Path Tracing

Path Tracing

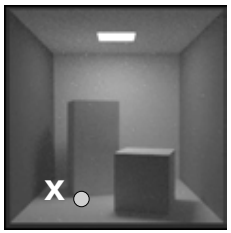
Monte Carlo algorithm for solving the rendering equation

Solving Rendering Equation

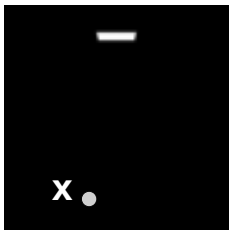
- advantages
  - predictive simulation
    - can be used for architecture, engineering, …
  - photorealistic
    - if simulation if correct, images will look real
- disadvantages
  - (really) slow
    - simulation of physics is computationally very expensive
  - need accurate geometry, materials and lights
    - otherwise just a correct solution to the wrong problem

Rendering Equation

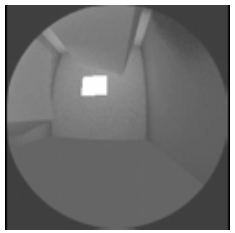
$$L(\mathbf{x} \to \Theta) = L_e(\mathbf{x} \to \Theta) + \int_{\Psi \in \Omega_\mathbf{x}} L(\mathbf{x} \leftarrow \Psi)\rho(\mathbf{x}, \Psi \to \Theta)\cos\theta_\Psi d\omega_\Psi$$



$L(\mathbf{x} \to \Theta)$



$L_e(\mathbf{x} \to \Theta)$



$L(\mathbf{x} \leftarrow \Psi)$

[Bala]

Basic Path Tracing

- need to evaluate radiance at x in direction Θ

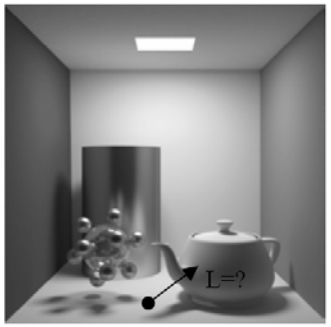
$$L(\mathbf{x} \to \Theta) = L_e(\mathbf{x} \to \Theta) + L_r(\mathbf{x} \to \Theta)$$

- determine visible point

$$\mathbf{x} = r(\mathbf{e} \to -\Theta)$$

- look up emission

$$L_e(\mathbf{x} \to \Theta)$$



[Bala]

Basic Path Tracing

- need to evaluate

$$L_r(\mathbf{x} \to \Theta) = \int_{\Psi \in \Omega_{\mathbf{x}}} L(\mathbf{x} \leftarrow \Psi)\rho(\mathbf{x}, \Psi \to \Theta)\cos(\mathbf{N}_{\mathbf{x}}, \Psi)d\omega_{\Psi}$$

- use Monte Carlo estimation

$$L_r(\mathbf{x} \to \Theta) \approx \frac{1}{N}\sum_{i=1}^{N} \frac{L(\mathbf{x} \leftarrow \Psi_i)\rho(\mathbf{x}, \Psi_i \to \Theta)\cos(\mathbf{N}_{\mathbf{x}}, \Psi_i)}{p(\Psi_i)}$$

Basic Path Tracing

- generate random direction Ψ_i with $p(\Psi_i)$

$$L_r \approx \frac{1}{N}\sum_{i=1}^{N} \frac{L(\mathbf{x} \leftarrow \Psi_i)\rho(...)\cos(...)}{p(\Psi_i)}$$

  - evaluate BRDF
  - evaluate cosine
  - evaluate $L(\mathbf{x} \leftarrow \Psi_i)$


[Bala]

Basic Path Tracing

- need to evaluate

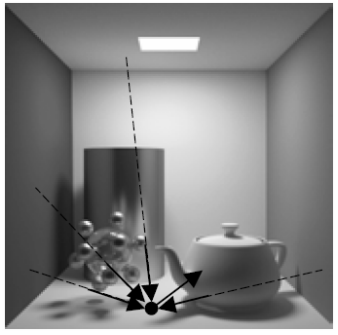
$$L(\mathbf{x} \leftarrow \Psi_i)$$

- determine visible point from x in direction Ψ_i

$$r(\mathbf{x} \to \Psi_i)$$

- in vacuum

$$L(\mathbf{x} \leftarrow \Psi_i) = L(r(\mathbf{x}, \Psi_i) \to \Psi_i)$$

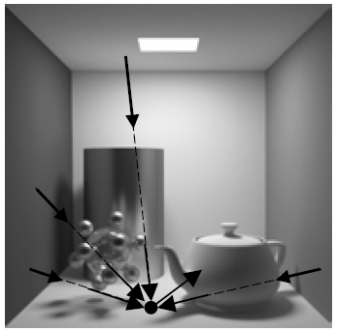

[Bala]

Basic Path Tracing

- need to evaluate

$$L(r(\mathbf{x}, \Psi_i) \rightarrow \Psi_i)$$

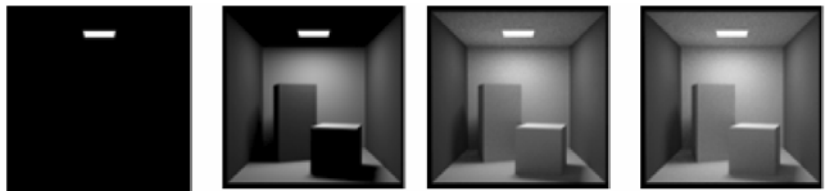
- recursively execute procedure



[Bala]

Russian Roulette

- when to stop recursion?
  - light bounce infinitely in the environment
  - but every bounce has less energy
    - in many cases 3-4 bounces are enough



[Bala]

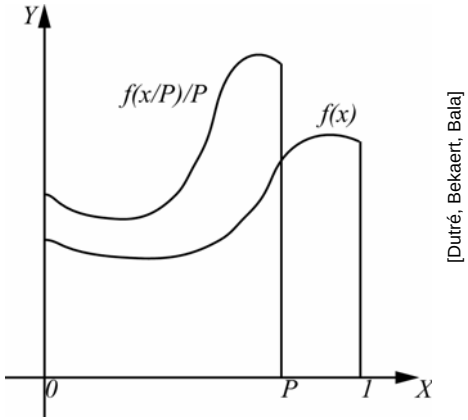
Russian Roulette

- stop after *k* bounces
  - introduces bias (consistent error) in the solution
  - need to make k large to capture all cases
- Monte Carlo strategy (Russian Roulette) as stopping criterion
  - pick a probability with which to stop the path
  - at each intersection, test the path
  - correct the radiance estimator accordingly

Russian Roulette

$$I = \int_0^1 f(x)dx = \int_0^1 \frac{f(x)}{P} P dx = \int_0^P \frac{f(y/P)}{P} dy \qquad P \leq 1$$

$$I_{RR} = \int_0^P \frac{f(y/P)}{P} dy = \int_0^P \frac{\bar{f}(y)}{P} dy$$



[Dutré, Bekaert, Bala]

Russian Roulette

$$y \sim 1 \qquad \bar{f}(y) = \begin{cases} \dfrac{f(y/P)}{P} & y \le P \\ 0 & y > P \end{cases}$$

$$E[\bar{f}(y)] = \int_0^1 \bar{f}(y)\,p(y)\,dy = \int_0^P \frac{f(y/P)}{P}\,dy = I$$

$$E[\bar{f}(x)] \approx \frac{1}{N}\sum_{i=1}^{N} \frac{\bar{f}(x_i)}{p(x_i)} \qquad \sigma[f] < \sigma[\bar{f}]$$

Russian Roulette

- if *f* is a recursive integral, only continue with probability $\alpha = 1 - P$
- but weight back the radiance $L_{RR} = L/(1-\alpha)$

- example: path tracing with $\alpha = 0.1$
  - only 1 chance in 10 the ray will continue
  - estimate radiance of the ray multiplies by 10
  - intuition: instead of shooting 10 rays, shoot 1 but weight its contribution 10 times

Russian Roulette

- how to choose the probability?
  - small fixed value: longer paths
    - slow but accurate
  - big fixed value: shorter paths
    - fast but inaccurate
  - proportional to integral of reflected light
    - adapts to material properties
    - darker patches will statistically shorten paths
    - exactly like in physics

Basic Path Tracing Pseudocode

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

computeRadiance(x, theta)
   estimatedRadiance = basicPT(x, theta)
   return estimatedRadiance
```

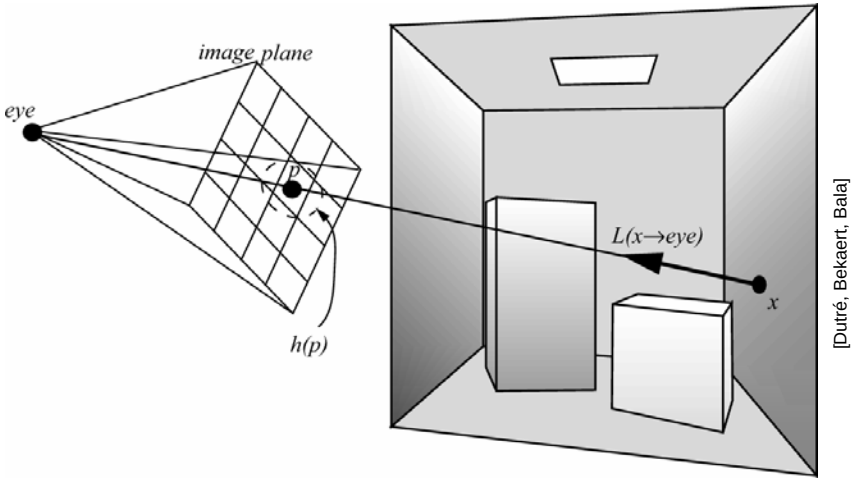
[Dutré, Bekaert, Bala]

Basic Path Tracing Pseudocode

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

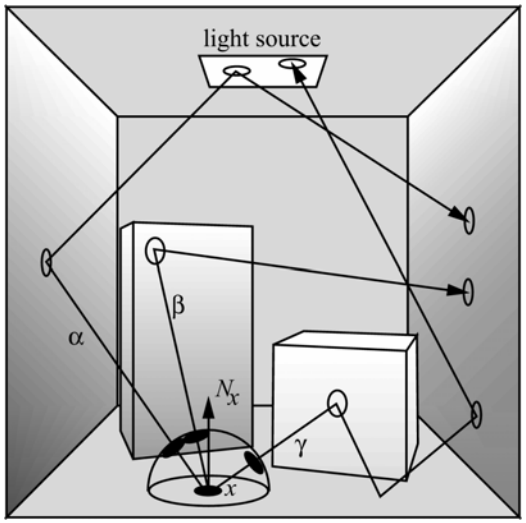
[Dutré, Bekaert, Bala]

Basic Path Tracing Intuition



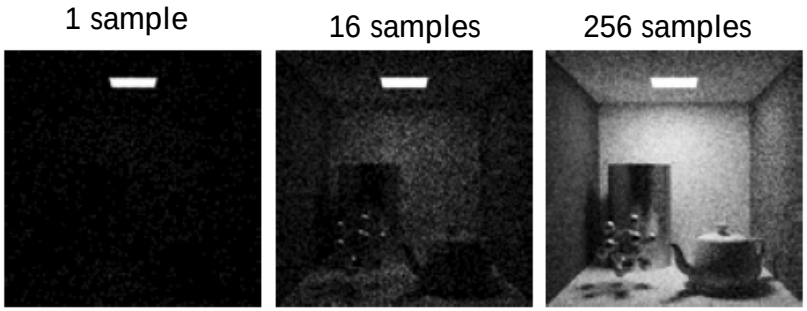
[Dutré, Bekaert, Bala]

Basic Path Tracing Intuition



[Dutré, Bekaert, Bala]

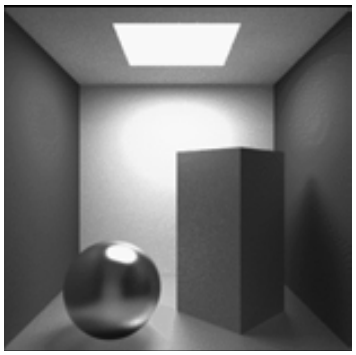
Basic Path Tracing Performance



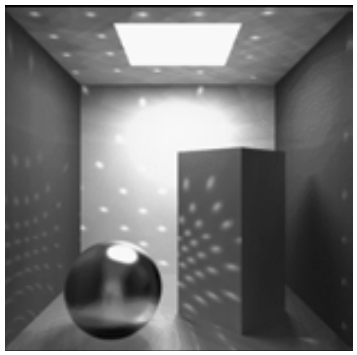
[Bala]

Monte Carlo vs. Deterministic Integration

Monte Carlo



Deterministic



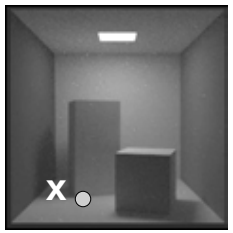
[Bala]

Next Event Estimation

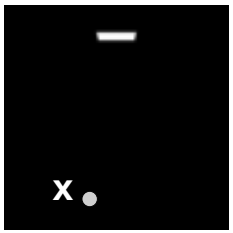
- in basic path tracing
  - if path does not hit a light, its radiance is 0
  - unlikely to hit a light by randomly picking dirs.
- next event estimation
  - want to directly sample light sources
  - by splitting direct and indirect illumination estimation
    - two separate Monte Carlo processes
  - by using area formulation for direct illumination
  - by using hemispherical formulation for indirect

Direct and Indirect Illum. Formulation

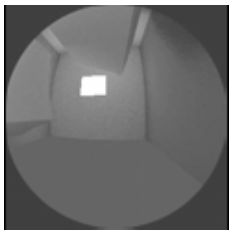
$$L(\mathbf{x} \to \Theta) = L_e(\mathbf{x} \to \Theta) + \int_{\Psi \in \Omega_x} L(\mathbf{x} \leftarrow \Psi)\rho(\mathbf{x}, \Psi \to \Theta)\cos\theta_\Psi d\omega_\Psi$$



$L(\mathbf{x} \to \Theta)$



$L_e(\mathbf{x} \to \Theta)$

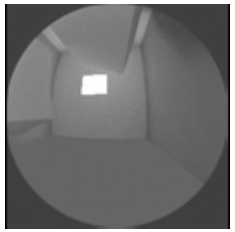


$L(\mathbf{x} \leftarrow \Psi)$

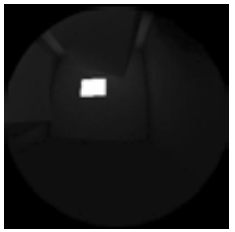
[Bala]

Direct and Indirect Illum. Formulation

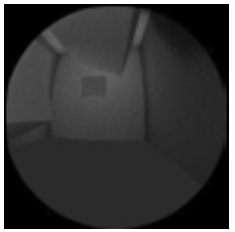
$$L(\mathbf{x} \leftarrow \Psi) = L_e(\mathbf{x} \leftarrow \Psi) + L_r(\mathbf{x} \leftarrow \Psi)$$



$L(\mathbf{x} \leftarrow \Psi)$



$L_e(\mathbf{x} \leftarrow \Psi)$



$L_r(\mathbf{x} \leftarrow \Psi)$

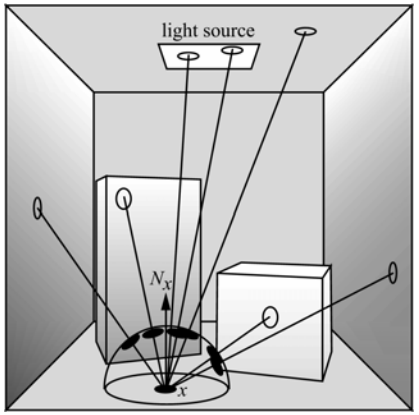
[Bala]

Direct and Indirect Illum. Formulation

$$L_r(\mathbf{x} \to \Theta) = \int_{\Psi \in \Omega_\mathbf{x}} L_e(\mathbf{x} \leftarrow \Psi)\rho(...)\cos\theta_\Psi d\omega_\Psi +$$
$$+ \int_{\Psi \in \Omega_\mathbf{x}} L_r(\mathbf{x} \leftarrow \Psi)\rho(...)\cos\theta_\Psi d\omega_\Psi =$$
$$= L_d(\mathbf{x} \to \Theta) + L_i(\mathbf{x} \to \Theta)$$

Direct Illum. - Hemisphere Sampling

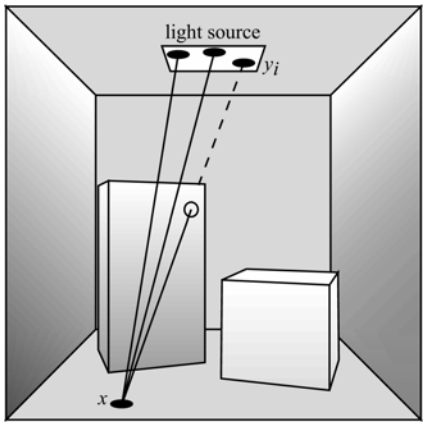
$$L_d(\mathbf{x} \to \Theta) = \int_{\Psi \in \Omega_\mathbf{x}} L_e(\mathbf{x} \leftarrow \Psi)\rho(...)\cos\theta_\Psi d\omega_\Psi$$



[Dutré, Bekaert, Bala]

Direct Illum. - Area Sampling

$$L_d(\mathbf{x} \to \Theta) = \int_{\mathbf{y} \in \text{lights}} L_e(\mathbf{y} \to \overrightarrow{\mathbf{yx}})\rho(...)G(\mathbf{x},\mathbf{y})V(\mathbf{x},\mathbf{y})dA_\mathbf{y}$$

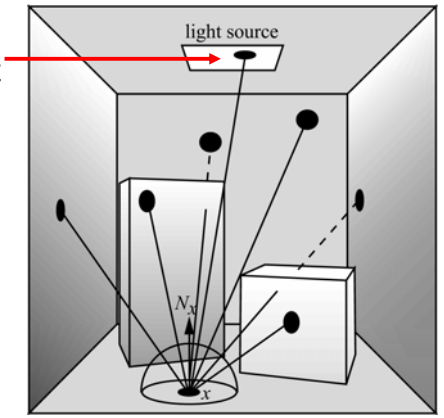


[Dutré, Bekaert, Bala]

Indirect Illum. - Hemisphere Sampling

$$L_d(\mathbf{x} \to \Theta) = \int_{\Psi \in \Omega_\mathbf{x}} L_r(\mathbf{x} \leftarrow \Psi)\rho(...)\cos\theta_\Psi d\omega_\Psi$$

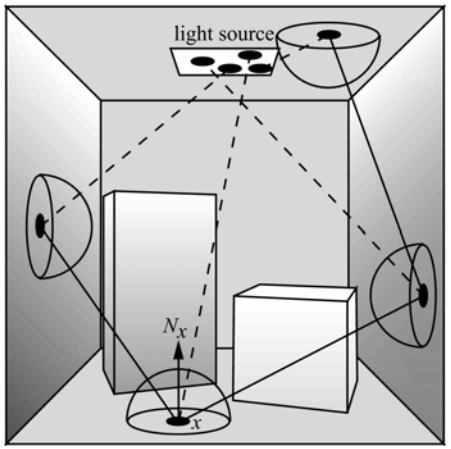
discard Le
if ray hits light



[Dutré, Bekaert, Bala]

Indirect Illum. – Recursive Evaluation

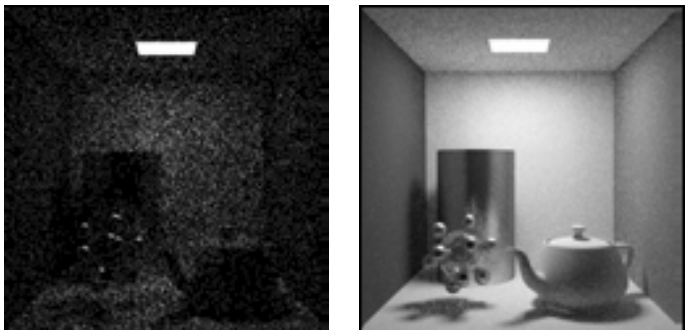
$$L_d(\mathbf{x} \to \Theta) = \int_{\Psi \in \Omega_\mathbf{x}} L_r(\mathbf{x} \leftarrow \Psi)\rho(...)\cos\theta_\Psi d\omega_\Psi$$



[Dutré, Bekaert, Bala]

Next Event Estimation Performance

without with

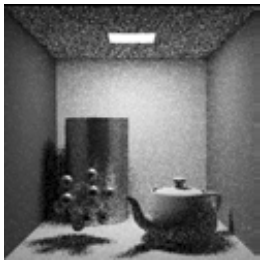


[Bala]

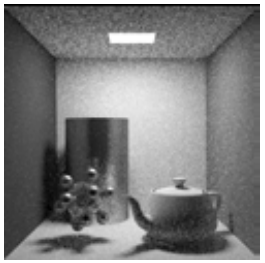
16 samples

Next Event Estimation Performance

1 sample



4 samples



[Bala]

16 samples

256 samples

Direct Illumination – One Light

$$L_d(\mathbf{x} \to \Theta) = \int_{\mathbf{y} \in \text{light}} L_e(\mathbf{y} \to \overrightarrow{\mathbf{yx}})\rho(...)G(\mathbf{x},\mathbf{y})V(\mathbf{x},\mathbf{y})dA_\mathbf{y}$$

$$L_d(\mathbf{x} \to \Theta) \approx \frac{1}{N}\sum_{i=1}^{N} \frac{L_e(\mathbf{y}_i \to \overrightarrow{\mathbf{y}_i\mathbf{x}})\rho(...)G(\mathbf{x},\mathbf{y}_i)V(\mathbf{x},\mathbf{y}_i)}{p(\mathbf{y}_i)}$$

- depends on
  - emitted radiance distribution Le
  - how to pick points y on the light
  - how many points to use
    - number of shadow rays

Direct Illumination – One Light

- each light type has its own physical models
  - angular distribution defines different light types
  - flood, fill, spot, ect…

- simplest model: emitted radiance is a constant

$$L_e(\mathbf{y} \to \Psi) = L_e$$

Direct Illumination – One Light

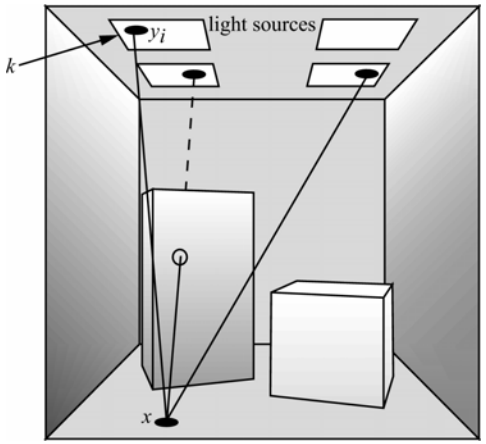
- uniform sampling of light area

$$p(\mathbf{y}) = \frac{1}{A_S}$$

  - simply sample that [0,1] square and rescale
  - works fairly well in practice
  - slightly better techniques exists tough

Direct Illumination – Many Lights

$$L_d(\mathbf{x} \to \Theta) = \int_{\mathbf{y} \in \text{lights}} L_e(\mathbf{y} \to \overrightarrow{\mathbf{yx}}) \rho(...) G(\mathbf{x},\mathbf{y}) V(\mathbf{x},\mathbf{y}) dA_{\mathbf{y}}$$



[Dutré, Bekaert, Bala]

Direct Illumination – Many Lights

$$L_d(\mathbf{x} \to \Theta) = \int_{\mathbf{y} \in \text{lights}} L_e(\mathbf{y} \to \overrightarrow{\mathbf{yx}}) \rho(...) G(\mathbf{x},\mathbf{y}) V(\mathbf{x},\mathbf{y}) dA_{\mathbf{y}}$$

- how to allocate samples between different lights
  - various techniques, some quite advanced

Direct Illumination – Many Lights

- split samples uniformly between lights
  - same as M light integrals with previous sampling

$$L_d(\mathbf{x} \to \Theta) = \sum_{j=1}^{M} \int_{\mathbf{y} \in \text{light}_j} L_e(\mathbf{y} \to \overline{\mathbf{yx}}) \rho(...) G(\mathbf{x}, \mathbf{y}) V(\mathbf{x}, \mathbf{y}) dA_{\mathbf{y}}$$

$$L_d(\mathbf{x} \to \Theta) = \sum_{j=1}^{M} \sum_{i=1}^{N/M} \frac{L_e(...) \rho(...) G(\mathbf{x}, \mathbf{y}_{ij}) V(\mathbf{x}, \mathbf{y}_{ij})}{p(\mathbf{y}_{ij})}$$

  - simple but inefficient
    - would like to weight more brighter lights
    - won't cover in this class

Indirect Illumination

$$L_d(\mathbf{x} \to \Theta) = \int_{\Psi \in \Omega_{\mathbf{x}}} L_r(\mathbf{x} \leftarrow \Psi) \rho(...) \cos\theta_\Psi d\omega_\Psi$$

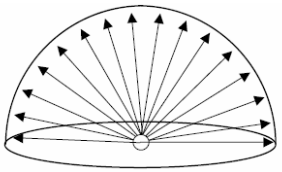
$$L_d(\mathbf{x} \to \Theta) \approx \frac{1}{N} \sum_{i=1}^{N} \frac{L_r(\mathbf{x} \leftarrow \Psi) \rho(...) \cos\theta_\Psi}{p(\Psi)}$$

- depends on how to sample the hemisphere
  - uniform distribution
  - importance sampling: pick p to match integral
    - cosine distribution
    - BRDF distribution
    - BRDF*cosine distribution

Indirect Illumination – Uniform Dist.

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

$$L_d(\mathbf{x} \to \Theta) \approx \frac{2\pi}{N} \sum_{i=1}^{N} L_r(\mathbf{x} \leftarrow \Psi) \rho(...) \cos\theta_\Psi$$

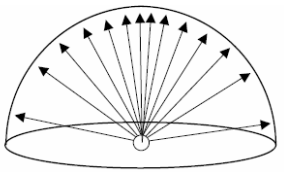


[Bala]

Indirect Illumination – Cosine Dist.

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

$$L_d(\mathbf{x} \to \Theta) \approx \frac{\pi}{N} \sum_{i=1}^{N} L_r(\mathbf{x} \leftarrow \Psi) \rho(...)$$

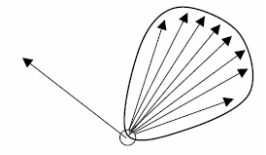


[Bala]

Indirect Illumination – BRDF Dist.

$$p(\Psi) \sim \rho(...)$$

$$L_d(\mathbf{x} \to \Theta) \sim \frac{1}{N} \sum_{i=1}^{N} L_r(\mathbf{x} \leftarrow \Psi) \cos\theta_\Psi$$

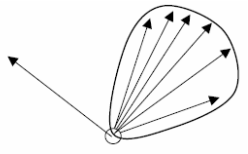


[Bala]

Indirect Illumination – BRDF*Cosine Dist.

$$p(\Psi) \sim \rho(...) \cos\theta_\Psi$$

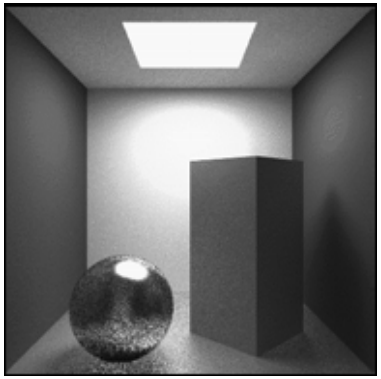
$$L_d(\mathbf{x} \to \Theta) \sim \frac{1}{N} \sum_{i=1}^{N} L_r(\mathbf{x} \leftarrow \Psi)$$



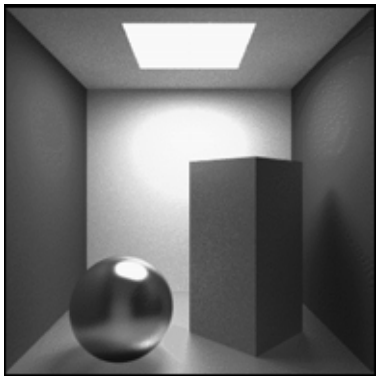
[Bala]

Importance Sampling Performance

without



with



[Bala]

PT Pseudocode – Pixel Sampling

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

[Dutré, Bekaert, Bala]

PT Pseudocode – Radiance Estimation

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

[Dutré, Bekaert, Bala]

PT Pseudocode – 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
```

[Dutré, Bekaert, Bala]

PT Pseudocode – Direct Illumination UL

```
directIllumination(x,theta)
   estimatedRadiance = 0
   for k = 1 to #lights
      for s = 1 to #shadowRays / #lights
         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
```

[Dutré, Bekaert, Bala]

PT Pseudocode – Indirect Illumination

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

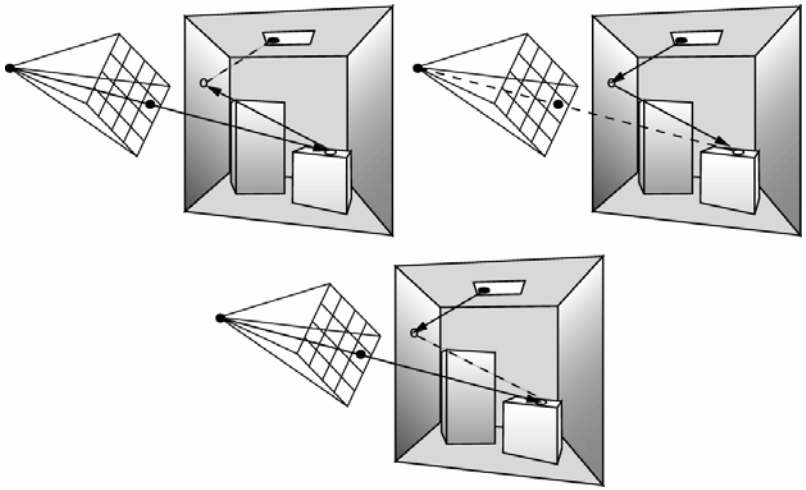
[Dutré, Bekaert, Bala]

Beyond Path Tracing
Bidirectional Techniues

Path Tracing

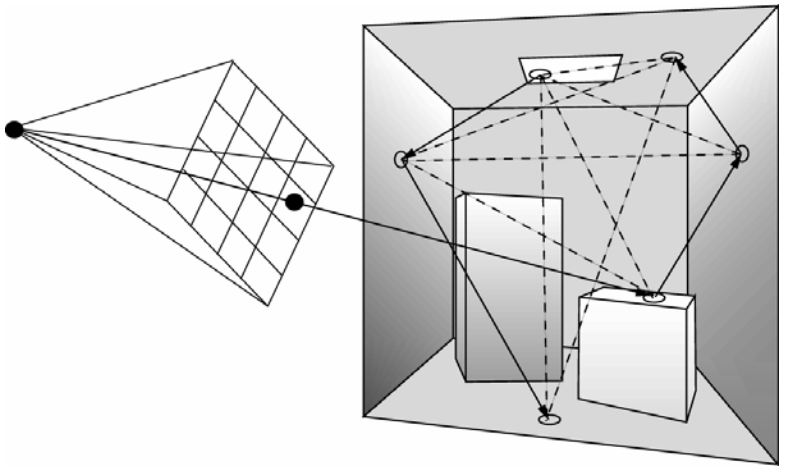
- perfectly accurate, but slow to converge
  - noise remains in the image for a long time
- intuition: is there are bright reflections, we cannot sample them directly
  - halogen lamps
- idea: shoot paths from the eye and from the light
  - eye paths: work well for reflections
  - light paths: pick up secondary sources
  - in reality very complex

Bidirectional Path Tracing



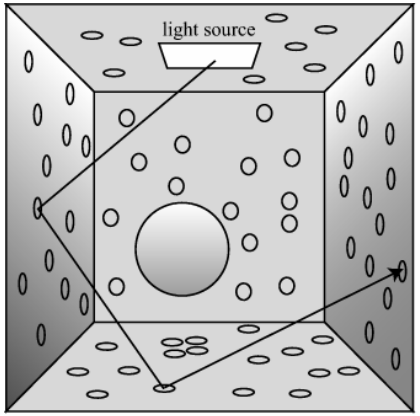
[Dutré, Bekaert, Bala]

Bidirectional Path Tracing



[Dutré, Bekaert, Bala]

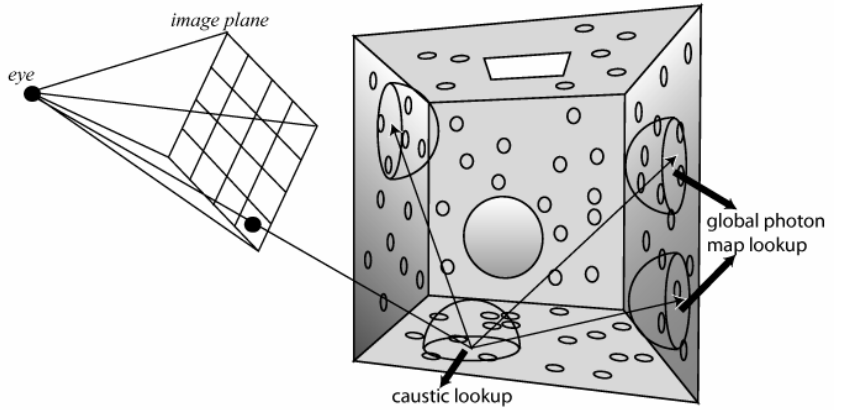
Photon Mapping



Pass 1: Shoot Photons

[Dutré, Bekaert, Bala]

Photon Mapping



Pass 2: Find Nearest Neighbors

[Dutré, Bekaert, Bala]