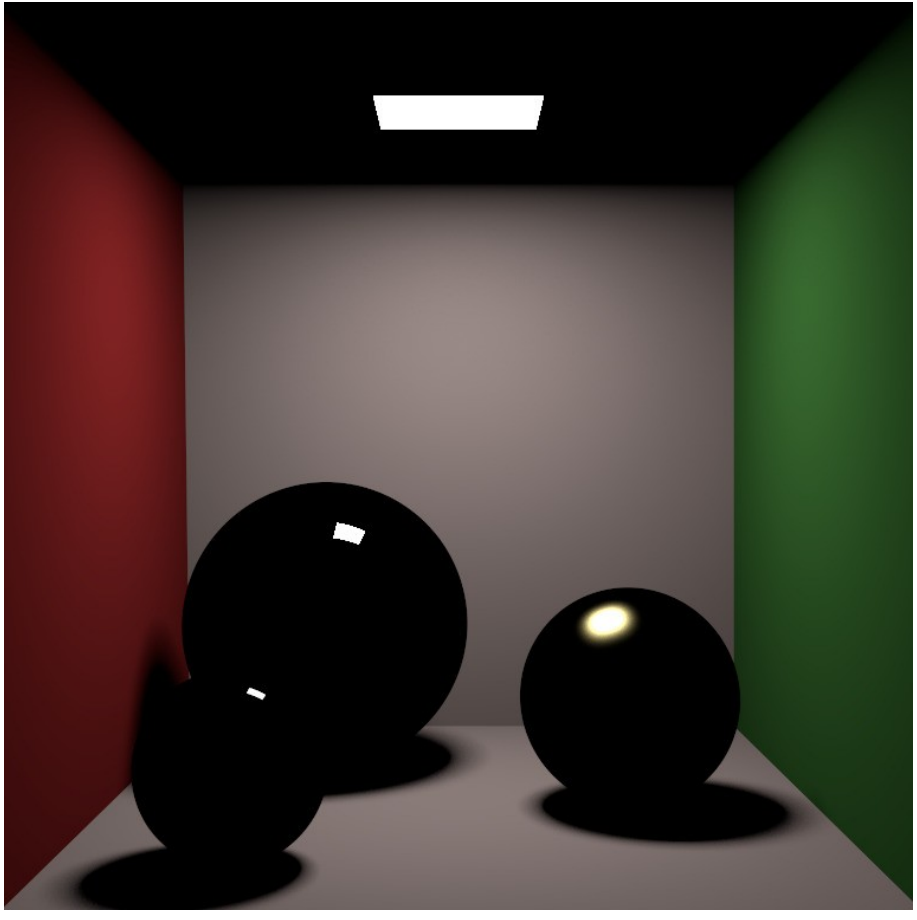
Tema 6

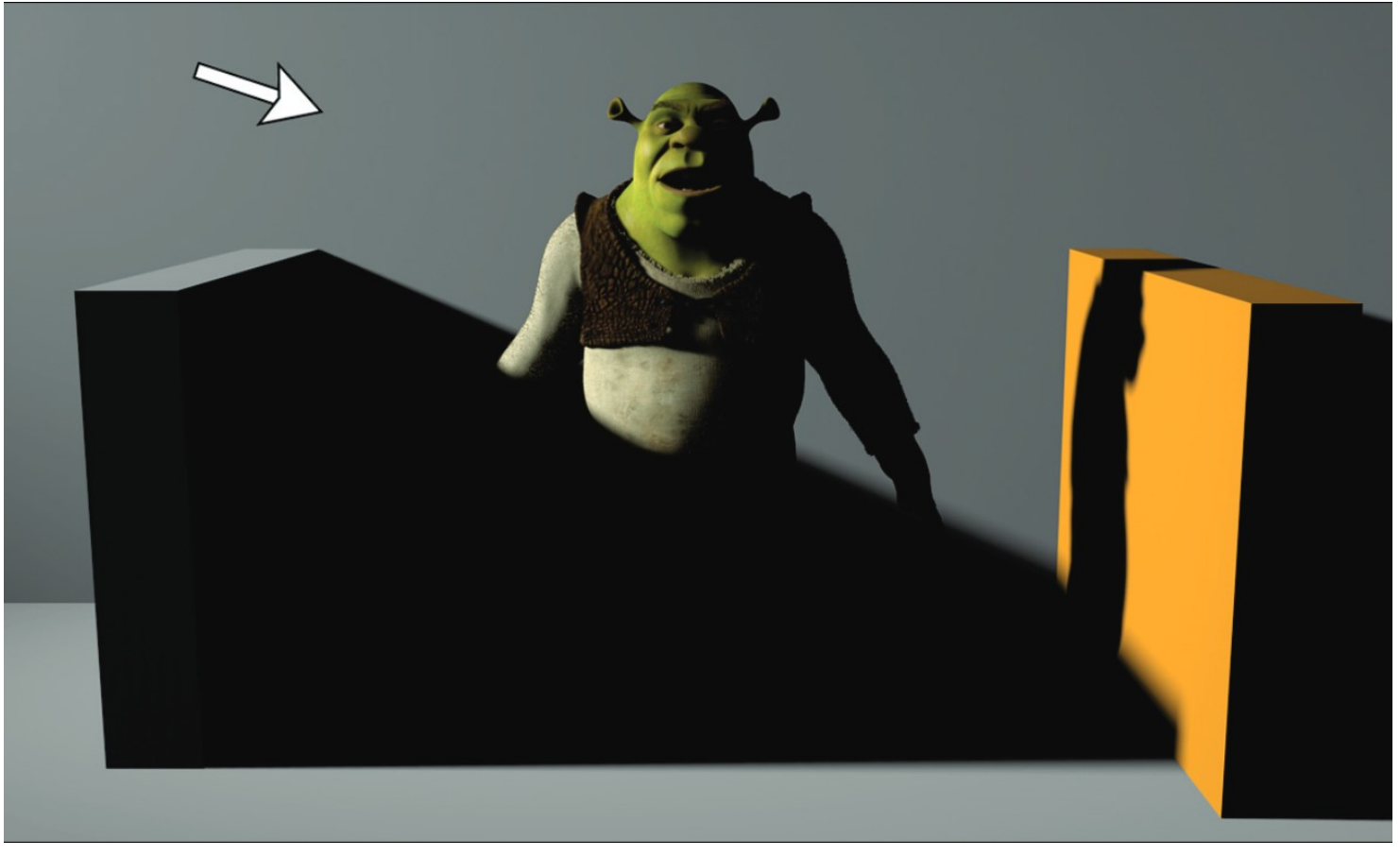
Iluminación global

Iluminación directa vs global



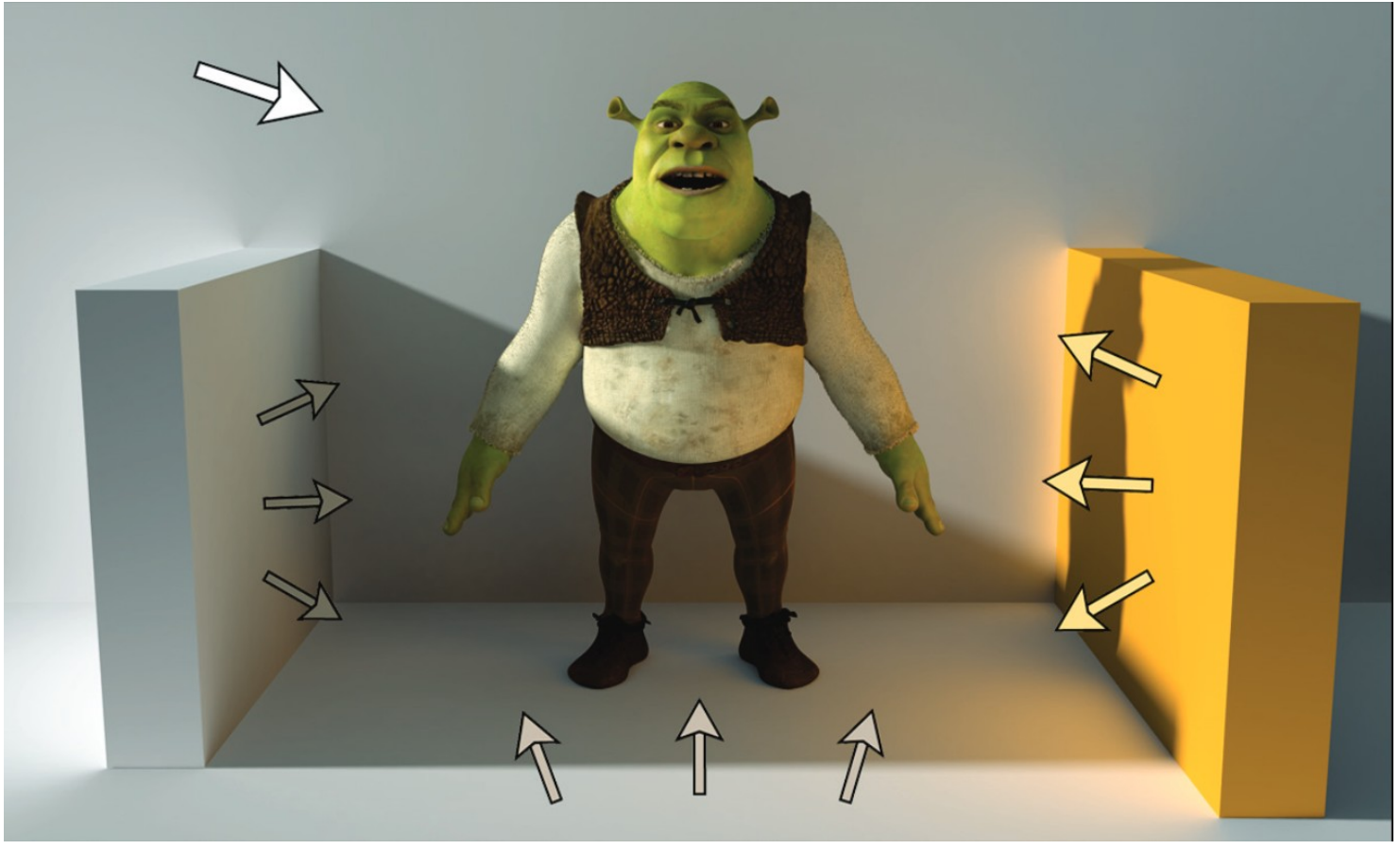
Iluminación directa vs global

Iluminación directa



Iluminación directa vs global

Iluminación global

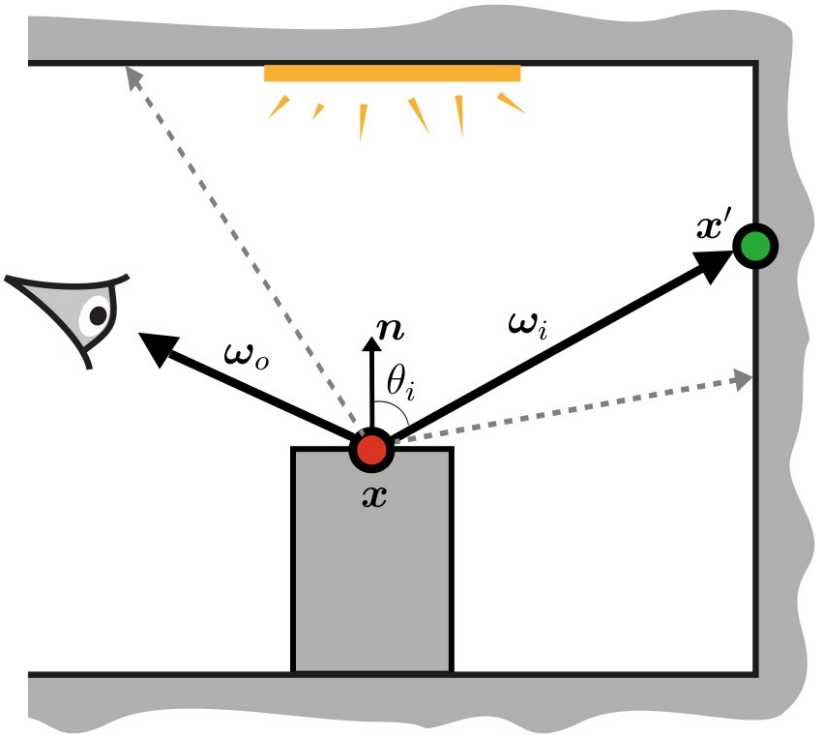


Intuición: caminos de luz

- Considera una fuente luminosa situada en algún lugar de una escena
- Se emite un fotón con alguna dirección (rayo)
- El fotón interactúa con los objetos de una escena
  - La primera intersección es iluminación directa
  - Pero el fotón tendrá posibles “rebotes” subsecuentes
- El fotón llega finalmente al sensor

La ecuación de renderizado

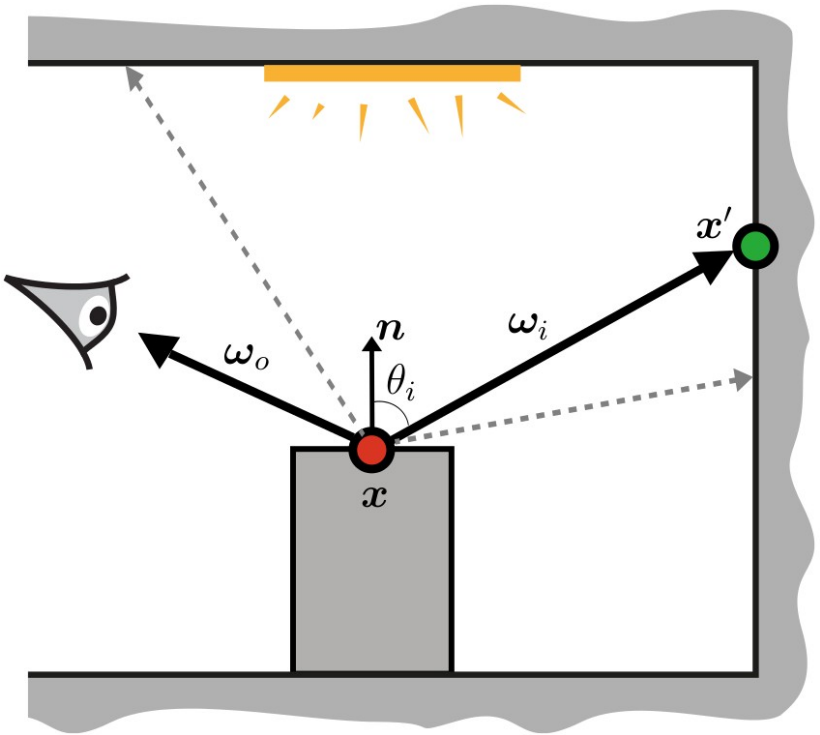
- *The rendering equation*:
  - Introducida por Kajiya en 1986
- También conocida como *Light Transport Equation (LTE)*



$$L(\mathrm{x}, \omega_\mathbf{o}) = L_e(\mathrm{x}, \omega_\mathbf{o}) + \int_{S^2} L(\mathrm{x}', -\omega_\mathbf{i}) f_s(\mathrm{x}, \omega_\mathbf{i}, \omega_\mathbf{o}) |\cos\theta_i| \mathrm{d}\omega_\mathbf{i}$$

La ecuación de renderizado

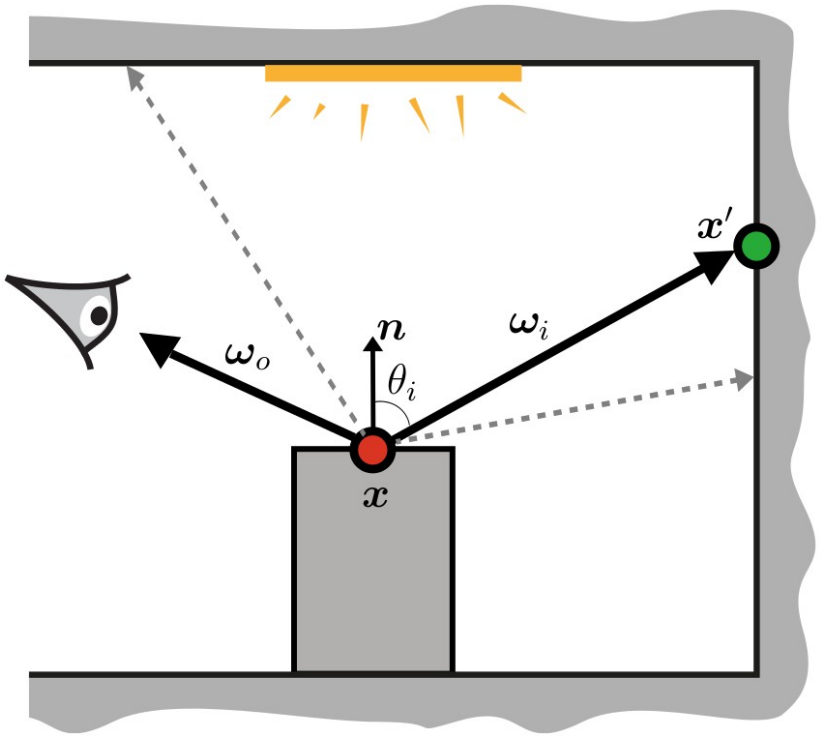
- Expresa la radiancia saliente desde un punto x en la escena
  - Hacia un sensor situado en dirección $\omega_\mathbf{o}$



$$L(\mathrm{x}, \omega_\mathbf{o}) = L_e(\mathrm{x}, \omega_\mathbf{o}) + \int_{S^2} L(\mathrm{x}', -\omega_\mathbf{i}) f_s(\mathrm{x}, \omega_\mathbf{i}, \omega_\mathbf{o}) |\cos\theta_i| \mathrm{d}\omega_\mathbf{i}$$

La ecuación de renderizado

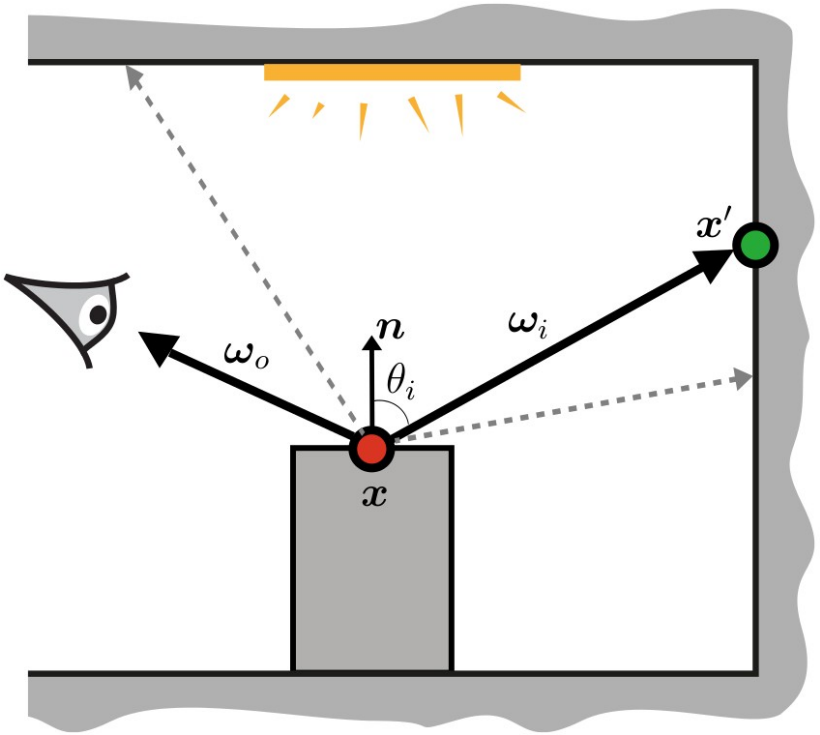
- Es la suma de dos términos:
  - El primero es la luz emitida desde x en la dirección saliente



$$L(\mathrm{x}, \omega_\mathbf{o}) = L_e(\mathrm{x}, \omega_\mathbf{o}) + \int_{S^2} L(\mathrm{x}', -\omega_\mathbf{i}) f_s(\mathrm{x}, \omega_\mathbf{i}, \omega_\mathbf{o}) |\cos\theta_i| \mathrm{d}\omega_\mathbf{i}$$

La ecuación de renderizado

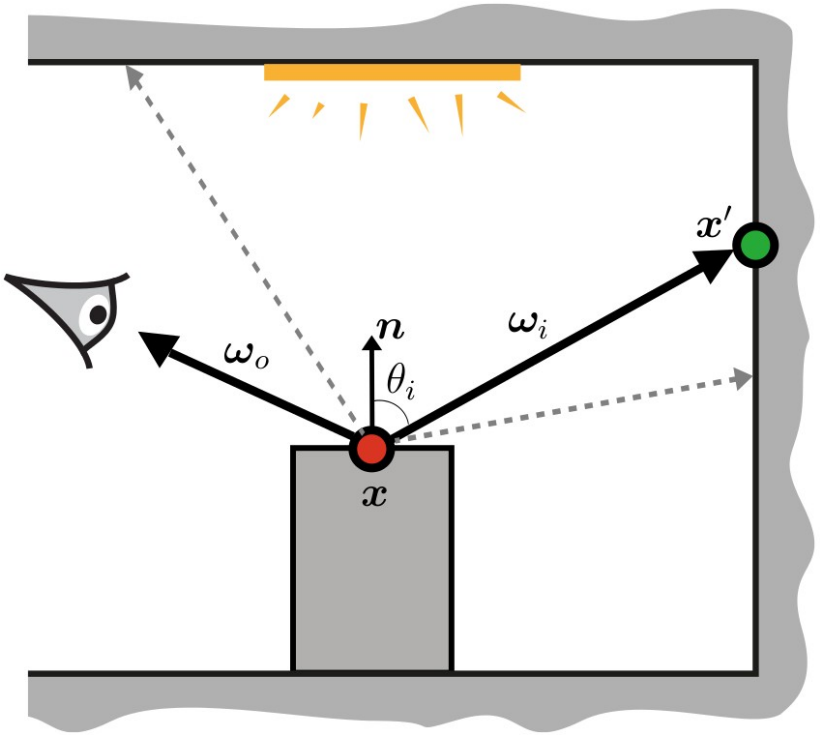
- Es la suma de dos términos:
  - El primero es la luz emitida desde x en la dirección saliente
  - El segundo es la luz total reflejada/ refractada en el punto x hacia la dirección saliente
    - Integrar sobre *todas* las direcciones incidentes



$$L(\mathrm{x}, \omega_\mathbf{o}) = L_e(\mathrm{x}, \omega_\mathbf{o}) + \int_{S^2} L(\mathrm{x}', -\omega_\mathbf{i}) f_s(\mathrm{x}, \omega_\mathbf{i}, \omega_\mathbf{o}) |\cos\theta_i| \mathrm{d}\omega_\mathbf{i}$$

La ecuación de renderizado

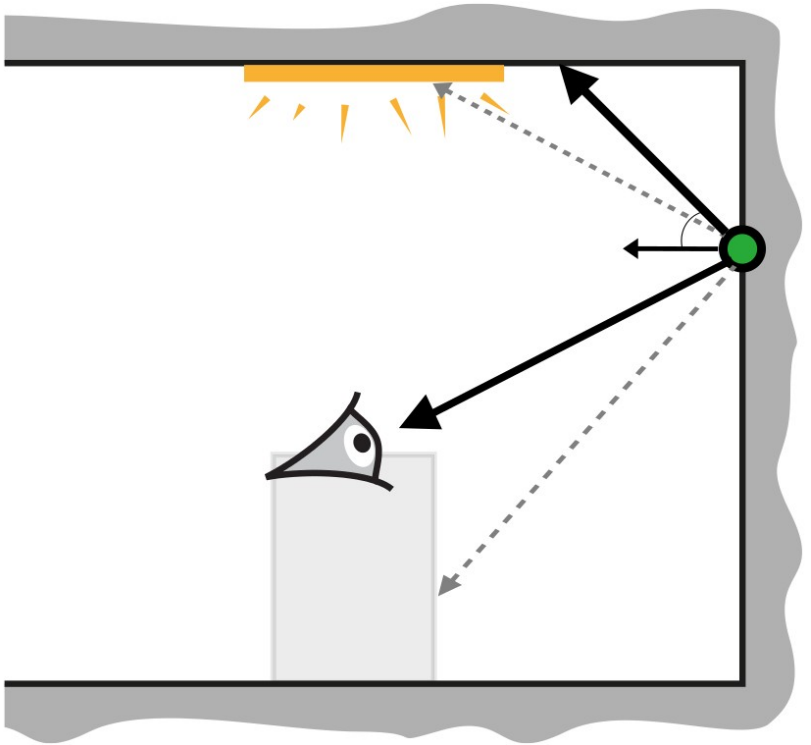
- Esta integral considera la radiancia *entrante* desde *todas* las direcciones incidentes
- Y noten que radiancia entrante es simplemente radiancia saliente desde otro punto x´ en la escena



$$L(\mathrm{x}, \omega_\mathbf{o}) = L_e(\mathrm{x}, \omega_\mathbf{o}) + \int_{S^2} L(\mathrm{x}', -\omega_\mathbf{i}) f_s(\mathrm{x}, \omega_\mathbf{i}, \omega_\mathbf{o}) |\cos\theta_i| \mathrm{d}\omega_\mathbf{i}$$

La ecuación de renderizado

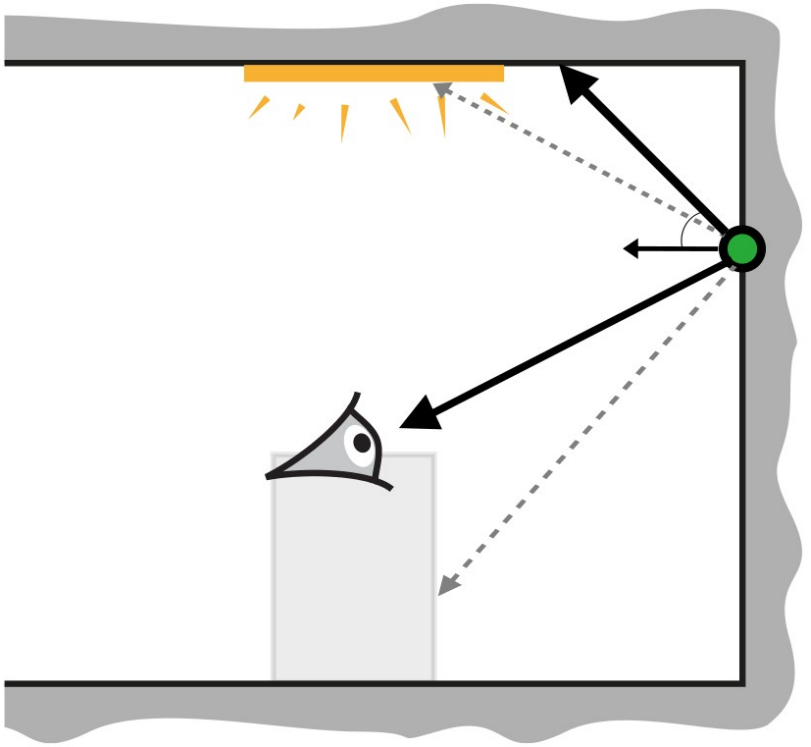
- Y regresamos al problema inicial pero ahora para otro punto en la escena…
- Y resolver para este otro punto, involucra resolver de nuevo L



$$L(\mathrm{x}, \omega_\mathbf{o}) = L_e(\mathrm{x}, \omega_\mathbf{o}) + \int_{S^2} L(\mathrm{x}', -\omega_\mathbf{i}) f_s(\mathrm{x}, \omega_\mathbf{i}, \omega_\mathbf{o}) |\cos\theta_i| \mathrm{d}\omega_\mathbf{i}$$

La ecuación de renderizado

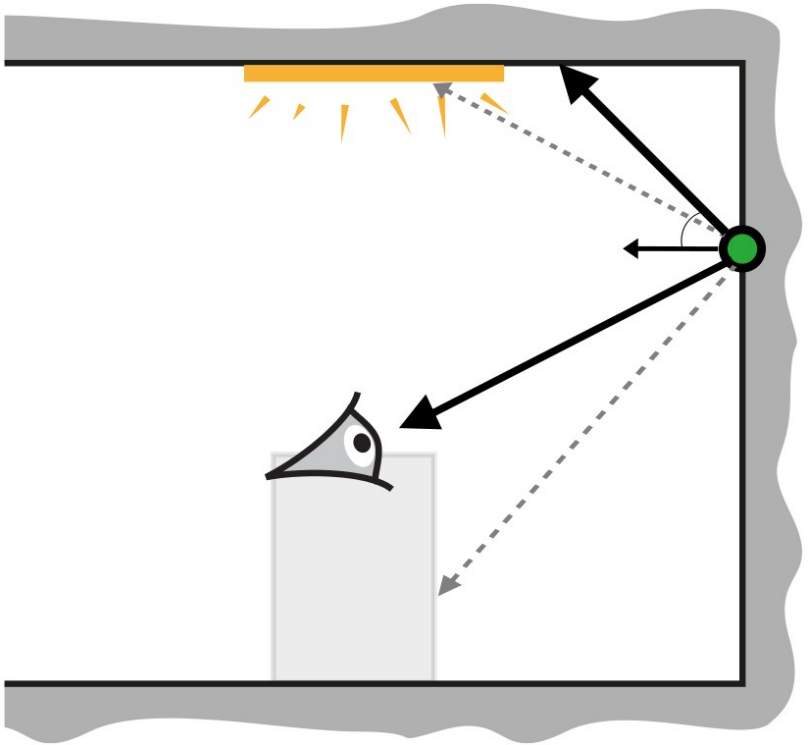
- Esta es una ecuación Fredholm del segundo tipo
- Espacio de caminos de luz infinito-dimensional
- No existe una solución analítica
  - Obligatorio recurrir a métodos numéricos



$$L(\mathrm{x}, \omega_\mathbf{o}) = L_e(\mathrm{x}, \omega_\mathbf{o}) + \int_{S^2} L(\mathrm{x}', -\omega_\mathbf{i}) f_s(\mathrm{x}, \omega_\mathbf{i}, \omega_\mathbf{o}) |\cos\theta_i| \mathrm{d}\omega_\mathbf{i}$$

La ecuación de renderizado

- Múltiples métodos numéricos:
  - Radiosity
  - Photon mapping
  - Ray tracing
  - Path tracing
  - Bidirectional path tracing
  - Metropolis LT



$$L(\mathrm{x}, \omega_\mathbf{o}) = L_e(\mathrm{x}, \omega_\mathbf{o}) + \int_{S^2} L(\mathrm{x}', -\omega_\mathbf{i}) f_s(\mathrm{x}, \omega_\mathbf{i}, \omega_\mathbf{o}) |\cos\theta_i| \mathrm{d}\omega_\mathbf{i}$$

Monte Carlo Path Tracing

- Un estimador Monte Carlo está definido como:

$$F = \int_X f(x)\mathrm{d}x \approx \frac{1}{N}\sum_{j=0}^{N-1}\frac{f(x_{(j)})}{p(x_{(j)})}$$

- Entonces para la LTE el estimador sería:

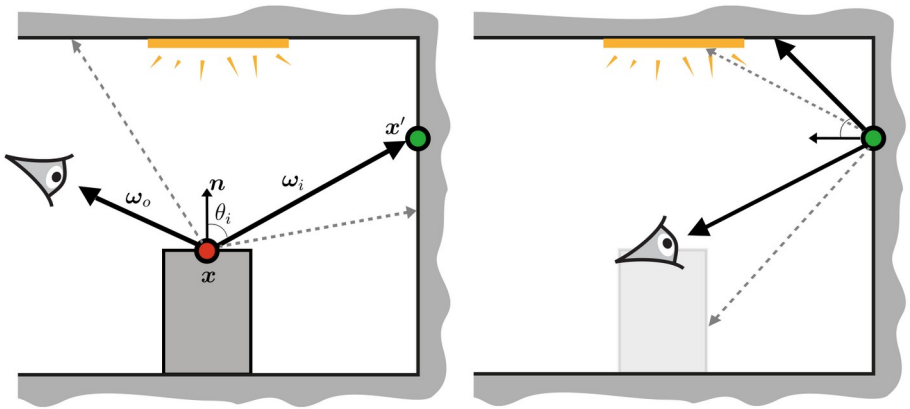
$$L(\mathrm{x},\omega_\mathbf{o}) \approx L_e(\mathrm{x},\omega_\mathbf{o}) + \frac{1}{N}\sum_{j=0}^{N-1}\frac{L(\mathrm{x}',-\omega_{\mathbf{i}(j)})f_s(\mathrm{x},\omega_{\mathbf{i}(j)},\omega_\mathbf{o})|\cos\theta_i|}{p(\omega_{\mathbf{i}(j)})}$$

Monte Carlo Path Tracing

- Caminos de luz
- Una muestra de nuestro estimador Monte Carlo para la LTE consiste en trazar un camino de rayos

$$
\begin{aligned}
L^0 &= L_e^1 + f_s^1 L^1 \\
&= L_e^1 + f_s^1 (L_e^2 + f_s^2 L^2) \\
&= L_e^1 + f_s^1 (L_e^2 + f_s^2 (L_e^3 + f_s^3 L^3)) \\
&= \dots
\end{aligned}
$$

Nota: por simplicidad en notación se asume que fs incluye el coseno y la probabilidad



- ¿recursivo?

Iluminación Global vs Iluminación Directa

- La ecuación para iluminación directa es:

$$L(\mathrm{x}, \omega_\mathbf{o}) = L_e(\mathrm{x}, \omega_\mathbf{o}) + \int_\Omega L_e(\mathrm{x}', -\omega_\mathbf{i}) f_r(\mathrm{x}, \omega_\mathbf{i}, \omega_\mathbf{o}) \cos\theta_\mathbf{i} \mathrm{d}\omega_\mathbf{i}$$

- Mientras que la LTE:

$$L(\mathrm{x}, \omega_\mathbf{o}) = L_e(\mathrm{x}, \omega_\mathbf{o}) + \int_{S^2} L(\mathrm{x}', -\omega_\mathbf{i}) f_s(\mathrm{x}, \omega_\mathbf{i}, \omega_\mathbf{o}) |\cos\theta_\mathbf{i}| \mathrm{d}\omega_\mathbf{i}$$

Path Tracing recursivo

- Con modificaciones mínimas, es posible convertir un integrador de iluminación directa a path tracing recursivo:
  - No muestrear la fuente de luz y en su lugar usar la BSDF
  - Considerar el valor absoluto del coseno
- Pseudocógido general:

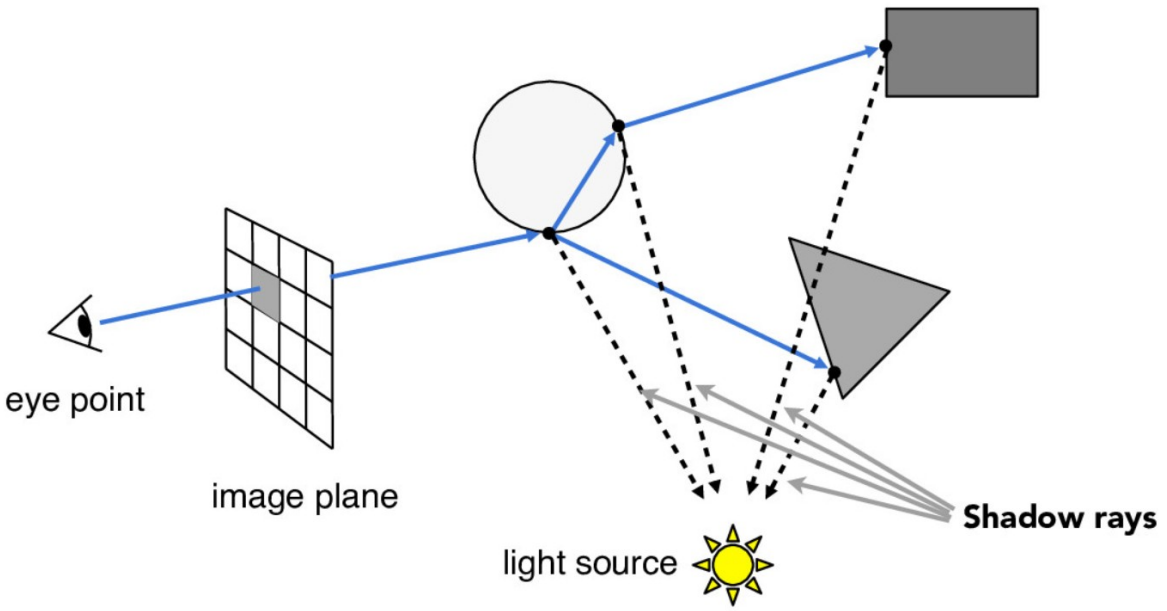
```
shade(ray)
{
        x = intersect(ray);
        n = NormalAt(x);
        wi = SampleBSDF(x);
        cosine = n.dot(wi);
        newray = CreateRay(x,wi);
        value = Le(x,wo);
        value += fs(x,wi,wo) * abs(cosine) * shade(newray) / p(wi);

        return value;
}
```

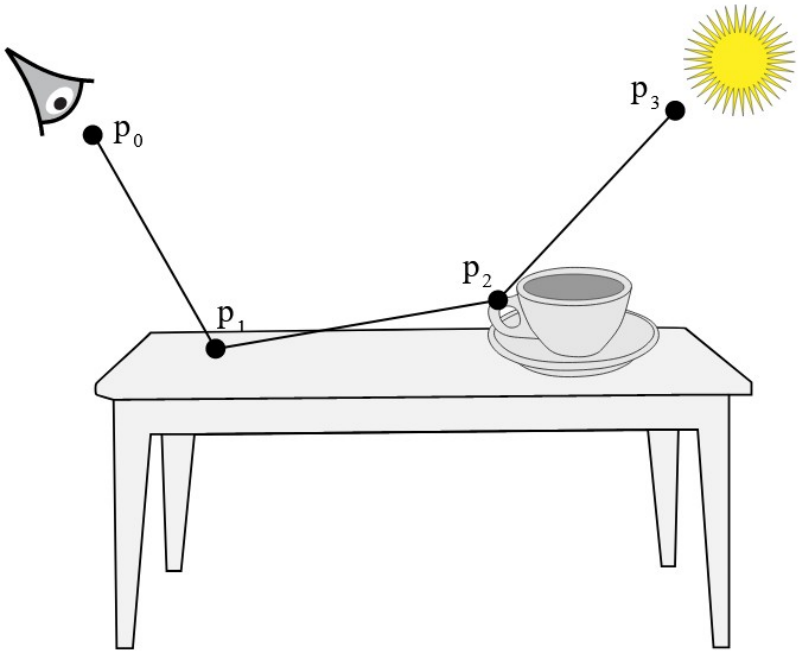
Path Tracing recursivo

- ¿Cuándo terminar la recursión?
  - El rayo escapa de la escena
    - ¿Qué pasa si nunca escapa? (ej. reflexión interna total)
  - El rayo ha perdido toda la energía (ej. bsdf evalúa a 0)
    - ¿Qué pasa si no es así?
  - Al alcanzar cierto número de rebotes
    - Se introduce sesgo (bias) a la solución

Ray Tracing vs Path Tracing



Ray Tracing: múltiples ramificaciones en cada intersección



Path Tracing: sólo un camino

Path Tracing Implícito (con sesgo)

- Si asumimos que un camino de luz desde la cámara *termina* cuando se alcanza una fuente luminosa, entonces

$$
\begin{aligned}
L^0 &= L_e^1 + f_s^1 L^1 \\
&= L_e^1 + f_s^1 (L_e^2 + f_s^2 L^2) \\
&= L_e^1 + f_s^1 (L_e^2 + f_s^2 (L_e^3 + f_s^3 L^3)) \\
&= \dots
\end{aligned}
$$

- Se simplifica a:

$$
\begin{aligned}
L^0 &\approx f_s^1 L^1 \\
&\approx f_s^1 f_s^2 L^2 \\
&\approx f_s^1 f_s^2 f_s^3 L^3 \\
&\approx f_s^1 f_s^2 f_s^3 \dots f_s^n L_e^n
\end{aligned}
$$

Que es un producto de las BSDFs (con coseno y probabilidad) en cada punto de intersección a lo largo del camino hasta la luz
→ *path throughput*

Path Tracing Implícito (con sesgo)

- Pseudocódigo recursivo:

```
shade(ray)
{
     x = intersect(ray);

     if(emitter(x))
          return Le(x);

     n = NormalAt(x);
     wi = SampleBSDF(x);
     cosine = n.dot(wi);
     newray = CreateRay(x,wi);
     value = fs(x,wi,wo) * abs(cosine) * shade(newray) / p(wi);

     return value;
}
```

Path Tracing Implícito (con sesgo)

- Pseudocódigo iterativo:

```
shade(ray)
{
     throughput = 1.0;

     x = intersect(ray);
     while(!emitter(x))
     {
          n = NormalAt(x);
          wi = SampleBSDF(x);
          cosine = n.dot(wi);
          throughput *= fs(x,wi,wo) * abs(cosine) / p(wi);

          newray = CreateRay(x,wi);
          x = intersect(newray);
     }
      return throughput*Le(x);
}
```

Monte Carlo Path Tracing

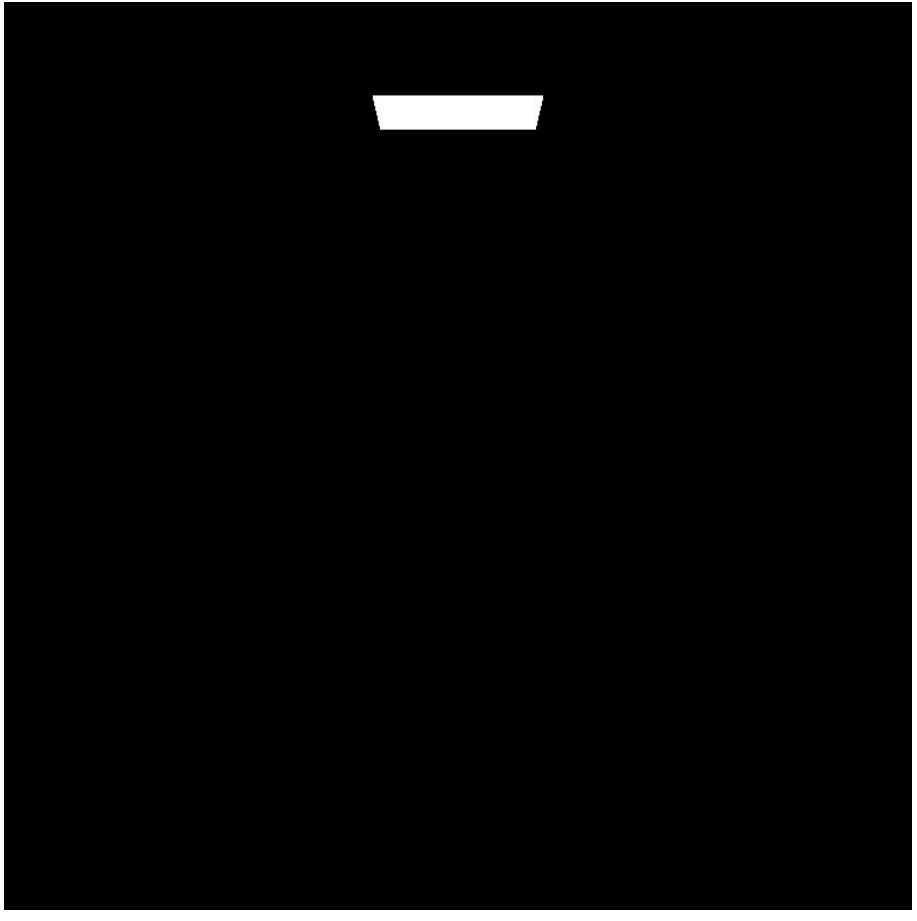
- Si aplicamos las operaciones podemos obtener una expresión que nos lleva a una implementación *iterativa*

$$
\begin{aligned}
L^0 &= L_e^1 + f_s^1 L^1 \\
&= L_e^1 + f_s^1 (L_e^2 + f_s^2 L^2) \\
&= L_e^1 + f_s^1 (L_e^2 + f_s^2 (L_e^3 + f_s^3 L^3)) \\
&= \dots
\end{aligned}
\qquad
\begin{aligned}
L^0 &= L_e^1 + f_s^1 L^1 \\
&= L_e^1 + f_s^1 L_e^2 + f_s^1 f_s^2 L^2 \\
&= L_e^1 + f_s^1 L_e^2 + f_s^1 f_s^2 L_e^3 + f_s^1 f_s^2 f_s^3 L^3 \\
&= \dots
\end{aligned}
$$

- Cada expresión en la suma es la contribución de los caminos según su longitud
  - El render final es el resultado de sumar las contribuciones de todos los caminos

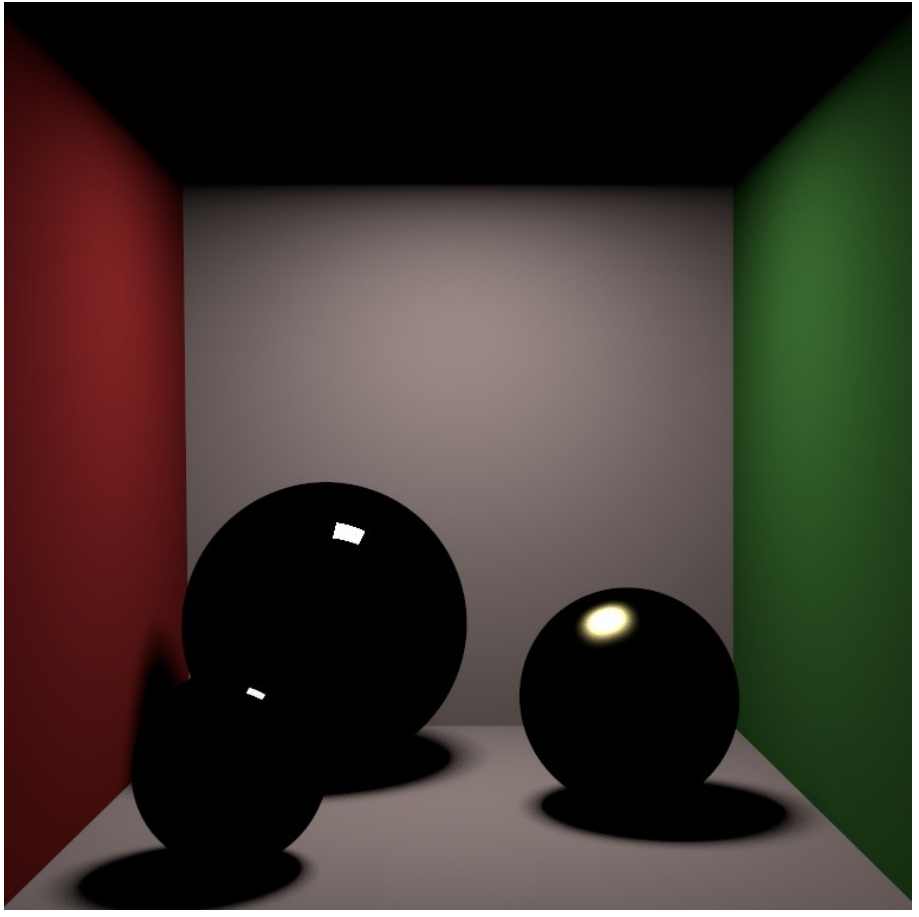
Contribución de un camino

- Longitud 1
  - Sólo las fuentes de luz visibles directamente



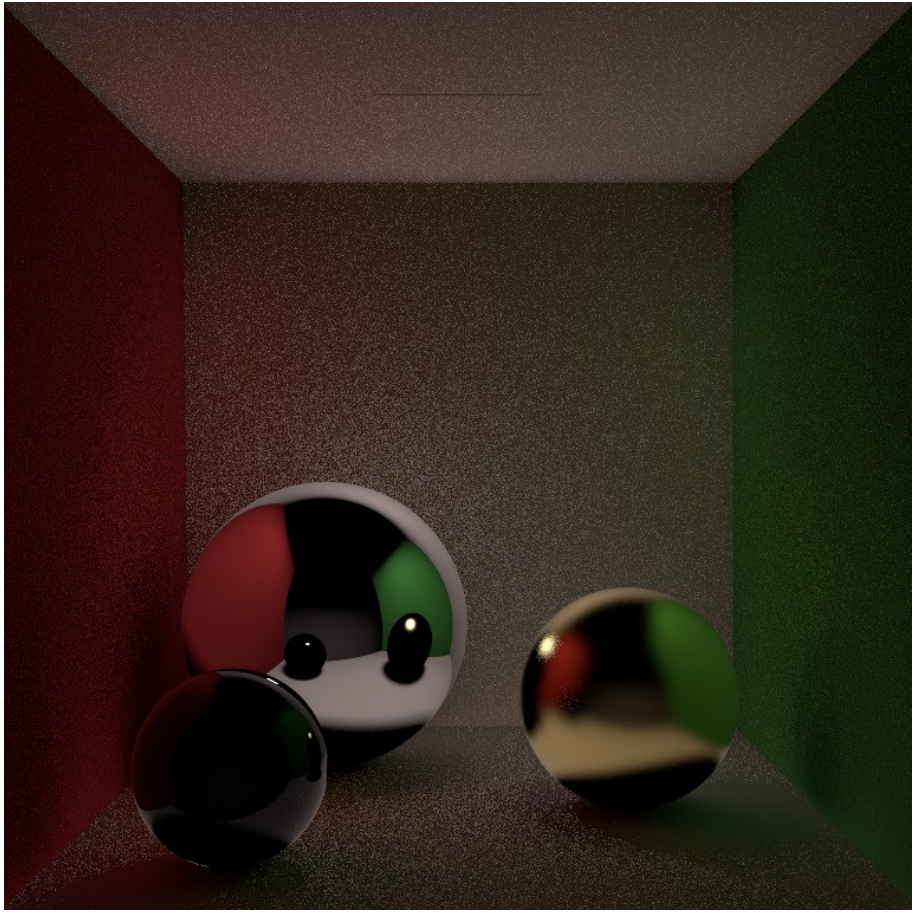
Contribución de un camino

- Longitud 2
  - Iluminación directa



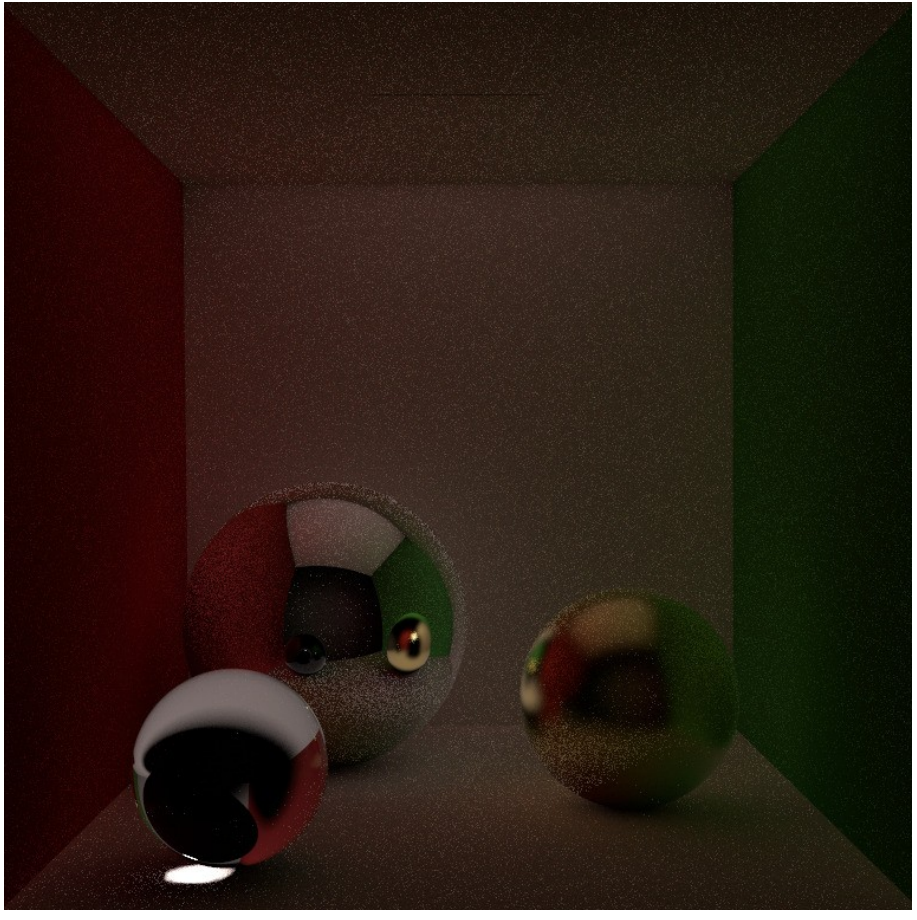
Contribución de un camino

- Longitud 3
  - (primer indirecta)



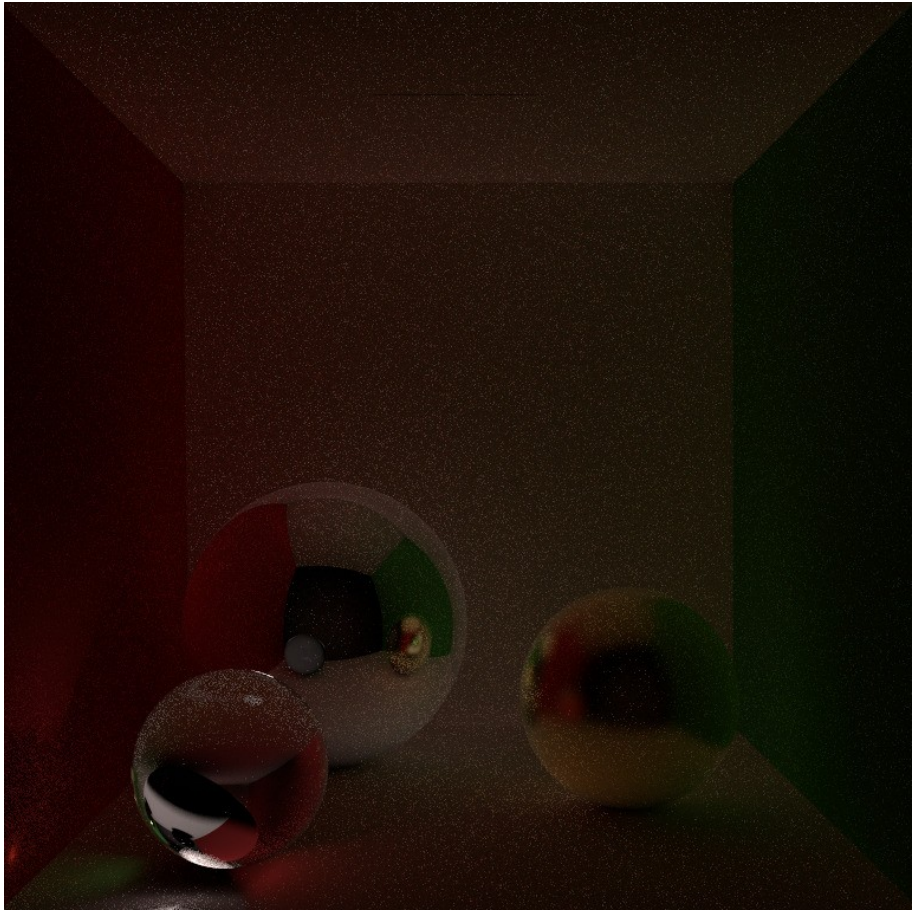
Contribución de un camino

- Longitud 4



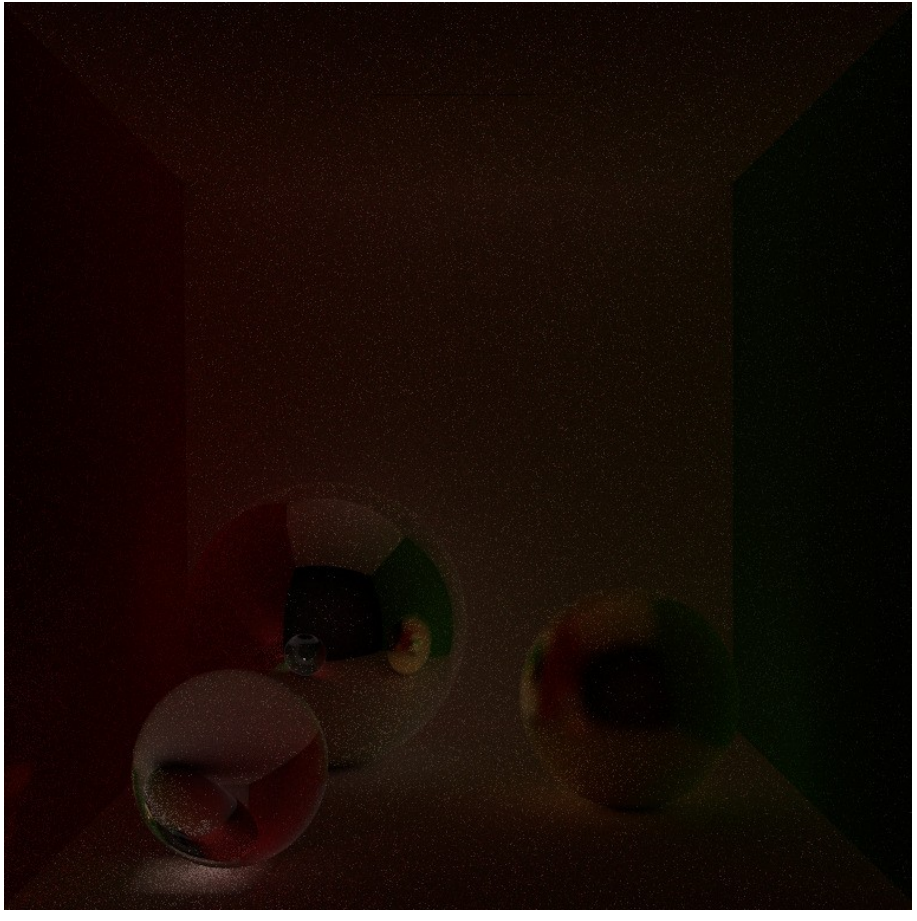
Contribución de un camino

- Longitud 5



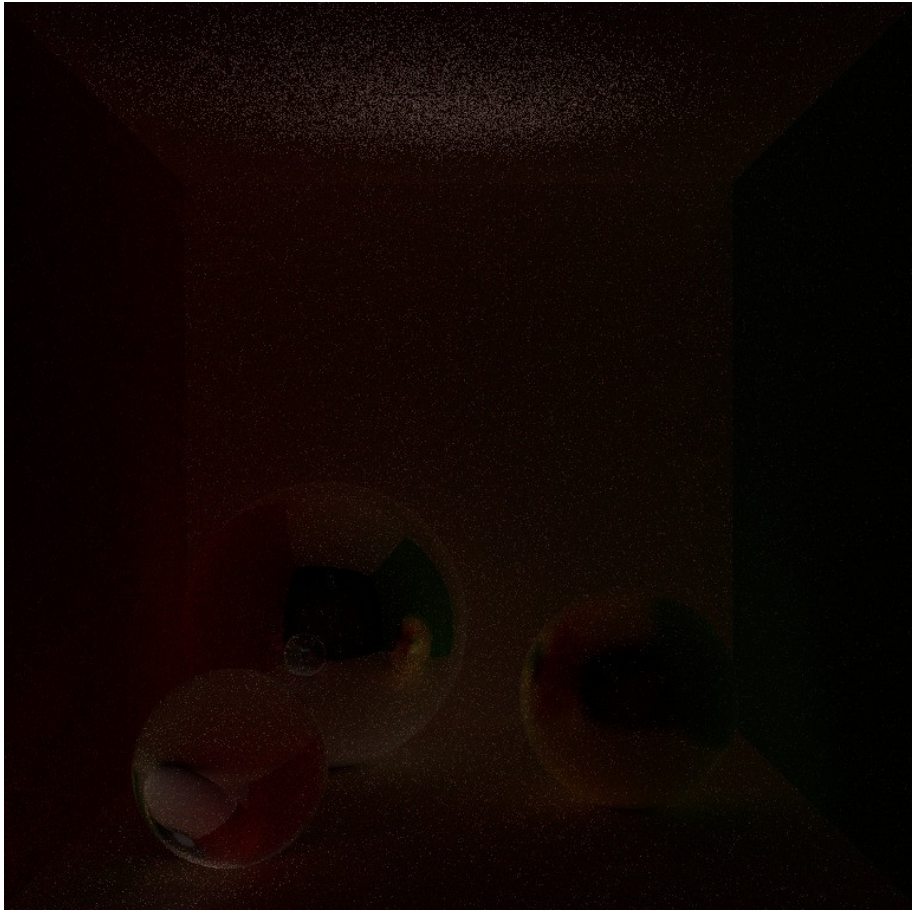
Contribución de un camino

- Longitud 6



Contribución de un camino

- Longitud 7



Contribución de un camino

- Render completo
  - Suma de los caminos de *TODAS* las longitudes

Terminación de caminos: ruleta rusa

- Al generar caminos se pueden presentar las siguientes situaciones:
  - Caminos muy largos con casi nula energía
  - Caminos que no logran escapar alguna zona de la escena
    - Ej. al interior de un objeto y reflexión interna total
  - Caminos que escapan la escena sin alcanzar alguna fuente luminosa
- En todas las situaciones anteriores, se desperdicia la costosa labor del renderizador
  - Hay que terminar los caminos *sin introducir sesgo*

Terminación de caminos: ruleta rusa

- La ruleta rusa permite terminar probabilísticamente un camino *sin introducir sesgo*:
  - Caminos de poca energía tendrán menos probabilidad de continuar
  - Caminos largos pueden ser tratados también con menor probabilidad de continuar
  - ¡El valor esperado del estimador es el correcto!
    - Se compensa determinando la probabilidad de continuar o no

Terminación de caminos: ruleta rusa

- Se define q como la probabilidad de no continuar el camino, entonces
  - Si $\xi < q$, el camino termina y se asume que las contribuciones subsecuentes son 0
  - De otro modo el camino continúa pero el estimado se divide por la probabilidad de continuar, es decir, por 1-q

Path Tracing Implícito (sin sesgo)

- Ruleta rusa con probabilidad de terminación de 0.1:

```
shade(ray)
{
     x = intersect(ray);
     n = NormalAt(x);
     wi = SampleBSDF(x);
     cosine = n.dot(wi);

     value = Le(x);

     q = 0.1;
     continueprob = 1.0 - q;
     if(Random() < q)
          return value; // no continuar el camino (regresa 0 en la integral)

     newray = CreateRay(x,wi);
     value += fs(x,wi,wo) * abs(cosine) * shade(newray) / (p(wi) * continueprob);

     return value;
}
```

Path Tracing Explícito

- La ecuación de renderizado considera el total de la radiancia saliente en un punto:

$$L(\mathrm{x}, \omega_\mathbf{o}) = L_e(\mathrm{x}, \omega_\mathbf{o}) + \int_{S^2} L(\mathrm{x}', -\omega_\mathbf{i}) f_s(\mathrm{x}, \omega_\mathbf{i}, \omega_\mathbf{o}) |\cos\theta_\mathbf{i}| \mathrm{d}\omega_\mathbf{i}$$

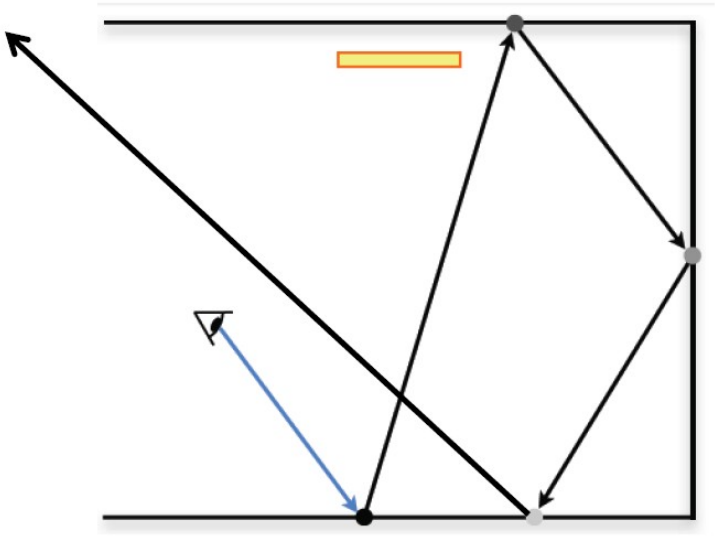
- Pero la integral, en realidad es la suma de la iluminación directa e indirecta:

$$L(\mathrm{x}, \omega_\mathbf{o}) = L_e(\mathrm{x}, \omega_\mathbf{o}) + L_d(\mathrm{x}, \omega_\mathbf{o}) + L_{ind}(\mathrm{x}, \omega_\mathbf{o})$$

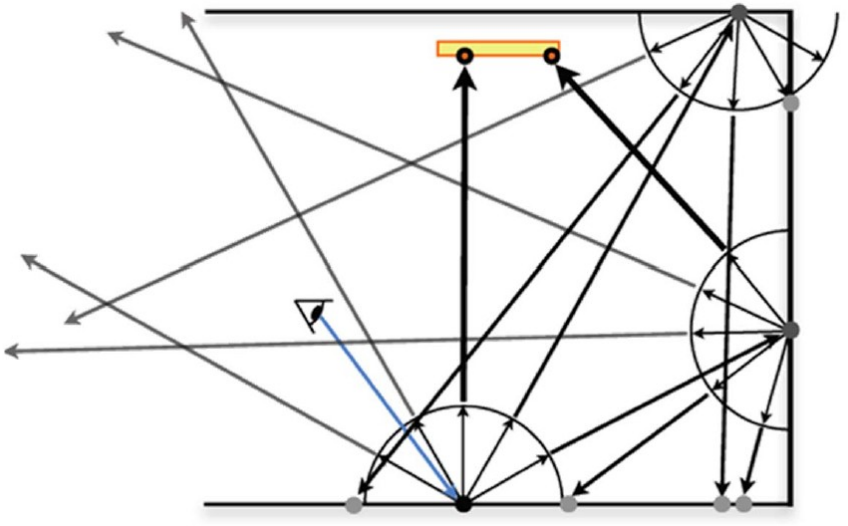
  - ¡¡¡Podemos mejorar la convergencia, utilizando un integrador MIS de iluminación directa!!!

Path Tracing Explícito

- Evita desperdiciar totalmente caminos que no llegan a una fuente luminosa o escapan de la escena



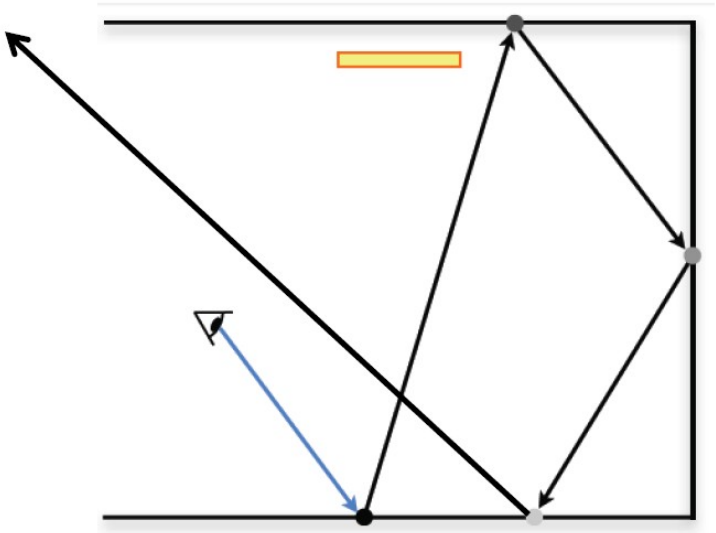
Path Tracing Implícito



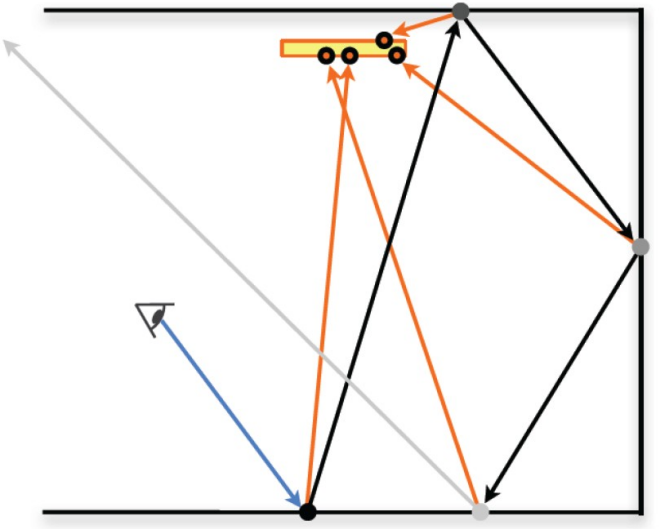
La radiancia saliente es una integral en cada punto del camino
¿por qué no dar el mejor estimado posible en cada punto?

Path Tracing Explícito

- Evita desperdiciar totalmente caminos que no llegan a una fuente luminosa o escapan de la escena
  - Intuición: los puntos del camino reciben iluminación directa



Path Tracing Implícito



Path Tracing Explícito

Path Tracing Explícito: intuición para implementación

- La contribución de un camino en path tracing implícito es:

$$\begin{aligned} L^0 &= L_e^1 + f_s^1 L^1 \\ &= L_e^1 + f_s^1 (L_e^2 + f_s^2 L^2) \\ &= L_e^1 + f_s^1 (L_e^2 + f_s^2 (L_e^3 + f_s^3 L^3)) \\ &= \dots \end{aligned}$$

$$\begin{aligned} L^0 &= L_e^1 + f_s^1 L^1 \\ &= L_e^1 + f_s^1 L_e^2 + f_s^1 f_s^2 L^2 \\ &= L_e^1 + f_s^1 L_e^2 + f_s^1 f_s^2 L_e^3 + f_s^1 f_s^2 f_s^3 L^3 \\ &= \dots \end{aligned}$$

- En Path Tracing Explícito ($L = Le + Ld + Lind$) tenemos:

$$\begin{aligned} L^0 &= L_e^1 + L_d^1 + L_{ind}^1 \\ &= L_e^1 + L_d^1 + f_s^1 (L_d^2 + L_{ind}^2) \\ &= L_e^1 + L_d^1 + f_s^1 (L_d^2 + f_s^2 (L_d^3 + L_{ind}^3)) \end{aligned}$$

Versión recursiva

$$\begin{aligned} L^0 &= L_e^1 + L_d^1 + f_s^1 L_d^2 + f_s^1 L_{ind}^2 \\ &= L_e^1 + L_d^1 + f_s^1 L_d^2 + f_s^1 f_s^2 L_d^3 + f_s^1 f_s^2 L_{ind}^3 \end{aligned}$$

Versión iterativa

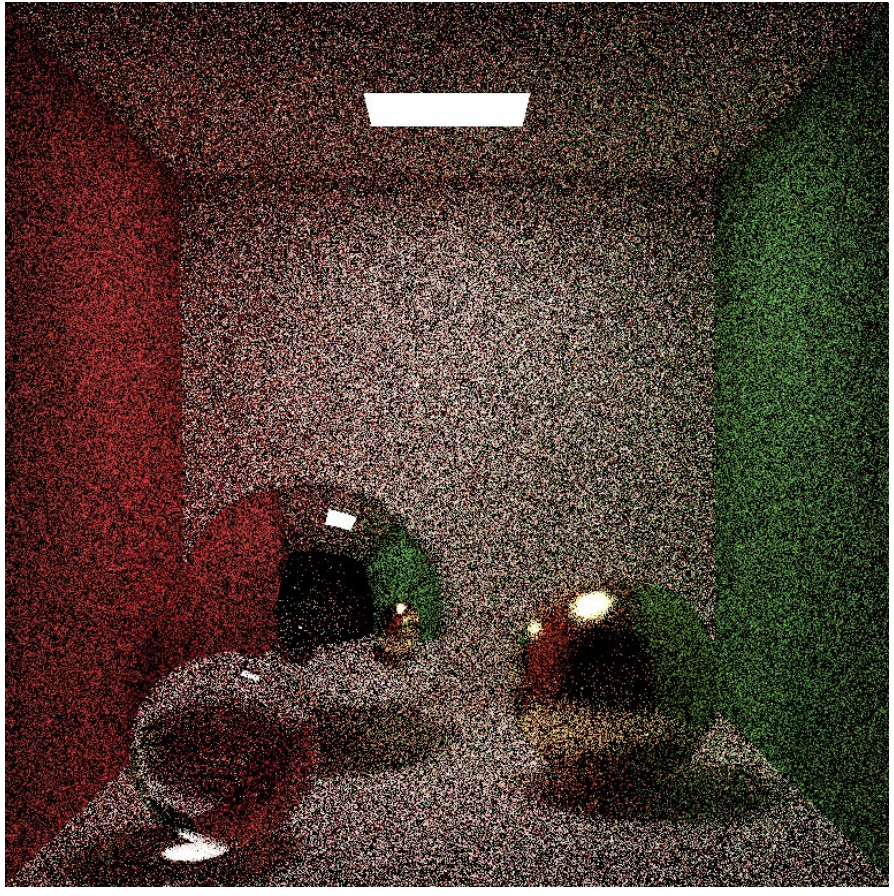
Path Tracing Explícito: Le

- La emisión Le sólo es calculada para aquellas fuentes visibles directamente desde el sensor:

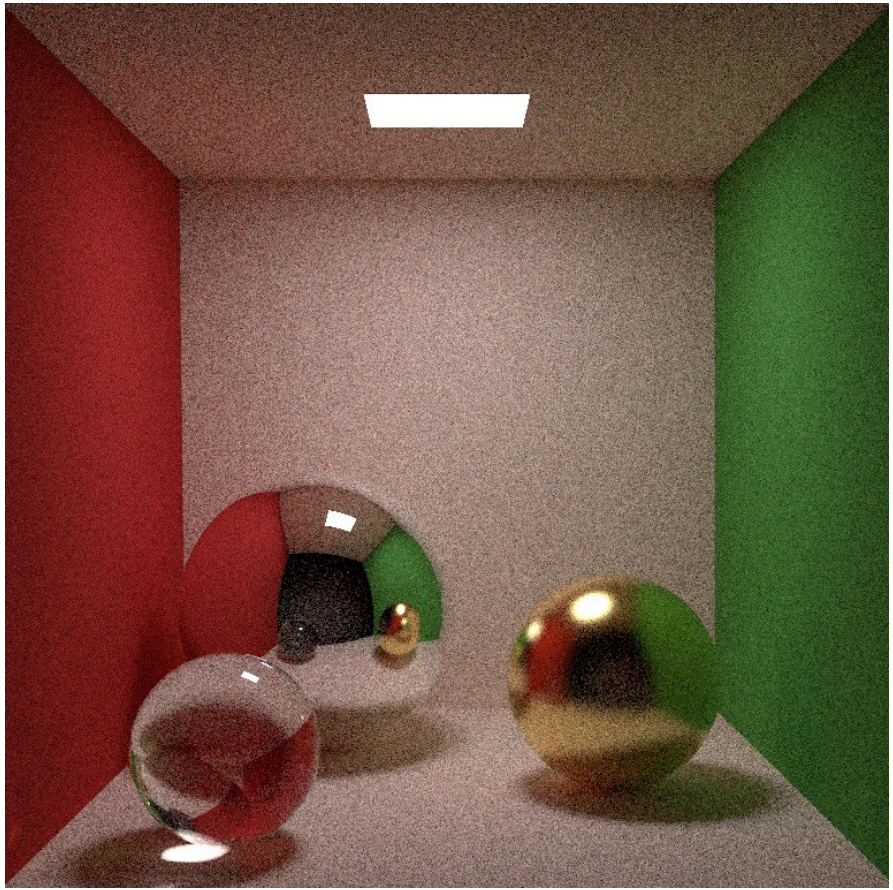
$$
\begin{aligned}
L^0 &= L_e^1 + L_d^1 + L_{ind}^1 \\
&= L_e^1 + L_d^1 + f_s^1(L_d^2 + L_{ind}^2) && = L_e^1 + L_d^1 + f_s^1 L_d^2 + f_s^1 L_{ind}^2 \\
&= L_e^1 + L_d^1 + f_s^1(L_d^2 + f_s^2(L_d^3 + L_{ind}^3)) && = L_e^1 + L_d^1 + f_s^1 L_d^2 + f_s^1 f_s^2 L_d^3 + f_s^1 f_s^2 L_{ind}^3
\end{aligned}
$$

- En puntos subsecuentes del camino ya ha sido considerada como iluminación directa del vértice anterior

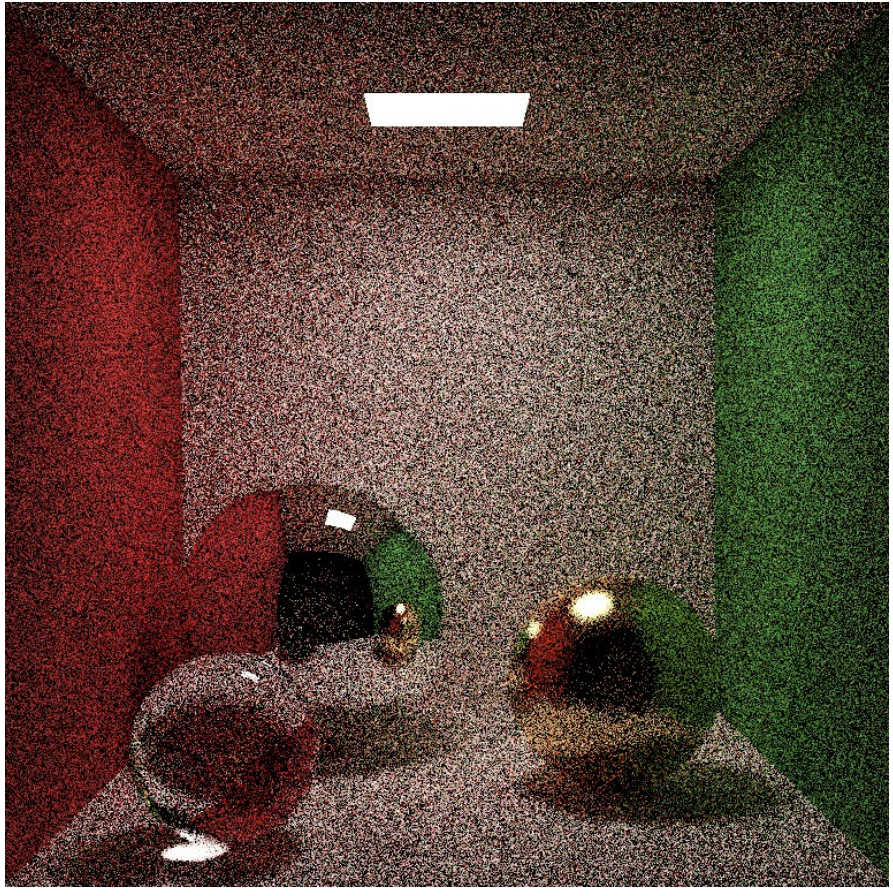
PT Implícito vs Explícito: 32spp



PT Implícito vs Explícito: 512spp



PT Implícito vs Explícito: mismo tiempo (49spp vs 32spp)



PT Implícito vs Explícito: mismo tiempo (780spp vs 512spp)

Path Tracing Explícito: conclusiones

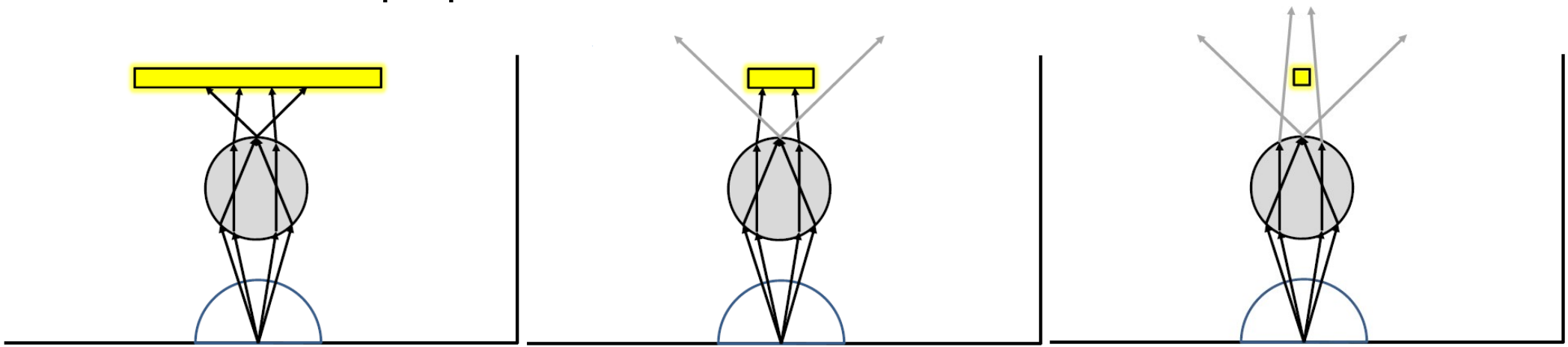
- Reduce significativamente la variancia en el valor estimado
- Cada muestra es más costosa
  - Se calcula la iluminación directa con MIS
- Cada muestra es mucho mejor
  - Aunque sea más costosa, en el mismo tiempo supera a la versión *simple* implícita

Pros y cons de Path Tracing

- Ventajas de path tracing:
  - Sencillo de implementar
  - Converge a la respuesta correcta
- Desventajas de path tracing:
  - En ocasiones hay convergencia lenta
  - Dificultades con caminos difíciles

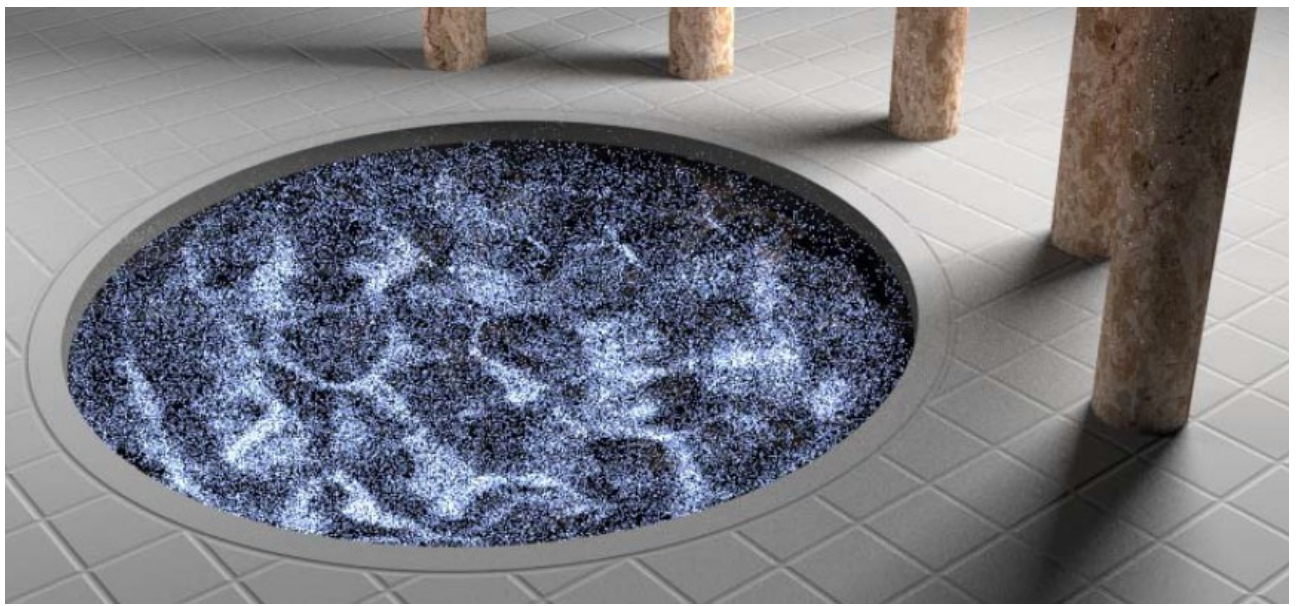
Problemas con Path Tracing

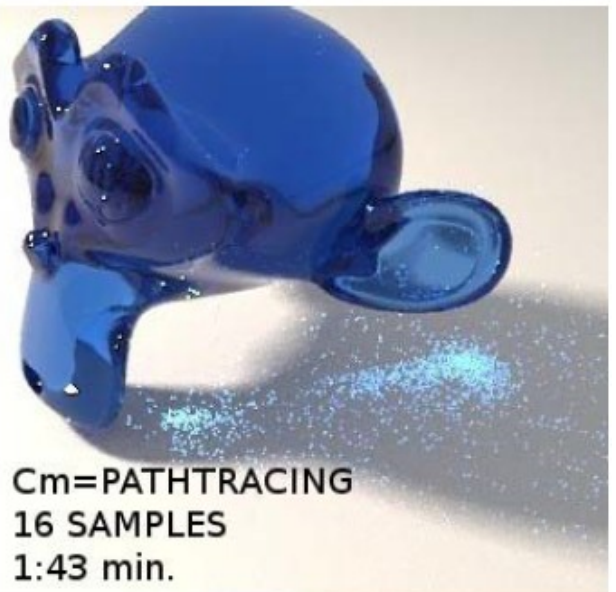
- Los cáusticos son difíciles
  - Fuentes pequeñas de alta emisión



Problemas con Path Tracing

- Los cáusticos son difíciles
  - Geometría compleja





Problemas con Path Tracing

- Los cáusticos son difíciles
  - Cáusticos reflejados

Problemas con Path Tracing

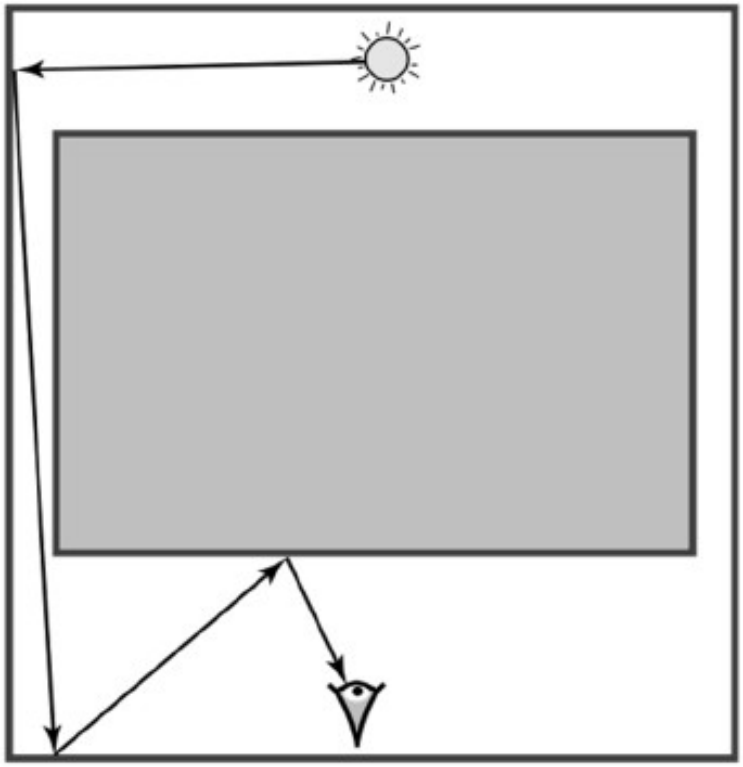
- Caminos complejos de iluminación
  - Muchos vértices no reciben iluminación directa
  - PT explícito no ayuda
    - La contribución viene de caminos indirectos

$$L^0 = L_e^1 + L_d^1 + f_s^1 L_d^2 + f_s^1 f_s^2 L_d^3 + f_s^1 f_s^2 L_d^3 + f_s^1 f_s^2 f_s^3 L_d^4 + \ldots + f_s^1 \cdots f_s^n L_{ind}^{n+1}$$

Problemas con Path Tracing

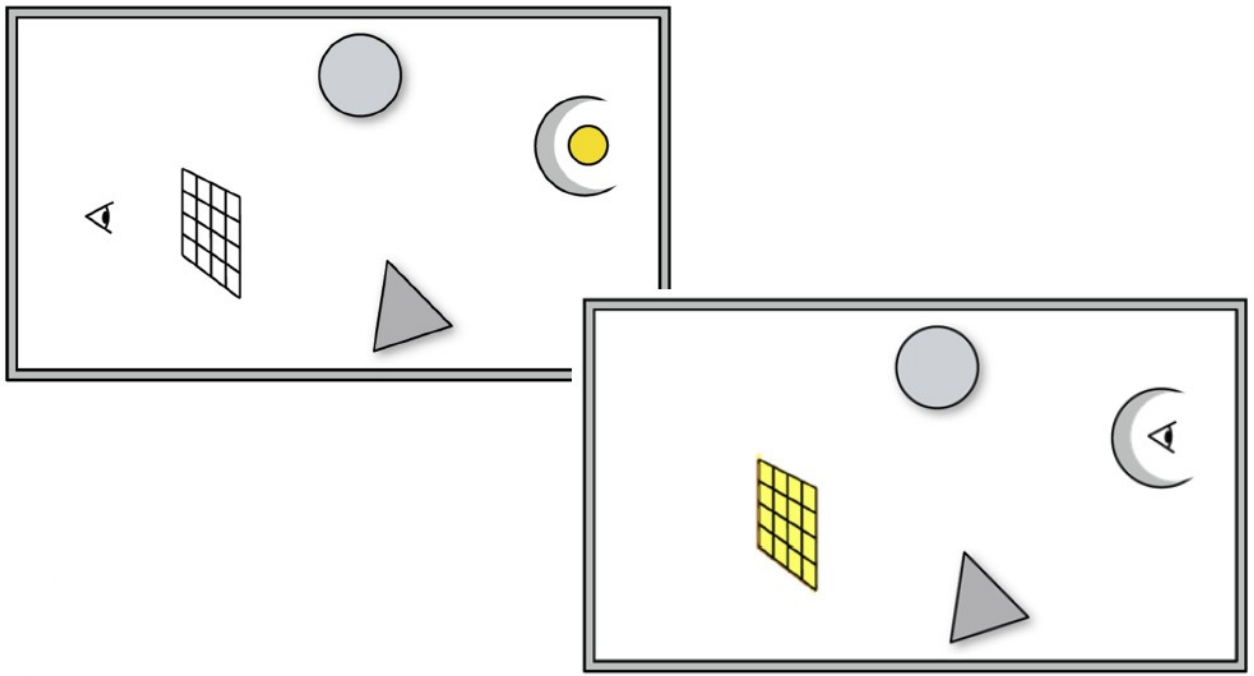
- Caminos complejos de iluminación
  - Muchos vértices no reciben iluminación directa
  - PT explícito no ayuda
    - La contribución viene de caminos indirectos

$$L^0 = L_e^1 + L_d^1 + f_s^1 L_d^2 + f_s^1 f_s^2 L_d^3 + f_s^1 f_s^2 L_d^3 + f_s^1 f_s^2 f_s^3 L_d^4 + \ldots + f_s^1 \cdots f_s^n L_{ind}^{n+1}$$



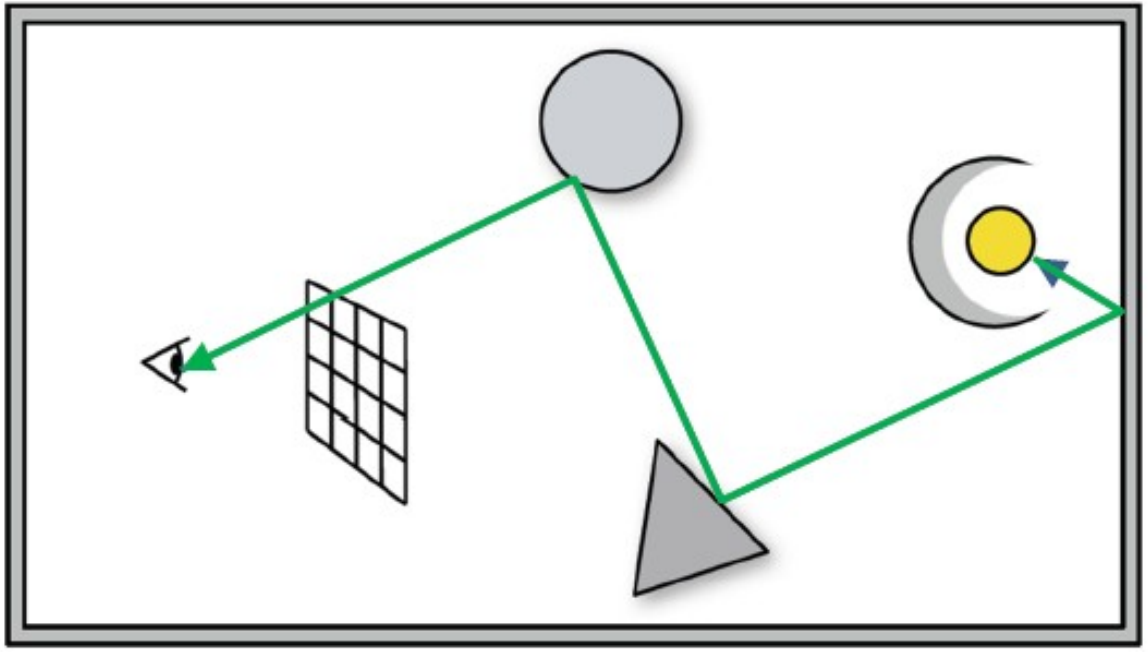
Dualidad cámara-luz

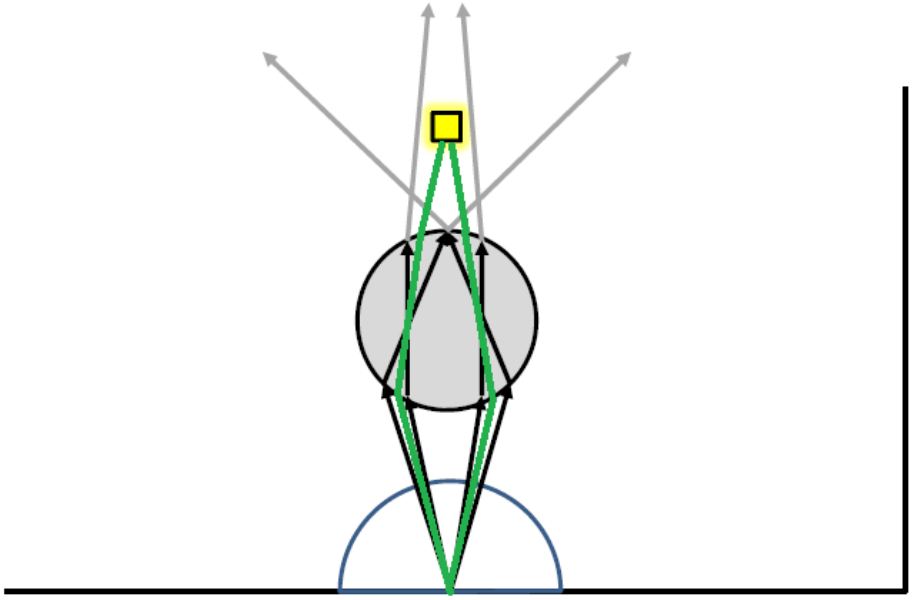
- El transporte de luz es bidireccional
- Físicamente los caminos de luz tienen su origen en las fuentes luminosas
  - ¡Pero PT lo hace en el otro sentido!



Light Tracing

- ¿Por qué no comenzar el camino desde la luz?
  - ¡Todos los caminos ya transportarían luz!





Light Tracing

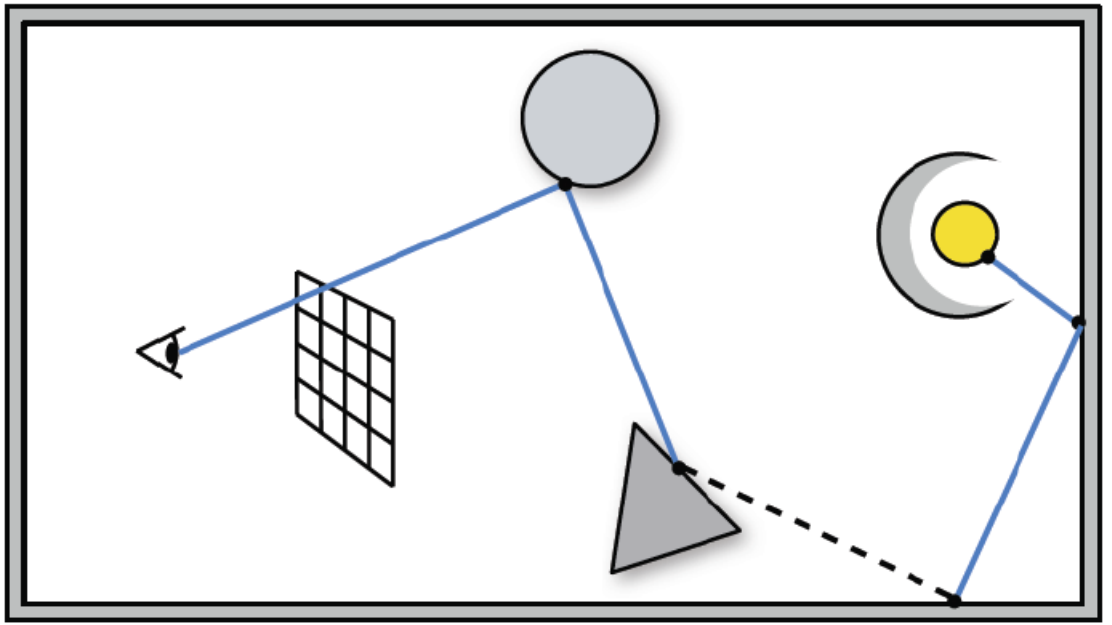
- ¿Por qué no comenzar el camino desde la luz?
  - ¡Todos los caminos ya transportarían luz!

- ¿Esto es eficiente?
  - Muchos rayos jamás alcanzarán el sensor
  - Regresamos al mismo problema...

Bidirectional Path Tracing

- ¡Combina lo mejor de dos mundos!

- Generamos subcaminos desde la cámara y desde la luz
- *Conectamos* estos subcaminos
- Obtenemos un camino completo



Bidirectional Path Tracing

$$L(\mathrm{x}, \omega_\mathbf{o}) = L_e(\mathrm{x}, \omega_\mathbf{o}) + \int_{S^2} L(\mathrm{x}', -\omega_\mathbf{i}) f_s(\mathrm{x}, \omega_\mathbf{i}, \omega_\mathbf{o}) |\cos\theta_\mathbf{i}| \mathrm{d}\omega_\mathbf{i}$$

- La formulación de la LTE con ángulo sólido es reemplazada con una parametrización por áreas:

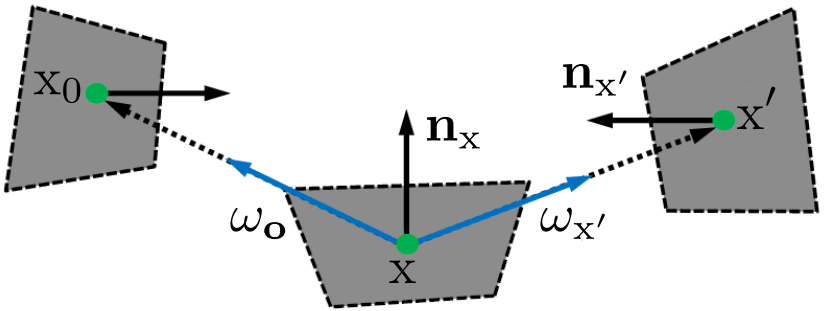
$$L(\mathrm{x}, \omega_\mathbf{o}) = L_e(\mathrm{x}, \omega_\mathbf{o}) + \int_A L(\mathrm{x}', -\omega_{\mathrm{x}'}) f_s(\mathrm{x}, \omega_{\mathrm{x}'}, \omega_\mathbf{o}) G(\mathrm{x} \leftrightarrow \mathrm{x}') \mathrm{d}A(\mathrm{x}')$$

Nota: por claridad mezclamos en la notación direcciones y puntos

Bidirectional Path Tracing

- Donde G es el término geométrico definido como:

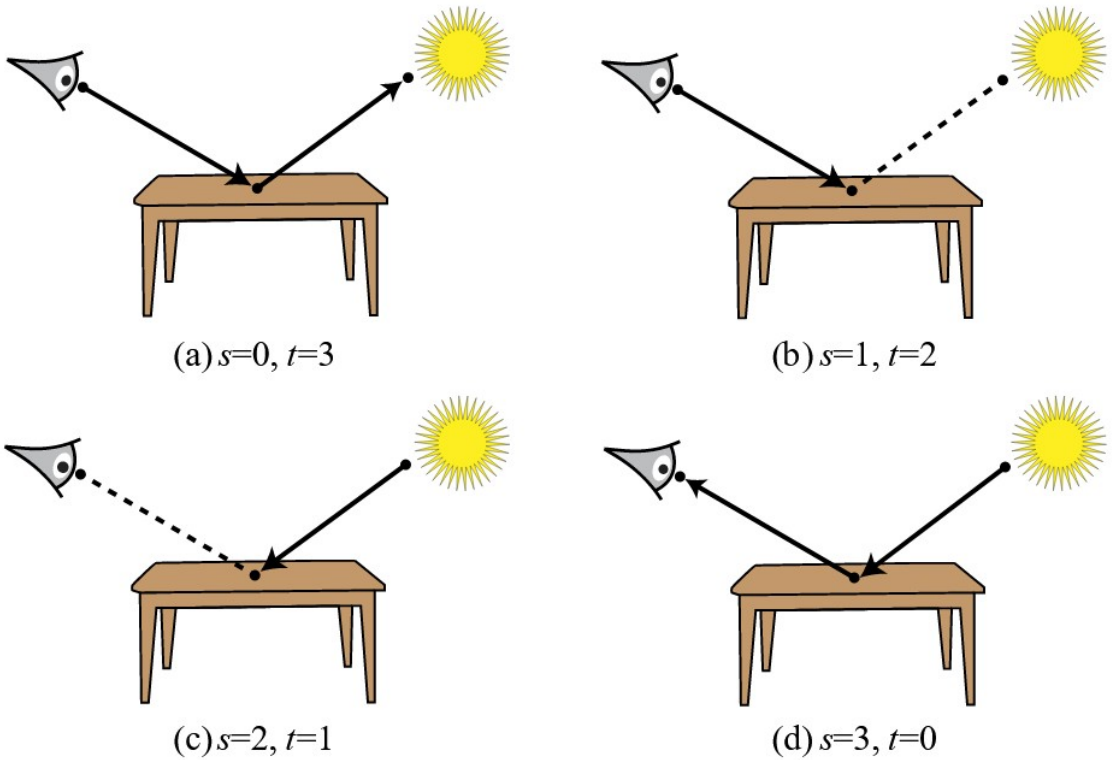
$$G(\mathrm{x} \leftrightarrow \mathrm{x}') = V(\mathrm{x} \leftrightarrow \mathrm{x}') \frac{|\mathbf{n}_{\mathrm{x}} \cdot \omega_{\mathrm{x}'}||\mathbf{n}_{\mathrm{x}'} \cdot -\omega_{\mathrm{x}'}|}{\|\mathrm{x} - \mathrm{x}'\|^2}$$



- Y el cuadrado de la distancia entre los puntos se vuelve ahora una nueva fuente de variancia en el resultado...

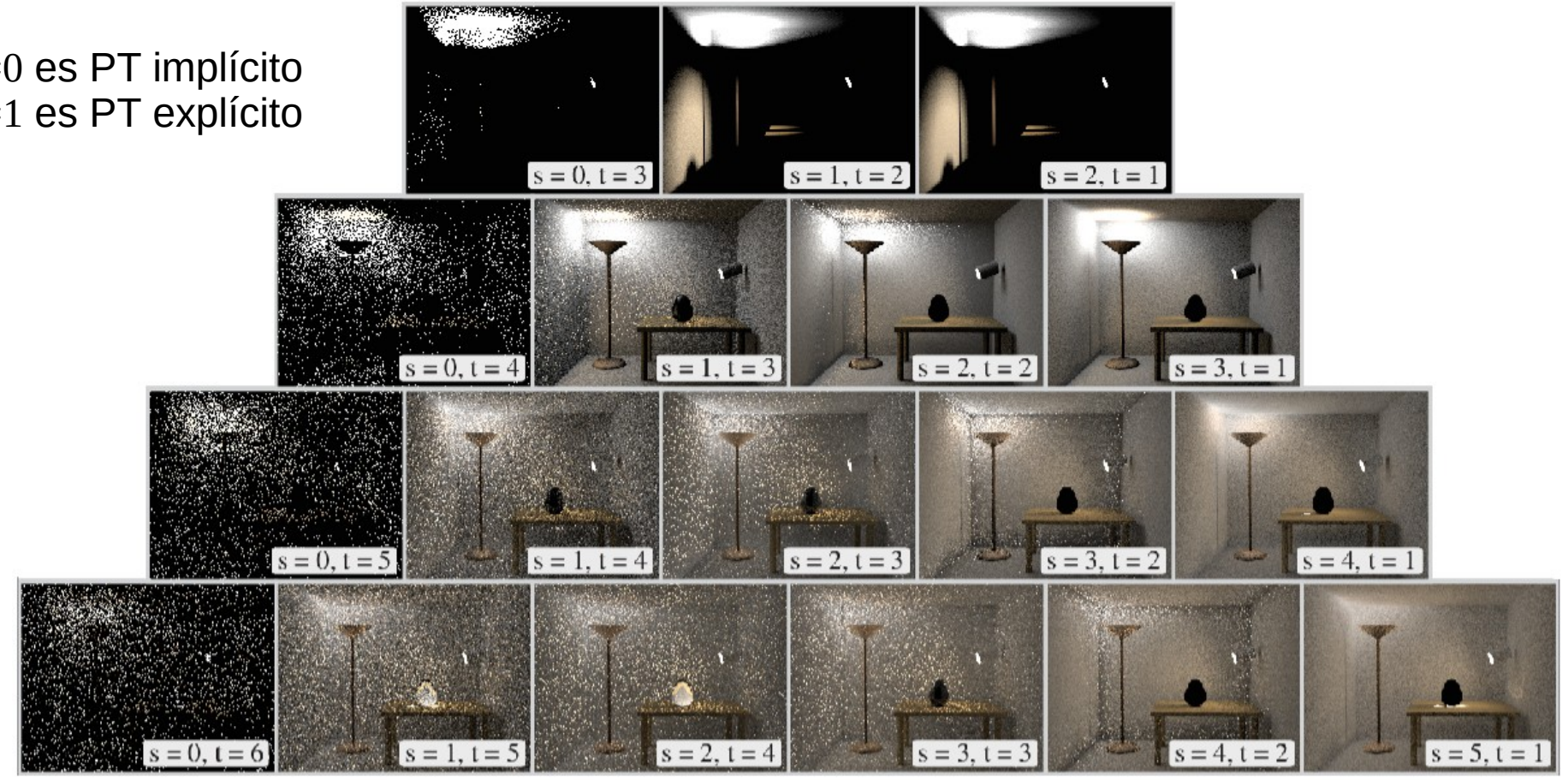
Bidirectional Path Tracing

- ¿Qué subcaminos generar y unir?
  - s indica los caminos desde la luz
  - t indica los caminos desde la cámara
  - Una estrategia (s,t) nos indica cuantos puntos se toman desde la cámara y desde la luz



Estretegias *(s,t)*

s=0 es PT implícito
s=1 es PT explícito



BDPT MIS

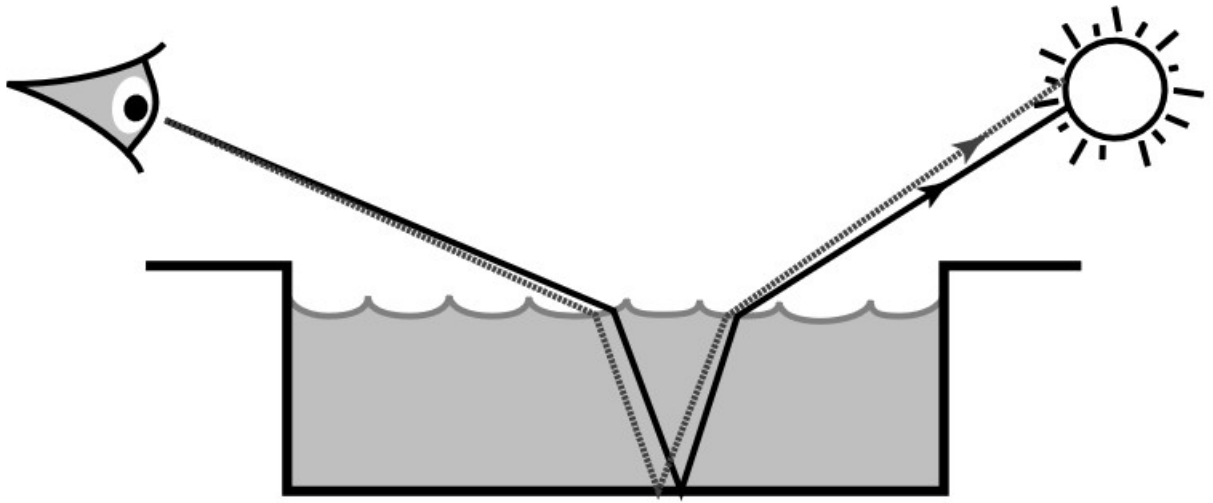
- Cada una de las estrategias tiene mucho ruido
- *¿Apagar* cierta estrategia cuando no se comporta bien?
  - Multiple Importance Sampling
  - Se toman en cuenta *las distintas formas* (estrategias) en que pudo ser generado el mismo camino
    - Muuuuucho overhead

BDPT MIS

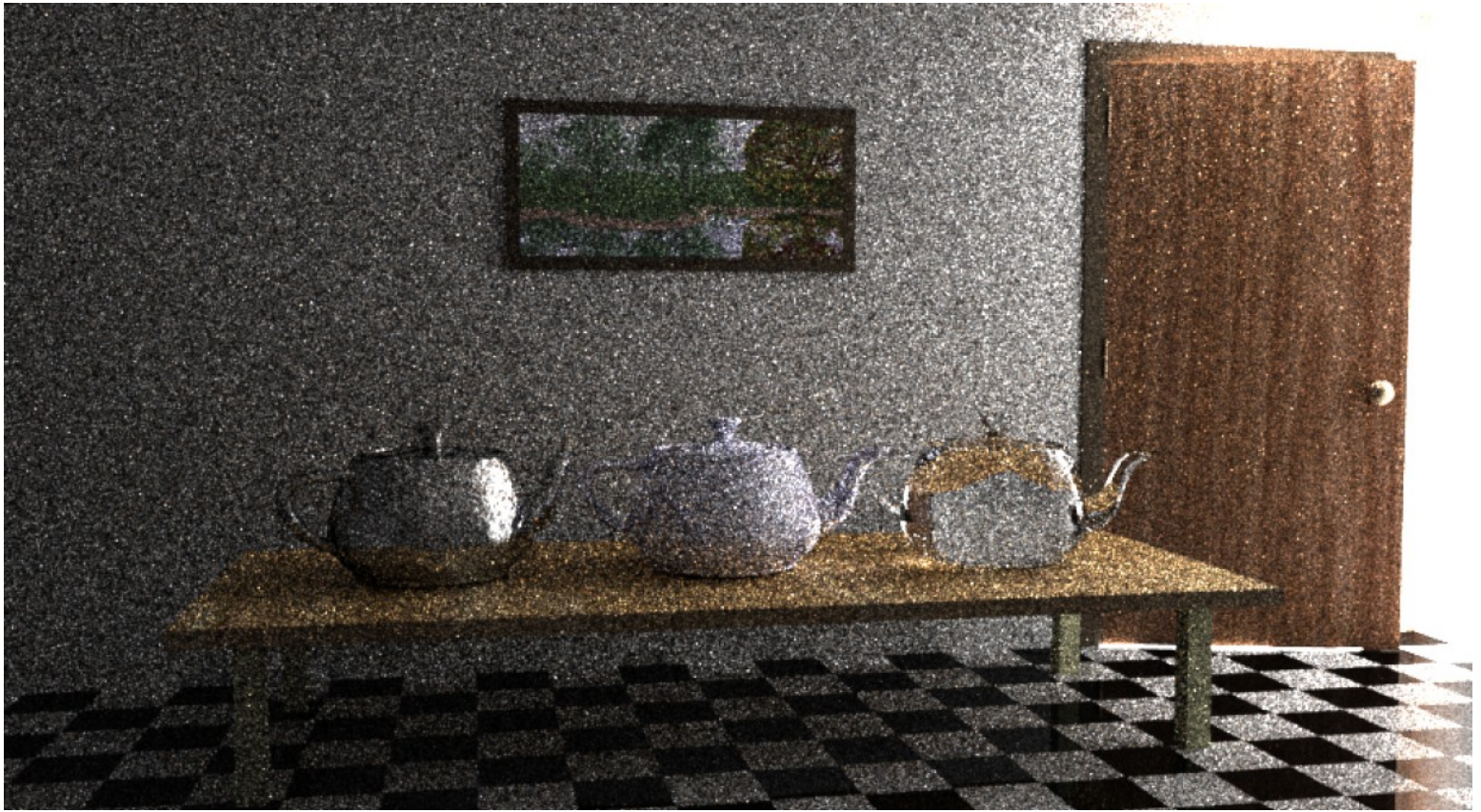

s = 0, t = 3
s = 1, t = 2
s = 2, t = 1
s = 0, t = 4
s = 1, t = 3
s = 2, t = 2
s = 3, t = 1
s = 0, t = 5
s = 1, t = 4
s = 2, t = 3
s = 3, t = 2
s = 4, t = 1
s = 0, t = 6
s = 1, t = 5
s = 2, t = 4
s = 3, t = 3
s = 4, t = 2
s = 5, t = 1

Metropolis Light Transport

- Si se encuentra un camino entonces se hacen *mutaciones* esperando que el camino resultante sea útil
  - Cadenas de Markov



Metropolis Light Transport



BDPT 40spp

Metropolis Light Transport

MLT (Veach) 250 mutaciones por pixel