Global Illumination

Adapted from…

Thomas Funkhouser
Princeton University
C0S 526, Fall 2002

Additional material from…

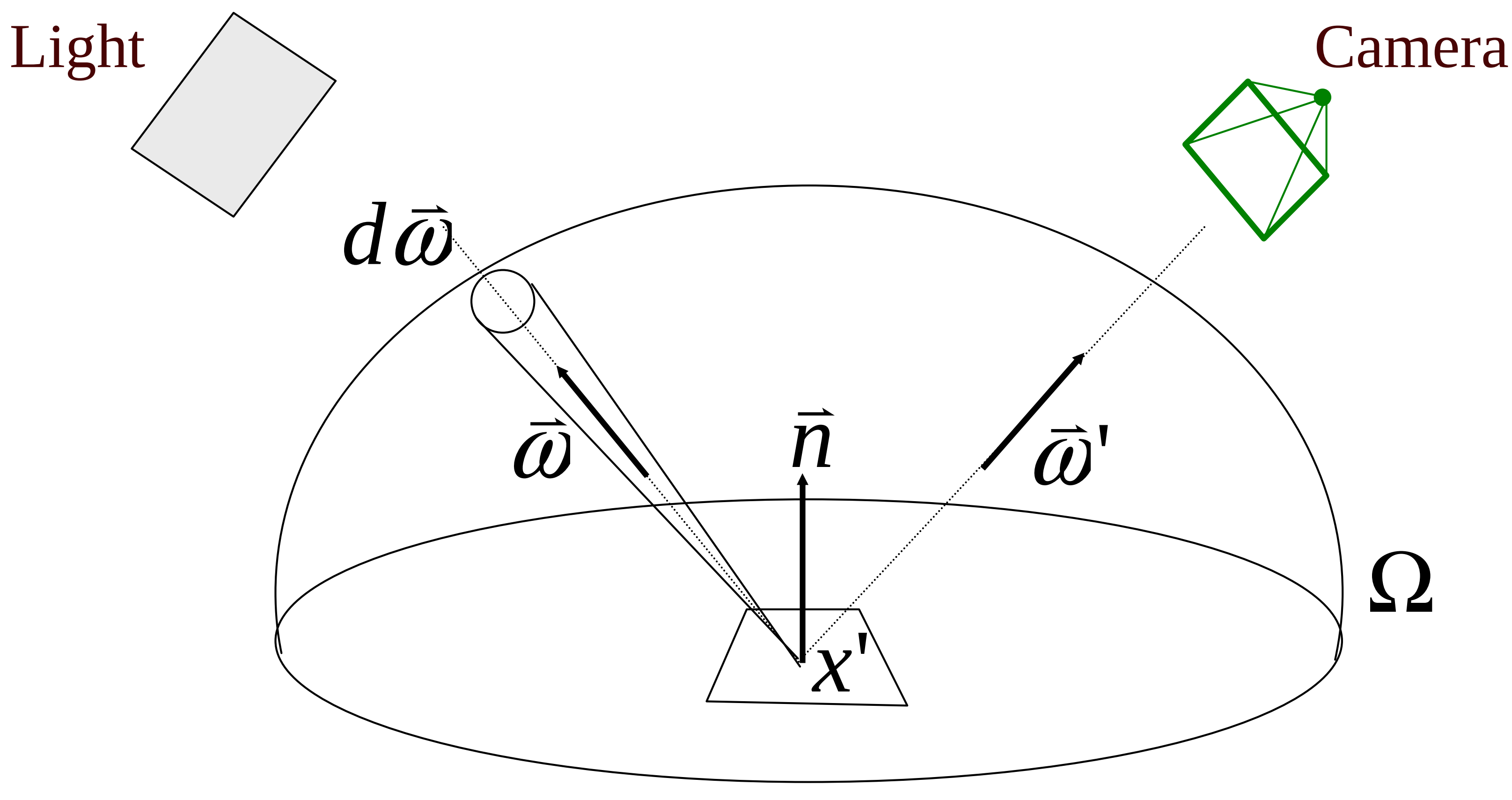
H.W. Jensen, Realistic Image Synthesis Using Photon Mapping, A.K. Peters Ltd., 2001

Overview

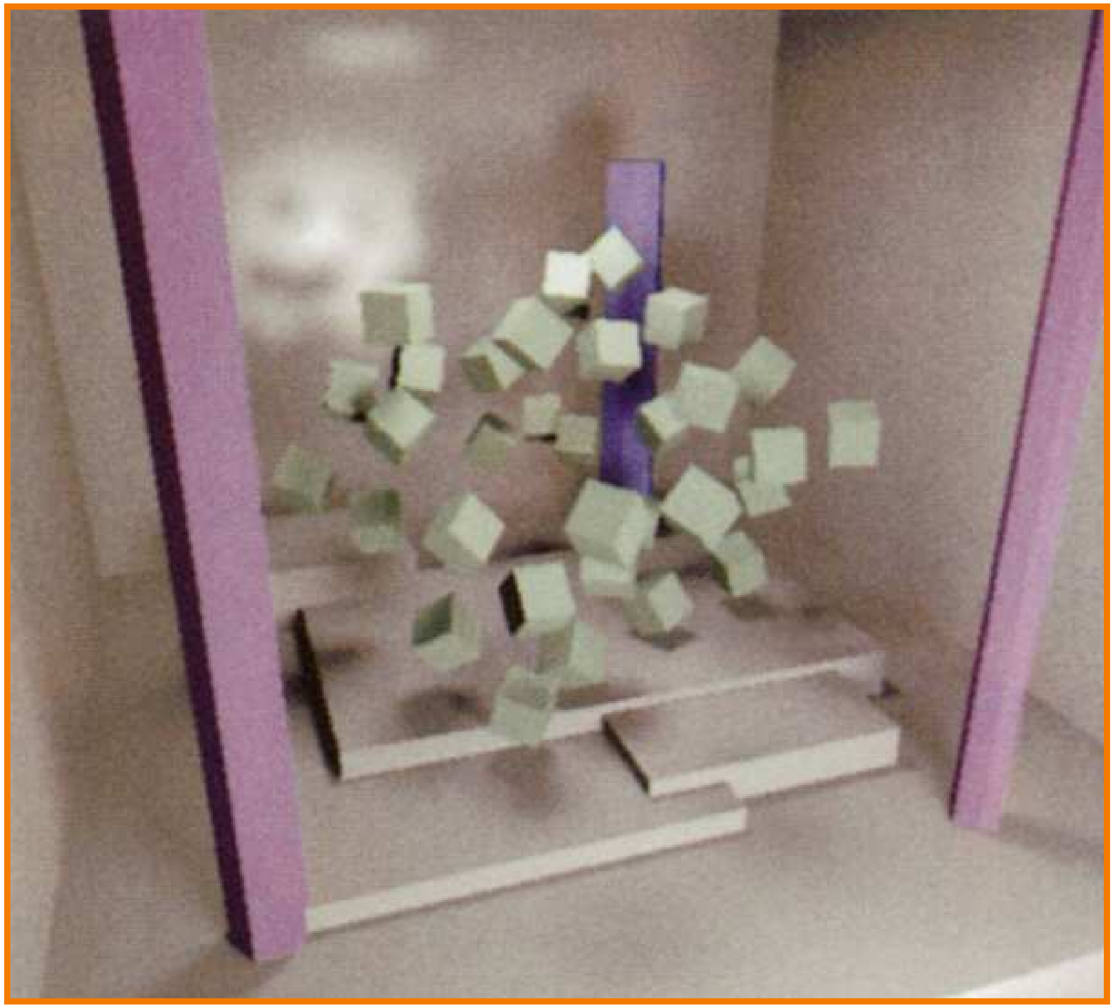
- Global illumination
- Rendering equation
- Overview of solution methods

Direct Illumination

$$L_o(x',\vec{\omega}') = L_e(x',\vec{\omega}') + \int_{\Omega_L} f_r(x',\vec{\omega},\vec{\omega}')L_i(x',\vec{\omega})(\vec{\omega}\bullet\vec{n})d\vec{\omega}$$

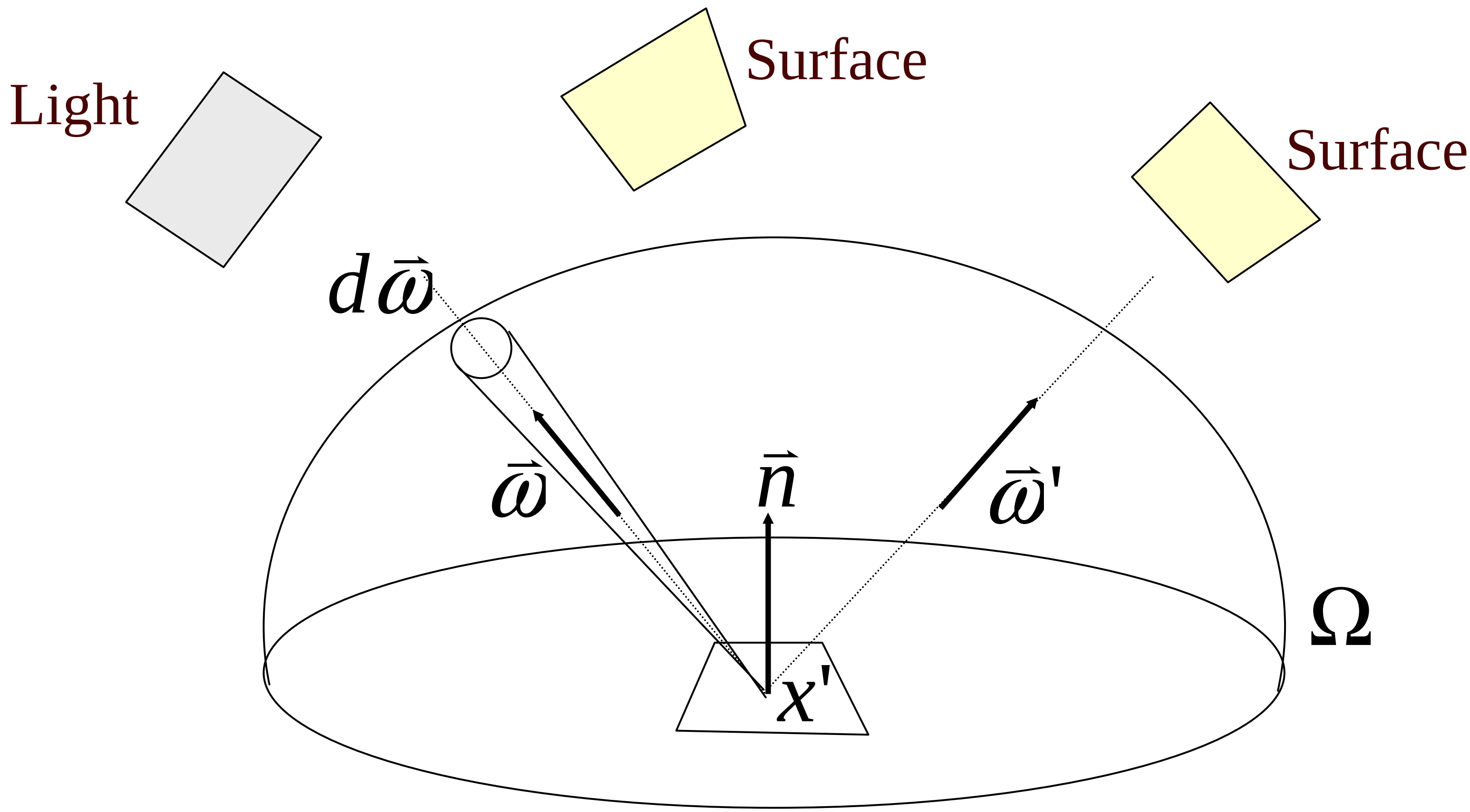


Direct Illumination

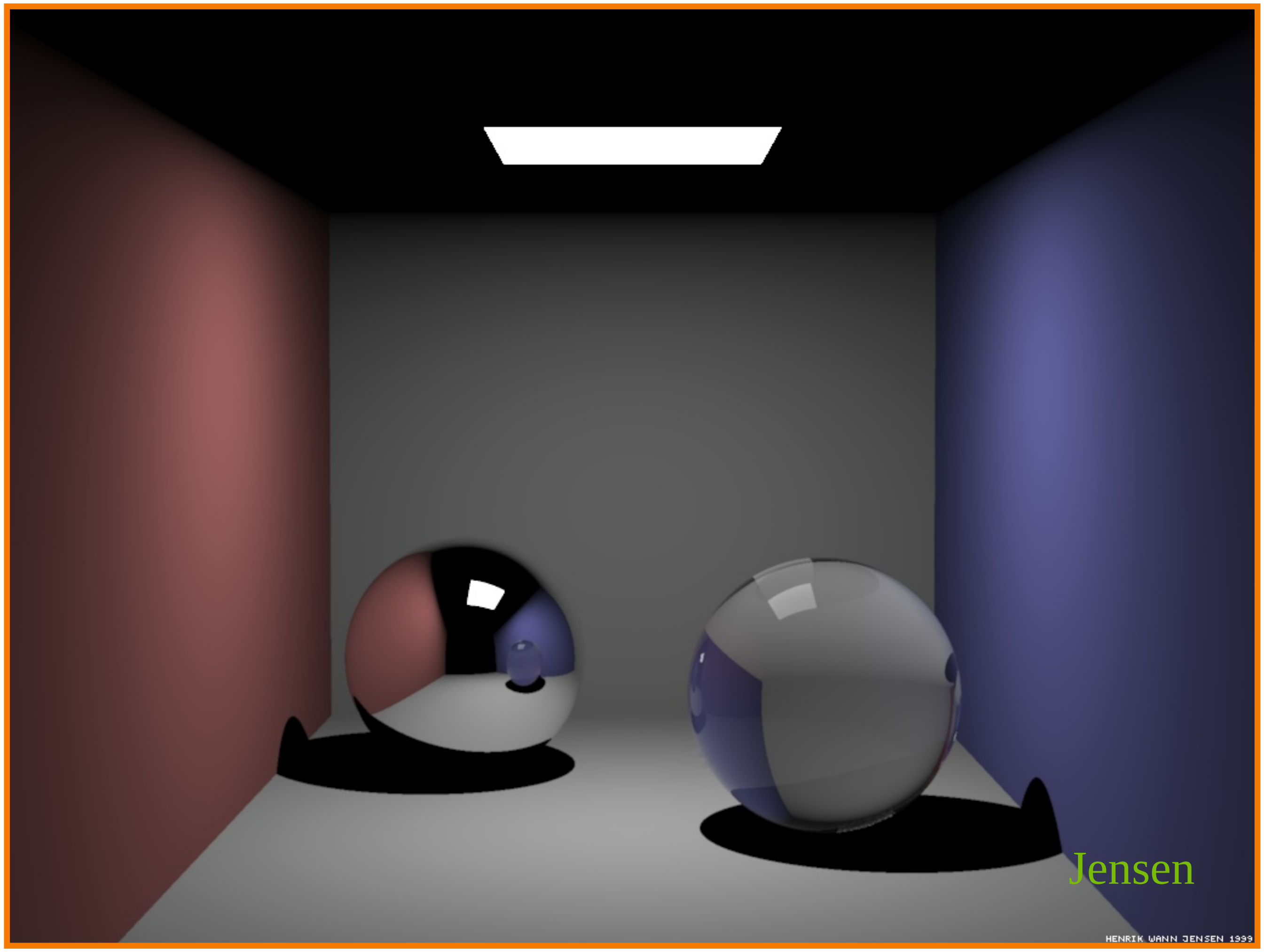


Global Illumination

$$L_o(x', \vec{\omega}') = L_e(x', \vec{\omega}') + \int_{\Omega} f_r(x', \vec{\omega}, \vec{\omega}') L_i(x', \vec{\omega})(\vec{\omega} \bullet \vec{n}) d\vec{\omega}$$



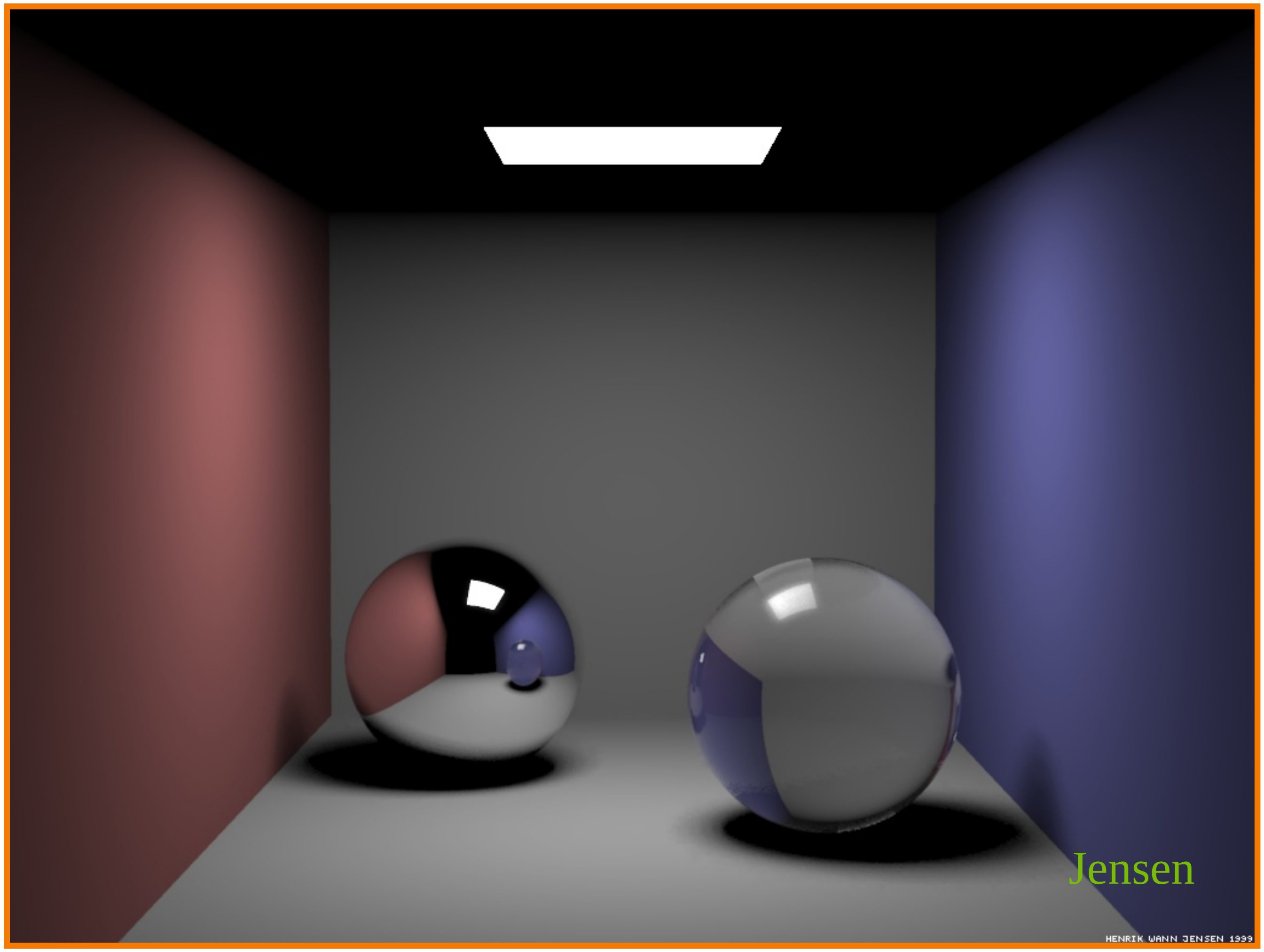
Global Illumination



Ray tracing

Henrik Wann Jensen

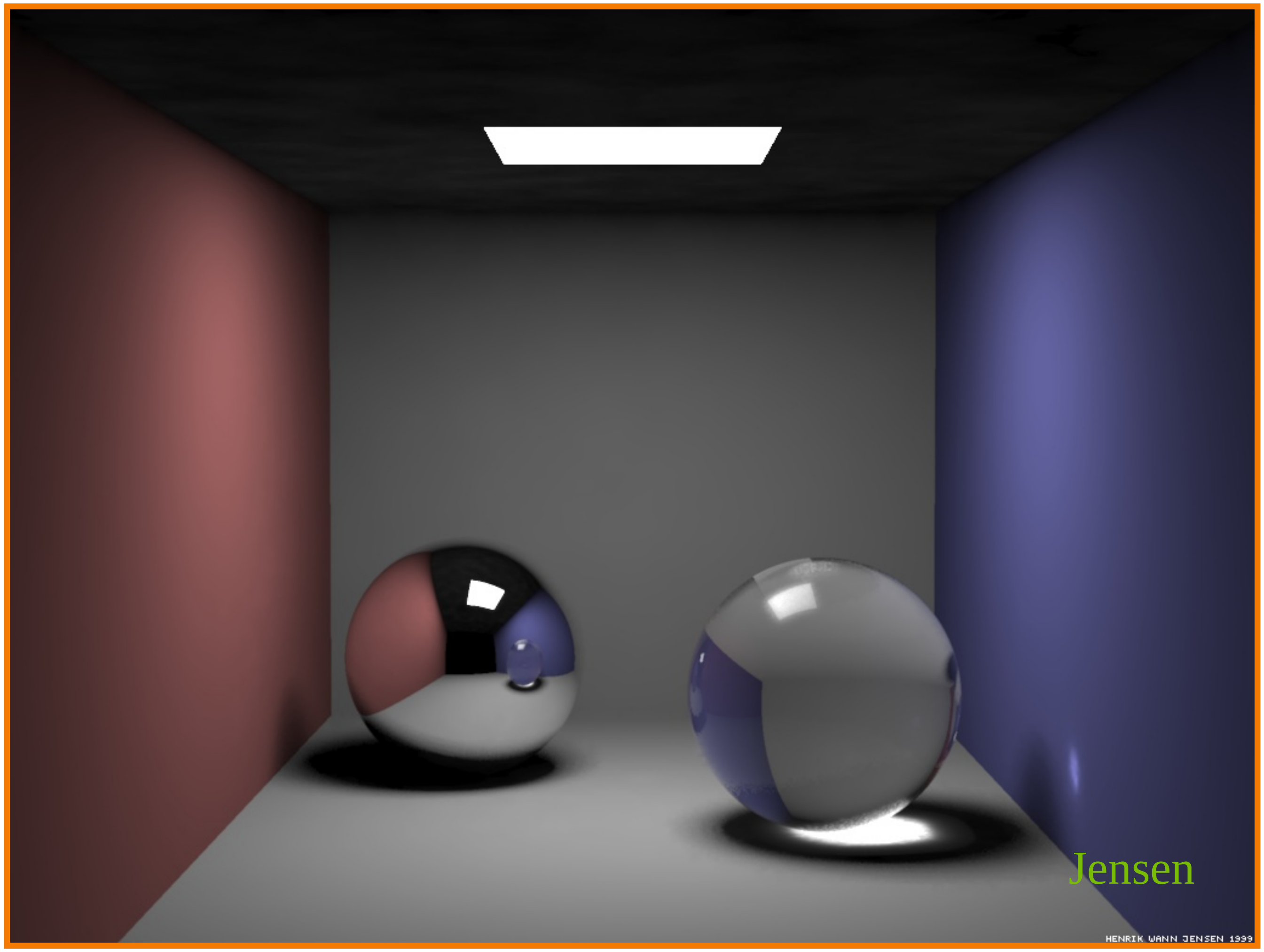
Global Illumination



+ soft shadows

Henrik Wann Jensen

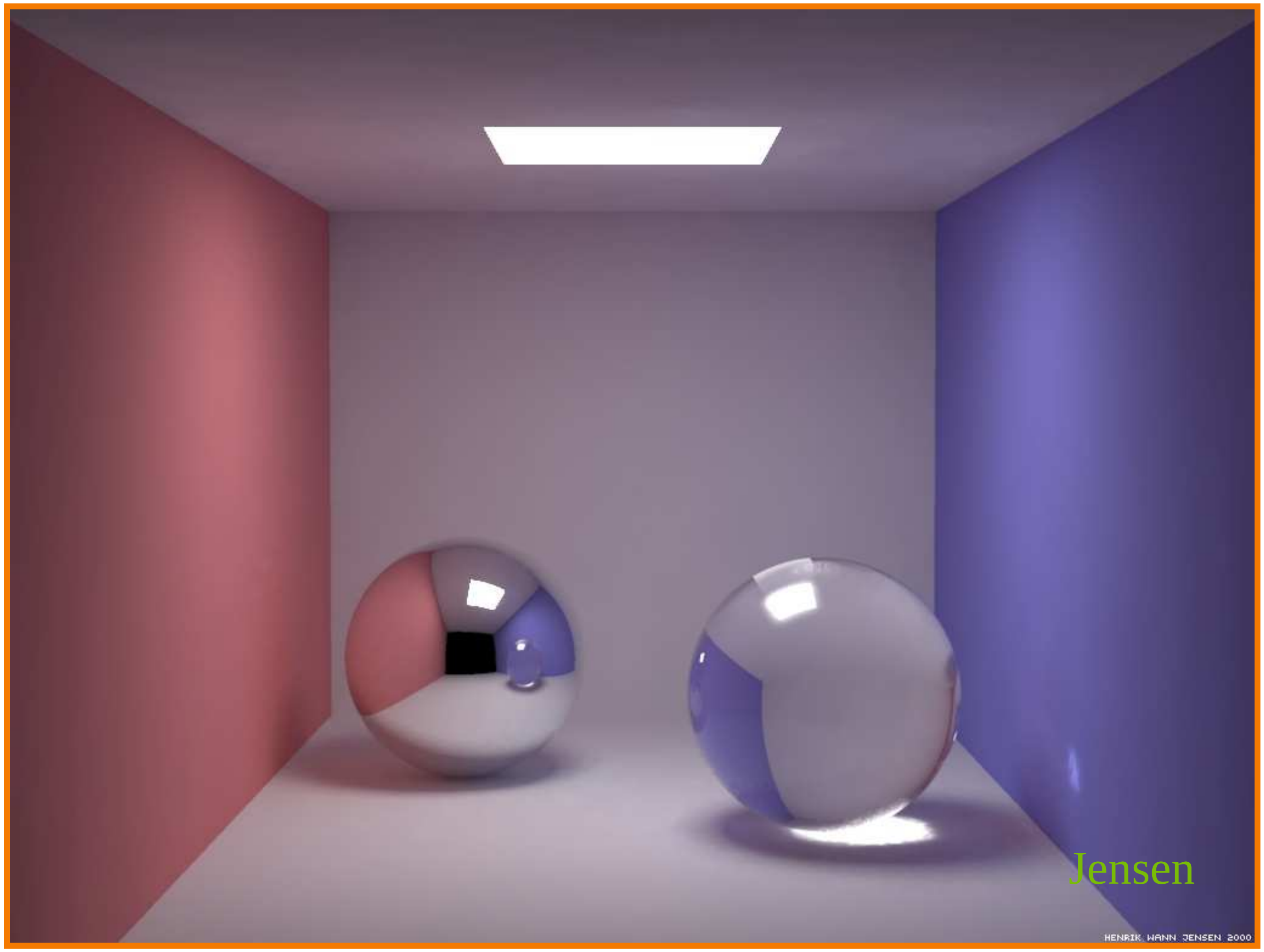
Global Illumination



+ caustics

Henrik Wann Jensen

Global Illumination

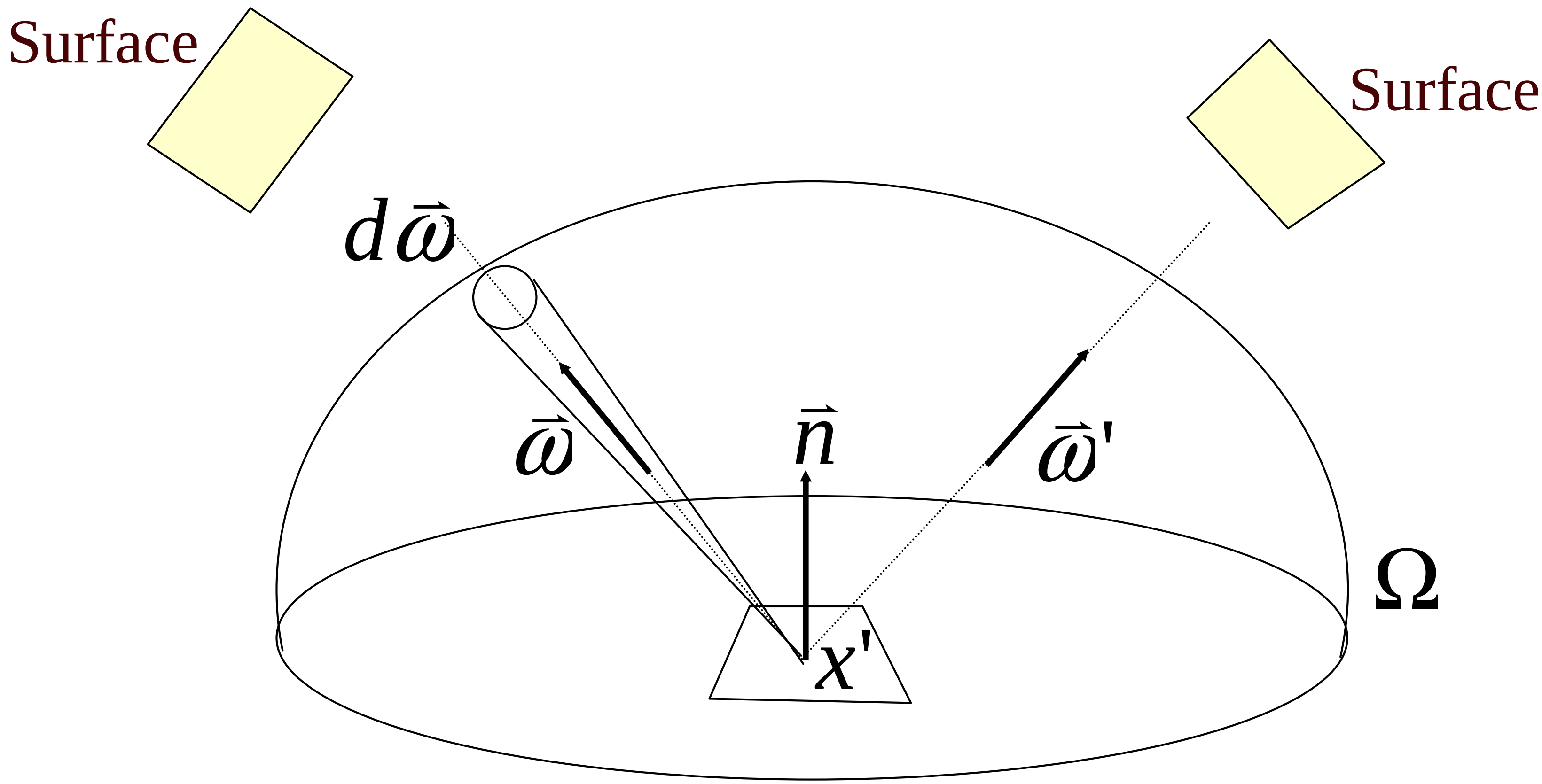


+ indirect diffuse illumination

Henrik Wann Jensen

Rendering Equation

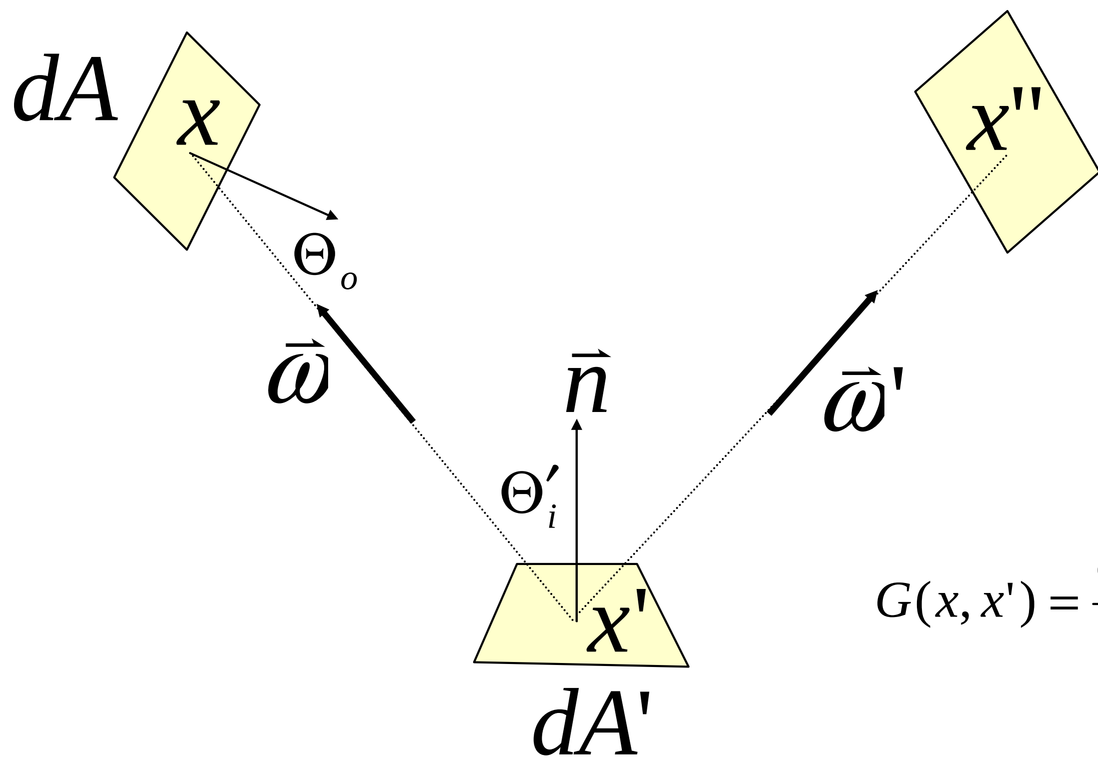
$$L_o(x',\vec{\omega}') = L_e(x',\vec{\omega}') + \int_{\Omega} f_r(x',\vec{\omega},\vec{\omega}')L_i(x',\vec{\omega})(\vec{\omega}\bullet\vec{n})d\vec{\omega}$$



Kajiya 1986

Rendering Equation (2)

$$L(x' \to x'') = L_e(x' \to x'') + \int_S f_r(x \to x' \to x'') L(x \to x') V(x,x') G(x,x') dA$$



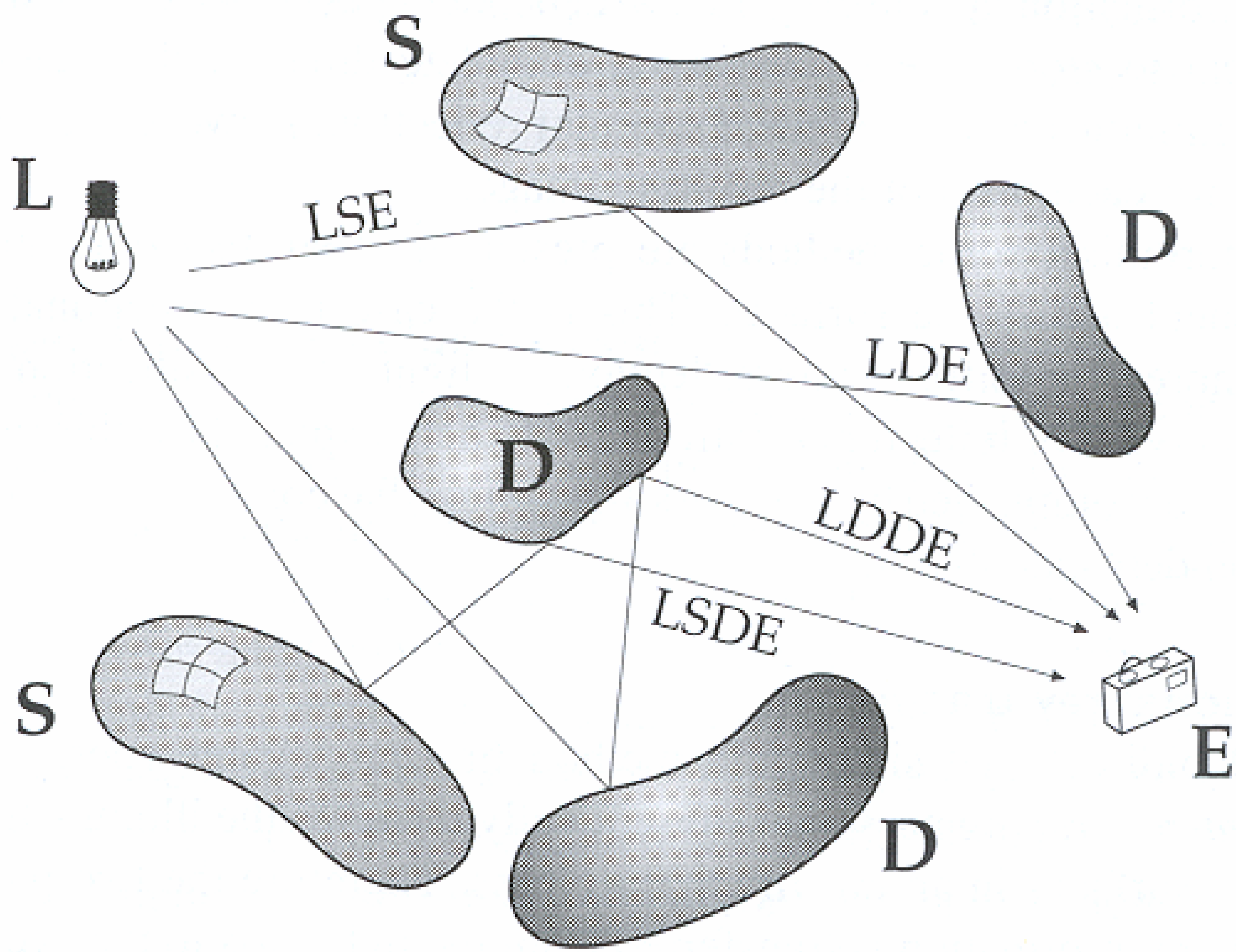
$$G(x,x') = \frac{\cos\Theta_i' \cos\Theta_o}{\left\| x - x' \right\|^2}$$

Kajiya 1986

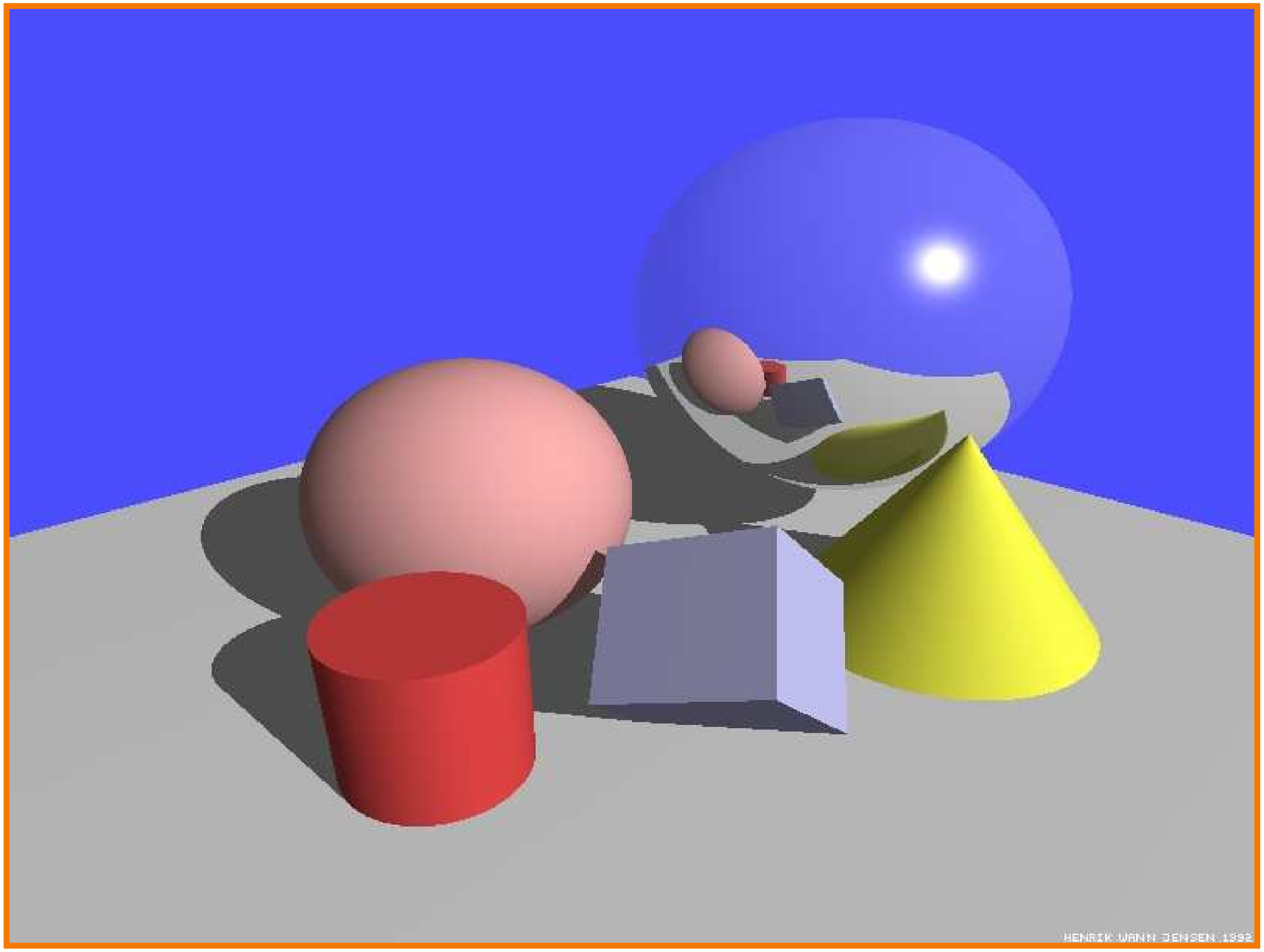
Solution Methods

- OpenGL
- Radiosity
- Ray tracing
- Distribution ray tracing
- Path tracing

Path Types



Path Types?

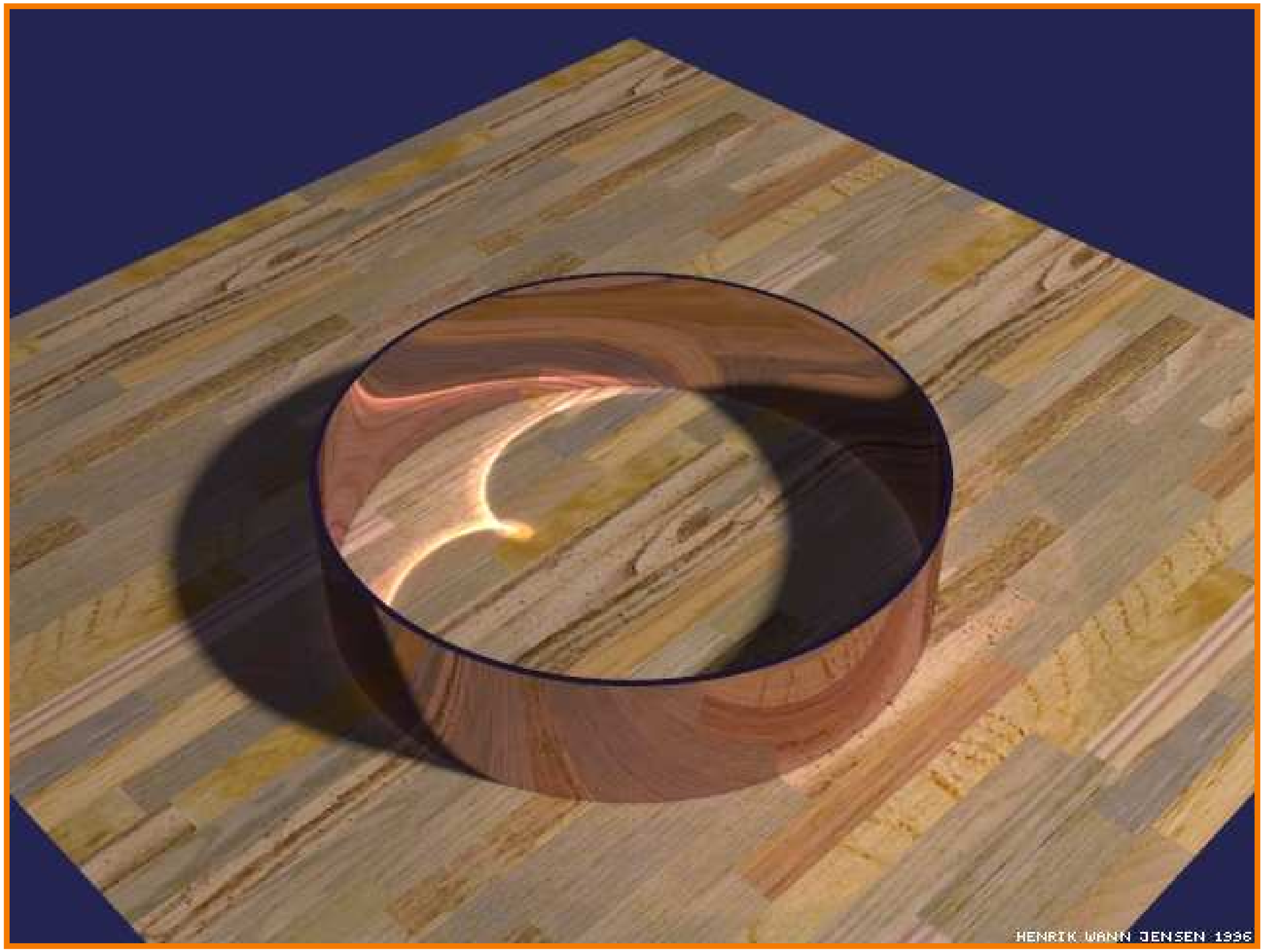


Henrik Wann Jensen

Path Types?

Paul Debevec

Path Types?



Henrik Wann Jensen

Path Types?

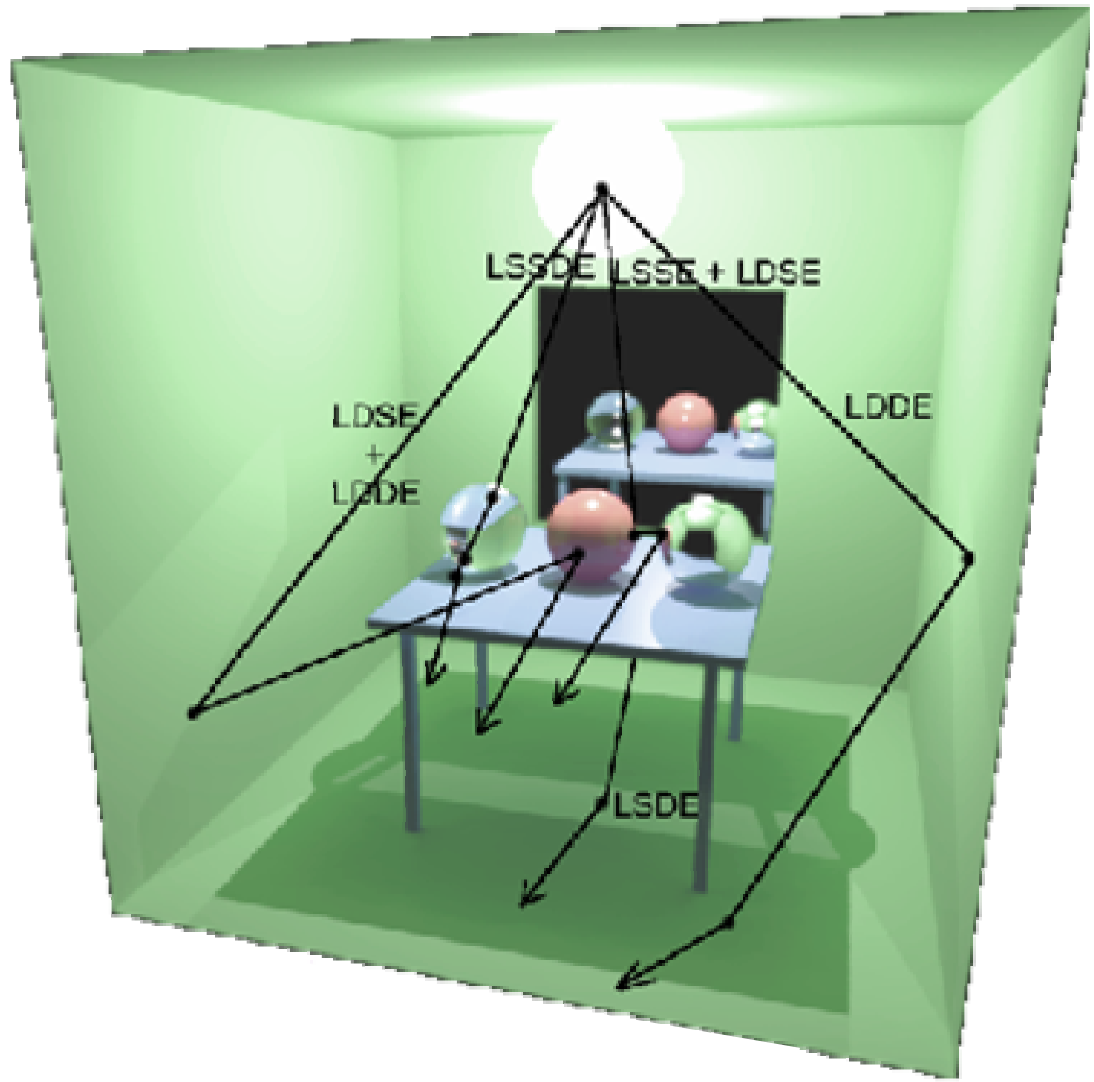
RenderPark

Path Type Notation

- Introduced by [Heckbert, 1990]
- Vertices of the light path can be:
  - L – a light source
  - E – the eye
  - S – a specular reflection
  - D – a diffuse reflection
- Combinations of paths:
  - (k)+ one or more of k events
  - (k)* zero or more of k events
  - (k)? zero or one k event
  - (k|k') a k or a k' event
- Examples:
  - Radiosity: LD*E
  - Ray Tracing: LD?S*E
  - Path Tracing: L(D|S)*E
  - Caustics: LS+DE

Path Types

- OpenGL
  - LDE
- Ray tracing
  - LD?S*E
- Radiosity
  - LD*E
- Path tracing
  - L(D|S)*E

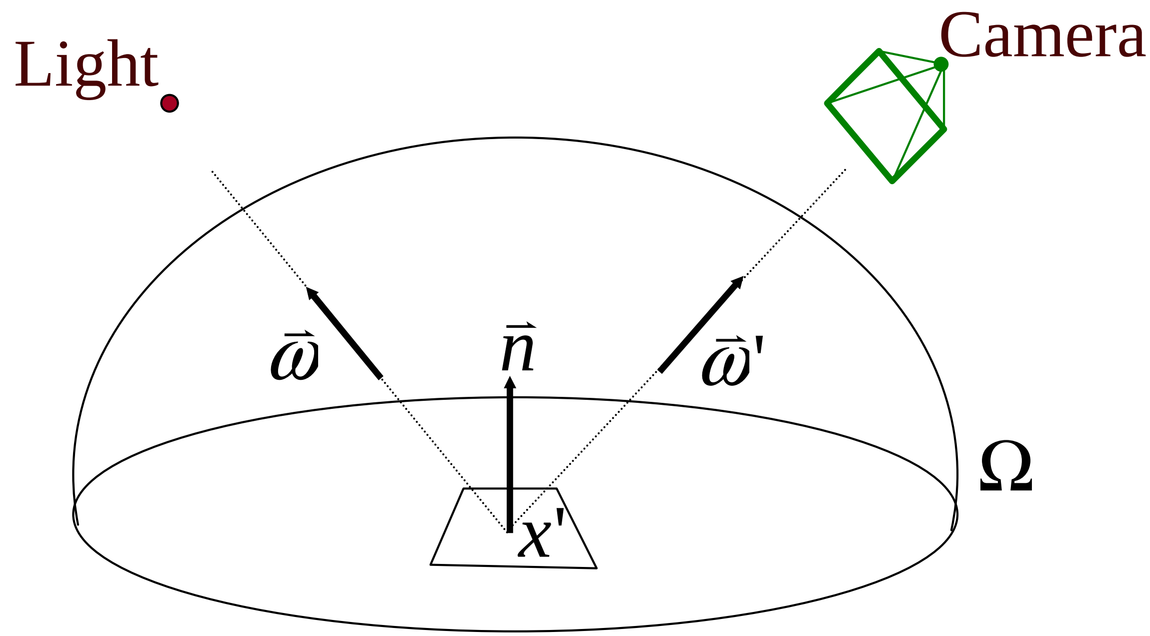


John Hart

OpenGL

$$L_o(x',\vec{\omega}') = L_e(x',\vec{\omega}') + \int_{\Omega} f_r(x',\vec{\omega},\vec{\omega}') L_i(x',\vec{\omega})(\vec{\omega} \bullet \vec{n}) d\vec{\omega}$$

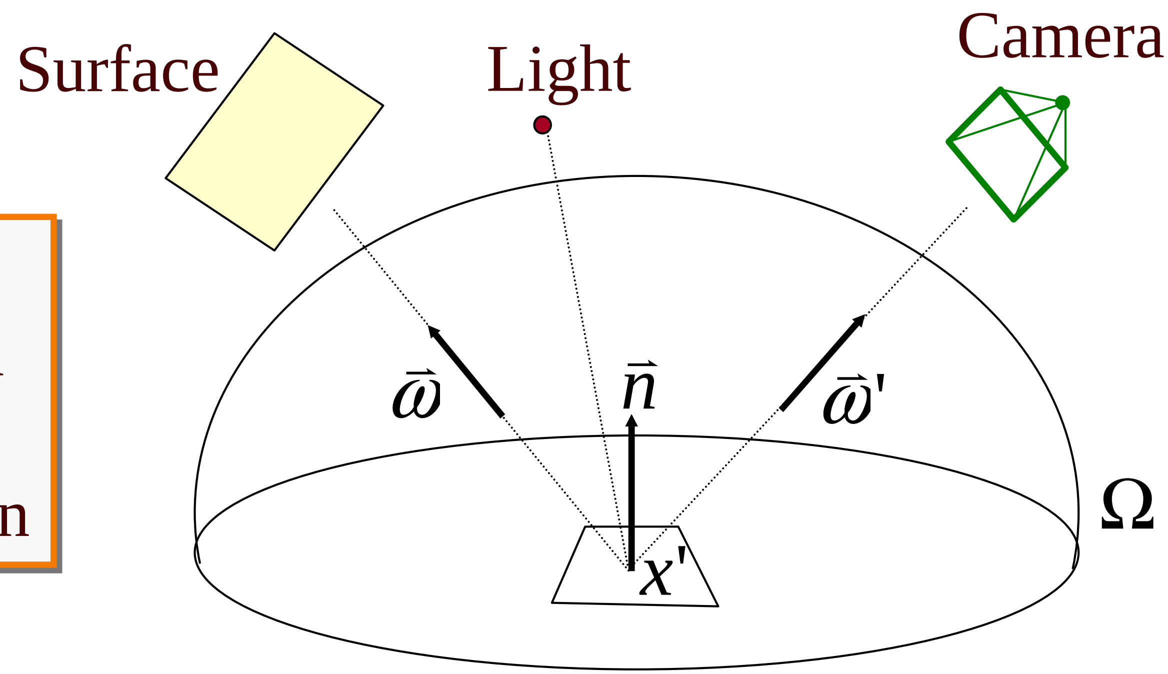
Assume
direct illumination
from point lights
and ignore visibility



$$L_o(x',\vec{\omega}') = L_e(x',\vec{\omega}') + \sum_{i=1}^{nlights} f_r(x',\vec{\omega},\vec{\omega}') L_i(x',\vec{\omega})(\vec{\omega} \bullet \vec{n})$$

Ray Tracing

$$L_o(x',\vec{\omega}') = L_e(x',\vec{\omega}') + \int_{\Omega} f_r(x',\vec{\omega},\vec{\omega}')L_i(x',\vec{\omega})(\vec{\omega}\bullet\vec{n})d\vec{\omega}$$



Assume
specular reflection
is only significant
indirect illumination

$$L_o(x',\vec{\omega}') = L_e(x',\vec{\omega}') + \sum_{i=1}^{nlights} f_r(x',\vec{\omega},\vec{\omega}')L_i(x',\vec{\omega})(\vec{\omega}\bullet\vec{n}) + specular$$

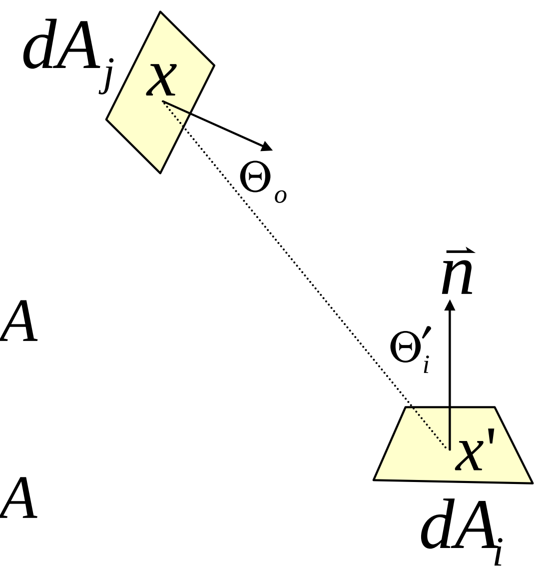
Ray Tracing Algorithm

```
render image using ray tracing
  for each pixel
    pick a ray from the eye through this pixel
    pixel color = trace(ray)

trace( ray )
  find nearest intersection with scene
  compute intersection point and normal
  color = shade( point, normal )
  return color

shade( point, normal )
  color = 0
  for each light source
    trace shadow ray to light source
      if shadow ray intersects light source
        color = color + direct illumination
  if specular
    color = color + trace( reflected/refracted ray )
  return color
```

Radiosity

$$L(x' \to x'') = L_e(x' \to x'') + \int_S f_r(x \to x' \to x'') L(x \to x') V(x,x') G(x,x') dA$$

Assume everything is Lambertian



$$B(x') = B_e(x') + \int_S f_{r,d}(x') B(x) V(x,x') G(x,x') dA$$

$$B(x') = B_e(x') + \frac{\rho_d(x')}{\pi} \int_S B(x) V(x,x') G(x,x') dA$$

$$B_i = B_{e,i} + \rho_i \sum_{j=1}^{N} B_j F_{ij}$$ where $$F_{ij} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{V(x,x') G(x,x')}{\pi} dA_j dA_i$$

Path Tracing

$$L_o(x',\vec{\omega}') = L_e(x',\vec{\omega}') + \int_{\Omega} f_r(x',\vec{\omega},\vec{\omega}')L_i(x',\vec{\omega})(\vec{\omega}\bullet\vec{n})d\vec{\omega}$$

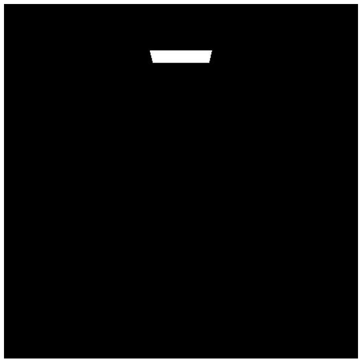
Perform Neumann series expansion

$$L = L_e + TL \quad \text{where} \quad T(x',\vec{\omega}')g = \int_{\Omega} f_r(x',\vec{\omega},\vec{\omega}')g(x',\vec{\omega})(\vec{\omega}\bullet\vec{n})d\vec{\omega}$$

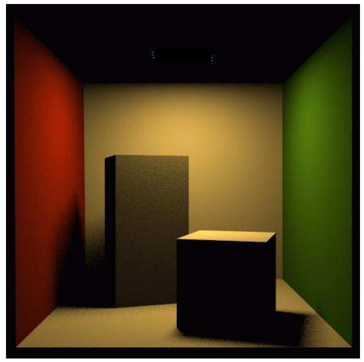
$$L = L_e + TL_e + T^2L_e + T^3L_e + \ldots$$

- Convergent approximation
- Also suggested by [Kajiya, 1986]

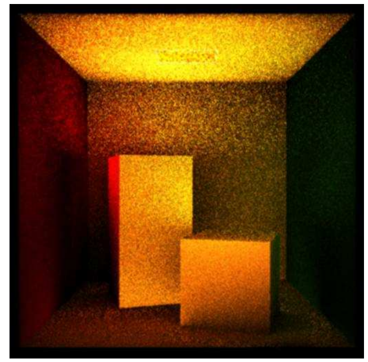
Path Tracing



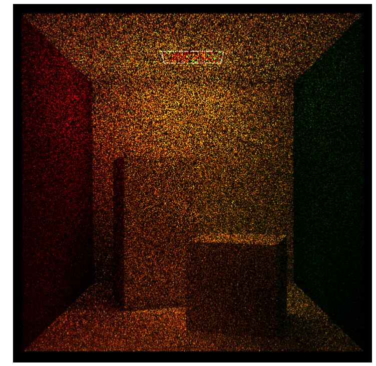
L_e



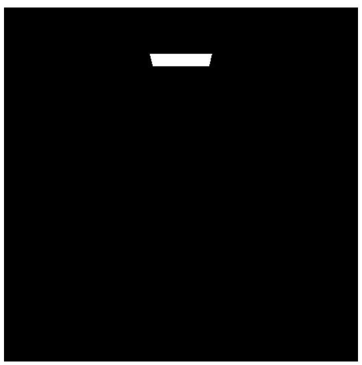
TL_e



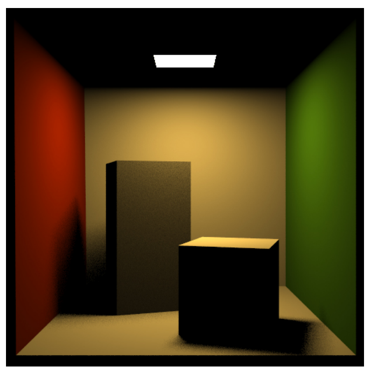
T^2L_e



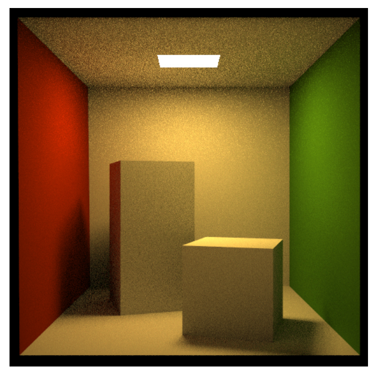
T^3L_e



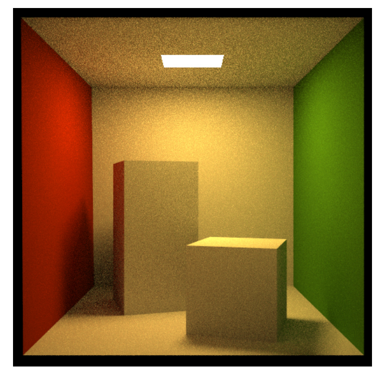
L_e



$L_e + TL_e$



$L_e + TL_e + T^2L_e$



$L_e + \ldots + T^3L_e$

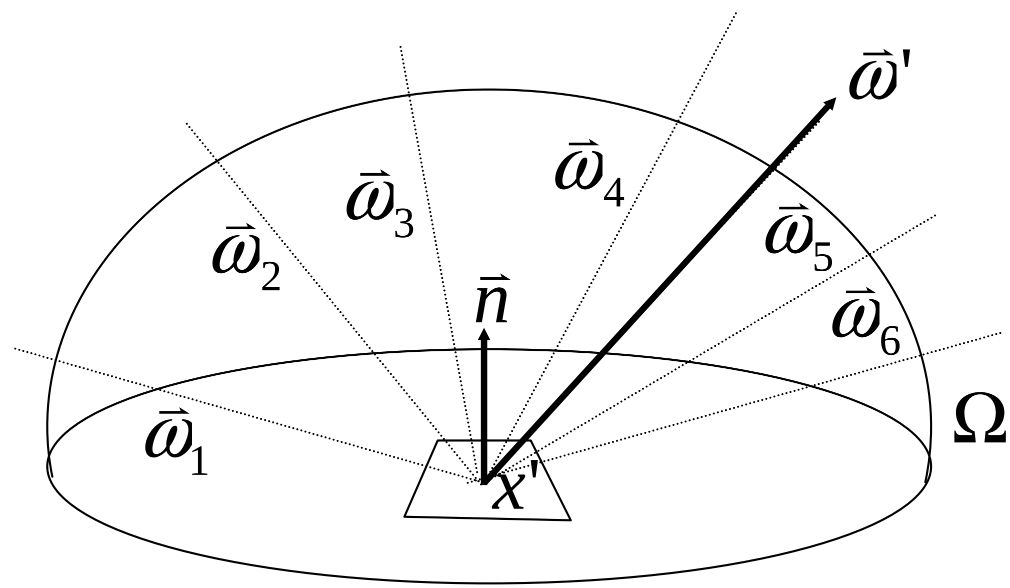
Path Tracing Algorithm

```
render image using path tracing
  for each pixel
    color = 0
    for each sample
      pick ray from observer through random position in pixel
      pick a random time and lens position for the ray
      color = color + trace( ray )
    pixel-color = color/#samples

trace( ray )
  find nearest intersection with scene
  compute intersection point and normal
  color = shade( point, normal )
  return color

shade( point, normal )
  color = 0
  for each light source
    test visibility of random position on light source
    if visible
      color = color + direct illumination
  color = color + trace( a randomly reflected ray )
  return color
```

Distribution Ray Tracing

$$L_o(x', \vec{\omega}') = L_e(x', \vec{\omega}') + \int_{\Omega} f_r(x', \vec{\omega}, \vec{\omega}') L_i(x', \vec{\omega})(\vec{\omega} \bullet \vec{n}) d\vec{\omega}$$

Estimate integral
for each reflection
by random sampling



Also:

- Depth of field
- Motion blur
- etc.

Distribution Ray Tracing

- Random direction $\vec{\omega}_d$ is computed as follows.
- Given two uniformly distributed random variables,

$\xi_1 \in [0,1]$ and $\xi_2 \in [0,1]$ we find that this randomly reflected direction, $\vec{\omega}_d$, is distributed as:

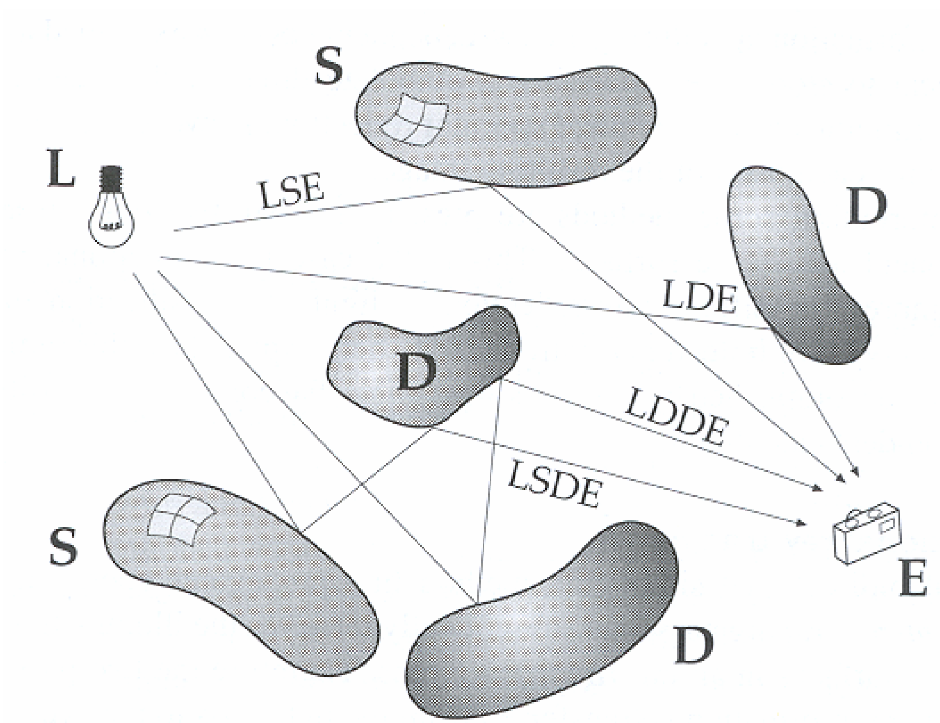
$$\vec{\omega}_d = (\theta, \phi) = (cos^{-1}(\sqrt{\xi_1}), 2\pi\xi_2) \ , \tag{2.24}$$

where we have used spherical coordinates (θ, ϕ) for the direction: θ is the angle with the surface normal, and ϕ is the rotation around the normal.

Monte Carlo Path Tracing

$$L_o(x',\vec{\omega}') = L_e(x',\vec{\omega}') + \int_{\Omega} f_r(x',\vec{\omega},\vec{\omega}')L_i(x',\vec{\omega})(\vec{\omega}\bullet\vec{n})d\vec{\omega}$$

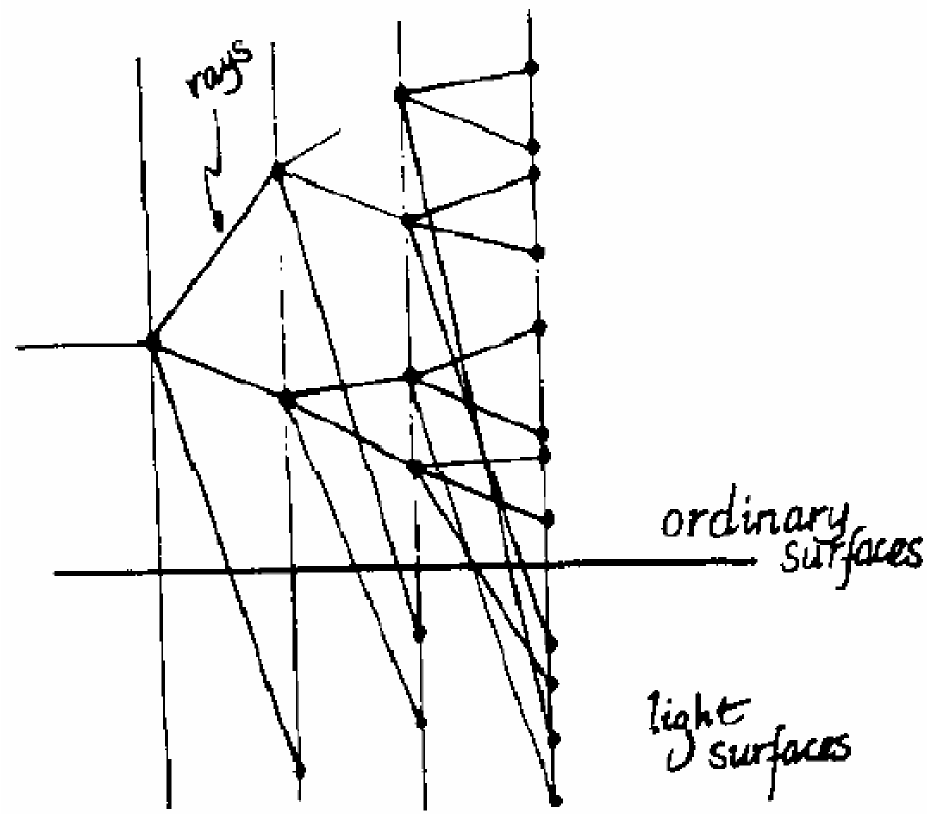
Estimate integral
for each pixel
by random sampling



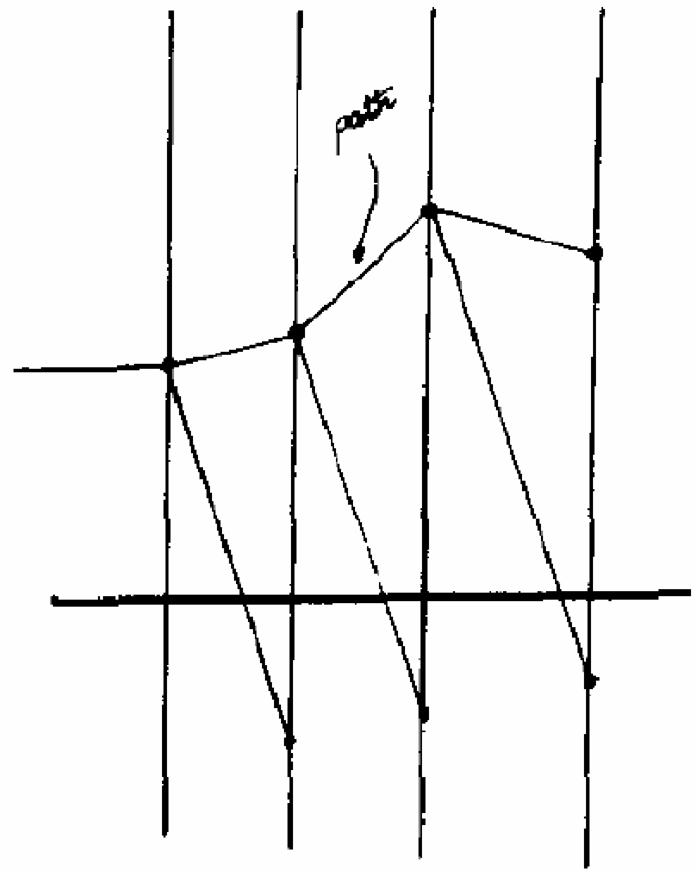
Also:

- Depth of field
- Motion blur
- etc.

Ray Tracing vs. Path Tracing



Ray tracing

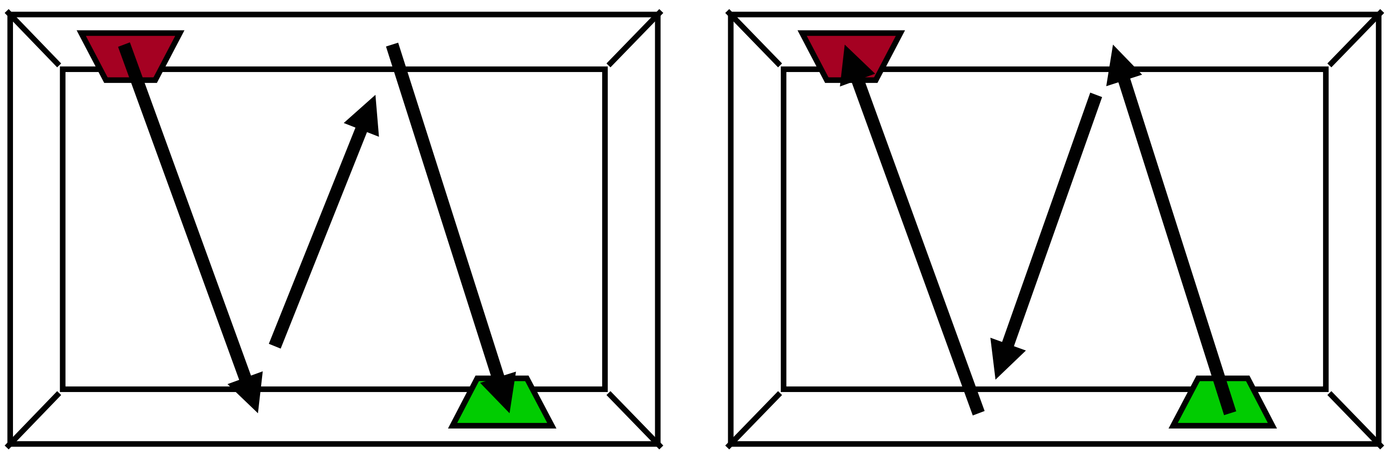


Path tracing

Jim Kajiya

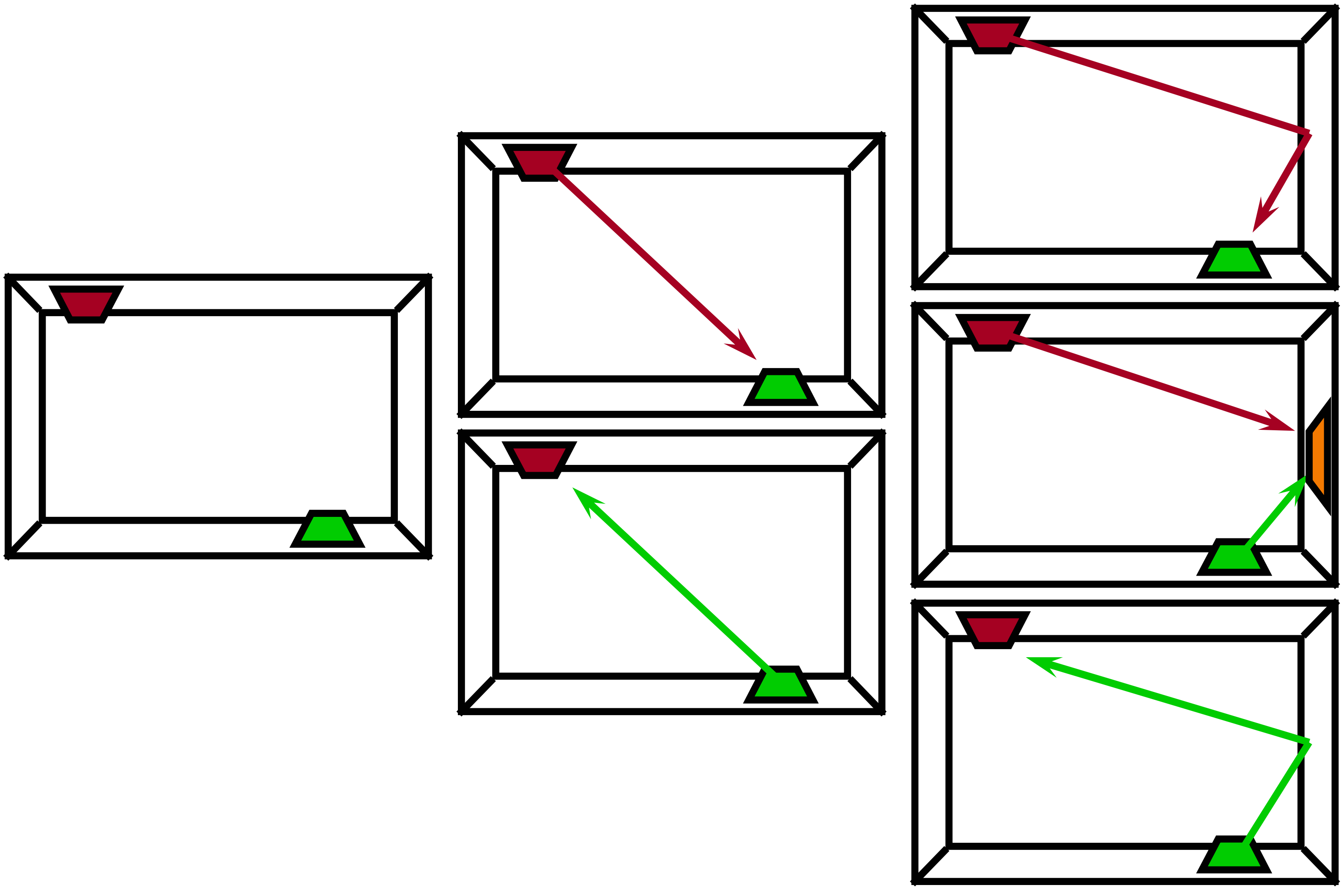
Bidirectional Path Tracing

- Role of source and receiver can be switched, flux does not change
- Exploiting duality can increase convergence rate



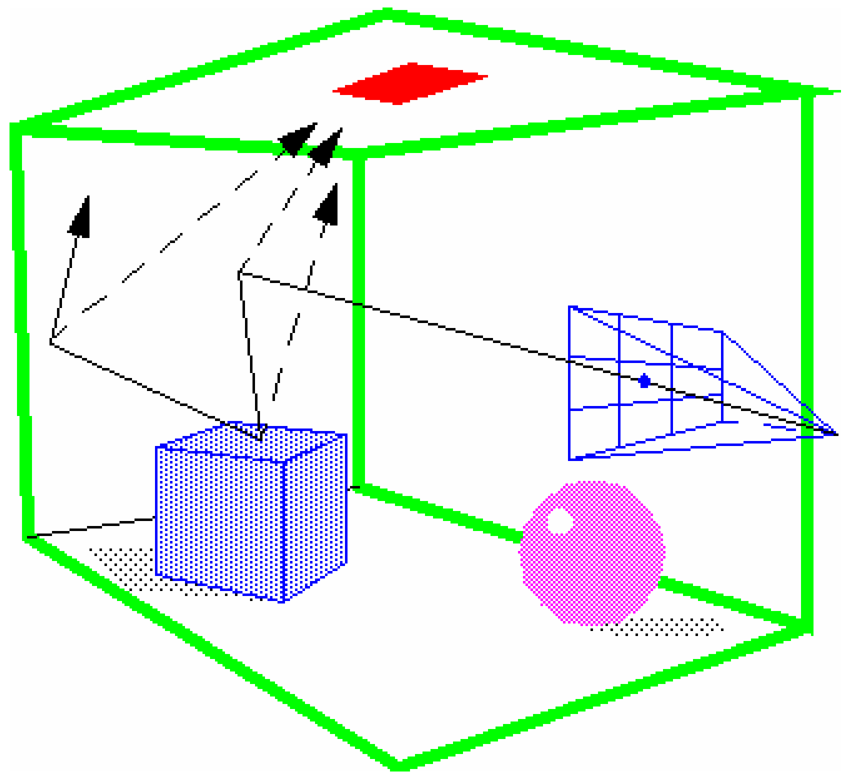
Philip Dutré

Bidirectional Path Tracing

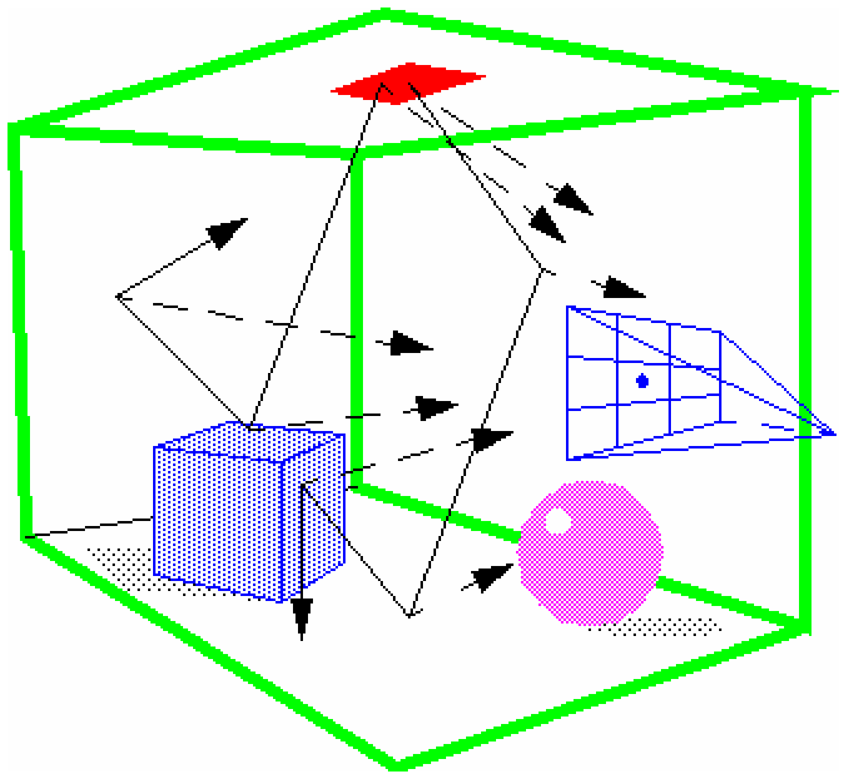


Philip Dutré

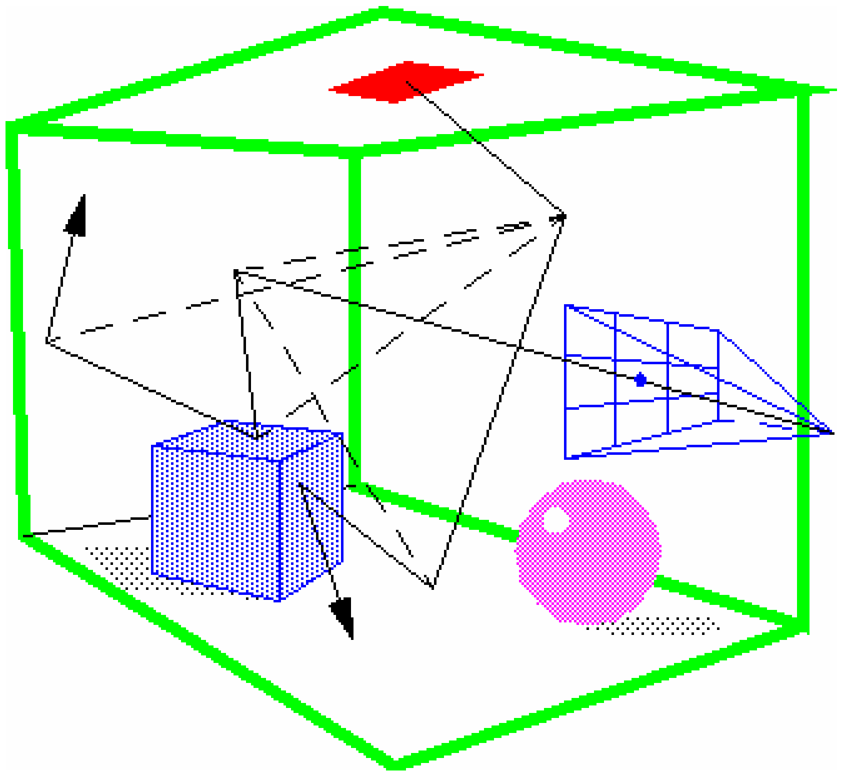
Tracing From Eye



Tracing from Lights



Bidirectional Path Tracing

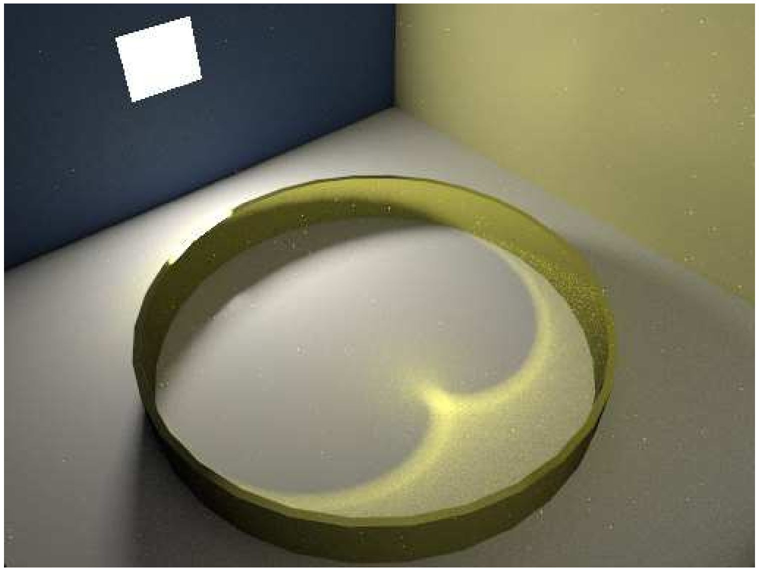
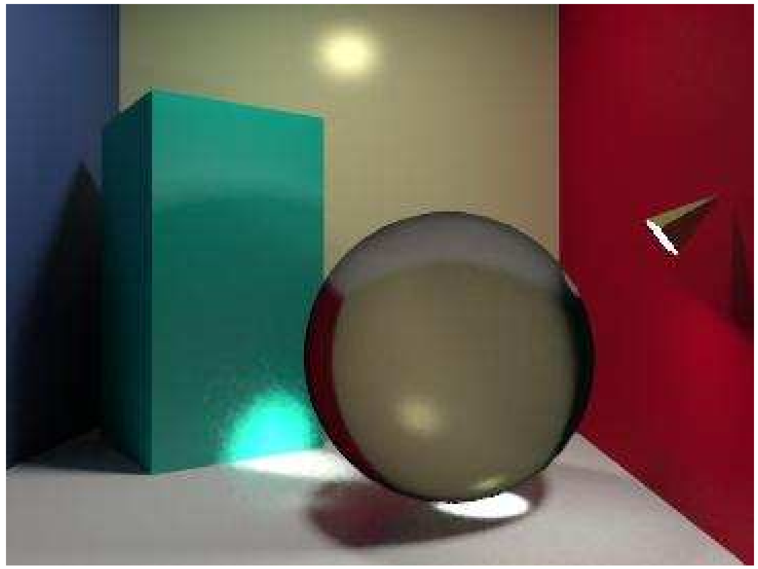


Philip Dutré

Bidirectional Path Tracing Algorithm

```
render image using bidirectional path tracing
  for each pixel
    for each sample
      pos = random position in pixel
      trace_paths(pos)

trace_paths( pixel pos )
  trace primary ray from observer through pixel pos
  generate an eye path of scattering events from the primary ray
  emit random photon from the light source
  generate a light path of scattering events from the photon
  combine( eye path, light path )

combine( eye path, light path )
  for each vertex y_j on the light path
    for each vertex x_i on the eye path
      if V(x_i,y_j) == 1            Are the vertices mutually visible?
        compute weight for the x_i - y_j path
        add weighted contribution to the corresponding pixel
```

Bidirectional Path Tracing

(RenderPark 98)

Summary

- Global illumination
  - Rendering equation
- Overview of solution methods
  - OpenGL
  - Radiosity
  - Ray tracing
  - Distribution ray tracing
  - Path tracing