Stochastic Ray Tracing

Kadi Bouatouch
IRISA
Email: kadi@irisa.fr

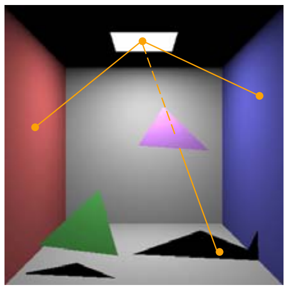
IFSIC IRISA

Contents

- Monte Carlo Integration
  - Application to direct lighting by area light sources
- Solving the radiance equation with the Monte Carlo Method
  – Distributed Ray Tracing
  – Path Traycing

IFSIC IRISA

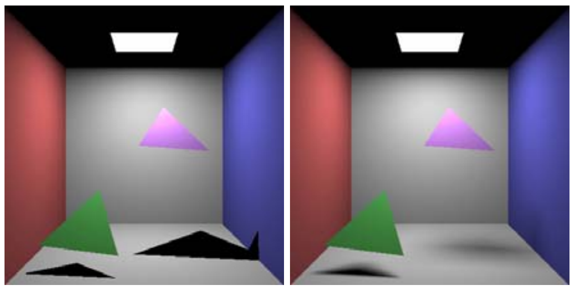
Classical Ray Tracing



- One shadow ray by intersection point
- Only point light sources
- Hard shadows: umbra

IFSIC IRISA

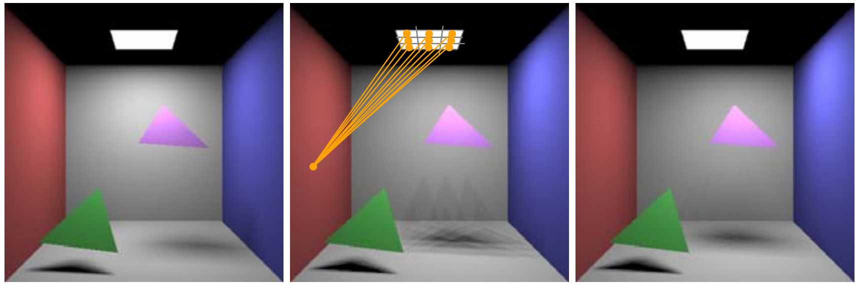
With Area Light Sources



- Soft shadows
- Area light sources != point light sources

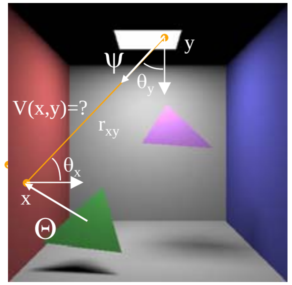
IFSIC IRISA

More Sample Points on the Light Source



- Approximate the source by a set of points
- Aliasing along the shadows' borders

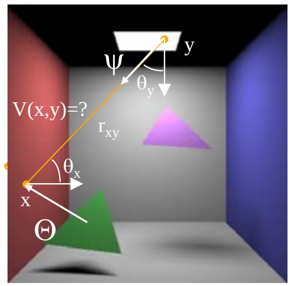
Solution to the Rendering Equation



- Analytical solution: too much difficult
- Use the Monte Carlo method

$$L(x\to\Theta)=\int_{A_L} f_r(x,\Theta\leftrightarrow\Psi)L_e(y\to\Psi)V(x,y)\frac{\cos\theta_x\cos\theta_y}{r_{xy}^2}dy$$

Direct Lighting

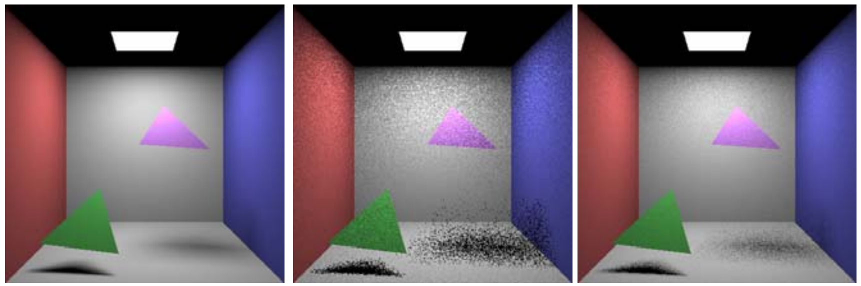


- Random point sampling of the area light sources
- Use these points to evaluate the integral

$$L(x\to\Theta)=\int_{A_L} f_r(x,\Theta\leftrightarrow\Psi)L_e(y\to\Psi)V(x,y)\frac{\cos\theta_x\cos\theta_y}{r_{xy}^2}dy$$

$$L(x\to\Theta)=\int_{A_L} D(y)dy \qquad \langle L(x\to\Theta)\rangle=\frac{1}{N}\sum_{i=1}^{N}\frac{D(y_i)}{p(y_i)}$$

Direct Lighting



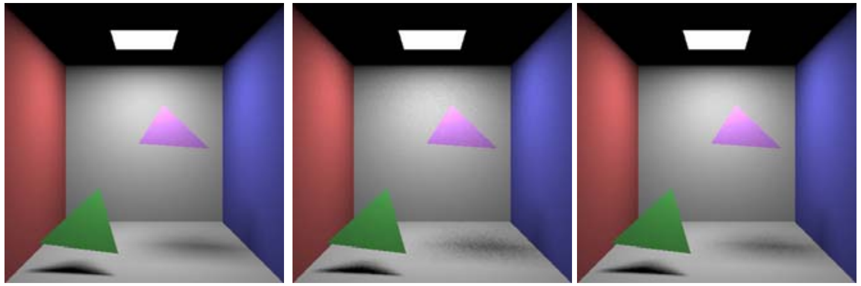
1 shadow ray

9 shadow rays

$$p(y)=\frac{1}{A_s}$$

$$\langle L(x\to\Theta)\rangle=\frac{1}{N}\sum_{i=1}^{N}A_s f(x,\Theta\leftrightarrow\Psi)L_e(y_i\to\Psi_i)\frac{\cos\theta_x\cos\theta_{y_i}}{r_{xy_i}^2}Vis(x,y_i)$$

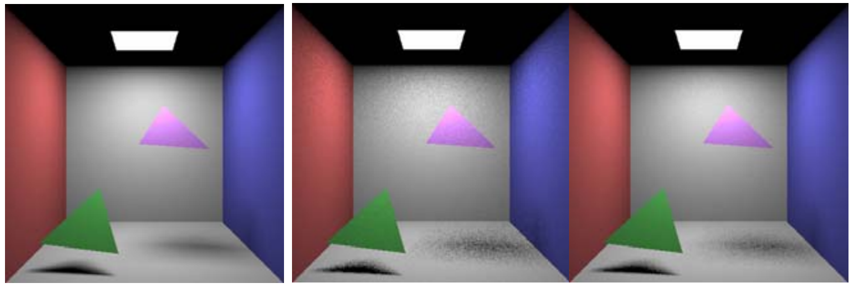
Direct Lighting



36 shadow rays　　100 shadow rays

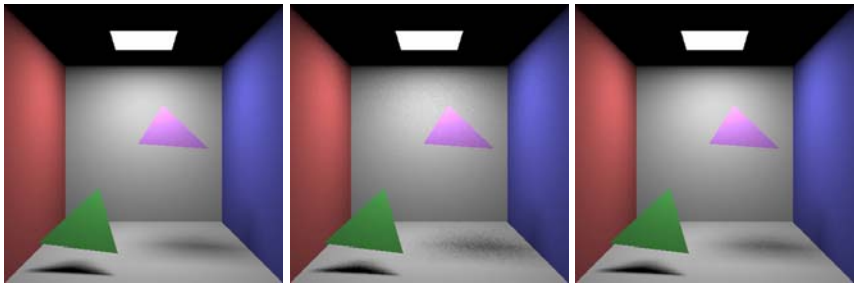
$$\langle L(x\to\Theta)\rangle = \frac{1}{N}\sum_{i=1}^{N} A_s\, f(x,\Theta\leftrightarrow\Psi)\, L_e(y_i\to\Psi_i)\frac{\cos\theta_x\cos\theta_{y_i}}{r_{xy_i}^2}\, Vis(x,y_i)$$

Stratified Sampling



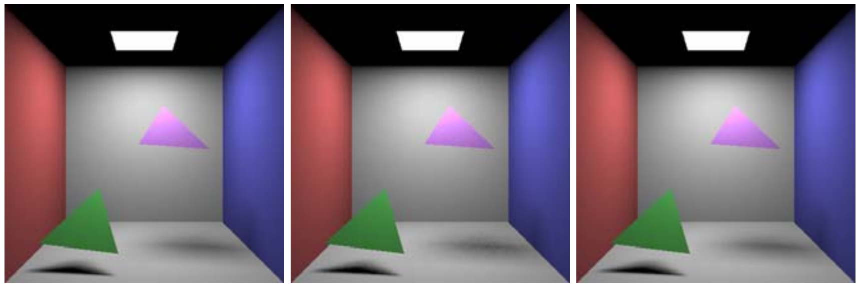
9 shadow rays without stratification　　9 shadow rays with stratification

Stratified Sampling



36 shadow rays no stratification　　36 shadow rays with stratification

Stratified Sampling



100 shadow rays no stratification　　100 shadow rays with stratification

Multiple Light Sources

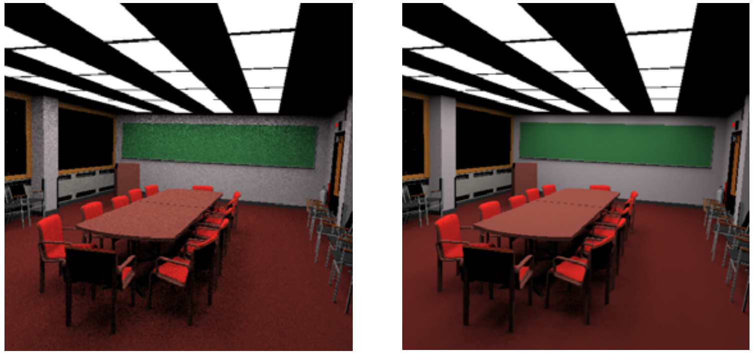
- The integral does not change : rather than integrating over one light source, integrate over all the surfaces of the light sources

$$L(x\to\Theta)=\int_{A_L} f_r(x,\Theta\leftrightarrow\Psi)L_e(y\to\Psi)V(x,y)\frac{\cos\theta_x\cos\theta_y}{r_{xy}^2}dy$$

- The pdf for selecting points is modified : first select a light source S using pdf p(S), then a point y on S with p(y|S)

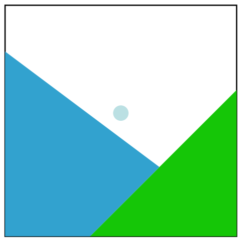
$$p(S)=\frac{\Phi_S}{\Phi_T} \qquad p(y|S)=\frac{1}{A_s} \qquad p(y)=p(S)p(y|S)$$

Multiple Light Sources



36 shadow rays per pixel in the 2 images but with different pdf

Application to pixels: oversampling



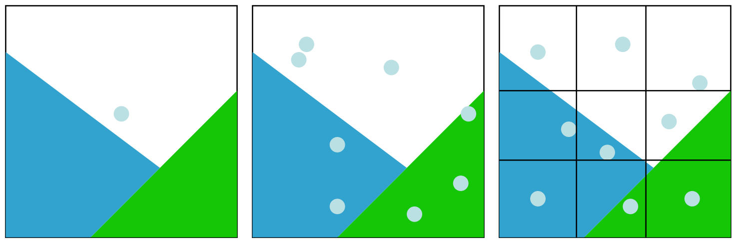
- Compute radiance at the centre of a pixel → aliasing
- Oversample a pixel and compute radiance for sub-pixels
- Use a filter

$$L=\int_{Pixel} f(x)L(x)dx$$

… evaluated by the Monte Carlo method.

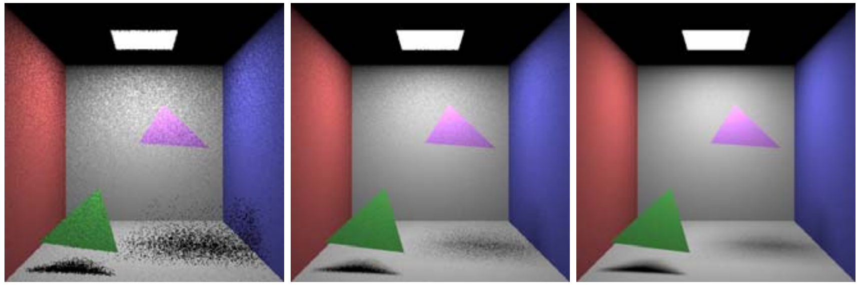
Application to pixels: oversampling

- Any sampling method



$$L\approx\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)L(x_i)}{p(x_i)}$$

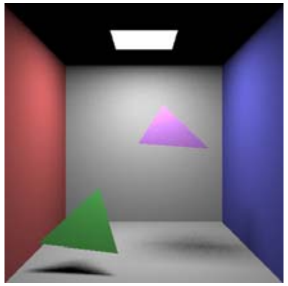
Application to pixels: oversampling



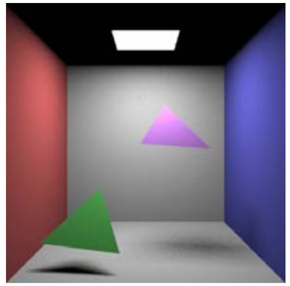
1 ray / pixel | 10 rays / pixel | 100 rays / pixel

Implementation

- Comparison :



1 centered ray per pixel
100 random shadow rays per intersection



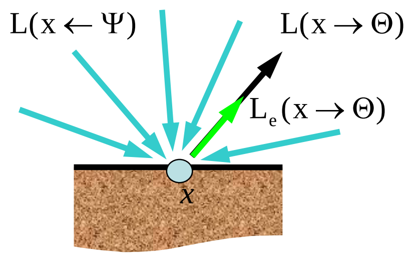
100 centered rays per pixel
1 random shadow rays per intersection

The Rendering Equation

- Evaluation of the rendering equation
- How to write the rendering equation and how to evaluate it using Monte Carlo integration?
  - Which pdf to use for the rendering equation ?
  - Algorithms and results

The Rendering Equation

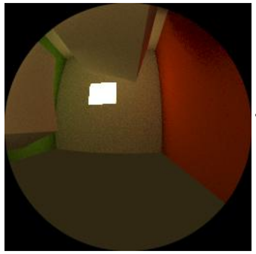
$$L(x\to\Theta)=L_e(x\to\Theta)+\int_{\Omega_x} f_r(x,\Psi\leftrightarrow\Theta)\cdot L(x\leftarrow\Psi)\cdot\cos(\Psi,n_x)\cdot d\omega_\Psi$$



The Rendering Equation

$$L(x \to \Theta) = L_e(x \to \Theta) + \int_{\Omega_x} f_r(x, \Psi \leftrightarrow \Theta) \cdot L(x \leftarrow \Psi) \cdot \cos(\Psi, n_x) \cdot d\omega_\Psi$$

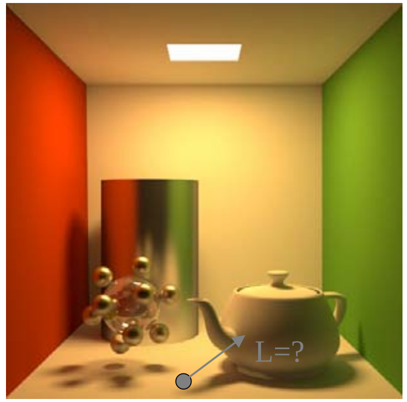
$$= L_e + \int_{\Omega_x}$$



$$\cdot f_r \cdot \cos$$

Computing Radiance

- How to evaluate L ?
  - Find $L_e(x \to \Theta)$
  - Add:



$$\int_{\Omega_x} f_r(x, \Psi \leftrightarrow \Theta) \cdot L(x \leftarrow \Psi) \cdot \cos(\Psi, n_x) \cdot d\omega_\Psi$$

Computing Radiance

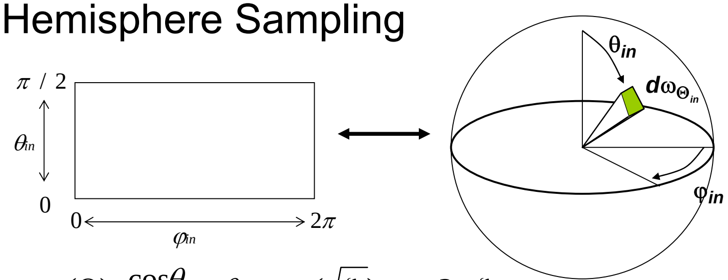
- How to evaluate L ?
  - Monte Carlo Integration
  - Generate random directions on Ω_x, using the probability density function $p(\Psi)$

$$L(x \to \Theta) = \int_{\Omega_x} f_r(x, \Psi \leftrightarrow \Theta) \cdot L(x \leftarrow \Psi) \cdot \cos(\Psi, n_x) \cdot d\omega_\Psi$$

$$\langle L(x \to \Theta) \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f_r(x, \Psi_i \leftrightarrow \Theta) \cdot L(x \leftarrow \Psi_i) \cdot \cos(\Psi_i, n_x)}{p(\Psi_i)}$$

Computing Radiance

- Hemisphere Sampling



$$p(\Theta) = \frac{\cos\theta}{\pi} \Rightarrow \theta = a\cos(\sqrt{\xi_1})\, et\, \varphi = 2\pi\xi_2$$

$$\Theta = (\theta, \varphi) \Rightarrow d\omega_\Theta = \sin\theta \cdot d\theta \cdot d\varphi$$

$$p(\Theta) = \frac{n+1}{2\pi} \cos^n\theta \Rightarrow \theta = a\cos(\xi_1^{\frac{1}{n+1}})\, et\, \varphi = 2\pi\xi_2$$

Computing Radiance

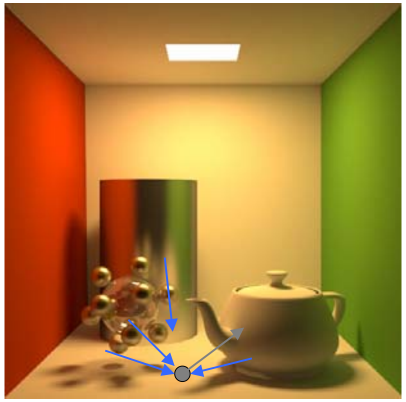
- Generate a random direction Ψ_i

$$\langle L\rangle = \frac{1}{N}\sum_{i=1}^{N}\frac{f_r(\ldots)\cdot L(x\leftarrow\Psi_i)\cdot\cos(\ldots)}{p(\Psi_i)}$$

Evaluate the BRDF

Evaluate the cos(…)

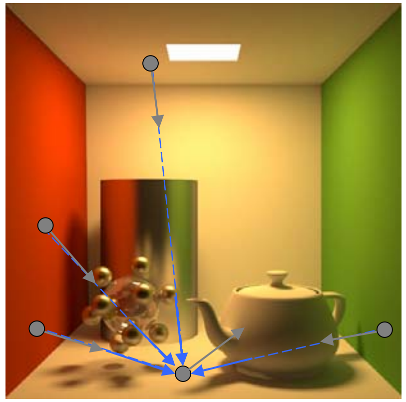
Evaluate $L(x\leftarrow\Psi_i)$



IRISA

Computing Radiance

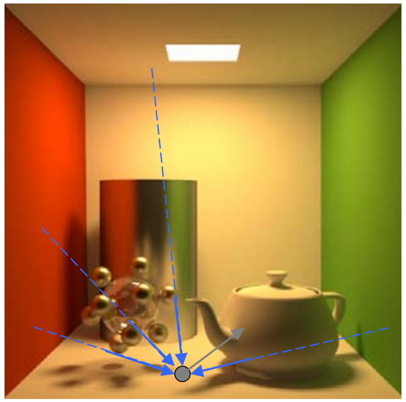
- Evaluation of $L(x\leftarrow\Psi_i)$?
- Radiance is constant along the propagation direction.
- $rc(x, \Psi_i)$ = first visible point.
- $L(x\leftarrow\Psi_i) = L(rc(x, \Psi_i) \rightarrow \Psi_i)$



IRISA

Computing Radiance

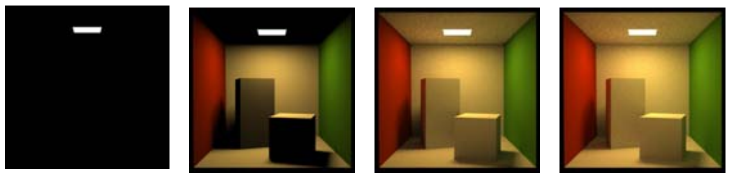
- Recursive Evaluation
- Each bounce adds a level of indirect lighting.



IRISA

Stopping Recursion

- When recursion is stopped?



The contributions of higher order reflections are negligible.

If we ignore them, the estimates are biased !

IRISA

Terminating the Recursion

- When/how do we stop the recursion?
  - When the ray doesn't hit any object
    - Can be very hard/impossible for dense scenes
  - When a maximum depth is reached
    - This is highly scene dependent
      - Primarily specular scenes requires far more bounces than diffuse scenes
    - Having a fixed path length results in a biased estimate
  - When the contribution of the ray falls below a certain threshold
    - More efficient than a fixed max depth, but still gives a biased result

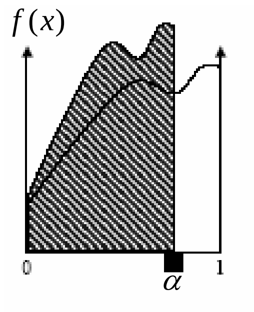
Russian Roulette

- Integral

$$I = \int_0^1 f(x)dx = \frac{1}{\alpha}\int_0^{\alpha} f(x/\alpha)dx$$

- Estimator

$$< I_{roulette} >= \begin{cases} \frac{1}{\alpha} f(x/\alpha) & \text{if } x < \alpha \\ 0 & \text{if } x > \alpha \end{cases}$$



- Use Russian roulette to decide
  - ray is absorbed with probability 1-α
  - results in unbiased estimator

Russian Roulette

- A simple and unbiased termination criteria is *Russian roulette*:
  - Given a uniform random number ξ, terminate the ray if ξ ≥ α, otherwise scale the contribution of the ray by 1/α
    - Here α є [0,1] is the absorption probability
  - Recursion stops with a probability of p = 1- α
  - By scaling the contribution of rays that continue by 1/ α, the result remains unbiased
- Russian roulette is not practically useful until we add direct light to our ray tracer!!

Russian Roulette

- Example
  - p = 0.9 , then α = 1- p = 0.1
  - One chance in 10 that ray is reflected.
  - The radiance due to one reflected ray is multiplied by 10.
  - instead of shooting 10 rays, we shoot only 1, but count the contribution of this one 10 times

Russian Roulette

- Case of n incident rays

$$\langle L(x \to \Theta) \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f_r(x, \Psi_i \leftrightarrow \Theta) \cdot L(x \leftarrow \Psi_i) \cdot \cos(\Psi_i, n_x)}{p(\Psi_i)}$$

- Case of one incident ray

$$\langle L(x \to \Theta) \rangle = \frac{f_r(x, Di \leftrightarrow \Theta) \cdot L(x \leftarrow Di) \cdot \cos(Di, n_x)}{p(Di)}$$

$$= R(x, Di).L(x \leftarrow Di)$$

Russian Roulette

- With Russian roulette the pseudo code now looks like this:

```
RGB radiance(Ray r)
   if(r hits at x)
        if(ξ < α)
            Generate new direction, Di, from p(Ψi) and
                  the surface normal at x
            Ray ray(x, Di)
            return Le(x) + R(x,Di)*radiance(ray) / α
        else
           return Le(x)
   else
      return background
```


Russian Roulette

- Non biased Estimate
- The expected value is correct
- Bigger variance
- But more efficient

Distribution Ray Tracing

- Algorithm
  - Trace N rays per pixel
  - At each intersection point, trace 1 ray (or more) randomly chosen on the hemisphere to evaluate the rendering equation
  - End recursion using the Russian roulette

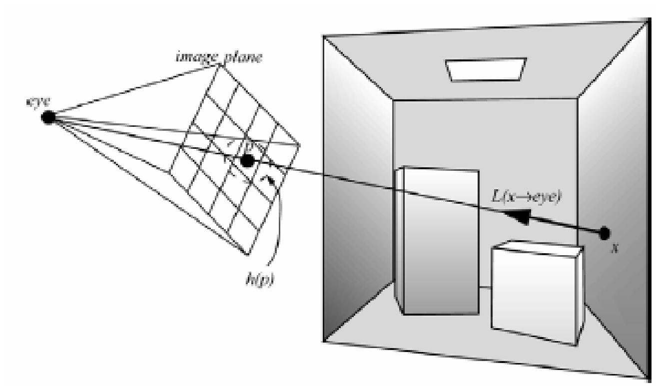
Distribution Ray Tracing

```
computeImage() {
    for each pixel (i,j) {
        estimatedRadiance[i,j] = 0
        for s = 1 to #samples-in-pixel {
            generate Q in pixel (i,j)
            theta = (Q – E)/|Q-E|   // E is the Eye
            x = trace(E,theta)
            estimatedRadiance [i,j] += computeRadiance(x,-theta)
        }
        estimatedRadiance [i,j] /= # samples-in-pixel
    }
}
computeRadiance(x, theta) {
    estimatedRadiance = basicPT(x, theta)
    return estimatedRadiance
}
```

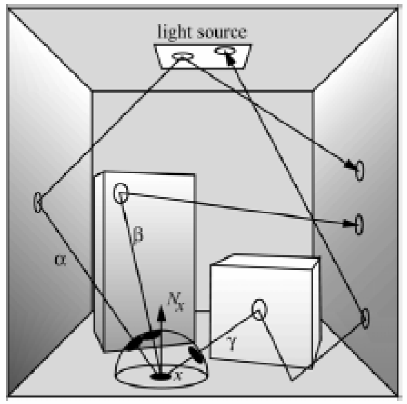

Distribution Ray Tracing

```
basicPT(x, theta) {
    estimatedRadiance = Le(x, theta)
    if(not absorbed)  {   // russian roulette
        for s = 1 to #radianceSamples  { // ray directions
            psi = generate random direction on hemisphere
            y = trace(x, psi)
            estimatedRadiance +=
                basicPT(y,-psi) * BRDF(x,psi,theta)*
                cos(Nx,psi) / pdf(psi)
        }
        estimatedRadiance /= #radianceSamples
    }
    return estimatedRadiance/(absorption)
}
```

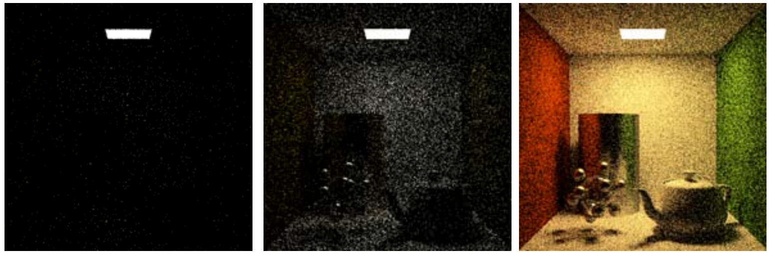

Distribution Ray Tracing



Distribution Ray Tracing



Distribution Ray Tracing

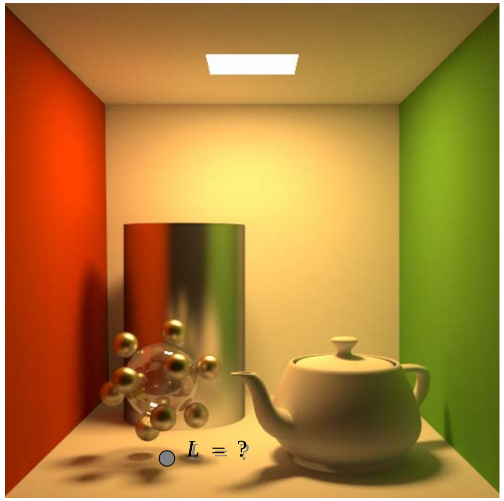


1 ray/pixel 16 rays/pixel 256 rays/pixel

Very noisy : null contribution as long as the path does not reach a light source!!

IRISA

Distribution Ray Tracing



IRISA

Distribution Ray Tracing

IRISA

Distribution Ray Tracing

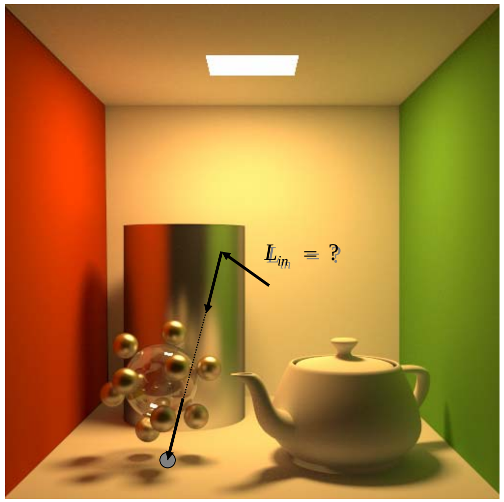
IRISA

Distribution Ray Tracing

Distribution Ray Tracing

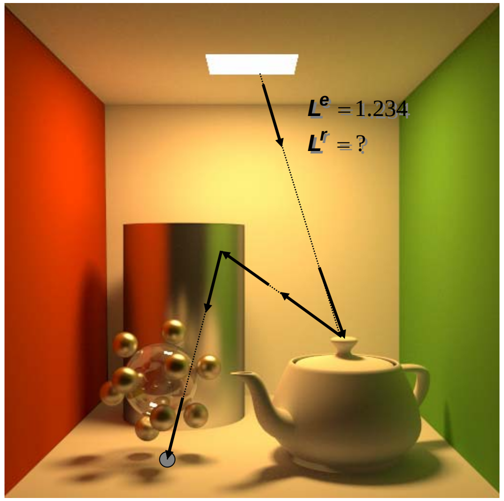
Distribution Ray Tracing

Distribution Ray Tracing

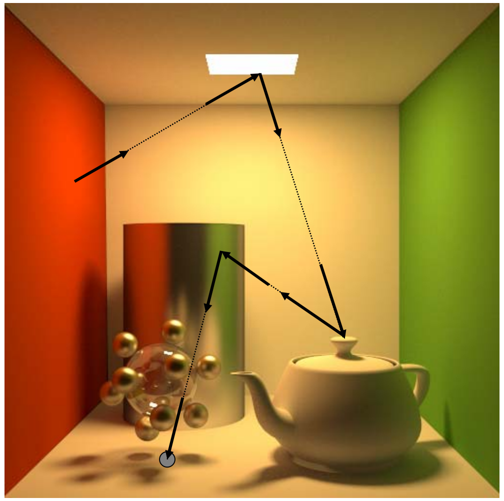


Distribution Ray Tracing

Distribution Ray Tracing



Distribution Ray Tracing



Distribution Ray Tracing

- Improve the algorithm by dividing the integral into two parts: direct and indirect

$$L_i(x\to\Theta)=\int_{\Omega_x} f_r(x,\Psi\leftrightarrow\Theta)\cdot L_e(x\leftarrow\Psi)\cdot\cos(\Psi,n_x)\cdot d\omega_\Psi$$

$$+\int_{\Omega_x} f_r(x,\Psi\leftrightarrow\Theta)\cdot L_i(x\leftarrow\Psi)\cdot\cos(\Psi,n_x)\cdot d\omega_\Psi$$

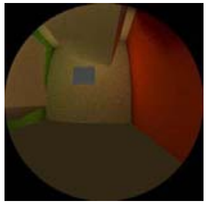
$$L(x\to\Theta)=L_e(x\to\Theta)+L_i(x\to\Theta)$$

Distribution Ray Tracing

– Evaluate differently the direct and the indirect components

$$\int_{Source}$$

$$\cdot f_r \cdot \cos + \int_{\Omega_x}$$



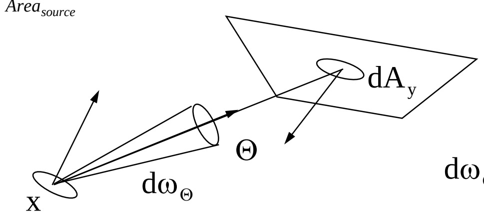
$$\cdot f_r \cdot \cos$$

Direct Illumination

$$L_{direct}(x \to \Theta) = \int_{\Omega_x} f_r(\ldots) \cdot L_e(x \leftarrow \Psi) \cdot \cos(\ldots) \cdot d\omega_\Psi$$

$$= \int_{\Omega_x} f_r(\ldots) \cdot L_e(y \to x) \cdot \cos(\ldots) \cdot d\omega_\Psi$$

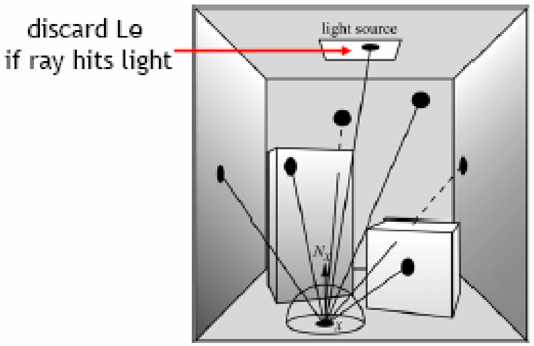
$$= \int_{Area_{source}} f_r(\ldots) \cdot L_e(y \to x) \cdot \cos(\ldots) \cdot (\frac{\cos(\ldots) dA_y}{r_{xy}^2}) \cdot V(x,y)$$



$$d\omega_\Theta = \frac{dA_y \cos\theta_y}{r^2}$$

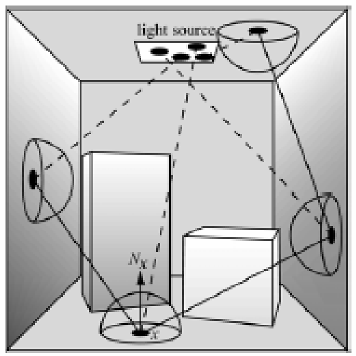
Indirect Illumination

$$L_i(x \to \Theta) = \int_{\Omega_x} f_r(x, \Psi \leftrightarrow \Theta) \cdot L_i(x \leftarrow \Psi) \cdot \cos(\Psi, n_x) \cdot d\omega_\Psi$$



Indirect Illumination

$$L_i(x \to \Theta) = \int_{\Omega_x} f_r(x, \Psi \leftrightarrow \Theta) \cdot L_i(x \leftarrow \Psi) \cdot \cos(\Psi, n_x) \cdot d\omega_\Psi$$



Distribution Ray Tracing

$$L(x\rightarrow\Theta)=\int_{\Omega_x} f_r(x,\Psi\leftrightarrow\Theta)\cdot L(x\leftarrow\Psi)\cdot\cos(\Psi,n_x)\cdot d\omega_\Psi$$

$$\langle L(x\rightarrow\Theta)\rangle=\frac{1}{N}\sum_{i=1}^{N}\frac{f_r(x,\Psi_i\leftrightarrow\Theta)\cdot L(x\leftarrow\Psi_i)\cdot\cos(\Psi_i,n_x)}{p(\Psi_i)}$$

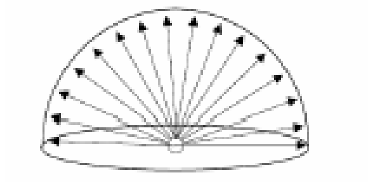
- Depends on how to sample the hemisphere
  - Uniform distribution
  - Importance sampling : pick p to match integral
    - Cosine distribution
    - BRDF distribution
    - BRDF*cosine distribution

Distribution Ray Tracing

- **Uniform distribution**

$$p(\Psi)=\frac{1}{2\pi}$$

$$\langle L(x\rightarrow\Theta)\rangle=\frac{2\pi}{N}\sum_{i=1}^{N} f_r(x,\Psi_i\leftrightarrow\Theta)\cdot L(x\leftarrow\Psi_i)\cdot\cos(\Psi_i,n_x)$$

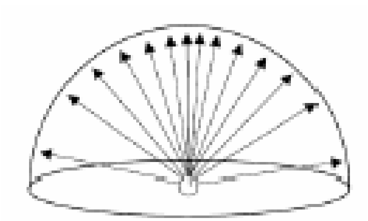


Distribution Ray Tracing

- **Cosine distribution**

$$p(\Psi)=\frac{\cos\theta_\Psi}{\pi}$$

$$\langle L(x\rightarrow\Theta)\rangle=\frac{\pi}{N}\sum_{i=1}^{N} f_r(x,\Psi_i\leftrightarrow\Theta)\cdot L(x\leftarrow\Psi_i)$$

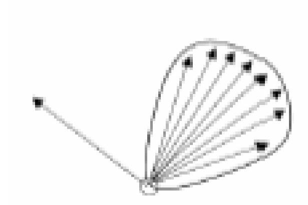


Distribution Ray Tracing

- **BRDF distribution**

$$p(\Psi)=f_r(\ldots)$$

$$\langle L(x\rightarrow\Theta)\rangle=\frac{1}{N}\sum_{i=1}^{N} L(x\leftarrow\Psi_i)\cdot\cos(\Psi_i,n_x)$$

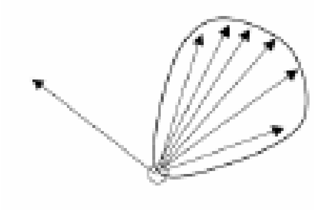


Distribution Ray Tracing

- **BRDF*cosine distribution**

$$p(\Psi) = f_r(\ldots).\cos\theta_\Psi$$

$$\langle L(x \to \Theta) \rangle = \frac{1}{N} \sum_{i=1}^{N} L(x \leftarrow \Psi_i)$$



Distribution Ray Tracing: pixel sampling

```
computeImage() {
   for (each pixel (i,j)) {
        estimatedRadiance[i,j] = 0
        for (s = 1 to #samples-in-pixel) {
                generate Q in pixel (i,j)
                theta = (Q – E)/|Q-E|
                x = trace(E,theta)
                estimatedRadiance [i,j] +=
                        computeRadiance(x,-theta)
        }
        estimatedRadiance [i,j] /= #samples-in-pixel
   }
}
```


Distribution Ray Tracing: radiance estimation

```
computeRadiance(x,theta) {
   estimatedRadiance = Le(x,theta)
   estimatedRadiance += directIllumination(x, theta)
   estimatedRadiance += indirectIllumination(x, theta)
   return estimatedRadiance
}
```


Distribution Ray Tracing: direct illumination

```
directIllumination(x,theta) {
   estimatedRadiance = 0
   for (s = 1 to #shadowRays) {
        k = pick random light
        y = generate random point on light k
        psi = (x-y) / |x-y|
        estimatedRadiance += Le_k(y,-psi) *
                BRDF(x,psi,tetha) *G(x,y) * V(x,y) /(p(k)*p(y|k))
   }
   estimateRadiance /= #shadowRays
   return estimatedRadiance
}
```

Distribution Ray Tracing: direct illumination

```
directIllumination(x,theta) {
   estimatedRadiance = 0
   for (k=1 to #lights) {
     for (s = 1 to #shadowRays) {
        y = generate random point on light k
        psi = (x-y) / |x-y|
        estimatedRadiance += Le_k(y,-psi) *
                BRDF(x,psi,tetha) *G(x,y) * V(x,y) /p(y)
     }
   }
   estimateRadiance /= #shadowRays
   return estimatedRadiance
}
```


Distribution Ray Tracing: indirect illumination

```
indirectIllumination(x,theta) {
   estimatedRadiance = 0
   if (not absorbed)  {      // russian roulette
        for (s = 1 to #indirectDirectionSamples) {
           psi = generate random direction on hemisphere
           y = trace(x, psi)
           estimatedRadiance += computeRadiance(y,-psi) *
                        BRDF(x,psi,theta) *cos(Nx,psi) / pdf(psi)
        }
        estimatedRadiance /= #indirectDirectionSamples
   }
   return estimatedRadiance /(absorption)
}
```


Distribution Ray Tracing

- **To sum up**
  - **For primary rays :**
    - **use many ray samples at the intersection point**
    - **Use uniform or cosine pdf to sample the hemisphere**
  - **For shadow rays :**
    - **Use uniform area-based pdf to sample the light sources**
    - **Use many samples**
  - **For secondary rays**
    - **Use one or more samples**
    - **Use BRDF based pdf**

Tracing Lambertian Materials

- We've already seen that for Lambertian materials, R is just a constant between 0 and 1
- To generate a random ray direction, we use the cosine density $p(\theta,\phi)=\frac{1}{\pi}\cos\theta$ and two uniform random numbers ξ_1 and ξ_2
  - Using the techniques presented before, we find that

$$\cos\theta = \sqrt{1-\xi_1}$$

$$\phi = 2\pi\xi_2$$

Imperfect Specular Reflections

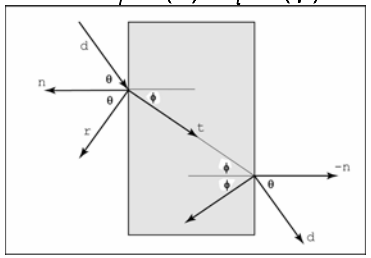
- Perfectly reflecting materials are rare
  - Usually, the reflection is slightly blurred
- To achieve an imperfect specular reflection, we can choose the reflection direction from a phong density:

$$p(\mathbf{k}) = \frac{m+1}{2\pi}(\mathbf{r}\cdot\mathbf{k})^m \Rightarrow \begin{array}{c} \cos\theta = (1-\xi_1)^{\frac{1}{m+1}} \\ \phi = 2\pi\xi_2 \end{array}$$

  - Where **r** is the mirror reflection direction and θ is the angle between **r** and **k**

Transparent Materials

- When a ray hits a transparent material, it is either reflected or transmitted
  - When a ray is transmitted from a medium with refractive index n_i to a medium with refractive index n_t, it is bend according to Snell's law: $n_i \sin(\theta) = n_t \sin(\varphi)$
  - For an incident angle of θ, the fraction of incident rays that are reflected is R(θ). 1-R(θ) is the fraction of transmitted rays



Path tracing

- At each intersection point one can make a choice
  - Reflection or refraction?
  - If reflection : diffuse or specular?

Reflection or Transmission?

- When striking a transparent surface, we need to make a choice: Should the new ray be a reflected or transmitted ray?
- We can set a transmission probability P, and then pick a random number ξ.
  - If ξ<P, the ray is transmitted, and the contribution is scaled by 1/P
  - Else, the ray is reflected, and the contribution is scaled by 1/(1-P)

Beer's Law

- When light travels through an 'impure' medium, it's radiance is attenuated according to Beer's law:

$$I(s) = I(0)e^{\ln(a)s}$$

  – Here I(*s*) is the radiance of a ray at a distance *s* from the interface and *a* is the RGB attenuation constant

IRISA

Specular-Diffuse Surfaces

- Most surfaces reflects light is some combination of specular and diffuse reflections
  – When the angle between the view vector and the normal increases, the specular reflection increases and the diffuse decreases
- We model such materials by linearly combining a specular and a diffuse material

IRISA

Specular-Diffuse Surfaces

- We can choose the specular ray with probability P and the diffuse ray with probability 1-P

if(ξ < P)

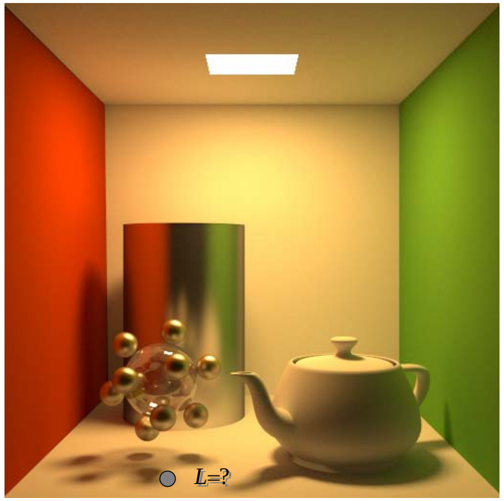
  return R(x,Di)*radiance(specular ray)/P

else

 return R(x)*radiance(diffuse ray)/(1-P)

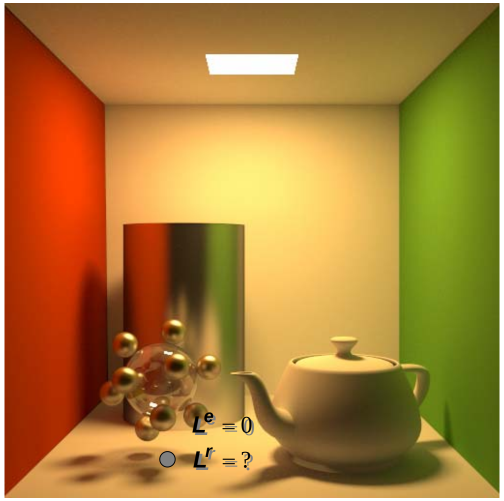
IRISA

Results

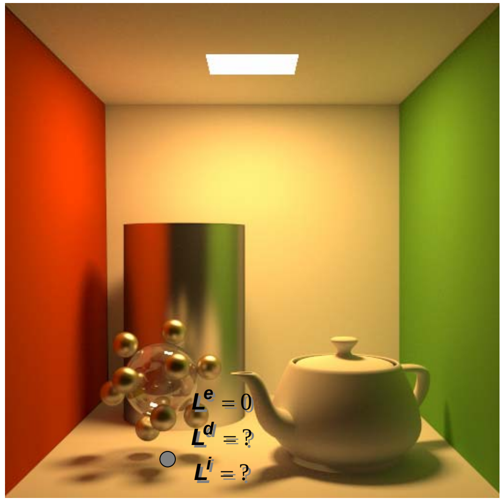


IRISA

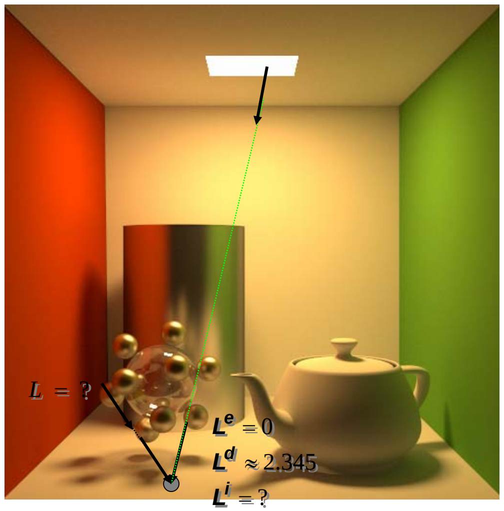
Results



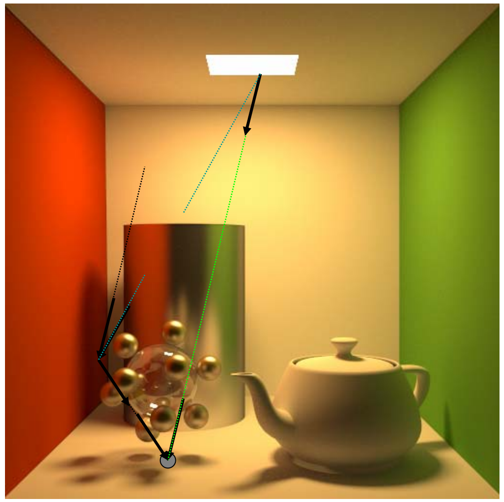
Results



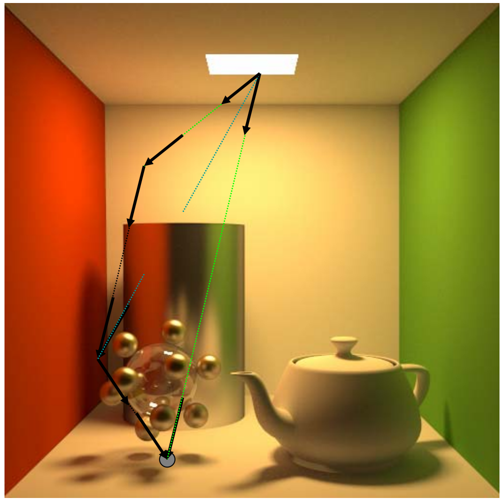
Results



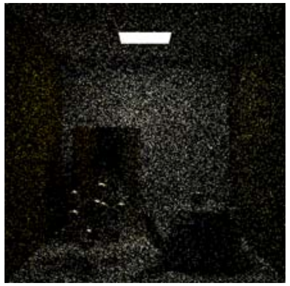
Results



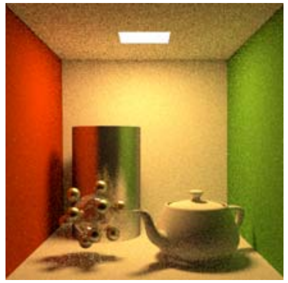
Results



Comparison



Without computing direct lighting



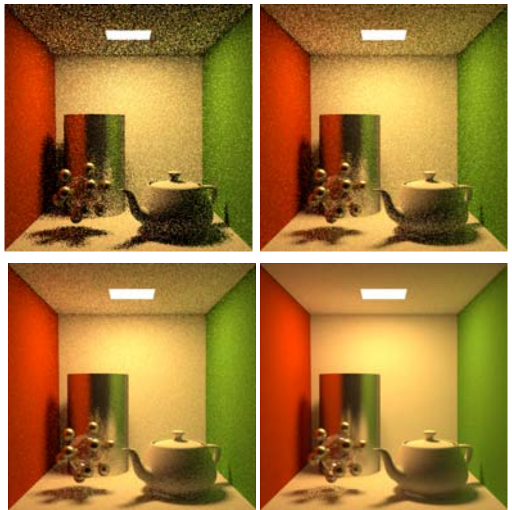
With direct lighting computation

16 rays/pixel

Comparison

1 ray/ pixel

4 rays/ pixel



16 rays/ pixel

256 rays/ pixel