

第六章 真实感图形生成

一、概述

- 1、定义 光栅显示器可以对每一个像素的颜色进行定义，因此，一个物体的显示就是每一点在一定光照环境下的反射值。即：

物体描述 ➡ 光照模型 ➡ 计算每一点的反射值 ➡ 显示 ➡ 图形

- 2、一般过程

几何造型：布景、物体的位置、物体的数学描述（多边形拟合、二次曲面、参数样条曲面、分形、八叉树等）

投影变换：平行投影、透视投影

消隐：画家算法、Z-buffer算法、ScanLine算法、八叉树算法等

光照模型：Lambert模型、Phong模型、witted模型等

画面绘制：多边形绘制、光线跟踪、辐射度方法等

一、概述

- 3、彩色模型

什么是颜色？

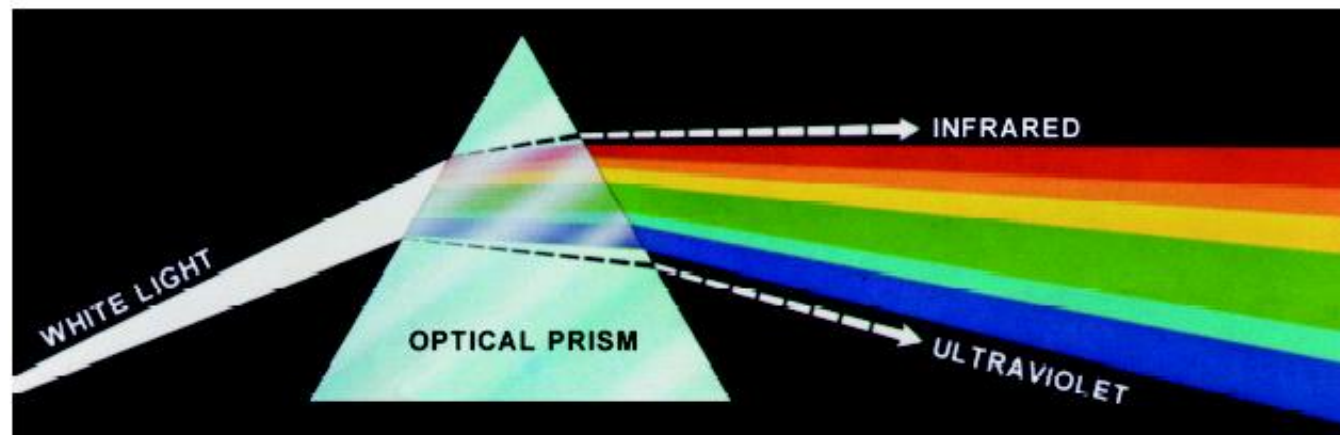
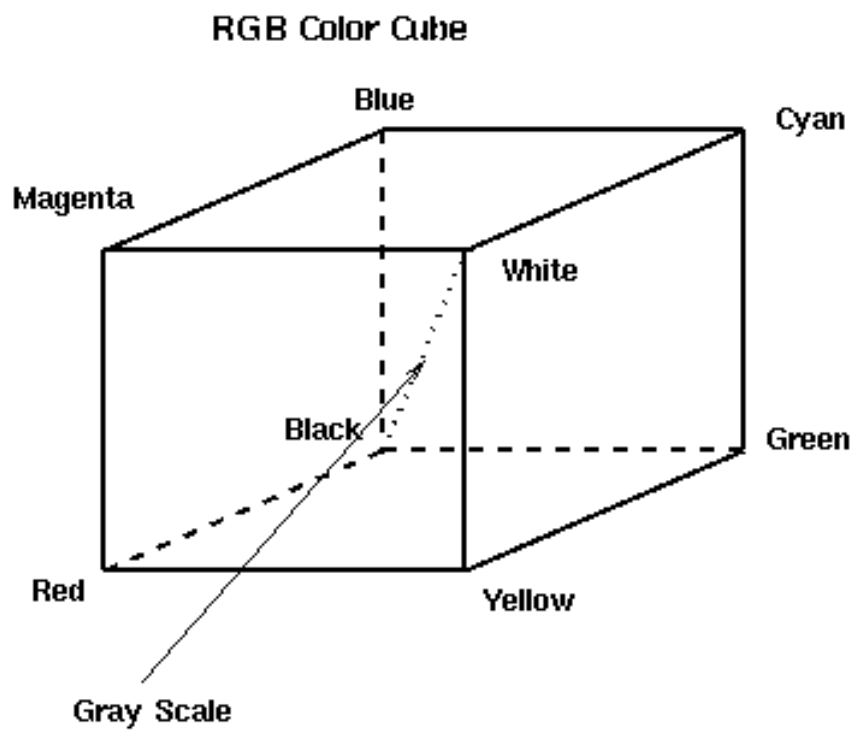


FIGURE 6.1 Color spectrum seen by passing white light through a prism. (Courtesy of the General Electric Co., Lamp Business Division.)

1666, Sir Isaac Newton, optical prism

一、概述

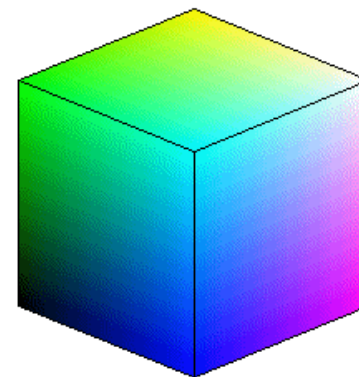
- 3、彩色模型



RGB模型



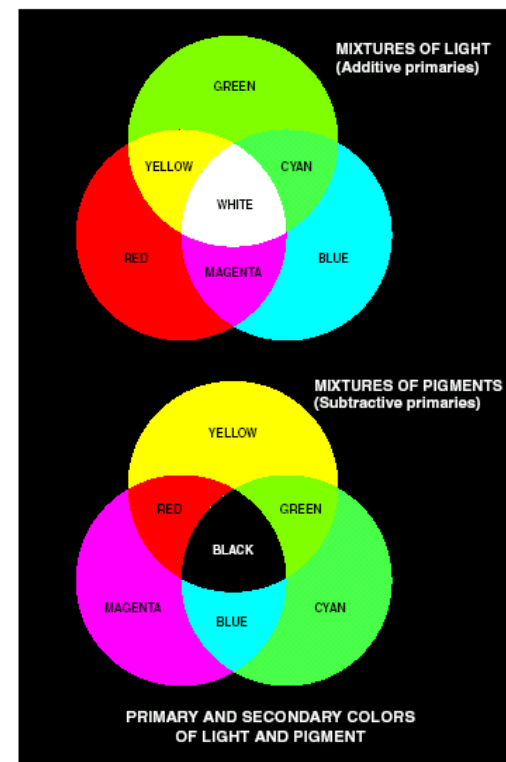
图 6.8 RGB 24-bit color cube.



一、概述

- 3、彩色模型

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$



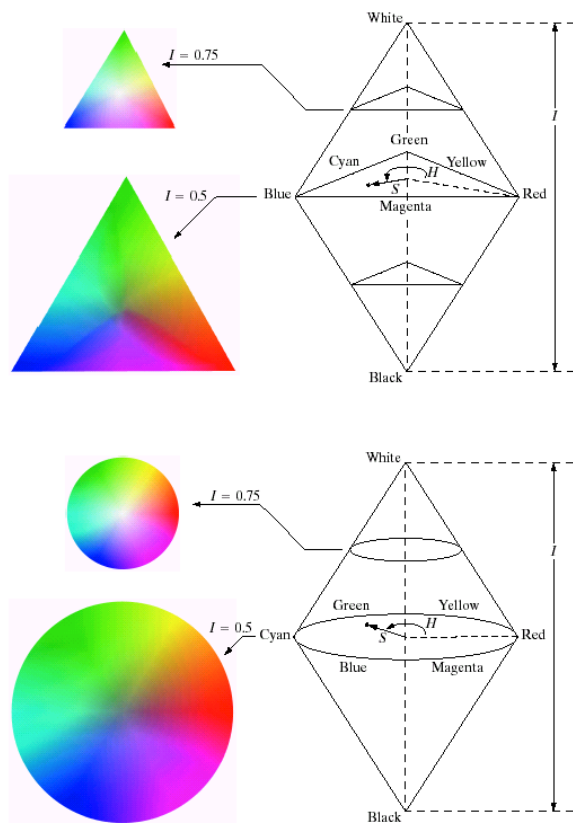
a
b

FIGURE 6.4 Primary and secondary colors of light and pigments. (Courtesy of the General Electric Co., Lamp Business Division.)

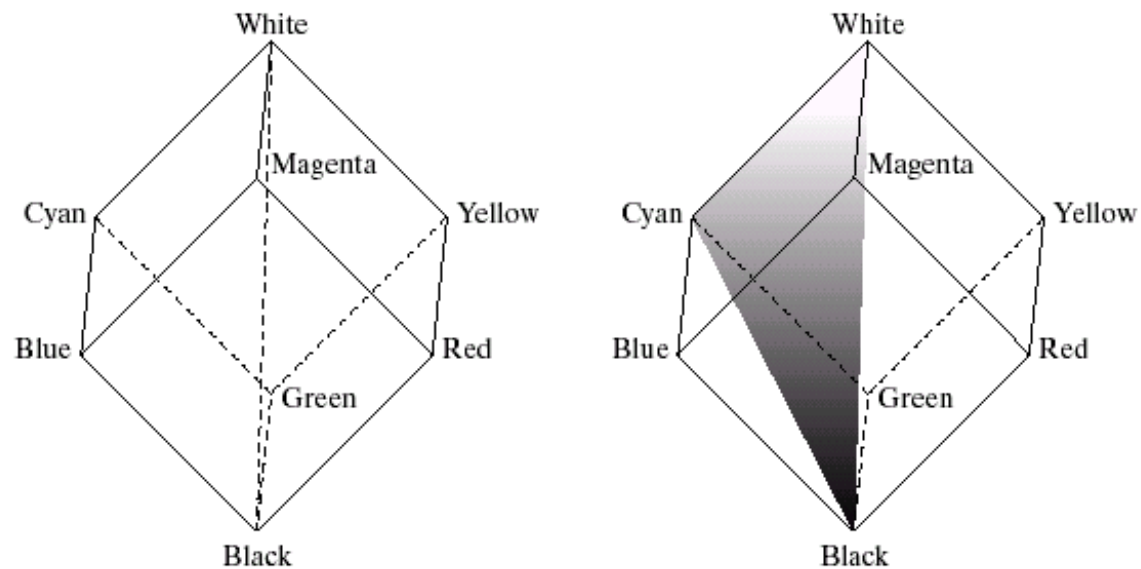
CMY模型

一、概述

• 3、彩色模型



HSI模型



a b

FIGURE 6.12 Conceptual relationships between the RGB and HSI color models.

$$I = \frac{1}{3}(R + G + B)$$

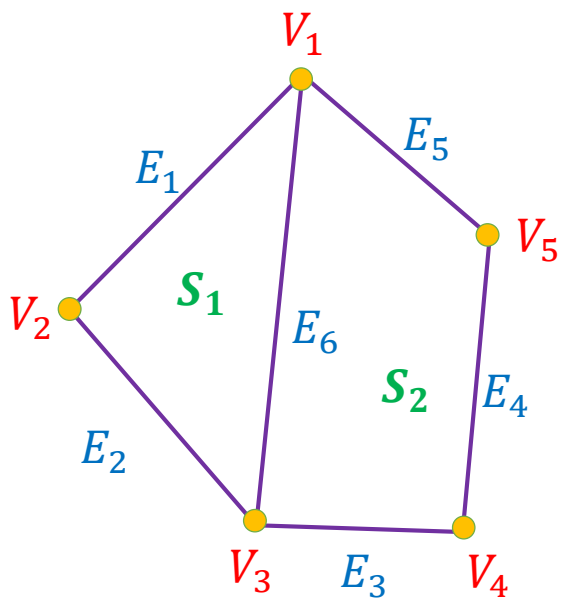
$$S = 1 - \frac{3}{(R + G + B)}[\min(R, G, B)]$$

$$\theta = \arccos \left\{ \frac{[(R - G) + (R - B)] / 2}{[(R - G)^2 + (R - B)(G - B)]^{1/2}} \right\}$$

$$H = \begin{cases} \theta & B \leq G \\ 360 - \theta & \text{others} \end{cases}$$

二、几何造型

- 1、多边形拟合



VT(vertex table)

V	(x,y,z)
V_1	(x_1, y_1, z_1)
V_2	(x_2, y_2, z_2)
V_3	(x_3, y_3, z_3)
V_4	(x_4, y_4, z_4)
V_5	(x_5, y_5, z_5)

ET(edge table)

E	V
E_1	V_1, V_2
E_2	V_2, V_3
E_3	V_3, V_4
E_4	V_4, V_5
E_5	V_5, V_1
E_6	V_1, V_3

PT(polygon table)

S	E
S_1	E_1, E_2, E_6
S_2	E_3, E_4, E_5, E_6

二、几何造型

- 1、多边形拟合

多边形的平面方程： $ax + by + cz + d = 0$

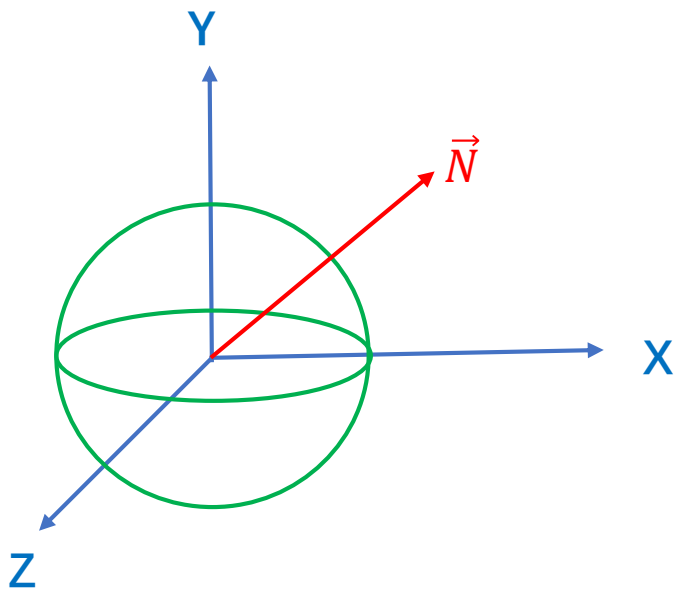
其中：

$$\begin{aligned}a &= y_1(z_2 - z_1) + y_2(z_3 - z_2) + y_3(z_1 - z_3) \\b &= z_1(x_2 - x_1) + z_2(x_3 - x_2) + z_3(x_1 - x_3) \\c &= x_1(y_2 - y_1) + x_2(y_3 - y_2) + x_3(y_1 - y_3) \\d &= -x_1(y_2z_3 - y_3z_2) - x_2(y_3z_1 - y_1z_3) - x_3(y_1z_2 - y_2z_1)\end{aligned}$$

法向矢量： $\vec{N} = \left(\frac{a}{\sqrt{a^2+b^2+c^2}}, \frac{b}{\sqrt{a^2+b^2+c^2}}, \frac{c}{\sqrt{a^2+b^2+c^2}} \right)$

二、几何造型

- 2、二次曲面
 - 球面



球面方程: $x^2 + y^2 + z^2 = R^2$

任一点 (x_0, y_0, z_0) 的法向矢量:

$$\vec{N} = \left(\frac{x_0}{R}, \frac{y_0}{R}, \frac{z_0}{R} \right)$$

二、几何造型

- 2、二次曲面
 - 柱面

柱面方程：

$$\begin{cases} x^2 + z^2 = R^2 \\ 0 \leq y \leq H \end{cases}$$

柱面任一点 (x_0, y_0, z_0) 的法向矢量：

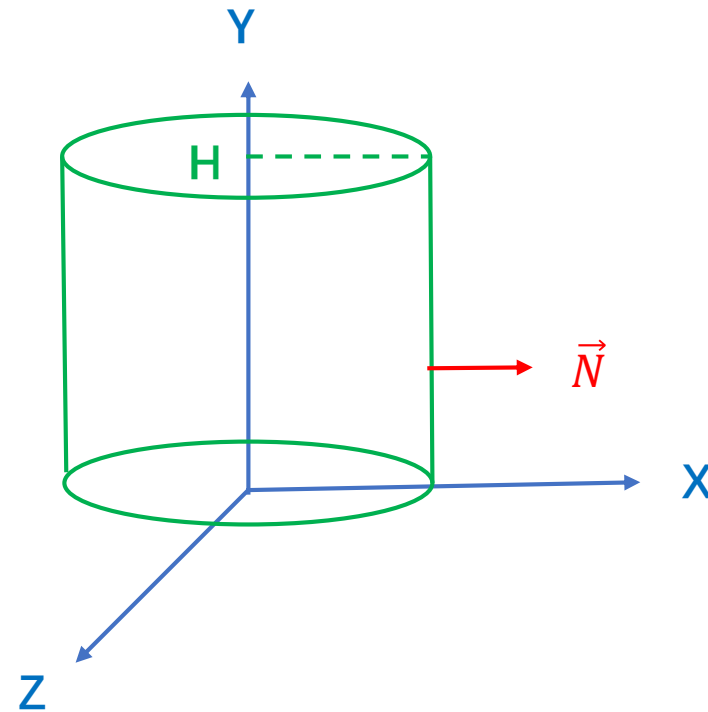
$$\vec{N} = \left(\frac{x_0}{R}, 0, \frac{z_0}{R} \right)$$

柱底任一点 (x_0, y_0, z_0) 的法向矢量：

或：

$$\vec{N} = (0, 1, 0)$$

$$\vec{N} = (0, -1, 0)$$

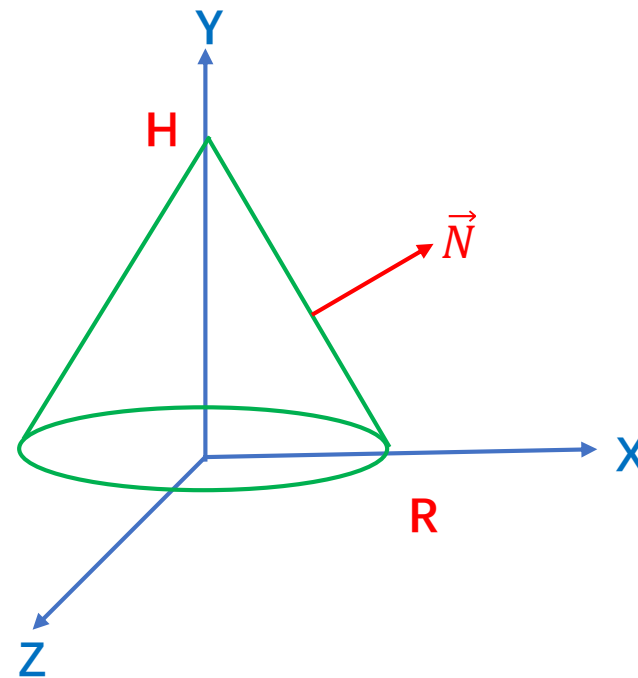


二、几何造型

- 2、二次曲面
 - 锥面

锥面方程：

$$\begin{cases} x^2 + z^2 = (R - \frac{y}{m})^2 \\ y = H - \frac{H}{R}x \\ 0 \leq y \leq H \end{cases}$$



锥面任一点 (x_0, y_0, z_0) 的法向矢量： $\vec{N} = (\frac{x_0}{R}, (H - y_0) \frac{R}{H^2}, \frac{z_0}{R})$

锥底任一点 (x_0, y_0, z_0) 的法向矢量： $\vec{N} = (0, -1, 0)$

二、几何造型

- 3、样条曲面

- 可以精确地计算出每一点的位置、法向矢量 $\vec{N} = \frac{p_{ij}^u \times p_{ij}^v}{|p_{ij}^u \times p_{ij}^v|}$
- 计算量大

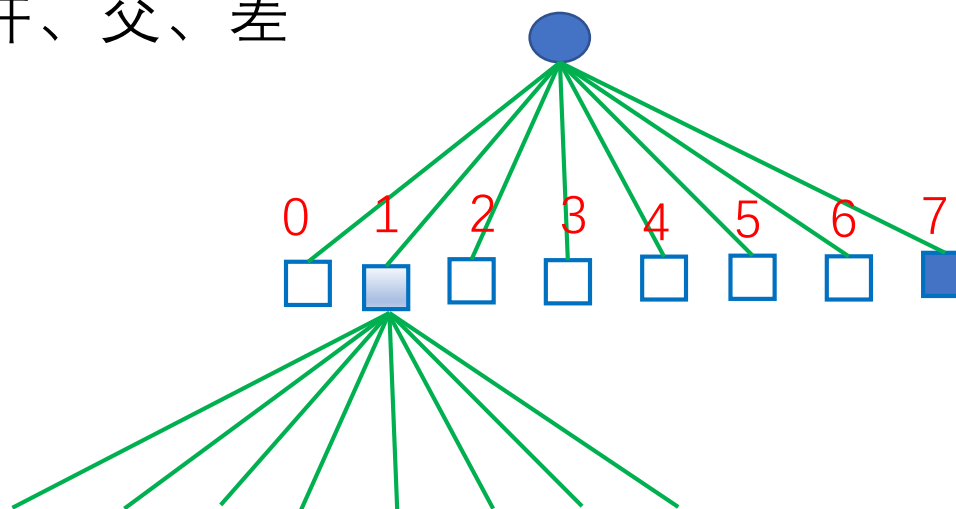
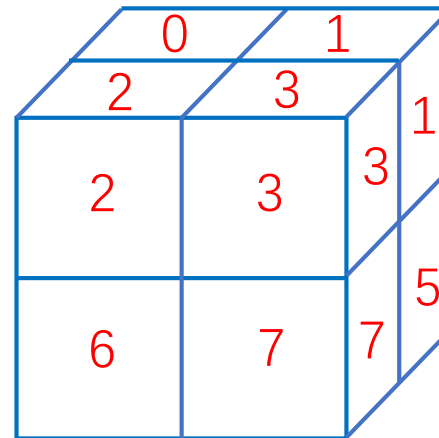
$$p(u, v) = \sum_{i=0}^m \sum_{j=0}^n a_{ij} u^i v^j \quad u, v \in [0, 1]$$

二、几何造型

- 4、八叉树

- 用统一的格式表示任意形状的实体
- 易实现消隐
- 法向矢量简单: $\vec{n}=(0,0,1)$ 等
- 易实现物体之间的集合运算: 并、交、差

- 存储量大
- 几何变换困难
- 输出精度有限



三、面消隐

- 1、概述

- 面消隐：隐藏面消除算法，即相对于观察者，确定场景中哪些点是可见哪些点是不可见的，消隐可以增加图形的真实感。

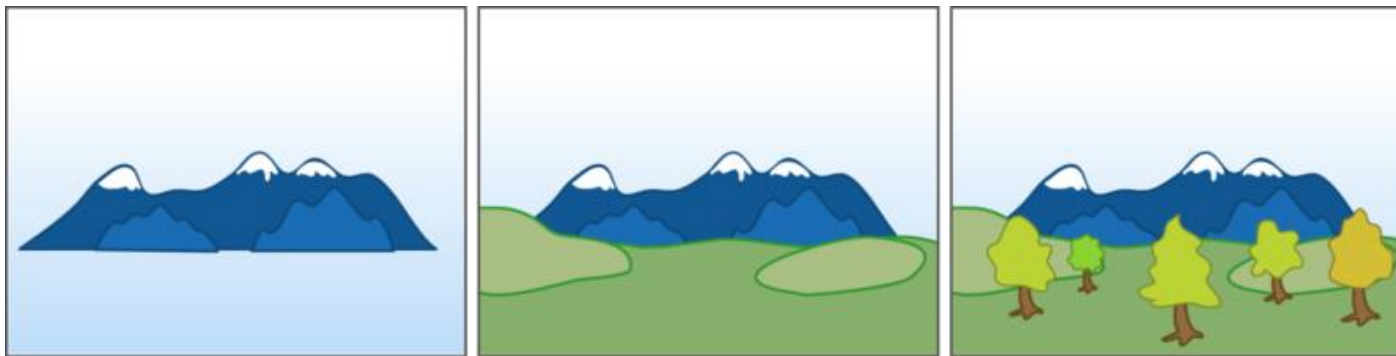
- 分类

- 物体空间的消隐：对 k 个面进行排序，计算复杂度 $O(k^2)$
 - 图像空间的消隐：计算屏幕上每一个点第一次遇到的物体位置，计算复杂度 $O(m \times n \times k)$

三、面消隐

- 2、Z-Sorting算法(物体空间的消隐)

- 离视点近的物体可能遮挡离视点远的物体
- 由远及近地绘制出正确的图像结果——**油画家算法**
- 条件：场景中物体在 z 方向上没有相互重叠



三、面消隐

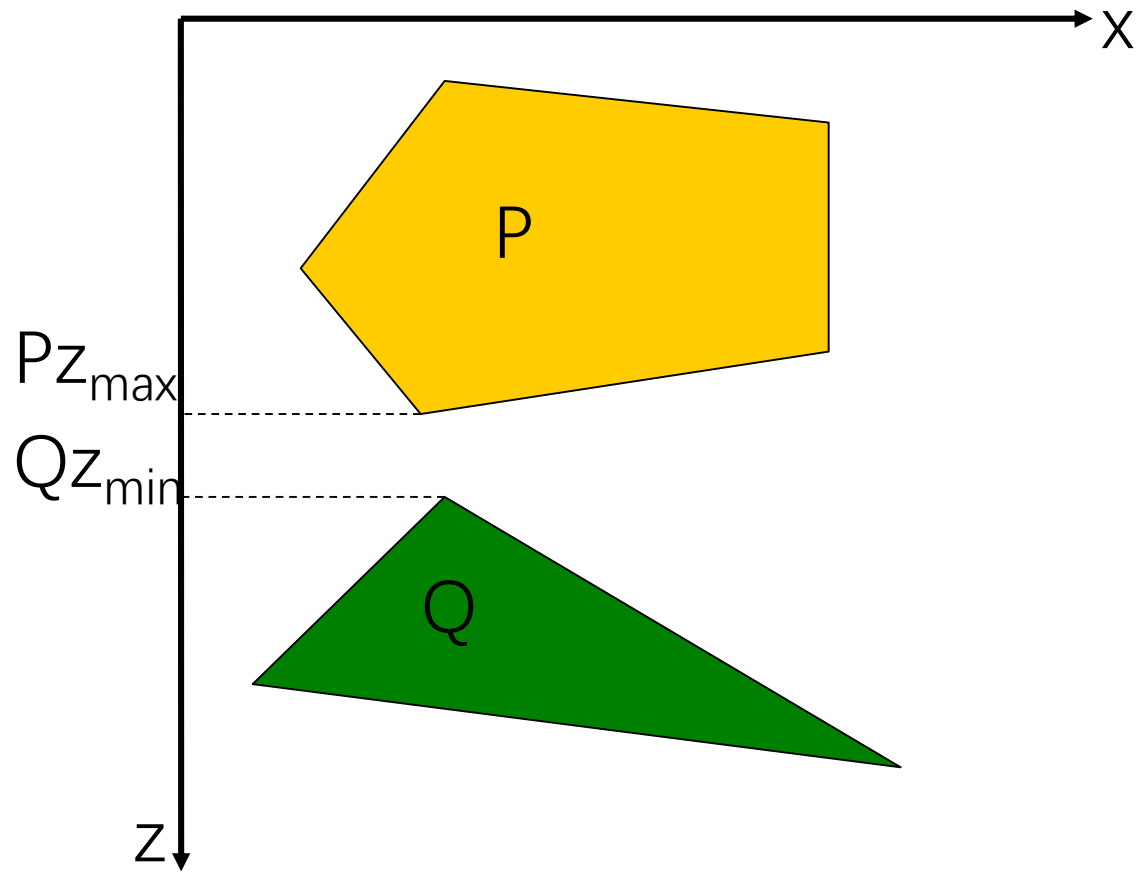
• 2、Z-Sorting算法

• 算法:

step1: 将面片按深度递减排序

step2: 将场景中的多边形序列按其z坐标的最小值 z_{min} (物体上离视点最远的点) 进行排序 ;

step3: 当物体间的z值范围不重叠时, 由远而近直接绘制;



物体间的z值范围不重叠

三、面消隐

- 2、Z-Sorting算法

否则，进行判别：

(1) 多边形P和Q的x坐标范围是否不重叠

(2) 多边形P和Q的y坐标范围是否不重叠

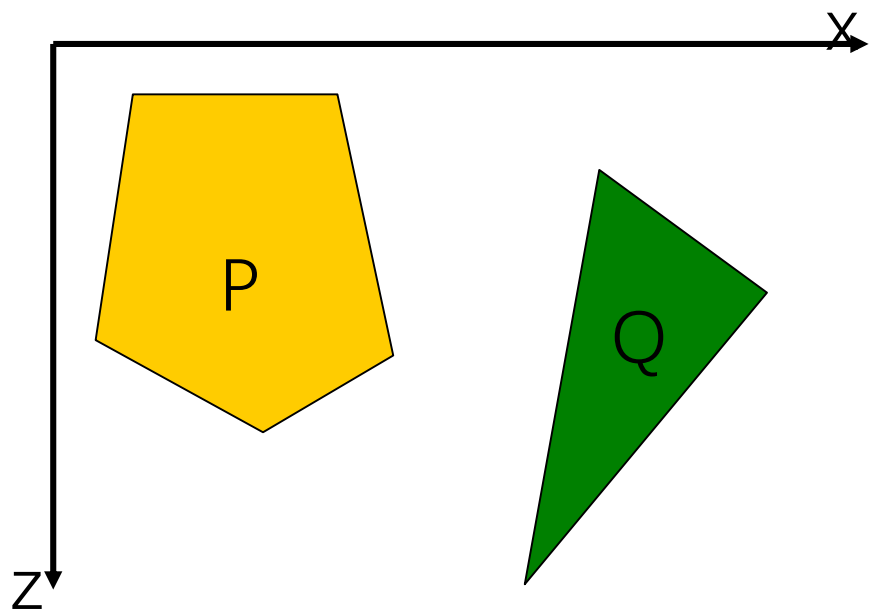
(3) 从视点看去，多边形P是否完全位于Q的背面

(4) 从视点看去，多边形Q是否完全位于P的同一侧

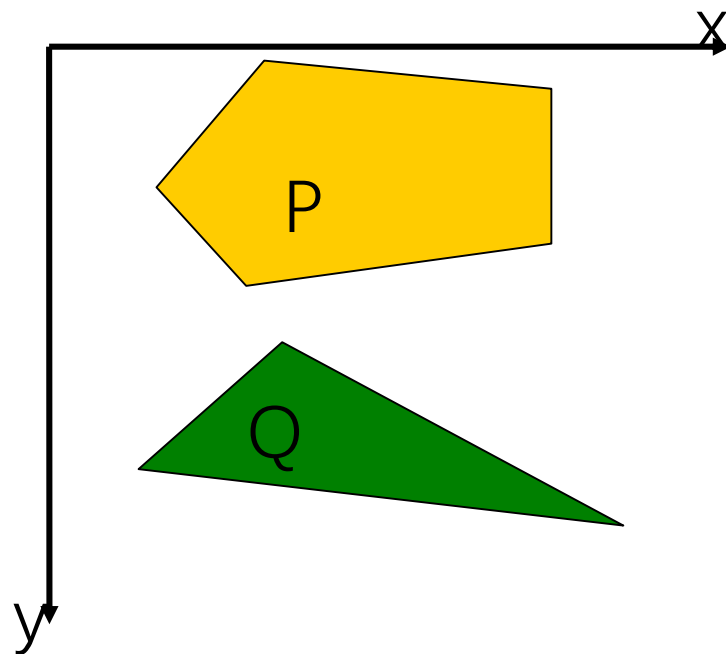
(5) 多边形P和Q在xy平面上的投影是否不重叠

三、面消隐

- 2、Z-Sorting算法



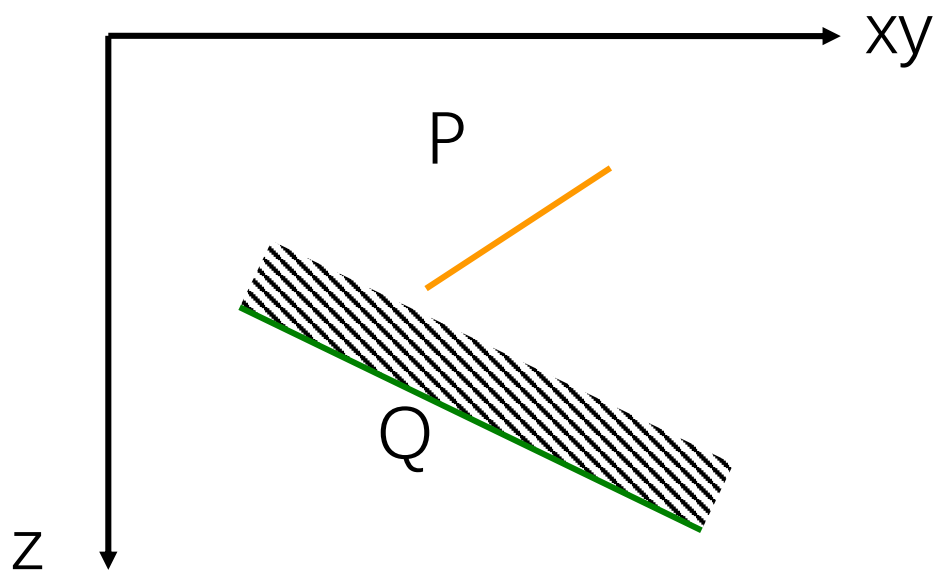
(1) 多边形P和多边形Q的x坐标范围不重叠



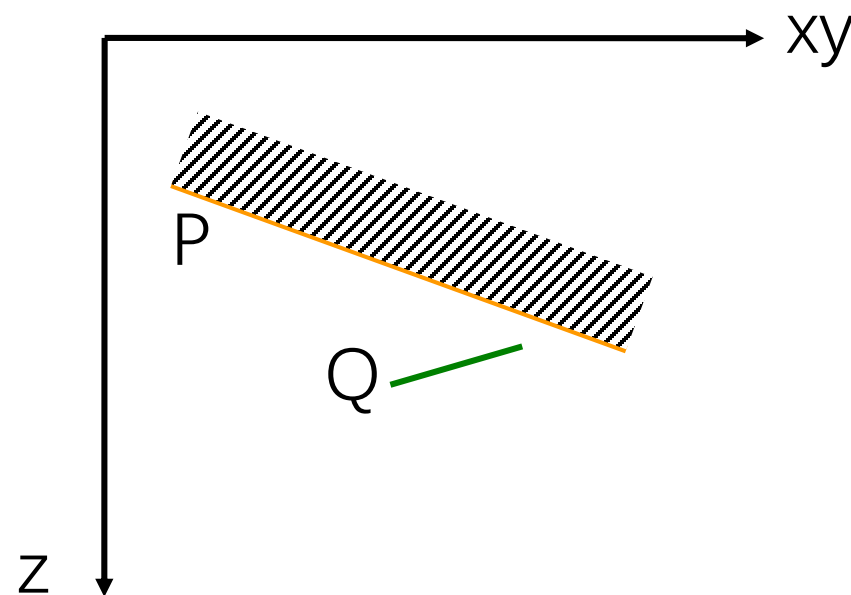
(2) 多边形P和多边形Q的y坐标范围不重叠

三、面消隐

- 2、Z-Sorting算法



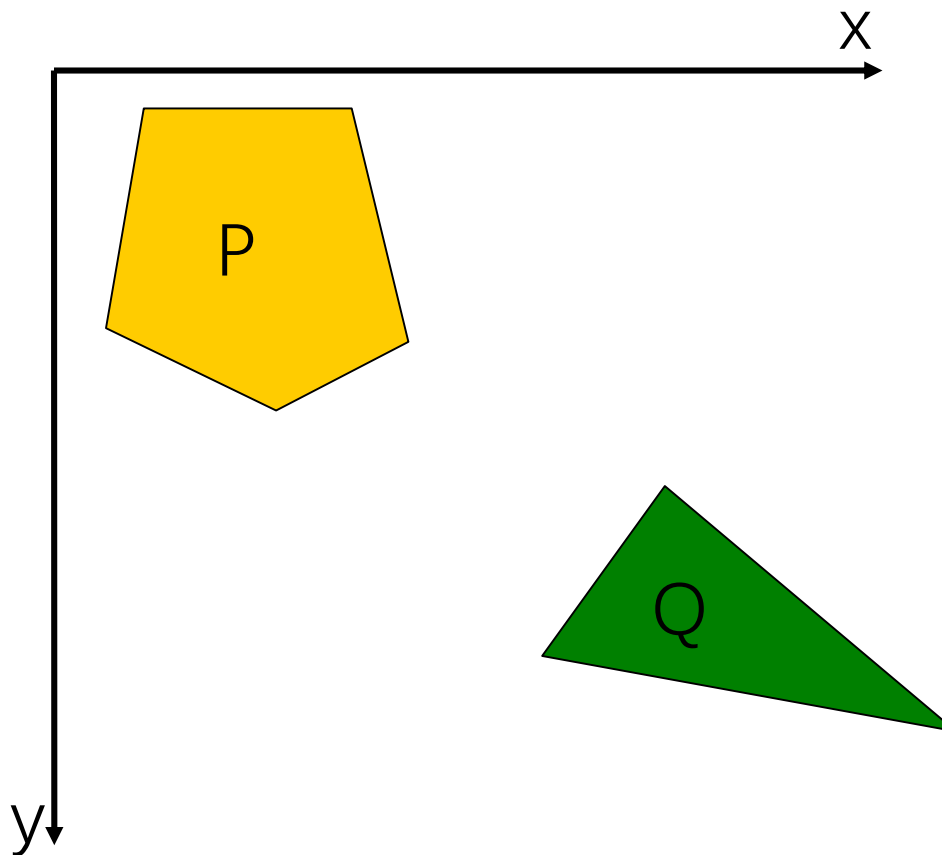
(3) 从视点看去，多边形P完全位于Q的背面



(4) 从视点看去，多边形Q完全位于P的同一侧

三、面消隐

- 2、Z-Sorting算法



(5) 多边形P和Q在xy平面上的投影不重叠

三、面消隐

- 2、Z-Sorting算法（物体空间的消隐）

step4: 如果上述五种情况中只要有一种成立，就表明多边形P和Q是互不遮挡的，即多边形P的绘制优先级低于Q，绘制P；

step5: 如果上述判断都不成立，说明多边形P有可能遮挡Q，此时把多边形P和Q进行互换重新进行判断，而重新判断只要对上述条件[\(3\)](#)和[\(4\)](#)进行即可；

step6: P和Q交换顺序后，仍不能判断其优先级顺序，可以按如下方法处理：将其中一个多边形沿另一个物体剖分。

三、面消隐

- Z-Buffer算法图像空间的消隐

- 思路： 定义两个 $M \times N$ 的数组：深度缓存器ZB，帧缓存器FB。对于任意一点 (i, j) ，设定 $ZB(i, j) = \text{最大值}$ ； $FB(i, j) = \text{背景色}$ ；

对任一多边形上的投影点 (i, j) ，如果颜色为 $P_k(i, j)$ ，求交点 $z_k(i, j)$ ， $k=0, 1, 2, 3 \dots$

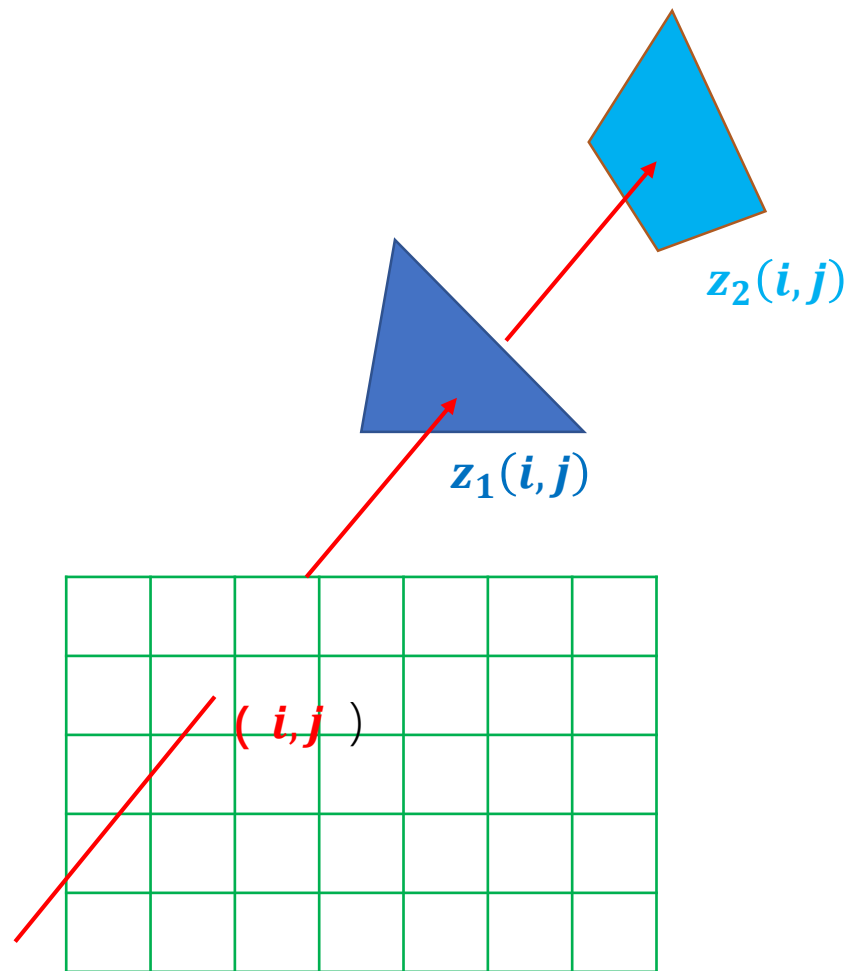
如果 $z_k(i, j) < ZB(i, j)$ ，则： $ZB(i, j) = z_k(i, j)$ ；

$FB(i, j) = P_k(i, j)$

- 特点：反复求交 $z_k(i, j)$

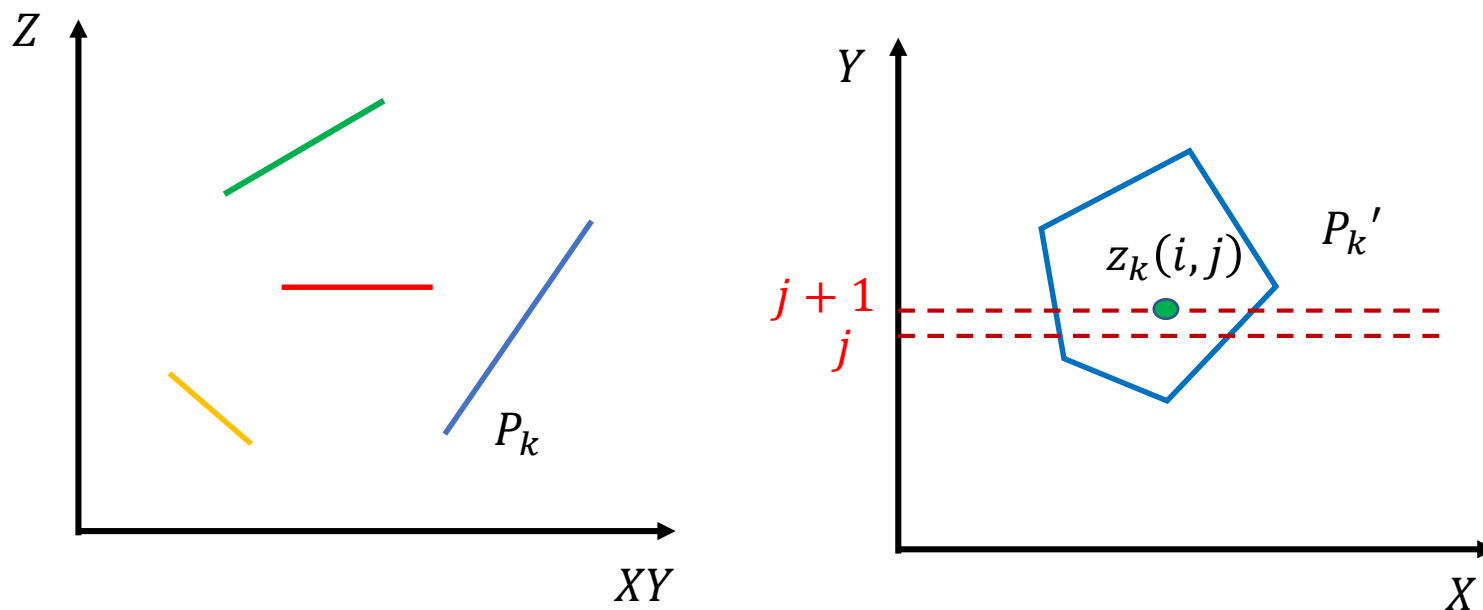
对于第 k 个多边形： $ax + by + cz + d = 0$

交点处的 z 为： $z_k(i, j) = -\frac{d + ai + bj}{c}$



三、面消隐

- Z-Buffer算法
 - scan-line算法



交点 $z_k(i, j)$ 的求解：已知任一多边形 P_k 的平面方程为： $ax + by + cz + d = 0$

P_k 在XY平面的投影为 P_k' ， P_k' 的任一条边为： $px + qy + r = 0$

三、面消隐

- Z-Buffer算法
- scan-line算法

(1) 计算 $y = j, x = i + 1$ 时的 z :

$$Z_{i,j} = -\frac{ai+bj+d}{c}$$

$$Z_{i+1,j} = Z_{i,j} - \frac{a}{c}$$

$$Z_{i+1,j} = Z_{i,j} - \Delta Z_x$$

(2) 计算 $y = j + 1$ 时的第一个 z :

$$\text{当 } y = j \text{ 时, } x_j = -\frac{qj+r}{p}, \quad Z_j = -\frac{ax_j+bj+d}{c}$$

$$\text{当 } y = j+1 \text{ 时, } x_{j+1} = x_j - \Delta x, \quad \text{其中 } \Delta x = \frac{q}{p}$$

$$Z_{j+1} = -\frac{ax_{j+1}+b(j+1)+d}{c} = Z_j + \frac{a}{c} \cdot \frac{q}{p} - \frac{b}{c}$$

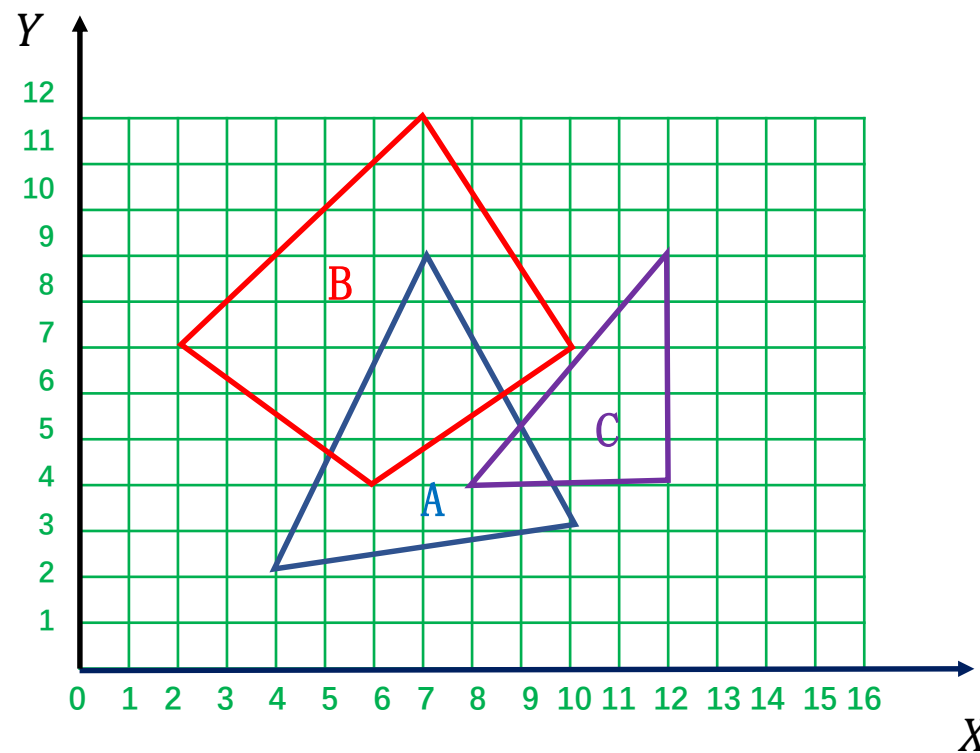
$$\text{令 } \frac{a}{c} = \Delta Z_x, \quad \frac{b}{c} = \Delta Z_y, \quad \text{则: } Z_{j+1} = Z_j + \Delta Z_x \cdot \Delta x - \Delta Z_y$$

三、面消隐

- Z-Buffer算法
- scan-line算法

(3) 建立数据结构

- A、多边形表Polygon-Table (PT)
- B、有效多边形表 (APT)
- C、边表 (ET)
- D、有效边表 (AET)



A: (4, 2)、(7, 9)、
(10, 3)

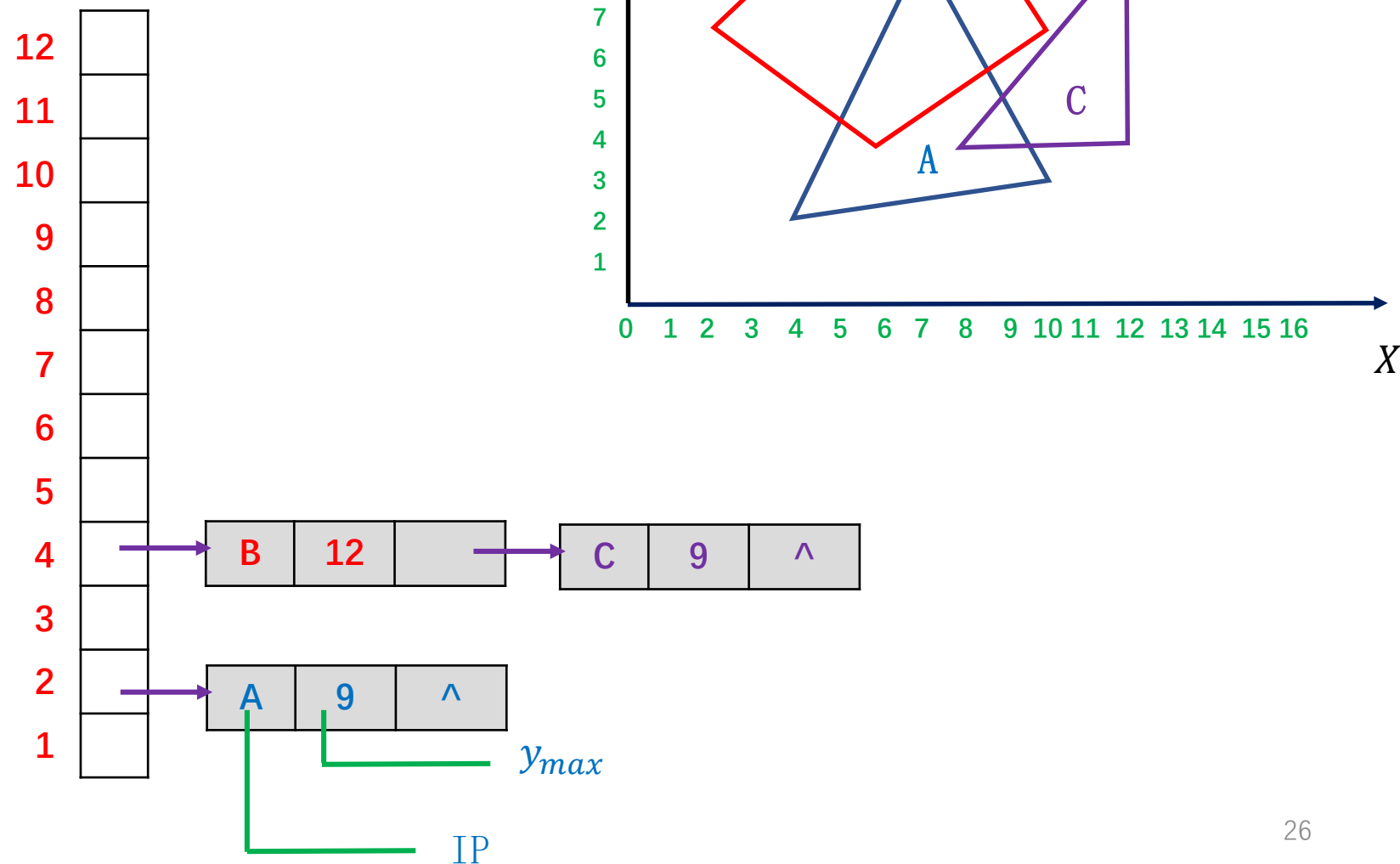
B: (6, 4)、(2, 7)、(7, 12)、(10, 7)

C: (8, 4)、(12, 9)、
(12, 4)

三、面消隐

- Z-Buffer算法
 - scan-line算法

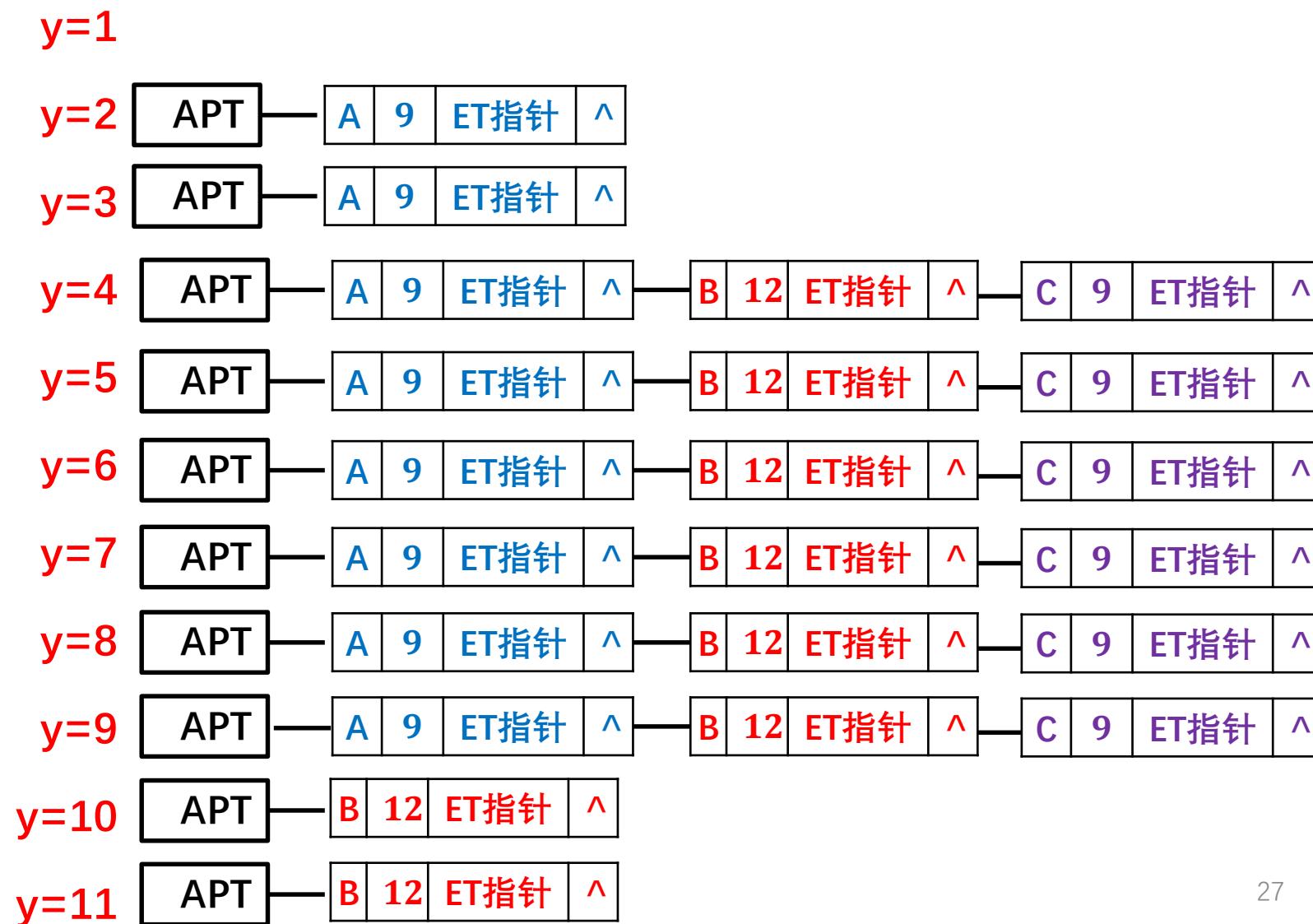
A、建立PT



三、面消隐

- Z-Buffer算法
- scan-line算法

B、建立APT

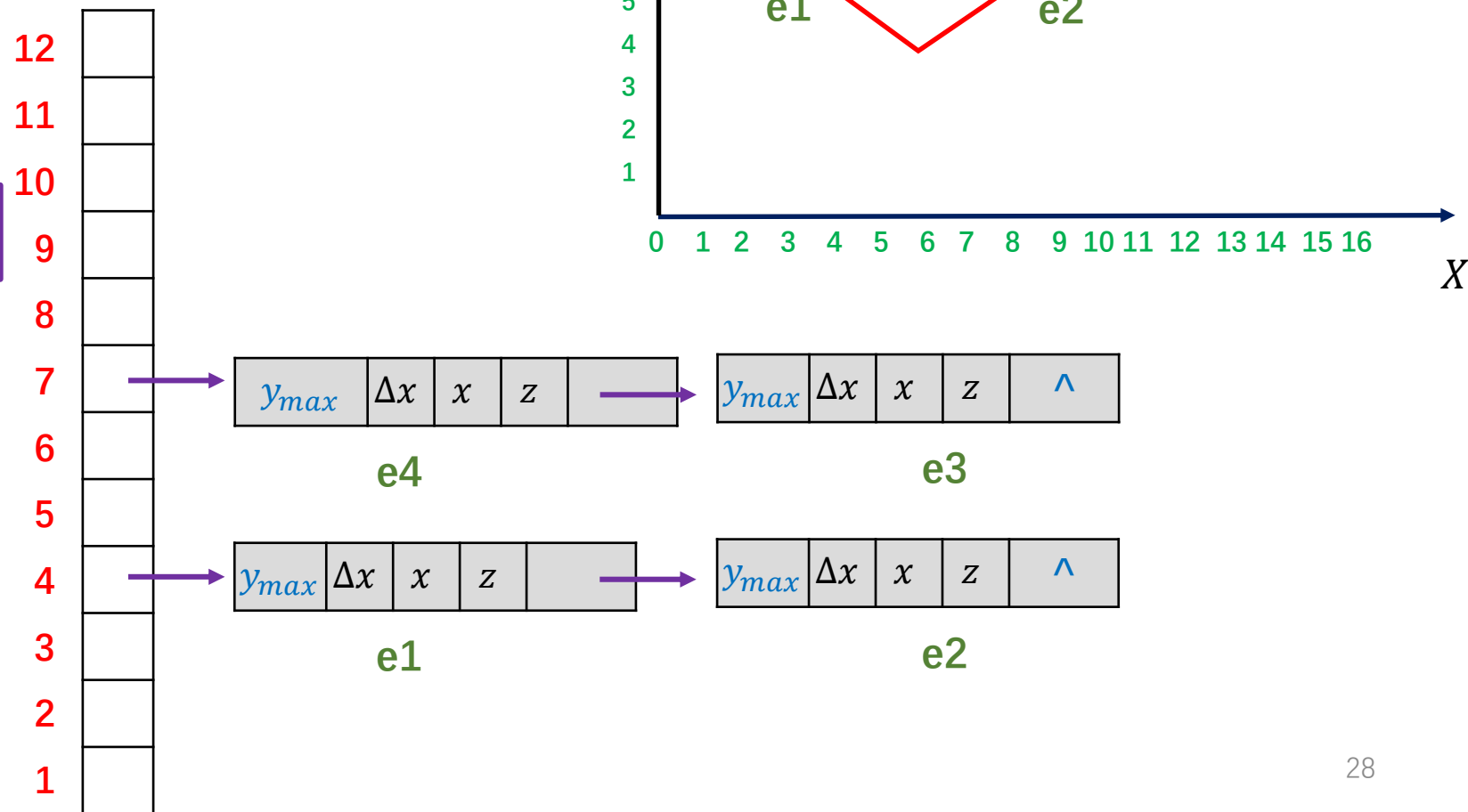


三、面消隐

- Z-Buffer算法

- scan-line算法

C、建立ET(以B)为例



三、面消隐

- Z-Buffer算法
 - scan-line算法

D、建立AET(以B)为

例

AET	—	x	Δx	y_{max}	z	Δz_x	Δz_y	IP	\wedge
-----	---	-----	------------	-----------	-----	--------------	--------------	----	----------

$y = j + 1$ 时更新: $x = x - \Delta x$

$$z = z + \Delta z_x \cdot \Delta x - \Delta z_y$$

(4) 对相邻两个边表中 x 之间的点, 更新深度缓存表(ZB)、帧缓存表(FB)

$$z_{i,j} = z_{i,j} - \Delta z_x$$

如果 $z_{i,j} < ZB(i,j)$,

则: $ZB(i,j) = z_{i,j}$;

$$FB(i,j) = P_k(i,j)$$

四、光照模型

- 真实感图形的显示就是模拟物体在一定照明条件下的光照效果，而光照模型就是计算给定点光强的公式。
- 光照明模型考虑物体表面上每一个点所代表的微小面元受到来自光源或周围环境光线的照射而产生的反射或透射光亮度
- 光照效果包括反射、透明、纹理、阴影等；
- 光源有：点光源、线光源、面光源、白色、彩色等；
- 数学模型越精确，光照效果越好，计算量越大。

四、光照模型

- 1、泛光

- 不直接由光源发光，没有方向特性，于视线无关，通常作为背景光
- 每个物体表面都得到同样大小的光照，不同的表面 k_a 不同，如夜晚、阴影下的物体。

$$I_{env} = K_a I_a$$

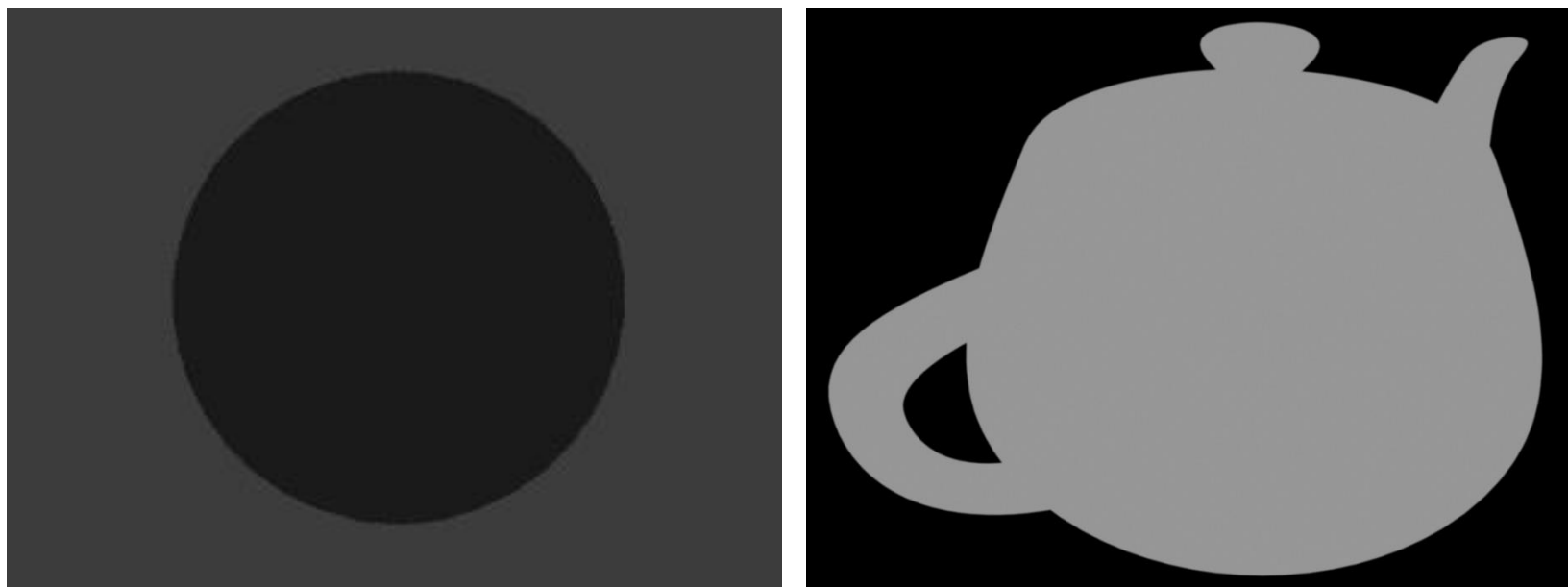
I_{env} : 物体表面对泛光的反射光亮度

I_a : 泛射光的入射光亮度

K_a : 物体表面对泛光的反射率

四、光照模型

- 1、泛光



泛光模型的光照明效果

四、光照模型

• 2、漫反射

- 每个面上的漫反射是恒定不变的，与观察方向无关。光线被物体漫反射后向各个方向以同等光强度发散。如树木、地面
- 大小只与入射光和物体的法向矢量有关
- 与漫反射光的反射方向无关

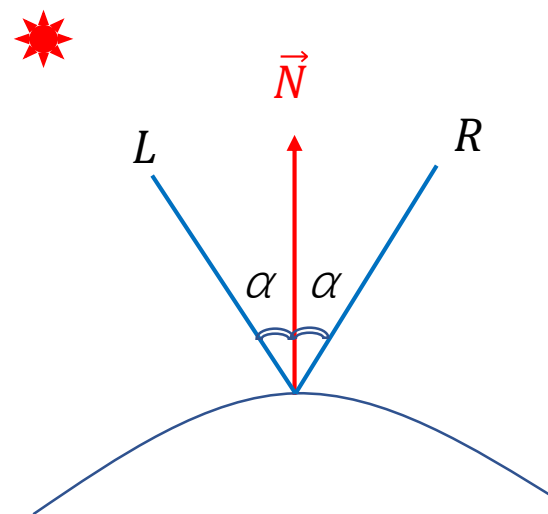


四、光照模型

- 2、漫反射 (diffuse reflection)

$$I_d = K_d I_e \cos \alpha$$

I_d : 表面漫反射光的光亮度
 K_d : 物体表面漫反射率
 I_e : 发自光源的入射光的光亮度
 α : 光源入射角



四、光照模型

- 2、漫反射

- Lambert光照模型

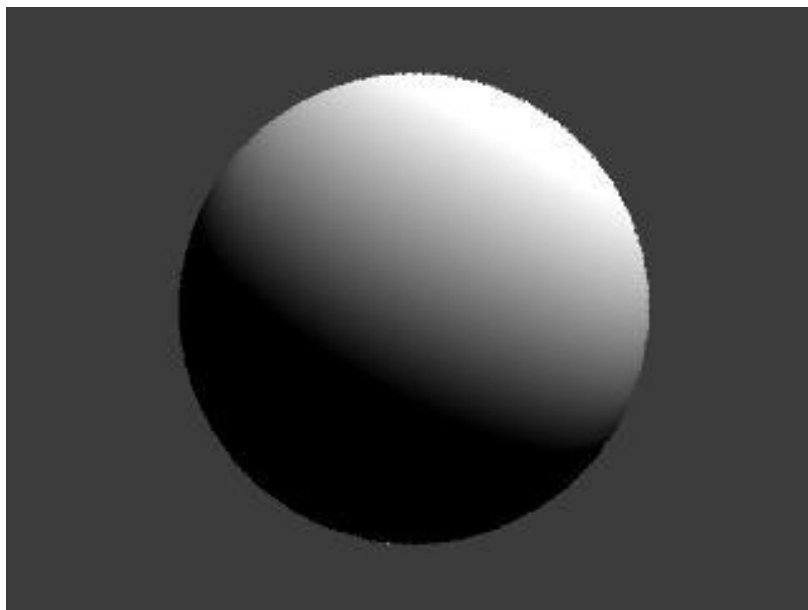
$$I = K_a I_a + K_d I_e \cos \alpha$$

I : 景物表面的反射光亮度

I_a : 环境泛光入射光亮度, 一般取值范围为
 $0.02I_e \sim 0.2I_e$

四、光照模型

- 2、漫反射



Lambert模型的光照明效果

四、光照模型

• 3、镜面反射

- 与视线有关, θ 越小高光越亮
- 与材料有关, k_s 越大, 高光越亮

$$I_s = k_s I_e \cos^n \theta$$

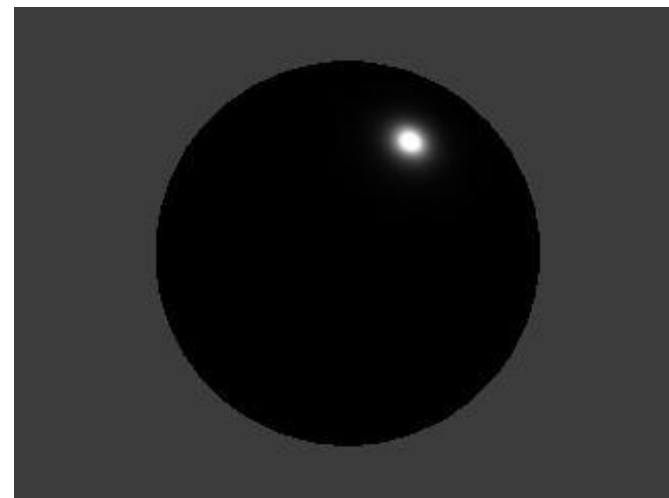
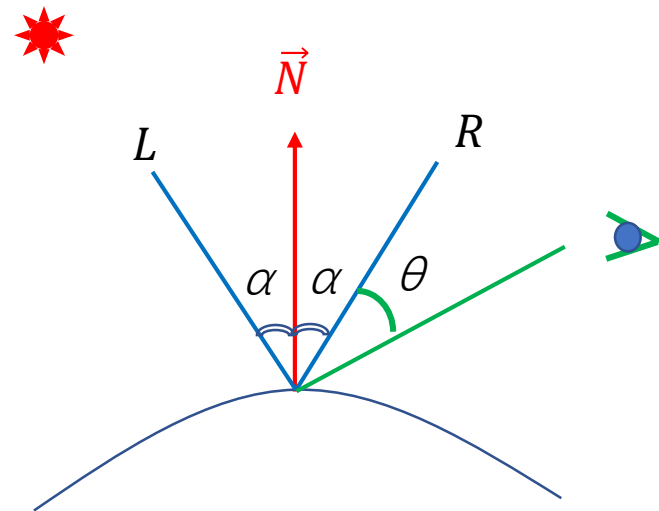
I_s : 物体表面镜面反射光亮度

I_e : 发自光源的入射光的光亮度

θ : 光线反射方向和视线方向的夹角

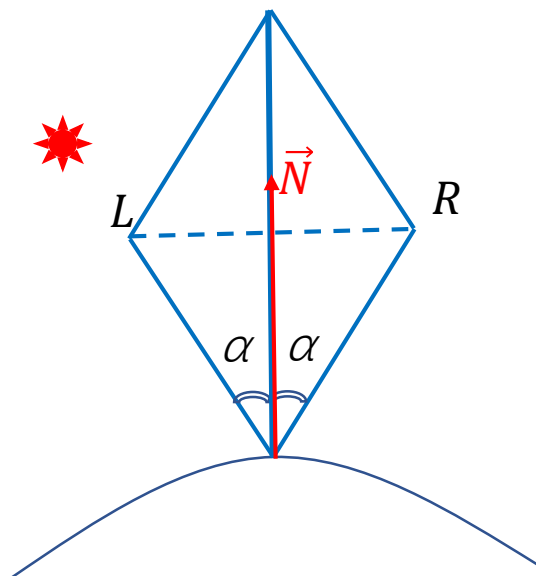
k_s : 表面的镜面反射率

n : 镜面反射光的会聚指数, 又称 “高光” 指数



四、光照模型

- 3、镜面反射
 - 反射矢量的求解



$$R + L = 2N\cos\alpha$$

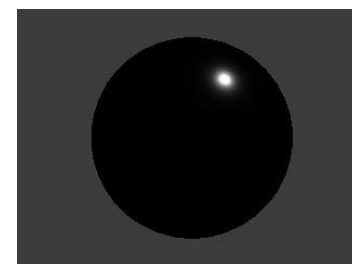
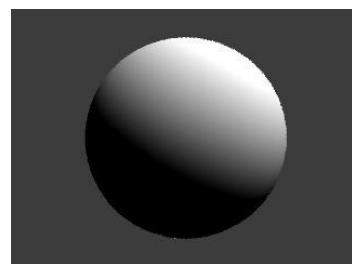
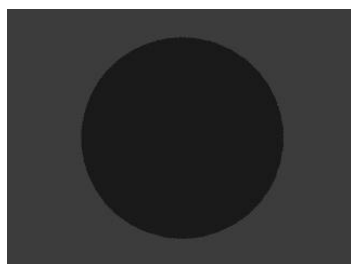
$$R = 2N\cos\alpha - L$$

四、光照模型

- 3、镜面反射

- Phong模型：综合了漫反射、镜面反射及泛光反射分量

$$I = K_a I_a + K_d I_e \cos \alpha + K_s I_e \cos^n \theta$$



四、光照模型

- 3、镜面反射

- Phong模型：多光源

$$I = K_a I_a + \sum_{i=1}^m I_i (K_d \cos \alpha + K_s \cos^n \theta)$$



Phong模型的光照明效果

四、光照模型

- 4、透射

- 透射光线

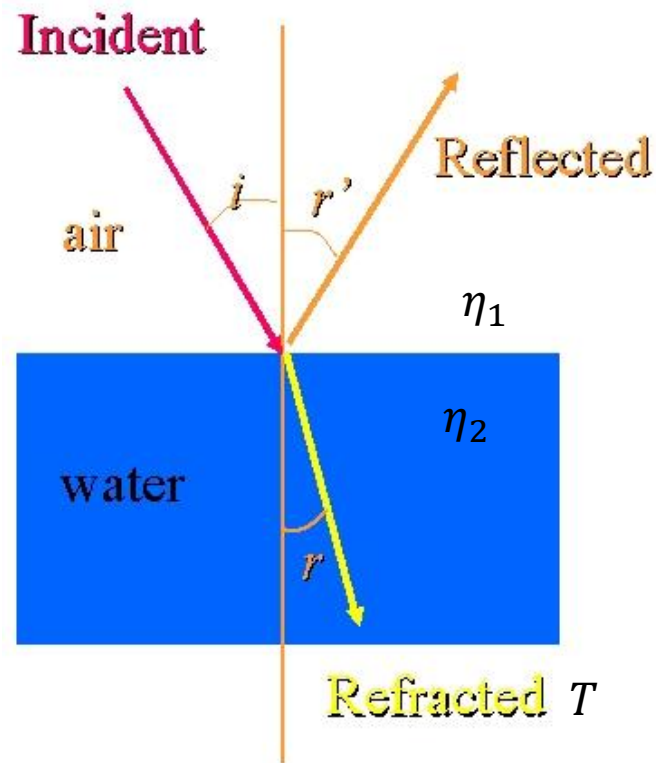
$$\frac{\sin r'}{\sin r} = \frac{\eta_2}{\eta_1}$$

$$T = \left(\frac{\eta_2}{\eta_1} \sin r - \cos r' \right) \cdot N - \frac{\eta_2}{\eta_1} L$$

- 透明模型

$$I = (1 - k_t) I_{ref} + k_t I_t$$

k_t : 透明系数, $k_t \in [0,1]$



四、光照模型

- 4、透射
 - Whitted模型整体光照明模型

$$I = I_c + k_s I_s + k_t I_t$$

I_c : 由光源直接照射在表面上引起的反射光亮度

I_s : 沿 v 的镜面反射方向 r 入射到表面上的环境光在表面上产生的镜面反射光

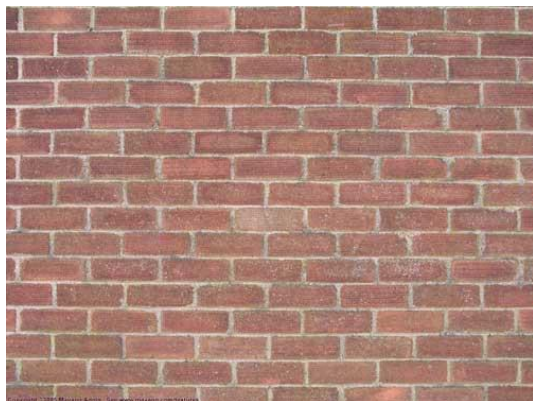
I_t : 沿 v 的规则透射方向 t 入射到表面上的环境光通过透射在表面上产生的规则透射光

k_s : 表面的镜面反射率

k_t : 表面的透射率

四、光照模型

- 5、纹理映射
 - 纹理（texture）通常指物体的表面细节

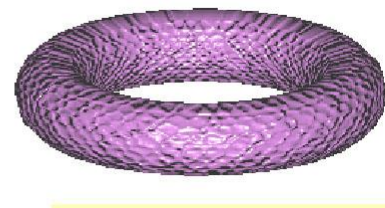
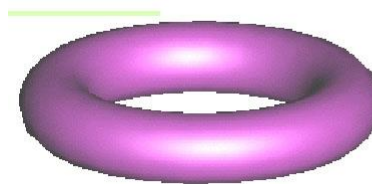
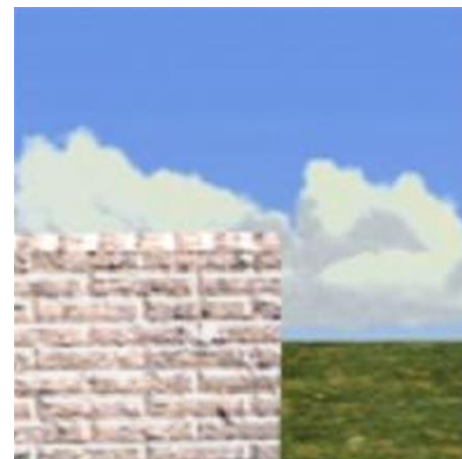
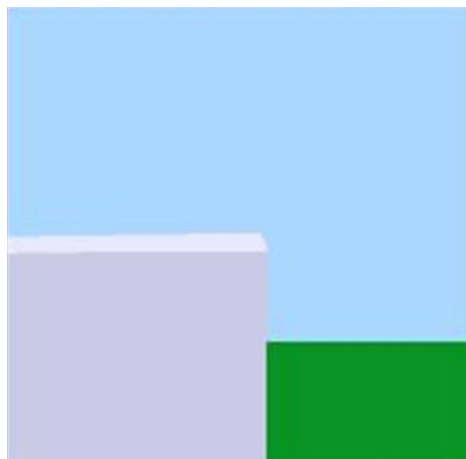


四、光照模型

- 5、纹理映射
 - 两类最常使用的纹理

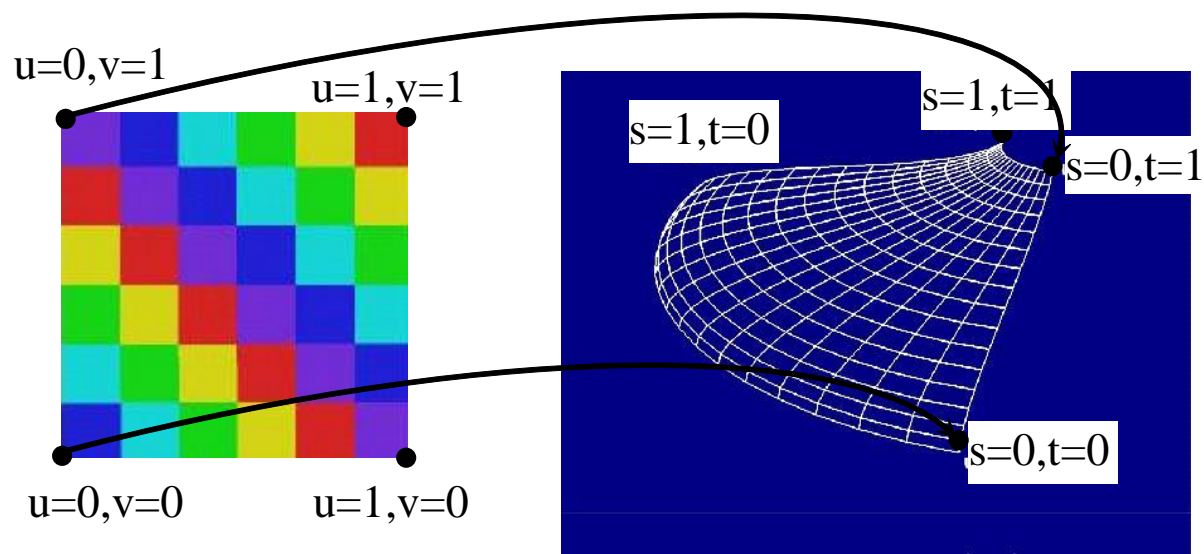
颜色纹理

几何纹理



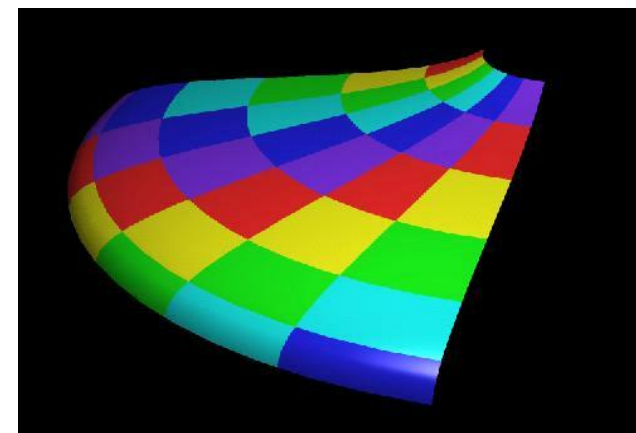
四、光照模型

- 5、纹理映射
 - 颜色纹理



(a) 纹理空间

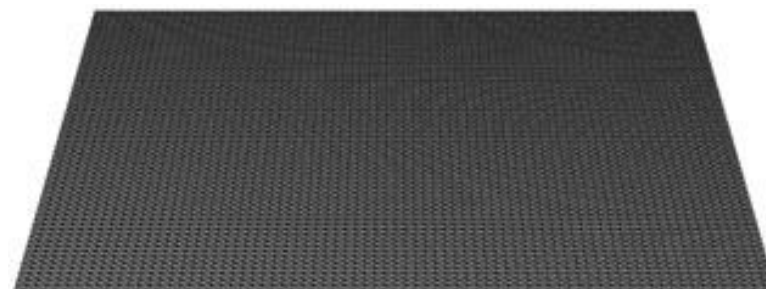
(b) 景物空间的曲面片



(c) 纹理映射后的曲面片

四、光照模型

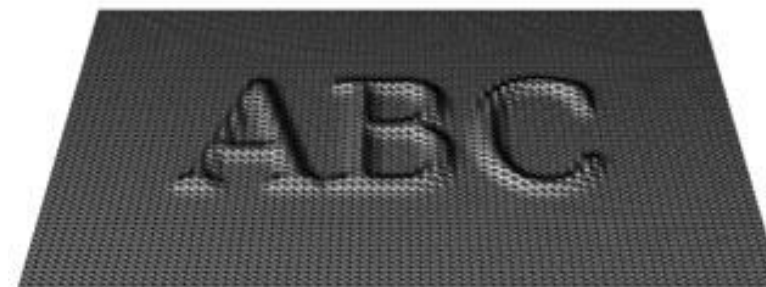
- 5、纹理映射
 - 几何纹理



ORIGINAL MESH



DISPLACEMENT MAP



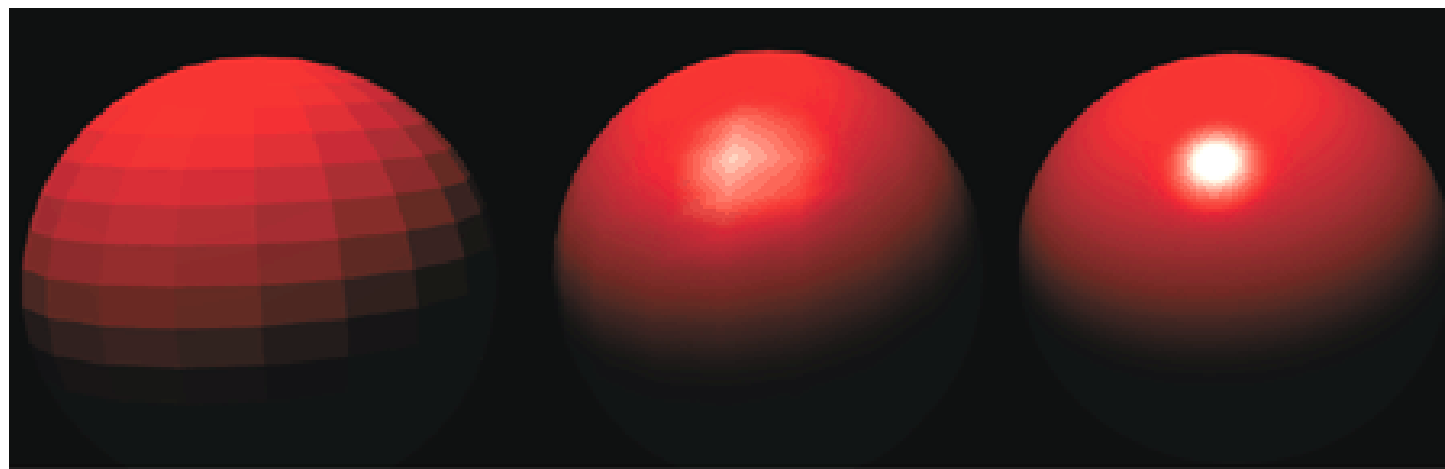
MESH WITH DISPLACEMENT

五、画面绘制

- 1、多边形物体的明暗处理

Flat Shading、Gouraud Shading、Phong Shading三种方式

From Computer Desktop Encyclopedia
Reproduced with permission.
© 2001 Intergraph Computer Systems



Flat

Gouraud

Phong

五、画面绘制

- 1、多边形物体的明暗处理
 - Flat Shading

方法：

- 依据局部光照明模型按每一个多边形的法向计算出一个颜色值C
- 将C赋给该多边形在屏幕上的投影所覆盖的全体像素

优缺点：

- 处理简单，计算量小
- 景物表面上相邻的多边形之间颜色差异较大，存在马赫带效应

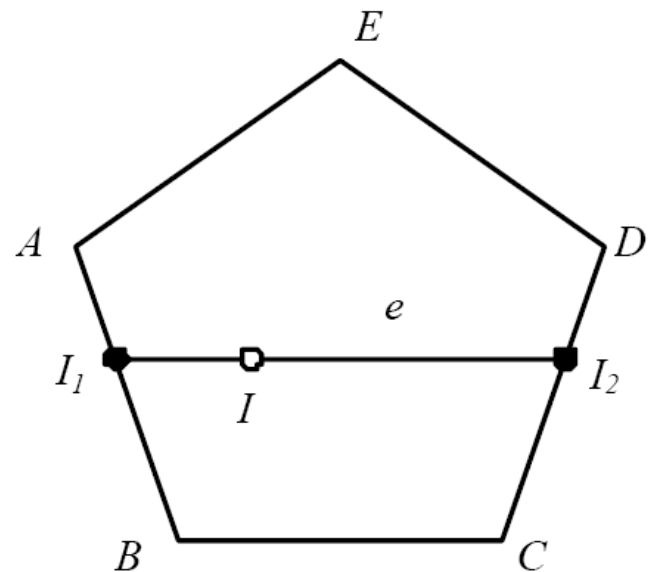


五、画面绘制

- 1、多边形物体的明暗处理
 - Gouraud Shading

方法

- 为多边形物体的每一个顶点赋一个法向矢量
- 利用局部照明模型计算每一顶点处的光亮度
- 多边形内部各点处的光亮度值通过对多边形顶点处的光亮度的双线性插值得到



$$I_1 = \left| \frac{y_1 - y_B}{y_A - y_B} \right| I_A + \left| \frac{y_A - y_1}{y_A - y_B} \right| I_B$$

$$I_2 = \left| \frac{y_2 - y_C}{y_D - y_C} \right| I_D + \left| \frac{y_D - y_2}{y_D - y_C} \right| I_C$$

$$I = \left| \frac{x - x_1}{x_2 - x_1} \right| I_2 + \left| \frac{x_2 - x}{x_2 - x_1} \right| I_1$$

五、画面绘制

- 1、多边形物体的明暗处理
 - Gouraud Shading

优缺点

- 简单快速，所生成的图形在真实感上较Flat Shading有了较大的提高
- 马赫带效应依然存在
- 不能正确模拟高光



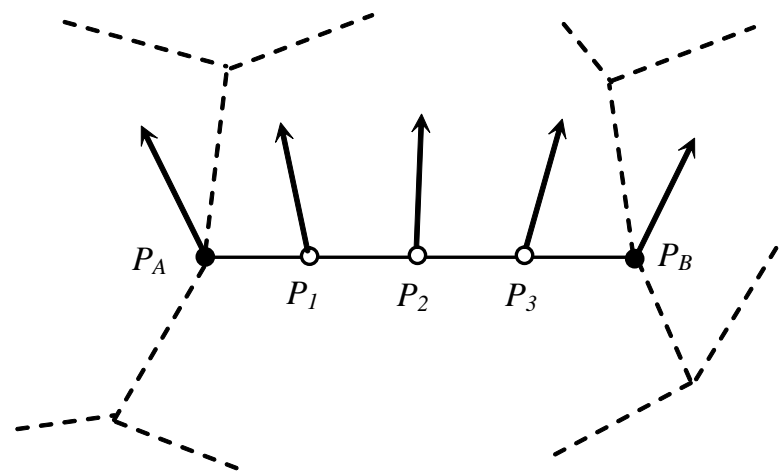
五、画面绘制

• 1、多边形物体的明暗处理

- Phong Shading

方法:

- 为多边形物体的每一个顶点赋一个法向矢量
- 多边形内部各点处的法向矢量则通过对多边形顶点处法向矢量的双线性插值得到
- 利用Phong模型计算每一点的光亮度



$$P_i = \left| \frac{x_i - x_A}{x_B - x_A} \right| P_B + \left| \frac{x_B - x_i}{x_B - x_A} \right| P_A$$

五、画面绘制

- 1、多边形物体的明暗处理
 - Phong Shading

优缺点：

- 能较好地模拟高光
- 相邻多边形之间的光亮度过渡更自然
- 计算量比Gouraud Shading要大得多

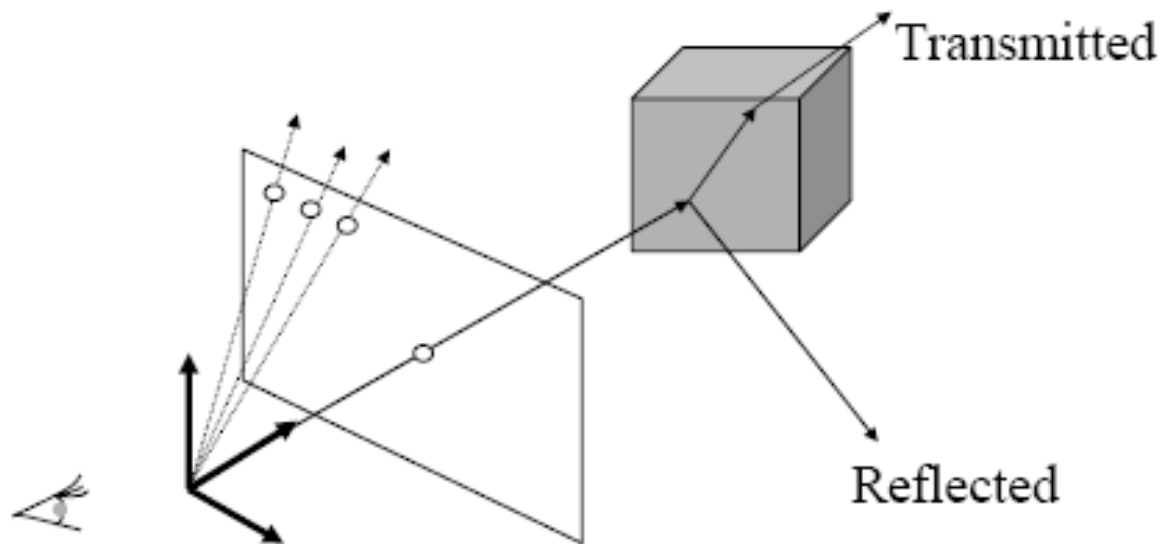


五、画面绘制

• 2、光线跟踪

迄今为止最为成功的生成真实感图形算法之一

- 算法简单
- 生成的图形真实感强
- 计算量大



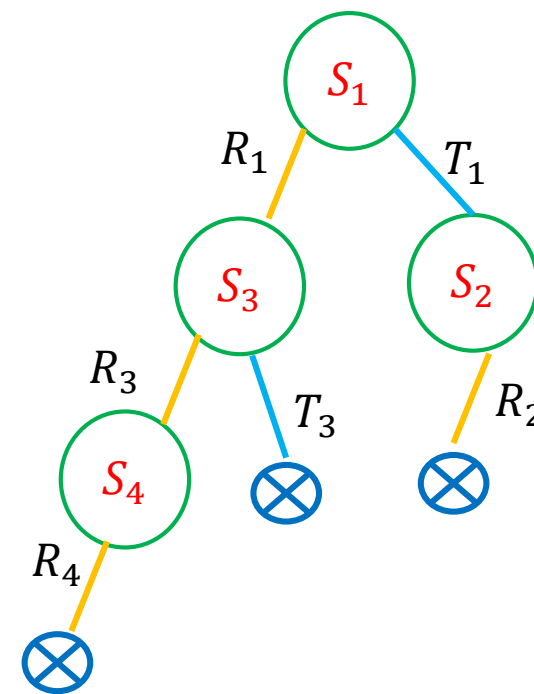
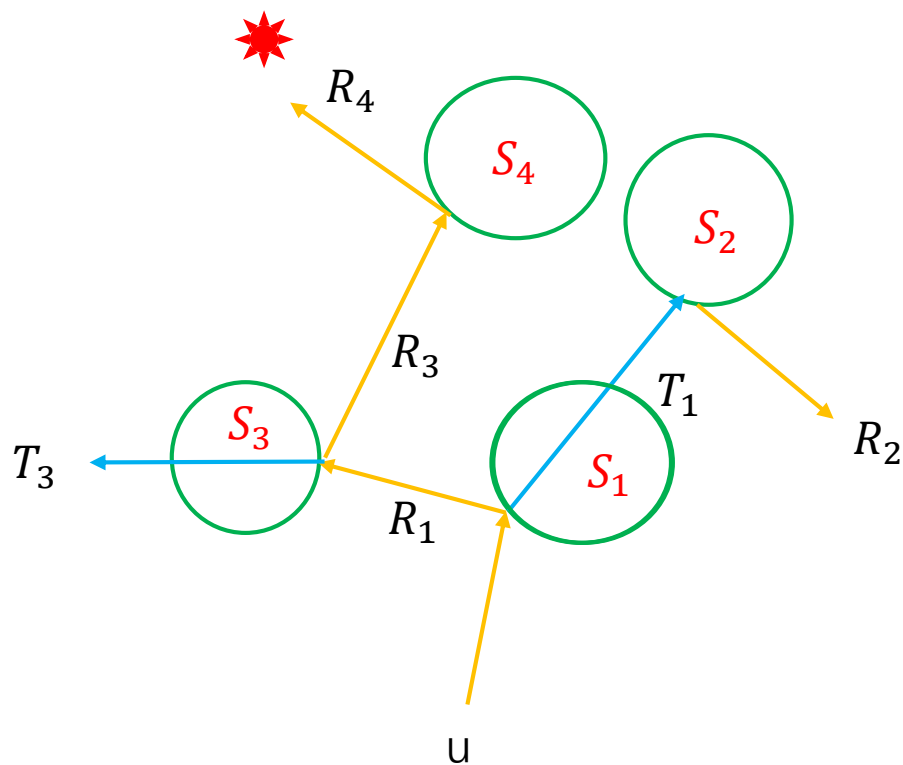
五、画面绘制

• 2、光线跟踪

- 对屏幕上每一像素，执行下述4个步骤：
 - 单位光线 u 与物体求交，距离最近的交点为可见点；
 - 计算可见点的反射、折射光线，生成第二光线；
 - 第二光线与物体求交，获得最近的交点，回到上一步；
 - 终止条件：光线到达光源或最大深度；
 - 建立二叉树；
 - 计算光强：从二叉树叶子点向上累加光强。
- 当所有屏幕像素都处理完毕时，即得到一幅真实感图形

五、画面绘制

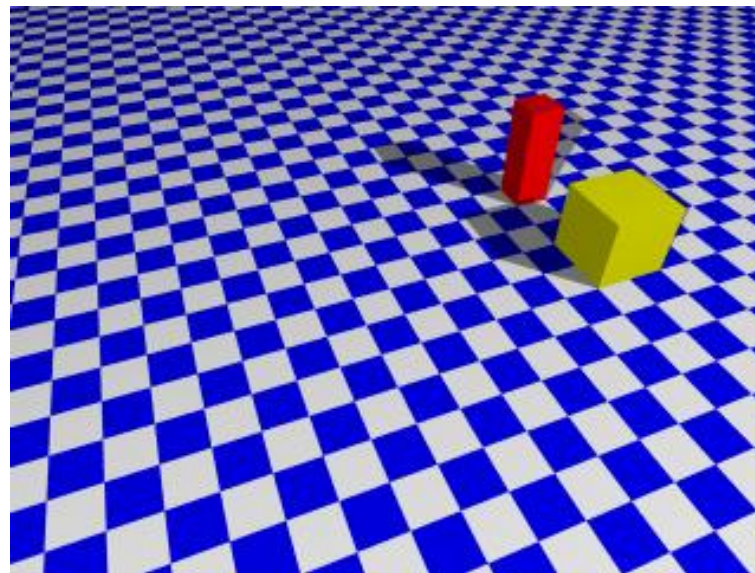
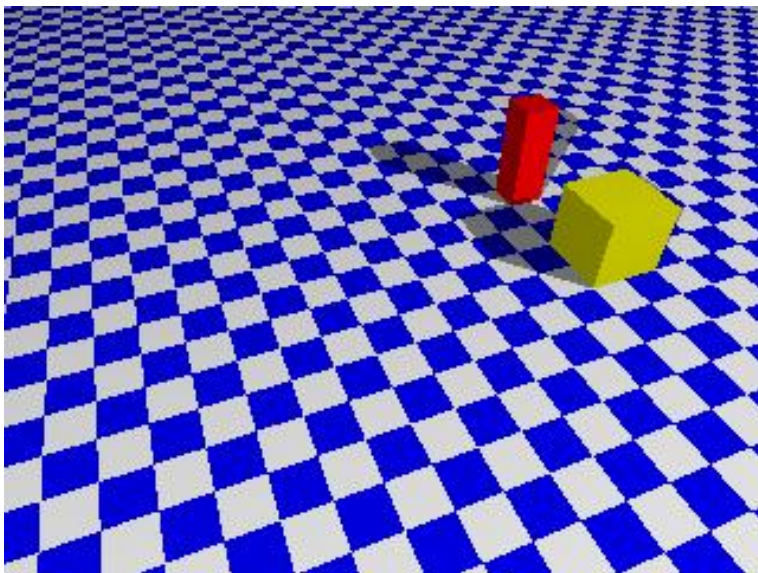
• 2、光线跟踪



- 关键问题：
 - 光线与物体的求交
 - 反射方向计算
 - 透射方向计算

五、画面绘制

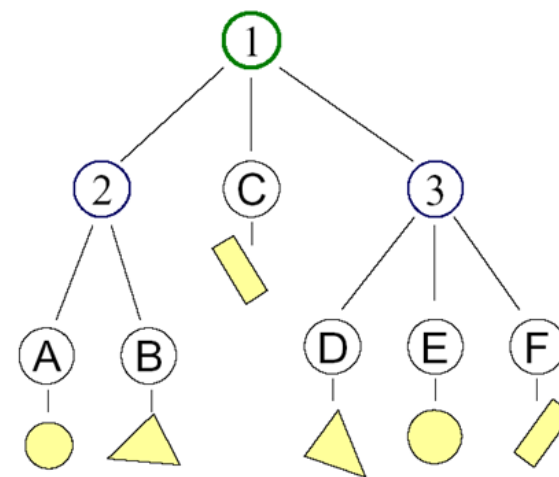
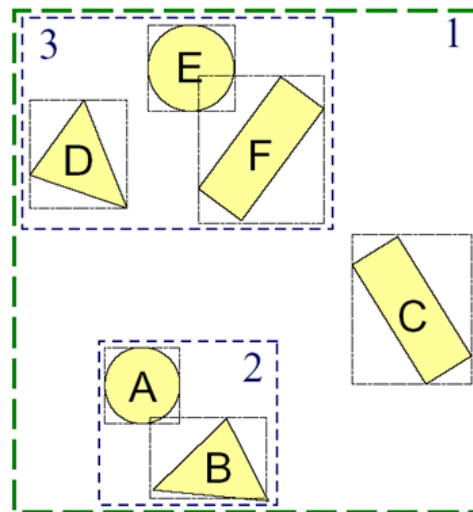
- 2、光线跟踪
 - 反走样：光线跟踪算法本质上是对画面的点采样



五、画面绘制

- 2、光线跟踪
 - 加速技术

包围盒技术

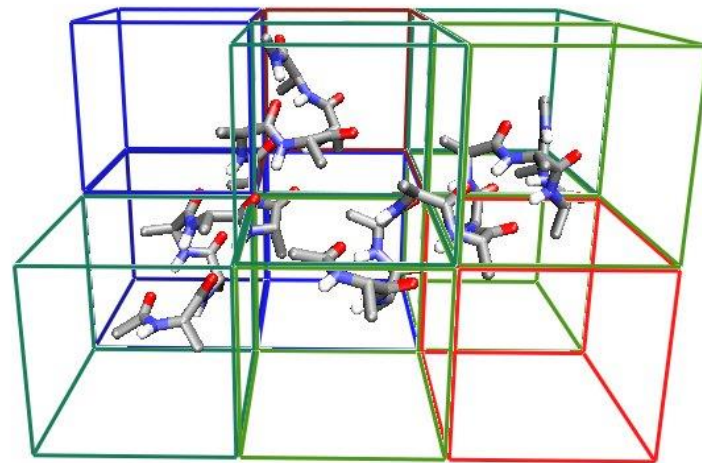


五、画面绘制

- 2、光线跟踪
 - 加速技术

空间分割技术

- 将景物空间分割成一个个小的空间单元
- 被跟踪的光线仅与它所穿过空间单元中所含物体表面进行求交测试
- 利用相邻空间单元的空间连贯性，使光线快速跨越空单元，迅速到达非空单元，求得光线与景物的第一个交点



五、画面绘制

- 2、光线跟踪

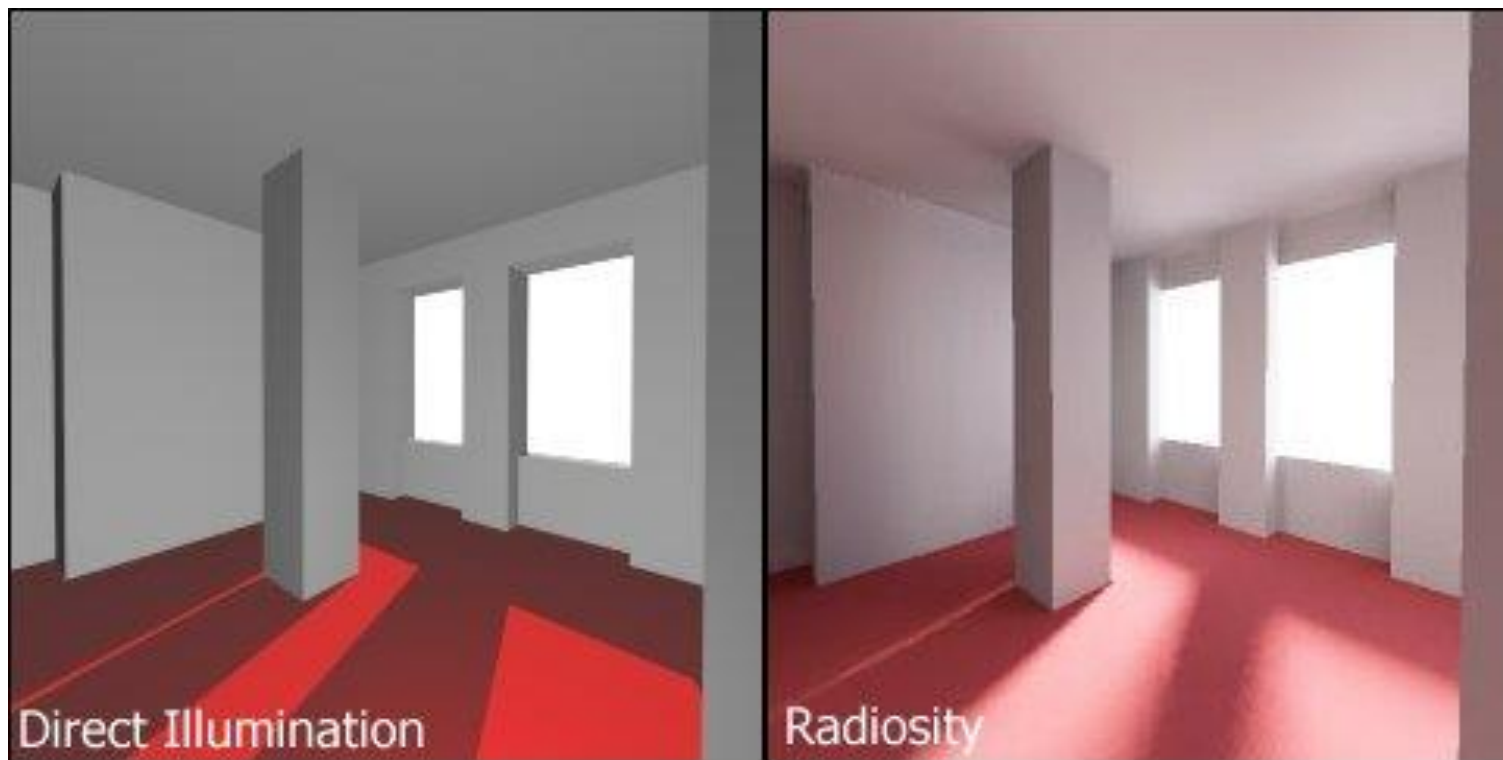
- 特点

- 绘制与消隐同时实现
 - 模拟全局的漫反射和折射现象
 - 计算量巨大：求交、逐点计算
 - 与视线有关，换个方向，重新计算



五、画面绘制

- 3、辐射度方法
 - 直接照明与辐射度效果对比



五、画面绘制

- 3、辐射度方法

- 原理

- 光是一种辐射能，在一个封闭环境中，场景中的光能经过表面之间的反射和透射，最终达到平衡状态
 - 场景中各表面的光亮度实际上是场景中光能分布的反映

- 前提：针对理想漫射环境

五、画面绘制

- 3、辐射度方法

- 对于每一小面片 A_i ，成立：

$$B_i = E_i + \rho_i H_i$$

辐射度

自身拥有的
辐射度

漫反射率

周围场景面片入射到 A_i 单位面积上的光能，是周围面片辐射度的函数

五、画面绘制

• 3、辐射度方法

- 设面片 A_j 为一周围场景面片，由辐射度定义， A_j 向外辐射的总的光能为 $B_j A_j$ ，其中一部分到达 A_i
- 设到达 A_i 的光能占 A_j 向外辐射的总光能的比例为 F_{ji} （称为 A_j 到 A_i 的形状因子），则从周围环境面片入射到 A_i 的总光能为：

$$\sum_{\substack{j=1 \\ j \neq i}}^n F_{ji} B_j A_j$$

五、画面绘制

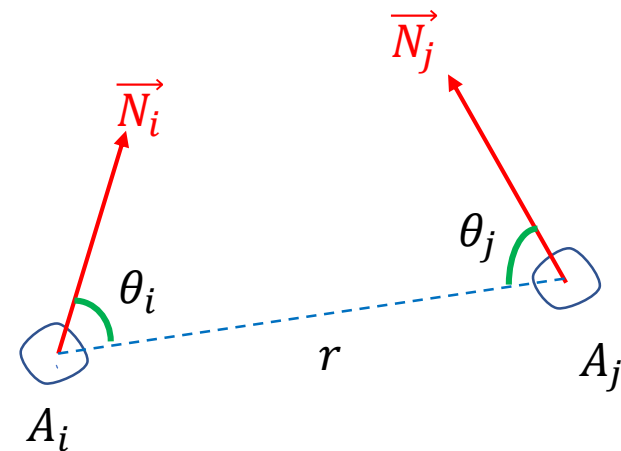
- 3、辐射度方法

- 入射到面片 A_i 单位面积上的光能 H_i 则为：

$$H_i = \sum_{\substack{j=1 \\ j \neq i}}^n F_{ji} B_j A_j / A_i$$

其中：


$$F_{ji} = \frac{1}{A_i} \int_{\text{surf } A_i} \int_{\text{surf } A_j} \frac{\cos \theta_i \cos \theta_j}{\pi r^2} dA_i dA_j$$



五、画面绘制

- 3、辐射度方法

- 由热能工程可知， $A_j F_{ji} = A_i F_{ij}$ ， 于是

$$H_i = \sum_{\substack{j=1 \\ j \neq i}}^n F_{ji} B_j A_j / A_i$$


$$A_j F_{ji} = A_i F_{ij}$$

$$H_i = \sum_{j=1}^n F_{ij} B_j$$

五、画面绘制

- 3、辐射度方法

- 综上，可得辐射度系统方程：

$$B_i = E_i + \rho_i \sum_{\substack{j=1 \\ j \neq i}}^n F_{ij} B_j$$

$$(i=1, 2, \dots, n)$$

- 假设 E_i, ρ_i, F_{ij} 均已知，此即为关于 n 个辐射度变量的线性方程组

五、画面绘制

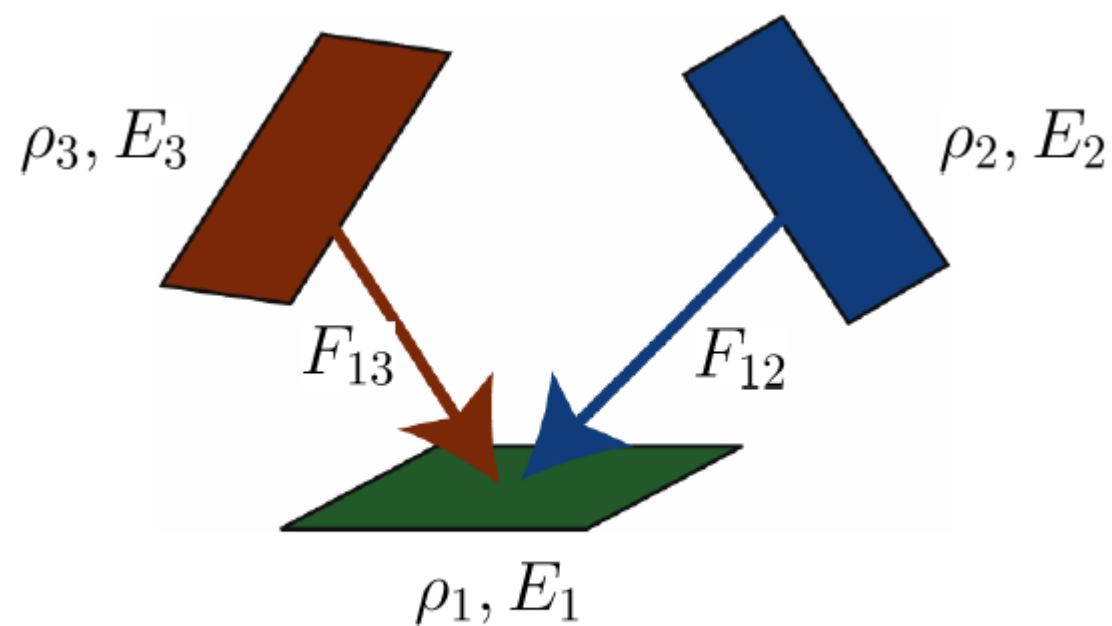
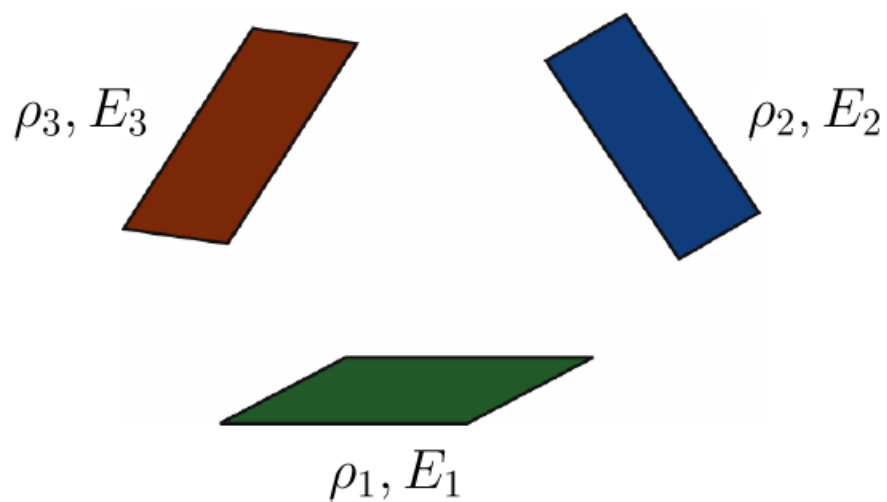
- 3、辐射度方法

- 矩阵形式 $\begin{bmatrix} F_{ij} \end{bmatrix} \begin{bmatrix} B_j \end{bmatrix} = \begin{bmatrix} E_j \end{bmatrix}$

$$\begin{bmatrix} 1 & -\rho_1 F_{12} & \dots & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1 & \dots & -\rho_2 F_{2n} \\ \dots & \dots & \dots & \dots \\ -\rho_n F_{n1} & -\rho_n F_{n2} & \dots & 1 \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \dots \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \dots \\ E_n \end{bmatrix}$$

五、画面绘制

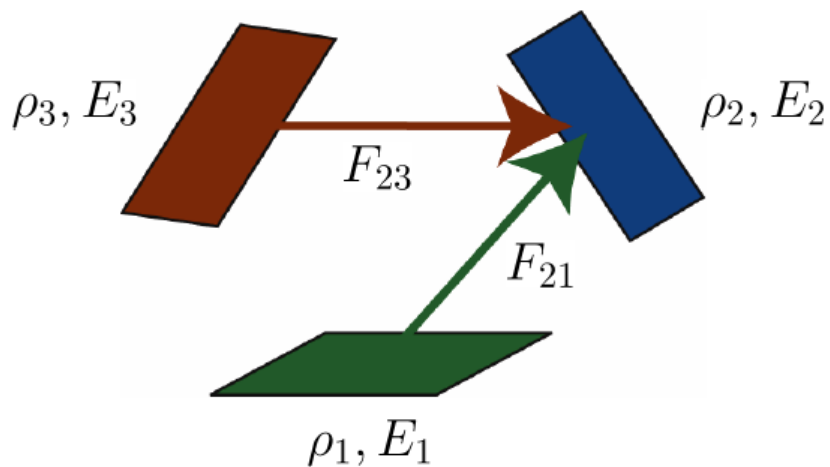
- 3、辐射度方法
 - 例子



$$B_1 = E_1 + \rho_1 F_{12} B_2 + \rho_1 F_{13} B_3$$

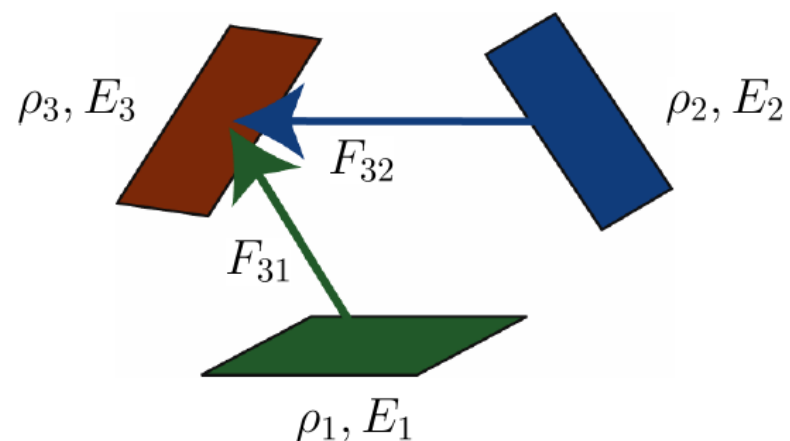
五、画面绘制

- 3、辐射度方法
 - 例子



$$B_1 = E_1 + \rho_1 F_{12} B_2 + \rho_1 F_{13} B_3$$

$$B_2 = E_2 + \rho_2 F_{21} B_1 + \rho_2 F_{23} B_3$$



$$B_1 = E_1 + \rho_1 F_{12} B_2 + \rho_1 F_{13} B_3$$

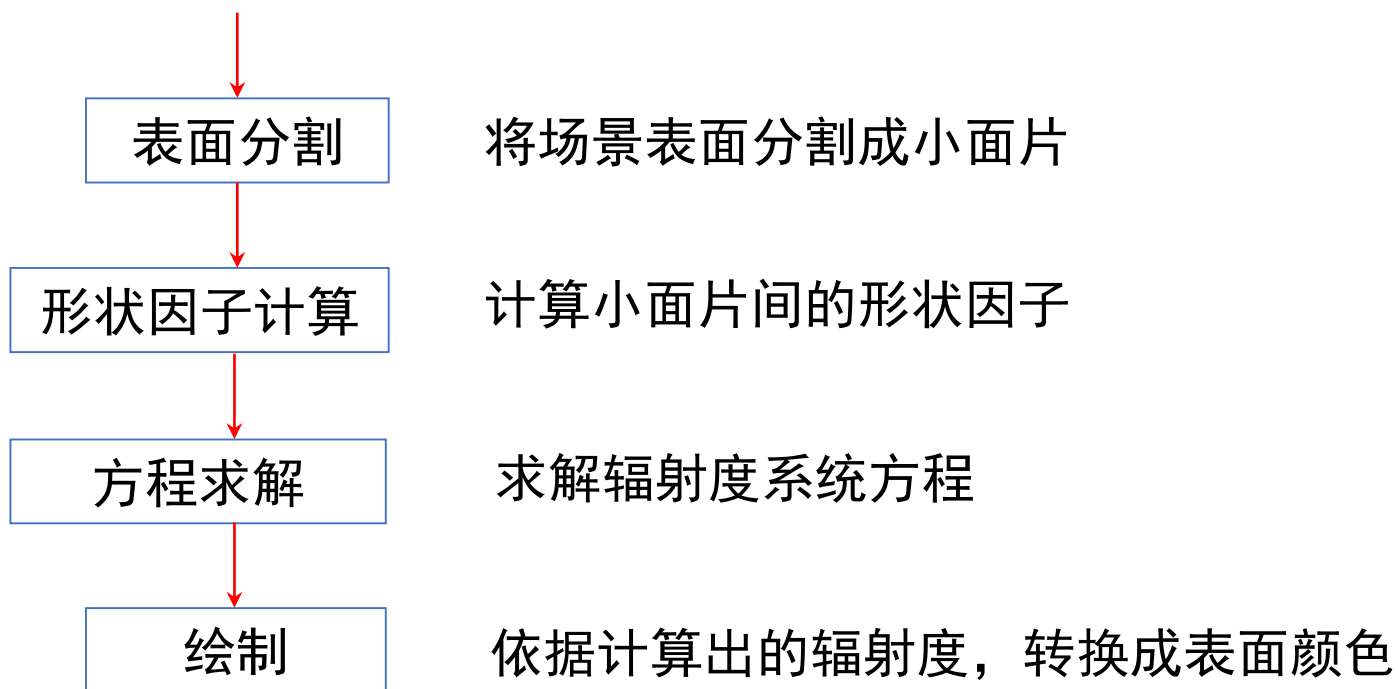
$$B_2 = E_2 + \rho_2 F_{21} B_1 + \rho_2 F_{23} B_3$$

$$B_3 = E_3 + \rho_3 F_{31} B_1 + \rho_3 F_{32} B_2$$

$$\begin{bmatrix} 1 & -\rho_1 F_{12} & -\rho_1 F_{13} \\ -\rho_2 F_{21} & 1 & -\rho_2 F_{23} \\ -\rho_3 F_{31} & -\rho_3 F_{32} & 1 \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ E_3 \end{bmatrix}$$

五、画面绘制

- 3、辐射度方法
 - 算法



五、画面绘制

- 3、辐射度方法

- 特点

- F_{ij} 计算量大;
 - F_{ij} 与 E_i 、 ρ_i 无关, 重新调节光源、景物表面颜色无需计算 F_{ij} ;
 - B_i 与观察方向无关, 可以根据观察方向迅速调整图形, 便于实现图像空间消隐。



[Cohen et al. 1988]

The End