中国科学技术大学
University of Science and Technology of China

计算机图形学
Computer Graphics

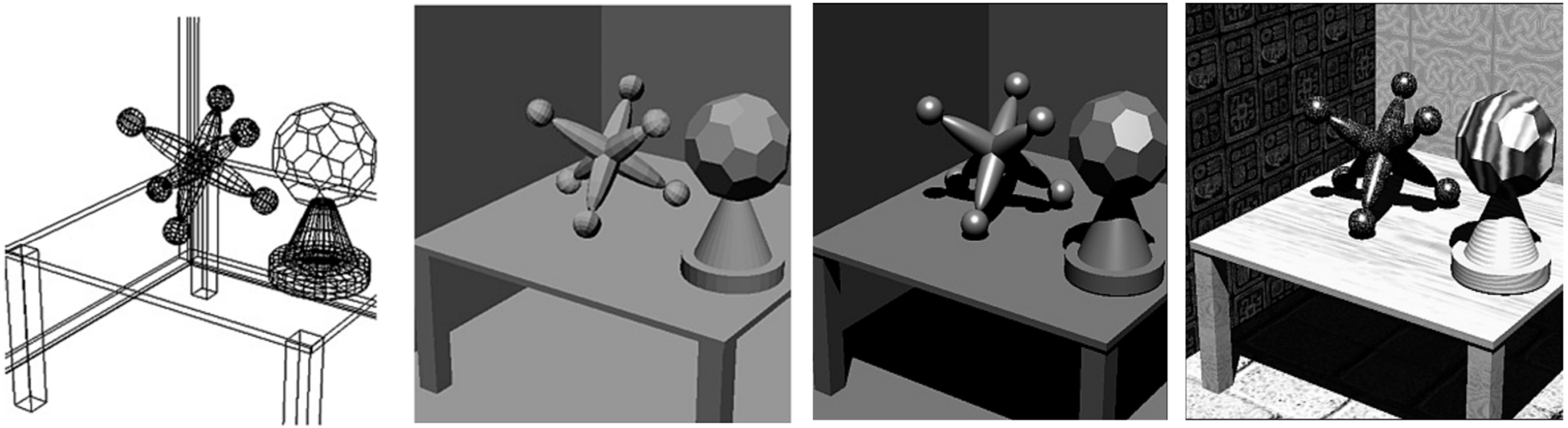
陈仁杰
renjiec@ustc.edu.cn
http://staff.ustc.edu.cn/~renjiec

真实感渲染
Realistic Rendering

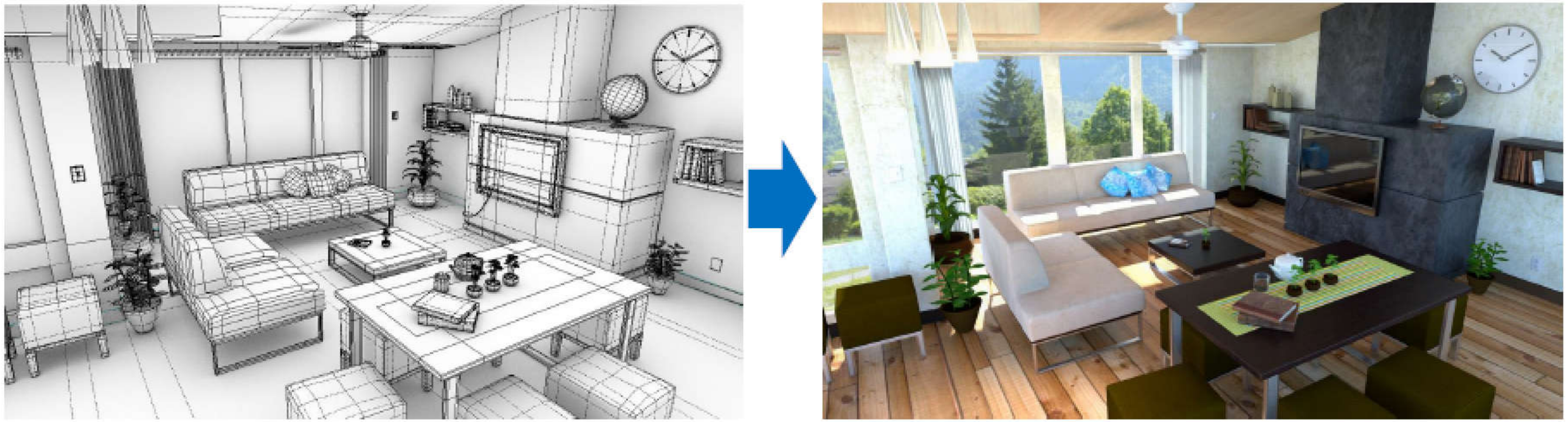
(Photorealistic Rendering)

Courtesy of Lingqi Yan, Rui Wang et al.

Rendering（渲染、绘制）

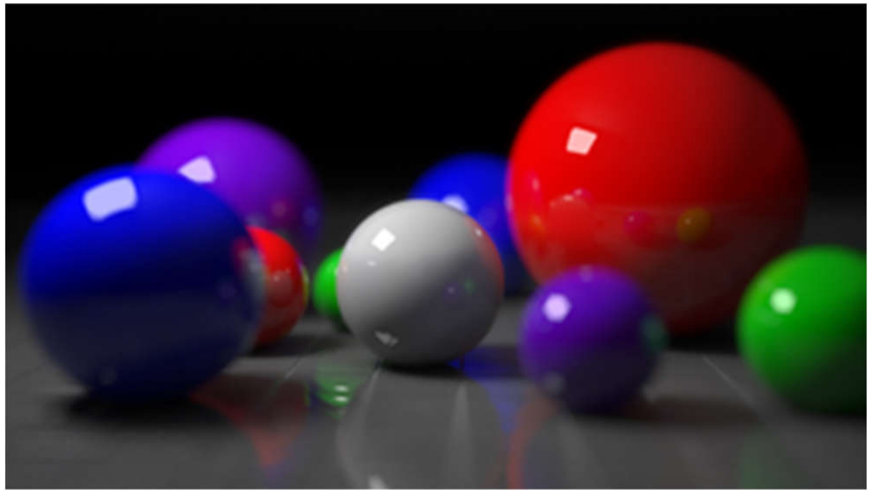


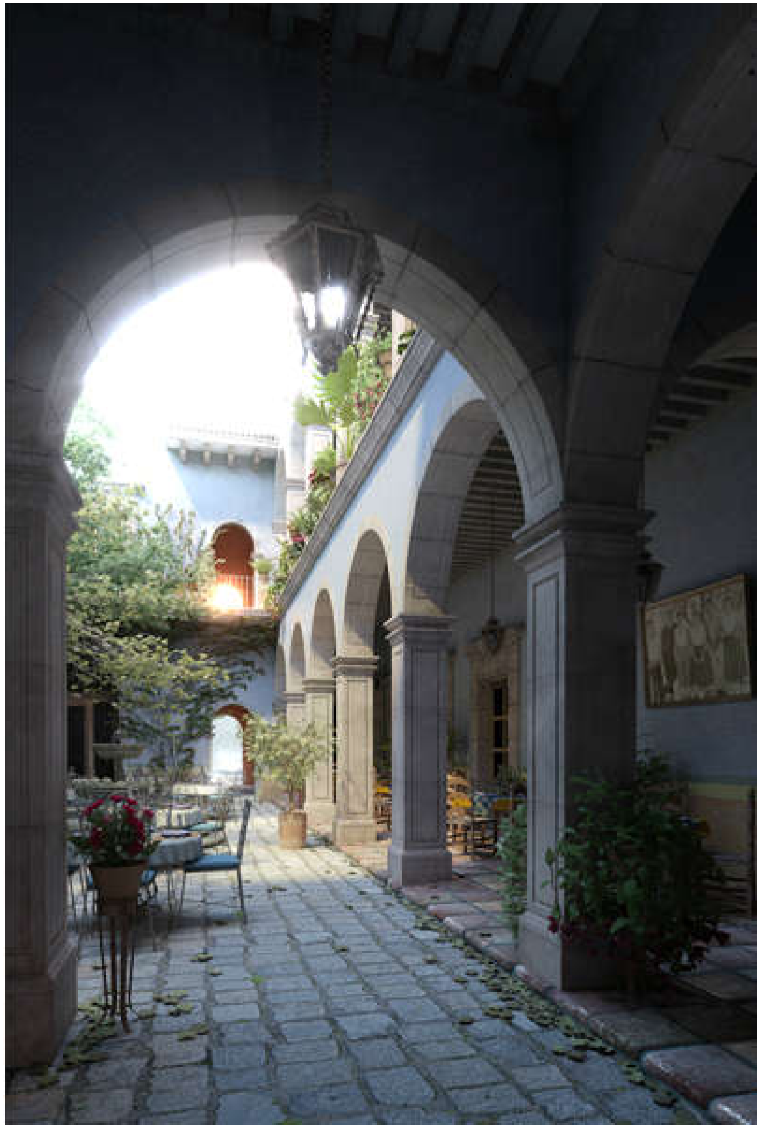
(a) (b) (c) (d)



Photorealistic Rendering

Realistic Rendering

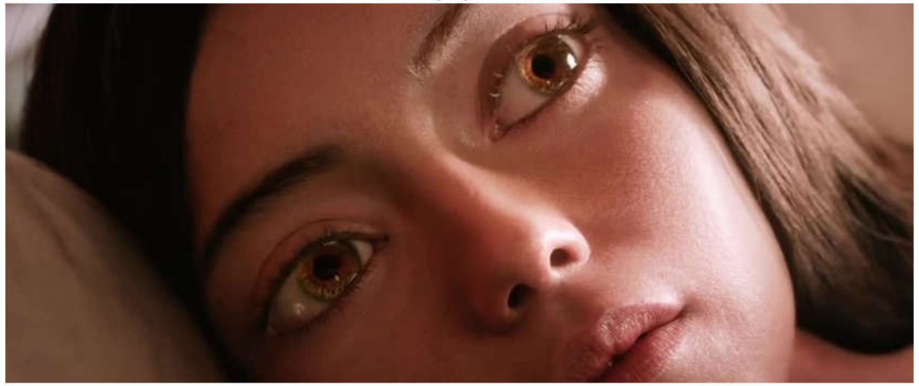




真实感渲染

(a)

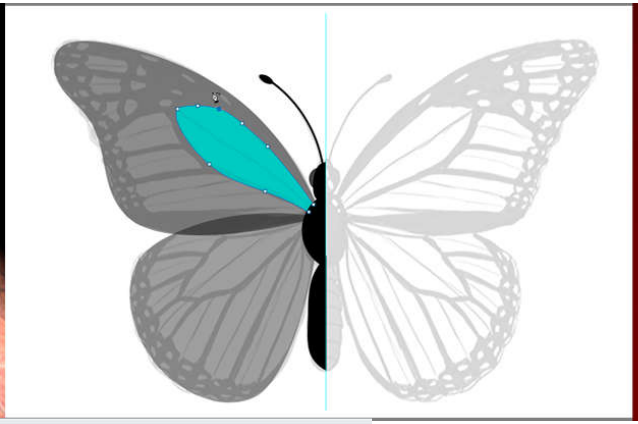
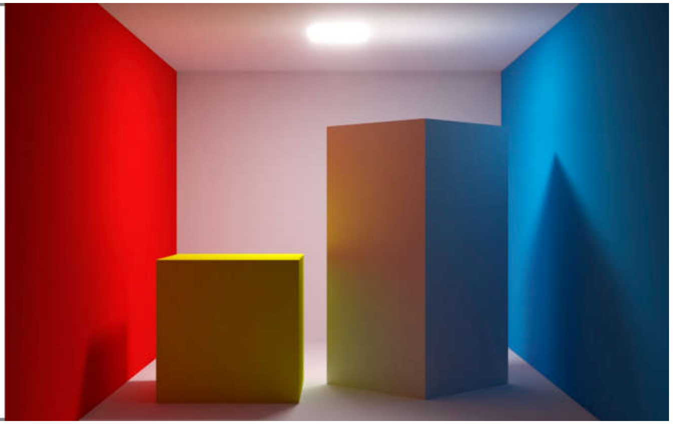
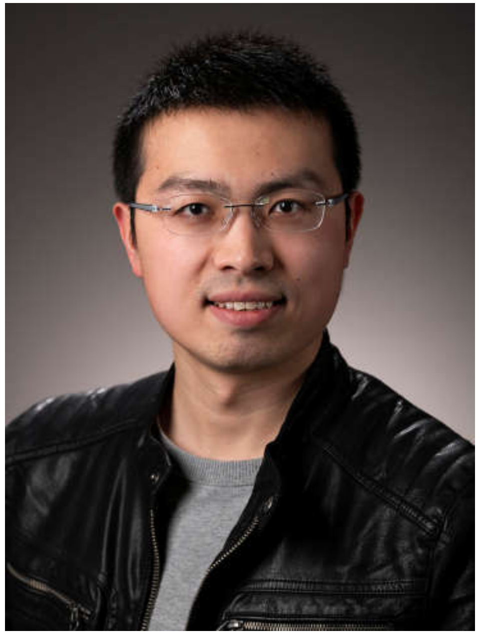
(b)



(c)

(d)

GAMES101: 现代计算机图形学入门

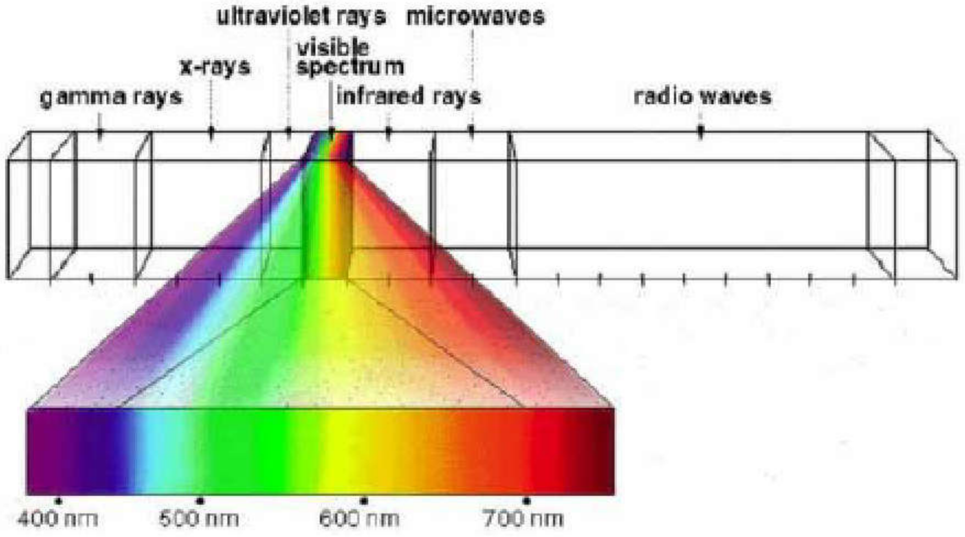
	日期	主题
第 1 周	Feb 11	**计算机图形学概述** [课件]
	Feb 14	**向量与线性代数** [课件] 阅读材料：第 2 章（Miscellaneous Math）；第 5 章（Linear Algebra）
第 2 周	Feb 18	**变换（二维与三维）** [课件] 阅读材料：第 6 章（Transformation Matrices），第 6.1、6.3 节
	Feb 21	**变换（模型、视图、投影）** [课件] [补充材料] 阅读材料：第 6 章（Transformation Matrices），第 6.2、6.4、6.5 节；第 7 章（Viewing）
第 3 周	Feb 25	**光栅化（三角形的离散化）** [课件] 阅读材料：第 3 章（Raster Images），第 3.1、3.2 节
	Feb 28	**光栅化（深度测试与抗锯齿）** [课件] 阅读材料：第 8 章（The Graphics Pipeline），第 8.2.3 节；第 9 章（Signal Processing）
第 4 周	Mar 3	**着色（光照与基本着色模型）** [课件] 阅读材料：第 10 章（Surface Shading），第 10.1 节
	Mar 7	**着色（着色频率、图形管线、纹理映射）** [课件] 阅读材料：第 10 章（Surface Shading），第 10.2 节；第 17 章（Using Graphics Hardware），第 17.1 节
第 5 周	Mar 10	**着色（插值、高级纹理映射）** [课件] 阅读材料：第 11 章（Texture Mapping），第 11.1、11.2 节
	Mar 13	**几何（基本表示方法）** [课件] 阅读材料：第 12 章（Data Structures for Graphics），第 12.1 节；第 22 章（Implicit Modeli
第 6 周	Mar 17	**几何（曲线与曲面）** [课件] 阅读材料：第 15 章（Curves），第 15.1、15.2、15.3、15.6 节
	Mar 20	**几何（网格处理）、阴影图** [课件] 阅读材料：无
第 7 周	Mar 24	**光线追踪（基本原理）** [课件] 阅读材料：第 4 章（Ray Tracing）
	Mar 27	**光线追踪（加速结构）** [课件] 阅读材料：第 12 章（Data Structures for Graphics），第 12.1、12.2、12.3 节
第 8 周	Mar 31	**光线追踪（辐射度量学、渲染方程与全局光照）** [课件] 阅读材料：第 18 章（Light），第 18.1、18.2 节
	Apr 5	**光线追踪（蒙特卡洛积分与路径追踪）** [课件] 阅读材料：无
第 9 周	Apr 7	**材质与外观** [课件] 阅读材料：第 24 章（Reflection），第 24.1、24.3 节
	Apr 10	**高级光线传播与复杂外观建模** [课件] 阅读材料：无
第 10 周	Apr 14	**相机与透镜** [课件] 阅读材料：无
	Apr 17	**光场、颜色与感知** [课件] 阅读材料：第 19 章（Color）
第 11 周	Apr 21	**动画与模拟（基本概念、质点弹簧系统、运动学）** [课件] 阅读材料：第 16 章（Computer Animation），第 16.5 节
	Apr 24	**动画与模拟（求解常微分方程，刚体与流体）** [课件] 阅读材料：无

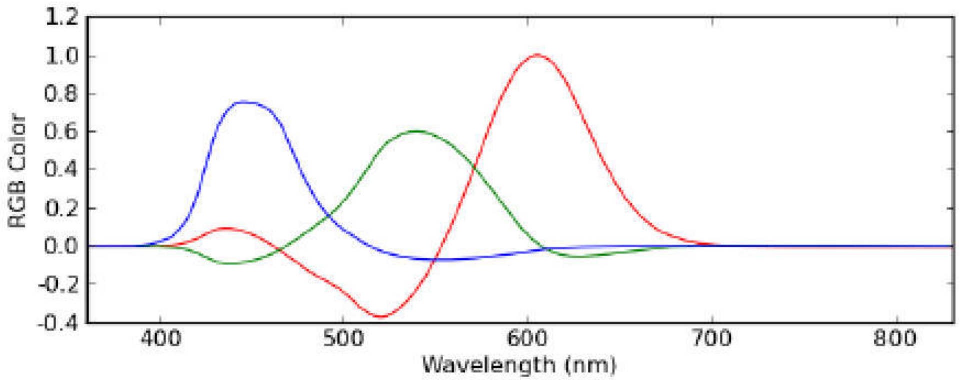
- https://sites.cs.ucsb.edu/~lingqi/teaching/games101.html

真实感渲染：
光的物理原理与仿真计算

What is color?

- Intensities of light at certain frequencies





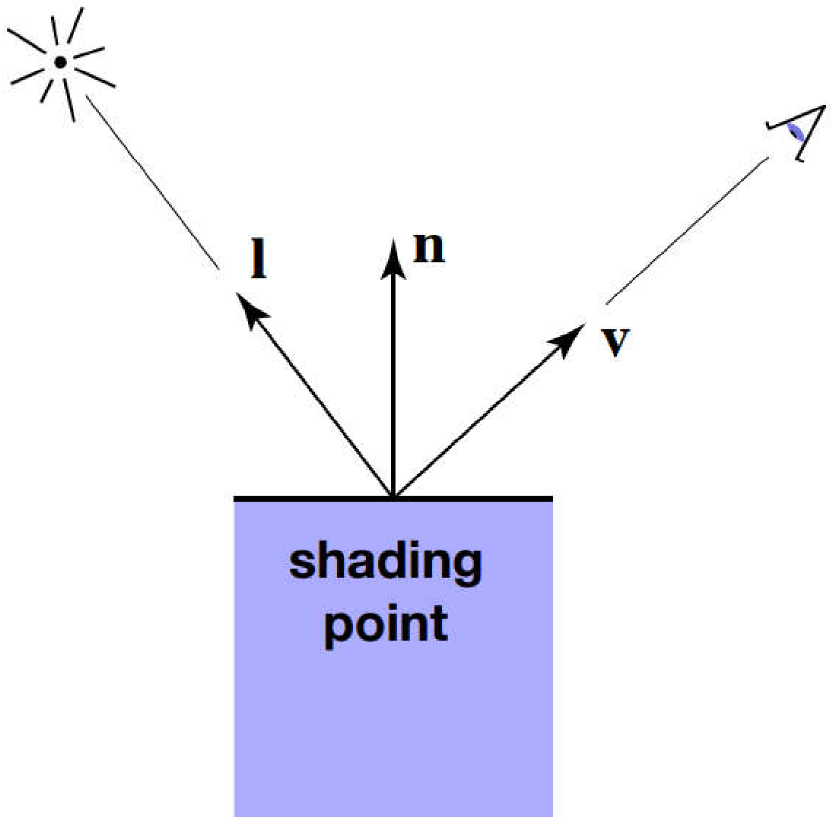
Recap: Local Shading Model

Credit by: Lingqi Yan

Local Shading

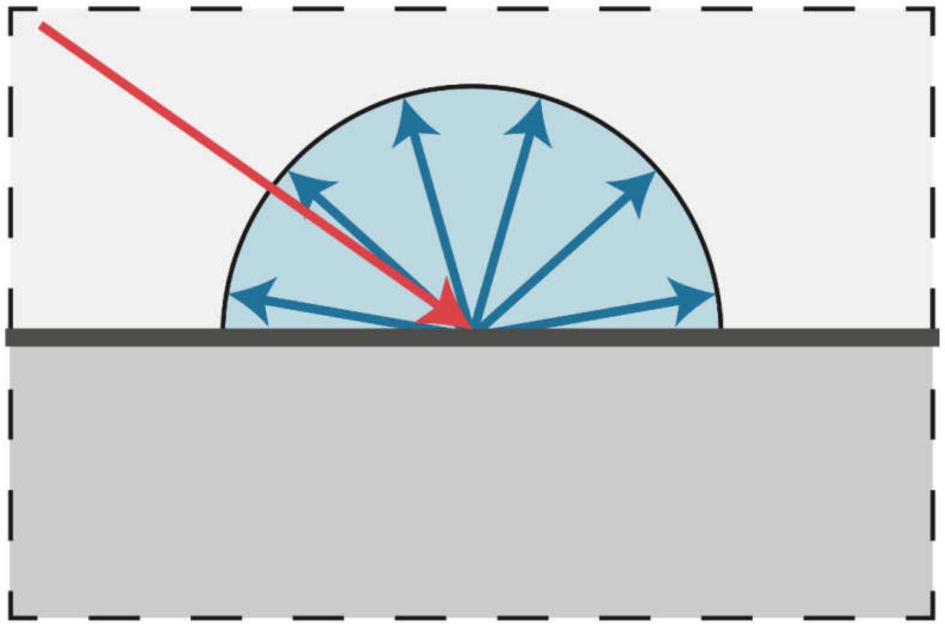
Inputs:

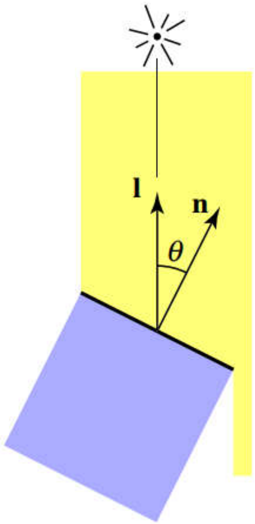
- Viewer direction, v
- Surface normal, n
- Light direction, l
  (for each of many lights)
- Surface parameters
  (color, shininess, …)



Diffuse Reflection

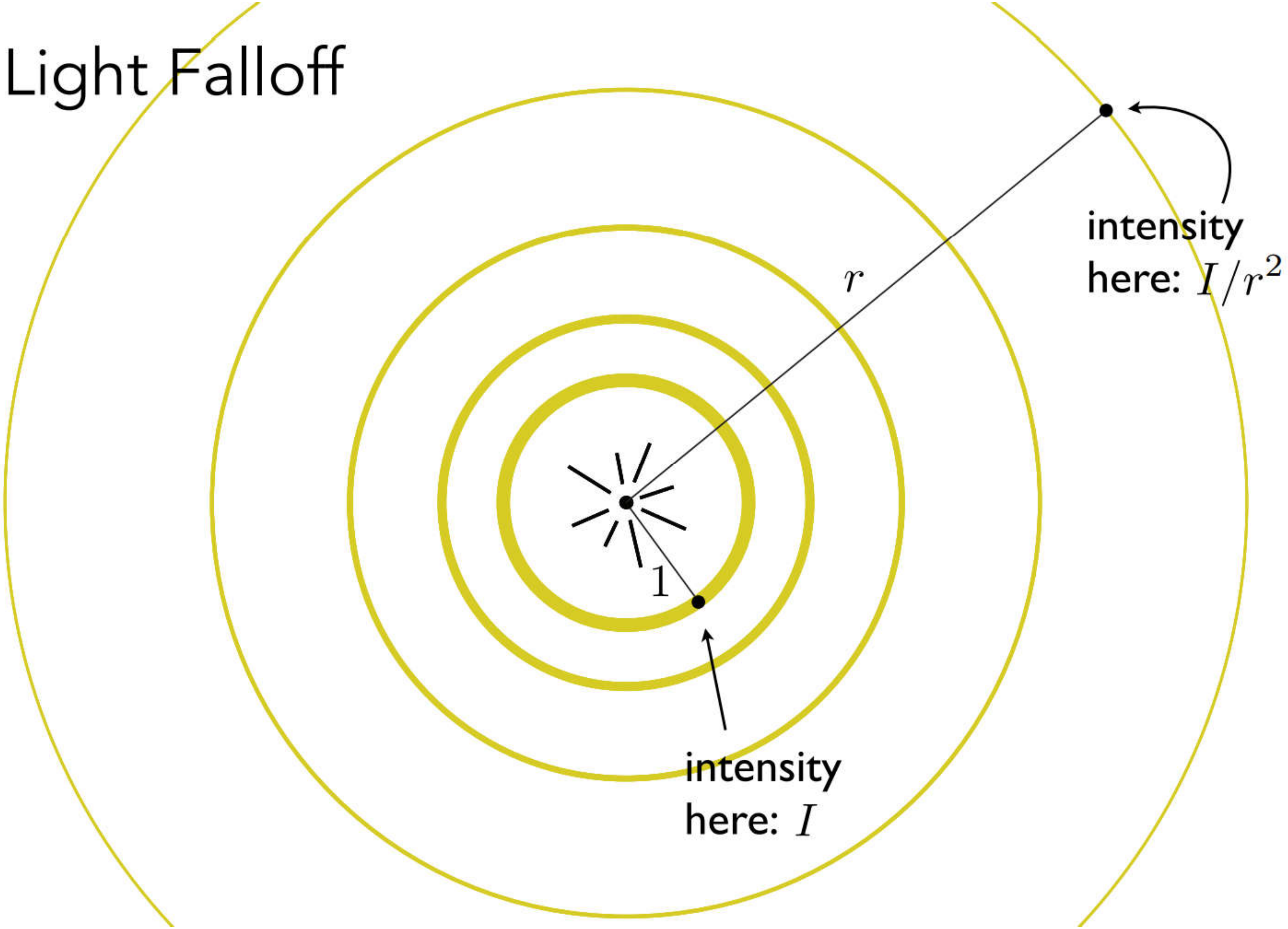
- Light is scattered uniformly in all directions
  - Surface color is the same for all viewing directions





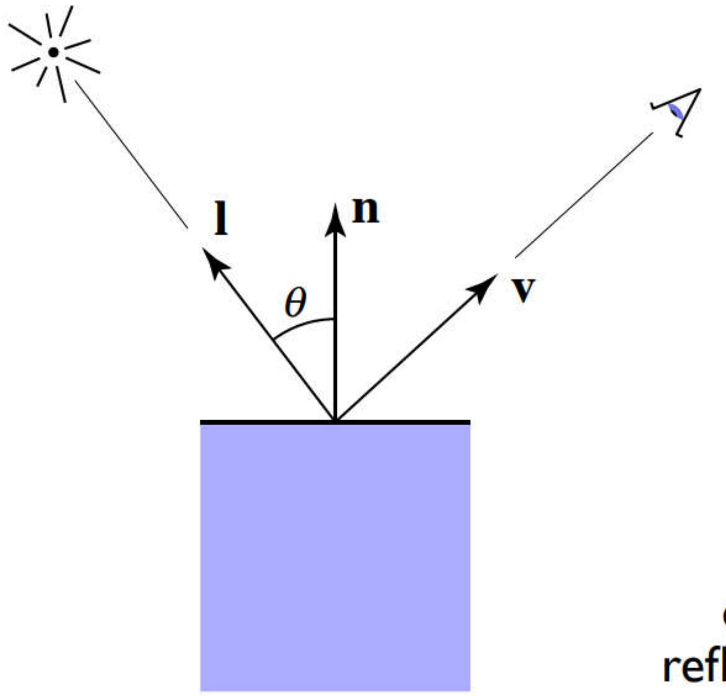
In general, light per unit area is proportional to $\cos\theta = \mathbf{l} \cdot \mathbf{n}$

Light Falloff



Lambertian (Diffuse) Shading

- Shading independent of view direction



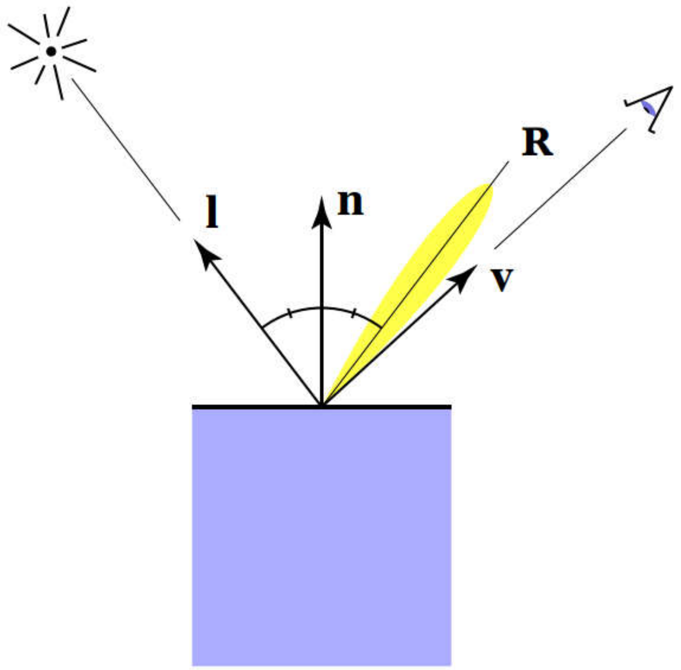
energy arrived at the shading point

$$L_d = k_d \, (I/r^2) \, \max(0, \mathbf{n} \cdot \mathbf{l})$$

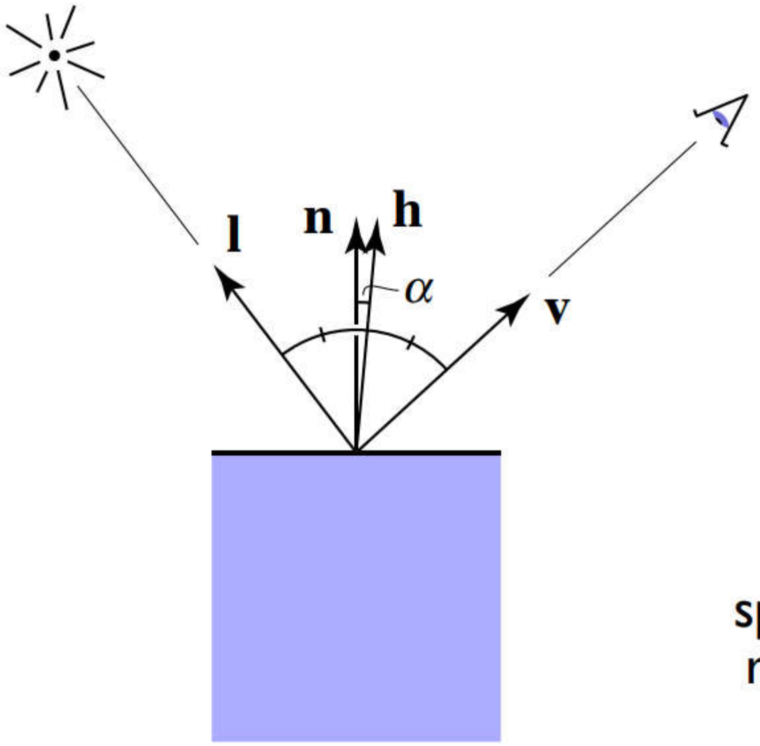
- L_d: diffusely reflected light
- k_d: diffuse coefficient (color)
- (I/r^2): energy arrived at the shading point
- $\max(0, \mathbf{n} \cdot \mathbf{l})$: energy received by the shading point

Specular Term

- Intensity depends on view direction
  - Bright near mirror reflection direction



Specular Term (Blinn-Phong)



$$\mathbf{h} = \mathrm{bisector}(\mathbf{v}, \mathbf{l})$$

(半程向量)

$$= \frac{\mathbf{v} + \mathbf{l}}{\|\mathbf{v} + \mathbf{l}\|}$$

$$L_s = k_s\,(I/r^2)\max(0, \cos\alpha)^p$$

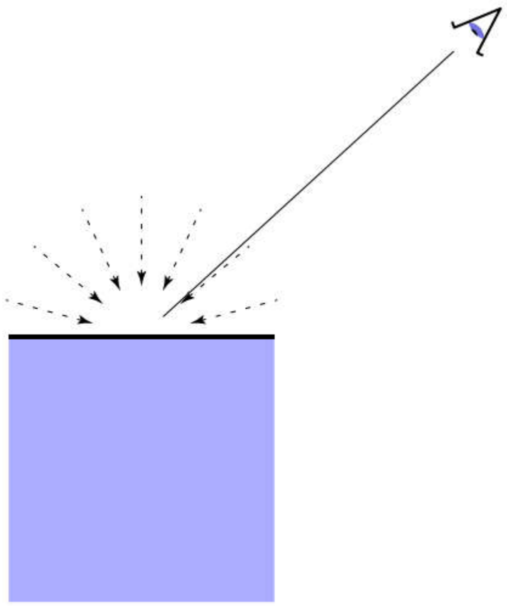
$$= k_s\,(I/r^2)\max(0, \mathbf{n} \cdot \mathbf{h})^p$$

specularly reflected light

specular coefficient

Ambient Term

- Shading that does not depend on anything
  - Add constant color to account for disregarded illumination and fill in black shadows

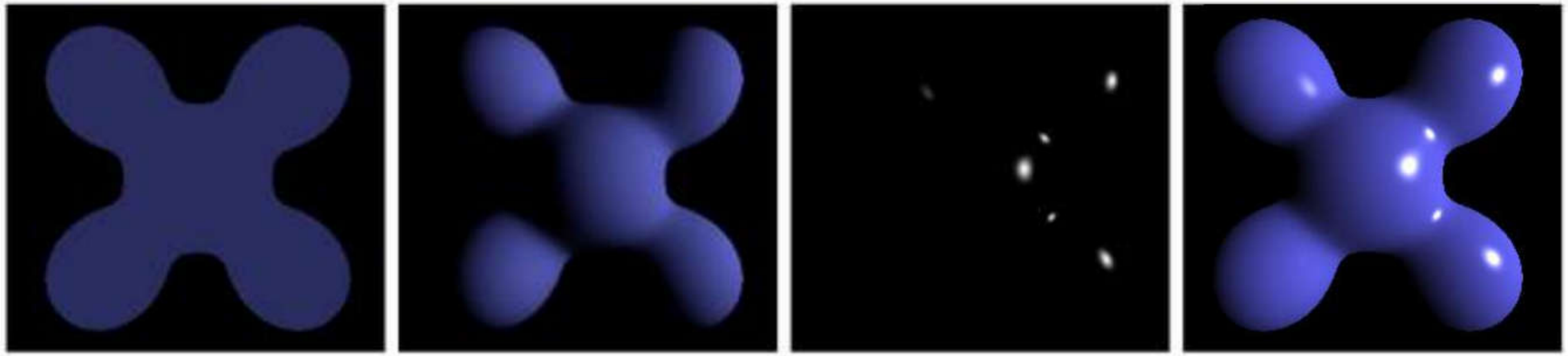


$$L_a = k_a \, I_a$$

reflected ambient light

ambient coefficient

Blinn-Phong Shading Model

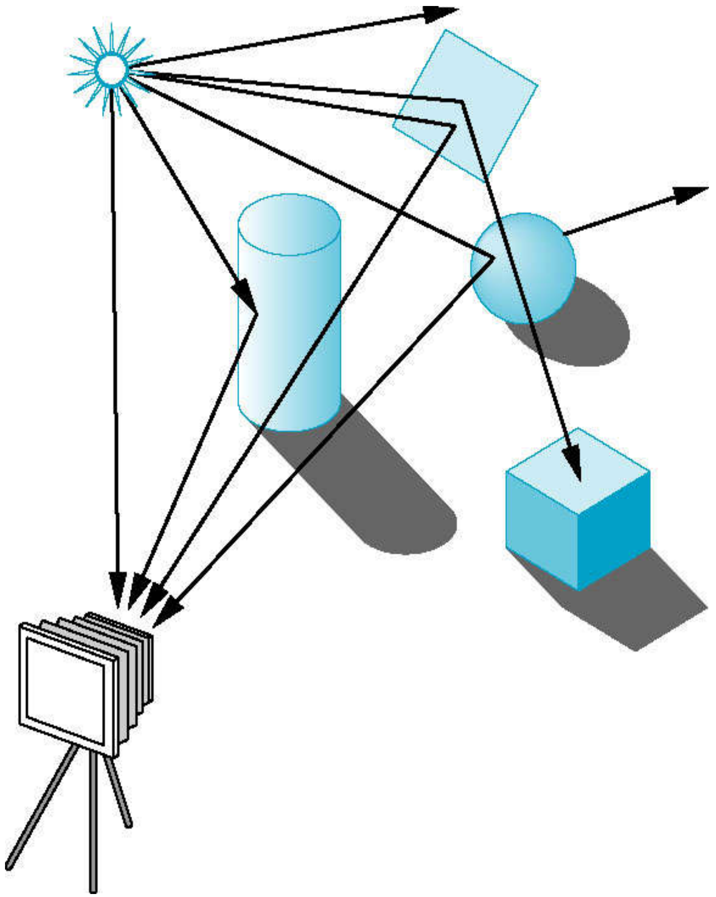


Ambient + Diffuse + Specular = Blinn-Phong Reflection

$$
\begin{aligned}
L &= L_a + L_d + L_s \\
&= k_a\, I_a + k_d\, (I/r^2)\max(0, \mathbf{n}\cdot\mathbf{l}) + k_s\, (I/r^2)\max(0, \mathbf{n}\cdot\mathbf{h})^p
\end{aligned}
$$

局部光照模型的不足

- 过于简化的反射模型：复杂材质
- 间接反射光
- 其他
  - 遮挡
  - 阴影
  - …



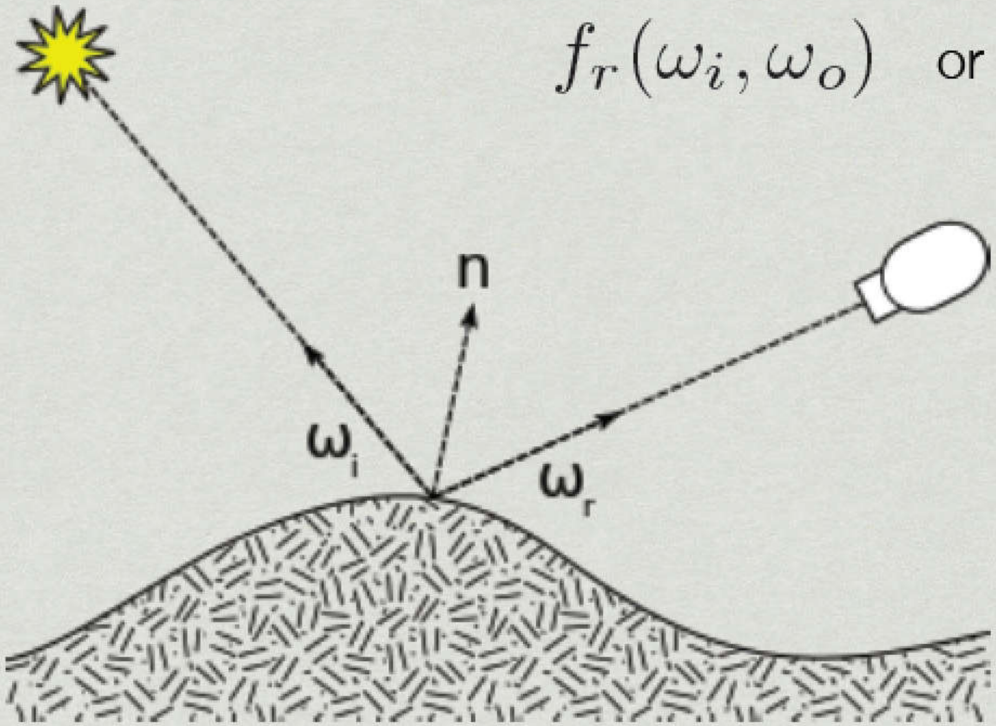
全局光照模型

材质：物体对光的视觉反应

Material = BRDF

Bidirectional Reflectance Distribution Function

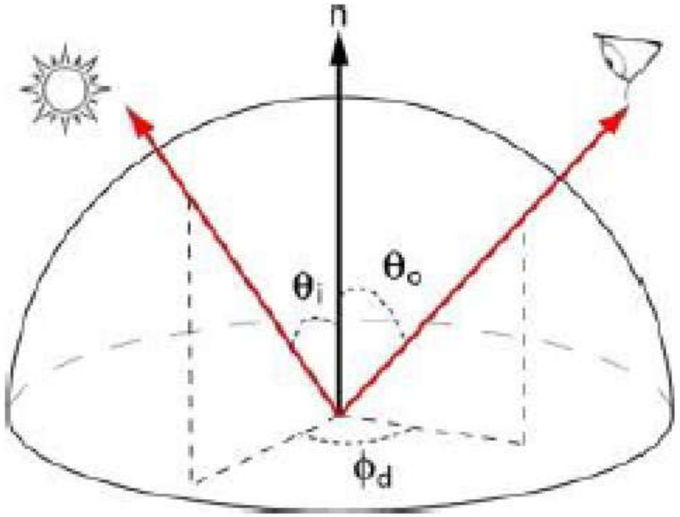
BRDF



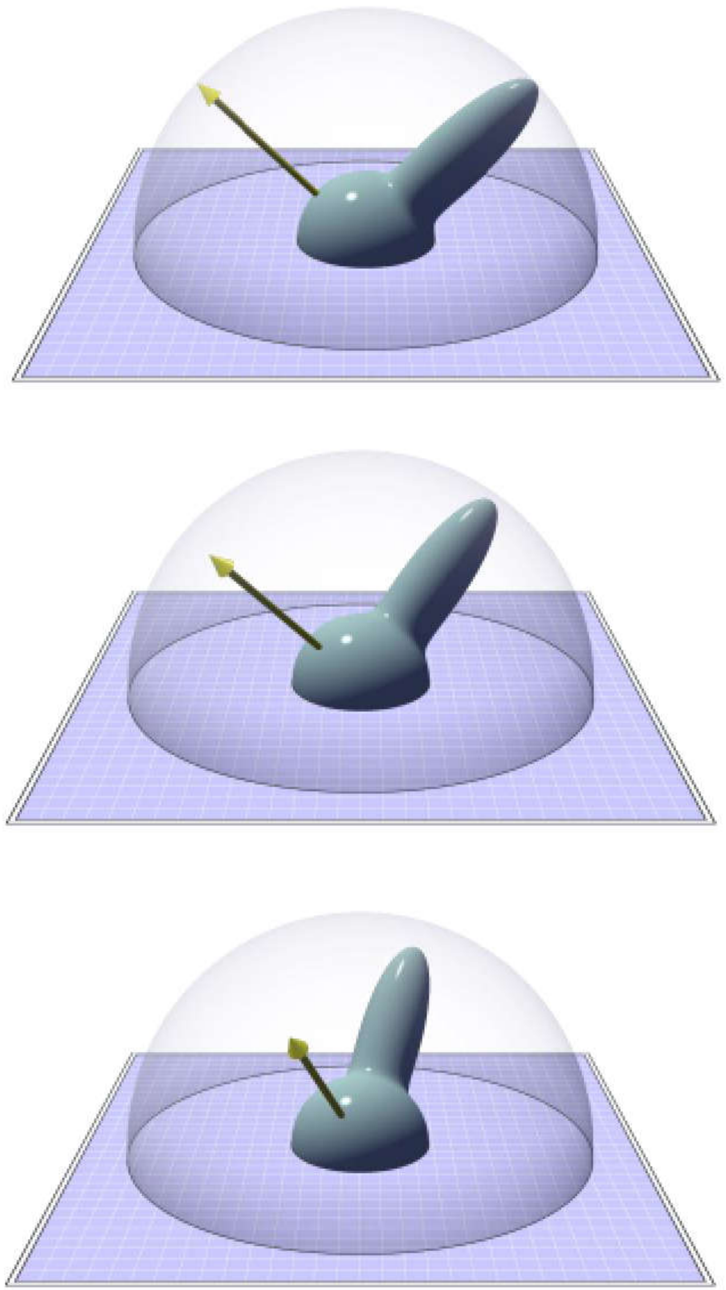
$$f_r(\omega_i, \omega_o) \quad \text{or} \quad f_r(\theta_i, \phi_i, \theta_o, \phi_o)$$

- At any position, it represents how much light is reflected into each outgoing direction from each incoming direction
- 4D function
  2D for incoming direction
  2D for outgoing direction

BRDF

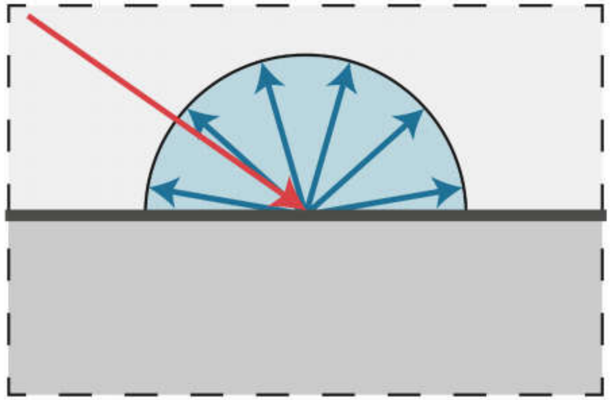


$$f_r(x, \omega_i \rightarrow \omega_o).$$

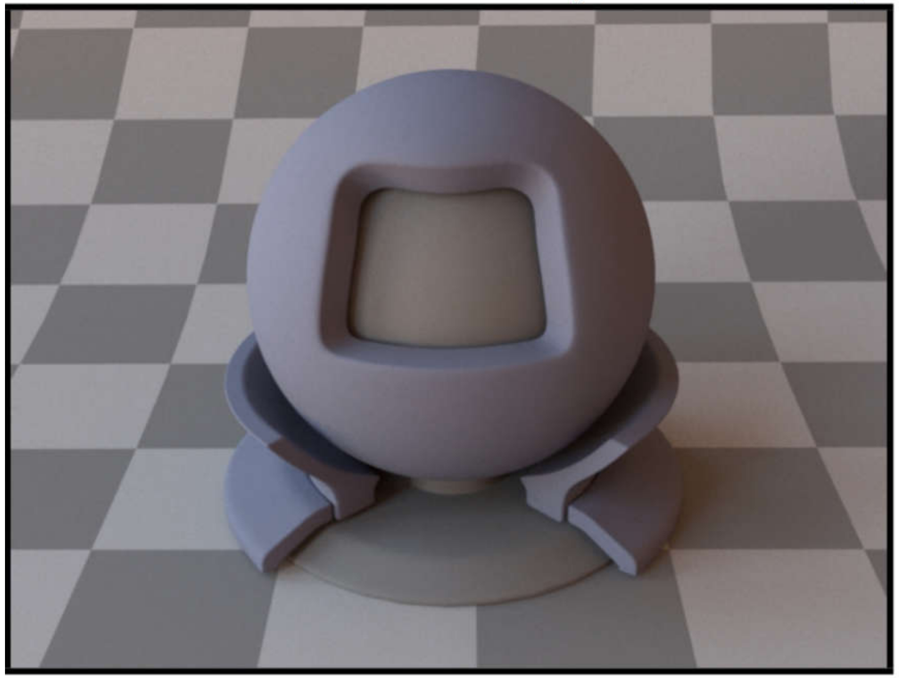


Lambertian Material

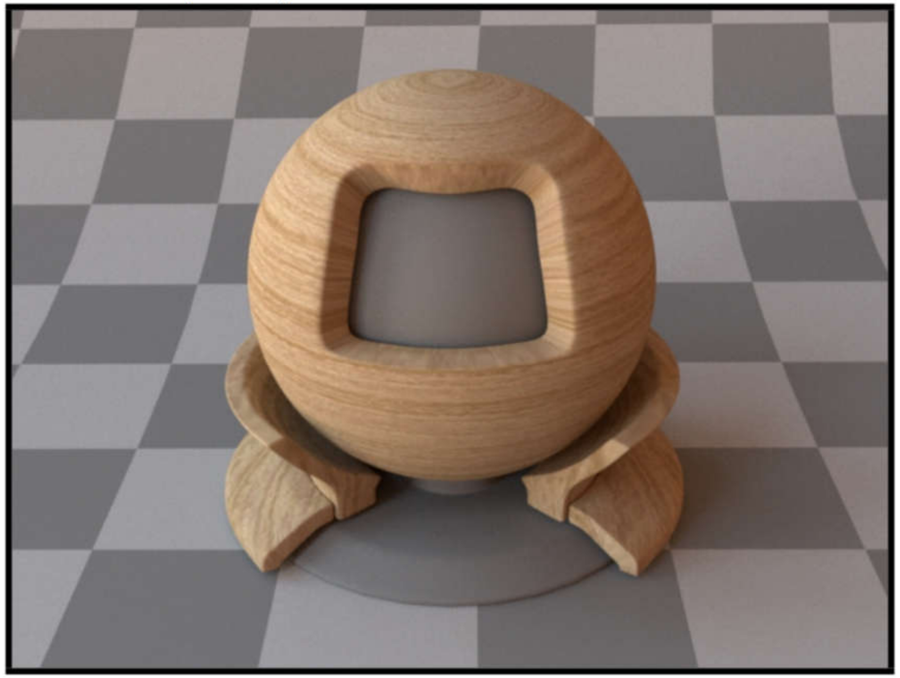
- Light is equally reflected in each output direction



[Mitsuba renderer, Wenzel Jakob, 2010]



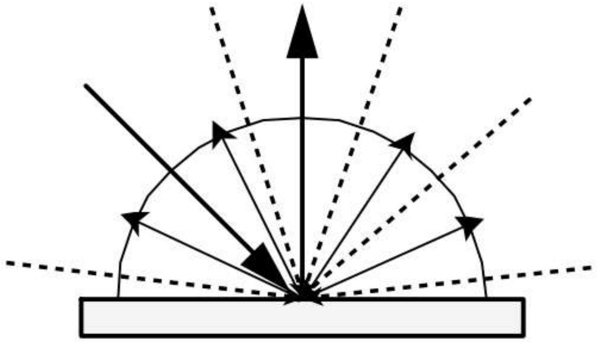
Uniform colored diffuse BRDF



Textured diffuse BRDF

Lambertian Material

- Suppose the incident lighting is uniform:

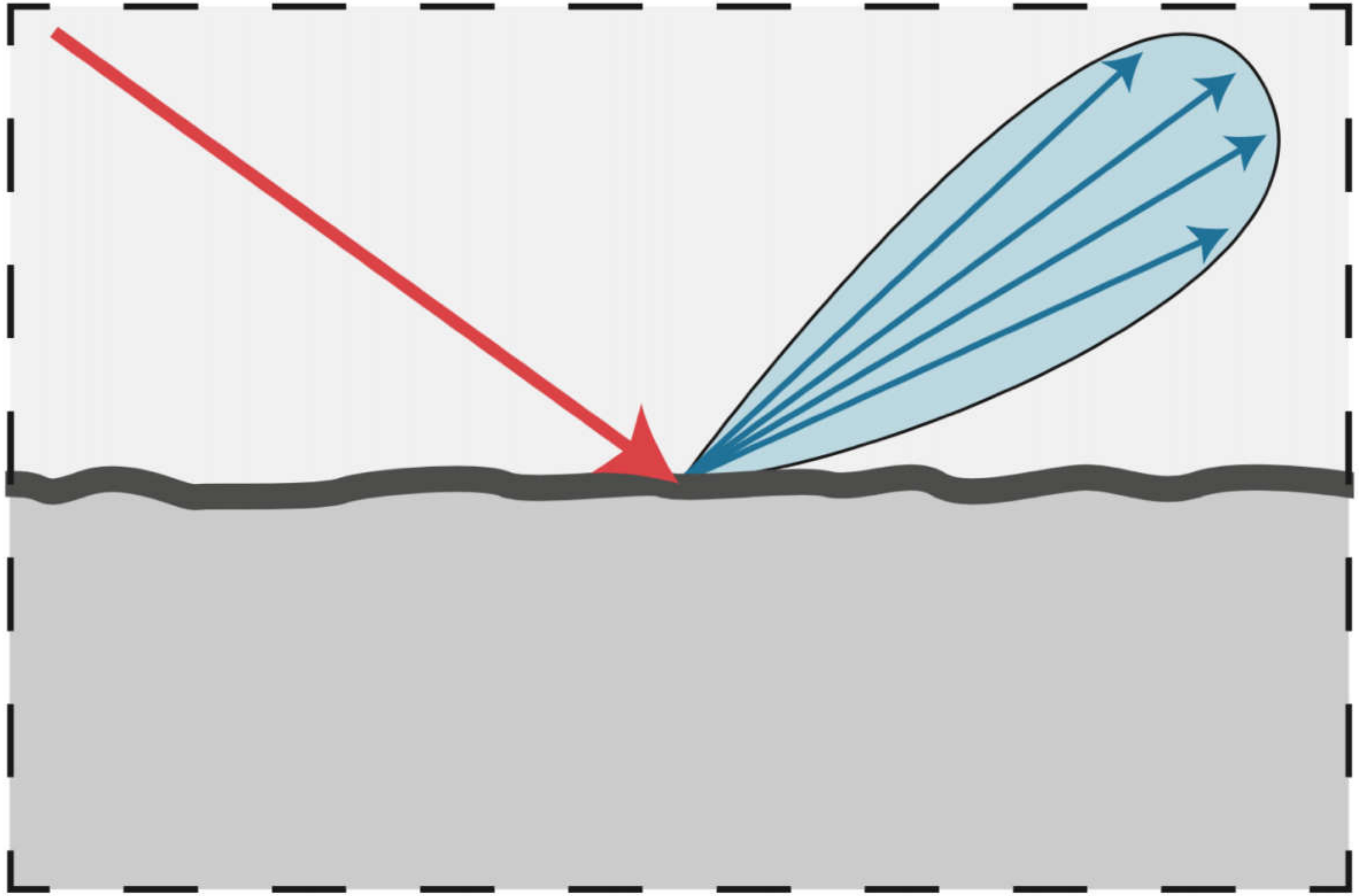


$$f_r = c$$

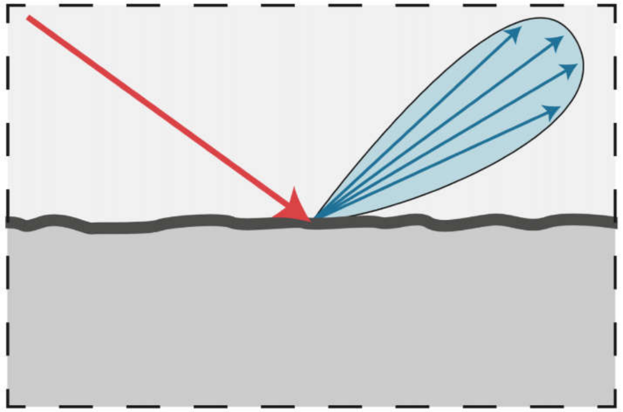
$$\begin{aligned} L_o(\omega_o) &= \int_{H^2} f_r L_i(\omega_i) \cos\theta_i \, \mathrm{d}\omega_i \\ &= f_r L_i \int_{H^2} \cancel{(\omega_i)} \cos\theta_i \, \mathrm{d}\omega_i \\ &= \pi f_r L_i \end{aligned}$$

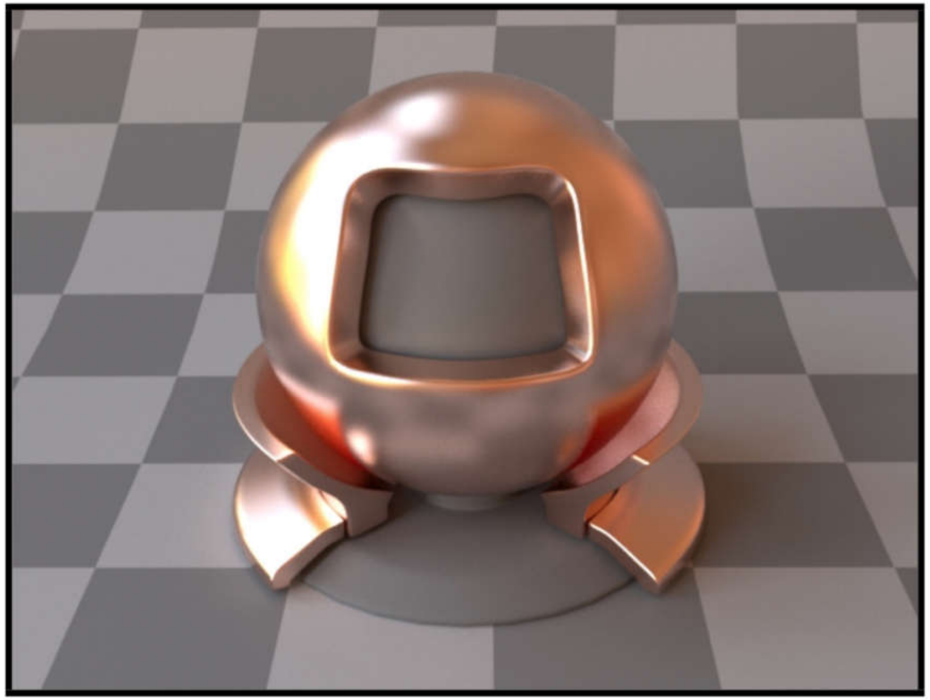
$$f_r = \frac{\rho}{\pi}$$ — albedo (color)

Glossy material (BRDF)

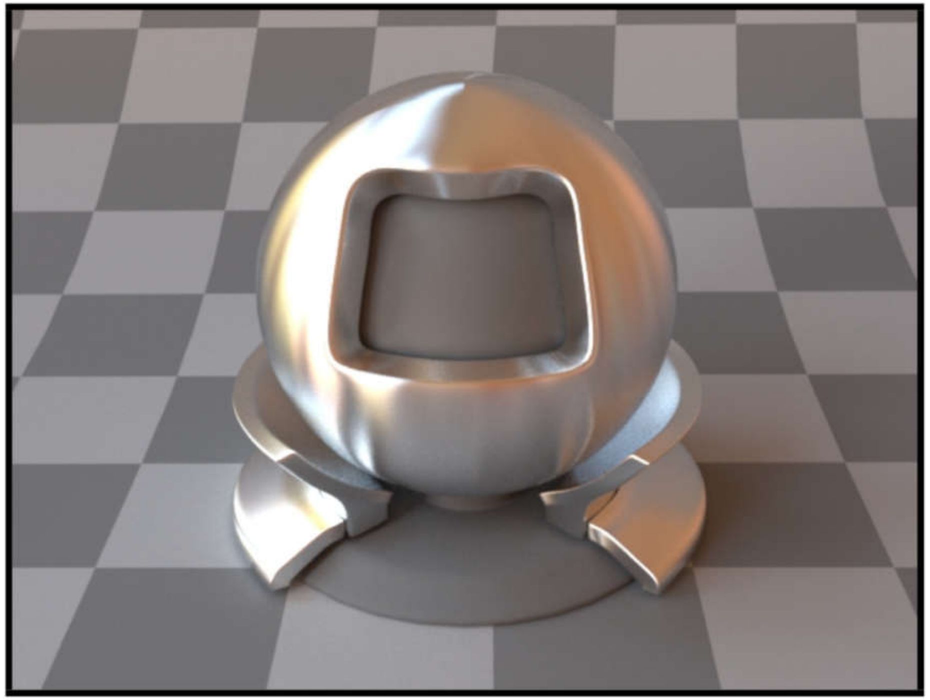


Glossy material (BRDF)



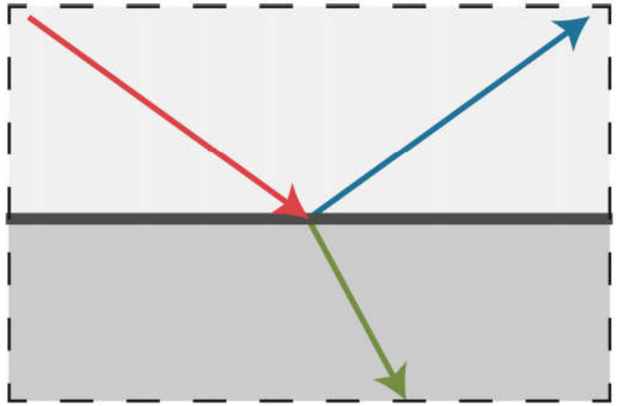


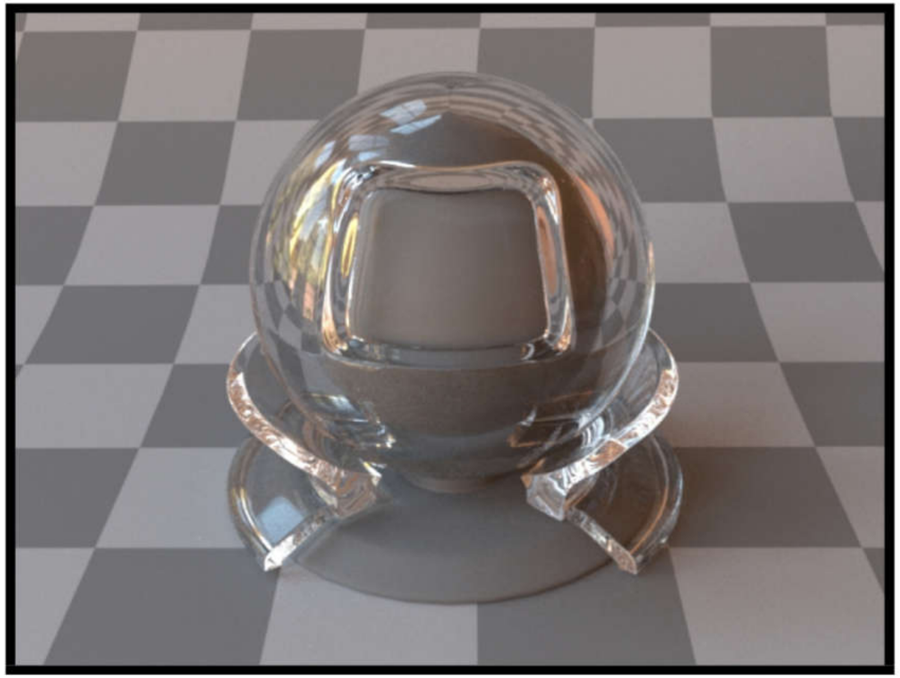
Copper



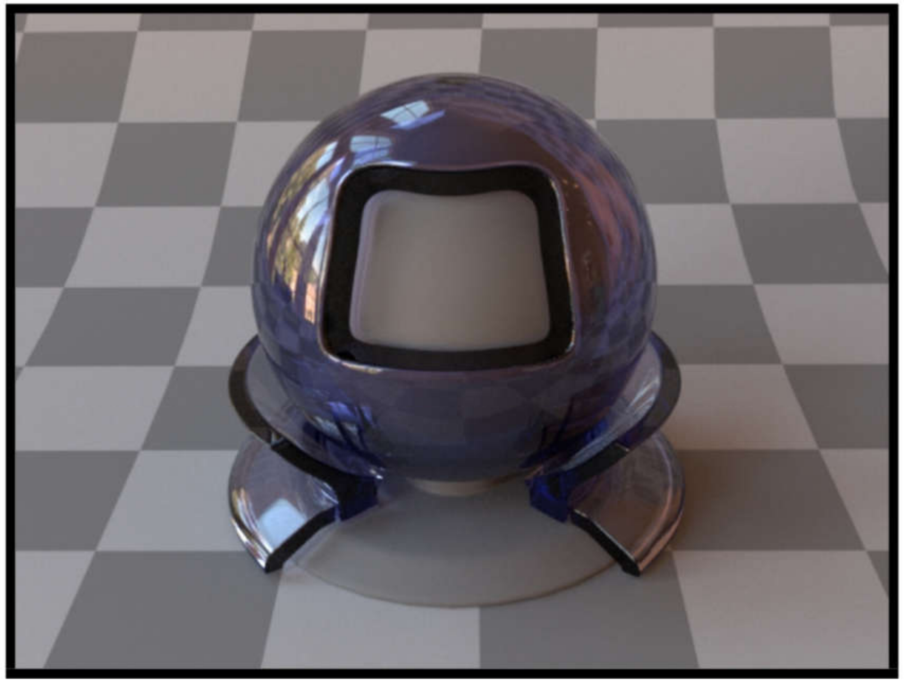
Aluminum

Reflective / refractive material (BRDF)





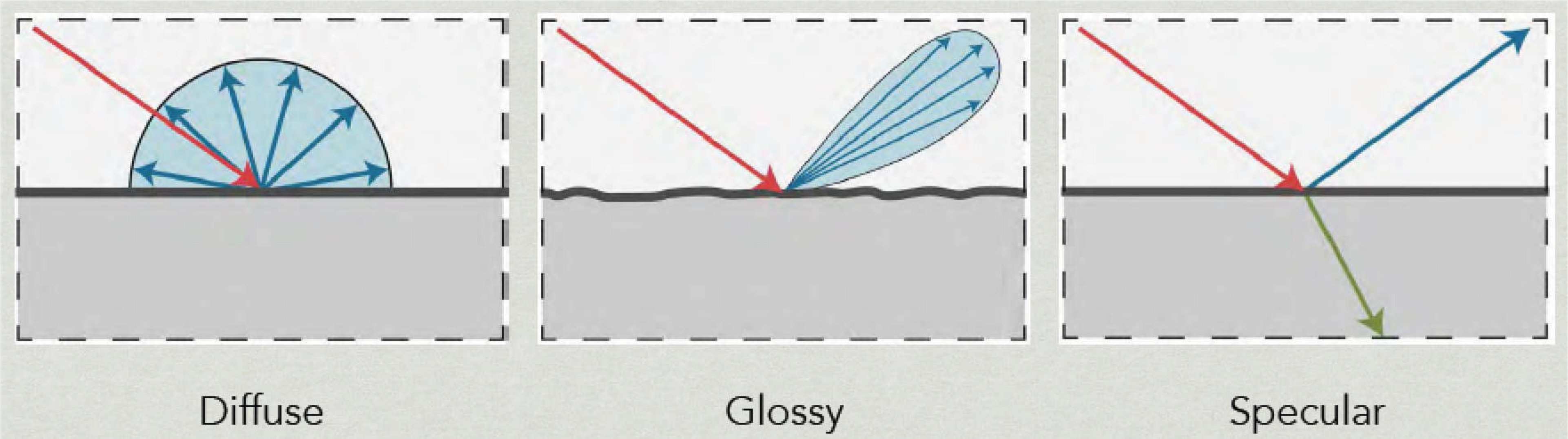
Air <-> water interface



Air <-> glass interface
(with absorption)

Material == BRDF

- Because BRDF defines how the light interacts objects



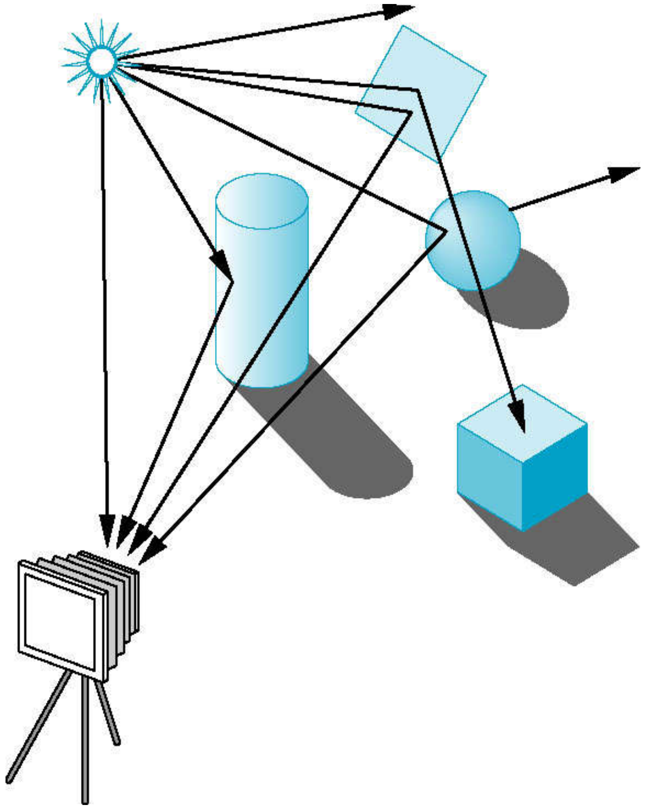
The Rendering Equation

光路的不断反射传播

- 1. Light travels in straight lines

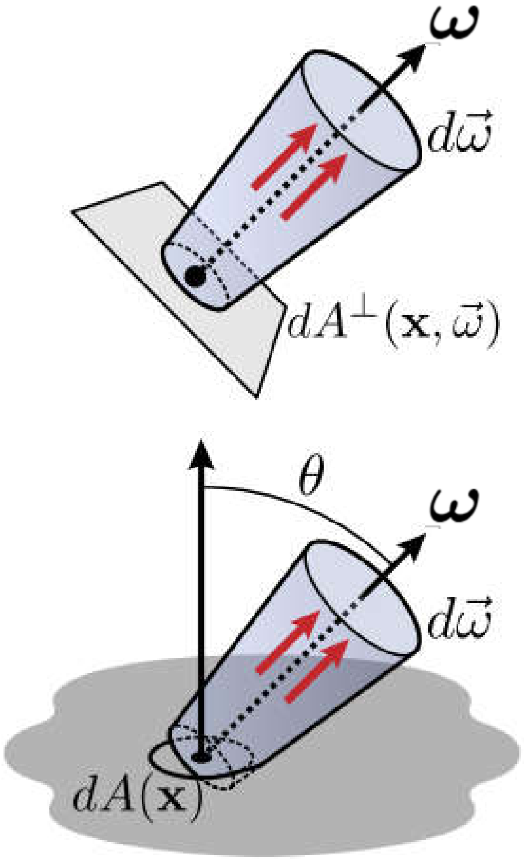
- 2. Light rays do not “collide” with each other if they cross

- 3. Light rays travel from the light sources to the eye



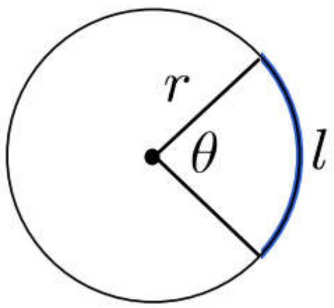
Radiance （辐射率）

- Radiant energy at ***x*** in direction ***ω***:
  - A 5D function $L(\boldsymbol{x}, \boldsymbol{\omega})$: Power
    - per projected surface area
    - per solid angle

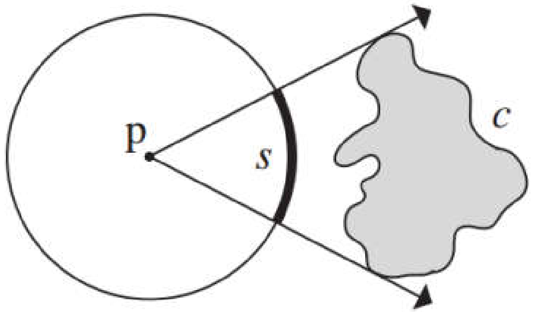


Solid Angle

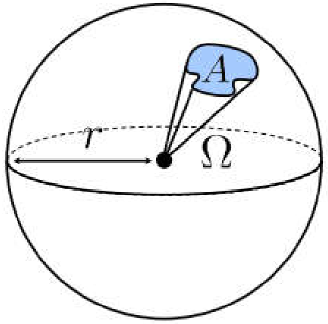
- Angle
  - circle: 2π radians



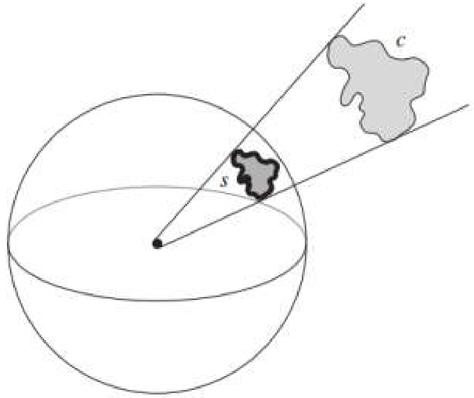
$$\theta = \frac{l}{r}$$



- Solid angle
  - sphere: 4π steradians



$$\Omega = \frac{A}{r^2}$$

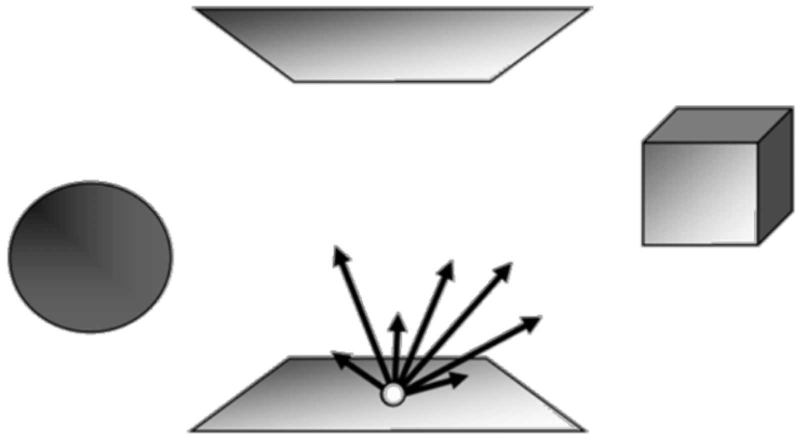


Light Transport

- Goal
  - Describe steady-state radiance distribution in virtual scenes
- Assumptions
  - Geometric optics
  - Achieves steady state instantaneously

Radiance at Equilibrium

- Radiance values at all points in the scene and in all directions expresses the equilibrium
  - 5D "Light-field"
- We only consider radiance on surfaces (4D)
  - Assuming no volumetric scattering or absorption



Rendering Equation (RE)

- RE describes the distribution of radiance at equilibrium

- RE involves:
  - Scene geometry
  - Light source info.
  - Surface reflectance info.
  - Radiance values at all surface points in all directions

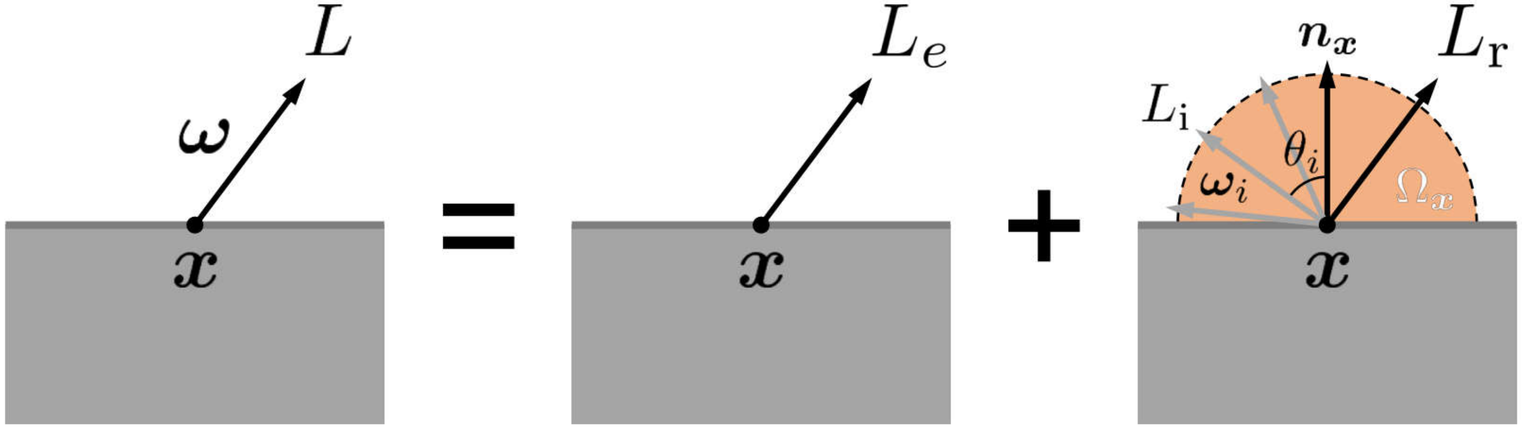
Known (scene geometry, light source info., surface reflectance info.)

Unknown (radiance values at all surface points in all directions)

Rendering Equation (RE)

Kajiya

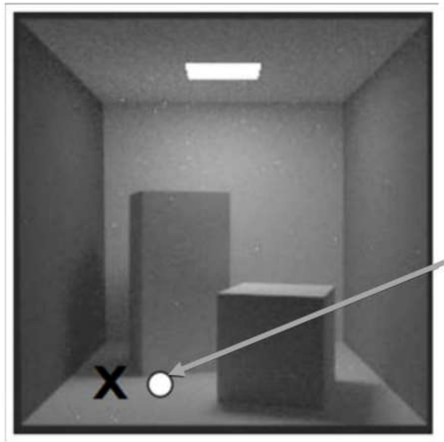
[*outgoing*] = [*emitted*] + [*reflected*]



$$L(\boldsymbol{x}, \boldsymbol{\omega}) = L_e(\boldsymbol{x}, \boldsymbol{\omega}) + L_r(\boldsymbol{x}, \boldsymbol{\omega})$$

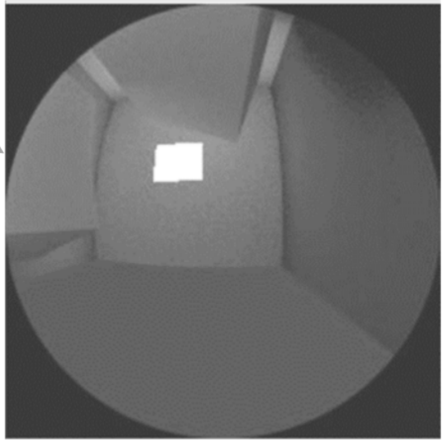
$$\int_{\Omega_{\boldsymbol{x}}} L_i(\boldsymbol{x}, \boldsymbol{\omega}_i)\, f_r(\boldsymbol{x}, \boldsymbol{\omega}_i \leftrightarrow \boldsymbol{\omega}) \underbrace{\langle \boldsymbol{n}_{\boldsymbol{x}}, \boldsymbol{\omega}_i \rangle}_{= \cos\theta_i} \mathrm{d}\boldsymbol{\omega}_i$$

Rendering Equation



$$L(\boldsymbol{x}, \boldsymbol{\omega}) = L_e(\boldsymbol{x}, \boldsymbol{\omega}) + \int_{\Omega_{\boldsymbol{x}}} L_i(\boldsymbol{x}, \boldsymbol{\omega}_i) f_r(\boldsymbol{x}, \boldsymbol{\omega}_i \leftrightarrow \boldsymbol{\omega}) \langle \boldsymbol{n}_{\boldsymbol{x}}, \boldsymbol{\omega}_i \rangle \mathrm{d}\boldsymbol{\omega}_i$$

Incoming radiance



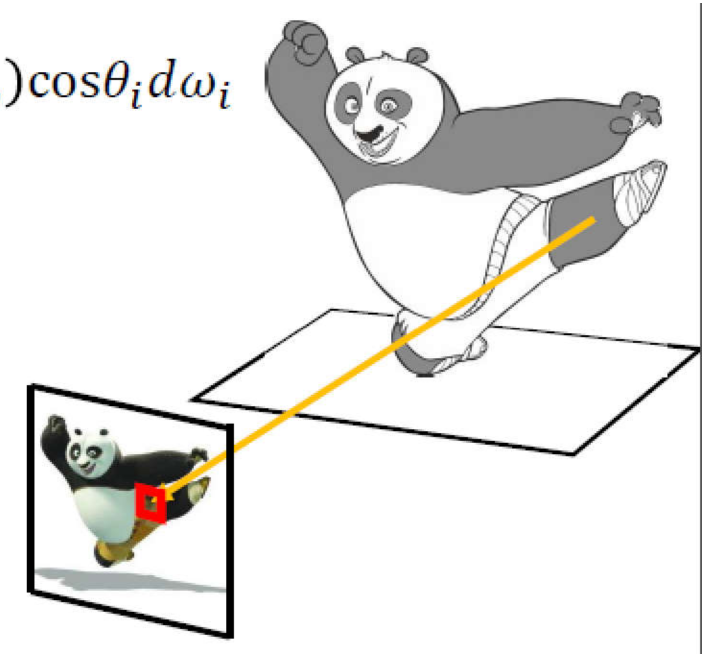
Solving Rendering Equation

Credit by Rui Wang, Shuang Zhao

Solving Rendering Equation

- What is the color of pixel (i,j)?

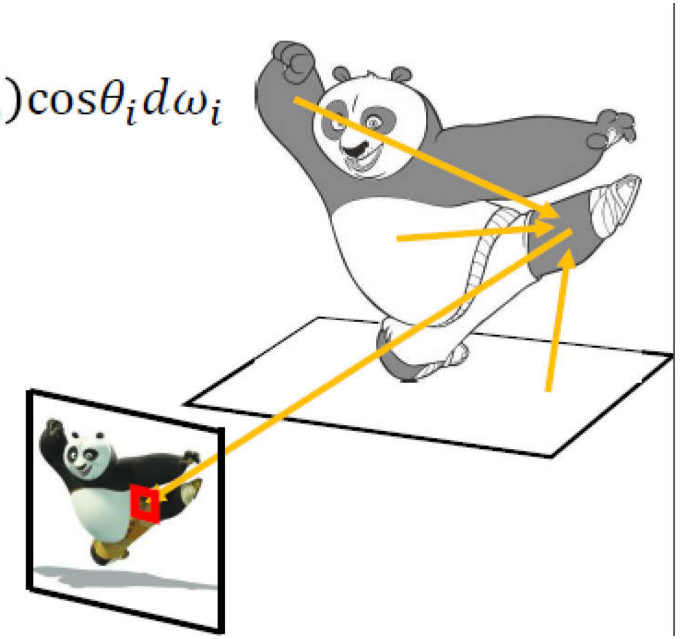
$$L_o(x,\omega_o) = L_e(x,\omega_o) + \int_{H^2} L_i(x,\omega_i) f_r(x,\omega_i \rightarrow \omega_o) \cos\theta_i d\omega_i$$



Solving Rendering Equation

- What is the color of pixel (i,j)?

$$L_o(x,\omega_o) = L_e(x,\omega_o) + \int_{H^2} L_i(x,\omega_i) f_r(x,\omega_i \rightarrow \omega_o)\cos\theta_i d\omega_i$$



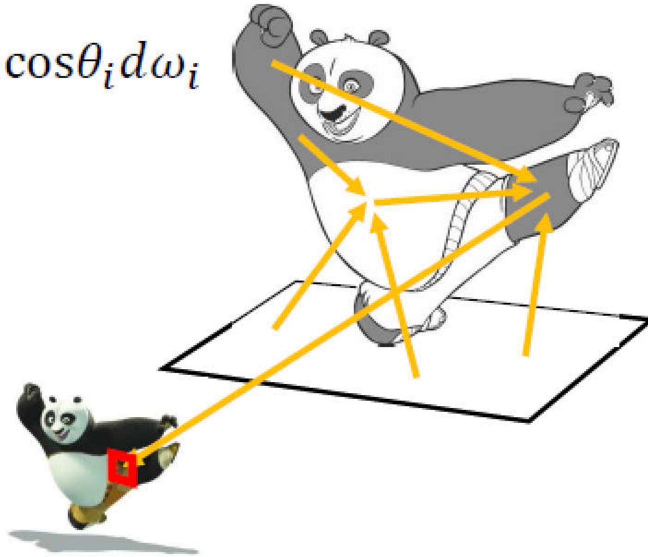
Solving Rendering Equation

- What is the color of pixel (i,j)?

$$L_o(x,\omega_o) = L_e(x,\omega_o) + \int_{H^2} L_i(x,\omega_i) f_r(x,\omega_i \rightarrow \omega_o) \cos\theta_i d\omega_i$$

$$\| $$

$$L_o(x,\omega_o) = \int_{H^2} L_i(x,\omega_i) f_r(x,\omega_i \rightarrow \omega_o) \cos\theta_i d\omega_i$$



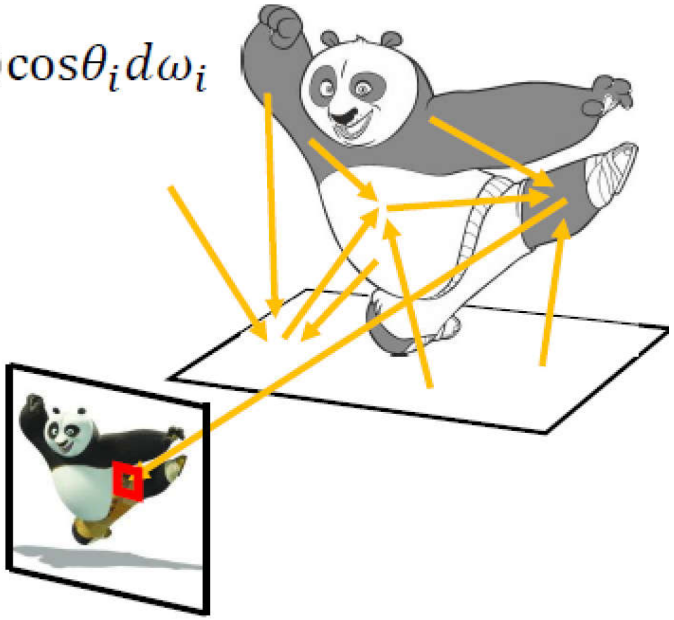
Solving Rendering Equation

- What is the color of pixel (i,j)?

$$L_o(x,\omega_o) = L_e(x,\omega_o) + \int_{H^2} L_i(x,\omega_i) f_r(x,\omega_i \to \omega_o)\cos\theta_i \, d\omega_i$$

$$L_o(x,\omega_o) = \int_{H^2} L_i(x,\omega_i) f_r(x,\omega_i \to \omega_o)\cos\theta_i \, d\omega_i$$

$$L_o(x,\omega_o) = \int_{H^2} L_i(x,\omega_i) f_r(x,\omega_i \to \omega_o)\cos\theta_i \, d\omega_i$$



Solving Rendering Equation

- What is the color of pixel (i,j)?

$$L_o(x,\omega_o) = L_e(x,\omega_o) + \int_{H^2} L_i(x,\omega_i) f_r(x,\omega_i \rightarrow \omega_o)\cos\theta_i d\omega_i$$

$$\parallel$$

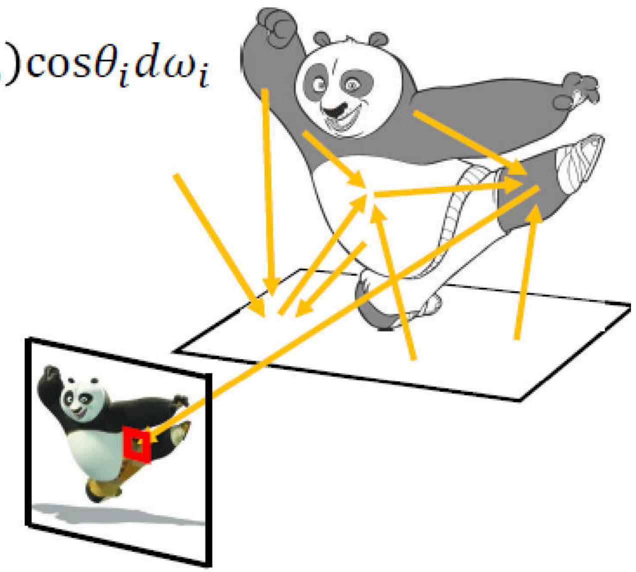
$$L_o(x,\omega_o) = \int_{H^2} L_i(x,\omega_i) f_r(x,\omega_i \rightarrow \omega_o)\cos\theta_i d\omega_i$$

$$\parallel$$

$$L_o(x,\omega_o) = \int_{H^2} L_i(x,\omega_i) f_r(x,\omega_i \rightarrow \omega_o)\cos\theta_i d\omega_i$$

$$\vdots$$

$$L_o(x,\omega_o) = \int_{H^2} L_i(x,\omega_i) f_r(x,\omega_i \rightarrow \omega_o)\cos\theta_i d\omega_i$$



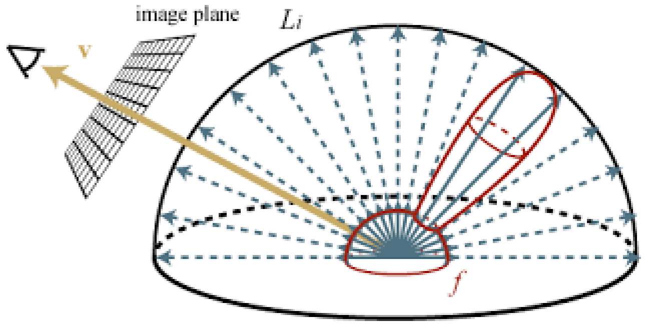
光线跟踪算法
Ray Tracing

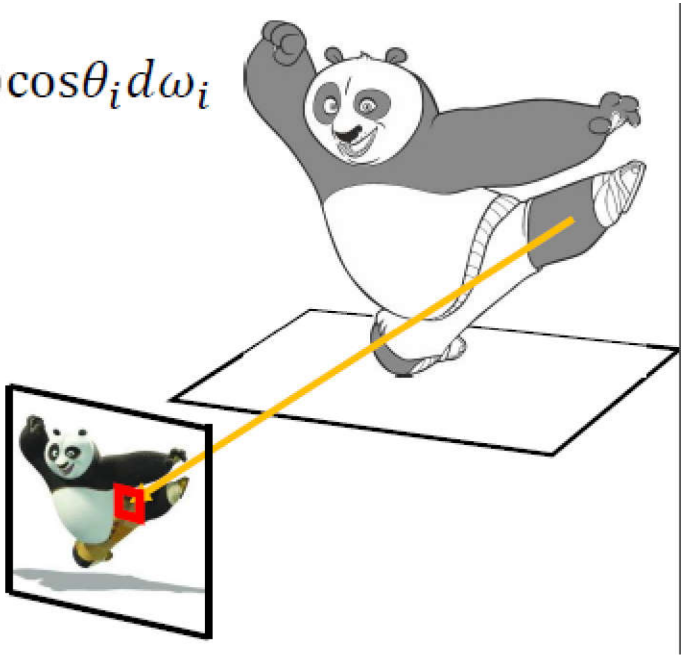
Courtesy of Lingqi Yan, Rui Wang, Shuang Zhao et al.

Solving Rendering Equation

- What is the color of pixel (i,j)?

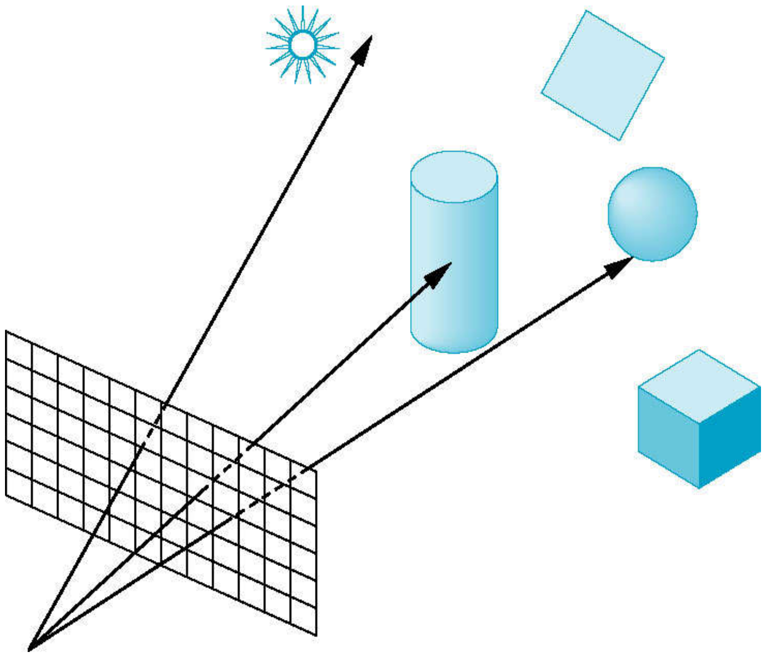
$$L_o(x,\omega_o) = L_e(x,\omega_o) + \int_{H^2} L_i(x,\omega_i) f_r(x,\omega_i \rightarrow \omega_o) \cos\theta_i \, d\omega_i$$





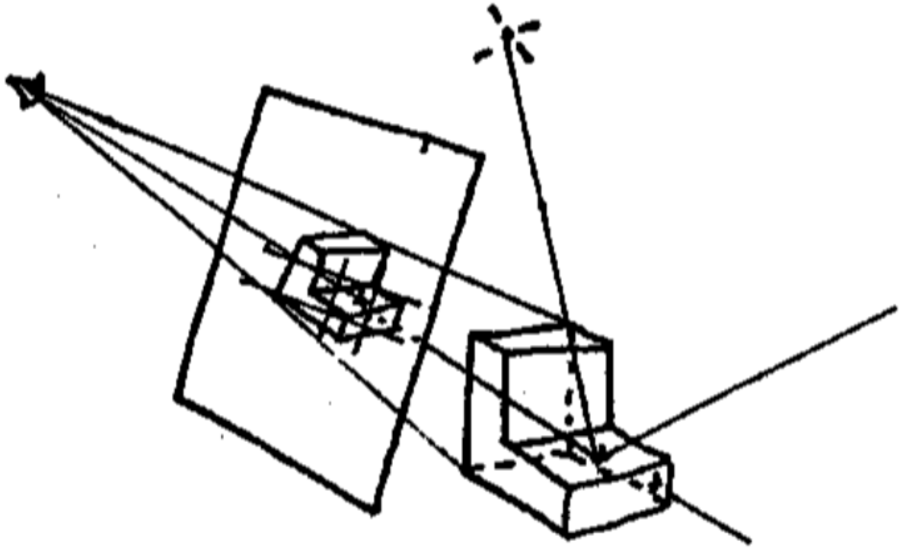
反向跟踪

- 只有到达视点的光线才有作用
- 逆光线方向投射光线
- 每个像素至少需要一条光线

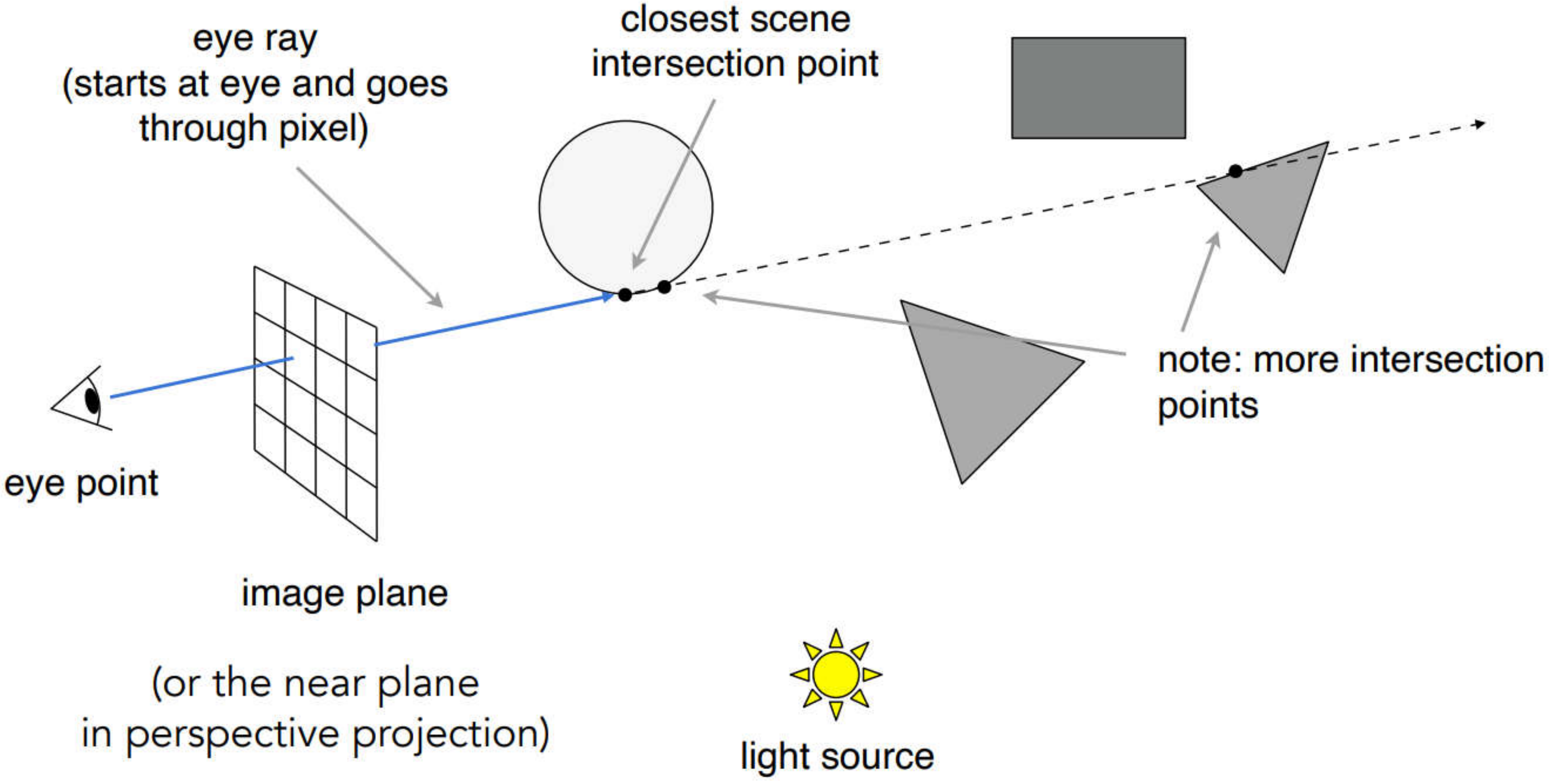


Ray Casting

- Appel 1968 - Ray casting
  - 1. Generate an image by **casting one ray per pixel**
  - 2. Check for shadows by **sending a ray to the light**

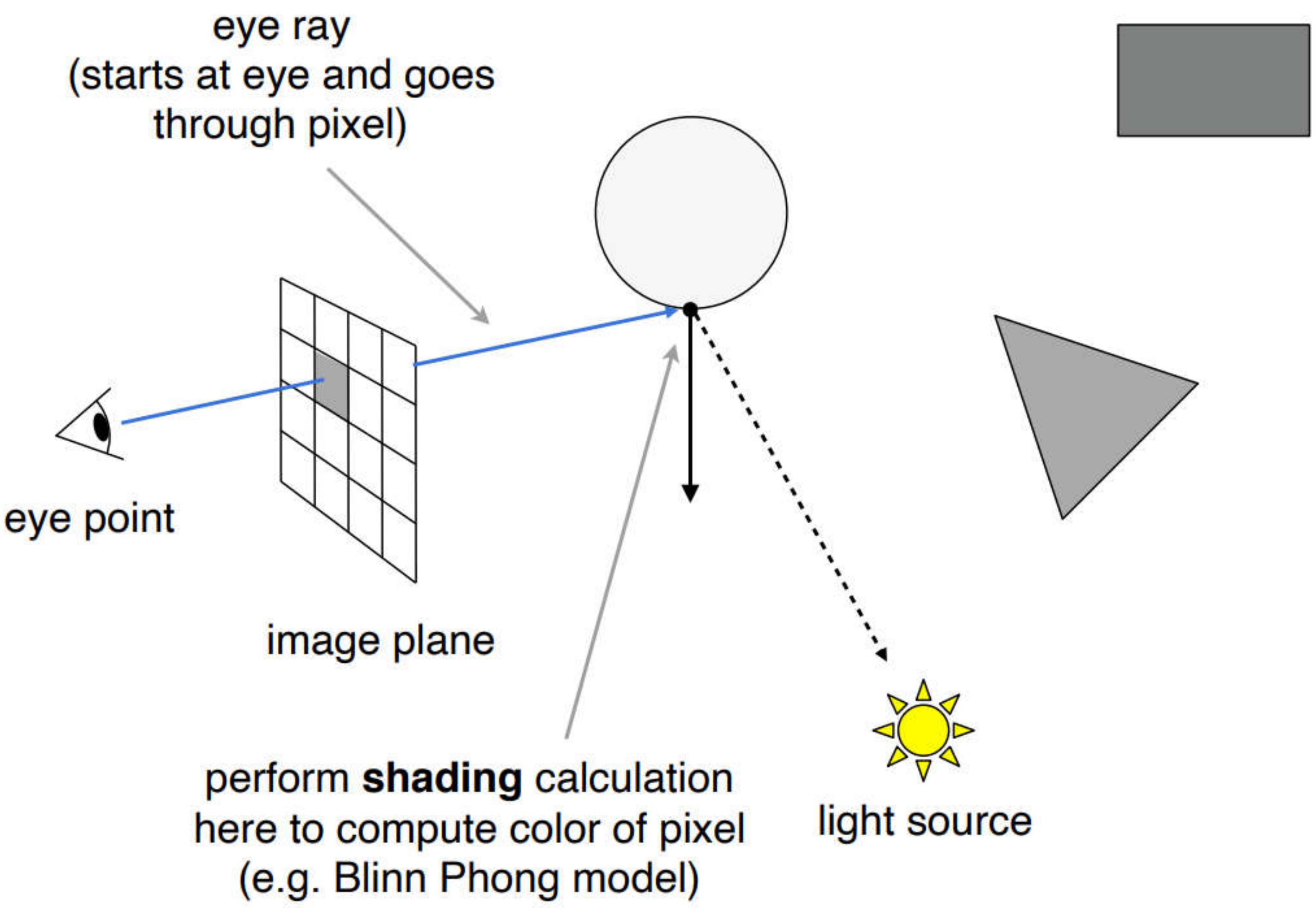


Ray Casting - Generating Eye Rays



(or the near plane
in perspective projection)

Ray Casting - Shading Pixels (Local Only)

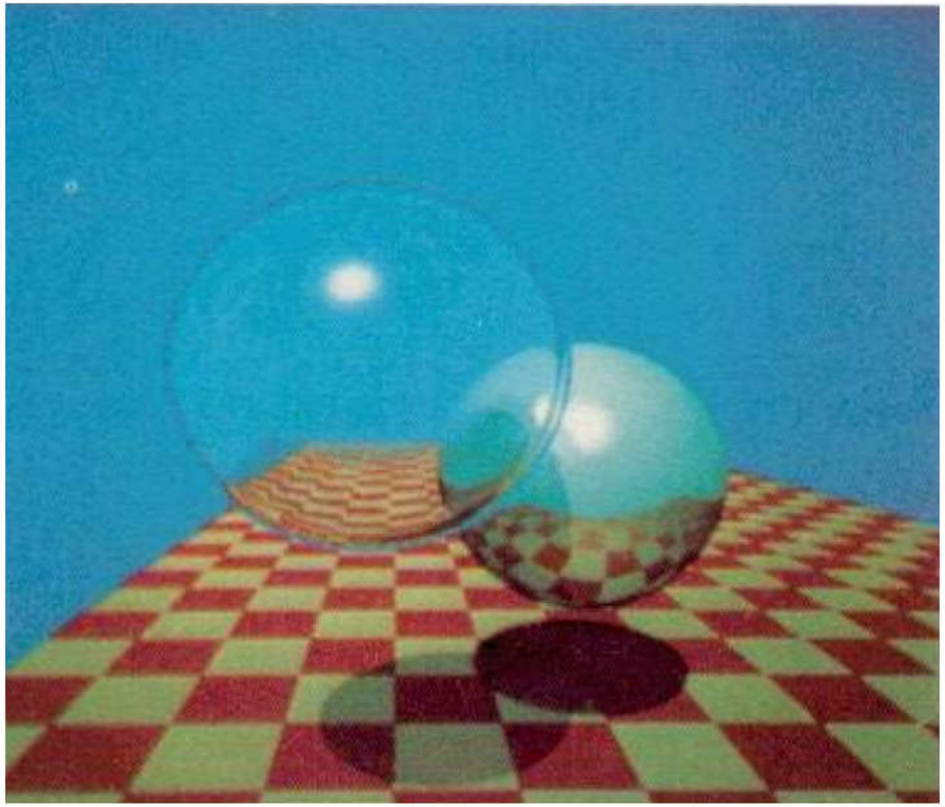


Recursive (Whitted-Style) Ray Tracing

"An improved Illumination model for shaded display"
T. Whitted, CACM 1980

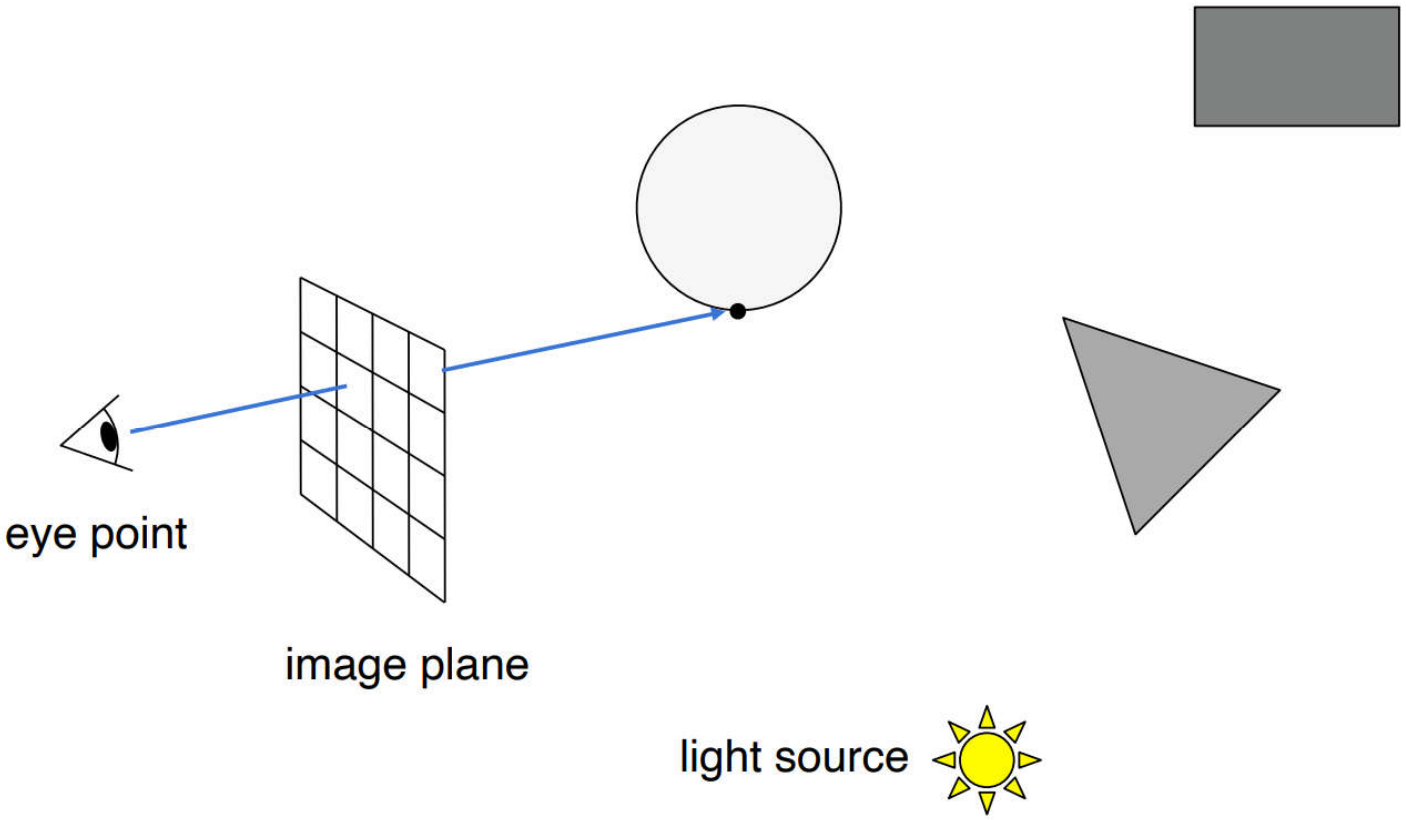
Time:

- VAX 11/780 (1979) 74m
- PC (2006) 6s
- GPU (2012) 1/30s

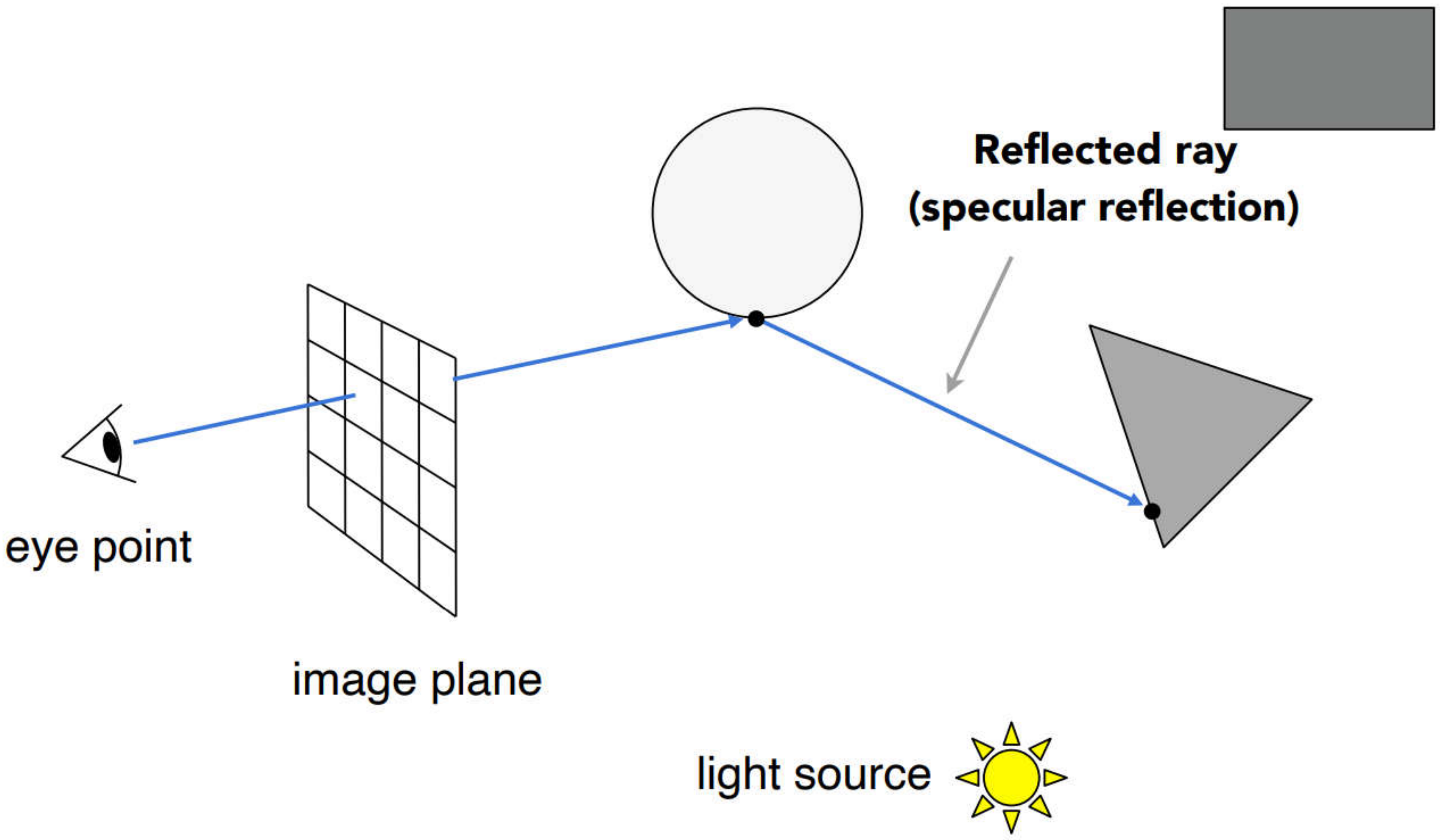


Spheres and Checkerboard, T. Whitted, 1979

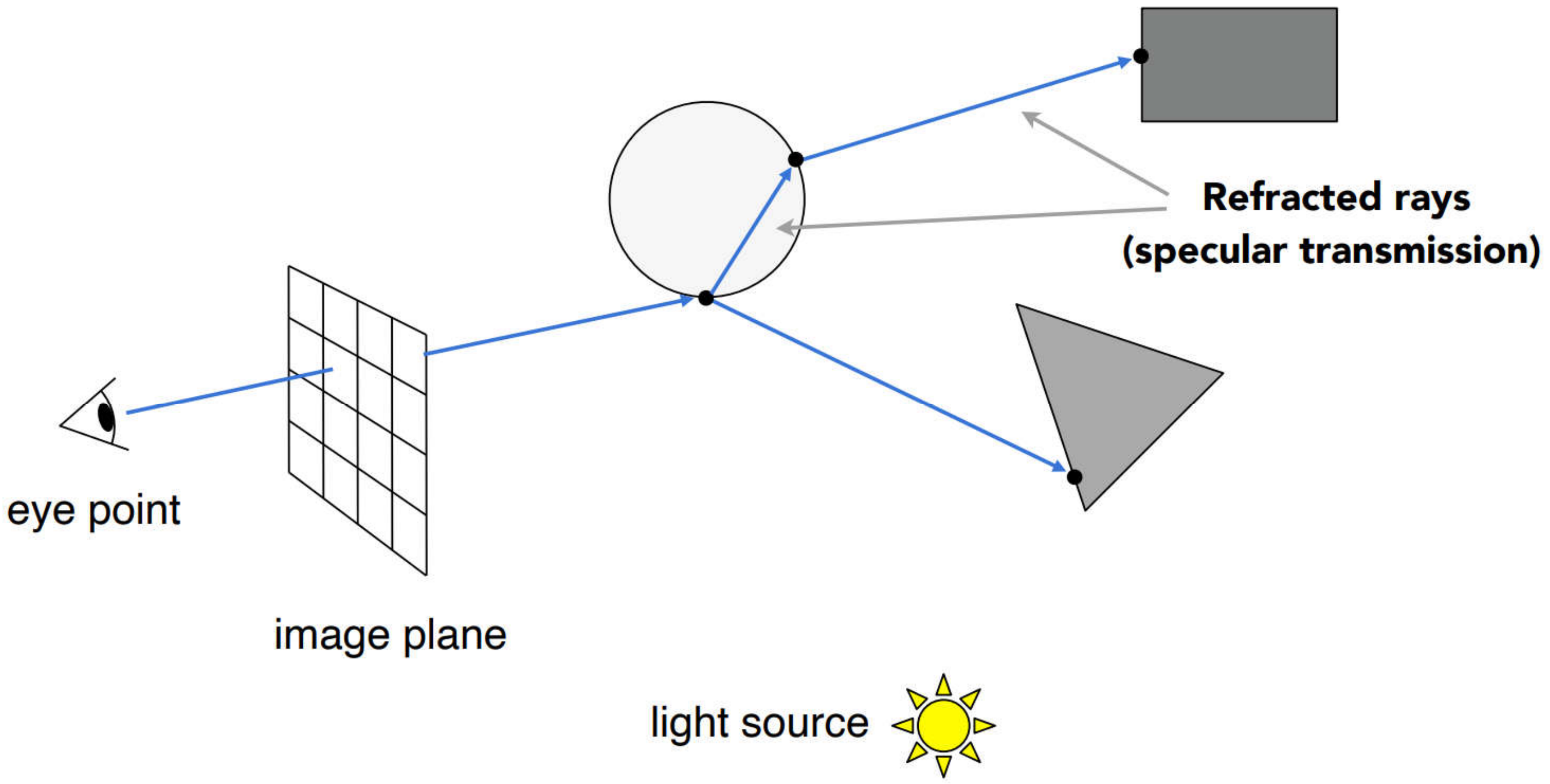
Recursive Ray Tracing



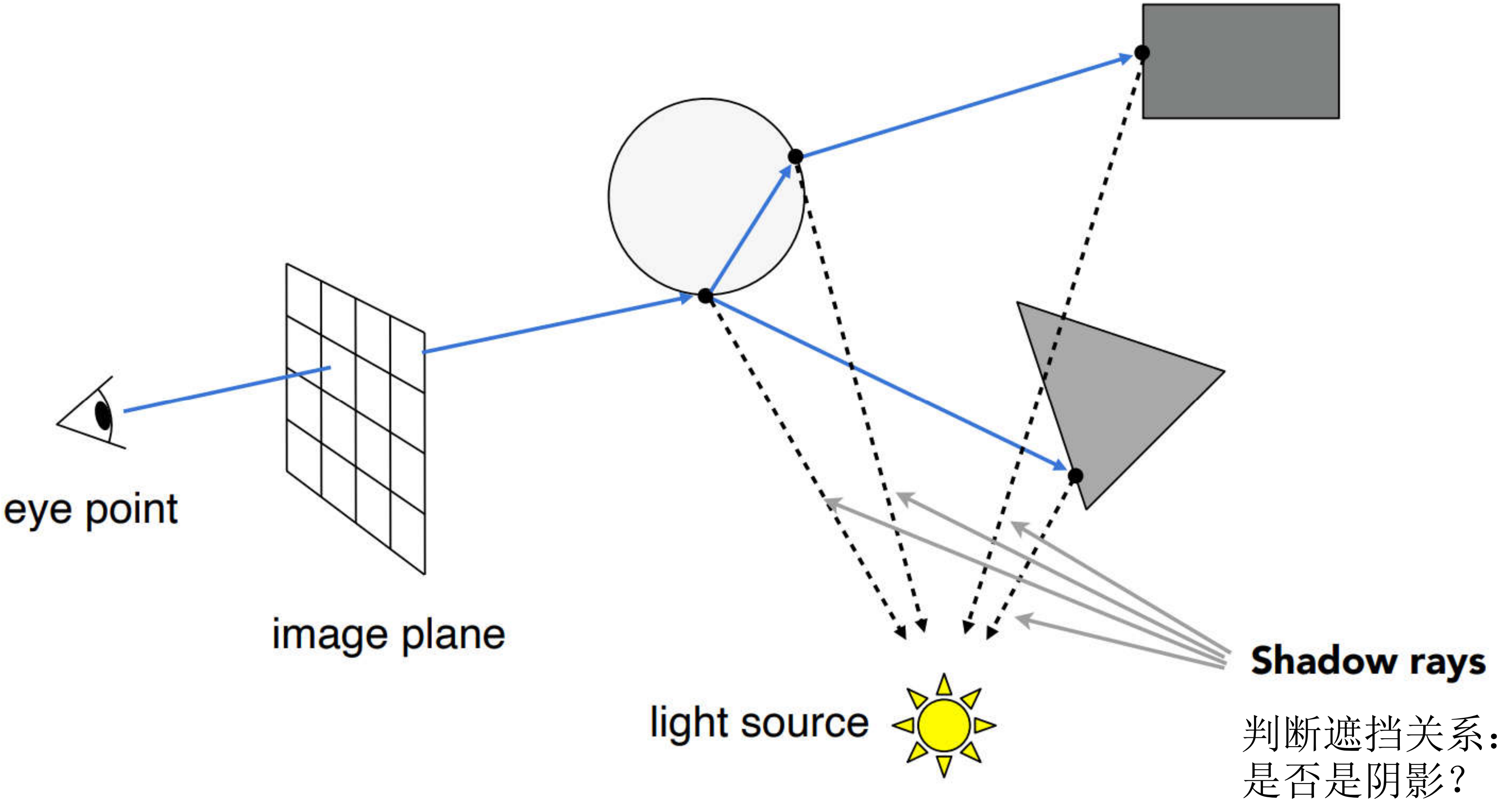
Recursive Ray Tracing



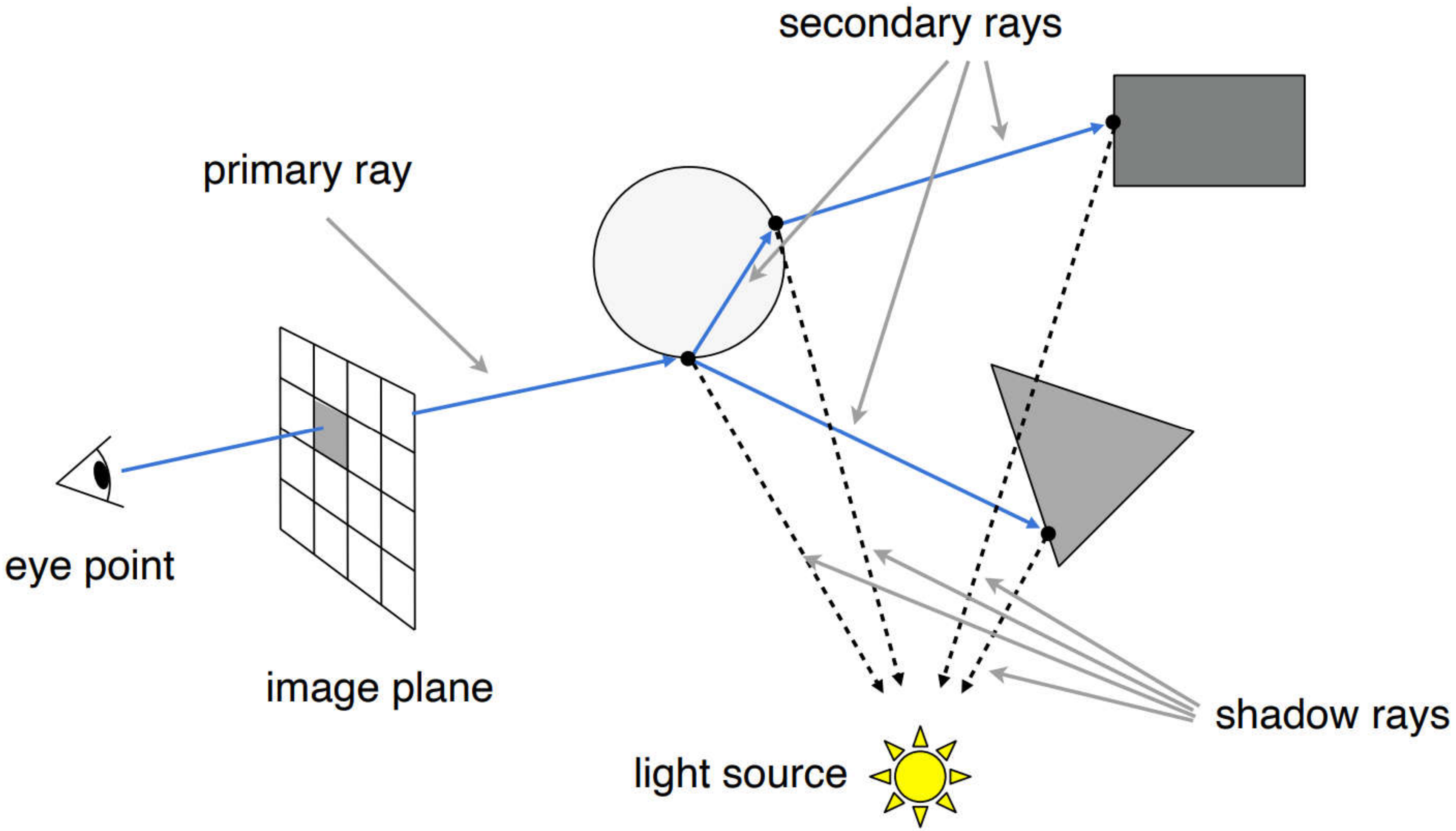
Recursive Ray Tracing



Recursive Ray Tracing



Recursive Ray Tracing



递归的终止条件

- 在每个相交时，有些光线被吸收
  - 跟踪余下的量
- 忽略进入无穷空间的光线（或背景光强）
  - 在场景周围放一个大球
- 递归步数有限制

算法框架

```
color trace(point p, vector d, int step)
{
   color local, reflected, transmitted;
   point q;
   normal n;
   if(step > max) return(background_color);

   q = intersect(p,d,status);
   if(status == light_source)
            return(light_source_color);
   if(status == no_intersection)
            return(background_color);

   n = normal(q);
   r = reflect(q,n);
   t = transmit(q,n);
   local = phong(q,n,r);
   reflected = trace(q,r,step+1);  // if this is a reflected
   transmitted = trace(q,t,step+1); // if this is a transmitted

   return(local+reflected+transmitted);
}
```

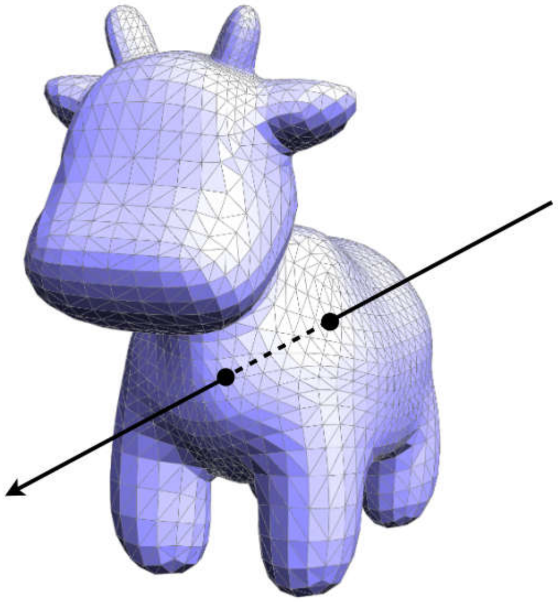
Recursive Ray Tracing

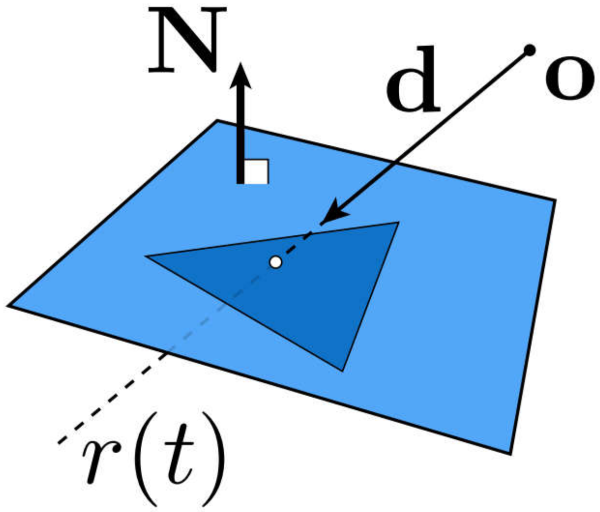
计算效率

大量的求交计算

- 点与三角网格的求交
- 点与三角形的求交

向量运算!

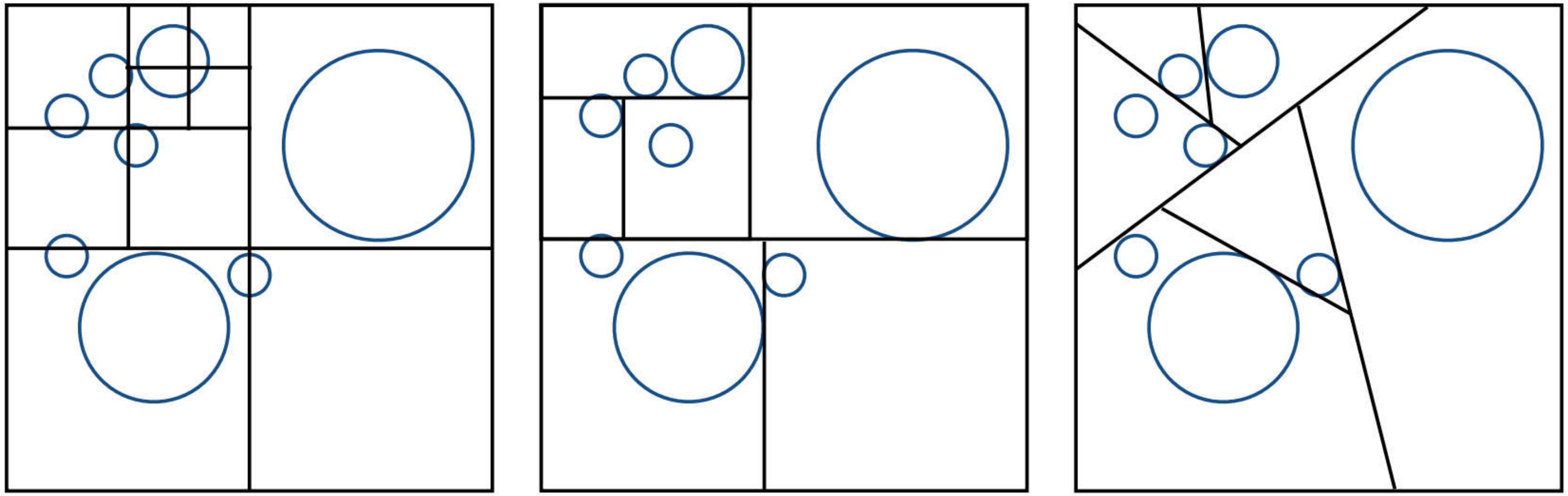




加速策略： Bounding Volumes

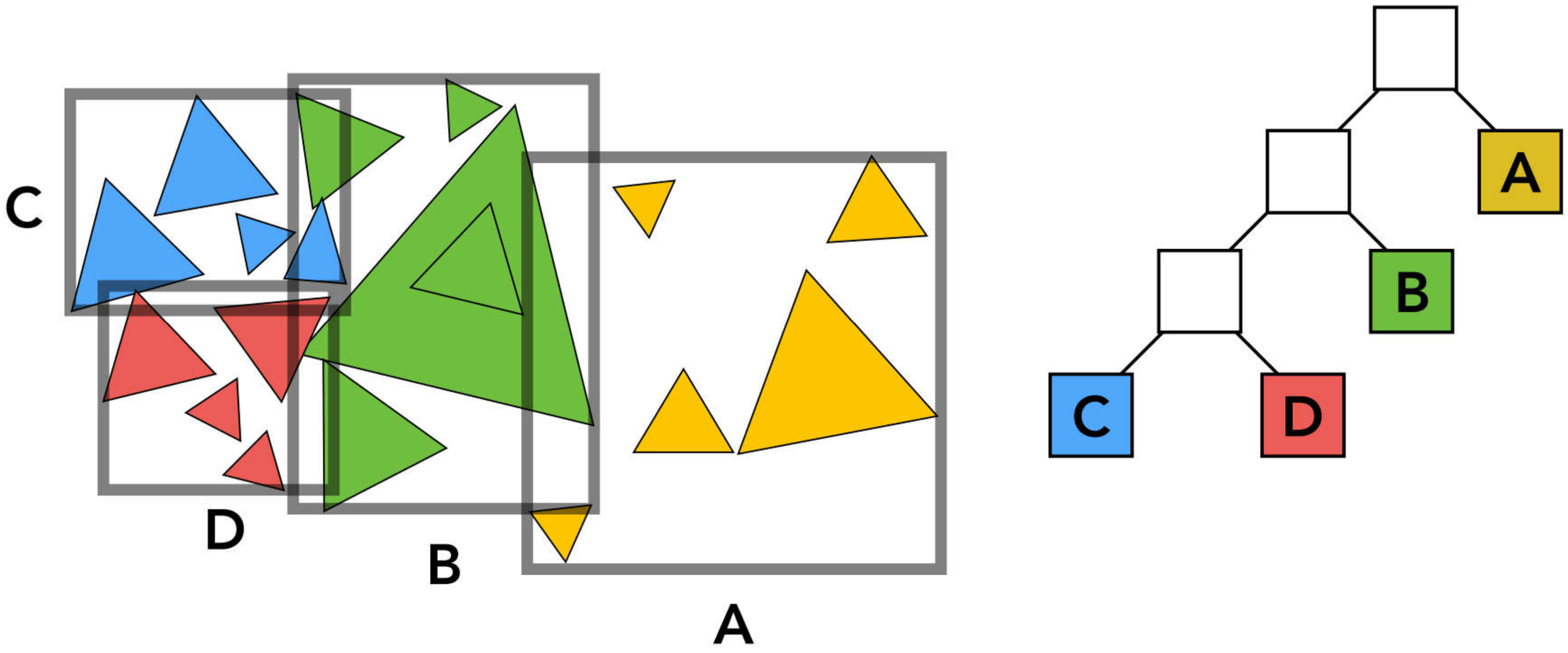
- Axis-Aligned Bounding Box (AABB) (轴对齐包围盒)
- 包围球

加速策略： 空间组织



Oct-Tree　　**KD-Tree**　　BSP-Tree

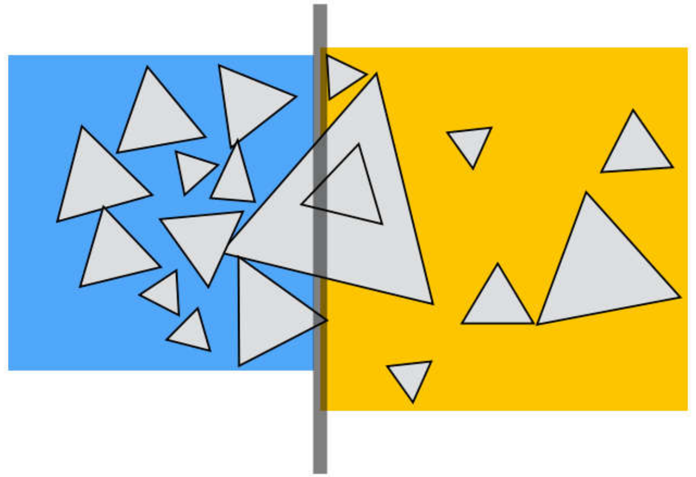
加速策略： Bounding Volume Hierarchy (BVH)



Spatial vs Object Partitions

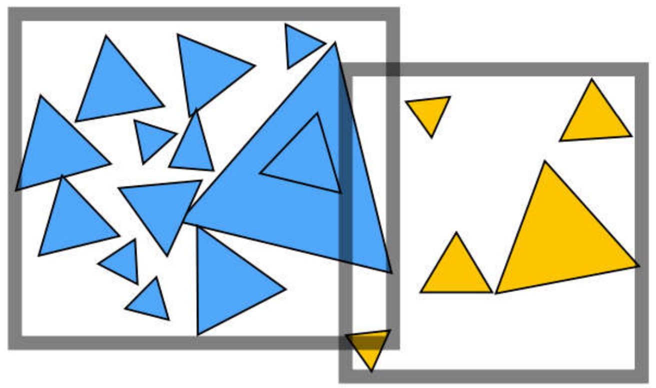
Spatial partition (e.g.KD-tree)

- Partition space into non-overlapping regions
- An object can be contained in multiple regions



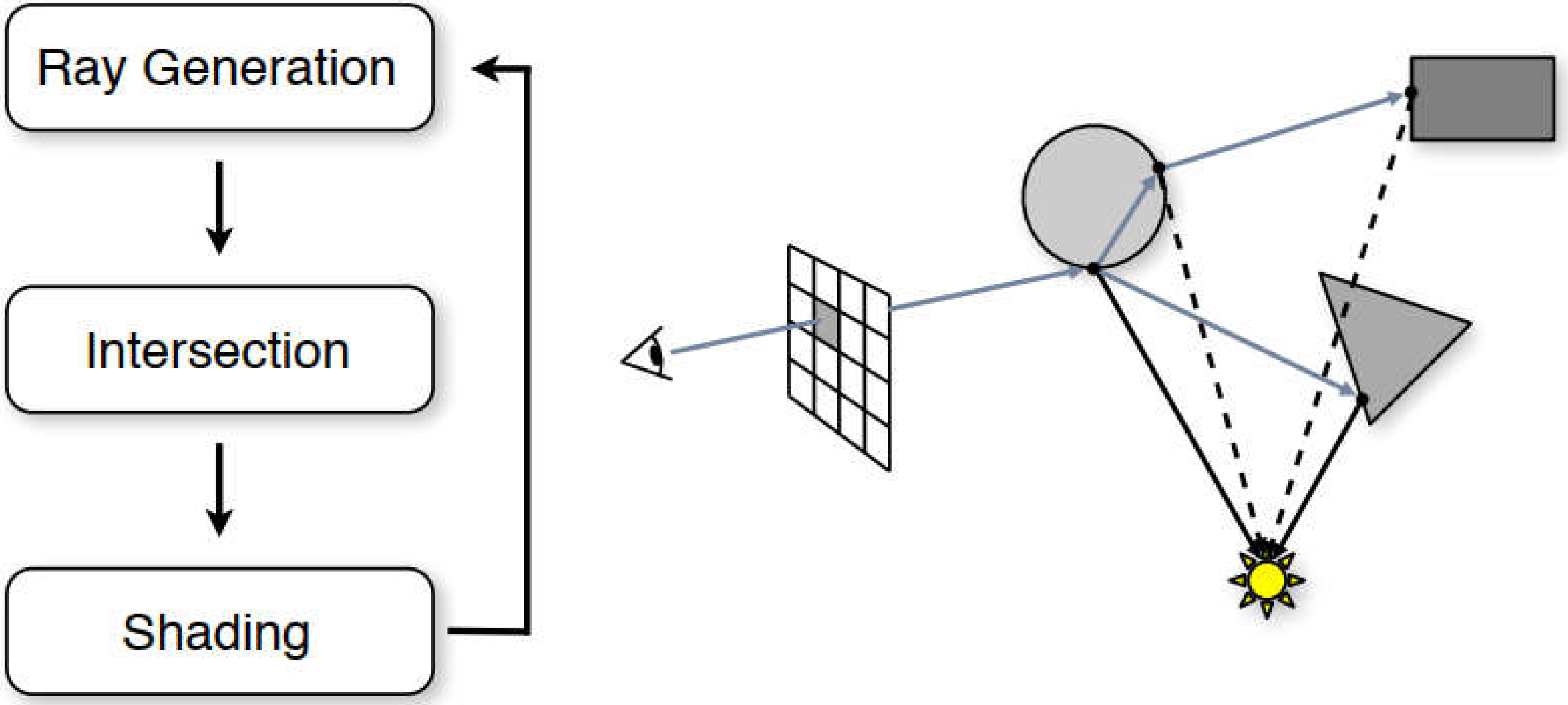
Object partition (e.g. BVH)

- Partition set of objects into disjoint subsets
- Bounding boxes for each set may overlap in space

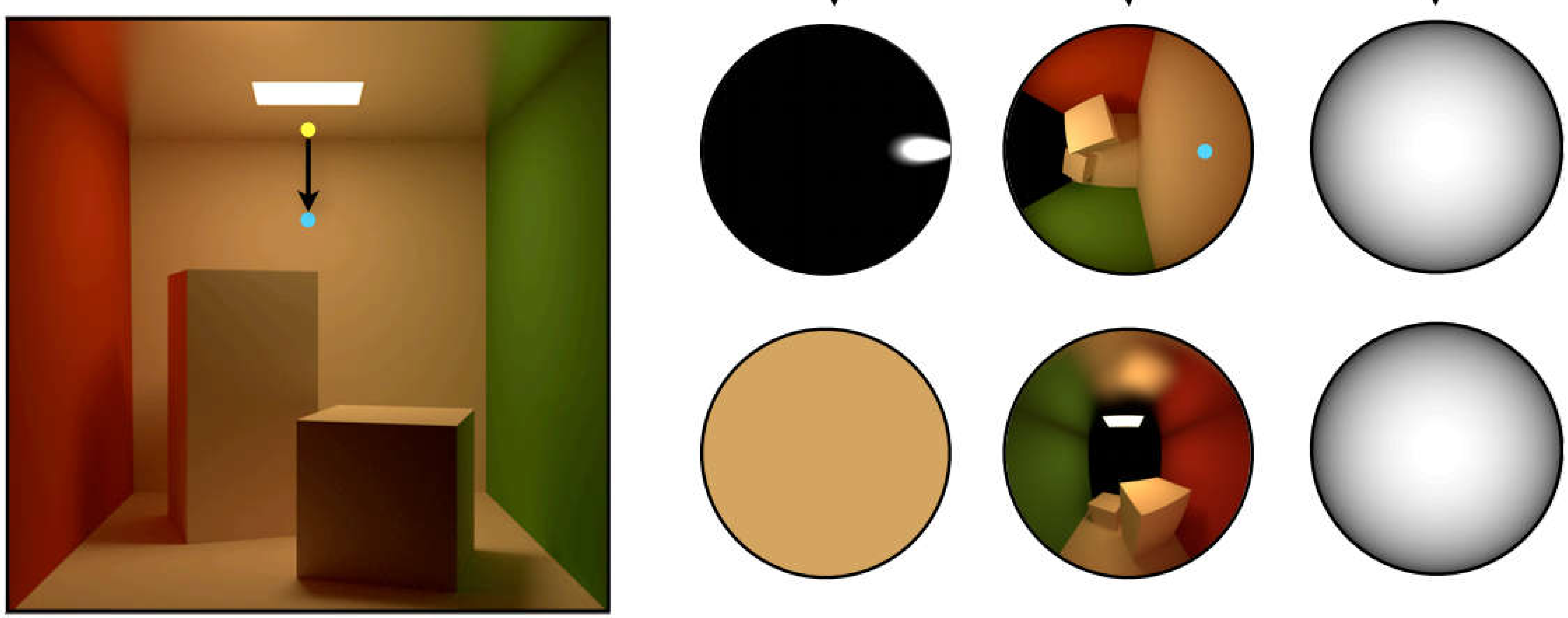


Discussions

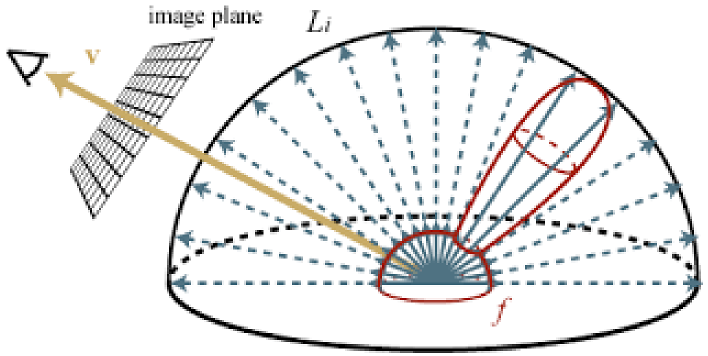
Recap: Ray Tracing

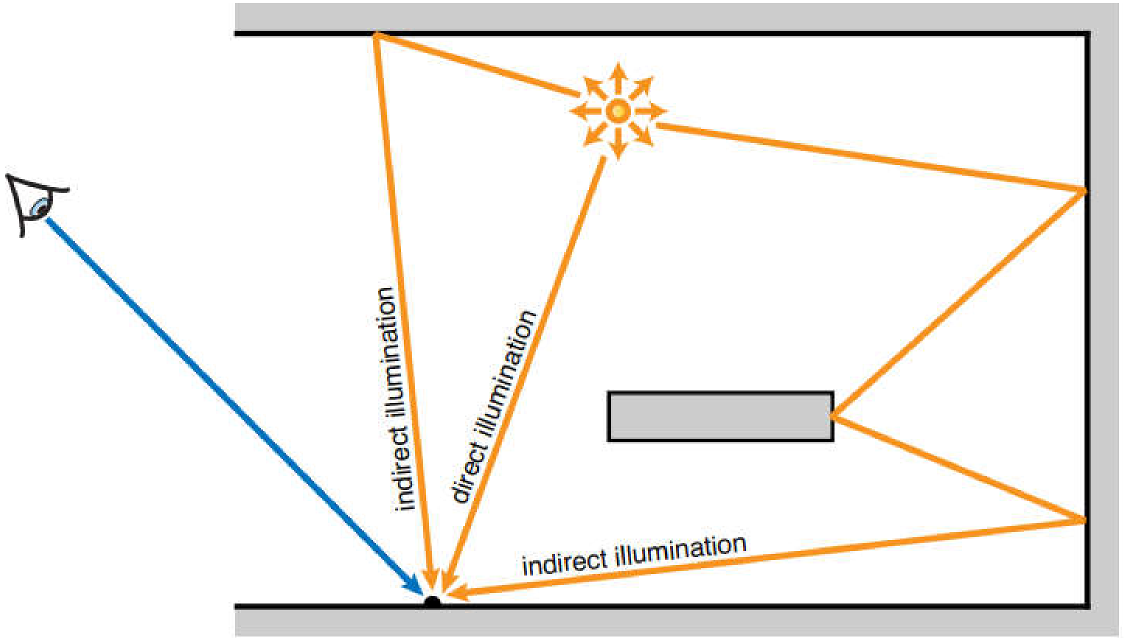


$$L(\mathbf{x}, \vec{\omega}) = L_e(\mathbf{x}, \vec{\omega}) + \int_{H^2} f_r(\mathbf{x}, \vec{\omega}', \vec{\omega}) L(r(\mathbf{x}, \vec{\omega}'), -\vec{\omega}') \cos\theta' d\vec{\omega}'$$



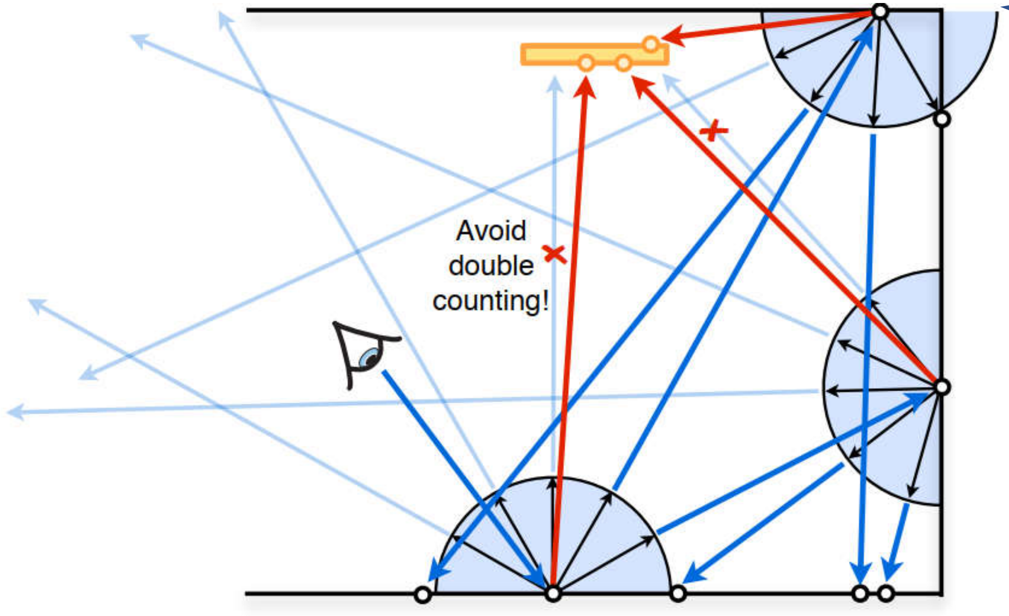
光线跟踪算法是在求解渲染方程吗？



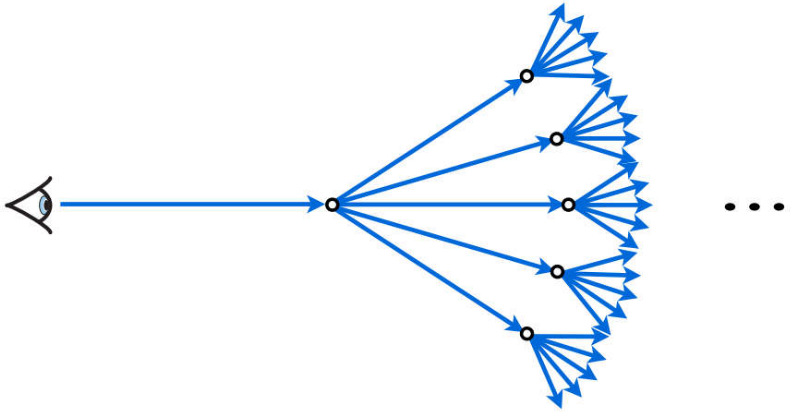


如何真正求解渲染方程？

- 积分方程的求解，如何计算积分？



光线数爆炸！



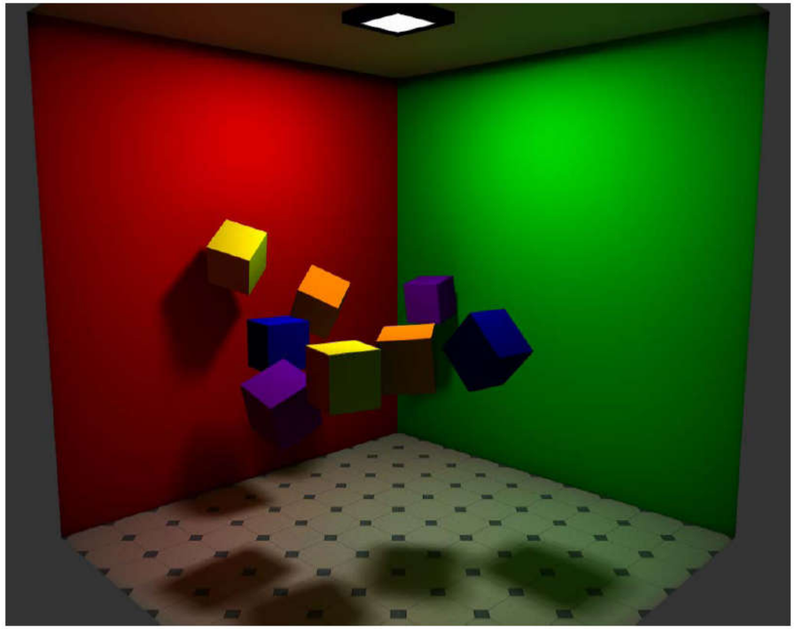
Radiosity
辐射度渲染方法

Courtesy of Lingqi Yan, Rui Wang et al.

辐射度方法

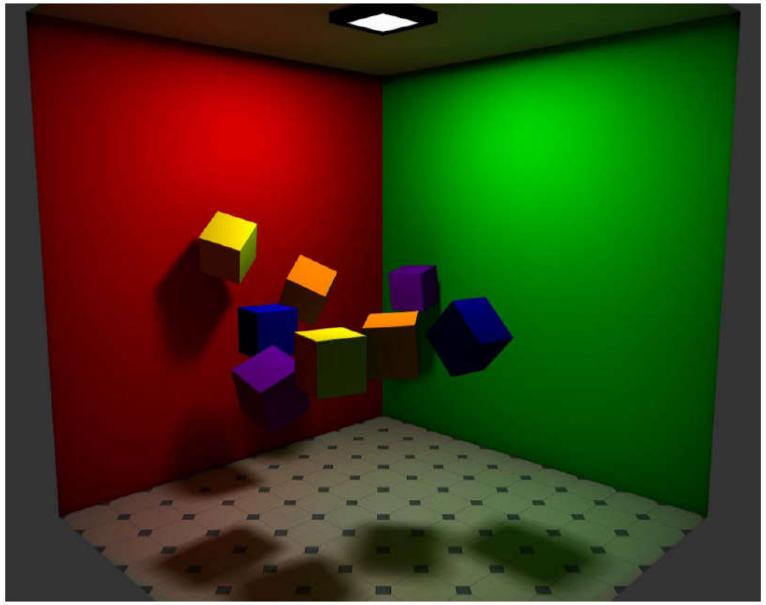
[Goral et al. 1984]

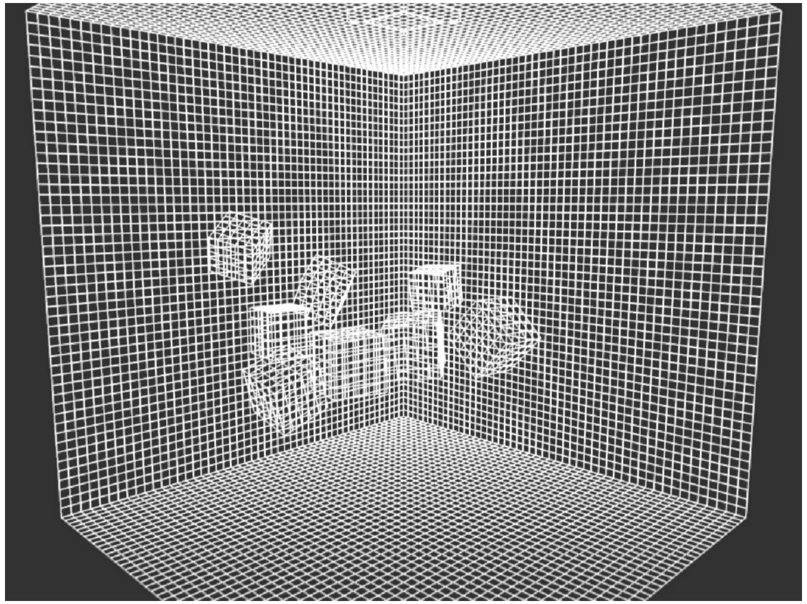
- 假定：场景中的物体均为理想漫反射表面(Diffuse)
  - Radiosity方法不能处理镜面反射的物体表面
- 求解渲染方程：与视点无关，只需求解一次
  - 稀疏方程组
  - 如光源不变，可预存为贴图



Idea: Radiance at Equilibrium

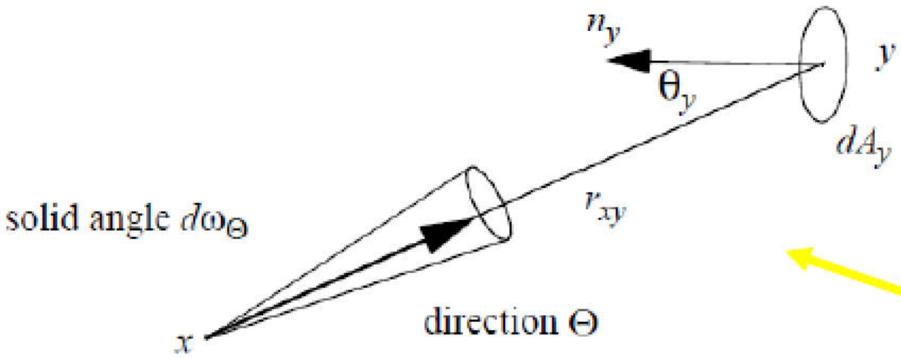
- Computing steady-state radiance distribution directly
- The scene is composed of small facets

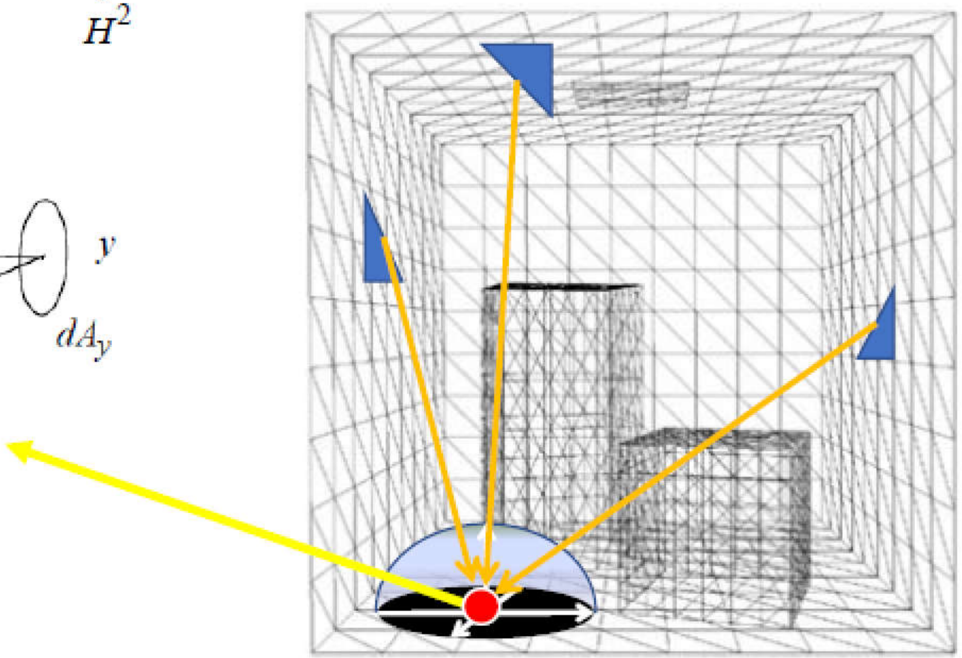




Rendering Equation

$$L_o(x,\omega_o) = L_e(x,\omega_o) + L_r(x,\omega_o)$$
$$= L_e(x,\omega_o) + \int_{H^2} f_r(x,\omega_i \to \omega_o) L_i(x,\omega_i)\cos\theta_i \, d\omega_i$$

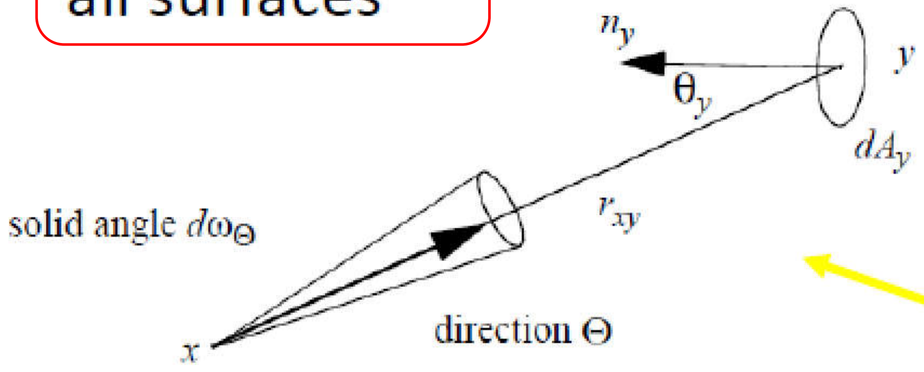


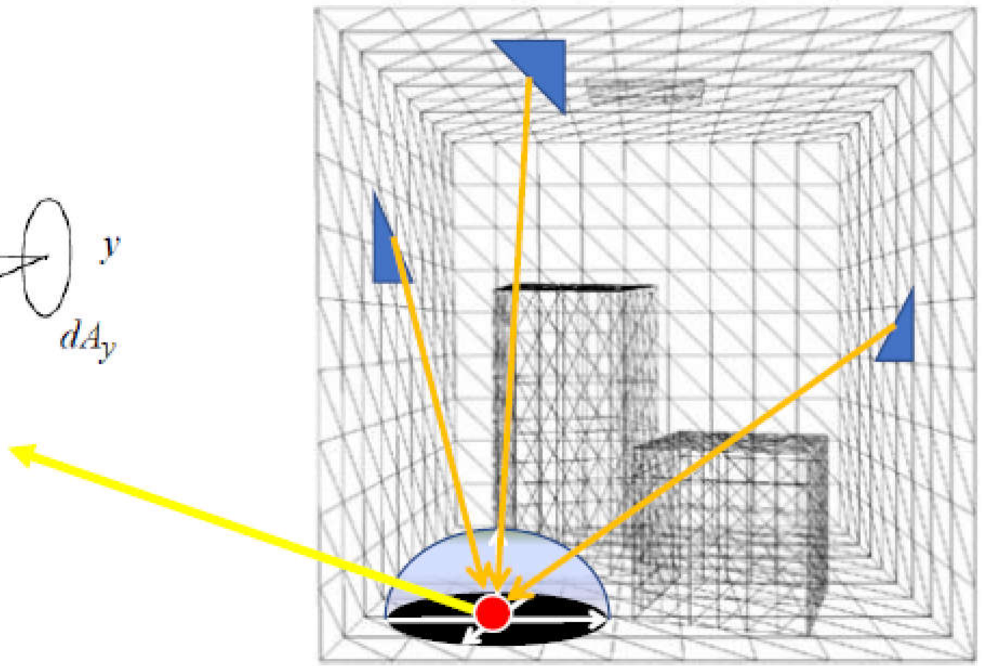


Another Form of Rendering Equation

$$L(x',x) = L_e(x',x) + \int_{M^2} f_r(x'',x',x)\, L(x'',x')\, G(x'',x')\, dA''(x'')$$

Integrate over all surfaces





Another Form of Rendering Equation

$$L(x',x) = L_e(x',x) + \int_{M^2} f_r(x'',x',x)\, L(x'',x')\, G(x'',x')\, dA''(x'')$$

Integrate over all surfaces

Geometry term

$$G(x'',x') = \frac{\cos\theta_i'' \cos\theta_o'}{\|x''-x'\|^2} V(x'',x')$$

Visibility term

$$V(x'',x') = \begin{cases} 1 & \text{visible} \\ 0 & \text{not visible} \end{cases}$$

Solving the Rendering Equation

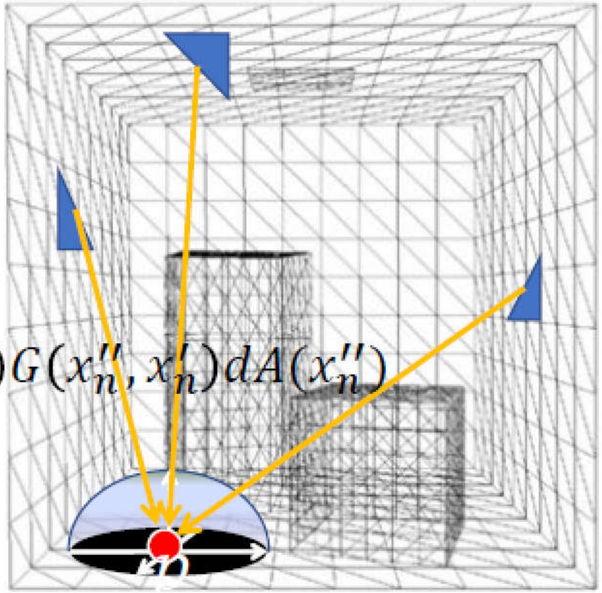
$$L_o(x'_0, x_0) = L_e(x'_0, x_0) + \int_{M^2} L_i(x'_0, x''_0) f_r(x_0, x'_0, x''_0) G(x''_0, x'_0) dA(x''_0)$$

$$L_o(x'_1, x_1) = L_e(x'_1, x_1) + \int_{M^2} L_i(x'_1, x''_1) f_r(x_1, x'_1, x''_1) G(x''_1, x'_1) dA(x''_1)$$

$$\vdots$$

$$\vdots$$

$$L_o(x'_n, x_n) = L_e(x'_n, x_n) + \int_{M^2} L_i(x'_n, x''_n) f_r(x_n, x'_n, x''_n) G(x''_n, x'_n) dA(x''_n)$$



Solving the Rendering Equation

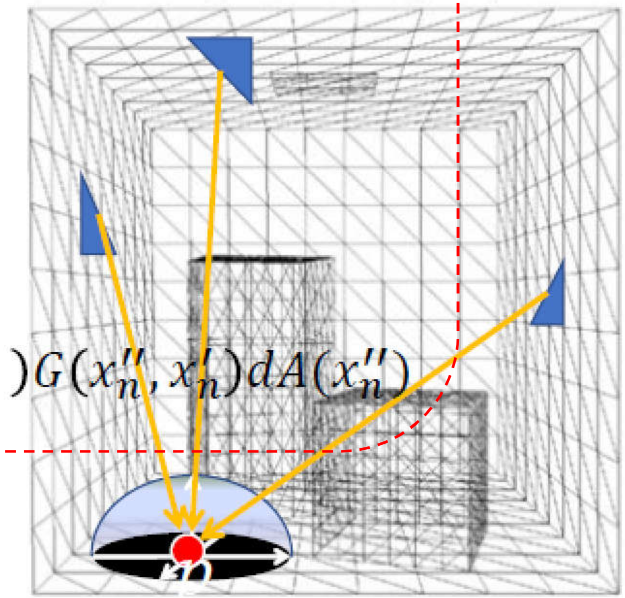
$$L_o(x_0', x_0) = L_e(x_0', x_0) + \int_{M^2} L_i(x_0', x_0'') f_r(x_0, x_0', x_0'') G(x_0'', x_0') dA(x_0'')$$

$$L_o(x_1', x_1) = L_e(x_1', x_1) + \int_{M^2} L_i(x_1', x_1'') f_r(x_1, x_1', x_1'') G(x_1'', x_1') dA(x_1'')$$

$$\vdots$$

$$L_o(x_n', x_n) = L_e(x_n', x_n) + \int_{M^2} L_i(x_n', x_n'') f_r(x_n, x_n', x_n'') G(x_n'', x_n') dA(x_n'')$$

L $\quad L_e$ $\quad K \circ L$



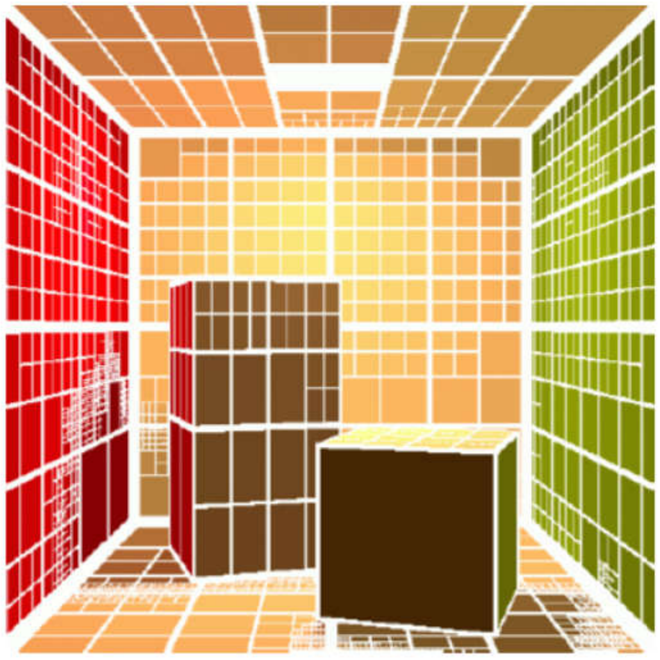
$$L = L_e + K \circ L$$

Solving the Rendering Equation

$$L = L_e + K \circ L \Rightarrow (I - K) \circ L = L_e$$

Sparse!

$$B_j = E_j + \rho_j \sum_{i=1}^{N} B_i F_{ij}, \qquad j = 1 \ldots N$$



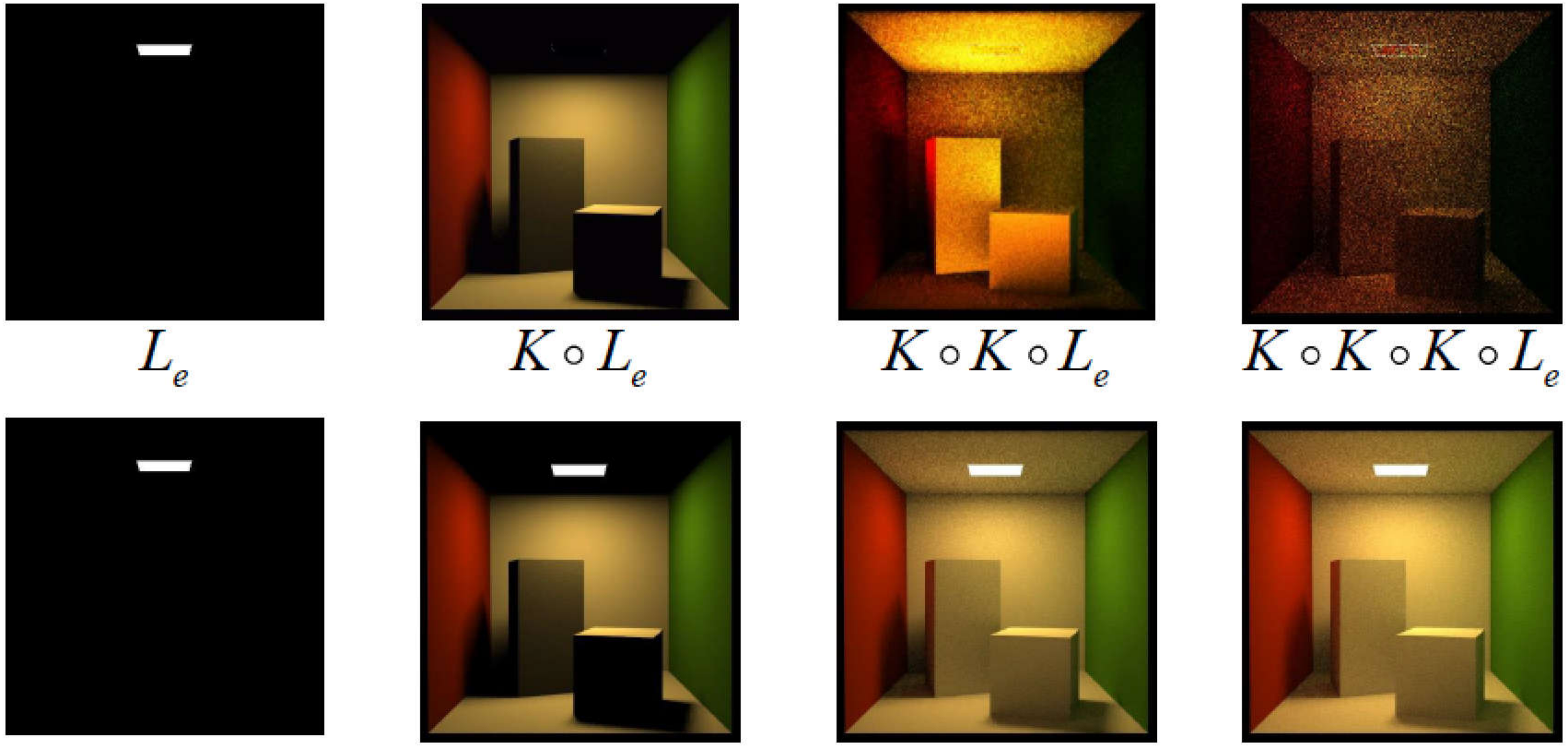
$$\begin{bmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1N} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \cdots & -\rho_2 F_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ -\rho_N F_{N1} & -\rho_N F_{N2} & \cdots & 1-\rho_N F_{NN} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_N \end{bmatrix}$$

$$L = (I - K)^{-1} \circ L_e$$

$$\begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix} = \begin{bmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1N} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \cdots & -\rho_2 F_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ -\rho_N F_{N1} & -\rho_N F_{N2} & \cdots & 1-\rho_N F_{NN} \end{bmatrix}^{-1} \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_N \end{bmatrix}$$

Successive Approximation

$$(I - K)^{-1} = \frac{1}{I - K} = I + K + K^2 + \cdots \quad \Rightarrow \quad L^n = L_e + K \circ L^n$$



L_e | $K \circ L_e$ | $K \circ K \circ L_e$ | $K \circ K \circ K \circ L_e$

L_e | $L_e + K \circ L_e$ | $L_e + \cdots K^2 \circ L_e$ | $L_e + \cdots K^3 \circ L_e$

Thank you!

Questions?