Global Illumination

COS 426, Spring 2022

Felix Heide
Princeton University


A.


B.

Overview

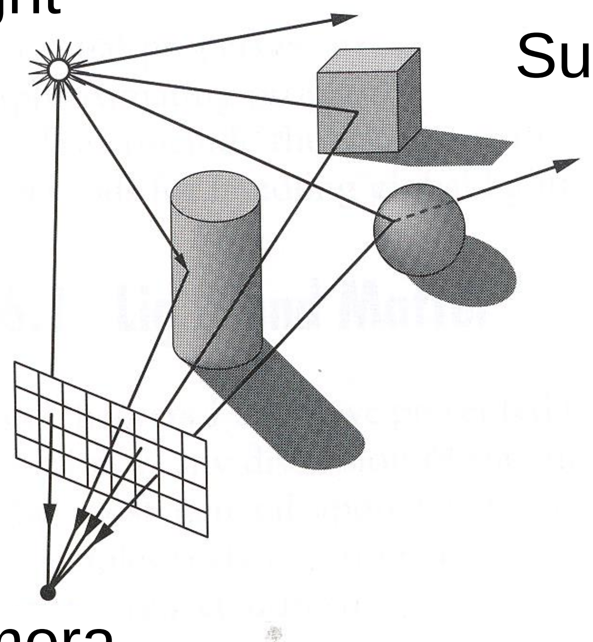
- Direct Illumination
  - Emission at light sources
  - Scattering at surfaces
- Global illumination
  - Shadows
  - Inter-object reflections
  - Rendering equation
  - Recursive ray tracing
  - More advanced ray tracing
  - Radiosity

Greg Ward

Direct Illumination (last lecture)

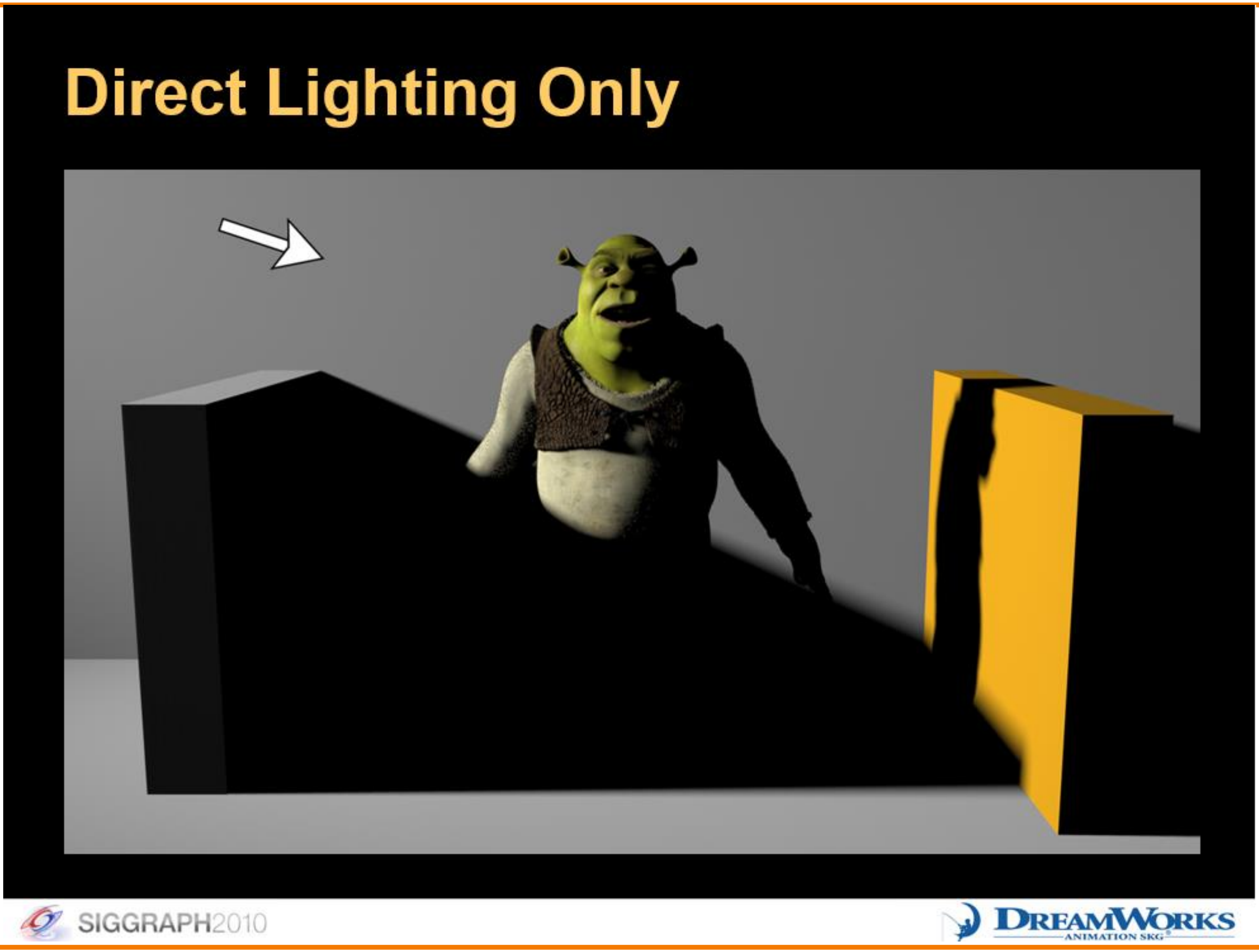
- For each ray traced from camera
  - Sum radiance from each light that is reflected at surface.

$$I_i = \frac{I_0}{c_a + l_a d + q_a d^2}$$



$$I = I_E + K_A I_{AL} + \sum_i \left(K_D (N \cdot L_i) + K_S (V \cdot R_i)^n \right) I_i$$

Direct illumination example



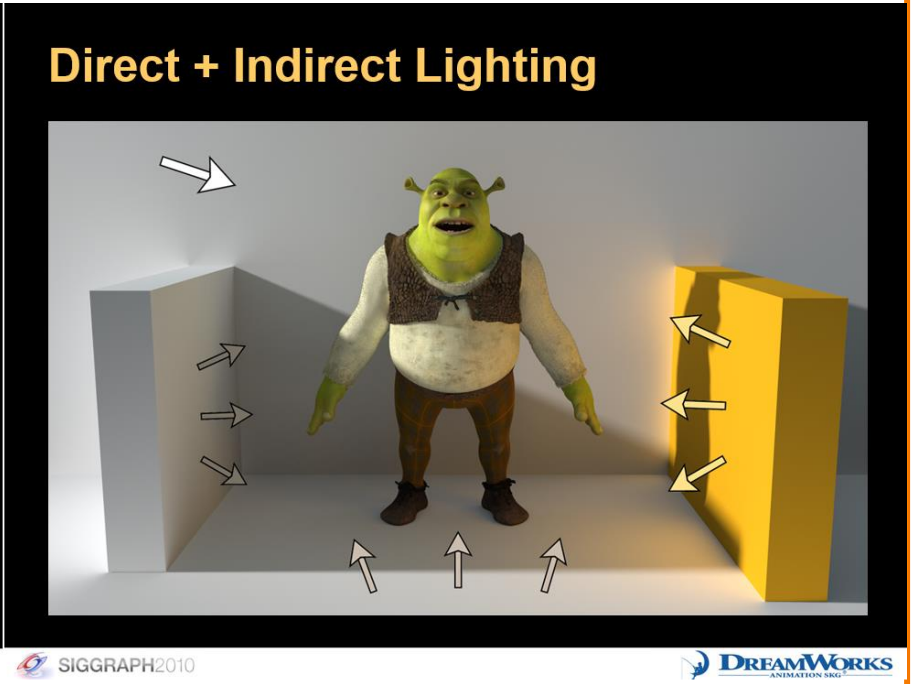
Overview

- Direct Illumination
  - Emission at light sources
  - Scattering at surfaces

➢ Global illumination
  - Shadows
  - Inter-object reflections
  - Rendering equation
  - Recursive ray tracing
  - More advanced ray tracing
  - Radiosity

Greg Ward

Global illumination example



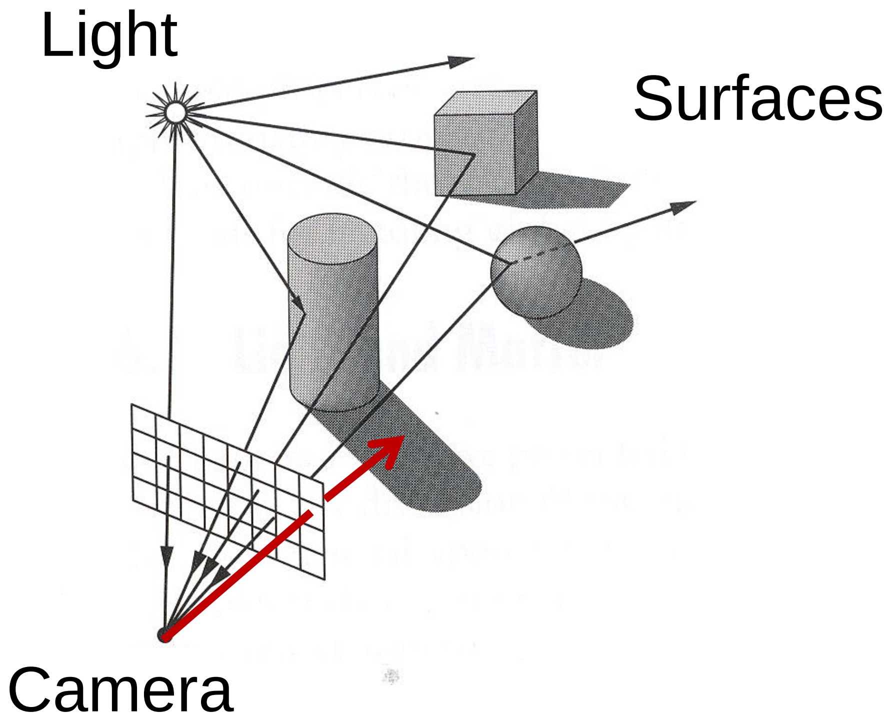
Overview

- Direct Illumination
  - Emission at light sources
  - Scattering at surfaces
- Global illumination
  - Shadows
  - Inter-object reflections
  - Rendering equation
  - Recursive ray tracing
  - More advanced ray tracing
  - Radiosity

Greg Ward

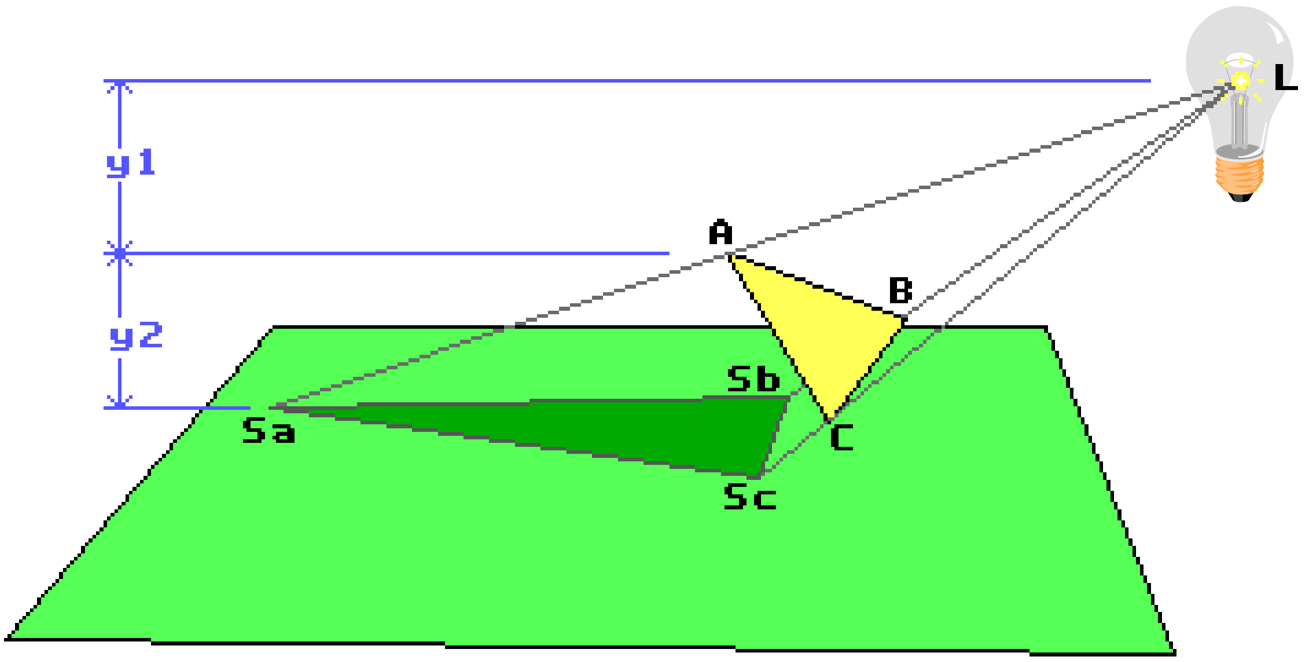
Shadows

- Hard shadows from point light sources



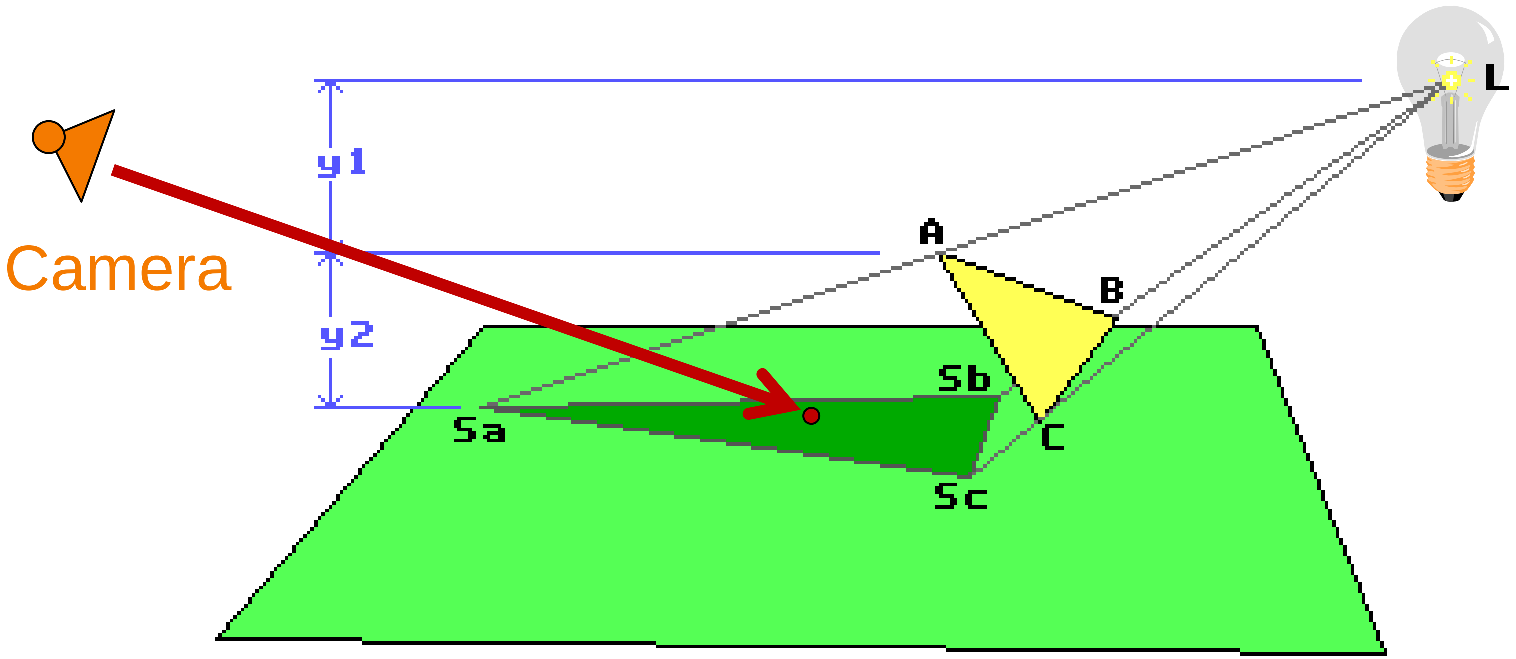
Shadows

- Hard shadows from point light sources



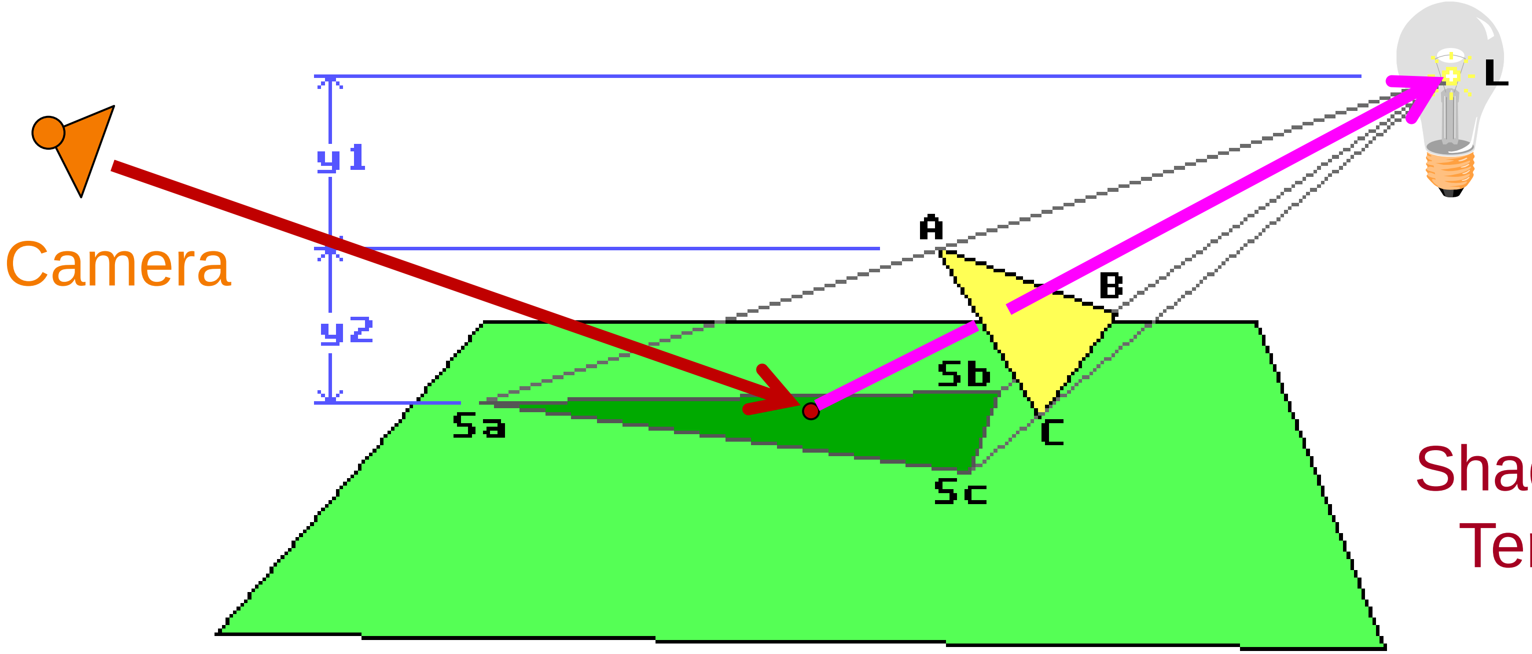
Shadows

- Hard shadows from point light sources



Shadows

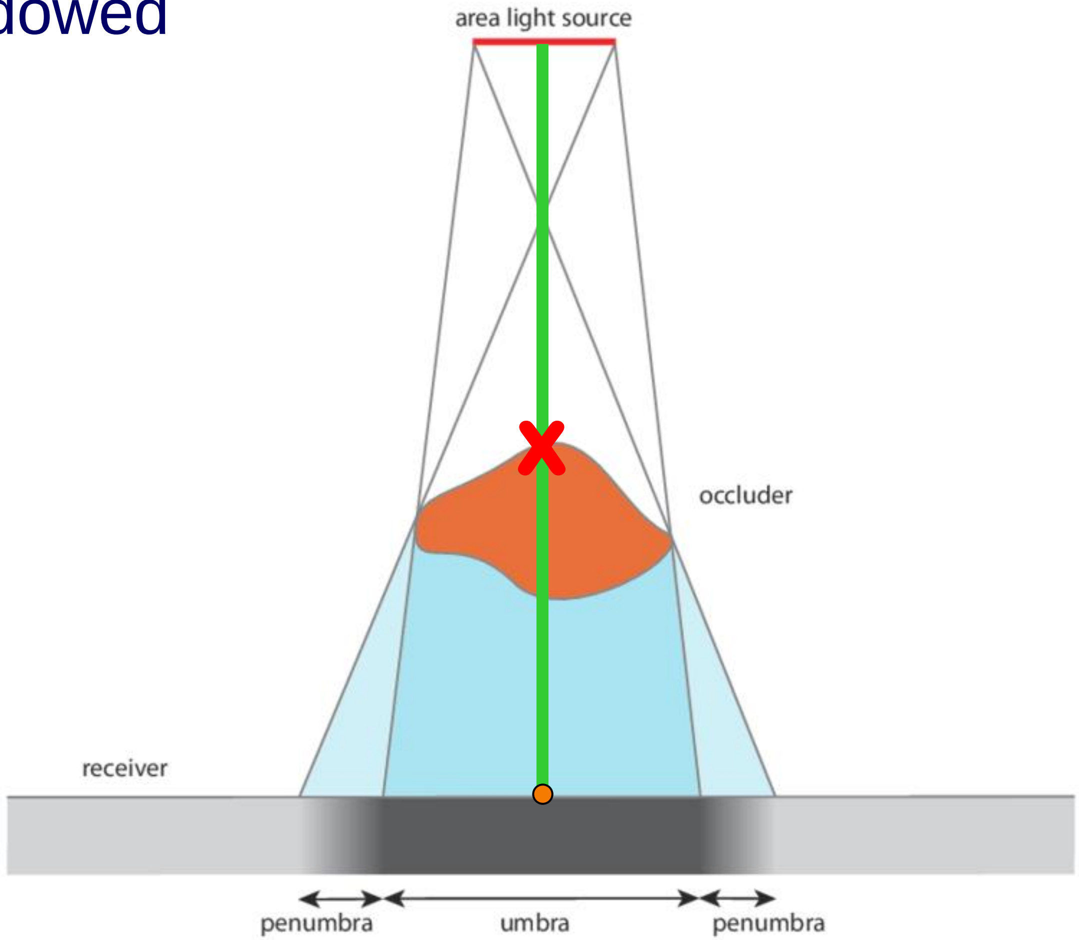
- Hard shadows from point light sources
  - Cast ray towards light; S_L=0 if blocked, S_L=1 otherwise



$$I = I_E + K_A I_{AL} + \sum_{i \in \text{lights}} \left(K_D (N \cdot L_i) + K_S (V \cdot R_i)^n \right) S_i I_i$$

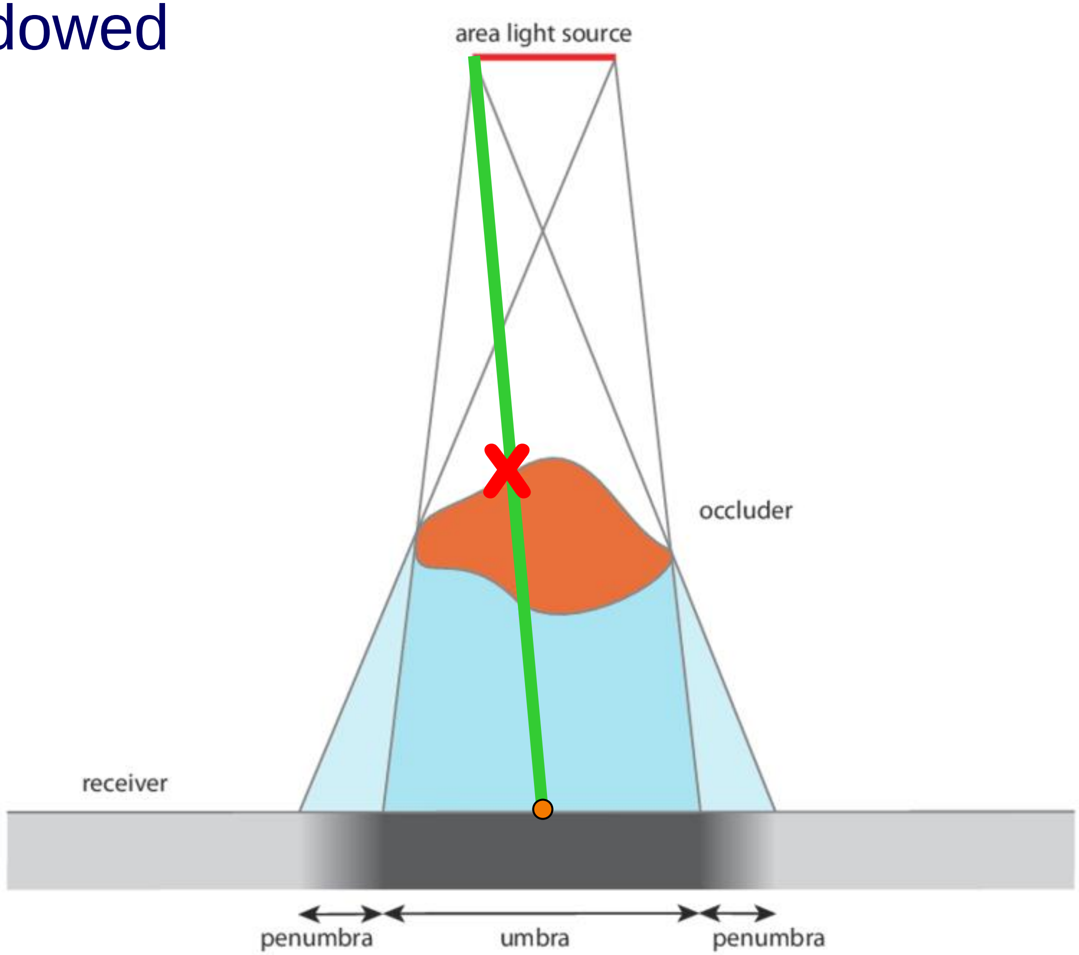
Shadows in 2D

- Soft shadows from area light sources
  - Umbra = fully shadowed



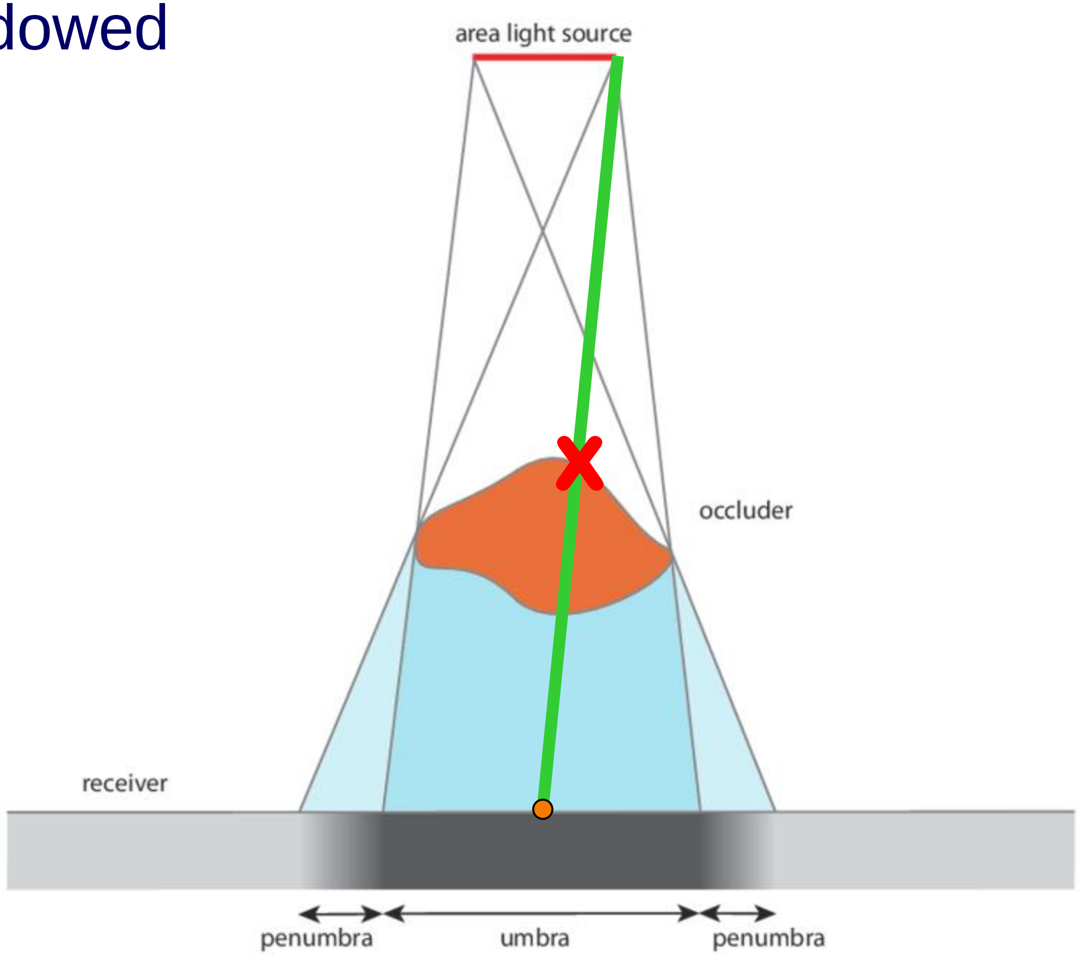
Shadows in 2D

- Soft shadows from area light sources
  - Umbra = fully shadowed



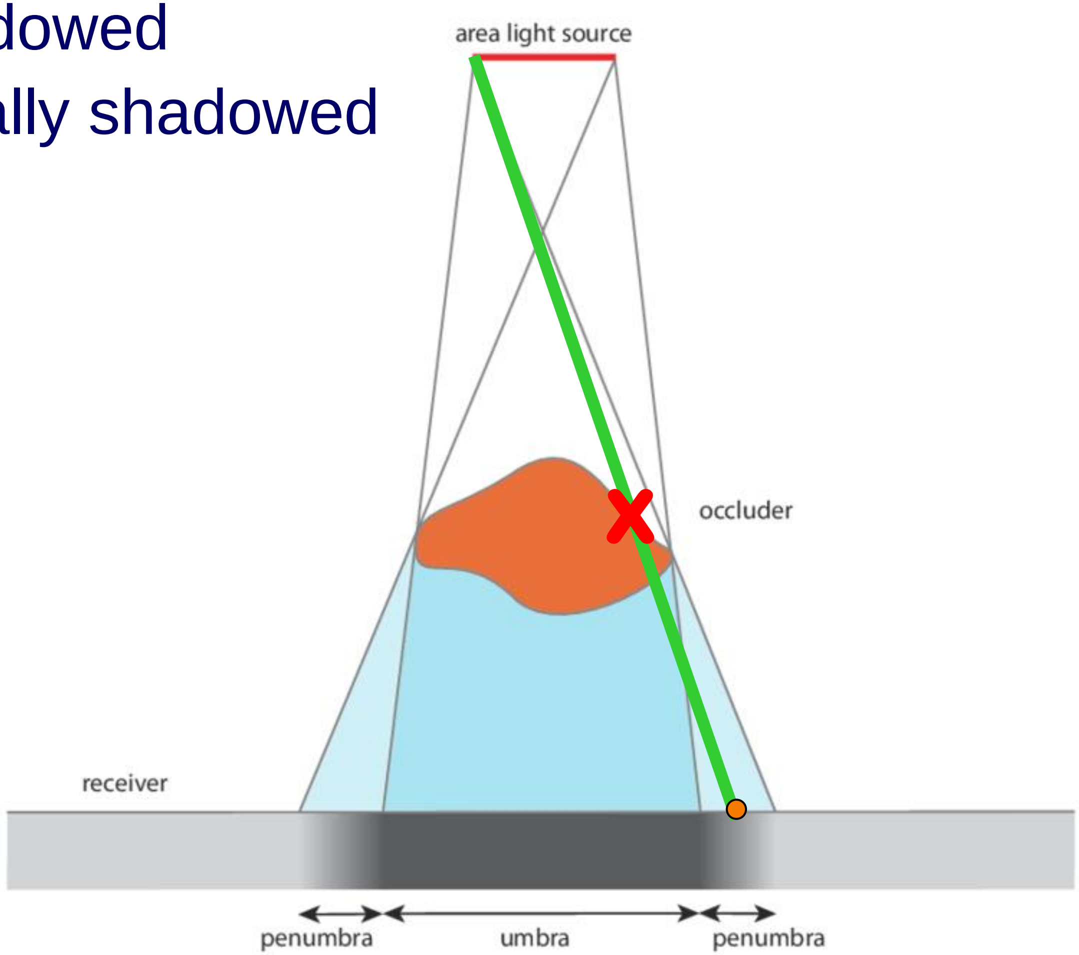
Shadows in 2D

- Soft shadows from area light sources
  - Umbra = fully shadowed



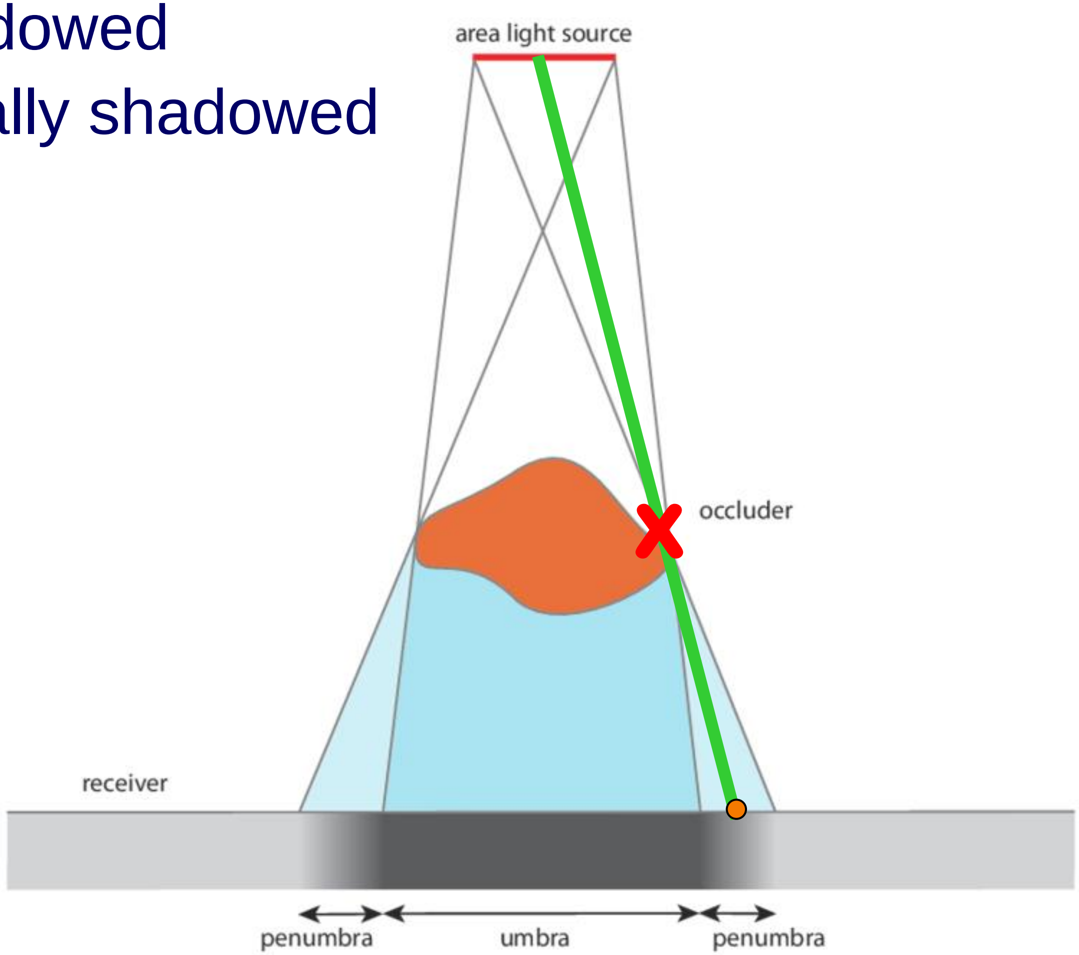
Shadows in 2D

- Soft shadows from area light sources
  - Umbra = fully shadowed
  - Penumbra = partially shadowed



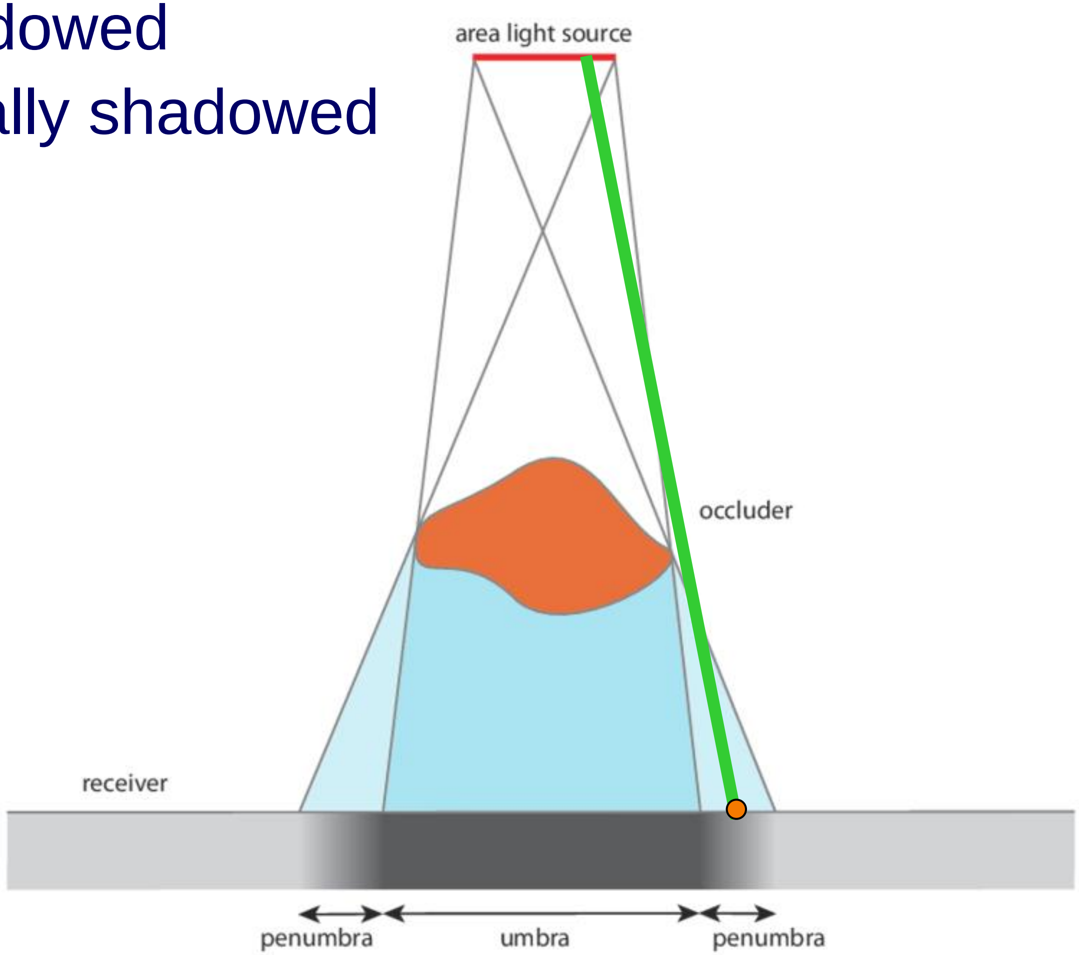
Shadows in 2D

- Soft shadows from area light sources
  - Umbra = fully shadowed
  - Penumbra = partially shadowed



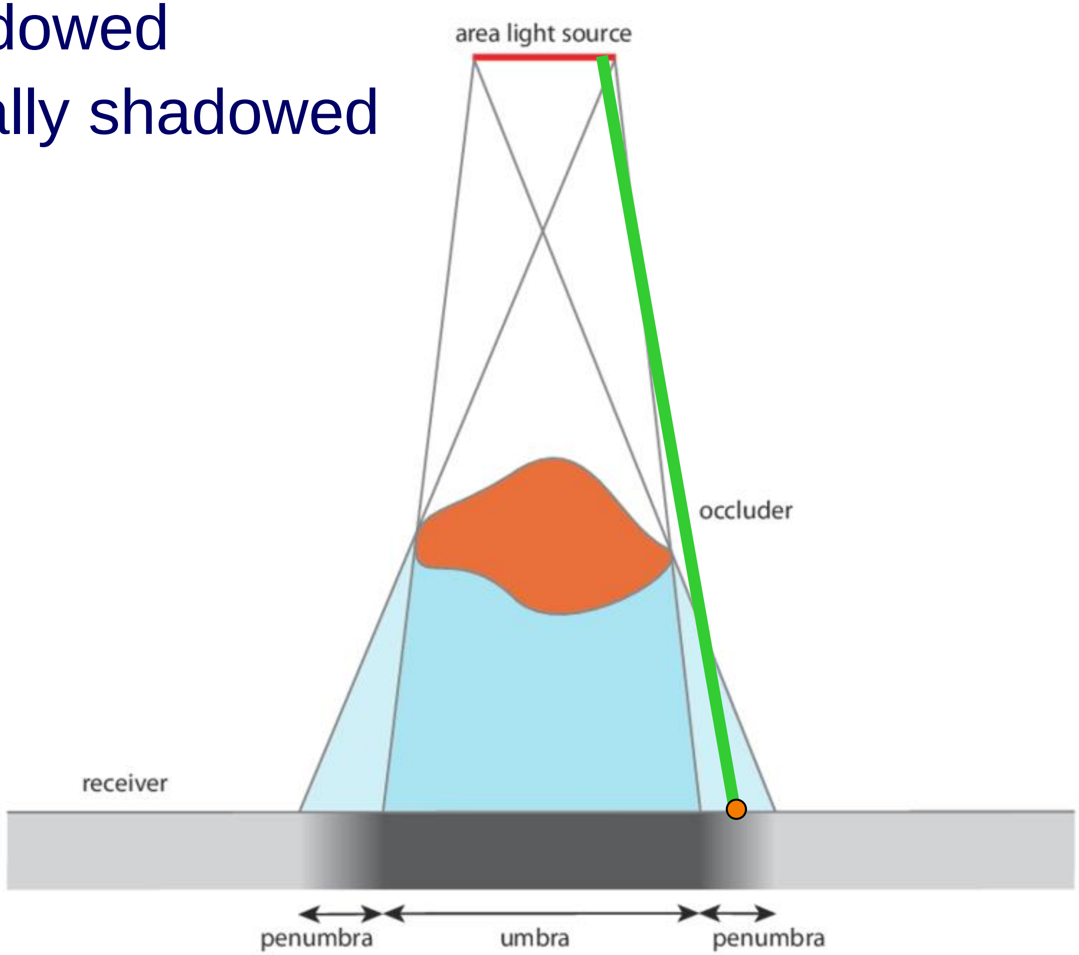
Shadows in 2D

- Soft shadows from area light sources
  - Umbra = fully shadowed
  - Penumbra = partially shadowed



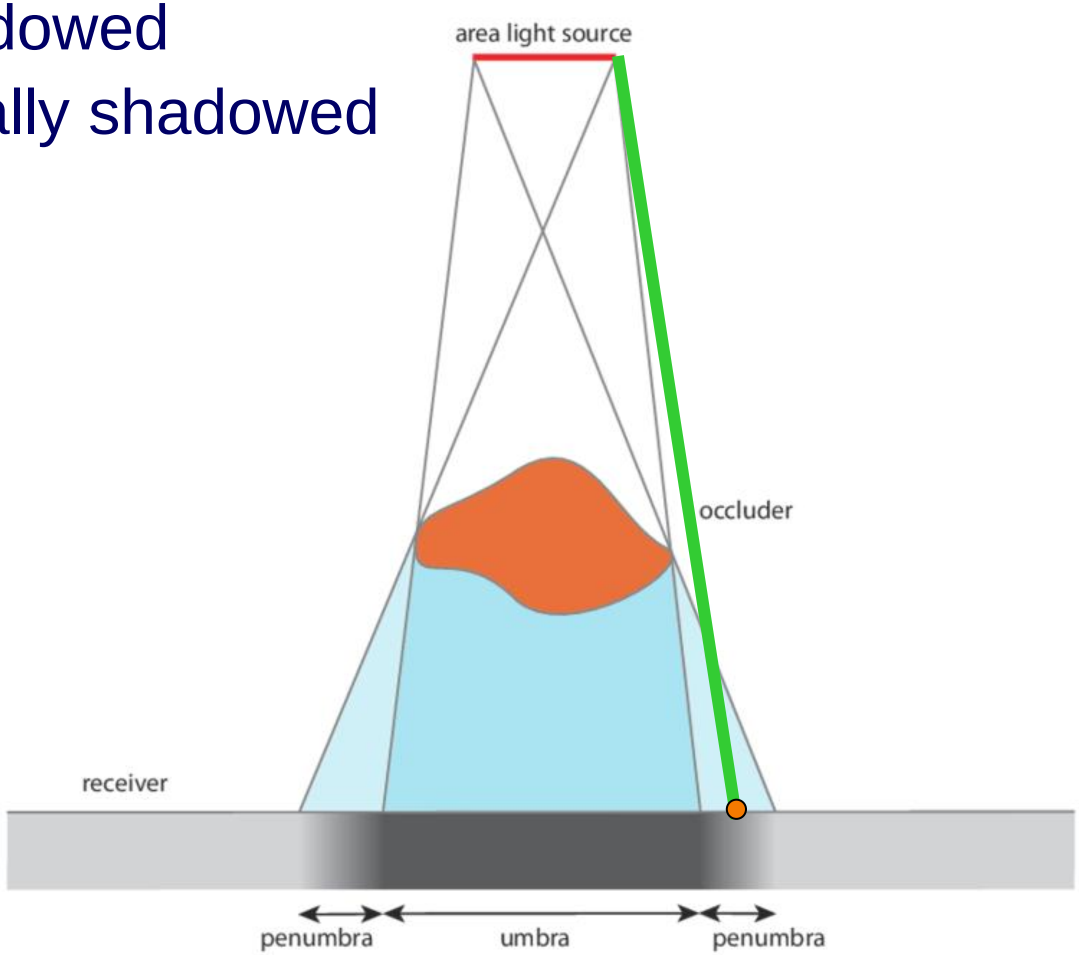
Shadows in 2D

- Soft shadows from area light sources
  - Umbra = fully shadowed
  - Penumbra = partially shadowed



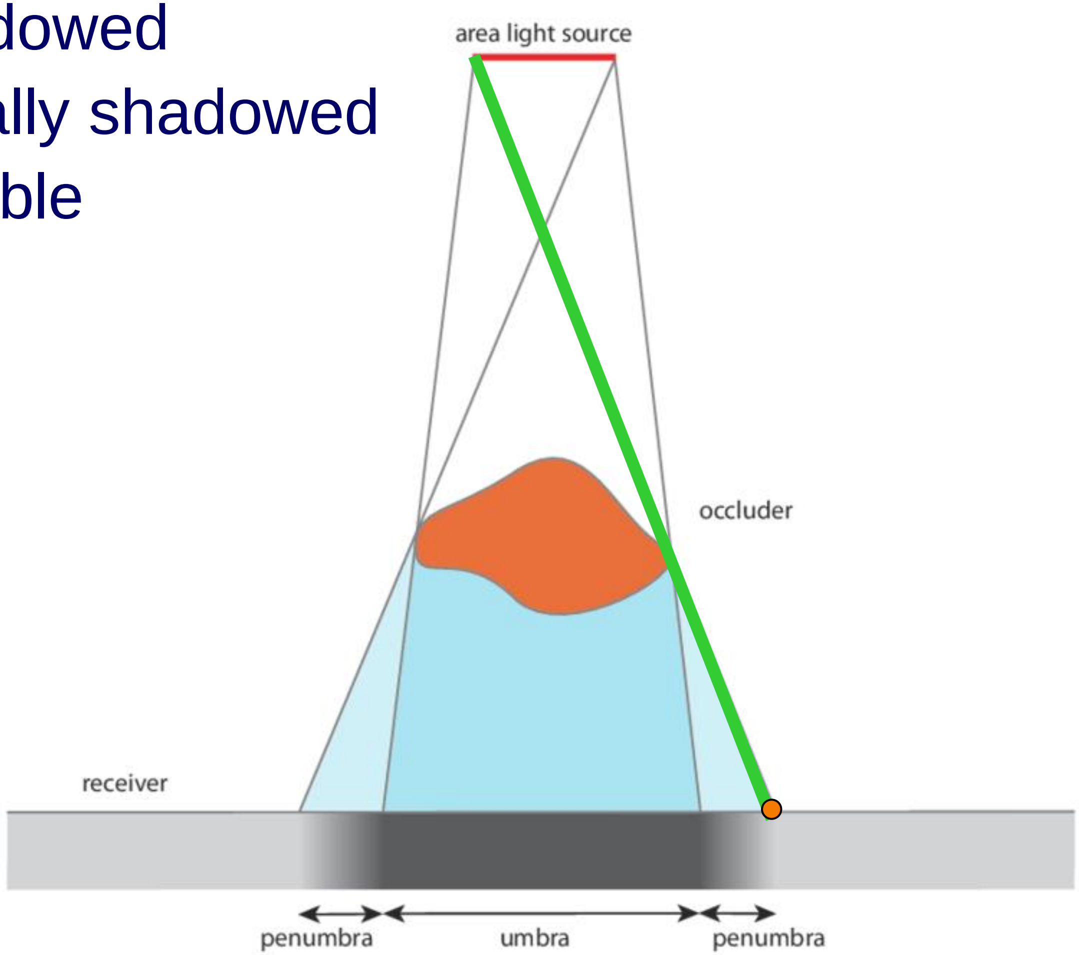
Shadows in 2D

- Soft shadows from area light sources
  - Umbra = fully shadowed
  - Penumbra = partially shadowed



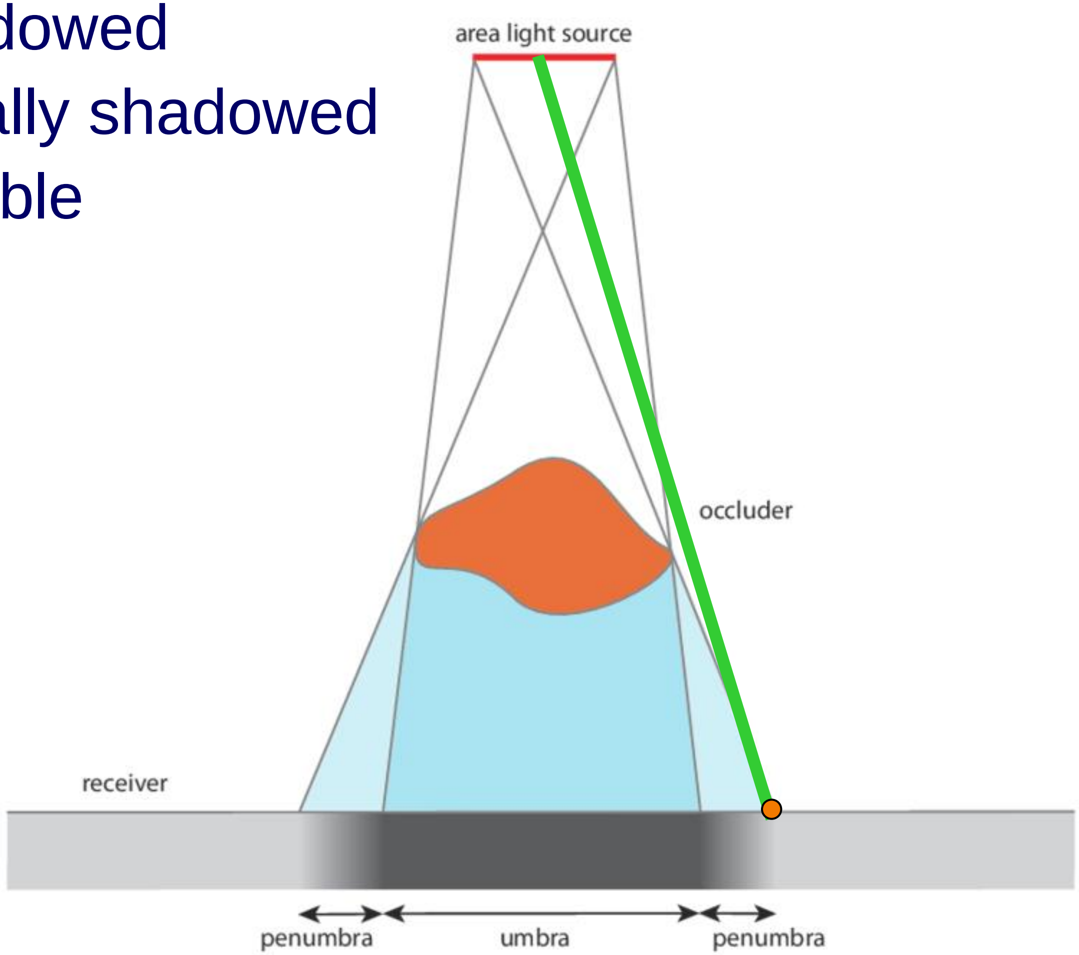
Shadows in 2D

- Soft shadows from area light sources
  - Umbra = fully shadowed
  - Penumbra = partially shadowed
  - Outside = fully visible



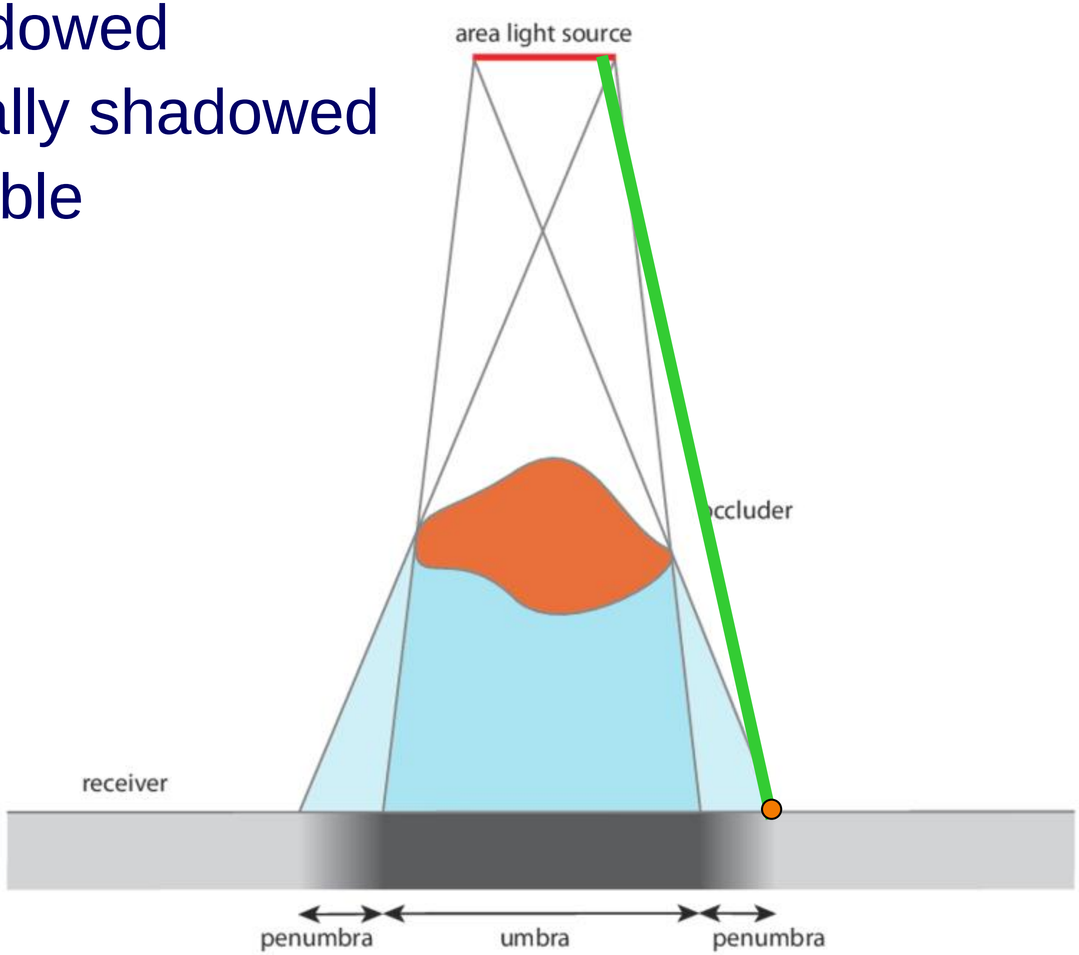
Shadows in 2D

- Soft shadows from area light sources
  - Umbra = fully shadowed
  - Penumbra = partially shadowed
  - Outside = fully visible



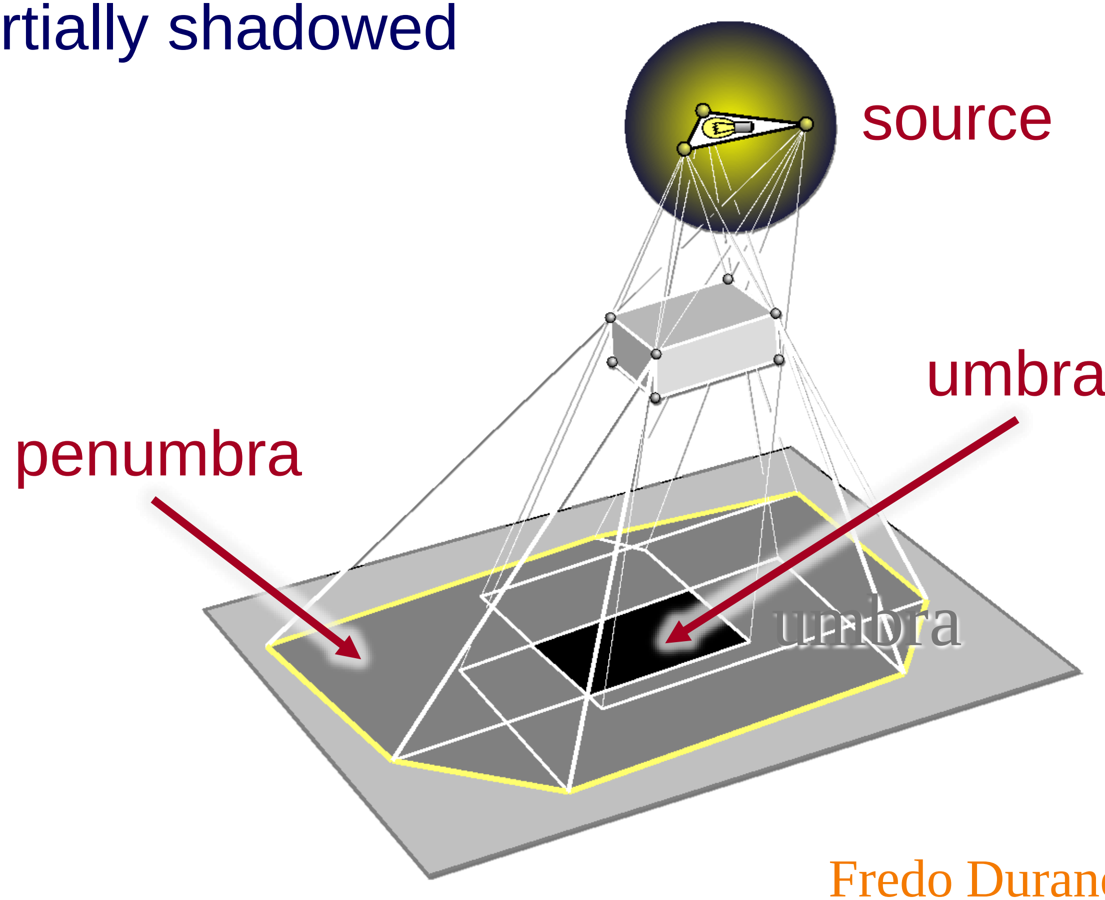
Shadows in 2D

- Soft shadows from area light sources
  - Umbra = fully shadowed
  - Penumbra = partially shadowed
  - Outside = fully visible



Shadows in 3D

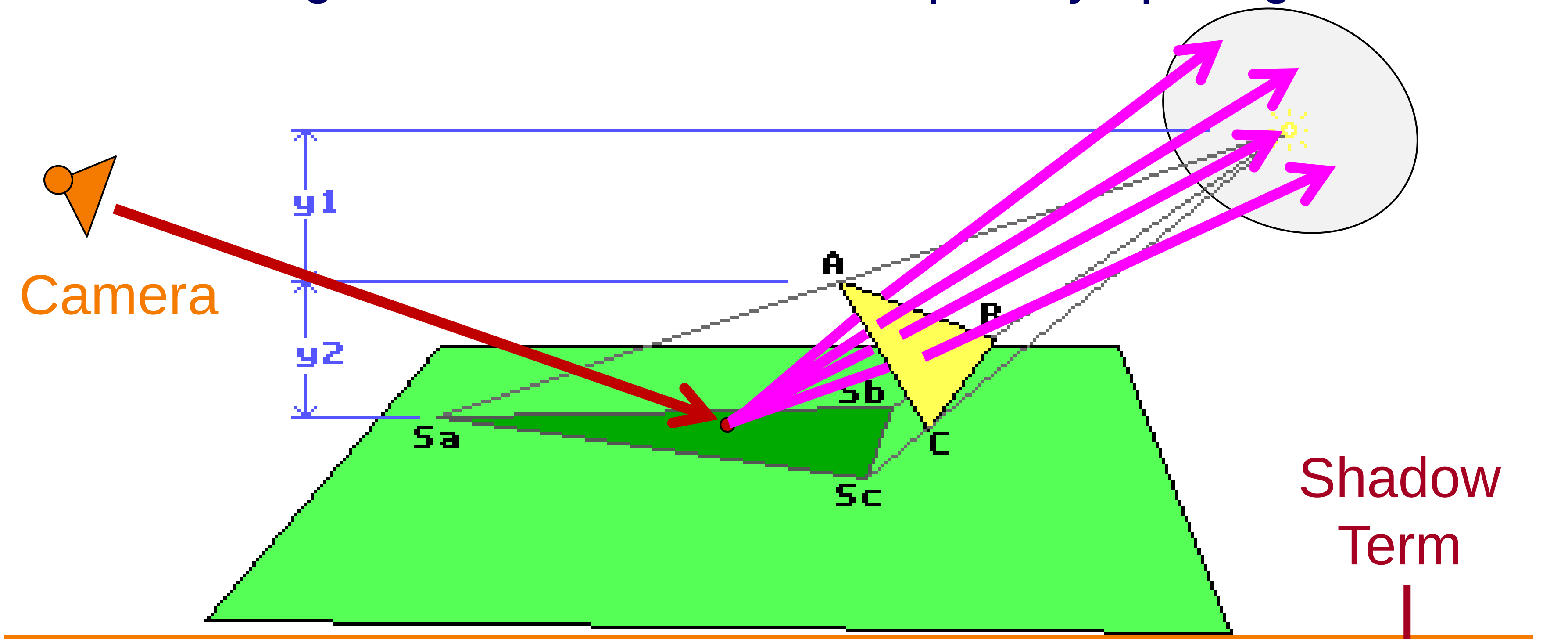
- Soft shadows from area light sources
  - Umbra = fully shadowed
  - Penumbra = partially shadowed



Fredo Durand

Shadows

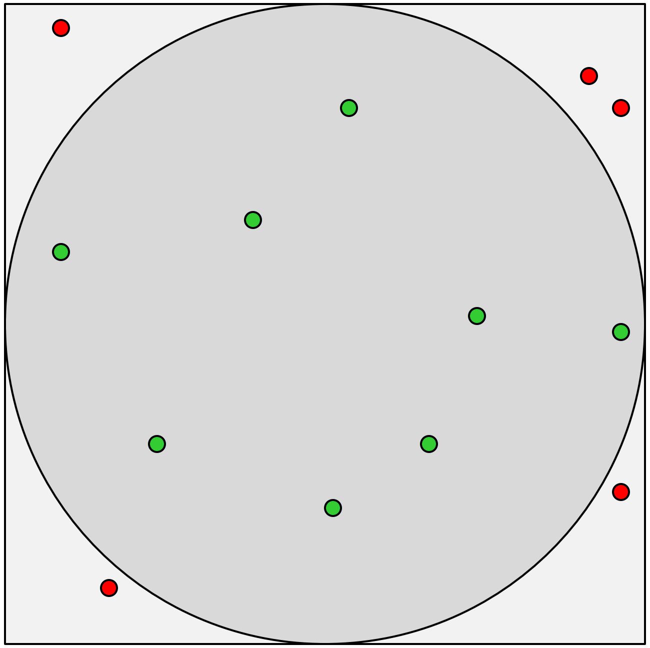
- Soft shadows from area light sources
  - Average illumination for M sample rays per light



$$I = \cdots + \sum_{i \in \text{AreaLights}} \tfrac{1}{M} \sum_{j \in \text{samples}}^{M} \left(K_D (N \cdot L_i) + K_S (V \cdot R_i)^n \right) S_{ij} I_{ij}$$

Shadows

- Soft shadows from circular area light sources
  - Average illumination for M sample rays per light
    - Generate M random sample points on area light (e.g., with rejection sampling)
    - Compute illumination for every sample
    - Average



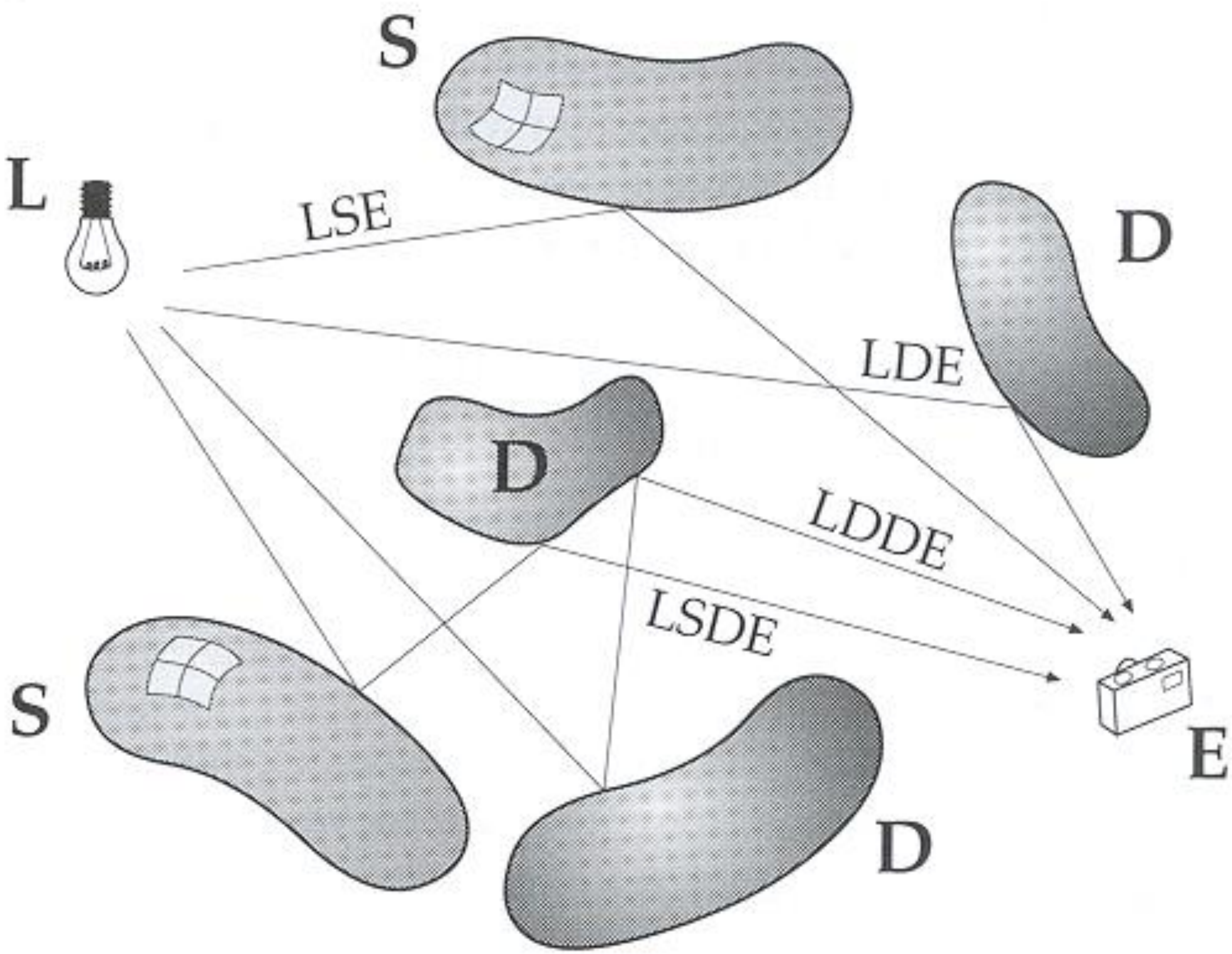
$$I = \cdots + \sum_{i \in \text{AreaLights}} \tfrac{1}{M} \sum_{j \in \text{samples}}^{M} \left(K_D (N \cdot L_i) + K_S (V \cdot R_i)^n \right) S_{ij} I_{ij}$$

Overview

- Direct Illumination
  - Emission at light sources
  - Scattering at surfaces
- Global illumination
  - Shadows
  - ➢ Inter-object reflections
  - Rendering equation
  - Recursive ray tracing
  - More advanced ray tracing
  - Radiosity

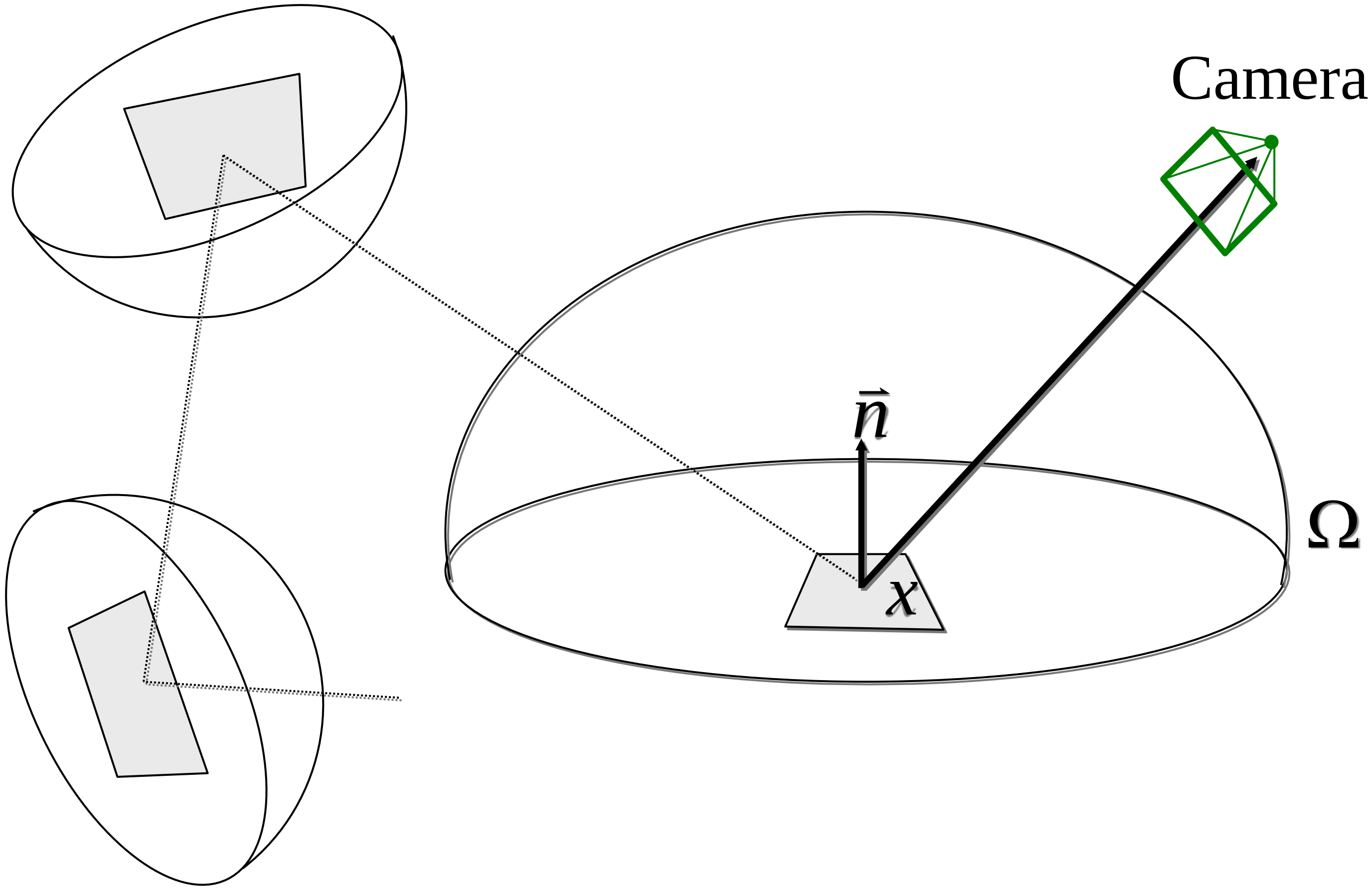
Greg Ward

Inter-Object Reflection



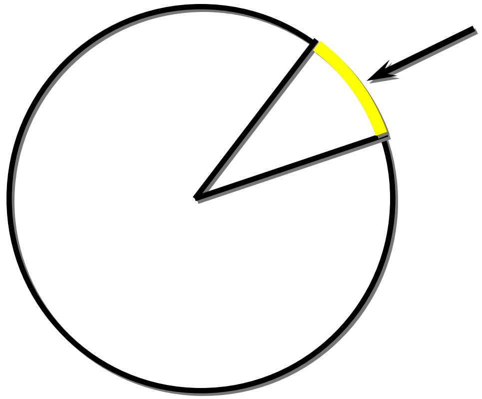
Inter-Object Reflection

- Radiance leaving point x on surface is sum of reflected irradiance arriving from other surfaces



Solid Angle

- Angle in radians



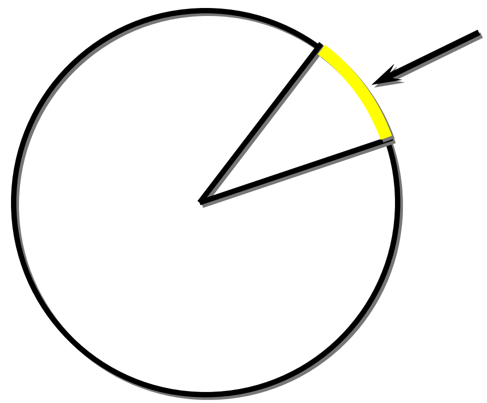
Length l

Angle $\theta = l/r$

(Full circle is 2π radians.)

Solid Angle

- Angle in radians

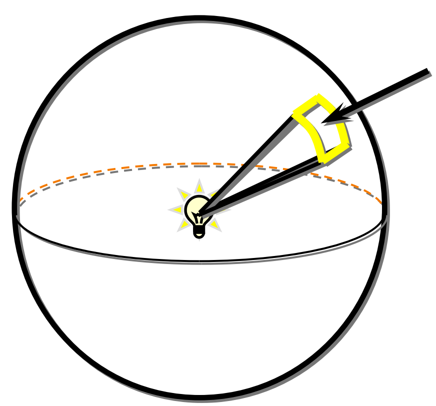


Length l

Angle $\theta = l/r$

(Full circle is 2π radians.)

- Solid angle in steradians



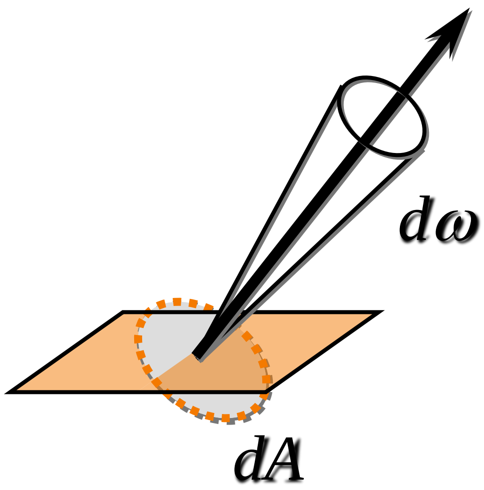
Area A

Solid angle $\omega = A/r^2$

(Full sphere is 4π steradians.)

Light Emitted from a Surface

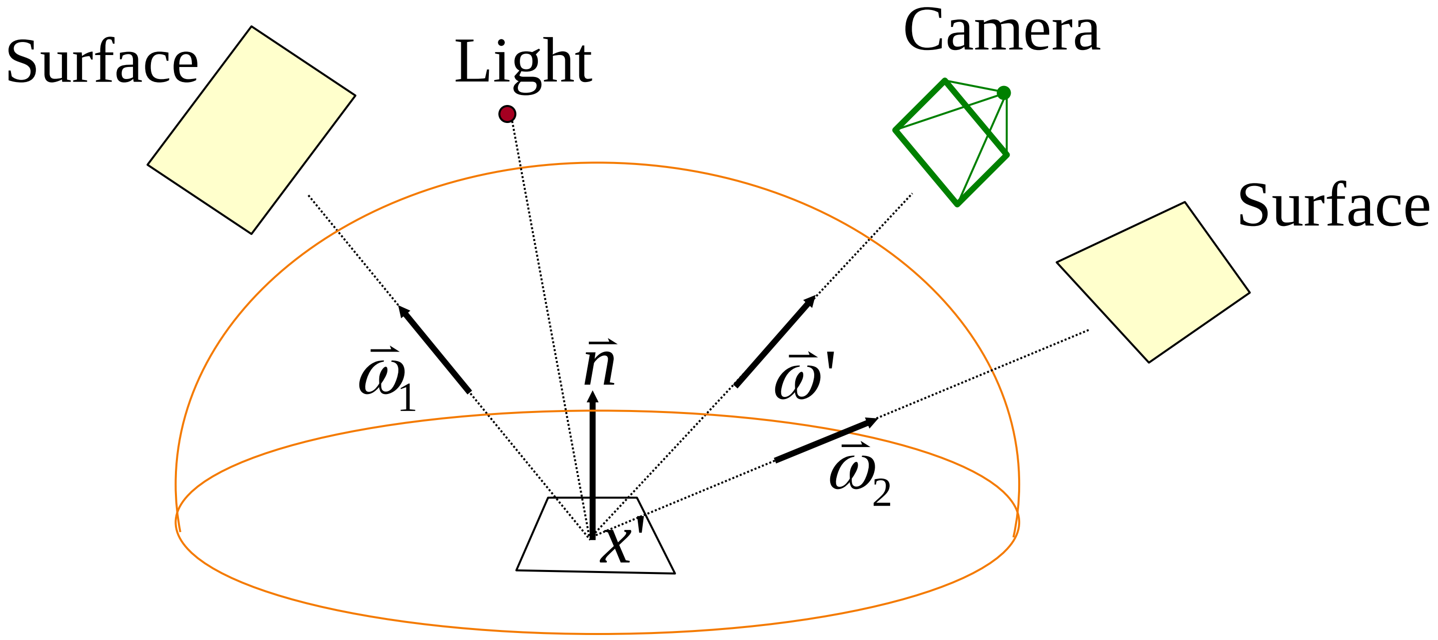
- Power per unit area per unit solid angle – *Radiance* (L)
  - Measured in W/m²/sr



$$L = \frac{d\Phi}{dA\,d\omega}$$

Rendering Equation [Kajiya 86]

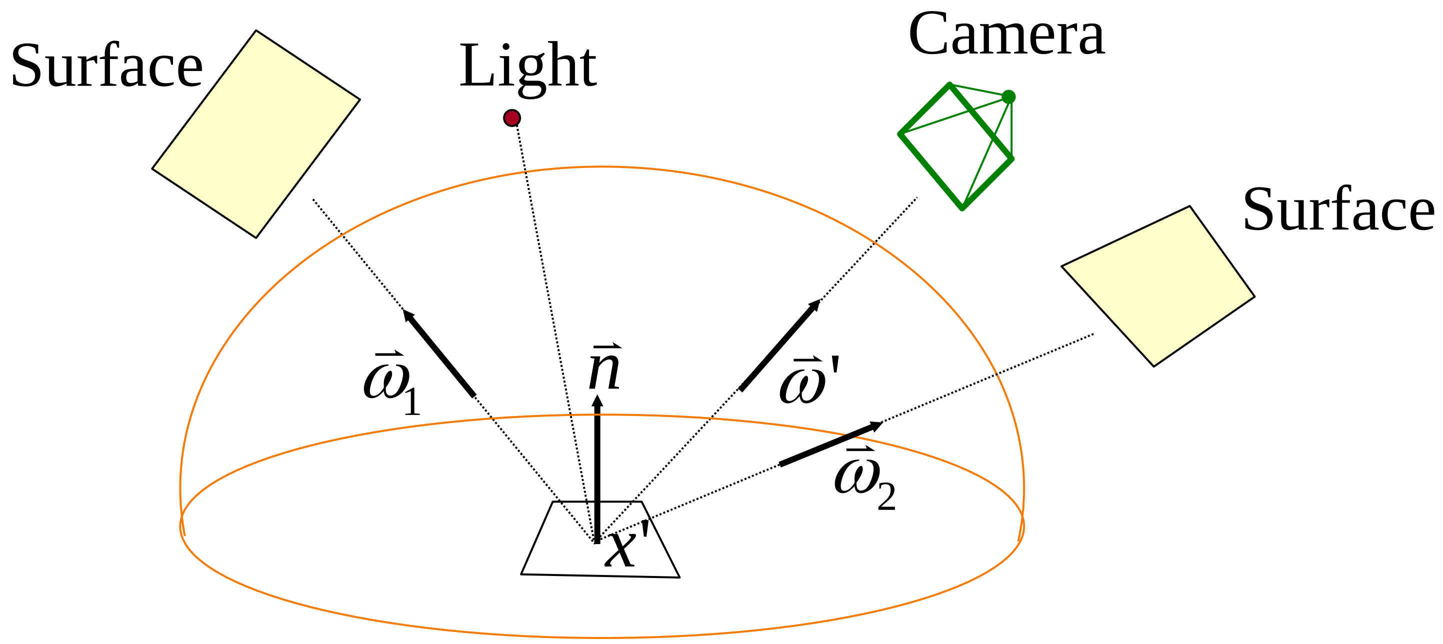
- Compute radiance in outgoing direction



$$L_o(x', \vec{\omega}') = L_e(x', \vec{\omega}')$$

Rendering Equation [Kajiya 86]

- Compute radiance in outgoing direction

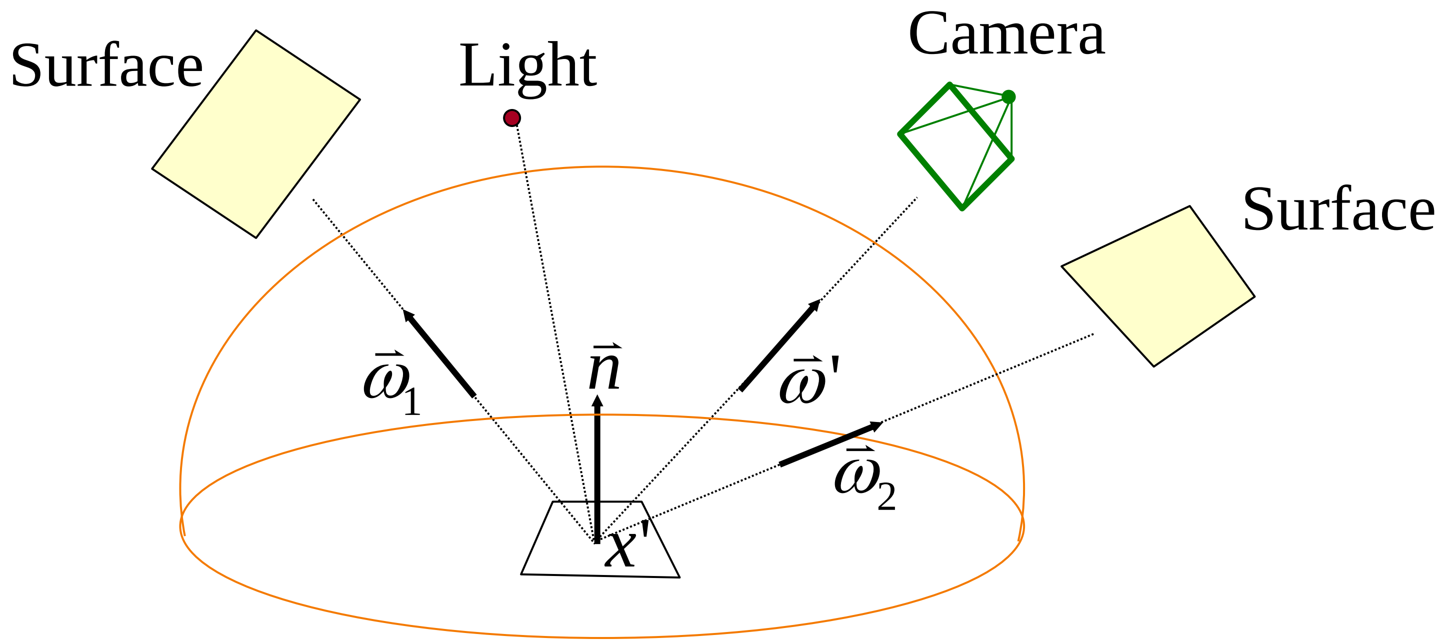


$$L_o(x',\vec{\omega}') = L_e(x',\vec{\omega}')$$

$$L_i(x',\vec{\omega})$$

Rendering Equation [Kajiya 86]

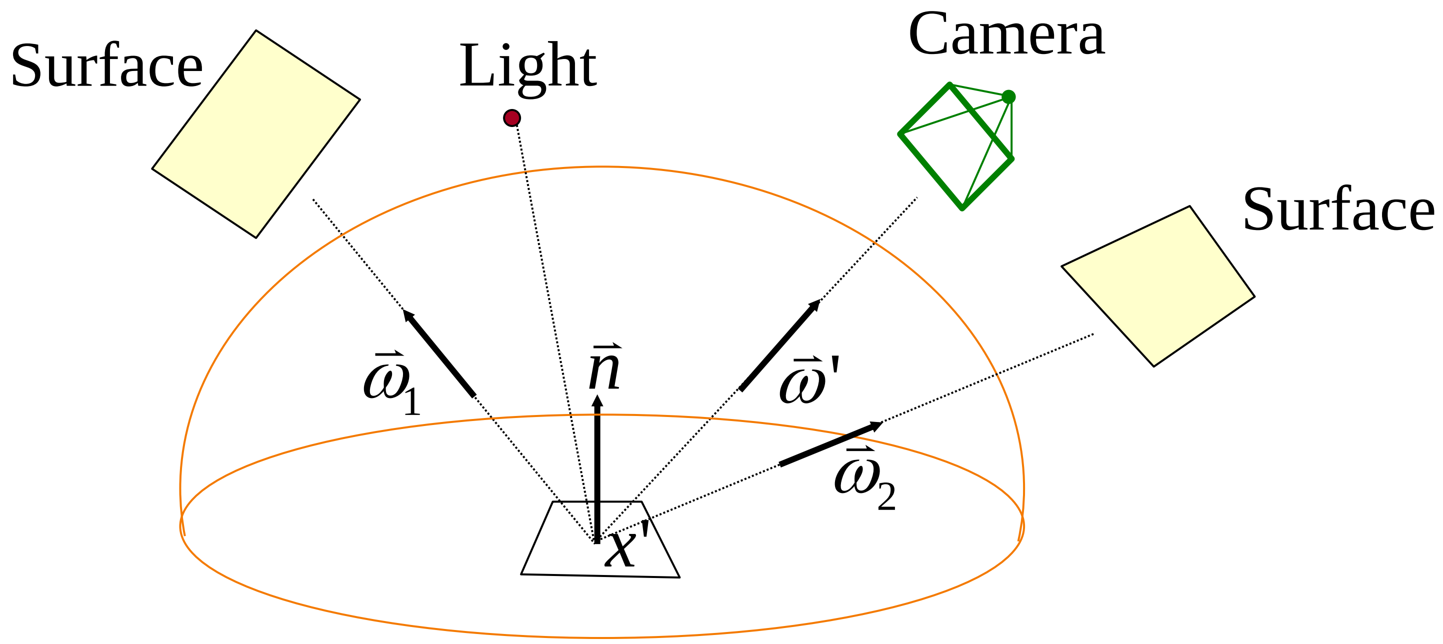
- Compute radiance in outgoing direction



$$L_o(x',\vec{\omega}') = L_e(x',\vec{\omega}') \qquad f_r(x',\vec{\omega},\vec{\omega}')(\vec{\omega}\cdot\vec{n})L_i(x',\vec{\omega})$$

Rendering Equation [Kajiya 86]

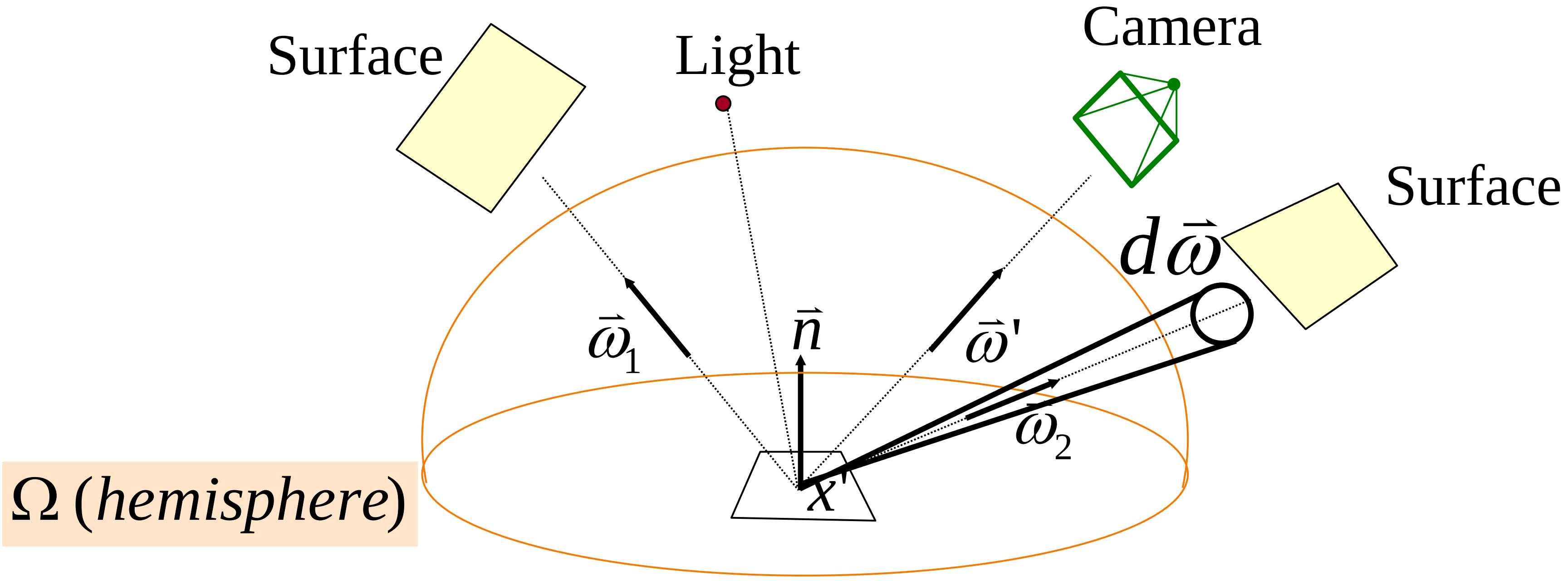
- Compute radiance in outgoing direction by integrating reflections over all incoming directions



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

Rendering Equation [Kajiya 86]

- Compute radiance in outgoing direction by integrating reflections over all incoming directions



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

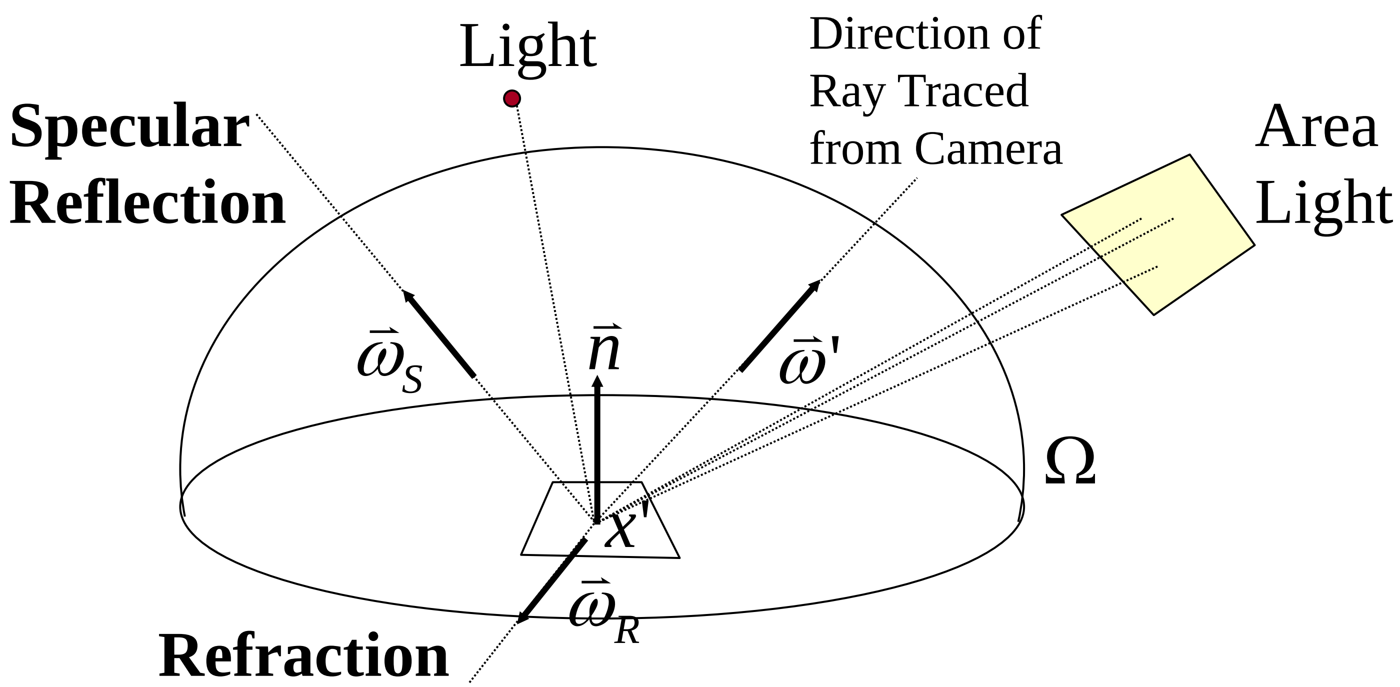
Overview

- Direct Illumination
  - Emission at light sources
  - Scattering at surfaces
- Global illumination
  - Shadows
  - Inter-object reflections
  - Rendering equation
  - ➢ Recursive ray tracing
  - More advanced ray tracing
  - Radiosity

Greg Ward

Recursive Ray Tracing

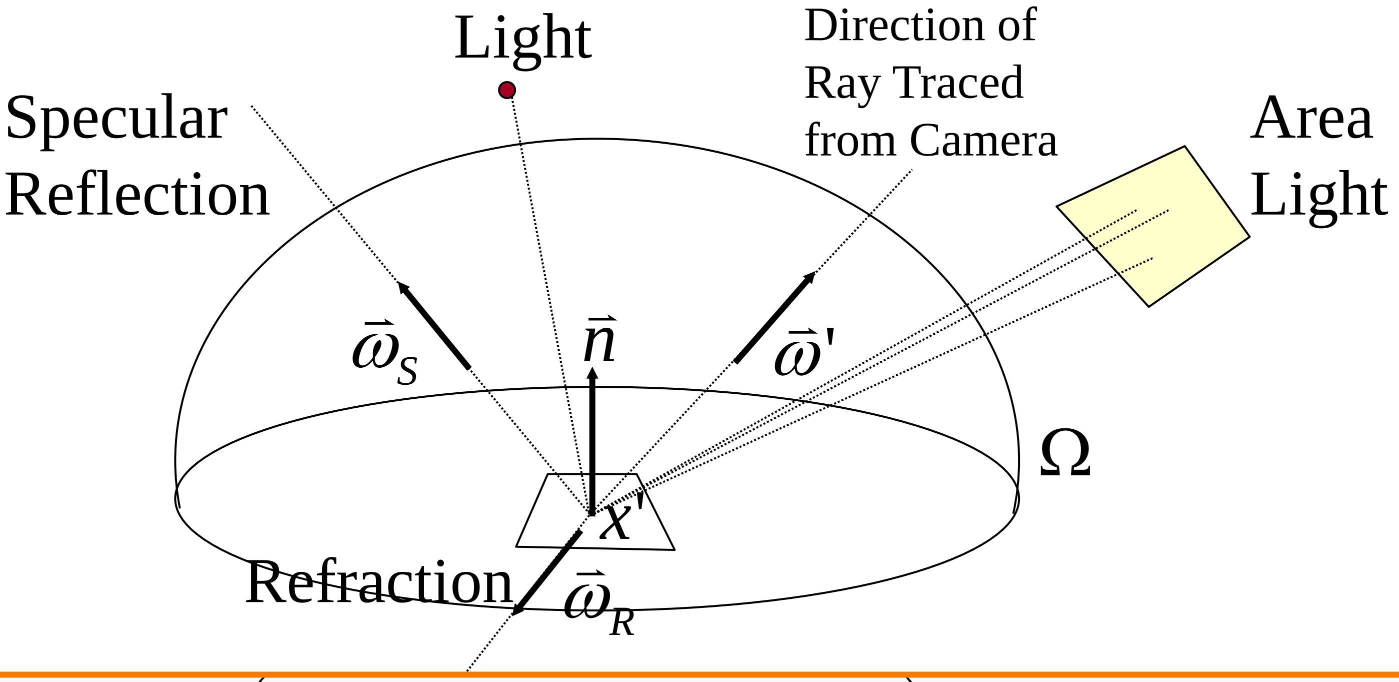
- Assume only significant irradiance is in directions of light sources, specular reflection, and refraction



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

Recursive Ray Tracing

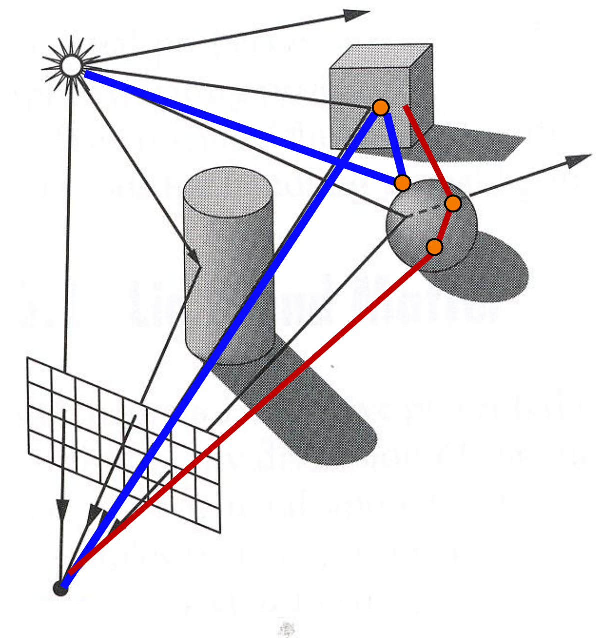
- Compute radiance in outgoing direction by summing reflections from directions of lights, specular reflections, and refractions



$$I = I_E + K_A I_{AL} + \sum_L \left(K_D (N \cdot L_i) + K_S (V \cdot R_i)^n \right) S_L I_L + K_S I_R + K_T I_T$$

Recursive Ray Tracing

- Same as ray casting, but trace secondary rays for specular (mirror) reflection and refraction

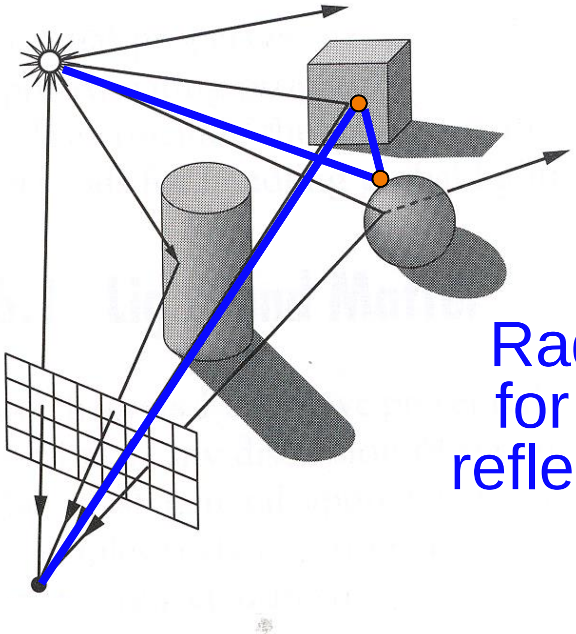


$$I = I_E + K_A I_{AL} + \sum_L \left(K_D (N \cdot L_i) + K_S (V \cdot R_i)^n \right) S_L I_L + K_S I_R + K_T I_T$$

Specular Reflection

- Trace secondary ray in direction of mirror reflection
  - Evaluate radiance along secondary ray and include it into illumination model

wikimedia.org/wikipedia/en/c/c1/Cloud_Gate_(The_Bean)_from_east'.jpg

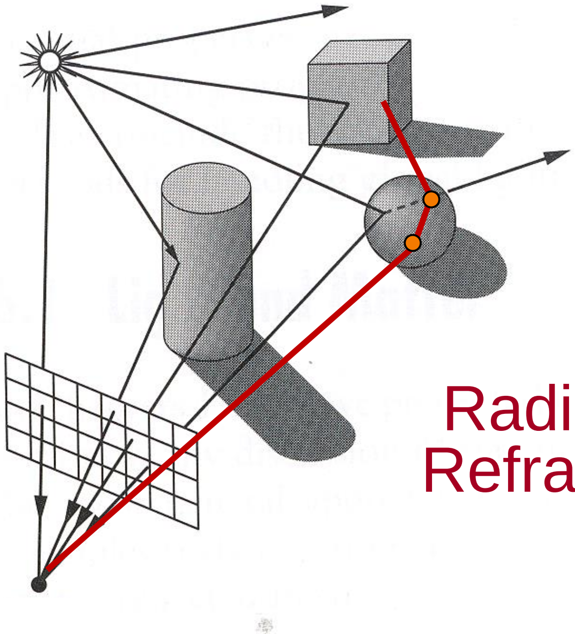


Radiance for mirror reflection ray

$$I = I_E + K_A I_{AL} + \sum_L \left(K_D (N \cdot L_i) + K_S (V \cdot R_i)^n \right) S_L I_L + K_S I_R + K_T I_T$$

Refraction

- Trace secondary ray in direction of refraction
  - Evaluate radiance along secondary ray and include it into illumination model



Radiance for Refraction Ray

$$I = I_E + K_A I_{AL} + \sum_L \left(K_D (N \cdot L_i) + K_S (V \cdot R_i)^n \right) S_L I_L + K_S I_R + K_T \boldsymbol{I_T}$$

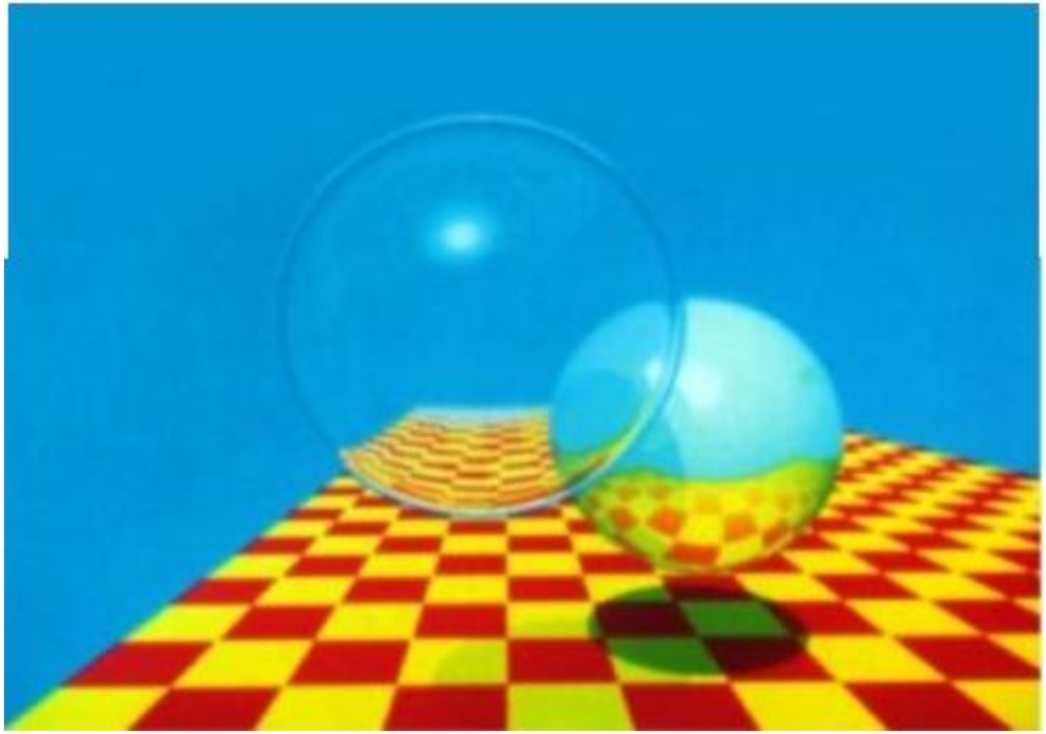
Recursive Ray Tracing

- ComputeRadiance is called recursively

```
R3Rgb ComputeRadiance(R3Scene *scene, R3Ray *ray, R3Intersection& hit)
{
        R3Ray specular_ray = SpecularRay(ray, hit);
        R3Ray refractive_ray = RefractiveRay(ray, hit);
        R3Rgb radiance = Phong(scene, ray, hit) +
                         Ks * ComputeRadiance(scene, specular_ray) +
                         Kt * ComputeRadiance(scene, refractive_ray);
        return radiance;
}
```

Recursive Ray Tracing

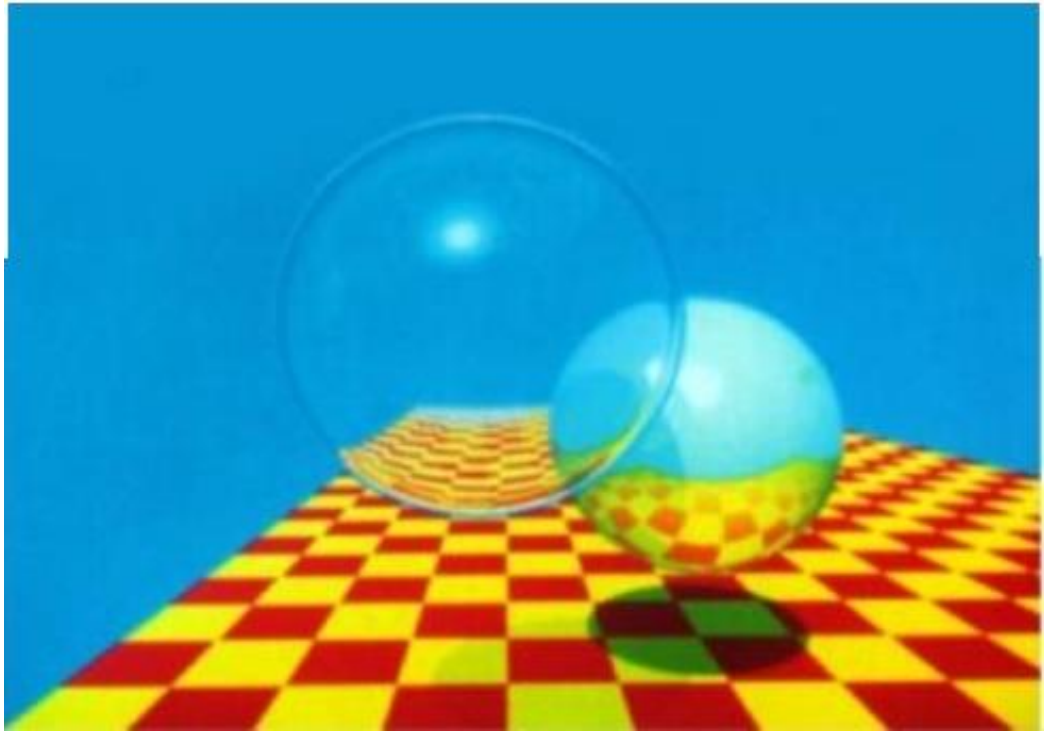
- Specular reflection and refraction



Whitted

Recursive Ray Tracing

- Specular reflection and refraction → **LD(S|R)*E**



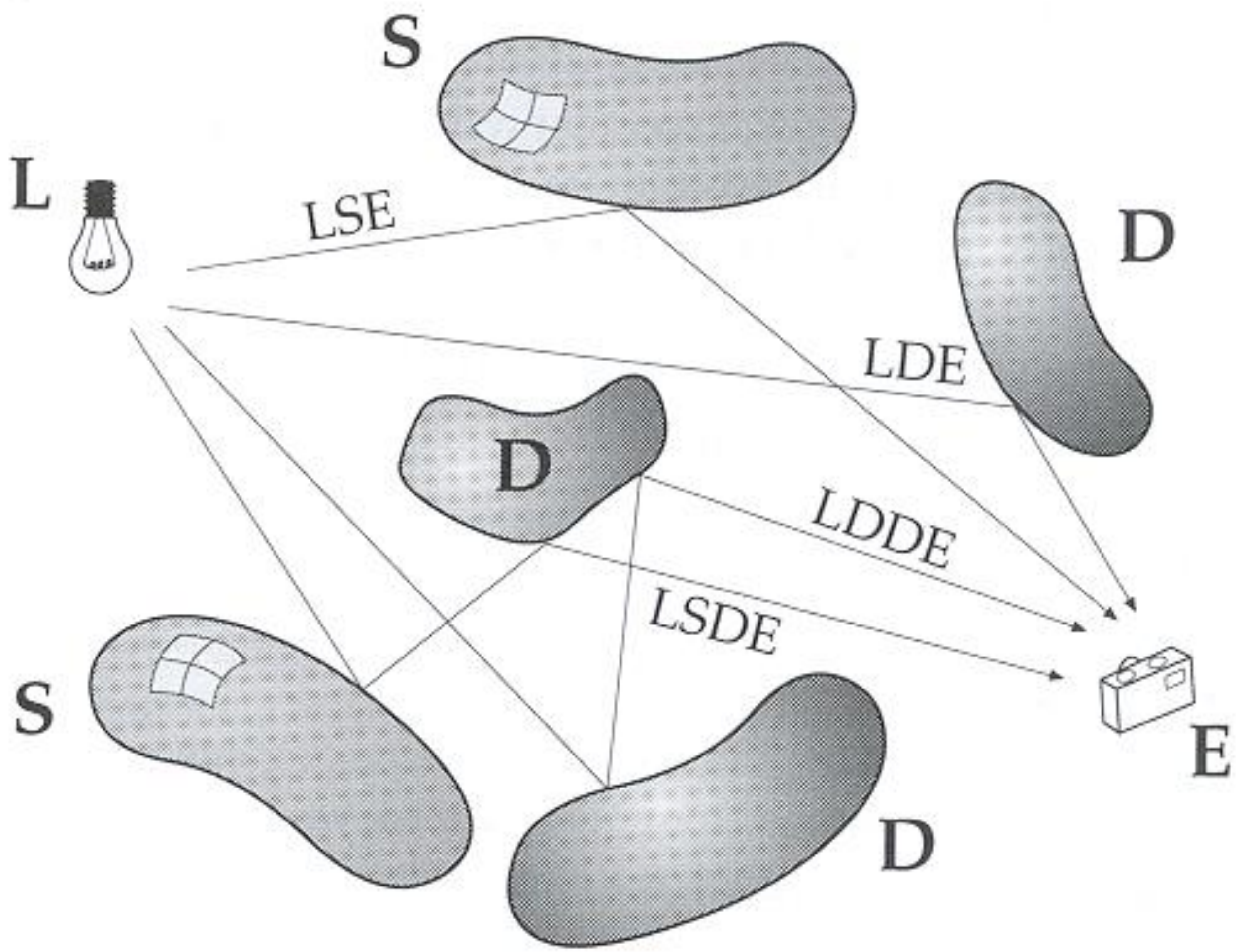
Whitted

Overview

- Direct Illumination
  - Emission at light sources
  - Scattering at surfaces
- Global illumination
  - Shadows
  - Inter-object reflections
  - Rendering equation
  - Recursive ray tracing
  - ➢ More advanced ray tracing
  - Radiosity

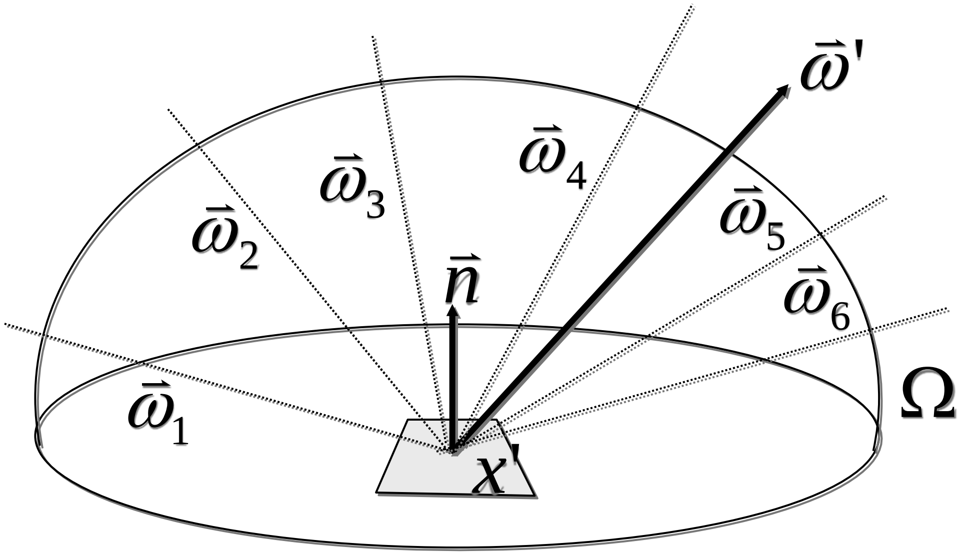
Greg Ward

Beyond Recursive Ray Tracing



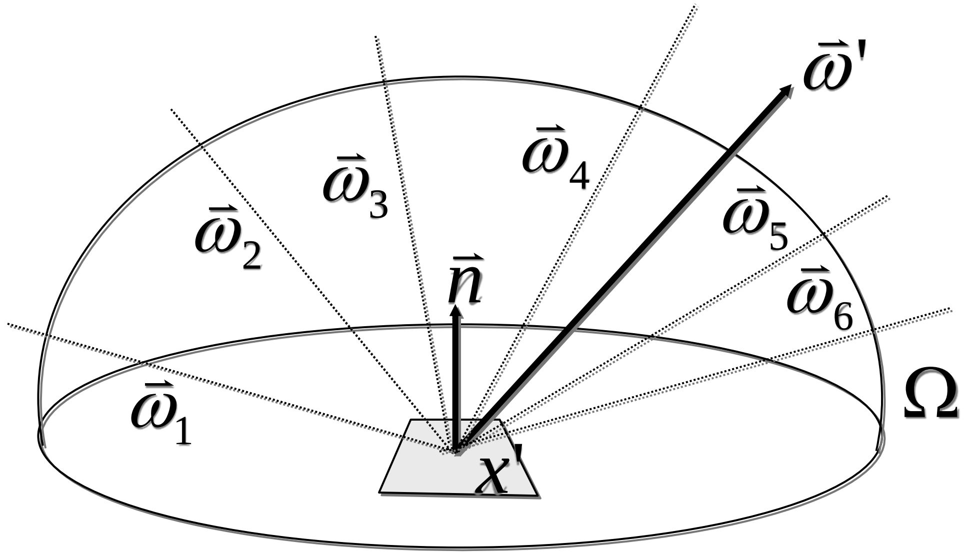
Distributed Ray Tracing

- Estimate integral for each reflection by sampling incoming directions



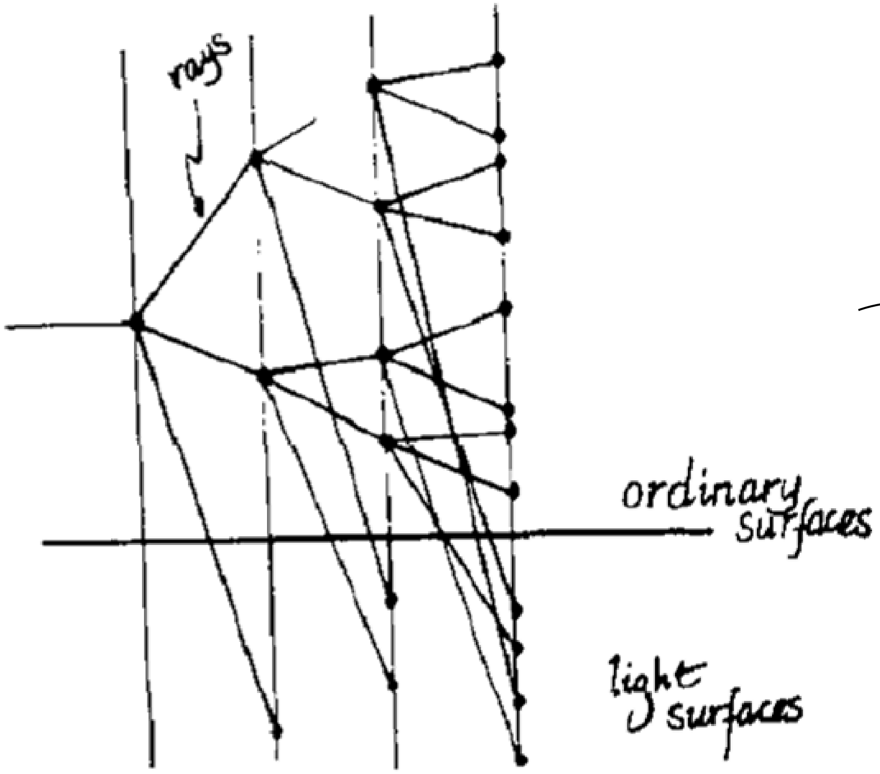
Distributed Ray Tracing

- Estimate integral for each reflection by sampling incoming directions

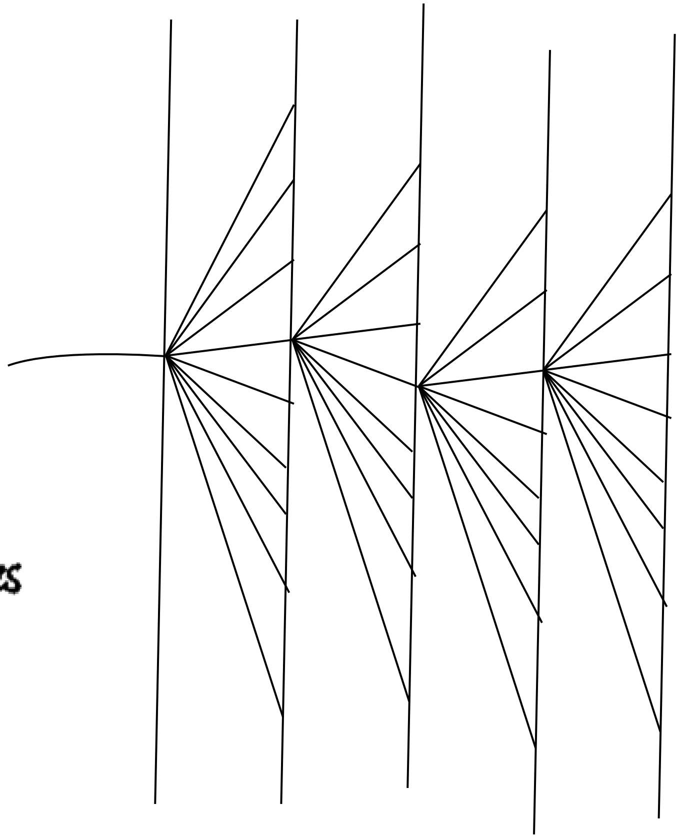


$$L_o(x',\vec{\omega}') = L_e(x',\vec{\omega}') + \sum_{\text{samples}} f_r(x',\vec{\omega},\vec{\omega}')(\vec{\omega}\cdot\vec{n})\,L_i(x',\vec{\omega})\,d\vec{\omega}$$

Ordinary Ray Tracing vs. Distribution Ray Tracing



Ray tracing

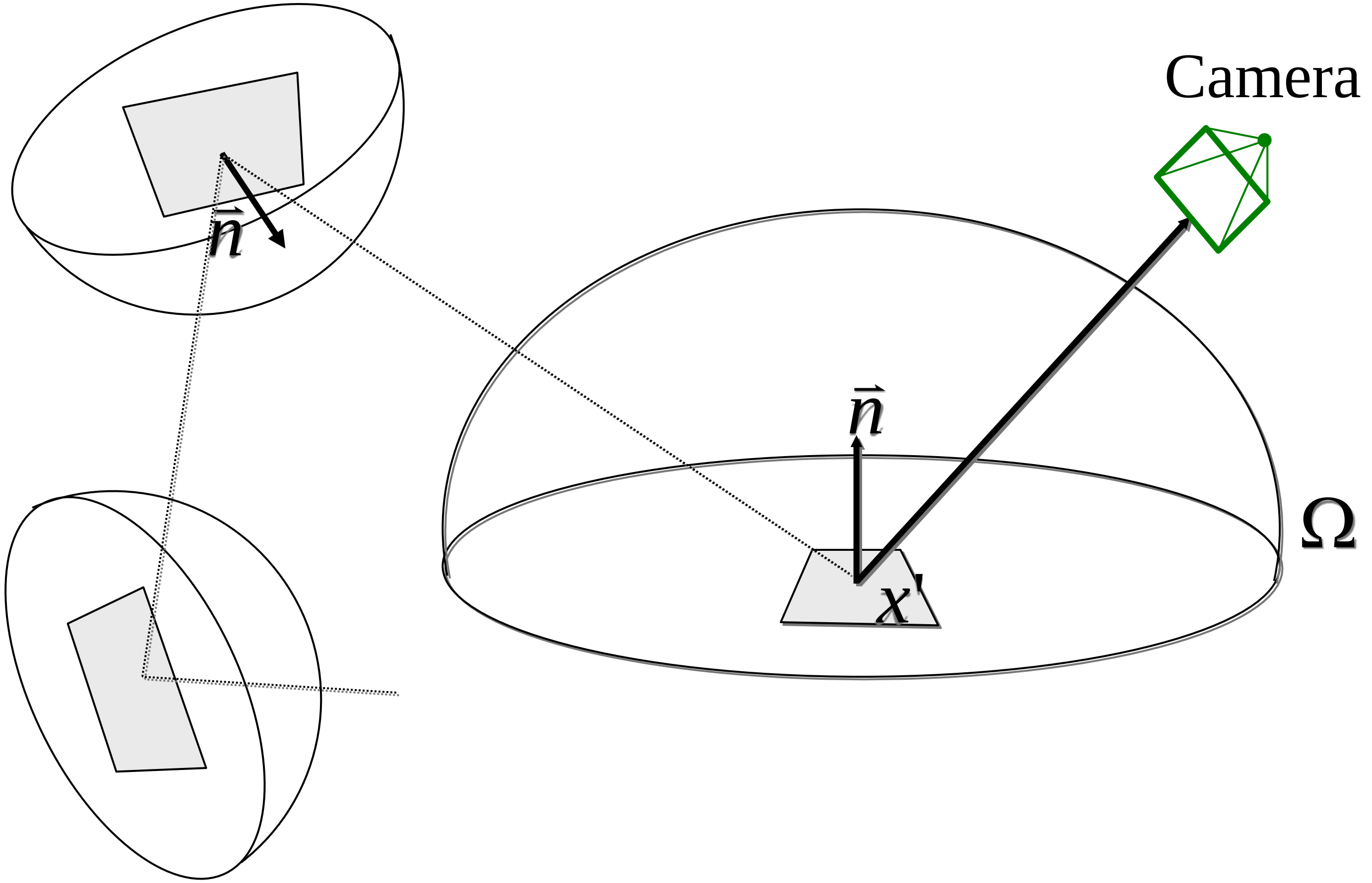


Distributed ray tracing

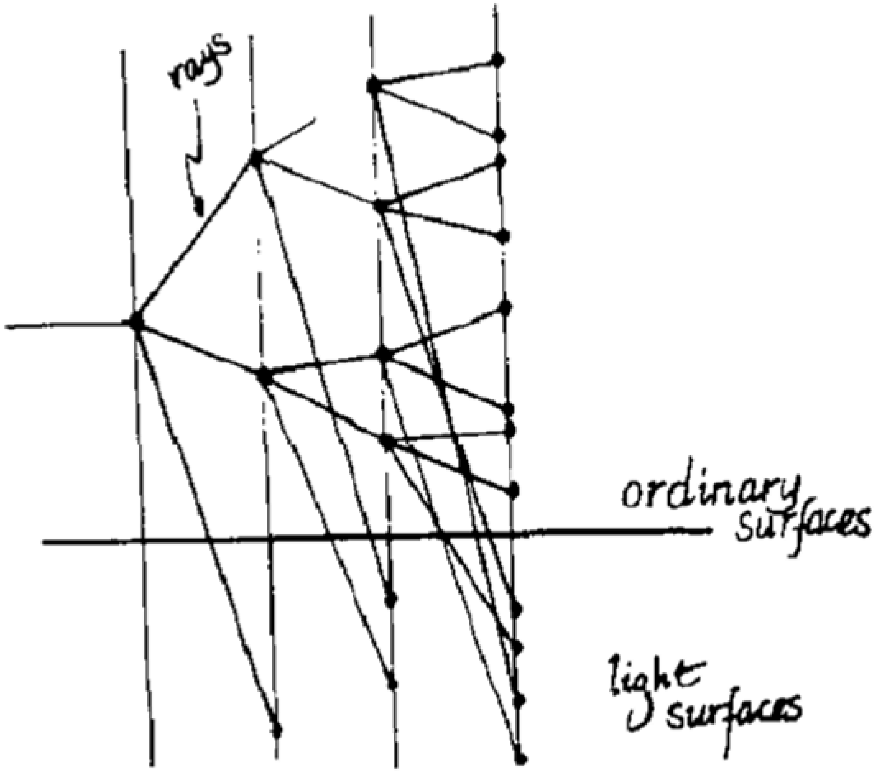
Kajiya

Monte Carlo Path Tracing

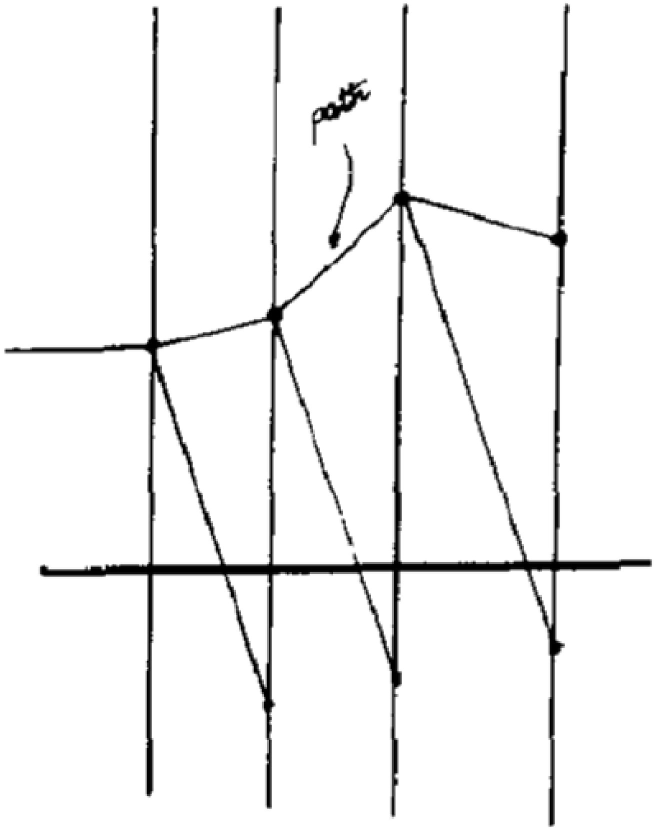
- Estimate integral for each pixel by sampling paths from camera



Ray Tracing vs. Path Tracing



Ray tracing



Path tracing

Kajiya

Overview

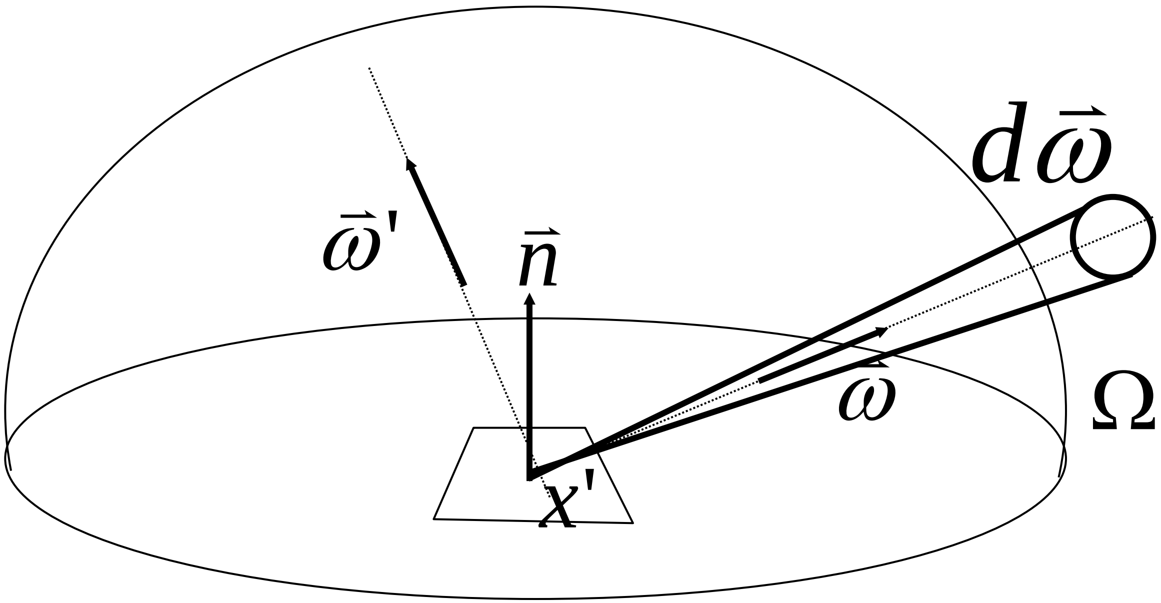
- Direct Illumination
  - Emission at light sources
  - Scattering at surfaces
- Global illumination
  - Shadows
  - Inter-object reflections
  - Rendering equation
  - Recursive ray tracing
  - More advanced ray tracing
  - ➢ Radiosity

Greg Ward

Radiosity

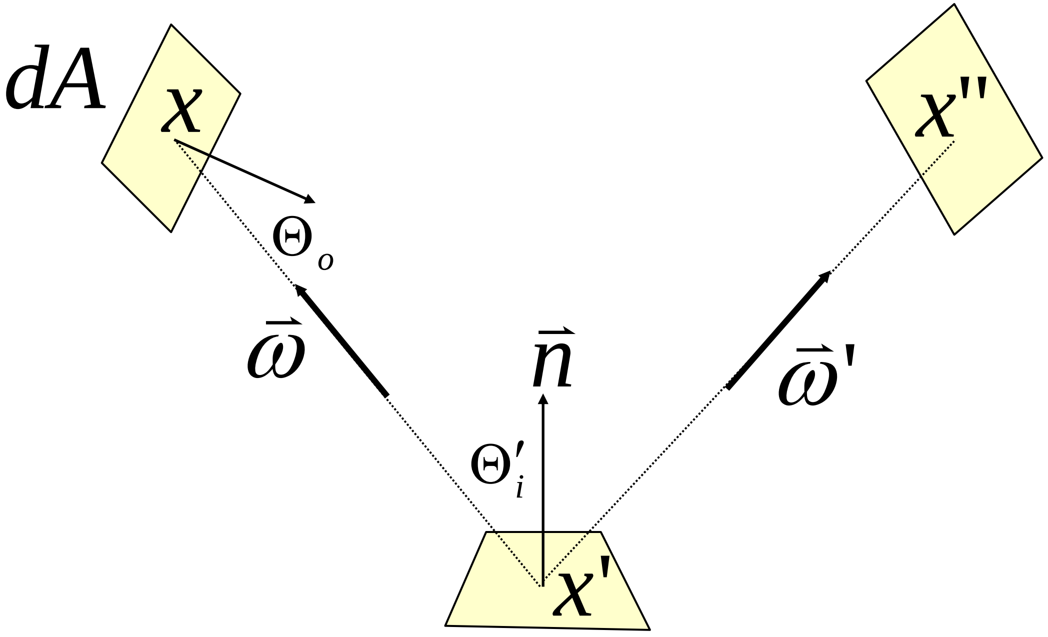
- Indirect diffuse illumination **– LD*E**

Rendering Equation (1)



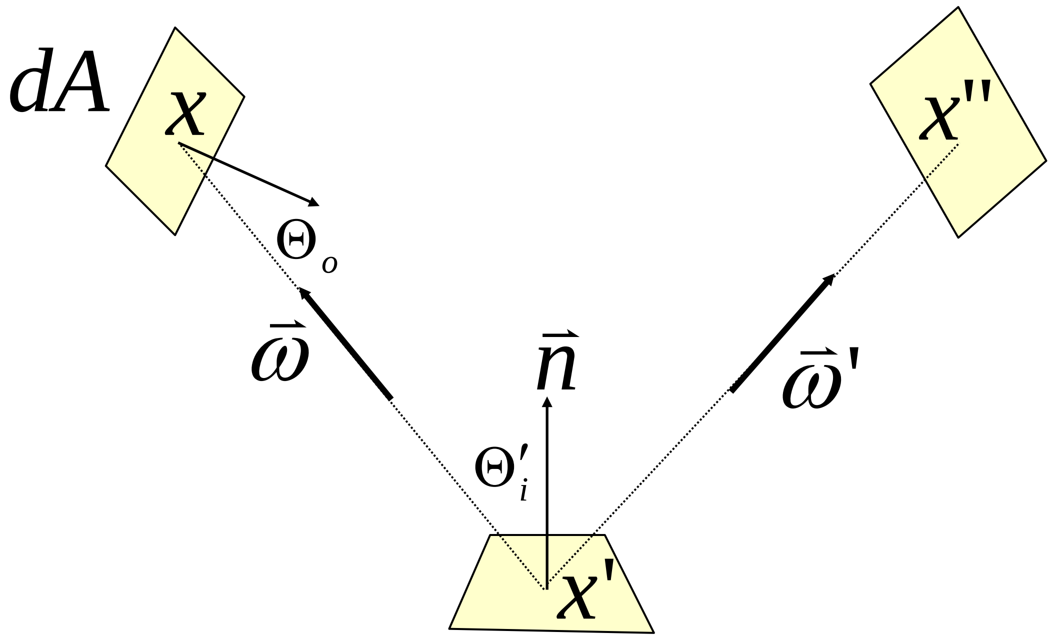
$$L_o(x',\vec{\omega}') = L_e(x',\vec{\omega}') + \int_{\Omega} f_r(x',\vec{\omega},\vec{\omega}')(\vec{\omega}\cdot\vec{n})\,L_i(x',\vec{\omega})\,d\vec{\omega}$$

Rendering Equation (2)



Kajiya 1986

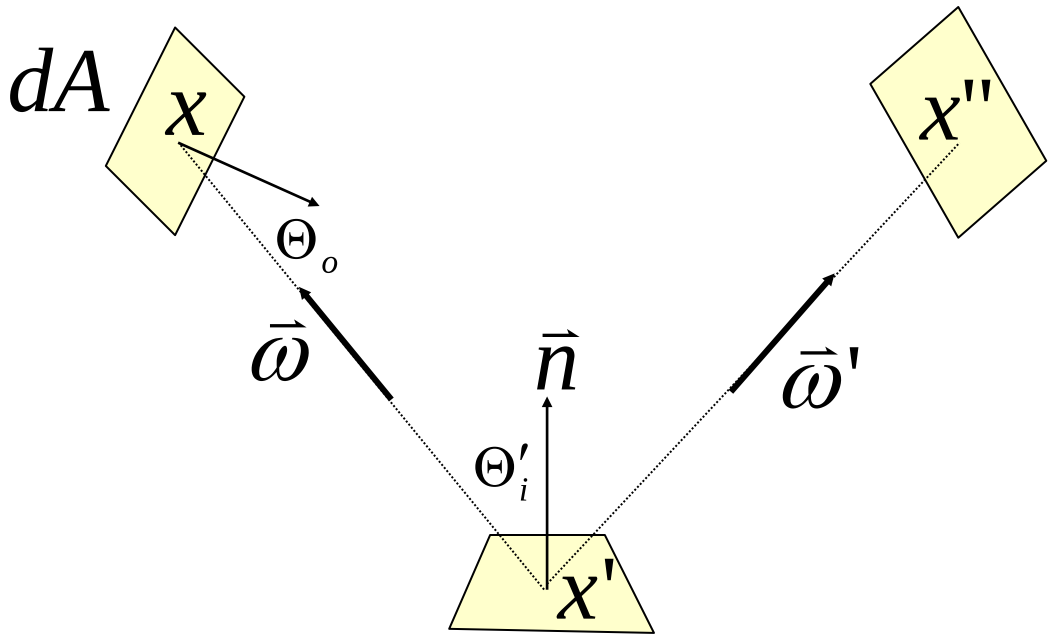
Rendering Equation (2)



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

Kajiya 1986

Rendering Equation (2)

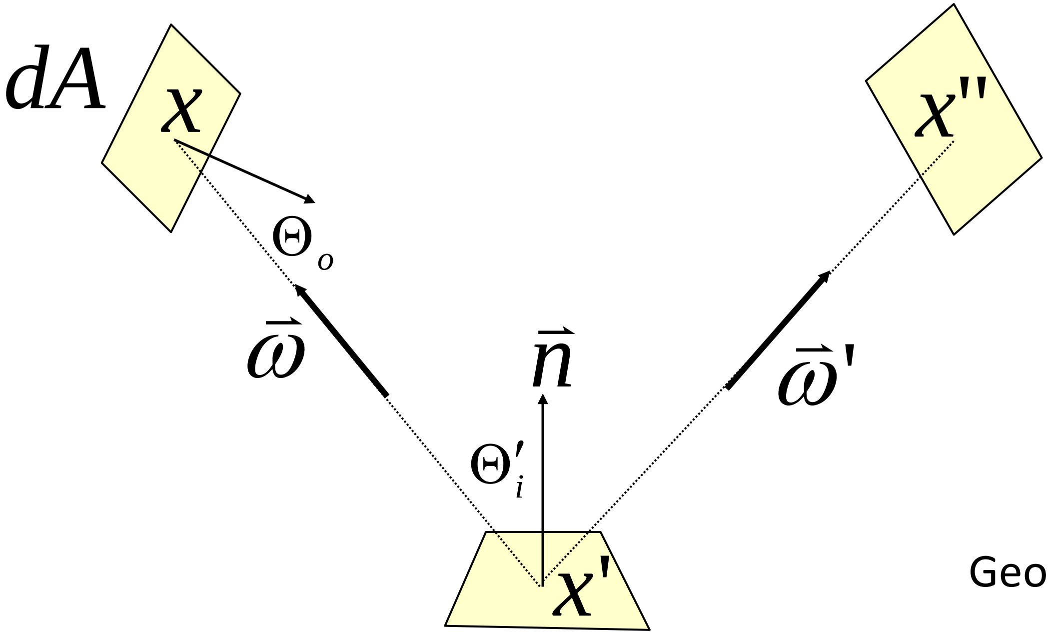


Visibility (V)

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

Kajiya 1986

Rendering Equation (2)



Geometry (G)

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

Visibility (V)

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

Kajiya 1986

Radiosity Equation

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

Assume everything is Lambertian

$$\boxed{\rho(x') = f_r(x \to x' \to x'')\pi}$$

$$L(x') = L_e(x') + \frac{\rho(x')}{\pi} \int_S L(x) V(x,x')\, G(x,x')\, dA$$

Convert to Radiosities

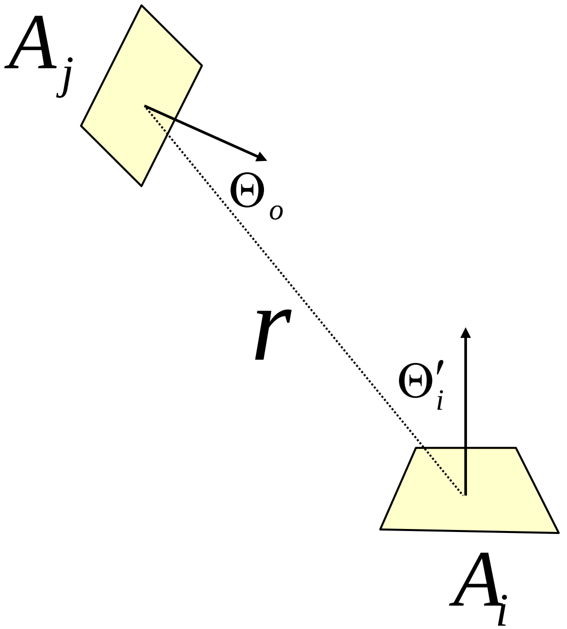
$$\boxed{B = \int_\Omega L_o \cos\theta\, d\omega} \quad \boxed{L = \frac{B}{\pi}}$$

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

Radiosity Approximation

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

Discretize the surfaces
into "elements"

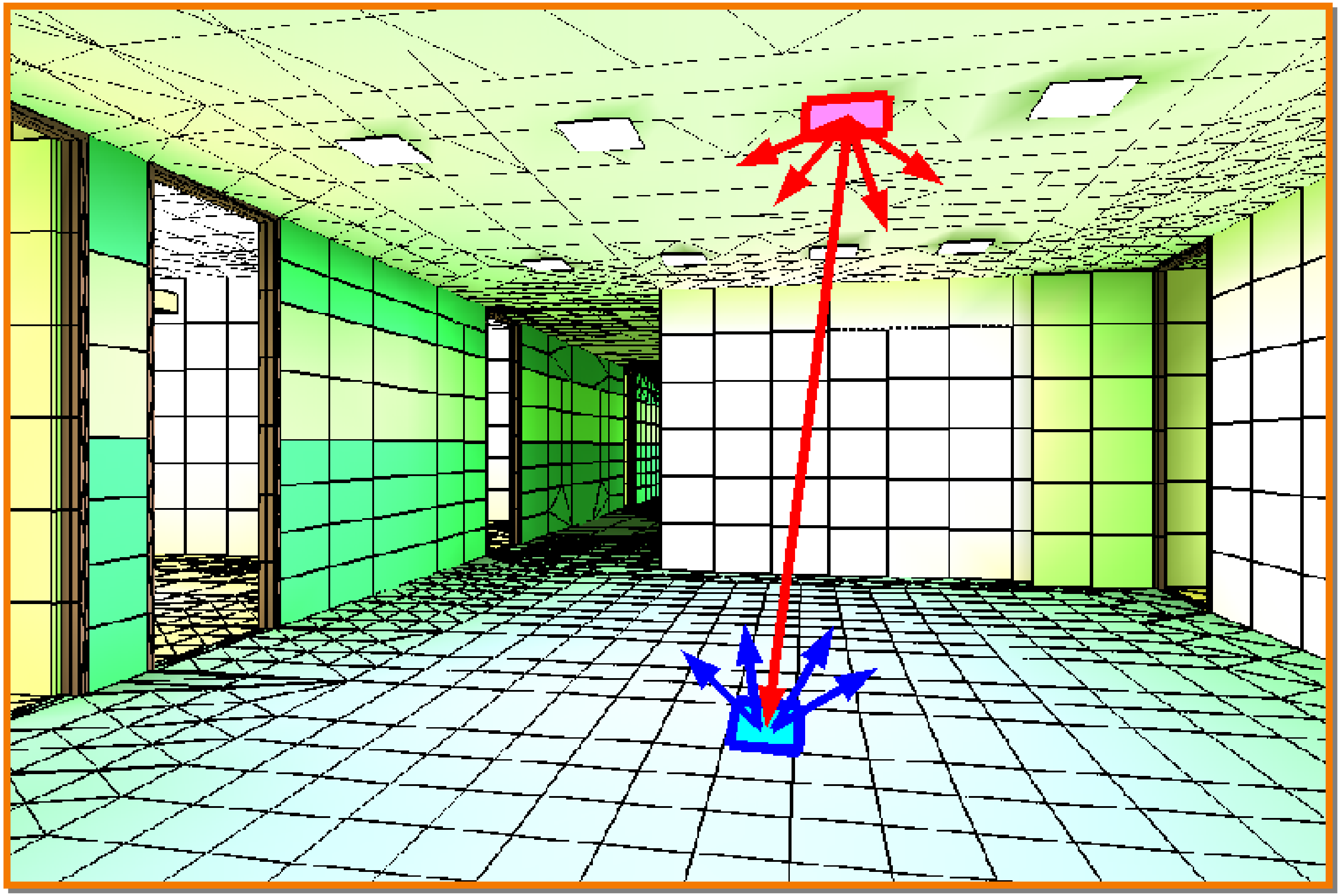


$$B_i = E_i + \rho_i \sum_{j=1}^{N} B_j F_{ij}$$

where "form factor" F is:

$$F_{ij} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{V_{ij} \cos\Theta_i' \cos\Theta_o}{\pi r^2} dA_j dA_i$$

Radiosity Approximation



Form Factor

1993

On the Form Factor between Two Polygons

Peter Schröder **Pat Hanrahan**

Department of Computer Science
Princeton University

Abstract

Form factors are used in radiosity to describe the fraction of diffusely reflected light leaving one surface and arriving at another. They are a fundamental geometric property used for computation. Many special configurations admit closed form solutions. However, the important case of the form factor between two polygons in three space has had no known closed form solution. We give such a solution for the case of general (planar, convex or concave, possibly containing holes) polygons.

CR Categories and Subject Descriptors: I.3.7 [Computer Graphics]: *Three-Dimensional Graphics and Realism – Radiosity*; J.2 [Physical Sciences and Engineering]: *Engineering*.

Additional Key Words and Phrases: Closed form solution; form factor; polygons.

1 Introduction

When using the radiosity techni
factor plays a central role. It

In this paper we present a formula for the form factor integral between two general polygons. The derivation of this formula is quite involved, and the interested reader is referred to [9] for a detailed derivation. The purpose of this paper is to bring this result to the attention of the graphics community.

2 Closed form solution

The form factor integral can be reduced to a double contour integral by two applications of Stokes' theorem [6]

$$\begin{aligned}\pi A_1 F_{12} &= \int_{A_1}\int_{A_2} \frac{\cos\theta_1 \cos\theta_2}{\|\vec{r}\|^2}\, dA_2\, dA_1 \\ &= \frac{1}{4}\int_{\partial A_1}\int_{\partial A_2} \ln(\vec{r}\cdot\vec{r})\, d\vec{x}_2 \cdot d\vec{x}_1\end{aligned}$$

where θ_1, θ_2 are the angles between the normal vector of the respective surface and a radius vector $\vec{r}$, which connects two points on the surfaces. The above equation holds for all surfaces such
he same contour of the

contour integral reduces

[5] Hanrahan, P., Salzman, D., and Aupperle, L. A Rapid Hierarchical Radiosity Algorithm. *Computer Graphics 25*, 4 (July 1991), 197–206.

[6] Herman, R. A. *A Treatise on Geometrical Optics*. Cambridge University Press, 1900.

[7] Lambert. *Photometria sive de mensura et gradibus luminis, colorum et umbrae*. 1760. German translation by E. Anding in *Ostwald's Klassiker der Exakten Wissenschaften*, Vol. 31-33, Leipzig, 1892.

System of Equations

$$B_i = E_i + \rho_i \sum_{j=1}^{N} B_j F_{ij}$$

$$E_i = B_i - \rho_i \sum_{j=1}^{N} B_j F_{ij}$$

$$B_i - \rho_i \sum_{j=1}^{N} B_j F_{ij} = E_i$$

$$\begin{bmatrix} 1-\rho_1 F_{1,1} & . & . & . & -\rho_1 F_{1,n} \\ -\rho_2 F_{2,1} & 1-\rho_2 F_{2,2} & . & . & -\rho_2 F_{2,n} \\ . & . & . & . & . \\ . & . & . & . & . \\ -\rho_{n-1} F_{n-1,1} & . & . & . & -\rho_{n-1} F_{n-1,n} \\ -\rho_n F_{n,1} & . & . & . & 1-\rho_n F_{n,n} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ . \\ . \\ . \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ . \\ . \\ . \\ E_n \end{bmatrix}$$

Compare with Direct Illumination

Hugo Elias, Wikipedia

Radiosity

- Application
  - Interior lighting design
  - **LD*E**
- Issues
  - Computing form factors
  - Solving large linear system of equations
  - Meshing surfaces into elements
  - Rendering images

Summary

- Global illumination
  - Rendering equation
- Solution methods
  - Sampling
    - Ray tracing
    - Distributed ray tracing
    - Monte Carlo path tracing
  - Discretization
    - Radiosity

Take-home message:

> Photorealistic rendering with global illumination is an integration problem