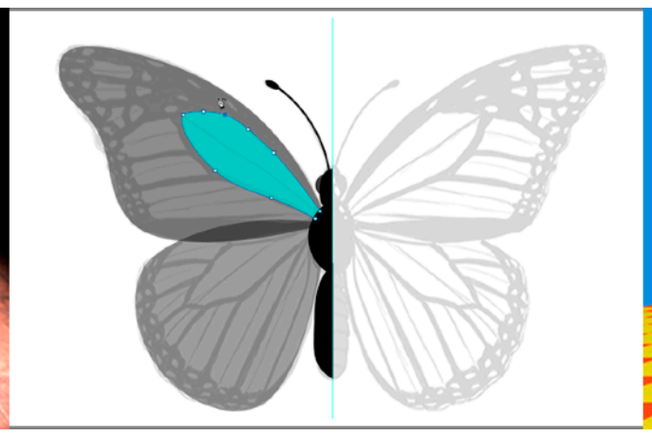
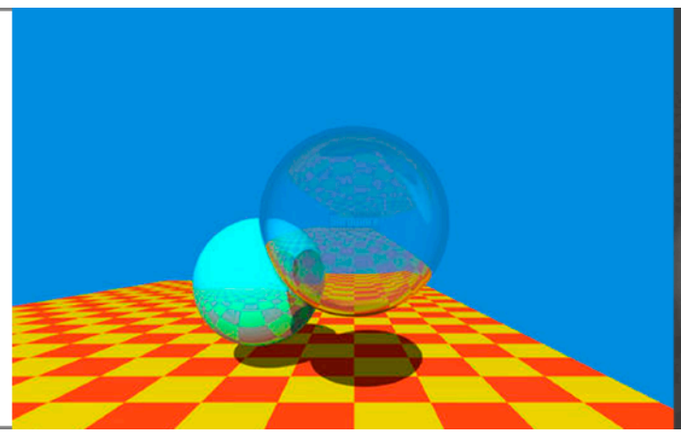
Introduction to Computer Graphics

GAMES101, Lingqi Yan, UC Santa Barbara

Lecture 7:
Shading 1 (Illumination, Shading and Graphics Pipeline)


http://www.cs.ucsb.edu/~lingqi/teaching/games101.html

Announcements

- Homework 1
  - 300+ submissions
  - Will start TA recruiting (from existing applications) soon
- Homework 2 will be out today
  - About Z-buffering
  - Much easier than HW1
- May need an additional lecture for shading

Last Lectures

- Rasterization
  - Rasterizing **one triangle**
  - Sampling theory
  - Antialiasing

Today

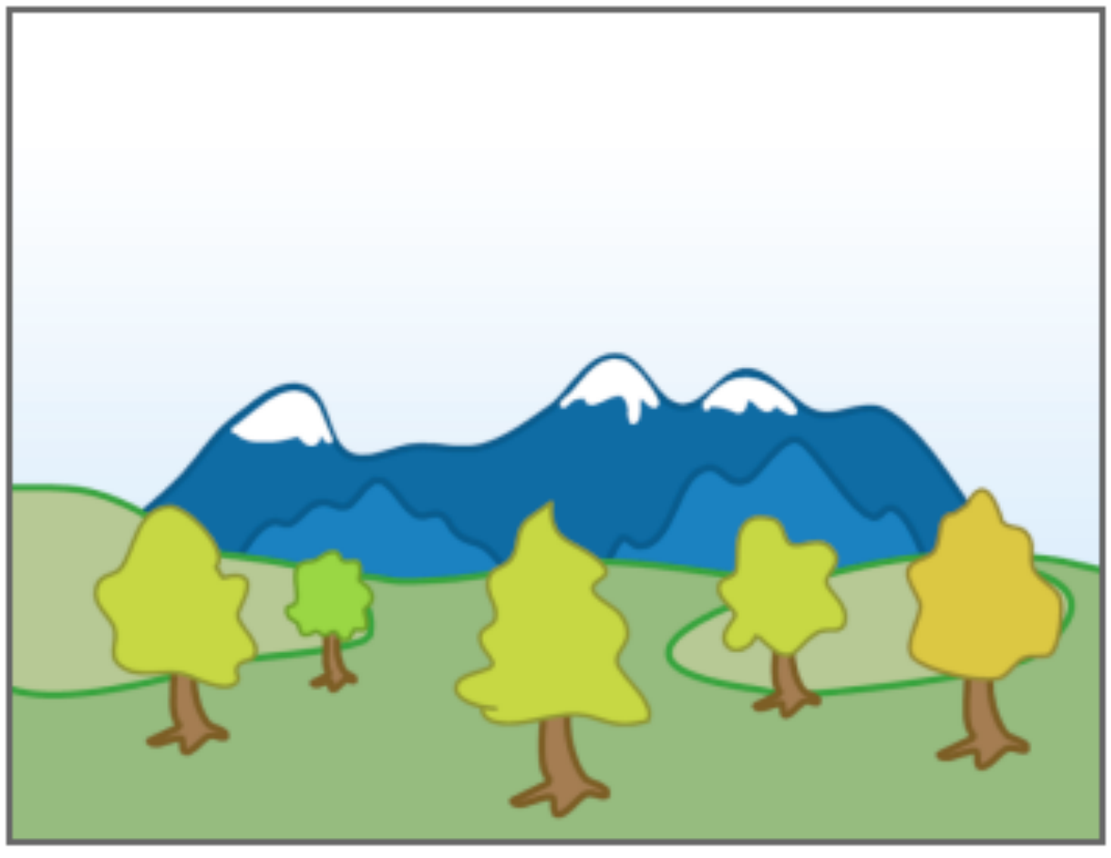
- Visibility / occlusion 遮挡视觉问题
  - Z-buffering
- Shading 着色问题
  - Illumination & Shading
  - Graphics Pipeline

Painter's Algorithm

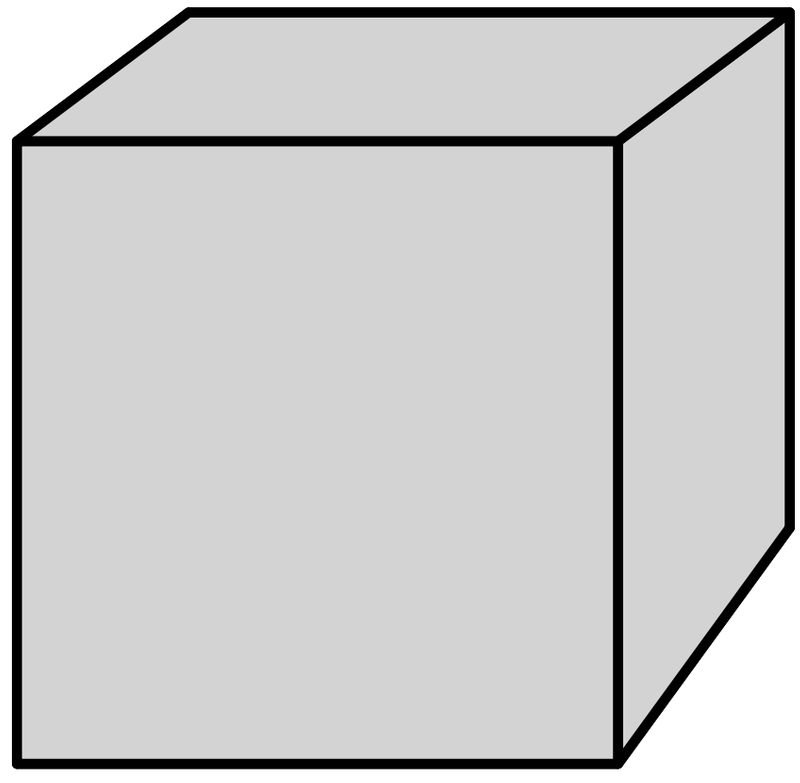
一种解决遮挡问题的算法： 画家算法， 由远及近画画，近处画面覆盖远处画面

Inspired by how painters paint

Paint from back to front, overwrite in the framebuffer



[Wikipedia]

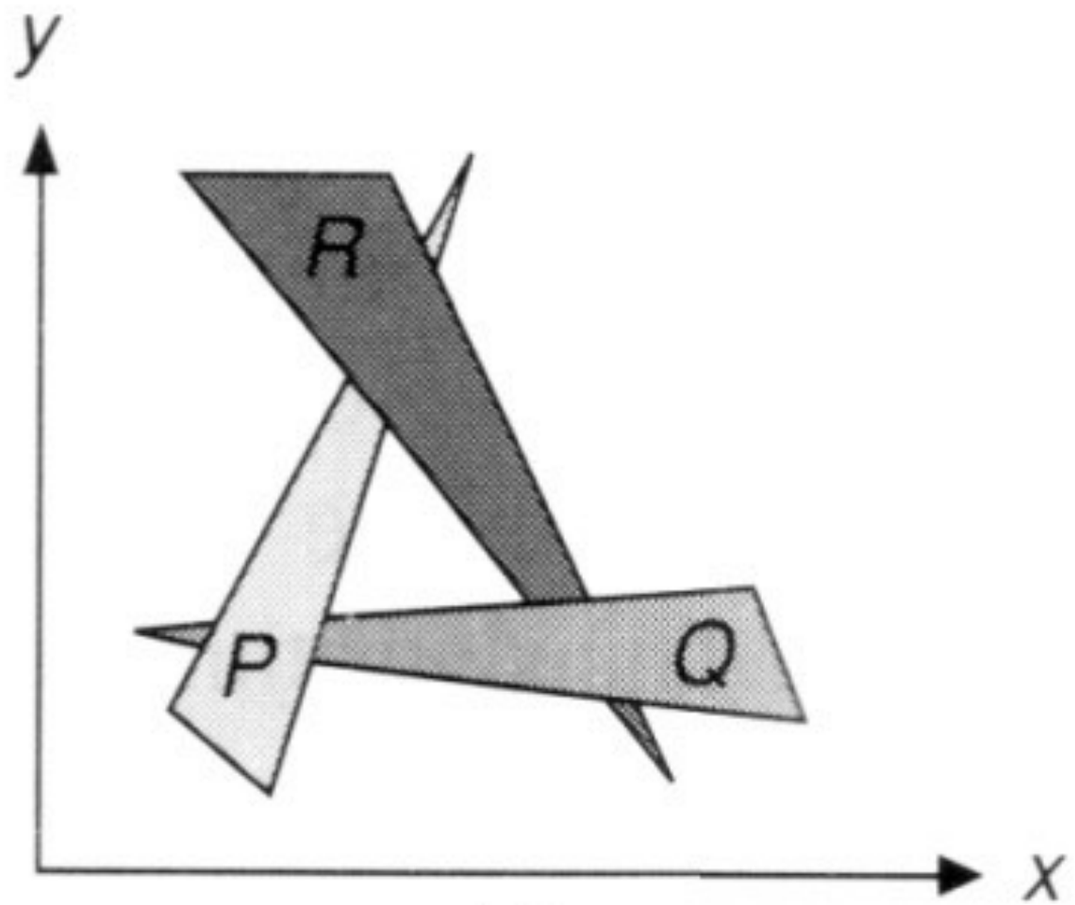


Painter's Algorithm

画家算法需要对所有的基元三角形排序，但是无法解决三个三角形互相重叠的问题

Requires sorting in depth (O(n log n) for n triangles)

Can have unresolvable depth order



[Foley et al.]

Z-Buffer

This is the algorithm that eventually won.

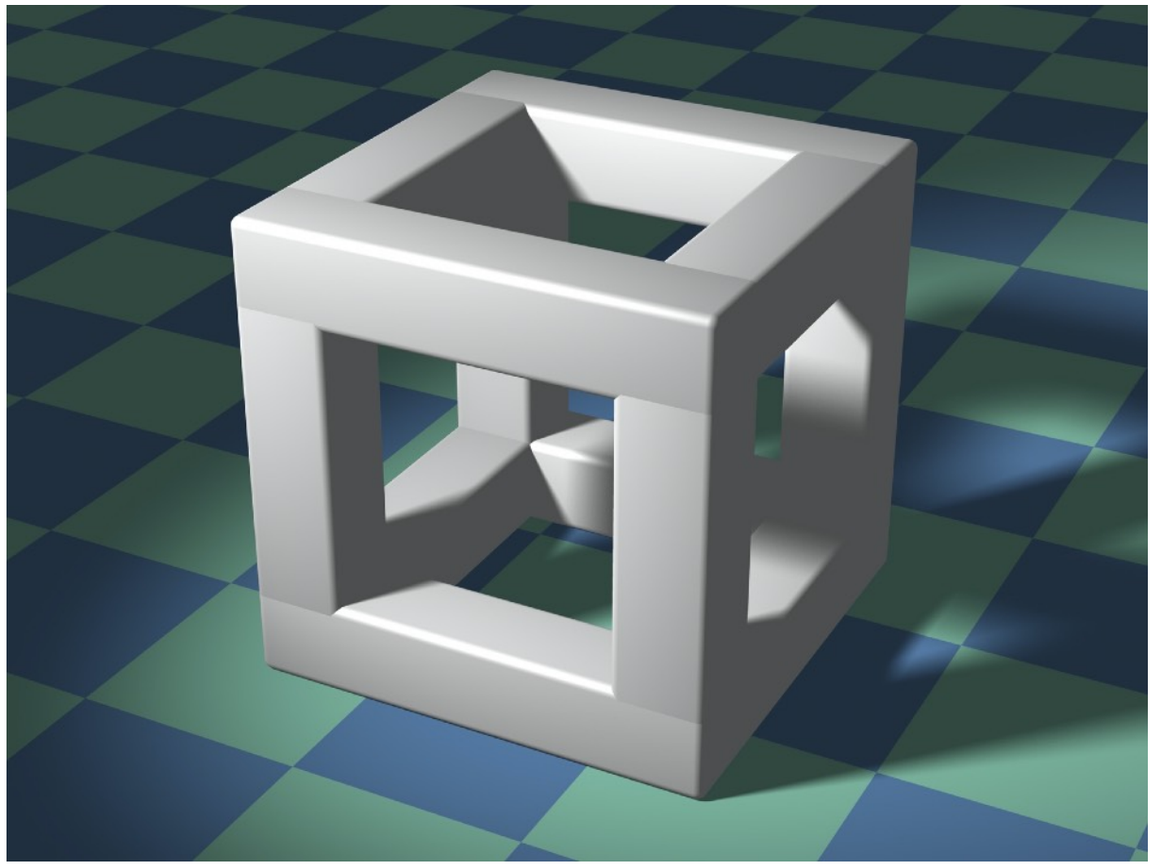
Idea: 对每个三角形基元多存一个深度，并会在屏幕上始终维护最小的深度值

- Store current min. z-value for each sample (pixel)
- Needs an additional buffer for depth values
  - frame buffer stores color values
  - depth buffer (z-buffer) stores depth

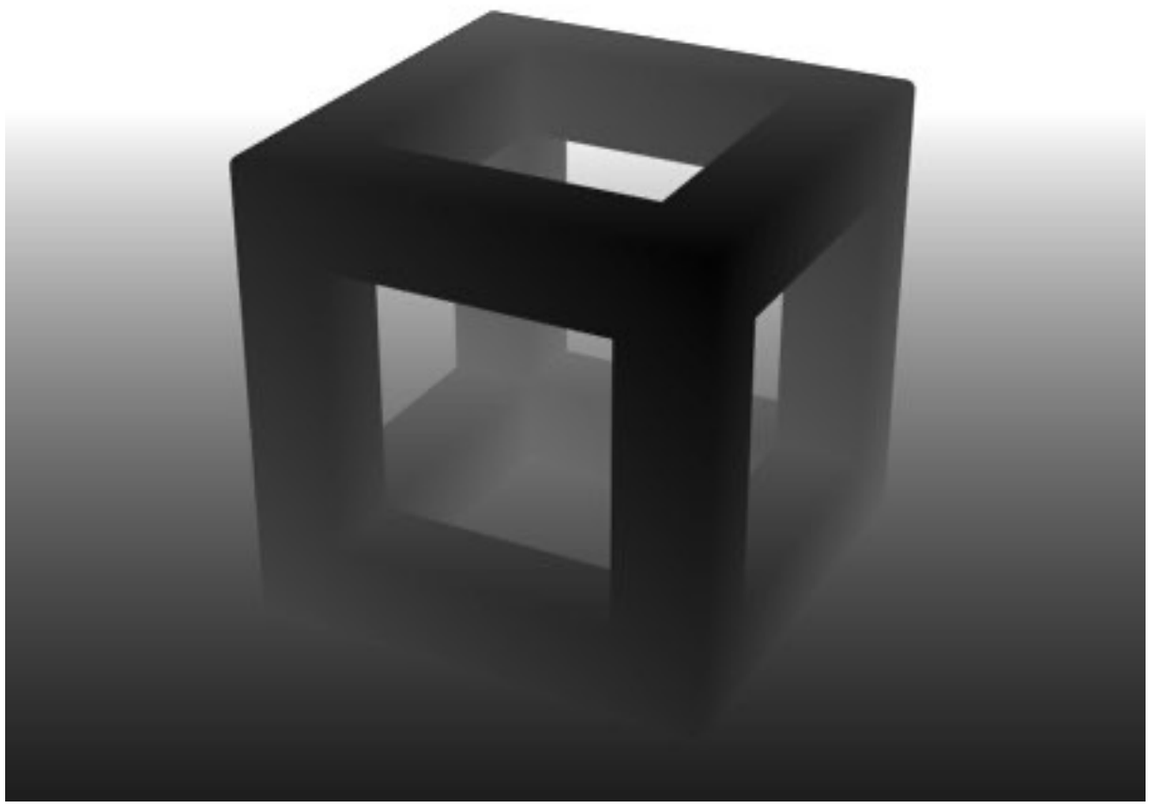
复杂度:O(n) for n triangles 并不是排序，而是求最值，需要保证三角形进入顺序和结果无关

IMPORTANT: For simplicity we suppose
z is always positive
(smaller z -> closer, larger z -> further)

Z-Buffer Example



Rendering



Depth / Z buffer

Image source: Dominic Alves, flickr.

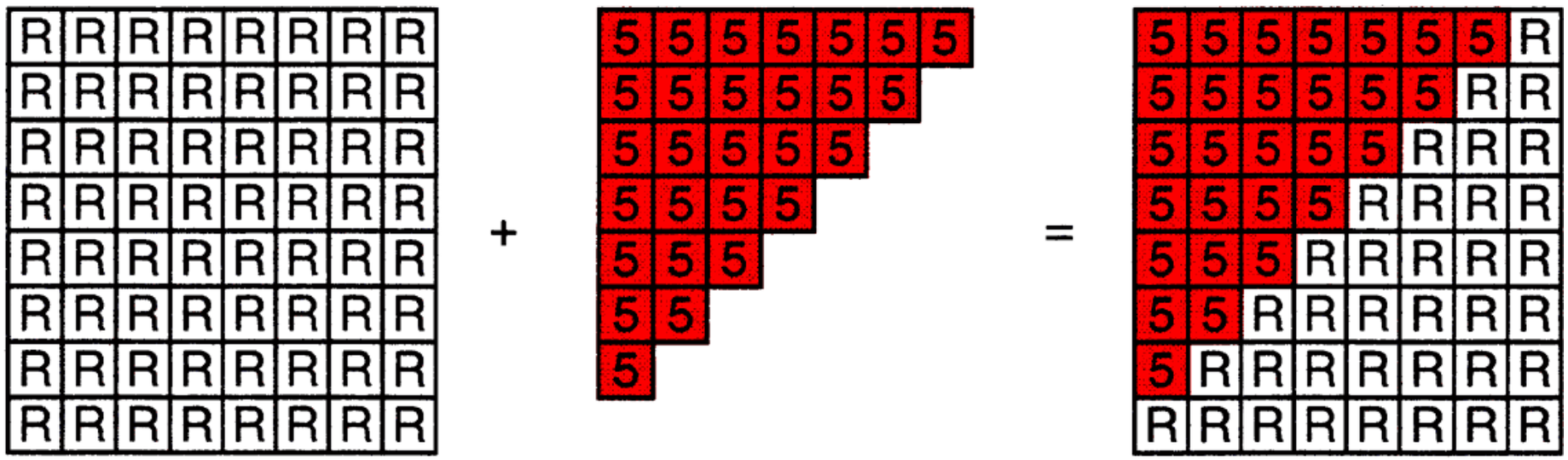
Z-Buffer Algorithm

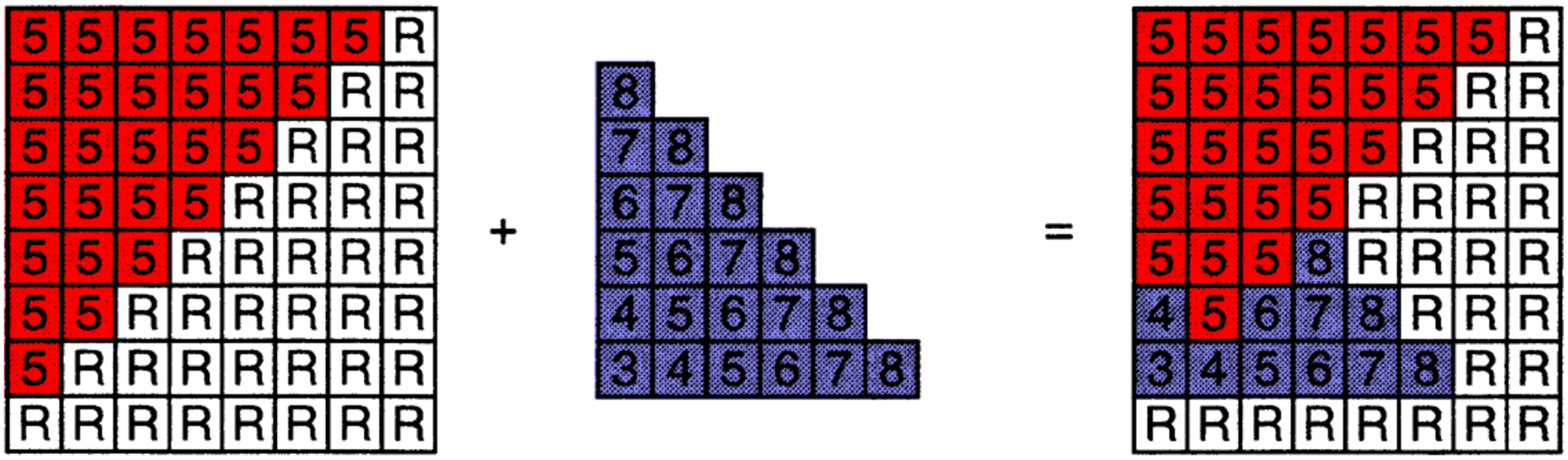
Initialize depth buffer to ∞

During rasterization:

```
for (each triangle T)
    for (each sample (x,y,z) in T)
        if (z < zbuffer[x,y])                // closest sample so far
            framebuffer[x,y] = rgb;          // update color
            zbuffer[x,y] = z;                // update depth
        else
            ;                                // do nothing, this sample is occluded
```

Z-Buffer Algorithm





Z-Buffer Complexity

Complexity

- O(*n*) for *n* triangles (assuming constant coverage)
- How is it possible to sort *n* triangles in linear time?

Drawing triangles in different orders?

Most important visibility algorithm

- Implemented in hardware for all GPUs

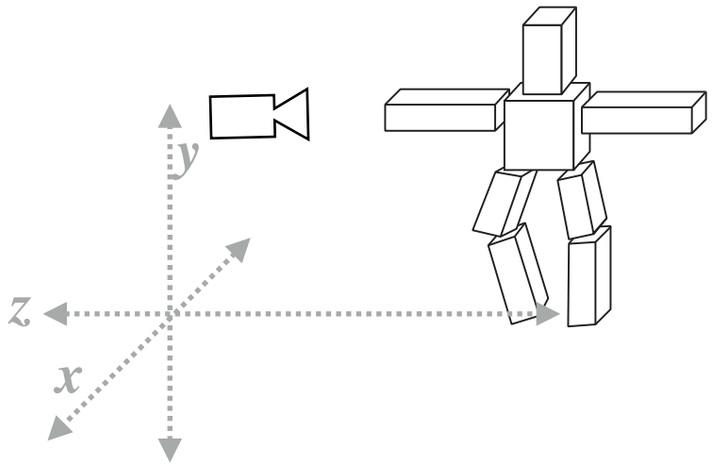
tips:无法处理透明物体，详情参考《入门精要》第八章

Questions?

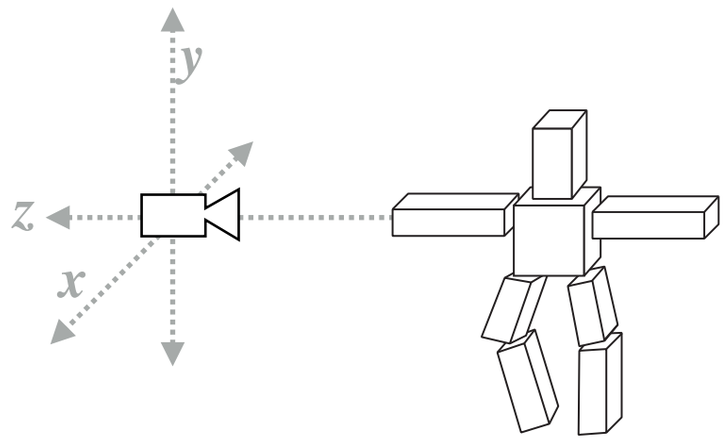
Today

- Visibility / occlusion
  - Z-buffering
- Shading
  - Illumination & Shading
  - Graphics Pipeline

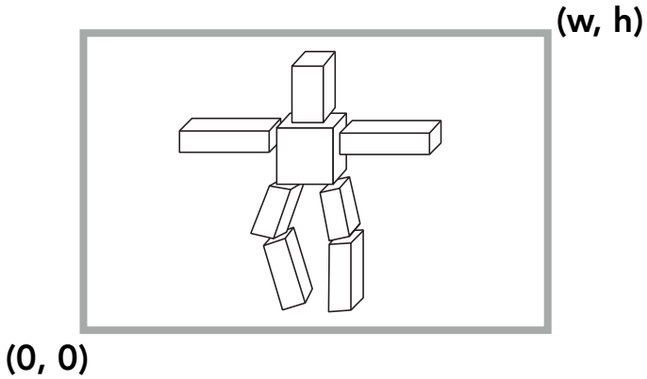
What We've Covered So Far



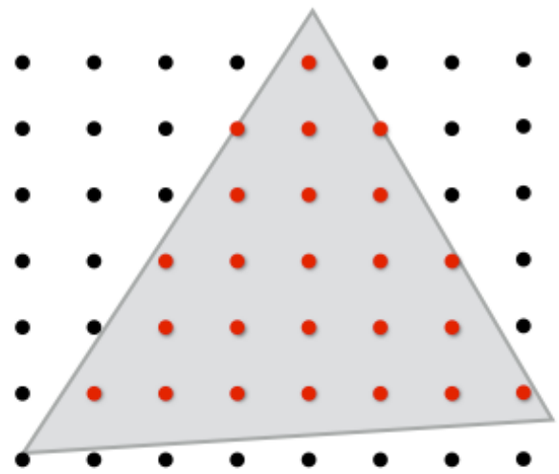
Position objects and the camera in the world



Compute position of objects relative to the camera

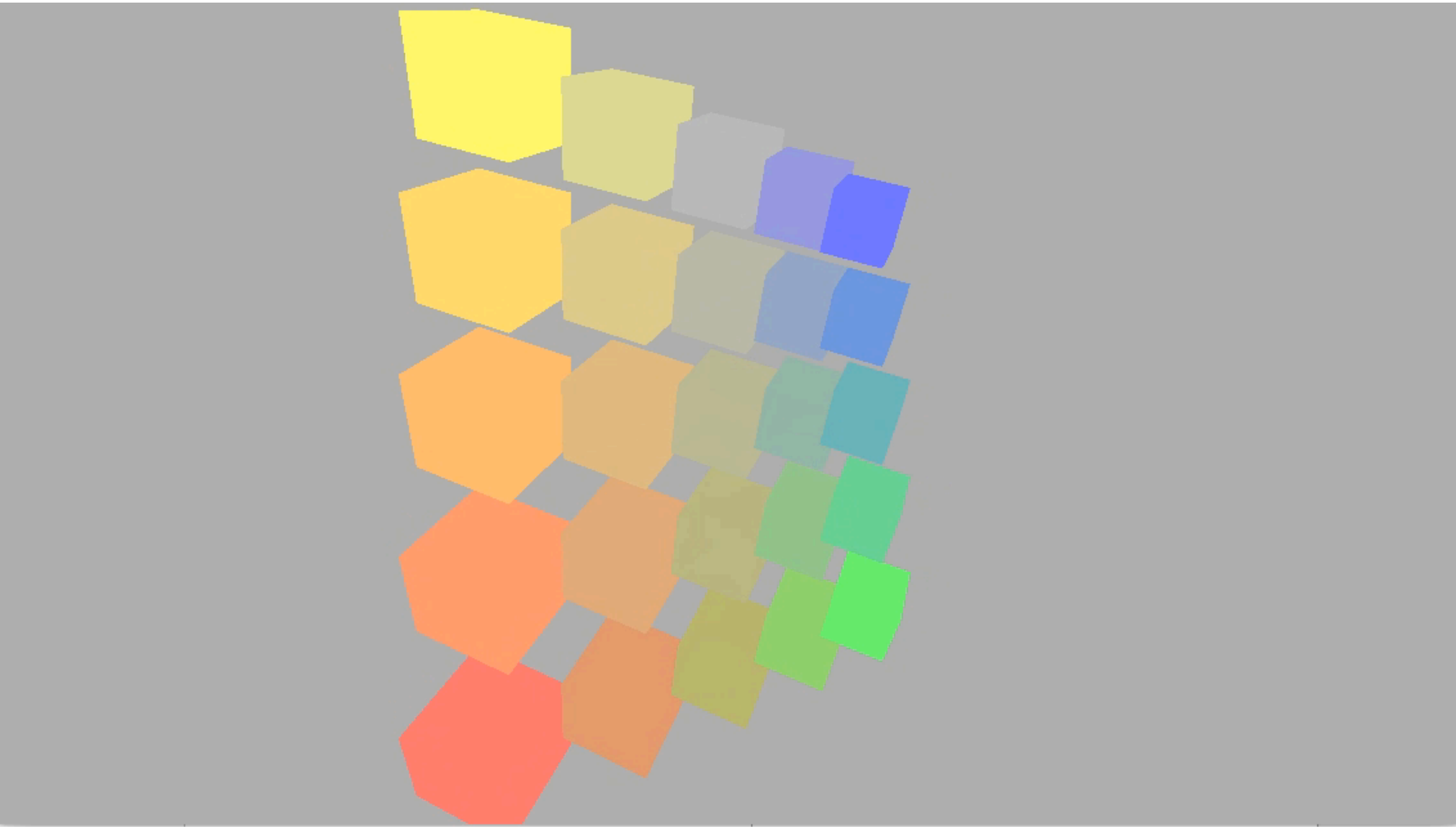


Project objects onto the screen

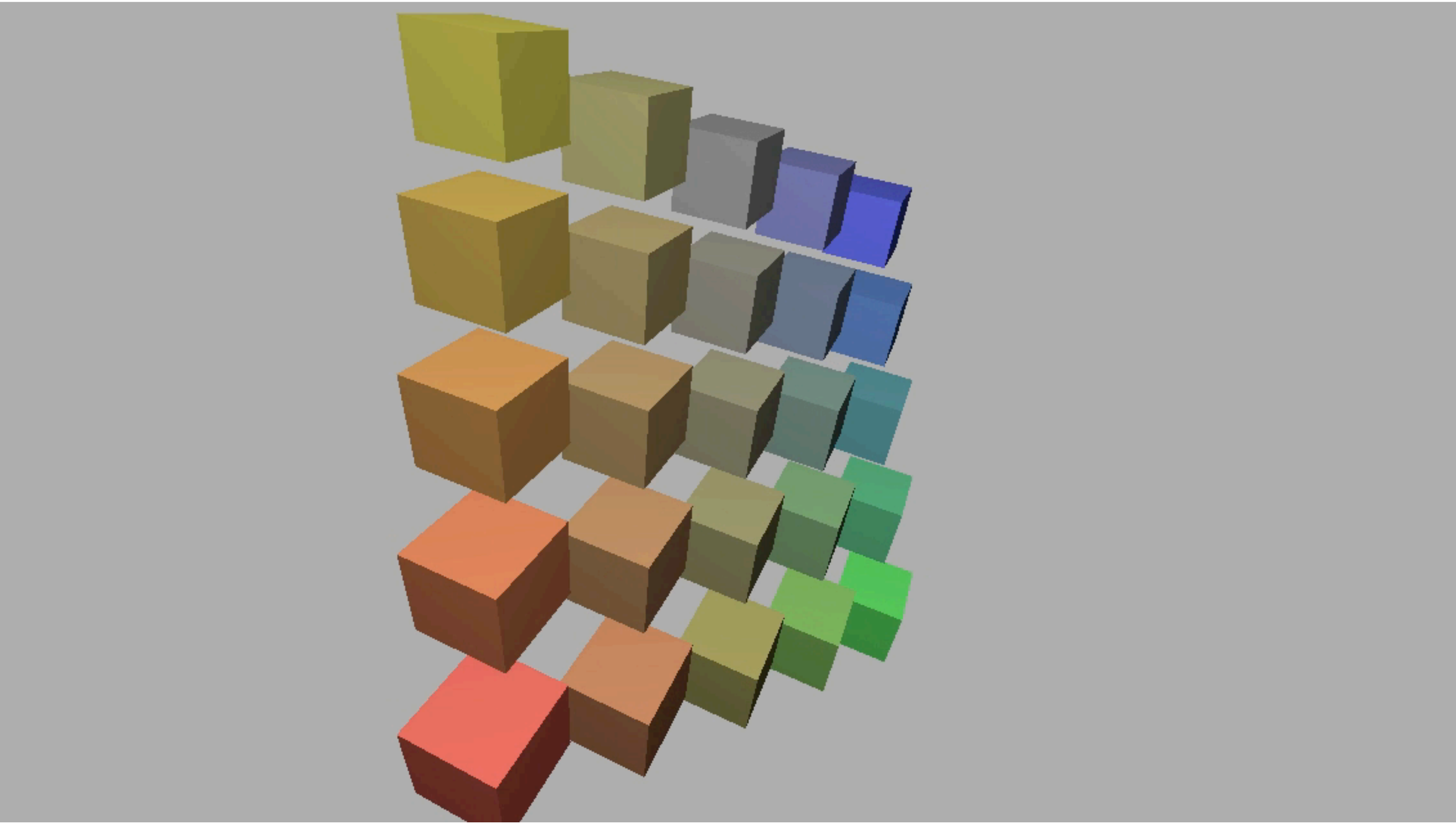


Sample triangle coverage

Rotating Cubes (Now You Can Do)



Rotating Cubes (Expected)



What Else Are We Missing?

Credit: Bertrand Benoit. “Sweet Feast,” 2009. [Blender /VRay]

Shading

Shading: Definition

- In Merriam-Webster Dictionary

  shad·ing, ['ʃeɪdɪŋ], noun
  The darkening or coloring of an illustration or diagram with parallel lines or a block of color.

- In this course

  The process of applying a material to an object.

A Simple Shading Model

(Blinn-Phong Reflectance Model)

Perceptual Observations

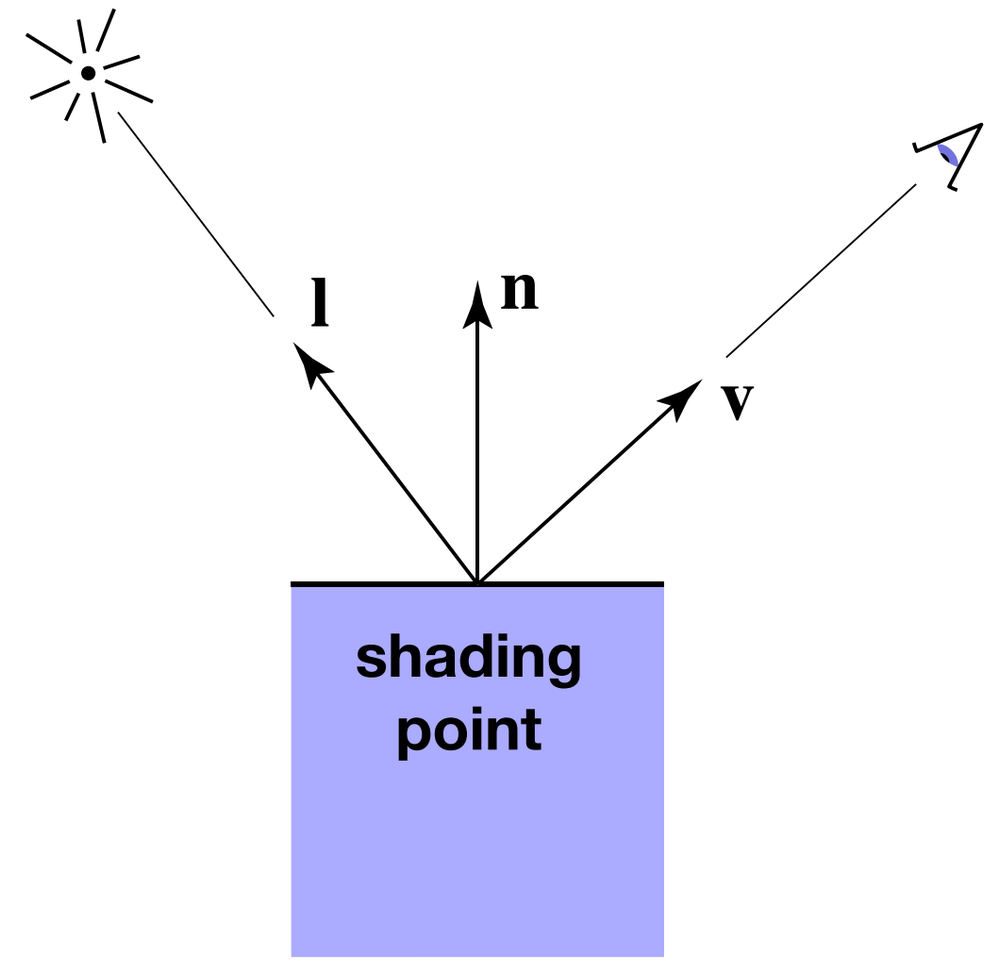
Photo credit: Jessica Andrews, flickr

Shading is Local

Compute light reflected toward camera
at a specific shading point

Inputs:

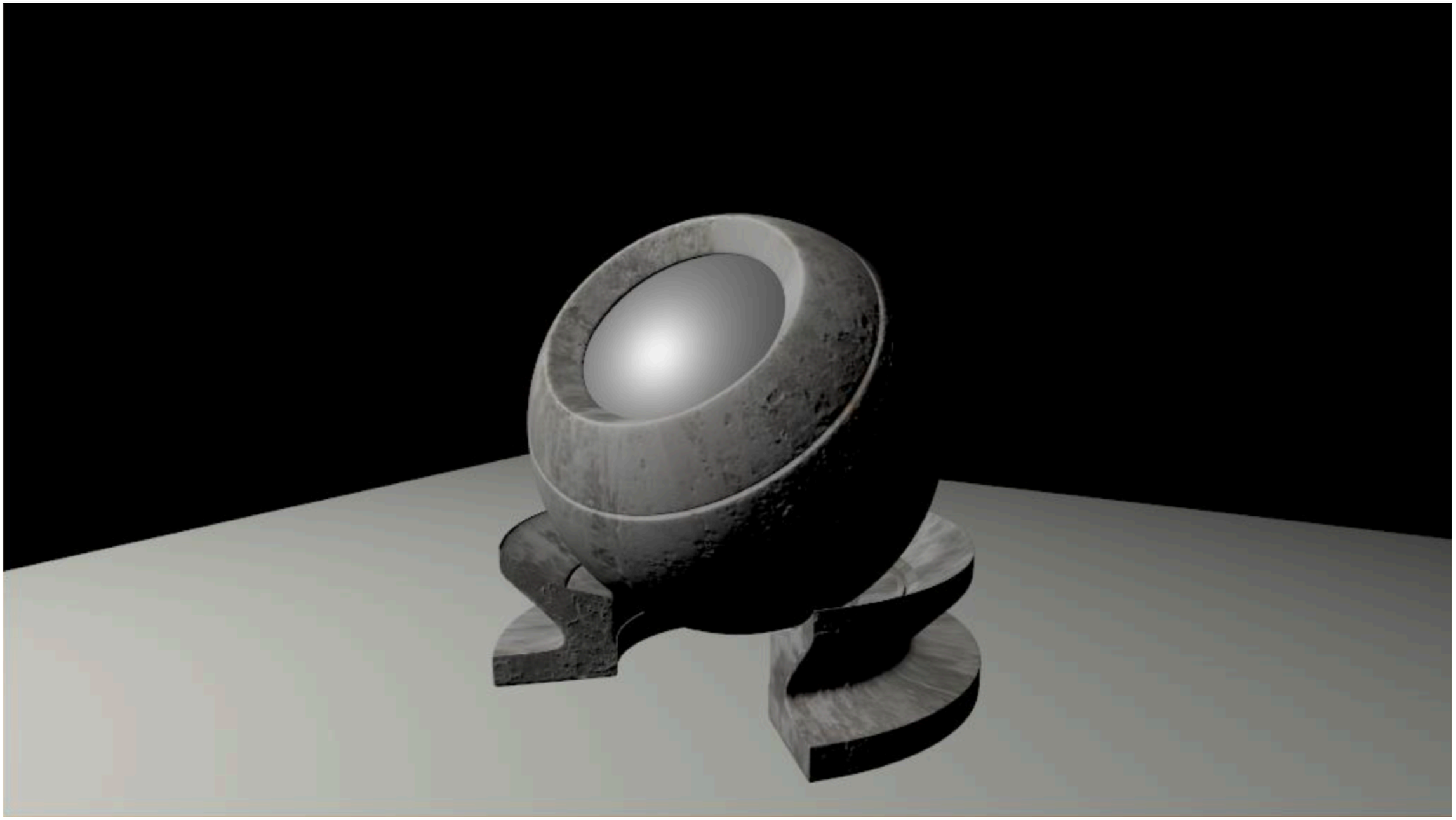
- Viewer direction, v 视角方向
- Surface normal, n 表面发现方向
- Light direction, l 光线方向
  (for each of many lights)
- Surface parameters 表面参数
  (color, shininess, …)



Shading is Local

No shadows will be generated! (**shading ≠ shadow**)

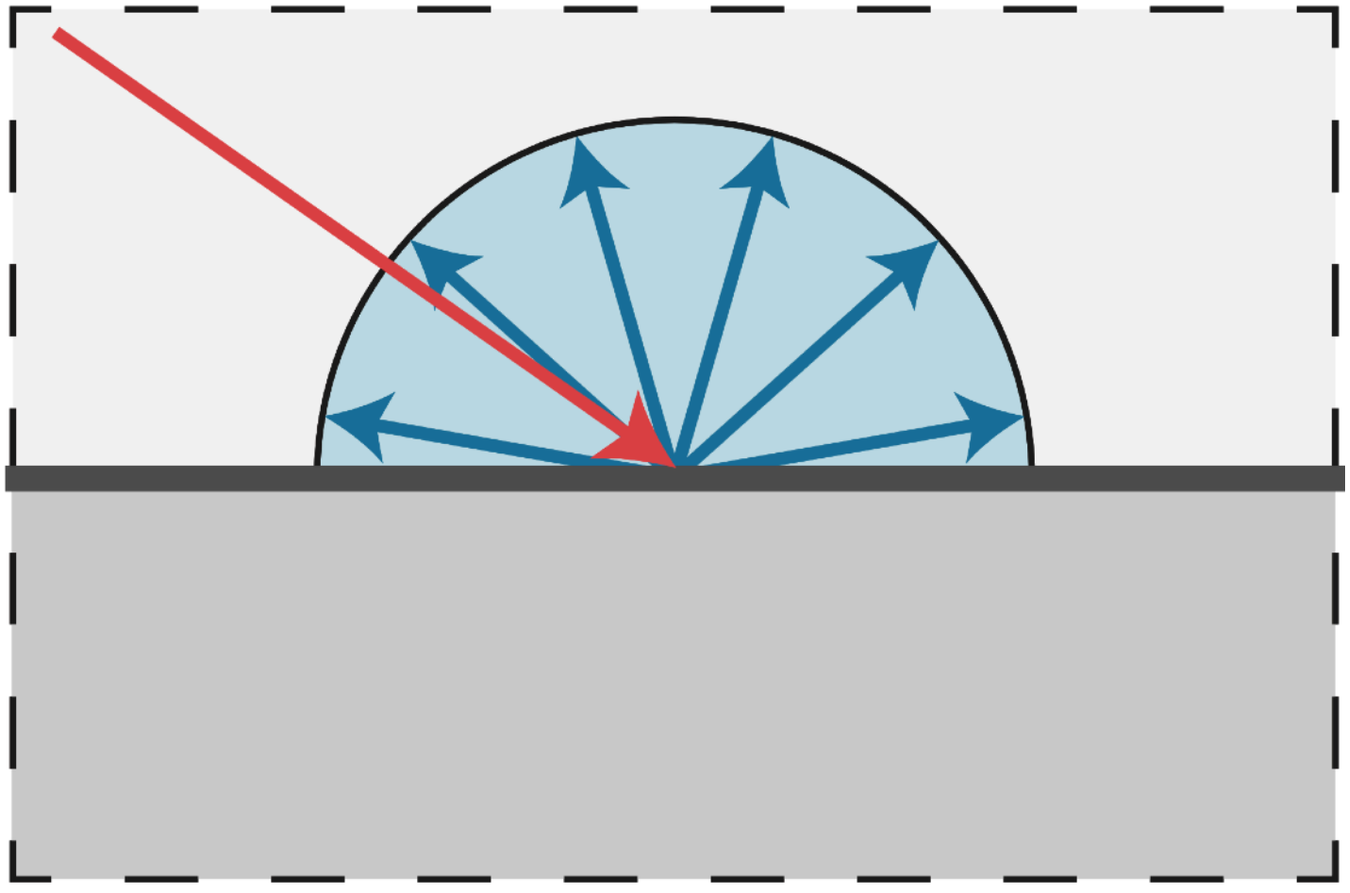
着色不考虑其它物体的存在，只考虑其自身，没有阴影



Diffuse Reflection

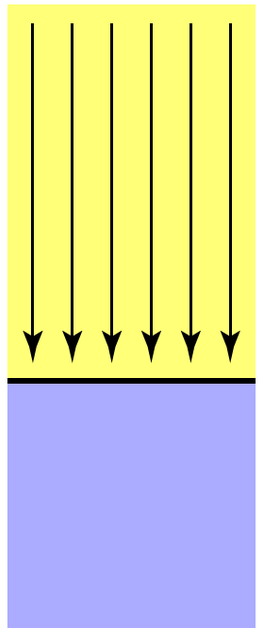
漫反射

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

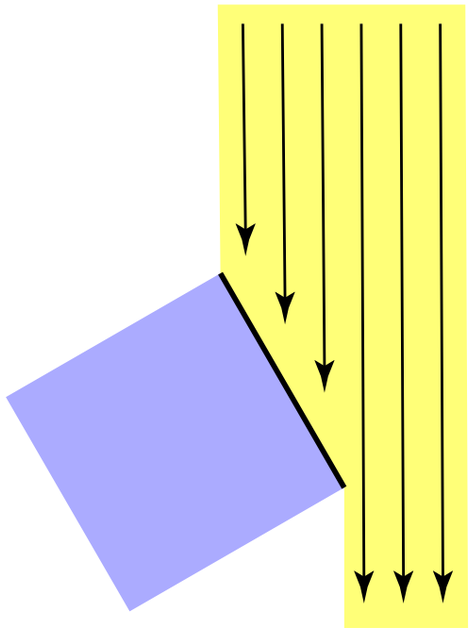


Diffuse Reflection

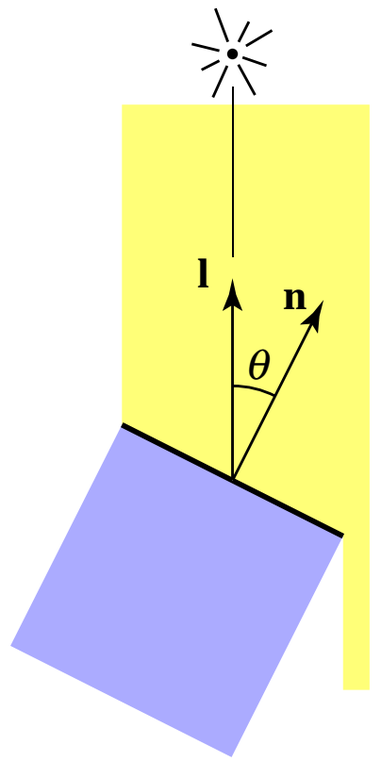
- But how much light (energy) is received?
  - Lambert's cosine law



Top face of cube receives a certain amount of light



Top face of 60° rotated cube intercepts half the light

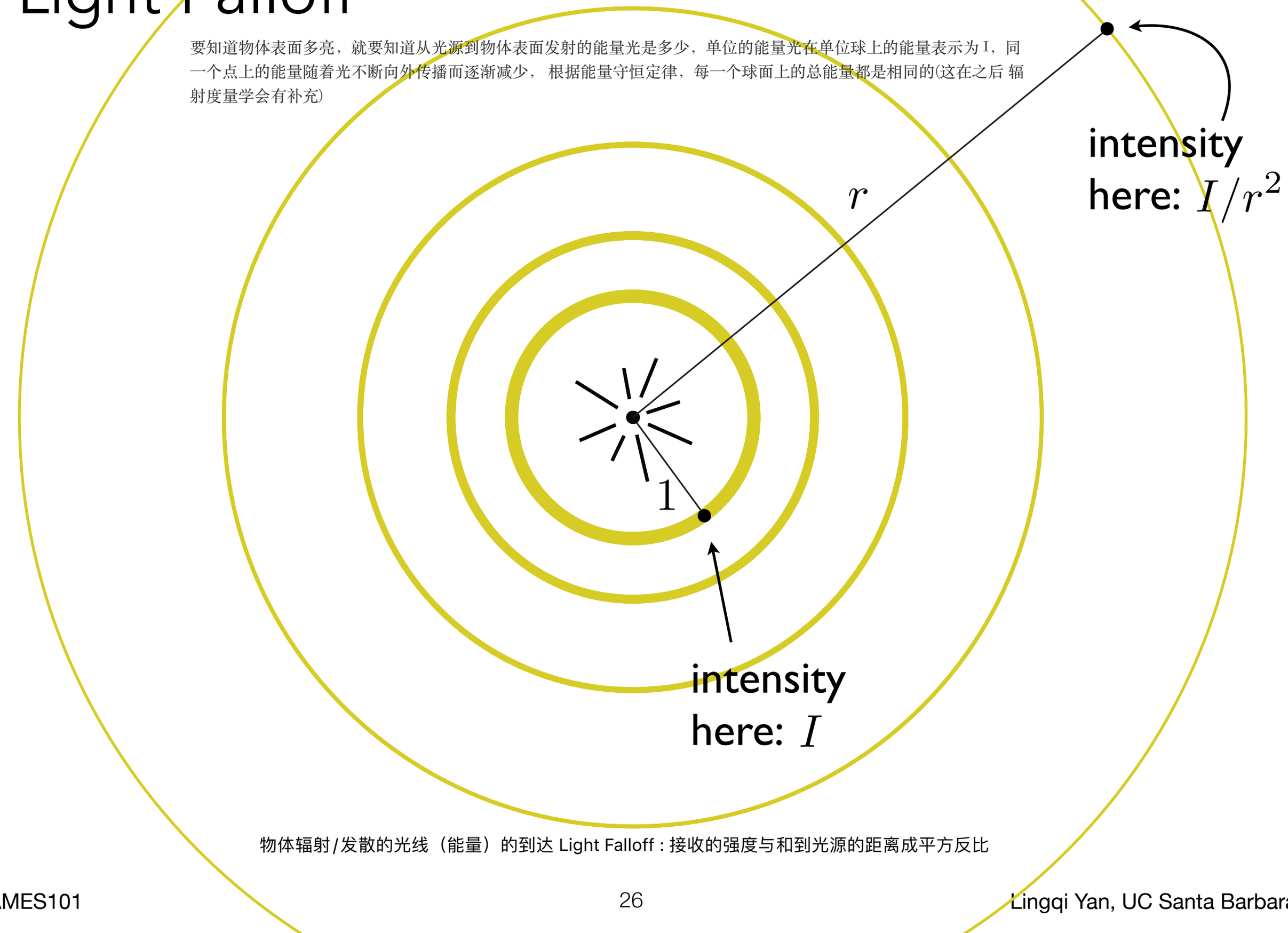


In general, light per unit area is proportional to $\cos\theta = l \cdot n$

Lambert余弦定律说明，接收到的光线能量，与接收方向和法线方向的夹角的余弦，成正比

Light Falloff

要知道物体表面多亮，就要知道从光源到物体表面发射的能量光是多少，单位的能量光在单位球上的能量表示为 I，同一个点上的能量随着光不断向外传播而逐渐减少， 根据能量守恒定律，每一个球面上的总能量都是相同的(这在之后 辐射度量学会有补充)

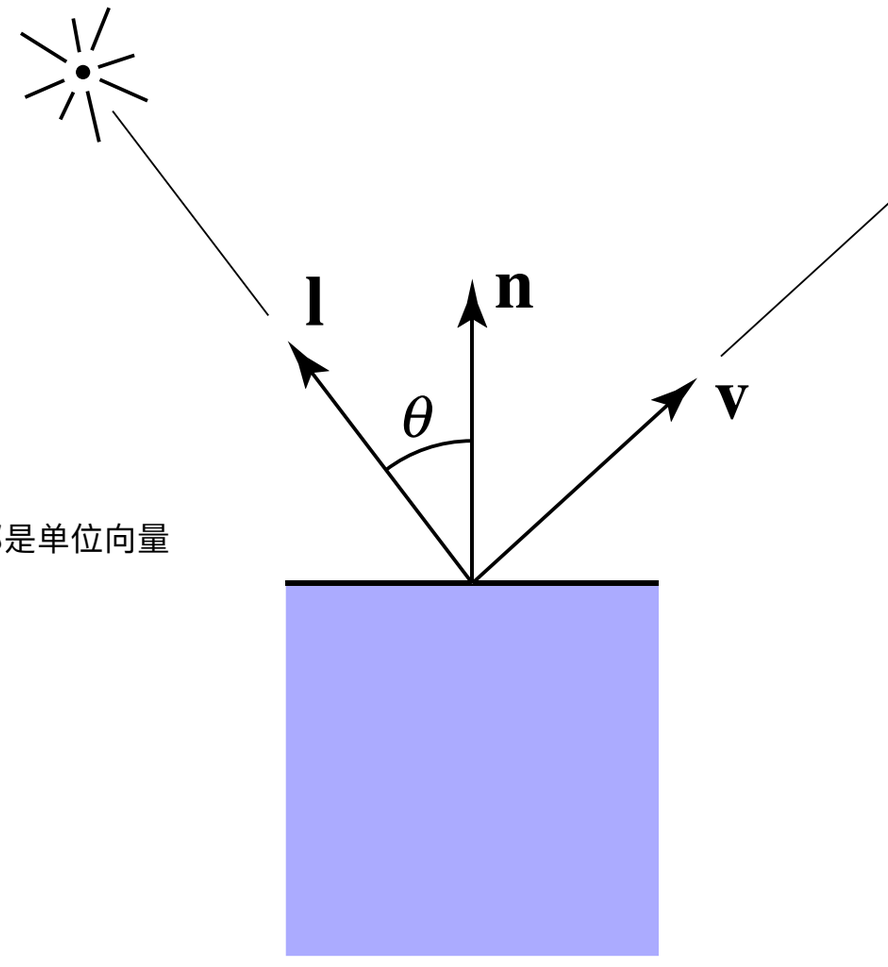


物体辐射/发散的光线（能量）的到达 Light Falloff : 接收的强度与和到光源的距离成平方反比

Lambertian (Diffuse) Shading

计算物体表面着色点反射多少能量光给摄像机接收

Shading independent of view direction



这里都是单位向量

energy arrived
at the shading point

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

shader point 距离光源的距离

diffuse
coefficient
(color)

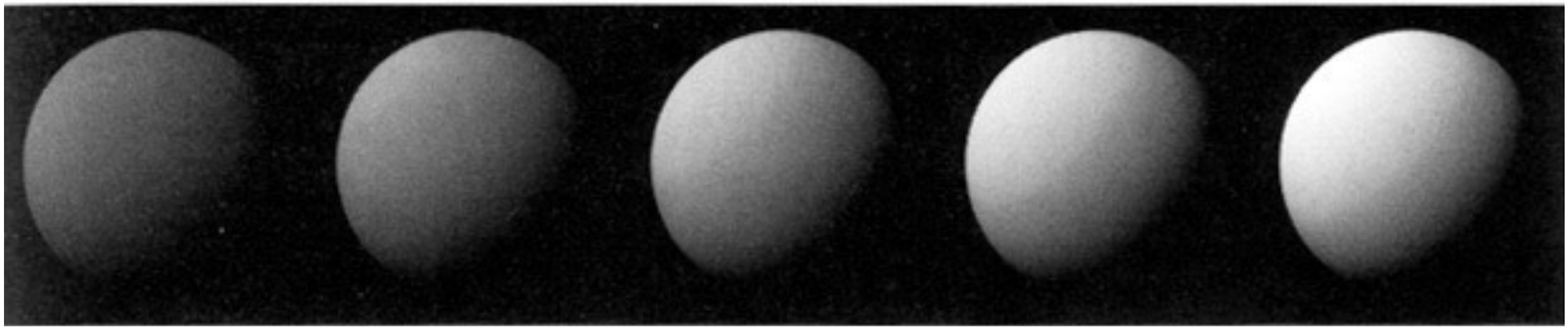
energy received
by the shading point

diffusely
reflected light

用一个系数 表示该着色点的光吸收率，范围是[0, 1]，如果该系数是 0，证明该着色点完全吸收能量，反之，如果 是 1，代表该点完全不吸收能量

Lambertian (Diffuse) Shading

Produces diffuse appearance



k_d ⟶

[Foley et al.]

把这个系数看作是 ，当 kd = 0 时，此时对应着RGB值是:[0, 0, 0]，在计算机里表示为黑色，黑色正是完 全吸收光的颜色，反之，当 kd = 1时，对应着 [255, 255, 255]，在计算机里表示为白色，白色正是完全反射光的颜 色

Thank you!

(And thank Prof. Ravi Ramamoorthi and Prof. Ren Ng for many of the slides!)