos que producen estos efectos lleguen a las fuentes de luz

Path Tracing - Ejemplos

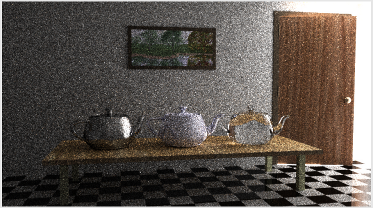
Ambos ejemplos fueron generados con *Path Tracing* bidireccional.

210 muestras por pixel



2 horas y media.

40 muestras por pixel



4 horas.

Path Tracing - Mejoras

- En 1997 Eric Veach y Leonidas Guibas proponen Metropolist Light Transport (MLT) como una solución a los problemas de *Path Tracing*
  - El nombre de la técnica se debe al Algoritmo de Metropolis
- Su objetivo es generar imágenes correctas en escenas "difíciles"sin incrementar el tiempo de procesamiento

Algoritmo de Metropolis - Definición

- Propuesto en 1953 por Nicolas Metropolis, Arianna Rosenbluth, Marshal Rosenbluth, Augusta Teller y Edward Teller
- Se utiliza para generar recorridos aleatorios dentro de un espacio de estados Ω
  - Un recorrido es una secuencia $\overline{X_0}, \overline{X_1}, ..., \overline{X_k}$, donde todo $\overline{X_i} \in \Omega$
  - Todo recorrido parte de un estado inicial $\overline{X_0} \in \Omega$
  - El recorrido generado se distribuye proporcionalmente a una función $f : \Omega \to \mathbb{R}^+$
  - Cada etapa $\overline{X_i}$ del recorrido se genera aplicando una modificación aleatoria a la etapa $\overline{X_{i-1}}$
    - En otras palabras, los recorridos generados son cadenas de Markov
  - Otra forma de verlo es que $\overline{X_0}, \overline{X_1}, ..., \overline{X_k}$ son muestras de f.

Algoritmo de Metropolis - Generalidades

En lineas generales el algoritmo de metropolis sigue estos pasos para generar aproximaciones de una función f:

1. Crear una distribución de muestreo S proporcional a f
2. Crear un histograma de muestras tomadas de S
3. Escalar el histograma para aproximar a f

Para crear un histograma proporcional a f, el algoritmo de Metropolis sigue la condición de balance detallado:

Balance detallado

Supongamos existe un histograma proporcional a f (llamado distribución estacionaria). Supongamos que también existe una función de transición $K(\overline{y}|\overline{x})$ que permite el flujo de muestras entre dos bins del histograma. Se dice que K cumple la condición de balance detallado si $K(\overline{y}|\overline{x}) = K(\overline{x}|\overline{y})$.

Algoritmo de Metropolis - Factores

Para calcular la función de transición K, se define entonces una estrategia de mutación y una probabilidad de aceptación:

Estrategia de mutación

Sea $T(\overline{y}|\overline{x})$ la estrategia de mutación definida como la probabilidad de escoger el estado $\overline{y}$ como siguiente punto de muestra, dado que el punto de muestra actual es $\overline{x}$.

Probabilidad de aceptación

Dada una estrategia de mutación T, el paso de una muestra de $\overline{x}$ a $\overline{y}$ será aceptado con una probabilidad a, donde:

$$a(\overline{y}|\overline{x}) = min\left(1, \frac{f(\overline{y} \times T(\overline{x}|\overline{y})}{f(\overline{x} \times T(\overline{y}|\overline{x})}\right) \tag{1}$$

Factor de escalamiento

Se obtiene dividiendo el valor promedio de f (f_{ave}, promedio de muchas muestras tomadas de S), entre el número promedio de muestras por *bin* en el histograma (h_{ave}):

$$s = f_{ave}/h_{ave} \tag{2}$$

Algoritmo de Metropolis - Pseudocódigo

Entradas:

$\overline{X_0}$ Estado inicial

f Función a muestrear

M Número de mutaciones

Salidas:

H Histograma

Algoritmo

- Sea $s \leftarrow f(\overline{X_0})$
- Para m en M hacer
  - Sea $s' \leftarrow f(\overline{X_m})$
  - Sea $a \leftarrow min\left((1, \frac{s' \times T(\overline{X_0}|\overline{X_m})}{s \times T(\overline{X_m}|\overline{X_0})}\right)$
  - Si uniforme(0, 1) $<$ a entonces
    - Sea $s \leftarrow s'$
    - Sea $\overline{X_0} \leftarrow \overline{X_m}$
  - Sea $H[\overline{X_0}] \leftarrow H[\overline{X_0}] + 1$

Algoritmo de Metropolis - Ejemplo

Aplicación en imágenes

f se define en algún subconjunto de $\mathbb{R}^2$ y el histograma posee un *bin* por cada pixel.

Aplicación en Path Tracing

f solo puede evaluarse estadísticamente. Cada muestra de f corresponde a un *path*. El histograma posee un *bin* por cada pixel de la imagen de salida.

Ejemplo

Entradas:

$\overline{X}$ Un *path*

M Número de mutaciones

Salidas:

H Histograma

Algoritmo

- Sea $s \leftarrow evaluar(\overline{X})$
- Para m en M hacer
  - Sea $Y \leftarrow mutar(T, \overline{X})$
  - Sea $s' \leftarrow evaluar(\overline{Y})$
  - Sea $a \leftarrow min\left(1, \frac{luminancia(s') \times T(\overline{X}|\overline{Y})}{luminancia(s) \times T(\overline{Y}|\overline{X})}\right)$
  - Si uniforme(0, 1) $<$ a entonces
    - Sea $s \leftarrow s'$
    - Sea $\overline{X} \leftarrow \overline{Y}$
  - Sea $H[\overline{X}] \leftarrow H[\overline{X}] + (s/luminancia(s))$

MLT depende principalmente de como se seleccionen las estrategias de mutación. La más sencilla es la mutación de nuevo camino.

Mutación de nuevo camino

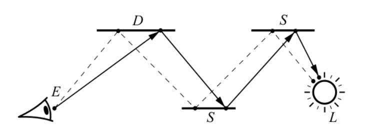
Se crea un nuevo *path* desde un pixel aleatorio de la imagen final.

Perturbación de lente

1. Se perturba la ubicación del pixel del *path* original en una cantidad aleatoria en una dirección aleatoria
2. Comenzando desde el ojo, el nuevo *path* se propaga por el mimso número de saltos especulares que el *path* original.
3. El primer vértice no especular del *path* nuevo se conecta directamente con el siguiente vértice del *path* original.

Perturbación multi-cadena

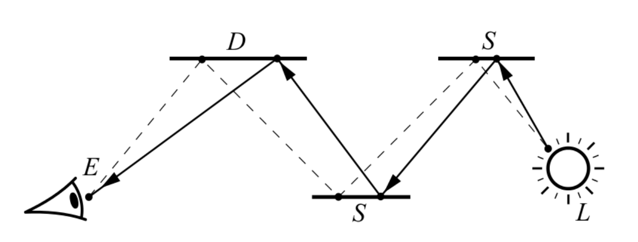
1. Pasos 1 y 2 son equivalentes a los de la perturbación de lente.
2. Cuando se alcanza un vértice no especular, su dirección de salida es perturbada ligeramente.
3. Si el siguiente vértice alcanzado es especular, se propaga el nuevo *path* por la misma cantidad de vértices especulares después de este en el *path* original.
4. Si el siguiente vértice alcanzado es no especular, entonces el *path* nuevo se conecta directamente con el *path* original.



Metropolis Light Transport - Mutaciones de la luz

Perturbación caústica

Es equivalente a una perturbación de lente, pero en lugar de modificar el punto de inicio del camino desde el ojo, se modifica su punto de llegada desde la fuente de luz.



Metropolis Light Transport - Finalizacióm

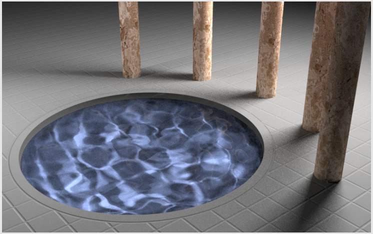
- La evaluación de los *paths* calculados y sus mutaciones se acumulan en un histograma
- Este histograma debe ser escalado para obtener la imagen final:
  - El factor de escalamiento se calcula promediando la luminancia de una gran cantidad de *paths* (f_{ave})
  - Se utiliza la ecuación anterior

$$s = f_{ave}/h_{ave} \tag{3}$$

Metropolis Light Transport - Ejemplos

Los tiempos de *rendering* son similares a los equivalentes con *Path Tracing*

100 mutaciones por camino



250 mutaciones por camino

Conclusiones

- MLT es una técnica que produce muy buenos resultados
- Esta técnica es relativamente fácil de implementar e incorporar en un *Path Tracer*
- Su dificultad consiste en entender como funciona

Referencias

1. E. Veach y L. Guibas, **Metropolis Light Transport**, Computer Graphics, 31(Annual Conference Series):65–76, 1997.
2. D. Cline y P. Egbert, **A Practical Introduction to Metropolis Light Transport**, Technical Report: Department of Computer Science, Brigham Young University, 2005.
3. E. Lafortune, **Mathematical Models and Monte Carlo Algorithms for Physically Based Rendering**, PhD. Dissertation: Department of Computer Science, Katholieke Universiteit Leuven, 1996.

¿Preguntas?