计算机图形学
Computer Graphics

刘利刚

lgliu@ustc.edu.cn

http://staff.ustc.edu.cn/~lgliu

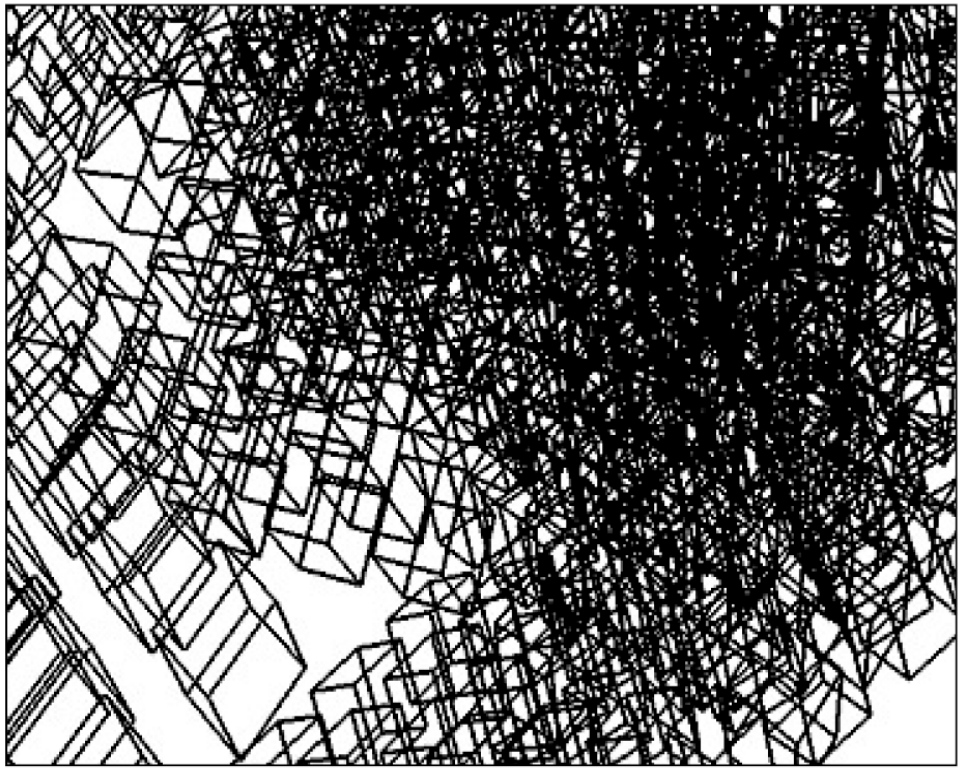
Graphics&Geometric Computing Lab @USTC

明暗（着色）处理

第一节　基本算法

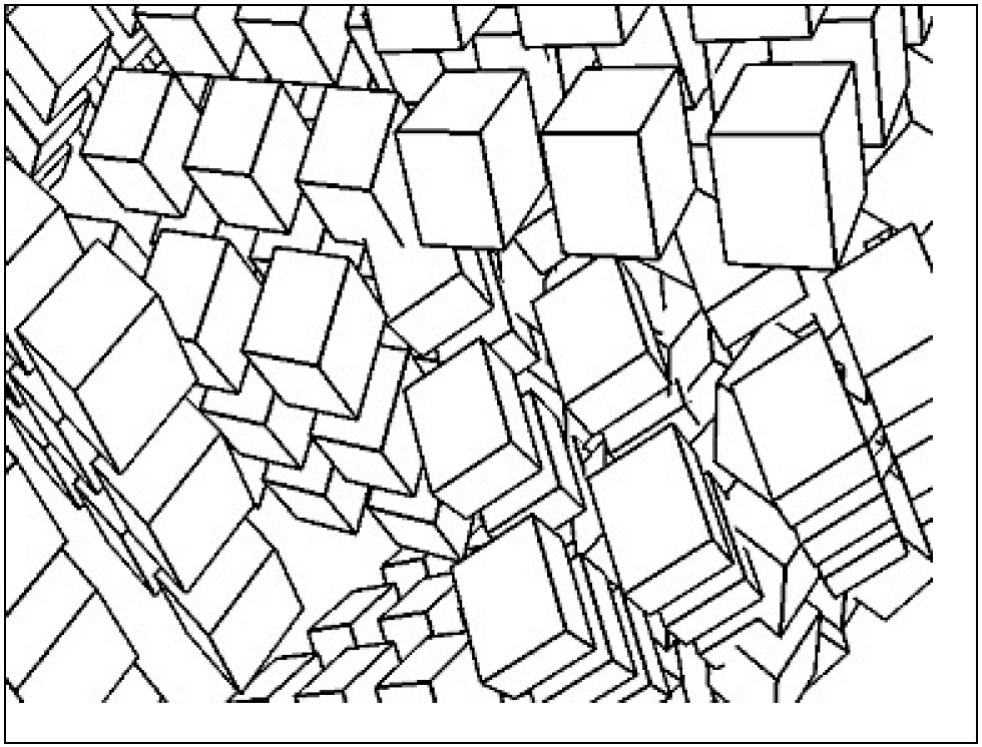
多边形网格模型

- 在计算机图形学中的模型通常是由多边形网格构成的
- 右图以线框形式显示一个场景，由540个立方体组成



隐藏面消除

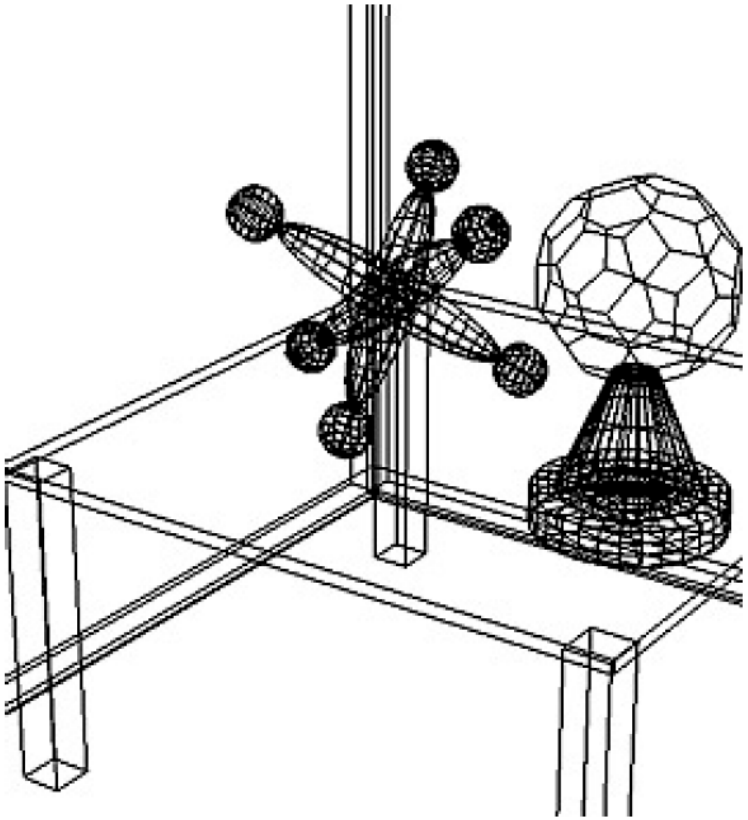
- 使多边形网格显得真实的第一种方法是隐藏面消除。
- 右图显示的是前一页场景经隐藏面消除后的结果
- 但结果仍然显得平坦



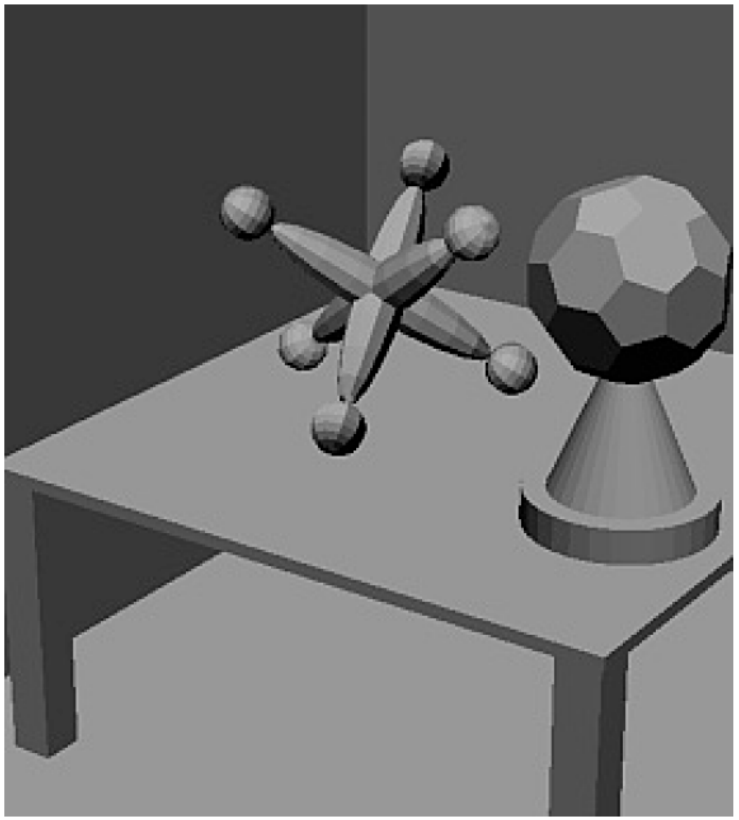
模拟光照

- 在隐藏面消除后，为了使对象看起来更真实，应当模拟光照在物体的状态，即应当通过计算确定表示对象的像素的适当状态
- 在这种计算中应充分根据对象表面的状态，光源的位置以及视点的位置
- 需要计算每帧图像中各个像素的颜色亮度，而不是由用户直接指定

示例



线框模型

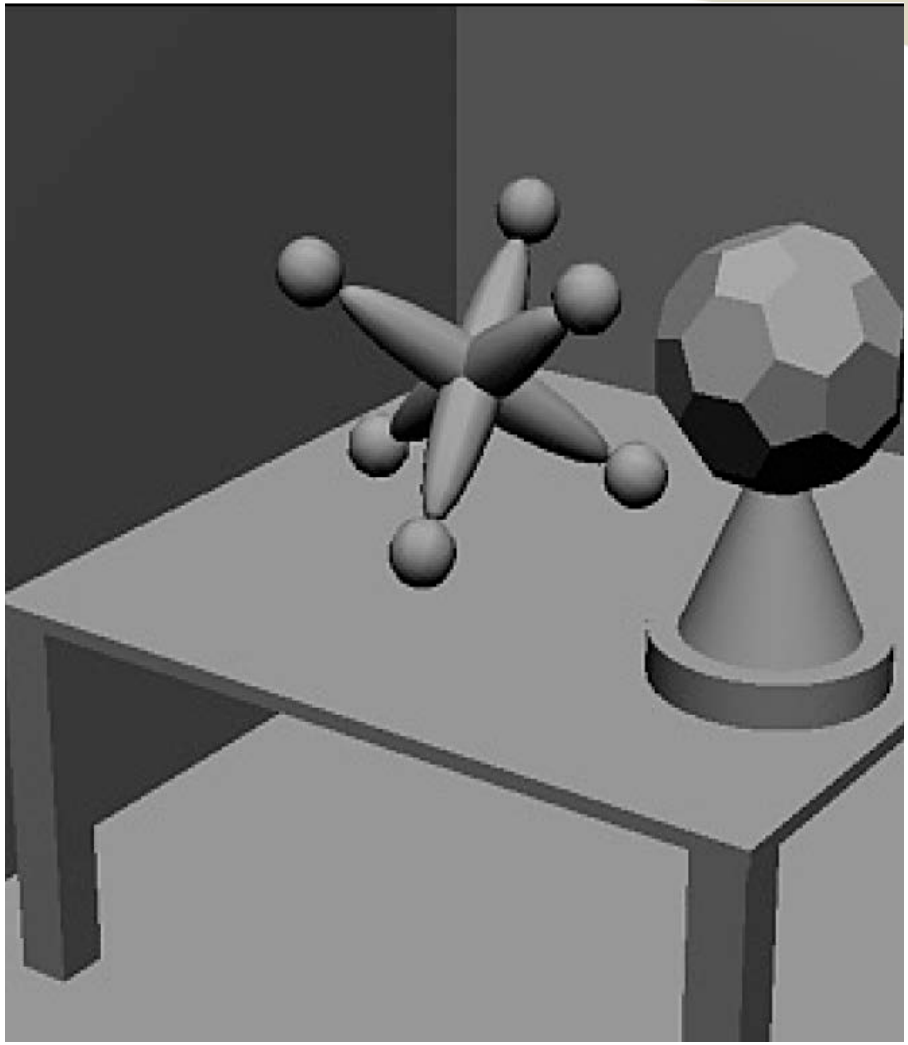


经过了简单光照处理和
隐藏面消除

平坦明暗处理

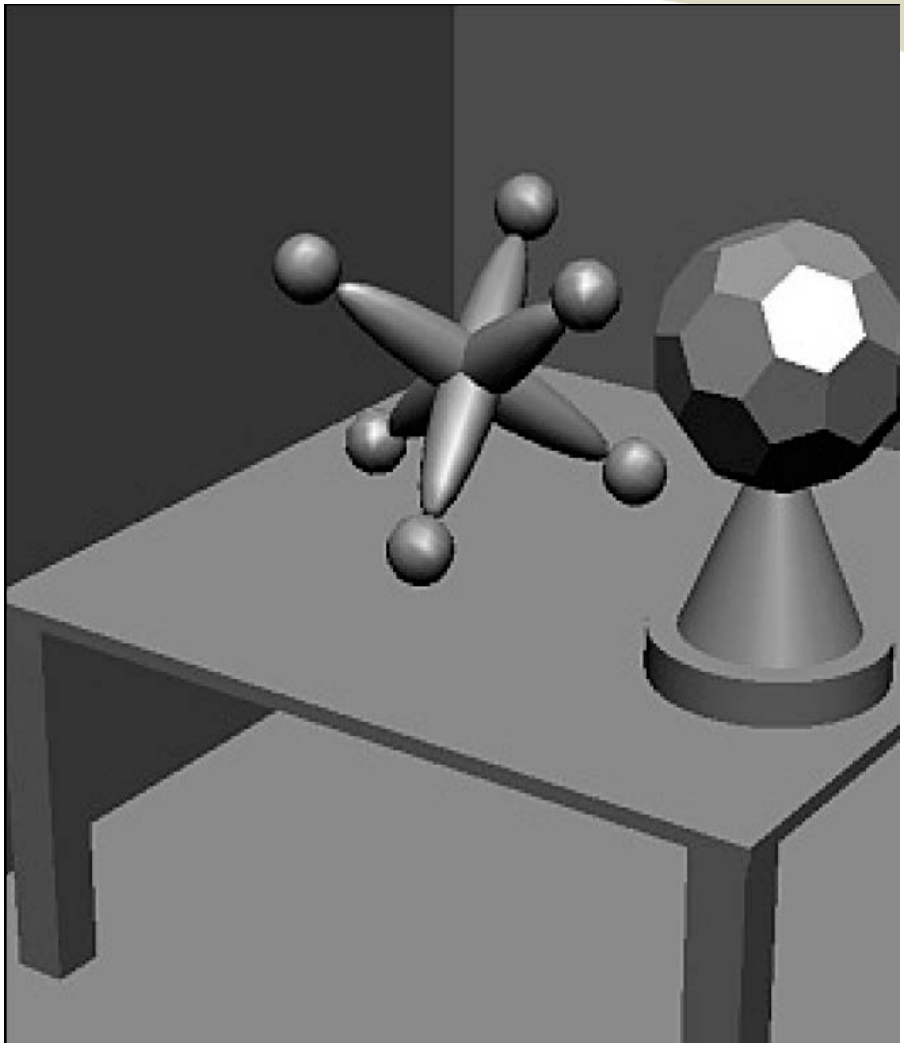
光滑明暗处理

- 如果多边形网格表示的对象为光滑对象，那么显示的结果应当反映这种光滑性
- 在计算了每个顶点处的亮度后，应用线性插值计算出内部的亮度——称为Gouraud明暗处理算法



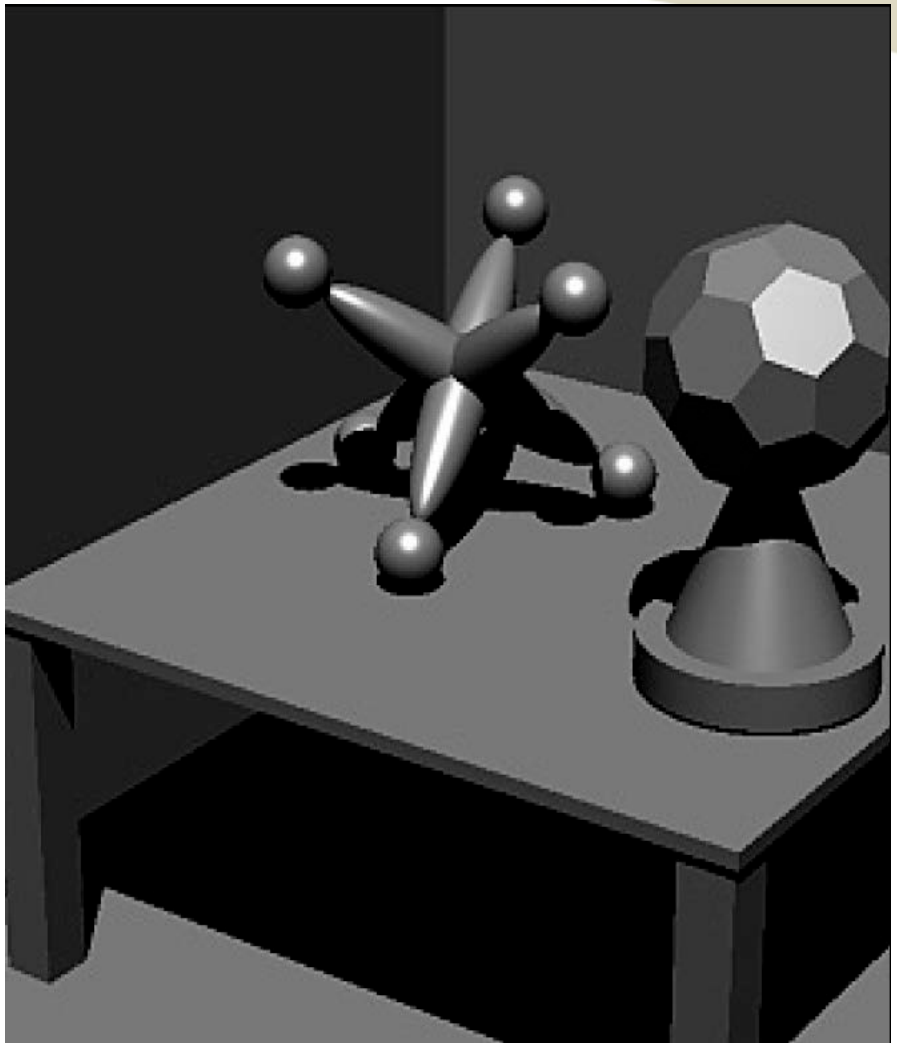
镜面光

- 为了得到更真实的效果，可以加入镜面光的效果



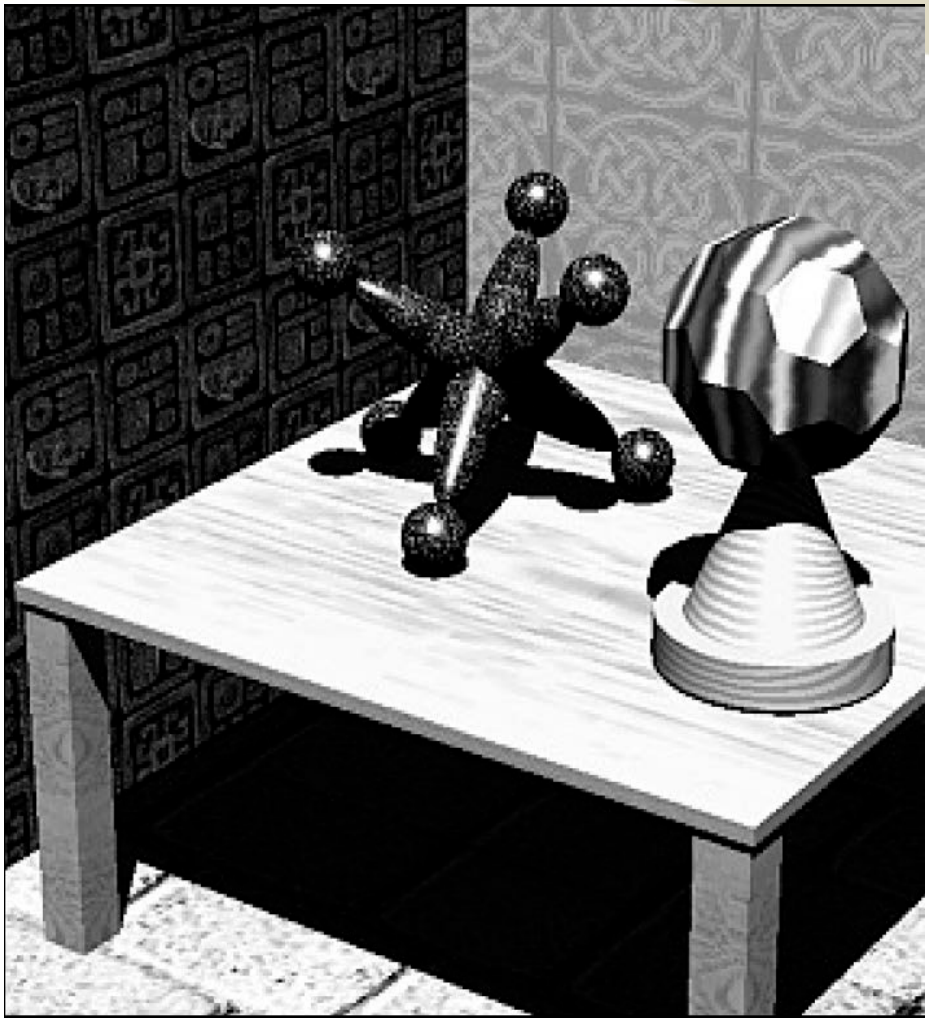
阴影

- 在加入阴影后，可以进一步提高图像的真实感



纹理映射

- 在对象表面上加上纹理效果，会使图像的真实感增强一大步



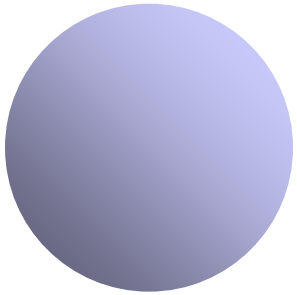
其它增加真实感的方法

- **光线跟踪**
  - 计算复杂
  - 容易实现，生成的图形中正确地反映阴影、镜面反射以及透明的效果
- **辐射度方法**

明暗处理

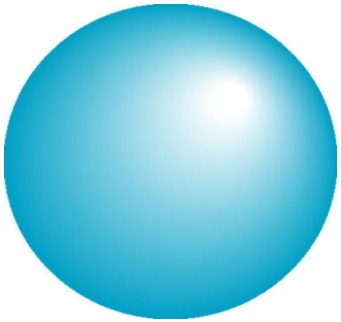
- 假设应用多边形网格建立了球面的模型，其颜色采用glColor定义，那么得到的结果为

- 而我们希望结果为



为什么？

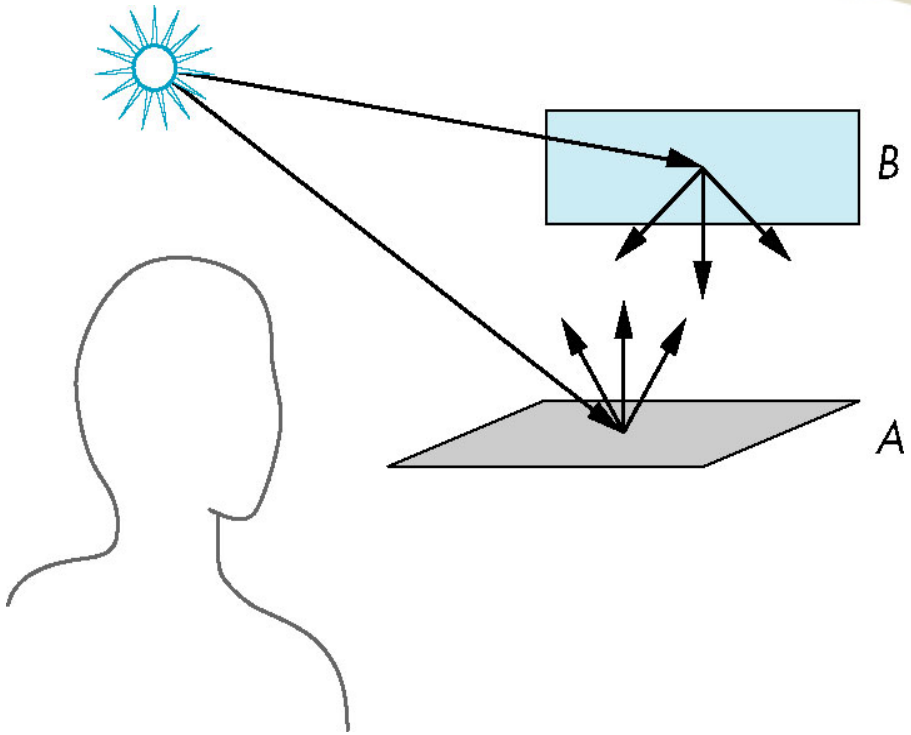
- 为何实际中球的图像应当类似于



- 光源与材料的交互作用导致每点有不同的颜色或者明暗效果
- 这时需要考虑
  - 光源
  - 材料属性
  - 观察者位置
  - 曲面定向

散射

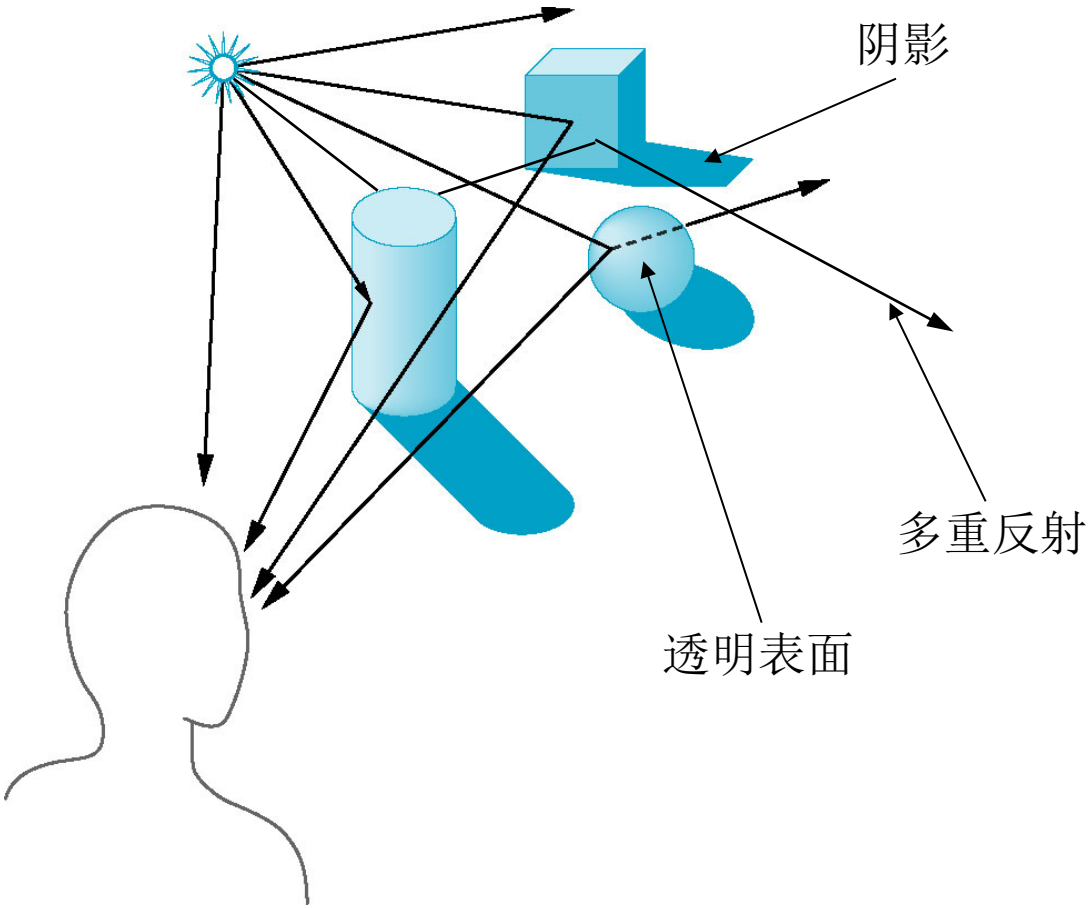
- 光照射到A点
  - 有些被反射
  - 有些被吸收
- 反射光中有些射到B点
  - 有些被反射
  - 有些被吸收
- 反射光中又有些射到A点，……



光照方程

- 光的无穷次反射和吸收过程可以用光照方程（rendering equation）描述
  - 一般是无法求解的，即使应用数值方法也是非常复杂的
  - 光线跟踪只适用于完全反射曲面这一特殊情形的
- 光照方程是全局的，并且包含
  - 阴影
  - 对象间的多次反射
- 以后会在光线跟踪和辐射度方法中详细考虑光照方程

全局效果

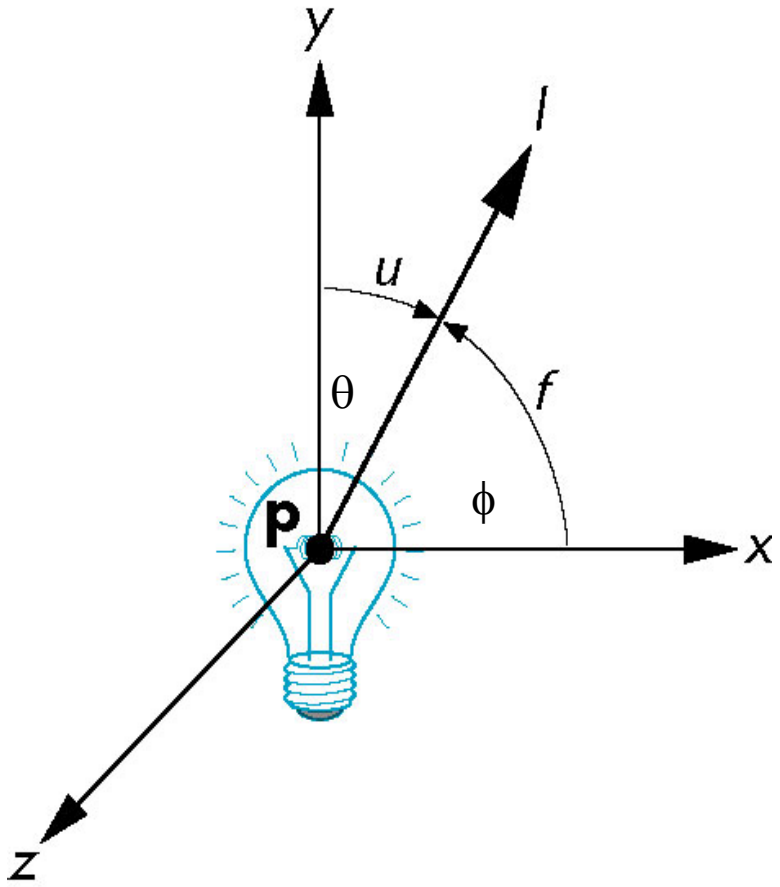


局部与全局光照

- 正确的明暗处理需要全局计算，即包含了所有的对象和光源
  - 这与流水线模型不兼容，在这个体系中对每个多边形单独进行明暗处理（局部光照）
- 然而在计算机图形学中，特别是在实时图形应用中，如果所得结果看起来可以的话，这种局部计算是可以接受的
  - 存在许多方法逼近全局效果

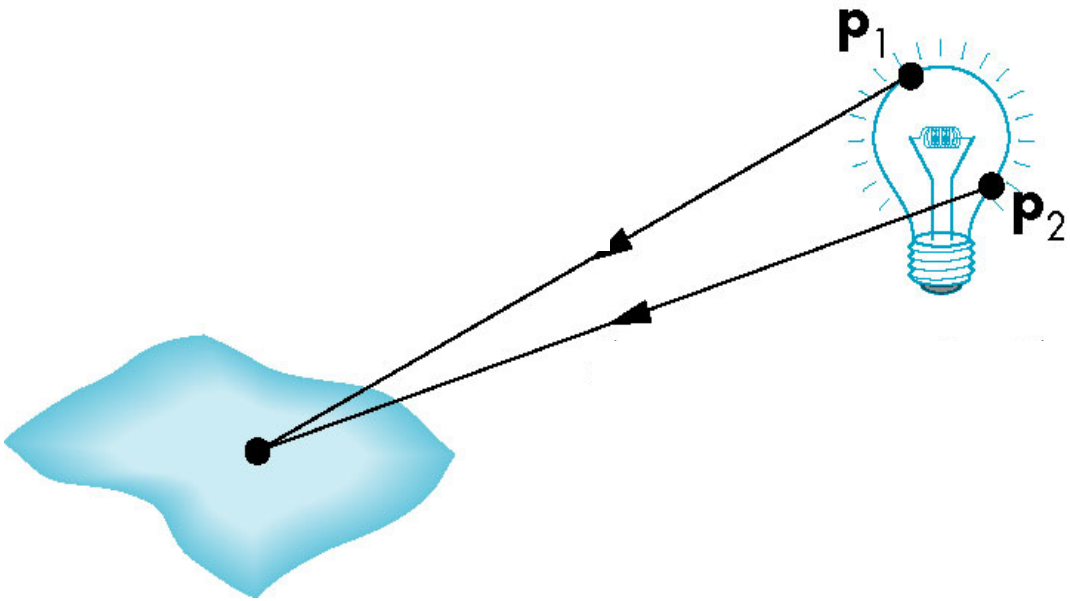
光源的描述

- 光线从光源表面离开的方式有两种
  - 自发射
  - 反射
- 在表面上任一点(x,y,z)所发出的光可以用θ，ϕ两个方位角表示以及每个波长λ的强度确定
  - 光照函数有六个变量



光源

- 一般的光源是很难处理的，因为我们需要对于在光源面上的所有点进行光强积分

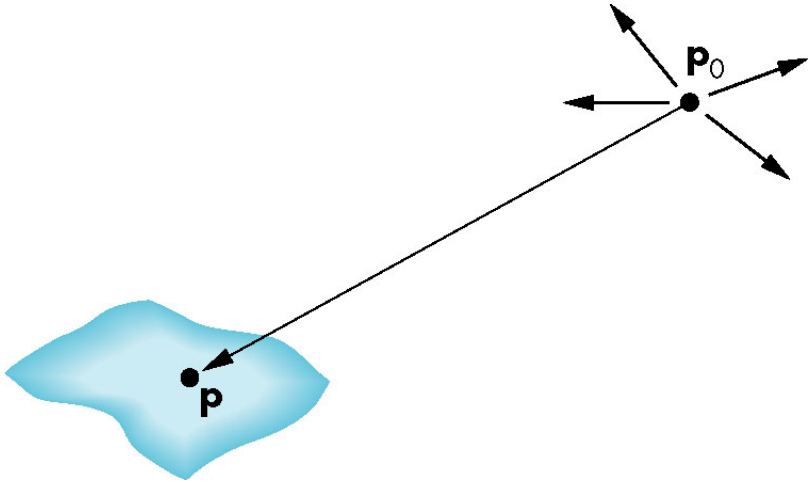


光源的颜色

- 光源不但发射出不同量的不同频率光，而且它们的方向属性随着频率也可能不同
  - 真正的物理模型将非常复杂
- 人的视觉系统是基于三原色理论的
- 在大多数应用中，可以用三种成分—红、绿、蓝—的强度表示光源
  - 光亮度(luminance)函数为$\mathbf{I} = [I_r, I_g, I_b]$

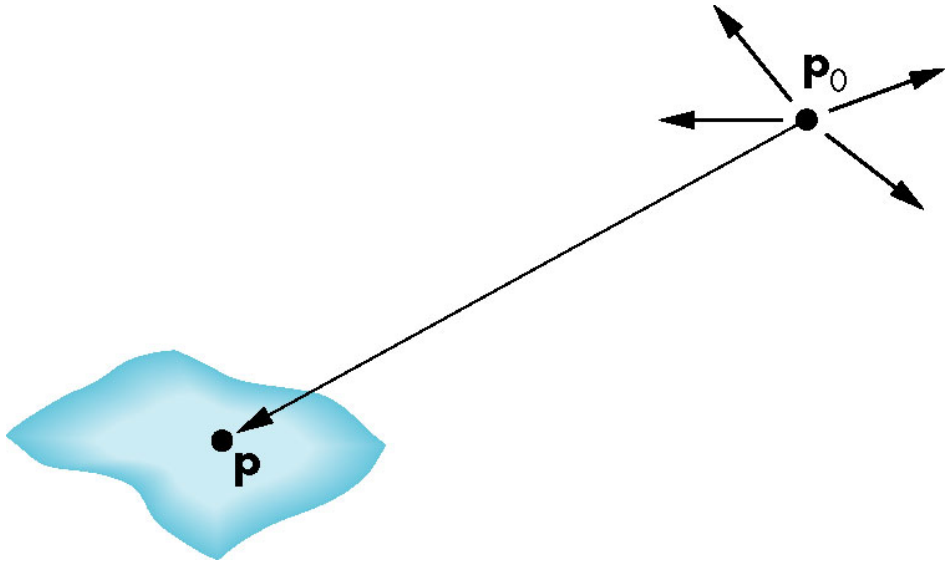
点光源

- 由位置和颜色表示
- 理想的点光源向各个方向发射光
- 远光源　即在无穷远处的光源，光线为平行线
- 点光源的亮度函数$I(p_0) = [I_r(p_0), I_g(p_0), I_b(p_0)]$
- 在点p接受的光亮度反比于
  光源与点的距离



点光源的亮度

$$i(\mathbf{p},\mathbf{p}_0)=\frac{1}{|\mathbf{p}-\mathbf{p}_0|}I(\mathbf{p}_0)$$

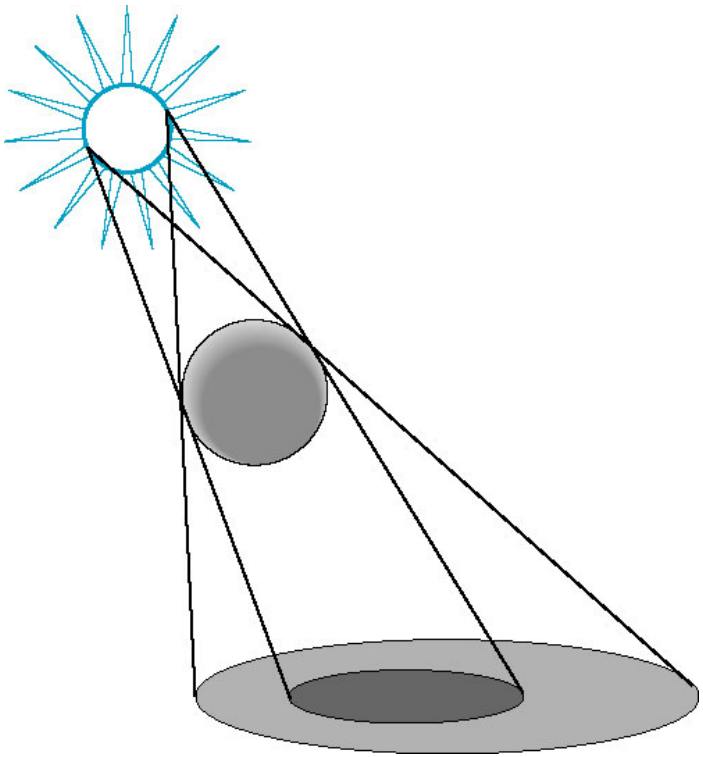


点光源的应用

- 在计算机图形学中大量应用点光源，是因为它易于使用
- 但不能很好地反映物理现实
  - 只有点光源的场景得到的图像中对比度比较高；对象显得要么很亮，要么很暗
  - 而真实的光源由于尺寸较大，因此场景的结果比较柔和

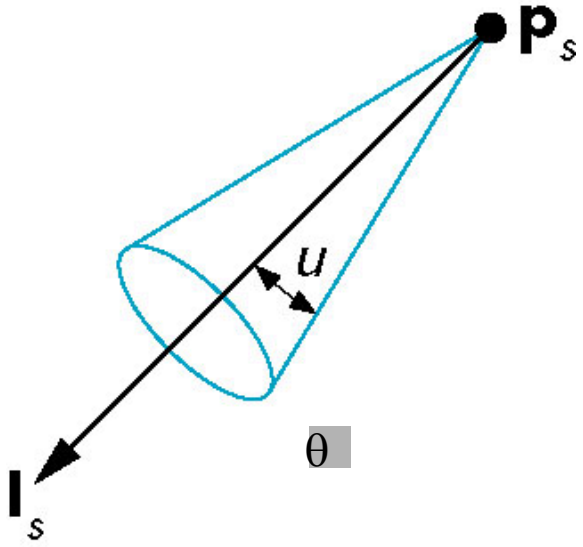
真实光照

- 完全在阴影中的区域称为本影（umbra）
- 部分在阴影中的区域称为半影（penumbra）
- 可以在点光源中加入环境光降低高对比度的问题



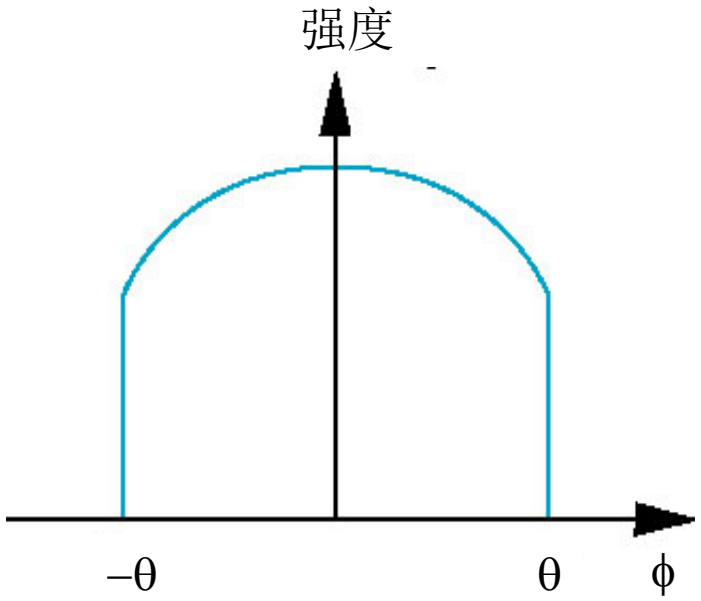
聚光灯(spotlight)

- 具有一个比较的窄的照明范围，通常为圆锥形半无穷区域
  - 可以给点光源加上一定的限制得到
  - 锥的顶点在P_s，而中心轴方向为l_s。
  - 如果$\theta = 180°$，聚光灯成为点光源



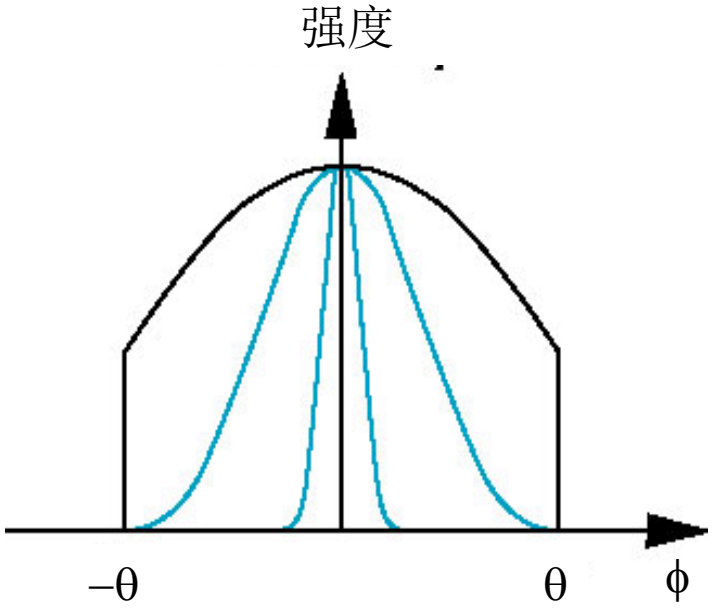
更真实的聚光灯

- 光亮度在照明锥内具有一定的分布
  - 通常绝大多数光集中在照明锥的中心
  - 因此照明强度是向量s与中心轴夹角ϕ的函数



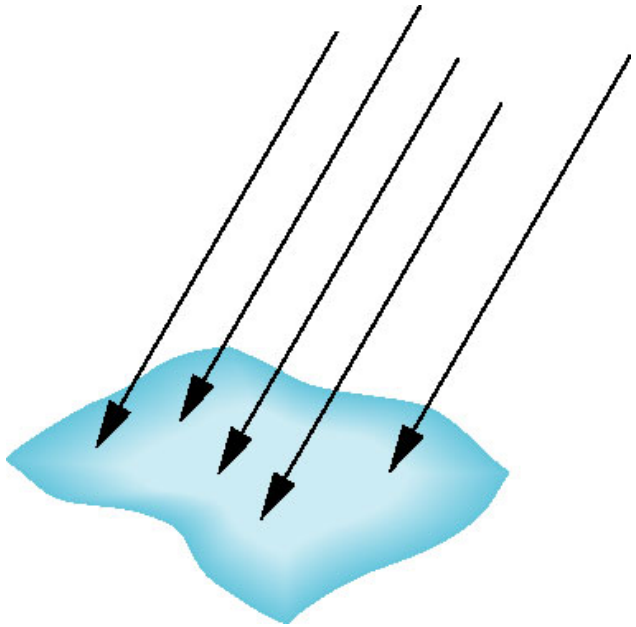
聚光灯的照明强度函数

- 可以采用各种方式定义强度函数
- 通常定义为$\cos^e\phi$
  - 指数e确定光强度衰减的快慢
  - 之所以采用余弦函数，是因为在这种情况下，非常容易计算出来它的值：$\cos\phi = s \cdot l$.



日光－无穷远光源

- 在光照计算中需要计算从光源到当前点的向量
  - 入射方向为上述向量的相反向量，而且需要单位化
- 如果光源在无穷远，该向量为常数
  - 太阳光是无穷远光源的典型代表



环境光

- 在场景中每个地方都具有相同的强度
- 严格意义上说，环境光也是来自于某个光源，但由于在光照计算中进行了某些简化，需要用环境光来模拟多次反射后的效果
- 环境光强是由$I_a = [I_{ar}, I_{ag}, I_{ab}]$确定的，在每点的值完全相同

光线与材料的相互作用

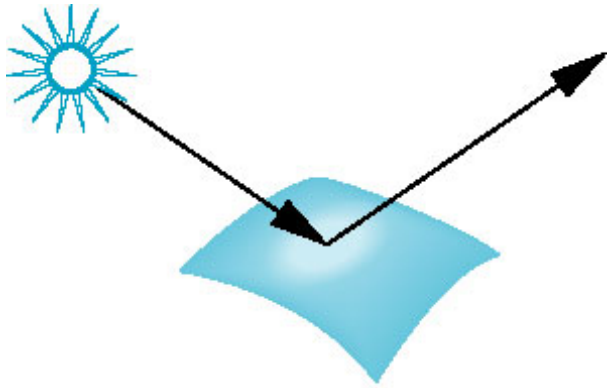
- 照射在对象上的光线部分被吸收，部分被反射
- 如果对象是透明的，有些光被折射
- 反射部分的多少确定对象的颜色与亮度
  - 对象表面在白光上看起来是红的，就是因为光线中的红色分量被反射，而其它分量被吸收
- 反射光被反射的方式是由表面的光滑程度和定向确定的

对象表面

- 如果对象表面光滑，对象显得明亮；如果表面粗糙，那么就显得暗淡
- 表面有三种类型
  - 镜面（specular surfaces）
  - 漫反射面（diffuse surfaces）
  - 透明面（translucent surfaces）

镜面

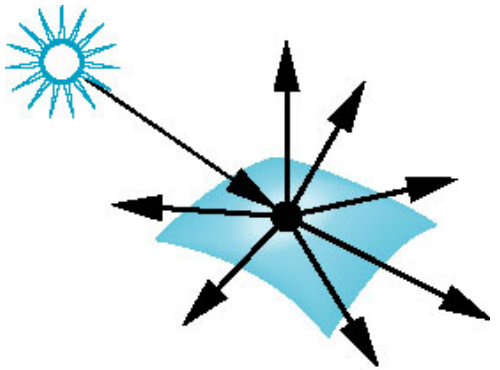
- 表面显得明亮，因为绝大多数光集中在严格镜面反射方向的周围
- 镜子是真正的镜面模型



光滑表面

漫反射面

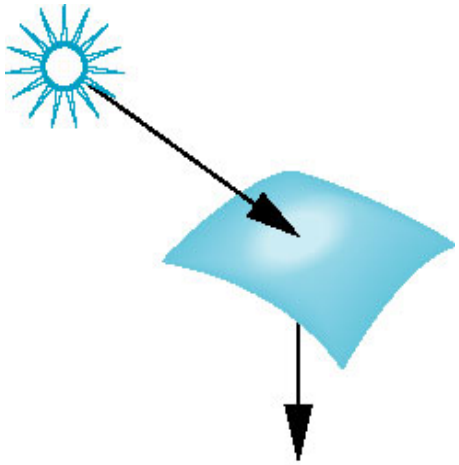
- 特征是反射光向各个方向进入空间
- 例如：涂有不光滑油漆的墙面



粗糙表面

透明面

- 有些光可以进入表面，从对象的另一处出来
- 例如：玻璃与水中的折射
- 这时也会有部分光被反射



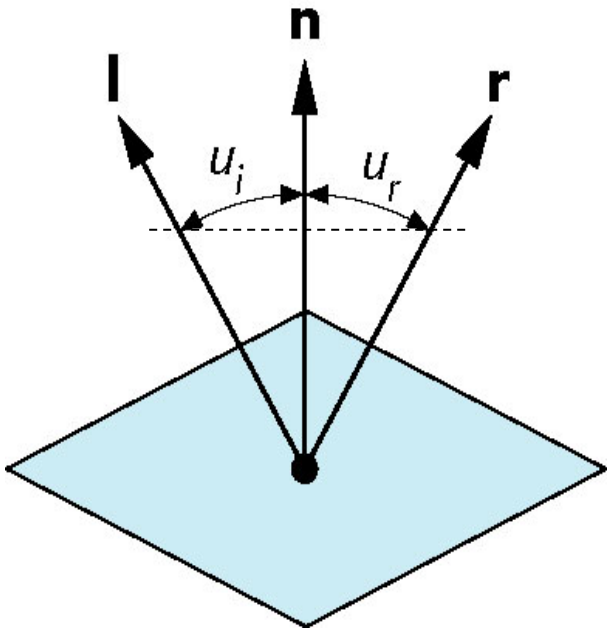
透明面

三种情形的表面

- 在三维场景中的每个对象的表面可以是上述三种情形中的任一种，也可以是其中两种或三种的综合
- 每种情形所占的比例由对象表面的性态确定

理想反射

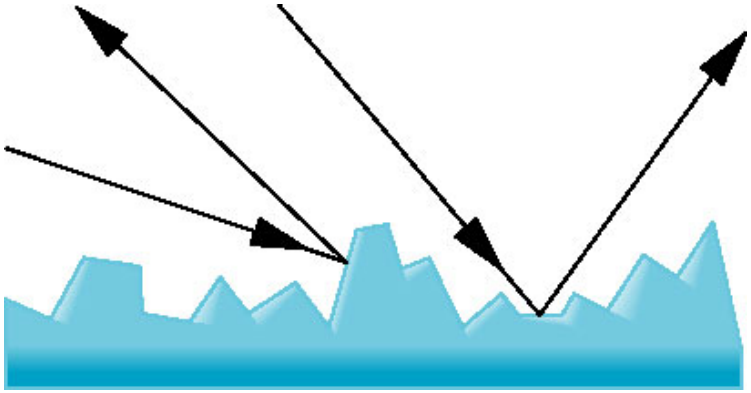
- 法向由局部定向确定
  - 单位化 |n| = 1
- 入射角 = 反射角
  - $\theta_i = \theta_r$
- 三个向量l, n, r必须共面
  - 均为单位向量

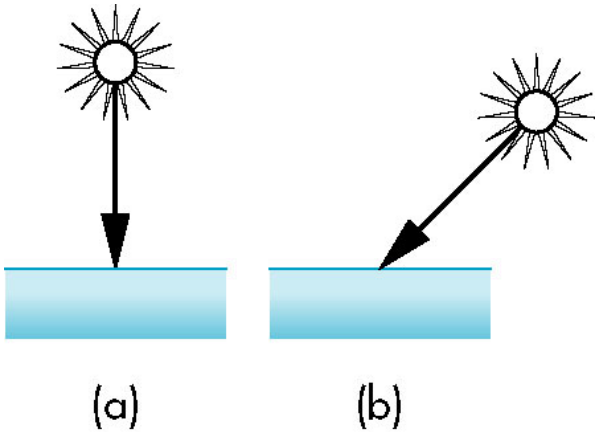


$$r = 2\,(l \cdot n)n - l$$

Lambertian曲面

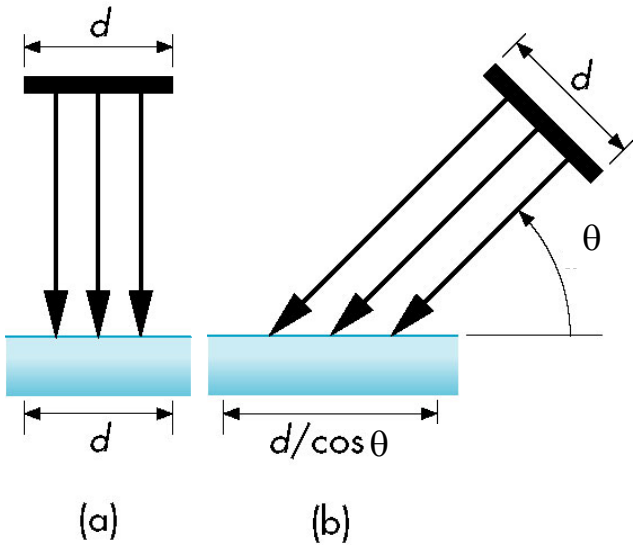
- 真正的漫反射
- 光向各个方向均匀地反射
- 模拟粗糙表面





漫反射光强

- 反射光的比例正比于入射光的竖直分量
  - 即反射光 ~ $\cos\theta_i$
  - 如果向量为单位向量，则
    $\cos\theta_i = l \cdot n$
  - 存在三个系数k_r, k_b, k_g分别相应于每种颜色的光反射的比例
  - 对每种光，反射光强
    $I_d = k\, I_l \cos\theta_i$



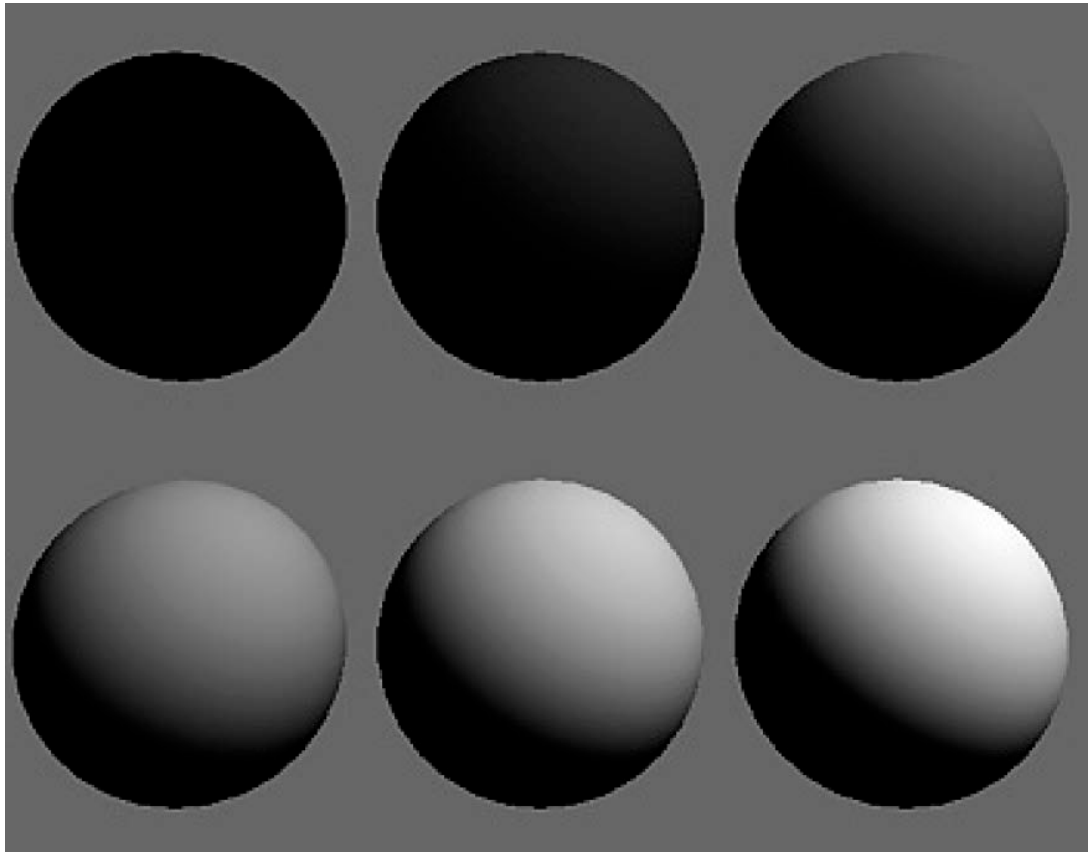
特点

- 粗糙表面之所以可见，主要是来自于表面的漫反射
- 基于简单Lambertian漫反射模型所生成的对象图形显得比较暗淡
  - 因为此时相当于假定在视点处有一个点光源，那些光源无法直接照到的地方显得较黑
  - 在真实的场景中，对象也接受到其它对象反射过来的光

漫反射参数的影响

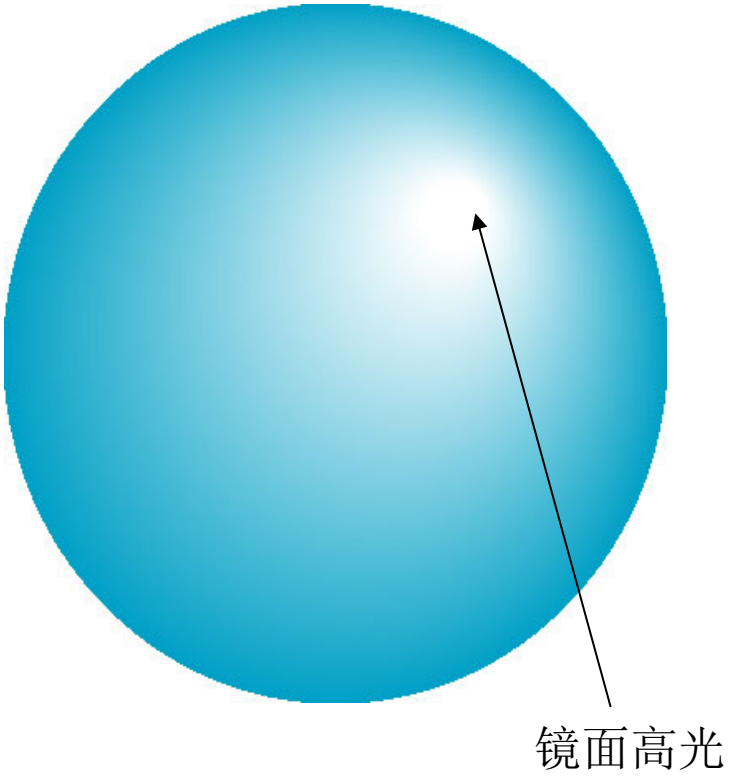
漫反射系数依次为0.0, 0.2, 0.4, 0.6, 0.8, 1.0

光强为1.0, 背景光强为0.4



镜面光

- 大多数曲面既不是理想的漫反射型曲面，也不是真正的镜面（理想反射）
- 光滑表面之所以显出镜面高光，是因为入射光被反射后，绝大多数集中在反射方向周围



镜面光的模型

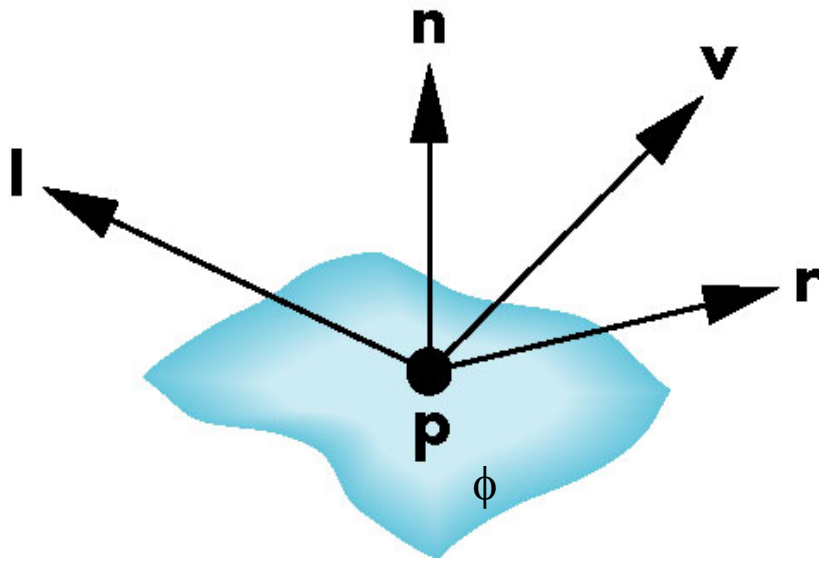
n: 法向量

l: 入射光方向

r: 反射方向

v: 视点方向

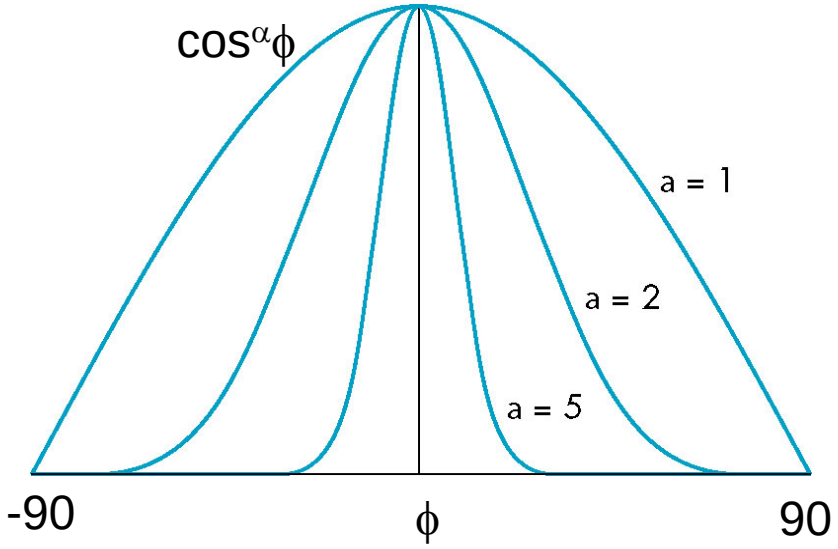
ϕ: r与v的夹角



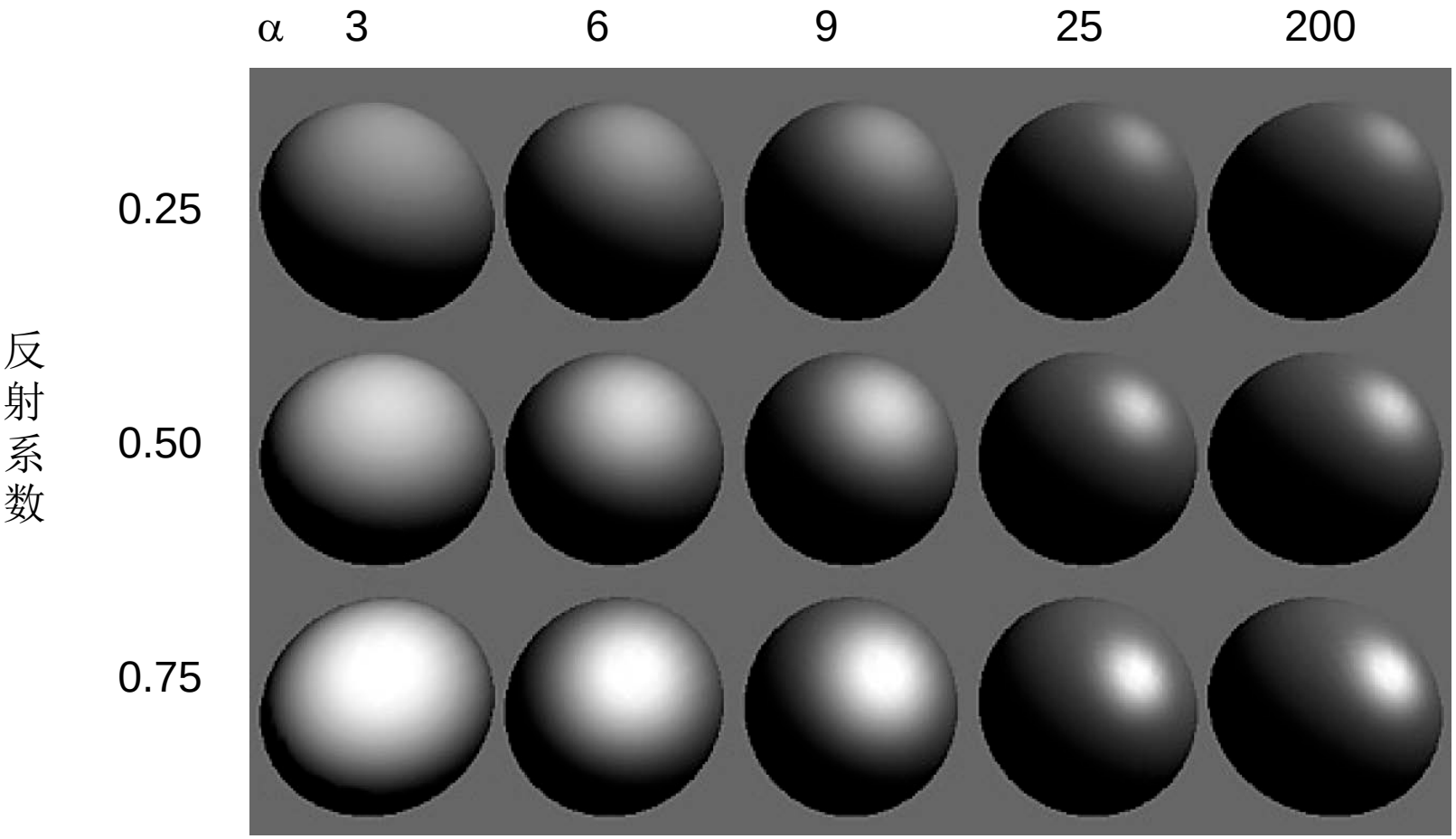
$I_r = k_s I \cos^{\alpha}\phi$

明亮系数

- 如果α的值介于100到200之间，那么对应于金属材料
- 如果α的值介于5到10之间，材料类似于塑料



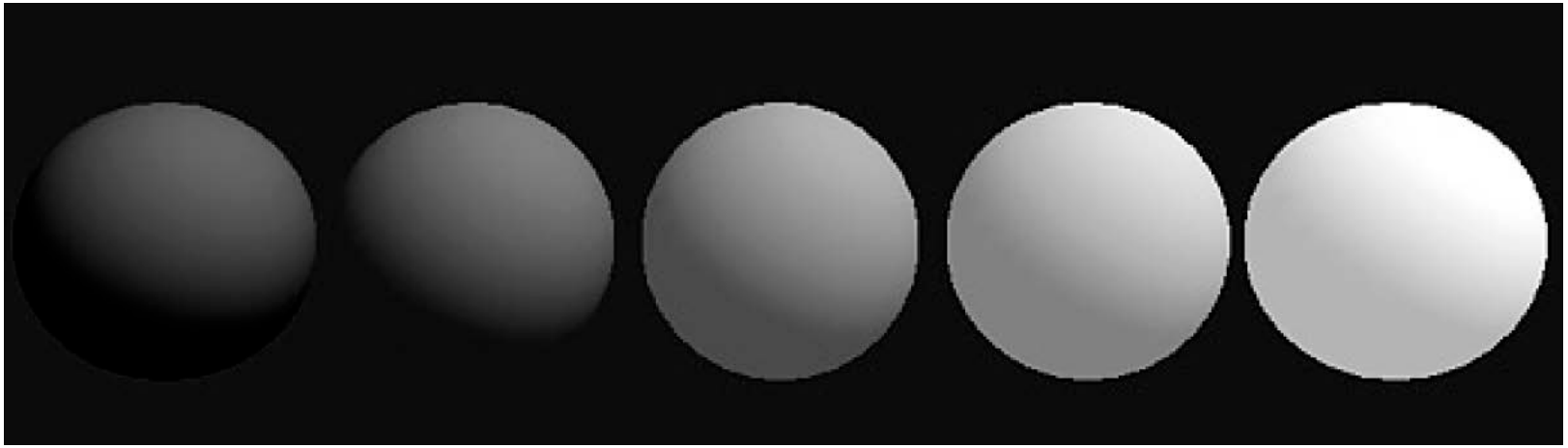
镜面反射参数的影响



环境光

- 环境光是由于在场景中光源与对象间的多次反射而导致的
- 环境光的量与颜色依赖于光源的颜色和对象的材料属性
- 向漫反射和镜面项中添加上$k_a l_a$项
  - k_a:反射系数
  - l_a：环境光强

环境光的影响



向漫反射光中加入不同的环境光的效果

两种光强都是1.0, 漫反射系数为0.04

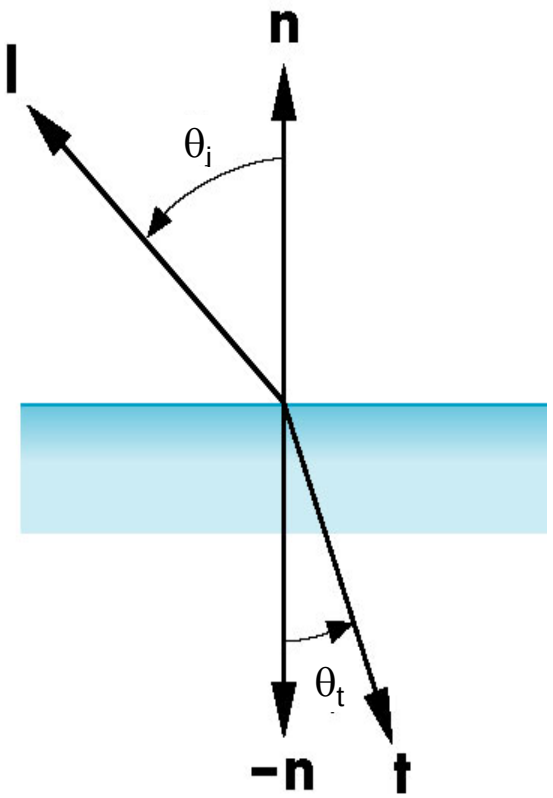
环境光反射系数依次为0.0, 0.1, 0.3, 0.5, 0.7

折射光

- Snell定律

$$\frac{\sin\theta_i}{\sin\theta_t}=\frac{\eta_t}{\eta_i}$$

η_t, η_i分别表示两种物质的折射系数



折射方向

- 令$\eta = \eta_t / \eta_i$, 则

$$\cos\theta_t = \sqrt{1 - \frac{1}{\eta^2}(1 - \cos^2\theta_i)}$$

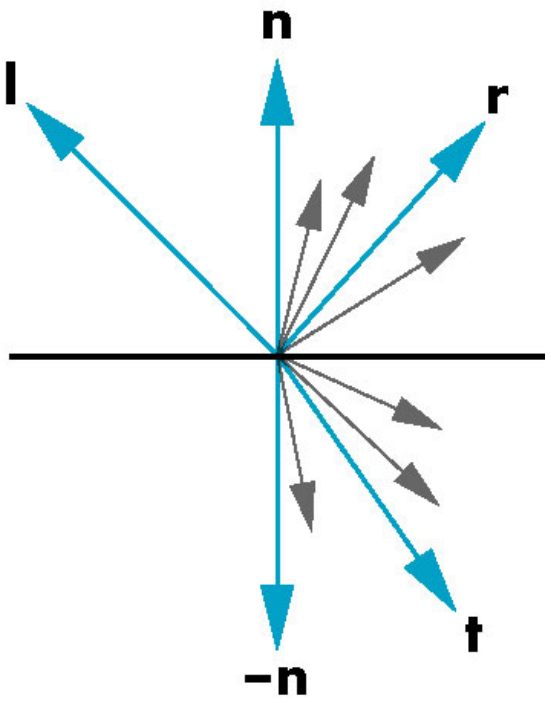
$$\mathbf{t} = -\frac{1}{\eta}\mathbf{l} - \left(\cos\theta_t - \frac{1}{\eta}\cos\theta_i\right)\mathbf{n}$$

常见材料的折射率

真空	1	空气	1.0003
水	1.33	酒精	1.36
冕牌玻璃	1.52	石英	1.54
无色玻璃	1.65	红宝石	1.77
钻石	2.42		

多种光的综合

- 在简单光照模型中，漫反射光、镜面光和折射光同时并存
- 即使在折射光中也有一部分光类似于漫反射光向各个方向发射



距离项

- 从点光源到达对象表面的光强反比于两者之间距离的平方
- 向漫反射项和镜面项中添加形式为

$$1/(a + bd + cd^2)$$

的因子，其中d表示距离

- 常数与线性项软化点光源的效果

光源

- 在光照模型中，把每个光源的结果叠加在一起
- 每个光源具有不同的漫反射、镜面和环境光项，从而充分发挥各自最大的弹性，虽然这种处理并没有任何物理上的理由
- 对三原色中每种分量单独处理
- 因此每个点光源有九个系数
  - 分别相应于三色光的三种反射系数

材料属性

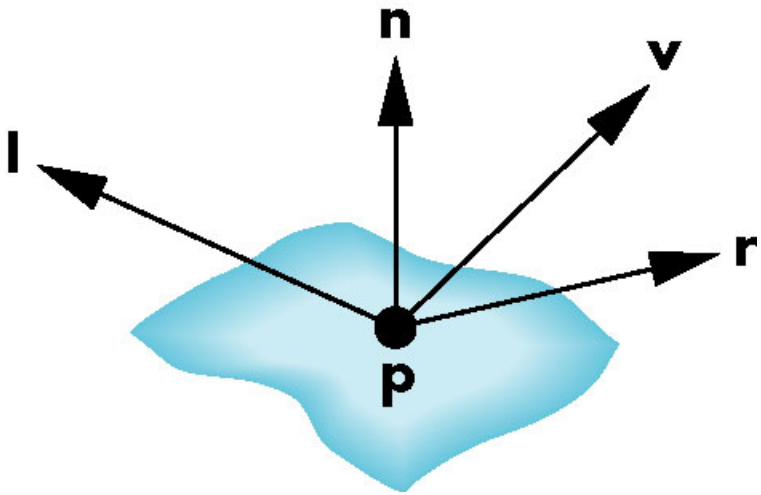
- 材料属性与光源属性完全匹配
  - 九个吸收系数 k_{dr}, k_{dg}, k_{db}, k_{sr}, k_{sg}, k_{sb}, k_{ar}, k_{ag}, k_{ab}
  - 明亮系数α

把各种分量叠加在一起

对每个光源和每种颜色成分的光，光照模型可以表示为(没有距离项)

$$I = k_d I_d l \cdot n + k_s I_s (v \cdot r)^{\alpha} + k_a I_a$$

对每个颜色分量，把所有来源的值加在一起



把各种分量叠加在一起

- 对每个光源和每种颜色成分的光，光照模型可以表示为（没有距离项）

$$I = k_d I_d l \cdot n + k_s I_s (v \cdot r)^{\alpha} + k_a I_a$$

- 对每个颜色分量，把所有来源的值加在一起

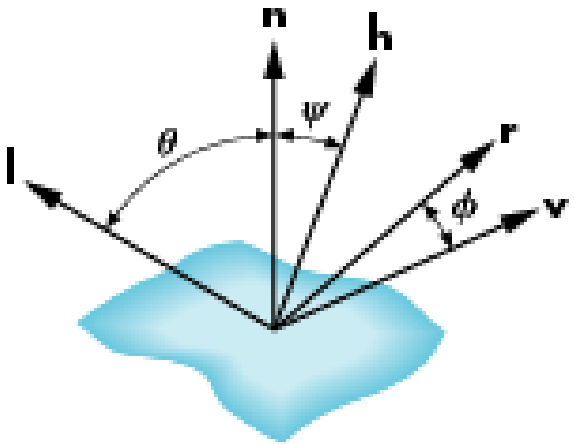
Phong模型的修正

- 在Phong模型中，镜面光项有一个问题，因为它需要为每个顶点计算一个新的反射向量和视点向量
- Blinn利用中分（halfway）向量给出了一个近似，从而使得效果更高

中分向量

- h是l和v的平分单位向量，即

$$h = (l+v)/|l+v|$$



中分角的应用

- 用$(n \cdot h)^{\beta}$代替$(v \cdot r)^{\alpha}$
  - 参数β恰当选取，以匹配明亮度
- 注意：中分角就是r和v的平分角
- 由此得到的模型称为修正的Phong模型或者Blinn光照模型
  - 在OpenGL标准中定义

向量的计算

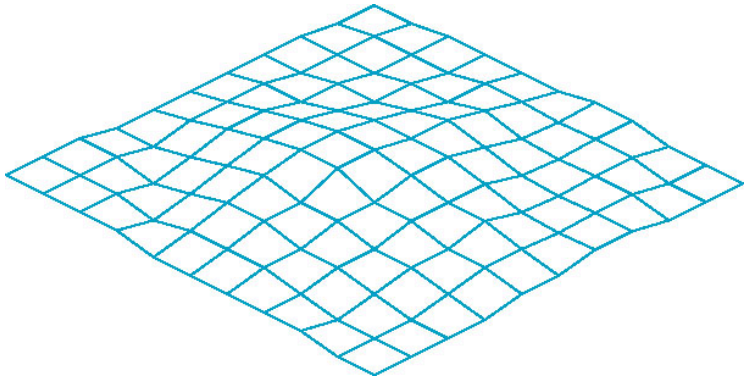
- l和v由应用程序指定
- 可以从l和n计算r
- 问题就剩下如何确定n
- 对于简单曲面，n可以被确定。但确定的方式要根据曲面的表示有所不同
- OpenGL把法向量的计算留给应用程序
  - 例外：GLU中的二次曲面和Bézier曲面

法向计算

- 当给定一组光源以及视点的位置，并可以计算出法向量，那么根据前面的模型可以计算出每点的颜色
- 但是在每点计算法向量是相当费时的工作
- 绝大多数模型是用多边形网格构成的，那么法向的计算可以大大节省

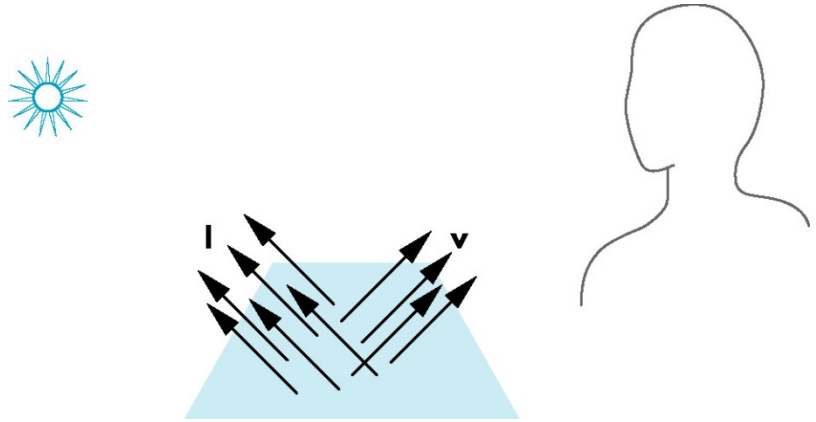
多边形网格

- 在多边形网格中每个多边形为平面，那么存在惟一的法向量
- 这样存在三种明暗处理的方法
  - 平坦处理
  - Gouraud方法(插值方法)
  - Phong方法



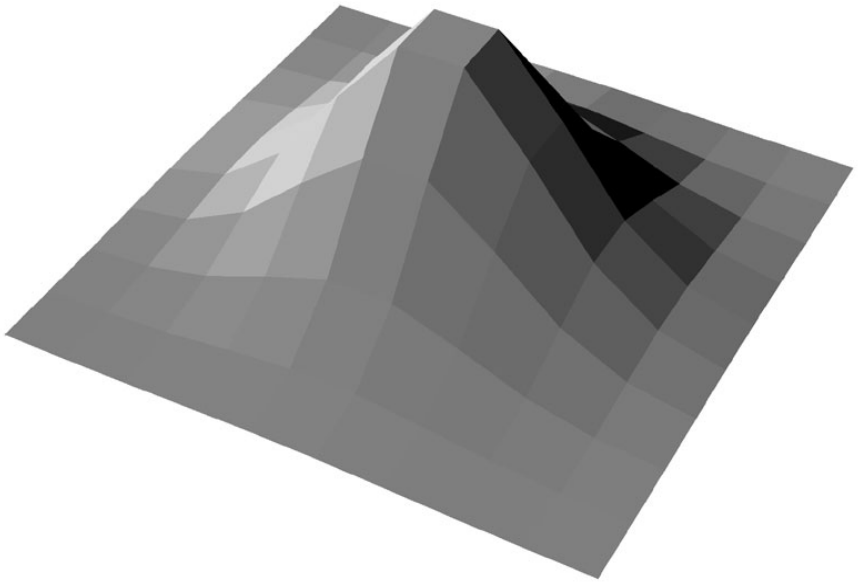
平坦方法

- 在同一多边形上法向为常向量
- 视点在无穷远，视点方向V是常向量
- 光源在无穷远，入射方向l也是常向量
- 从而对于每个多边形，只需要计算其上一点的颜色，其它点的颜色与它相同



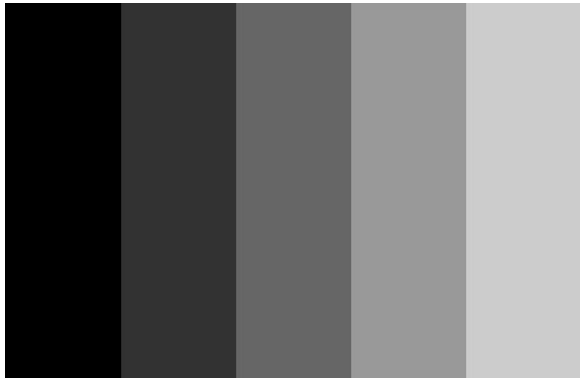
特点

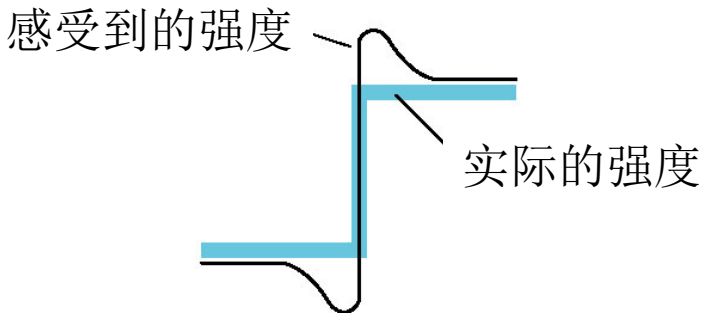
- 网格中每个多边形的颜色不同
  - 如果多边形网格表示的是一个光滑曲面，那么这种效果显然是不令人满意的



人类视觉系统

- 人类视觉系统对光强的变化非常敏感
  - 称为lateral inhibition性质
- 观察到下图边界上的条状效果，称为Mach带
- 没有办法避免这种情形，只有给出更光滑的明暗处理方法



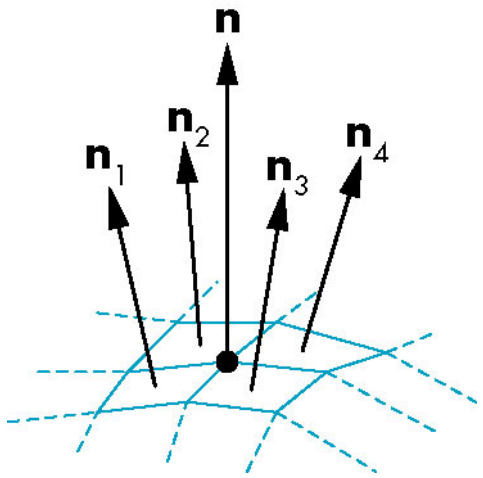


Gouraud方法

- 在网格中每个顶点处有几个多边形交于该点，每个多边形有一个法向，取这几个法向的平均得到该点的法向

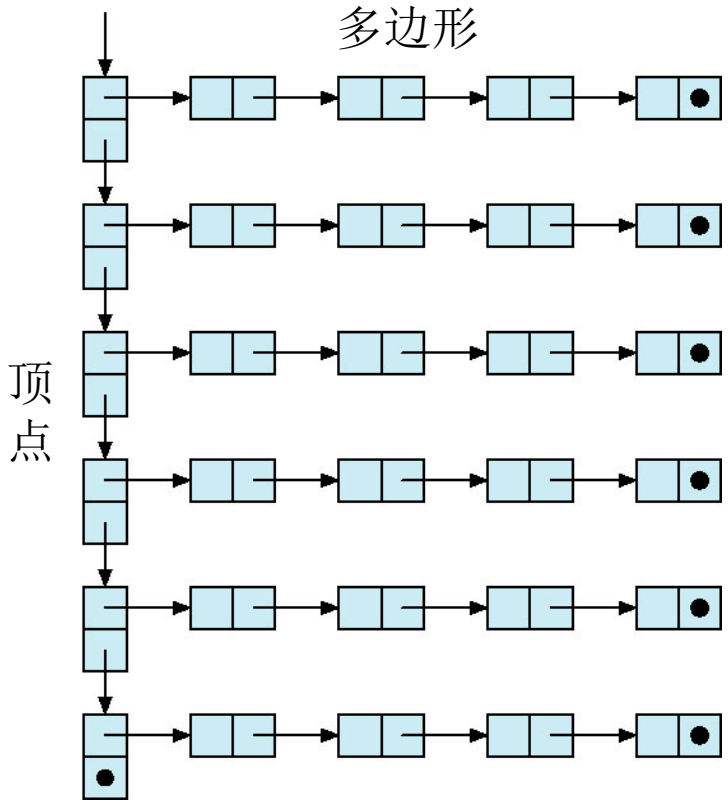
$$n = \frac{n_1 + n_2 + n_3 + n_4}{\left| n_1 + n_2 + n_3 + n_4 \right|}$$

- 然后利用简单光照模型计算出顶点的颜色
- 对于多边形内的点，采用线性插值确定颜色



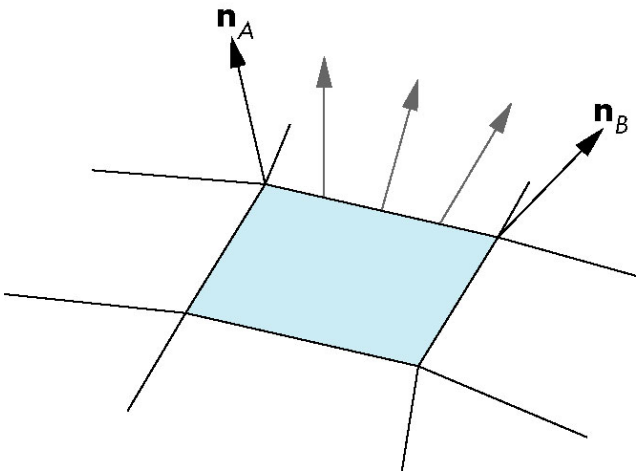
如何确定法向?

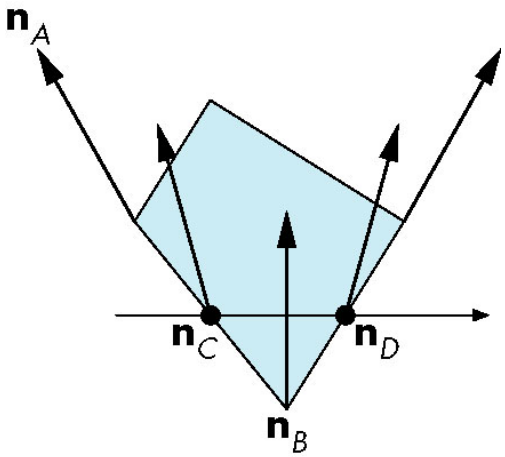
- 如何找出与某个顶点相邻的各多边形?
  - 如果程序中只是列出各顶点，那么没有信息找到上述多边形
  - 如右所示数据结果却可以做到这一点



Phong方法

- 与Gouraud方法不同，Phong方法是根据每个顶点的法向，插值出多边形内部各点的法向，然后基于光照模型计算出各点的颜色

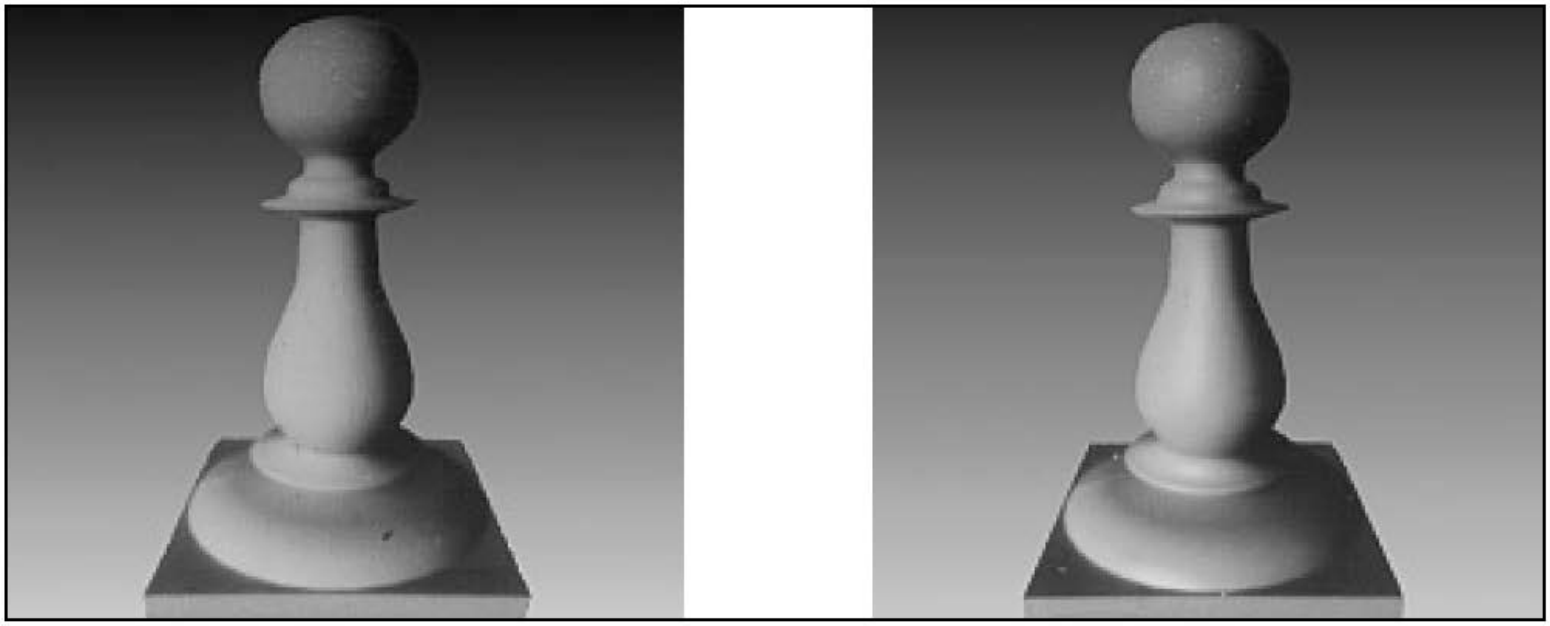




特点

- 通常会有效地降低Mach带效应
- 得到的图形比应用Gouraud方法的结果更光滑
- 但是由于法向的计算还是很复杂，一般无法得到实时图形
  - 所花费时间通常是Gouraud方法的6到8倍
- OpenGL实现的是Gouraud方法

Gouraud v.s. Phong

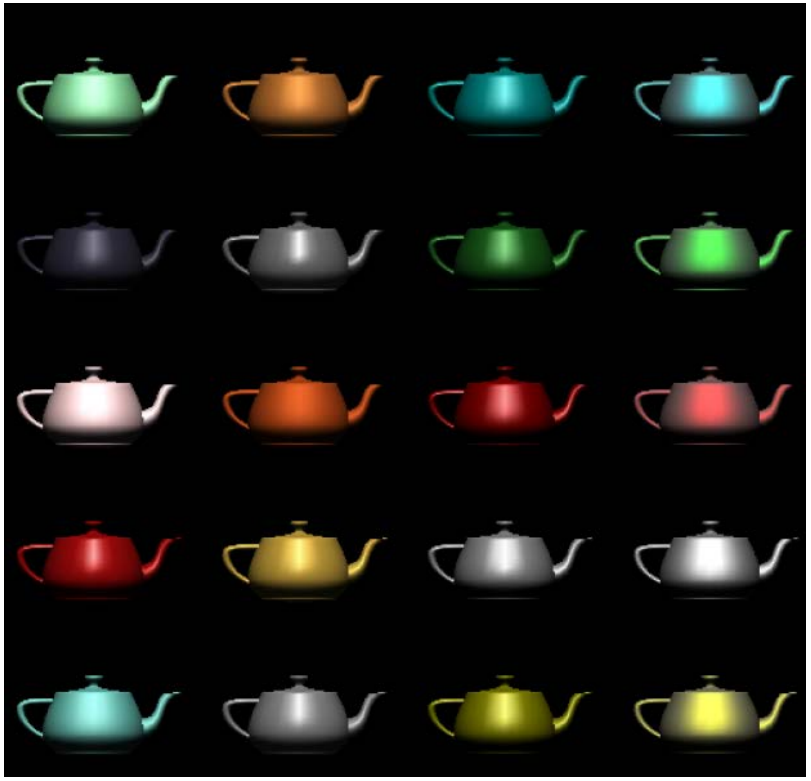


比较

- 如果用多边形网格逼近大曲率曲面，Phong方法的结果可能看起来光滑一些，而Gouraud方法就会使边有些明显
- Phong方法比Gouraud方法的复杂度高
  - 直到最近，在实时系统中还不可用
  - 目前可以用片段处理器实现
- 两种方法都需要特定数据结构表示网格，从而可以获取顶点法向量

示例

- 这些茶壶的差别就在于光照模型中参数



第二节 OpenGL中的明暗处理

在OpenGL中应用明暗处理的步骤

- 启用明暗处理功能，并选择模式
- 指定法向量
- 指定材料属性
- 指定光源

启用

- 明暗处理的计算由下述命令启用

  glEnable(GL_LIGHTING)

  - 如果光照被激活，glColor()命令被忽略
- 必须单独激活每个光源
  - glEnable(GL_LIGHTi), i = 0, 1, …7
- 可以选择光照模型的参数
  - glLightModel{if}[v](参数，值)

glLightModel*()

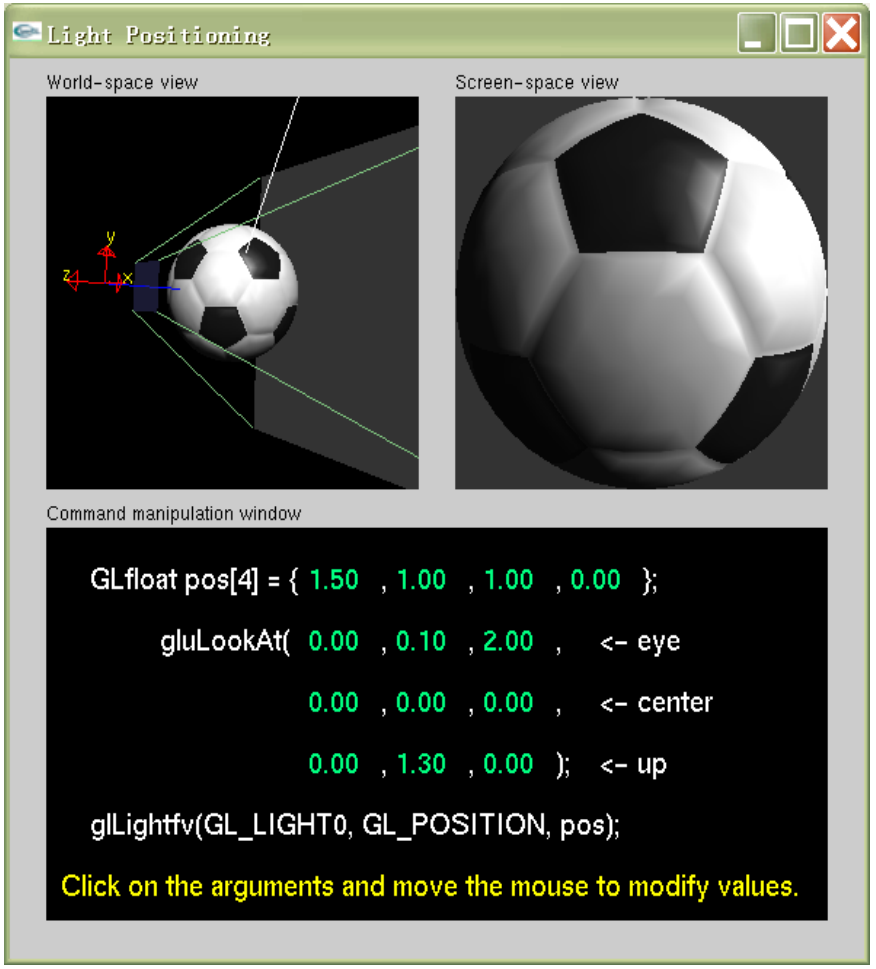
- 参数及其默认值，意义
  - GL_LIGHT_MODEL_AMBIENT，(0.2,0.2,0.2,1.0)，整个场景中的环境光强
  - GL_LIGHT_MODEL_LOCAL_VIEWER，0.0或GL_FALSE,在计算中不应用无穷远视点的假设简化计算
  - GL_LIGHT_MODEL_TWO_SIDED，0.0或GL_FALSE,单独对多边形的两面进行明暗处理
  - GL_LIGHT_MODEL_COLOR_CONTROL，GL_SINGLE_COLOR，镜面光是否与漫反射和环境光分开计算

法向量

- 在OpenGL中法向量是状态的一部分
- 利用glNormal*()设置，例
  - glNormal3d(x,y,z);
  - glNormal3dv(p);
- 通常需要法向量为单位向量，这样余弦计算就非常直接
  - 变换会影响其长度
  - 注意放缩并不保持其长度
  - glEnable(GL_NORMALIZE)可以使OpenGL自动进行单位化，当然是以损失效率为代价

光源位置定位的Tutor

- Nate Robin's lightposition tutor



定义点光源

- 对于每个光源，可以设置漫反射光、镜面光和环境光的RGB值以及光源的位置

```
GLfloat diffuse0[]={1.0,0.0,0.0,1.0};
GLfloat ambient0[]={1.0,0.0,0.0,1.0};
GLfloat specular0[]={1.0,0.0,0.0,1.0};
GLfloat light0_pos[]={1.0,2.0,3.0,1.0};

glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glLightfv(GL_LIGHT0, GL_POSITION, light0_pos);
glLightfv(GL_LIGHT0, GL_AMBIENT, ambient0);
glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse0);
glLightfv(GL_LIGHT0, GL_SPECULAR, specular0);
```

距离与方向

- 光源的颜色应当以RGBA模式定义
- 位置是以齐次坐标的形式给定
  - 如果w = 1.0, 指定的是一个有限位置
  - 如果w = 0.0, 指定的是一个平行光源，所给定的是入射光方向

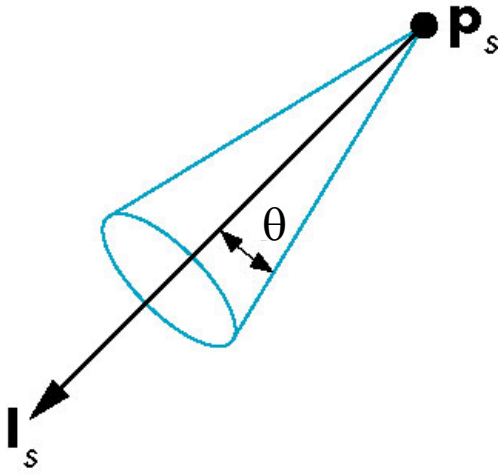
距离项的指定

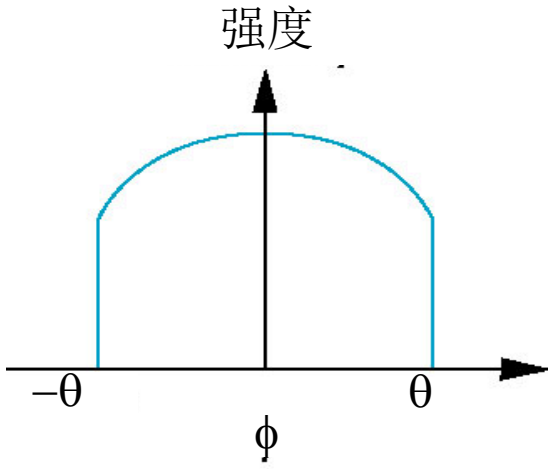
- 即光强反比于距离的因子 $a + bd + cd^2$
  - 默认值：$a = 1.0, b = c = 0.0$
  - 改变方法

```
glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, 2.0);
glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, 1.0);
glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, 0.0);
```

聚光灯

- 应用glLightfv设置聚光灯的各项参数
  - 方向：GL_SPOT_DIRECTION
  - 角度范围：GL_SPOT_CUTOFF
  - 衰减指数：GL_SPOT_EXPONENT
    - 正比于$\cos^{\alpha}\phi$





全局环境光

- 环境光依赖于每个光源的颜色，因此需要对每个光源指定环境光强
  - 在白屋中的红灯会使生成红色环境光，当灯被关闭后这种成分就消失
- OpenGL中也可以定义一个有时非常有用的全局环境光

```
GLfloat global_ambient[]={0.2,0,0,1};
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, global_ambient);
```

移动光源

- 光源是几何对象，它的位置或方向受模型视图矩阵的影响
- 把光源的位置和方向设置函数放置在不同的地方，可以达到不同的效果：
  - 和对象一起移动光源：所有的变换在光源位置和对象定义之前调用
  - 固定对象，移动光源：先定义对象，进行变换后，再定义光源位置
  - 固定光源，移动对象：先定义光源位置，进行变换后，再定义对象
  - 分别移动光源和对象：采用矩阵堆栈

静态光源代码

```
glViewport(0,0,(GLsizei)w,(GLsizei)h);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
if(w<=h) glOrtho(-1.5,1.5,-1.5*h/w,1.5*h/w,-10.0,10.0);
else glOrtho(-1.5*w/h,1.5*w/h,-1.5,1.5,-10.0,10.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
GLfloat light_pos[]={1.0,1.0,1.0,1.0};
glLightfv(GL_LIGHT0,GL_POSITION,light_pos);
```

光源与对象分开移动

```
static GLdouble spin;
void display(void){
   GLfloat light_pos[]={0.0,0.0,1.5,1.0};
   glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
   glPushMatrix();
     gluLookAt(0.0,0.0,5.0,0.0,0.0,0.0,0.0,1.0,0.0);
     glPushMatrix();
        glRotated(spin,1.0,0.0,0.0);
        glLightfv(GL_LIGHT0,GL_POSITION,light_pos);
     glPopMatrix();
     glutSolidTorus(0.275,0.85,8,15);
   glPopMatrix();
   glFlush();
}
```

与视点一起移动光源

- 为此需要在视图变换之前设置光源位置
  - 记住：光源位置是存贮在视点坐标系中，这是视点坐标系实现其作用的少数例子之一

```
GLfloat ligh_pos[]={0.0,0.0,0.0,1.0};
glViewport(0,0,(GLsizei)w,(GLsizei)h);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(40.0,(GLfloat)w/h,1.0,100.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glLightfv(GL_LIGHT0,GL_POSITION,light_pos);
//… (to be continued)
```

```
//continued
static GLdouble ex,ey,ez,upx,upy,upz;

void display(void){
 glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
  glPushMatrix();
    gluLookAt(ex,ey,ez,0.0,0.0,0.0,upx,upy,upz);
    glutSolidTorus(0.275,0.85,8,15);
  glPopMatrix();
  glFlush();
}
```

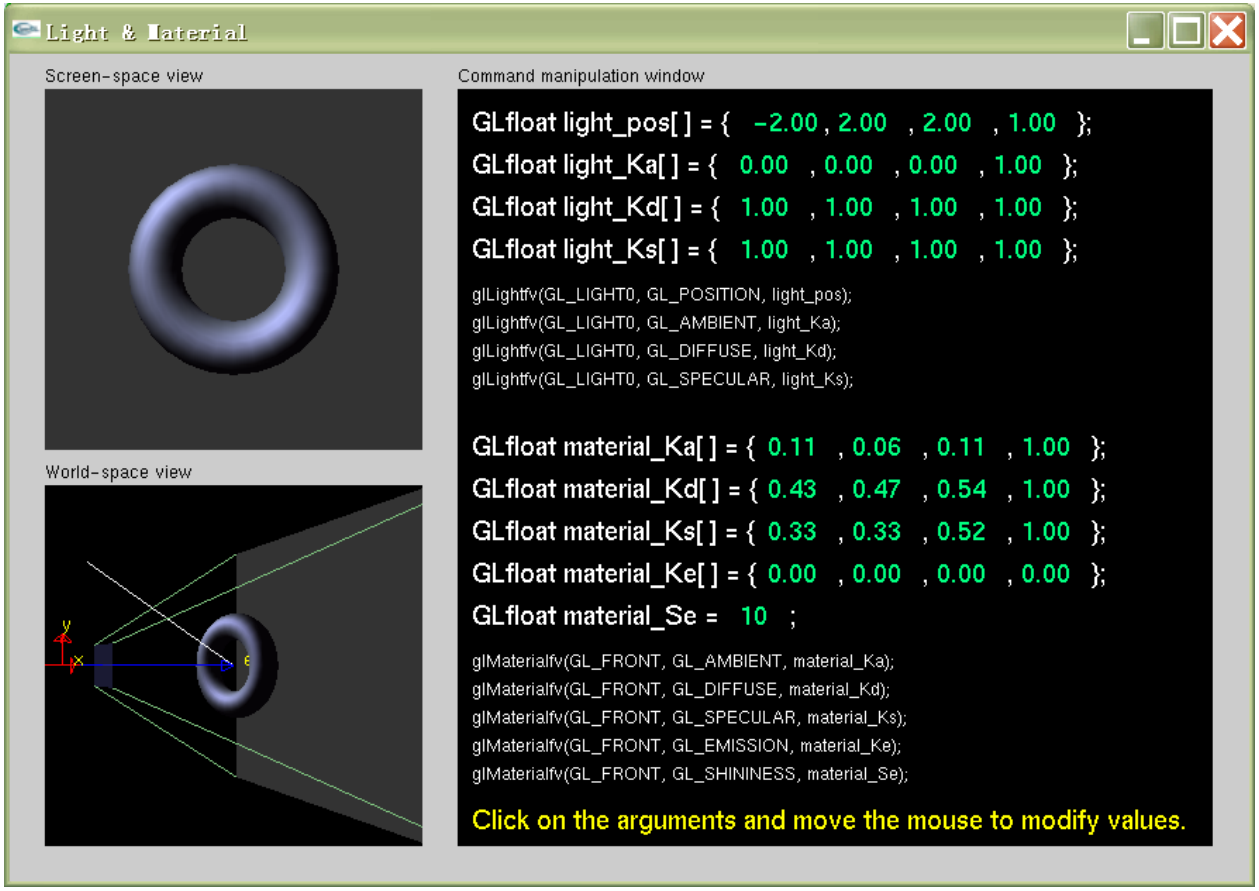
材料属性

- 材料属性也是OpenGL状态的一部分，与简单光照模型中的各项是匹配的
- 应用glMaterial{if}[v]()设置

```
GLfloat ambient[]={0.2,0.2,0.2,1.0};
GLfloat diffuse[]={1.0,0.8,0.0,1.0};
GLfloat specular[]={1.0,1.0,1.0,1.0};
GLint shine = 100;
glMaterialfv(GL_FRONT,GL_AMBIENT,ambient);
glMaterialfv(GL_FRONT,GL_DIFFUSE,diffuse);
glMaterialfv(GL_FRONT,GL_SPECULAR,specular);
glMateriali(GL_FRONT,GL_SHININESS,shine);
```

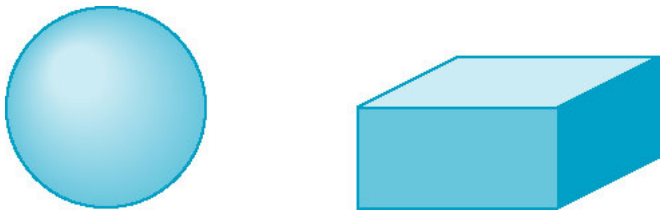
材料属性设置的Tutor

- Nate Robin's lightmaterial tutor

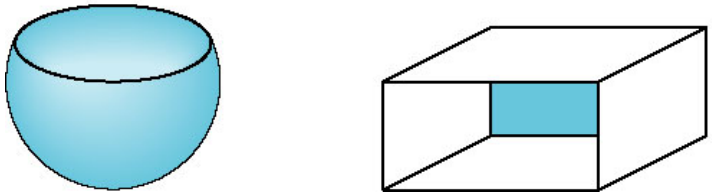


前面与后面

- 默认状态下只是把对象的前面进行明暗处理，对于凸对象这种处理结果是正确的
- 如果设置进行两面光照，那么OpenGL就会对曲面的双面进行明暗处理
- 每一面都可以具有自己的属性，面是用在glMaterialf()中的GL_FRONT, GL_BACK, 或者GL_FRONT_AND_BACK指定的



后面是不可见的



后面是可见的

发射项

- 在OpenGL中可以用材料的发射项来模拟一个光源
- 该项的颜色不受任何其它光源或者变换的影响

```
GLfloat emission[]={0.0,0.3,0.3,1.0);
glMaterialfv(GL_FRONT,GL_EMISSION, emission);
```

用颜色指定代替材料属性指定

- 通常当启用光照进行明暗处理后，原来的glColor*()命令失去原有的作用
- 如果调用了glEnable(GL_COLOR_MATERIAL)，那么就会使光照模型中的几种光根据glColor*()中的指定确定颜色：

```
glColorMaterial(GLenum face, GLenum mode);
```

其中face的取值GL_FRONT, GL_BACK与GL_FRONT_AND_BACK(默认值)
mode的取值为GL_EMISSION, GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR与GL_AMBIENT_AND_DIFFUSE(默认值)

透明效果

- 材料属性是利用RGBA值指定的
- A值用来使表面透明
- 默认状态下不管A的值是多少，所有表面都是不透明的
- 后面我们会讲到如何激活融合(blending)功能以及这种功能的应用

效率

- 由于材料属性为状态的一部分，因此如果对许多表面采用大量的不同材料，那么系统的行为就会大打折扣
- 可以通过定义一个材料结构，利用它在初始化时设置所有材料，从而净化代码

```
typedef struct materialStruct {
  GLfloat ambient[4];
  GLfloat diffuse[4];
  GLfloat specular[4];
  GLfloat shineness;
} MaterialStruct;
```

- 这样可以通过指针选择一种材料

多边形的明暗处理

- 对每个顶点进行明暗处理的计算
  - 顶点的颜色变为顶点的明暗效果
- 默认状态下，多边形内部的颜色是顶点颜色的线性插值
  - `glShadeModel(GL_SMOOTH);`
- 如果调用了`glShadeModel(GL_FLAT);`那么第一个顶点的颜色确定整个多边形的颜色

多边形的法向

- 多边形具有单个法向量
  - 应用简单光照模型在每个顶点处的明暗处理计算几乎完全相同
  - 对于无限远视点(默认)或者没有镜面光时，所有多边形的颜色一样
- 考虑球的模型
- 希望在每个顶点处具有不同的法向
  - 这个概念在数学上不太“正确”

光滑明暗处理

- 在每个顶点处设置新的法向
- 对于球的模型，这很容易做到
  - 如果球心在原点，那么n = p
- 现在进行光滑明暗处理
- 注意轮廓线处的结果

Q&A