光照明模型（中）：模型的实现

华中科技大学软件学院　万琳

提纲

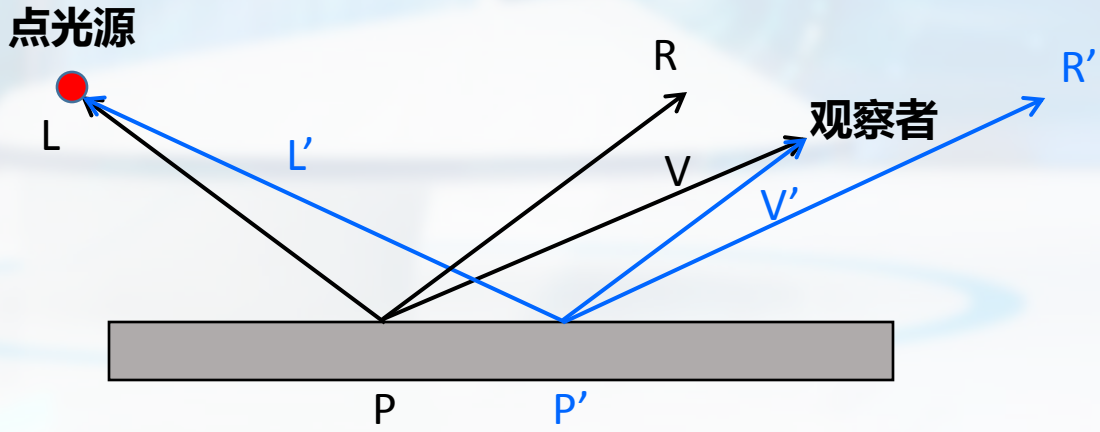
1. Phong模型的实现
2. OpenGL中的实现

1 Phong模型的实现

◆近似处理

$$I=I_aK_a+I_pK_d(L\cdot N)+I_pK_s(R\cdot V)^n$$

- 对物体表面上的每个点P（或者P’ ），均需计算光线的反射方向R（或者R’ ）
- 对物体表面上的每个点P（或者P’ ），均需计算观察方向V（或者V’ ）



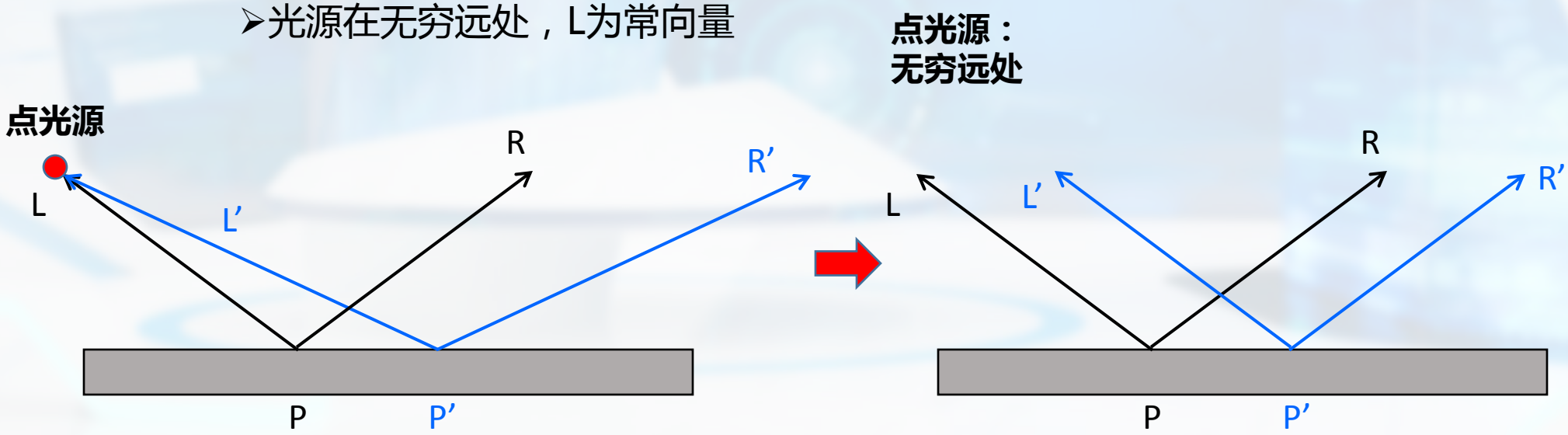
1 Phong模型的实现

◆近似处理

$$I=I_aK_a+I_pK_d(L\cdot N)+I_pK_s(R\cdot V)^n$$

为了减少计算量，假设：

- 光源在无穷远处，L为常向量



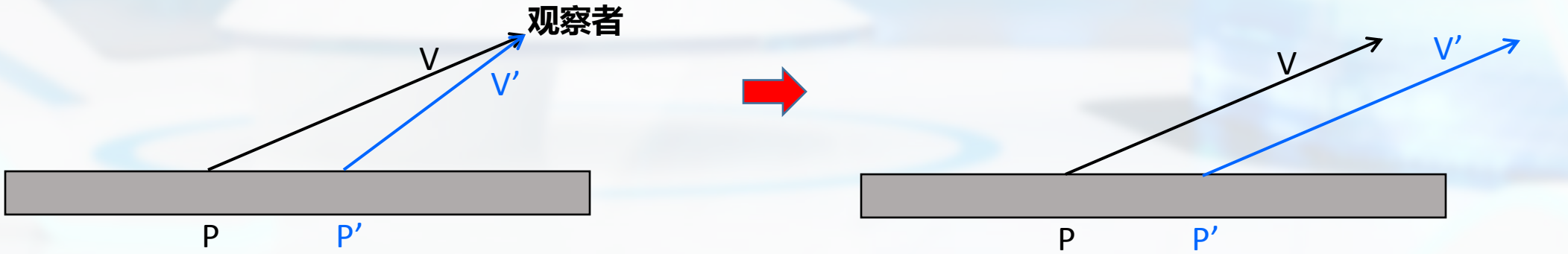
1 Phong模型的实现

◆近似处理

$$I=I_aK_a+I_pK_d(L\cdot N)+I_pK_s(R\cdot V)^n$$

为了减少计算量，假设：

- 视点在无穷远处，V为常向量



1 Phong模型的实现

◆近似处理

$$I=I_aK_a+I_pK_d(L\cdot N)+I_pK_s(R\cdot V)^n$$

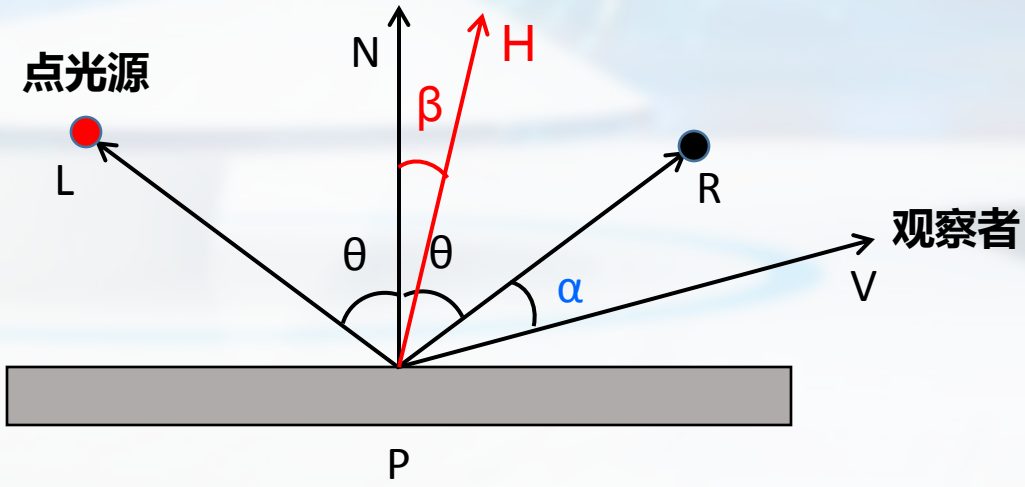
为了减少计算量，假设：

- H为L与V的二分向量H=(L+V)/2
- 用（H•N）近似（R•V）

节省计算时间的原因：
（1）在L和V为常量，对所有的点只需计算一次H的值
（2）H的计算比R的计算简单

$$I=I_aK_a+I_pK_d(L\cdot N)+I_pK_s(H\cdot N)^n$$

近似处理处理之后实际上是Blin-Phong模型

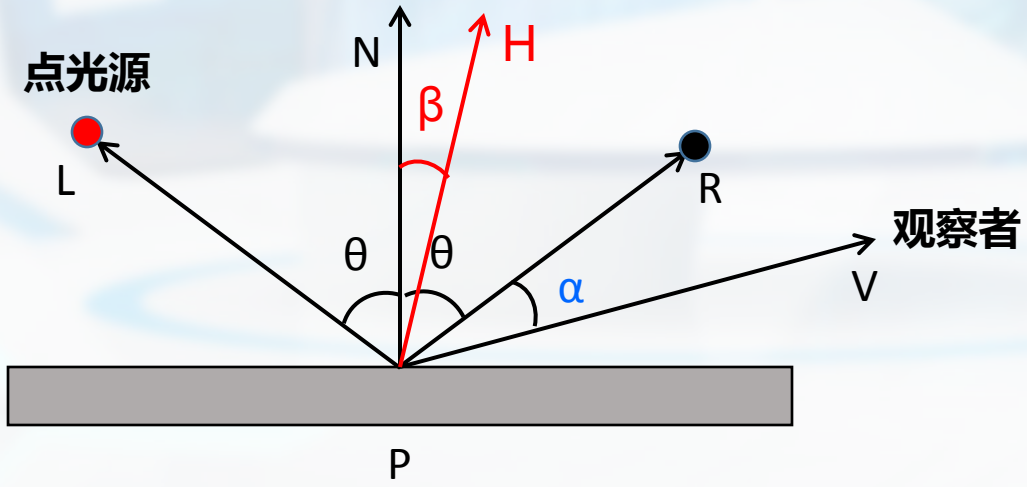


1 Phong模型的实现

◆近似处理

$$I=I_aK_a+I_pK_d(L\cdot N)+I_pK_s(R\cdot V)^n$$

$$I=I_aK_a+I_pK_d(L\cdot N)+I_pK_s(H\cdot N)^n$$



β为H和N之间的夹角

α为R和V之间的夹角

显然：

$\theta+\beta=(2\theta+\alpha)/2$

于是：$\beta=\alpha/2$

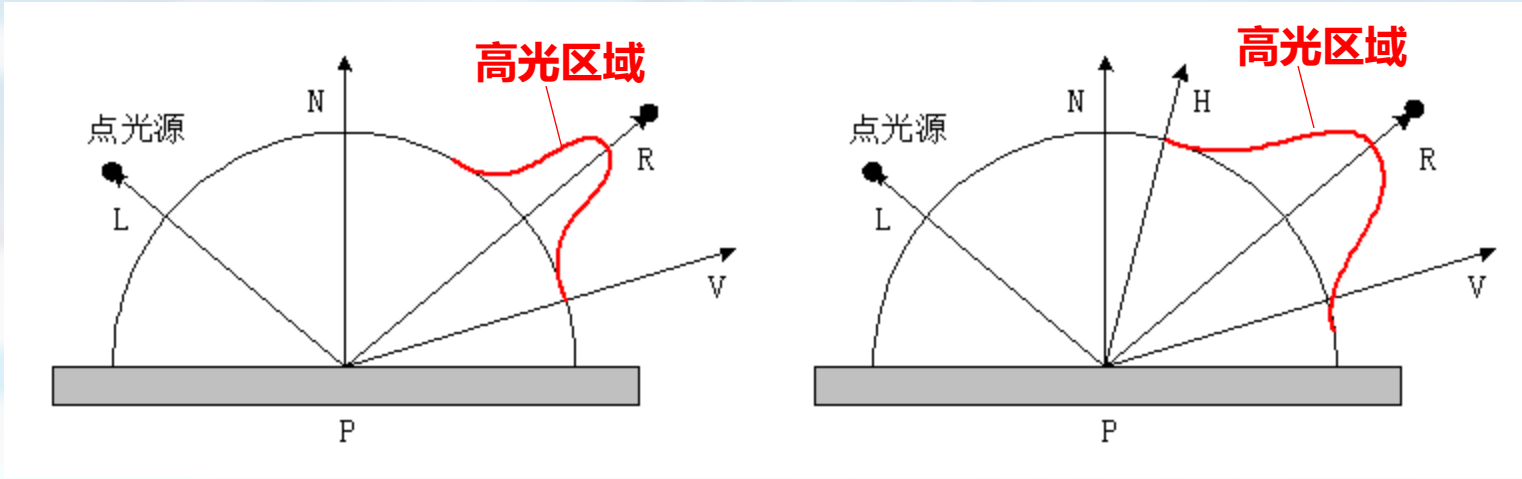
1 Phong模型的实现

◆近似处理

$$I=I_aK_a+I_pK_d(L\cdot N)+I_pK_s(R\cdot V)^n$$

$$I=I_aK_a+I_pK_d(L\cdot N)+I_pK_s(H\cdot N)^n$$

通过修正n来补偿



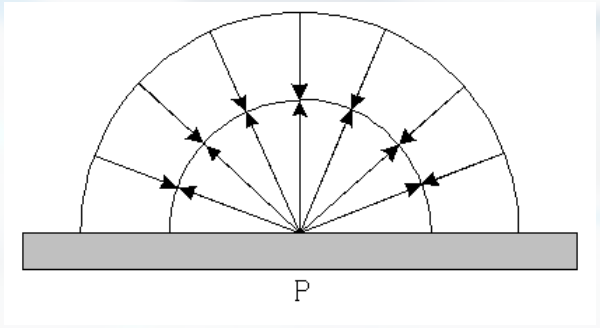
近似计算后的高光区域更大

1 Phong模型的实现

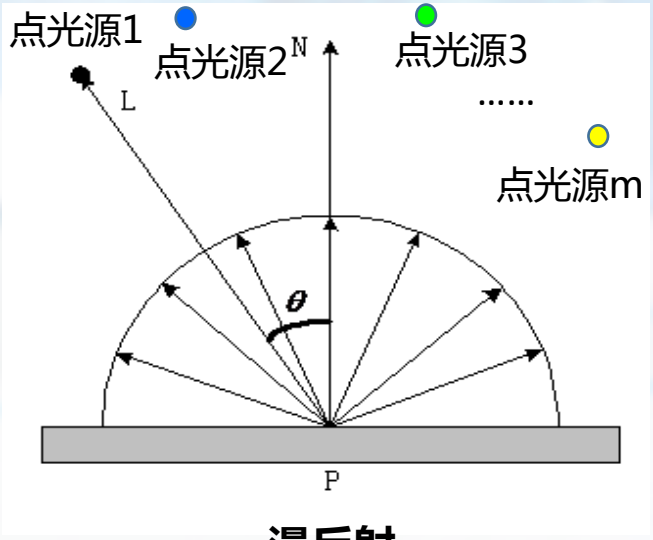
◆**多个光源**（m个点光源）

光强计算假定只有一个点光源，若在场景中有m个点光源，则可以在任一P点上叠加各个光源所产生的光照效果：

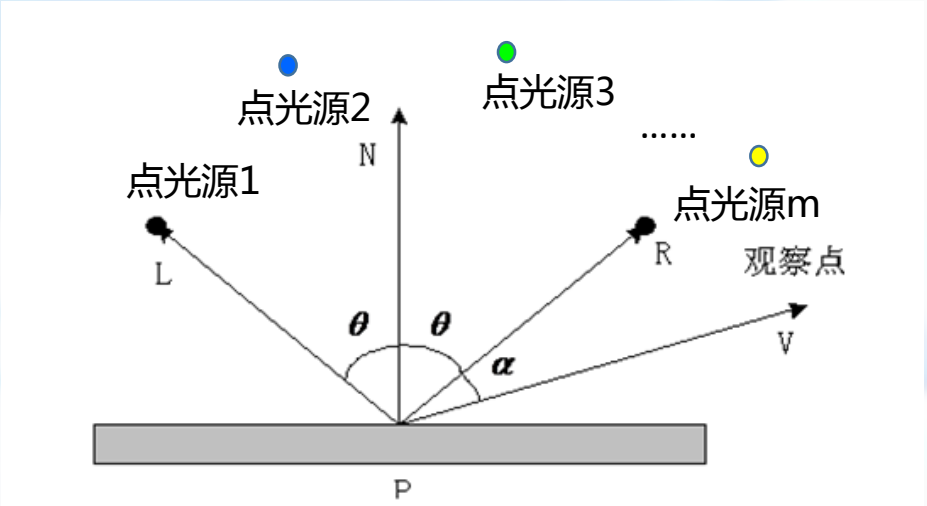
$$I=I_aK_a+\sum_{i=1}^{m}I_{p,i}K_d(L_i\cdot N)+\sum_{i=1}^{m}I_{p,i}K_s(H_i\cdot N)^n$$



环境光
Ambient



漫反射
Diffuse

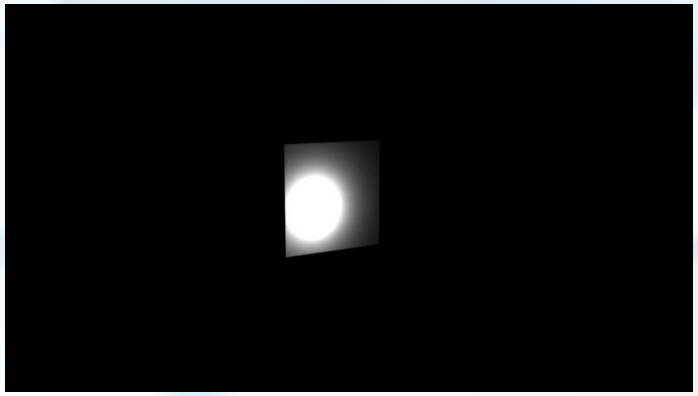


镜面反射
Specular

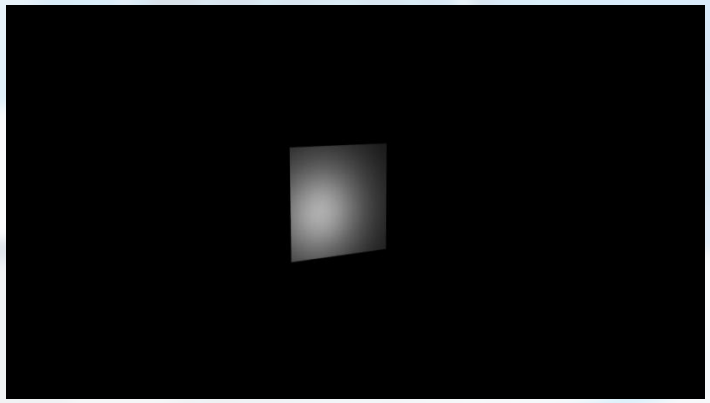
1 Phong模型的实现

◆光的衰减

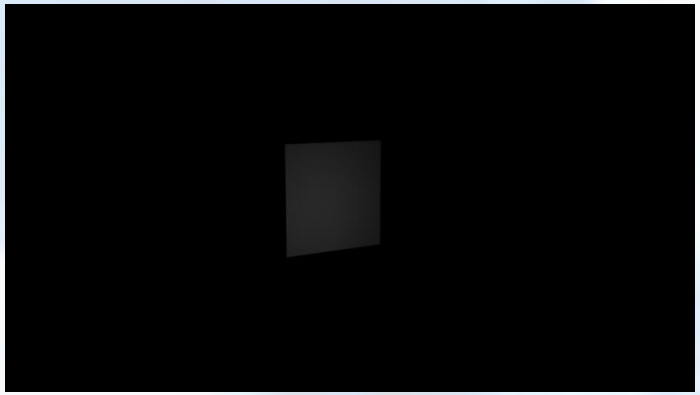
在同一光源的照射下，距光源比较近的景物看起来会亮一些，而距光源较远的景物看起来会暗一些，这是因为光在传播的过程中，其能量会发生衰减。



光源近



光源较远



光源远

1 Phong模型的实现

◆光的衰减

在同一光源的照射下，距光源比较近的景物看起来会亮一些，而距光源较远的景物看起来会暗一些，这是因为光在传播的过程中，其能量会发生衰减。

令距离为d，最简单的衰减因子为$1/d^2$

- d很小时， $1/d^2$产生过大的强度变化
- d很大时， $1/d^2$变化太小

1 Phong模型的实现

◆光的衰减

在同一光源的照射下，距光源比较近的景物看起来会亮一些，而距光源较远的景物看起来会暗一些，这是因为光在传播的过程中，其能量会发生衰减。

令距离为d，最简单的衰减因子为$1/d^2$

实际上采用一个与d相关的函数 $f(d)=\min(1, \frac{1}{c_0+c_1d+c_2d^2})$

1 Phong模型的实现

◆光的衰减

在同一光源的照射下，距光源比较近的景物看起来会亮一些，而距光源较远的景物看起来会暗一些，这是因为光在传播的过程中，其能量会发生衰减。

令距离为d，衰减因子实际上采用一个与d相关的函数 $f(d)=\min(1, \frac{1}{c_0+c_1d+c_2d^2})$

$$I=I_aK_a+\sum_{i=1}^{m}I_{p,i}K_d(L_i\cdot N)+\sum_{i=1}^{m}I_{p,i}K_s(H_i\cdot N)^n$$

$$I=I_aK_a+\sum_{i=1}^{m}f(d_i)I_{p,i}K_d(L_i\cdot N)+\sum_{i=1}^{m}f(d_i)I_{p,i}K_s(H_i\cdot N)^n$$

1 Phong模型的实现

◆Phong光照明模型的RGB颜色模型形式

假定选择RGB颜色模型

- 环境光的强度可以表示为： $I_a = (I_{aR}, I_{aG}, I_{aB})$

  入射光的光强I_p可以表示为：$I_p = (I_{pR}, I_{pG}, I_{pB})$

- 环境光的反射系数K_a可以表示为： $K_a = (K_{aR}, K_{aG}, K_{aB})$

  漫反射的反射系数K_d可以表示为： $K_d = (K_{dR}, K_{dG}, K_{dB})$

  镜面反射的反射系数K_s可以表示为：$K_s = (K_{sR}, K_{sG}, K_{sB})$

1 Phong模型的实现

◆Phong光照明模型的RGB颜色模型形式

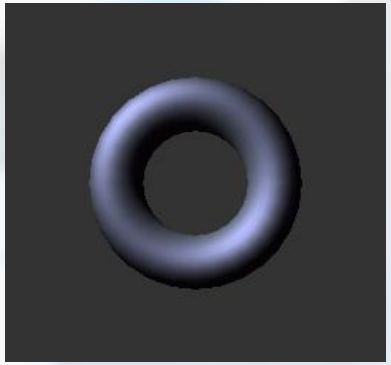
$$I_R = I_{aR}K_{aR} + \sum_{i=1}^{m} f(d_i) I_{pR,i} K_{dR} (L_i \cdot N) + \sum_{i=1}^{m} f(d_i) I_{pR,i} K_{sR} (H_i \cdot N)^n$$

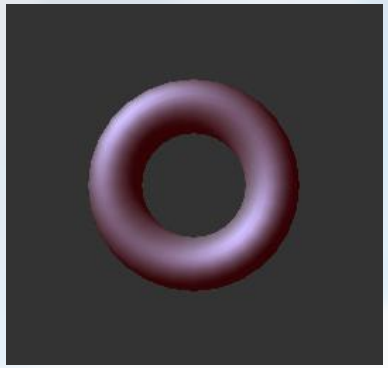
$$I_G = I_{aG}K_{aG} + \sum_{i=1}^{m} f(d_i) I_{pG,i} K_{dG} (L_i \cdot N) + \sum_{i=1}^{m} f(d_i) I_{pG,i} K_{sG} (H_i \cdot N)^n$$

$$I_B = I_{aB}K_{aB} + \sum_{i=1}^{m} f(d_i) I_{pB,i} K_{dB} (L_i \cdot N) + \sum_{i=1}^{m} f(d_i) I_{pB,i} K_{sB} (H_i \cdot N)^n$$

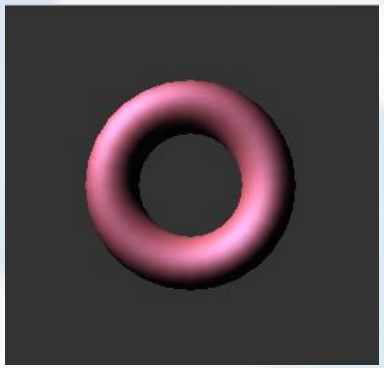
1 Phong模型的实现

◆Phong光照明模型的RGB颜色模型形式

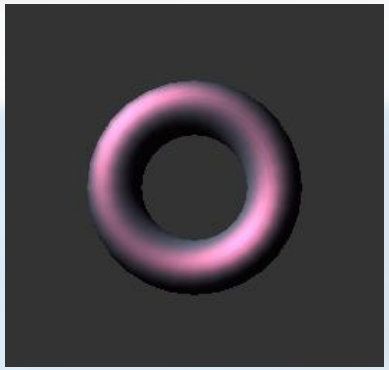




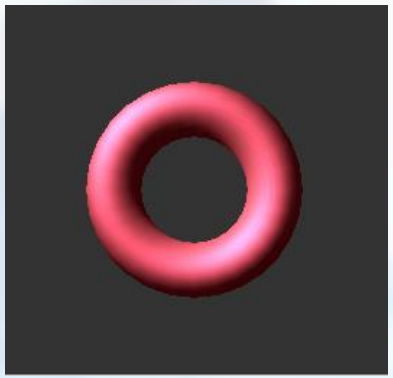
只提高环境光的红色分量反射系数



只提高漫反射光的红色分量反射系数



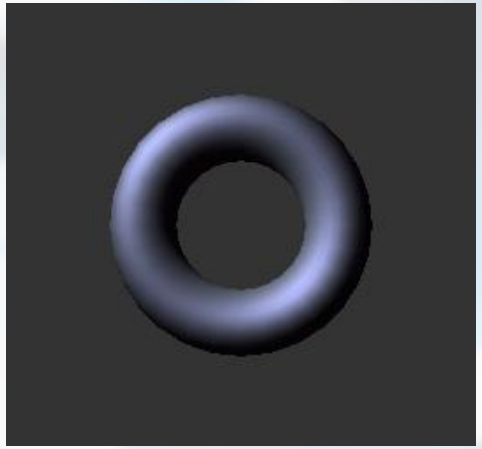
只提高镜面反射光的红色分量反射系数

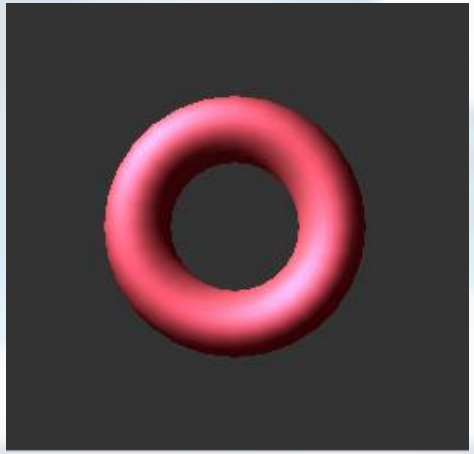


提高对三种光的红色分量反射系数

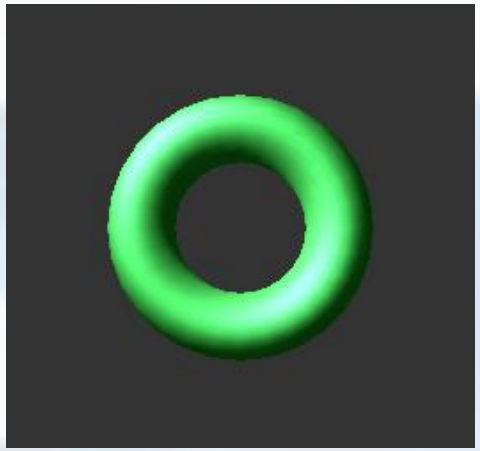
1 Phong模型的实现

◆Phong光照明模型的RGB颜色模型形式

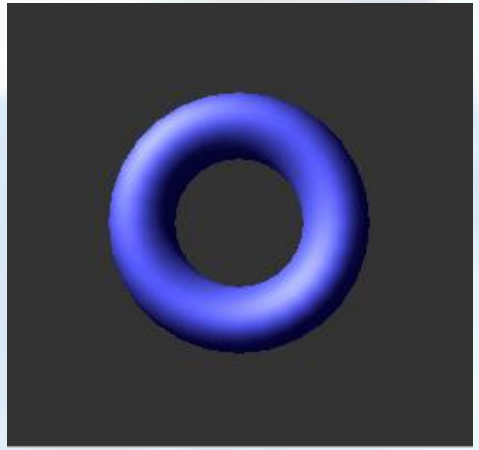




提高对三种光的红色分量反射系数



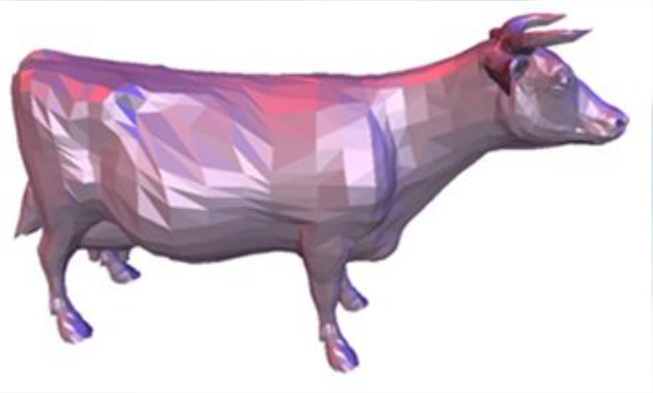
提高对三种光的绿色分量反射系数

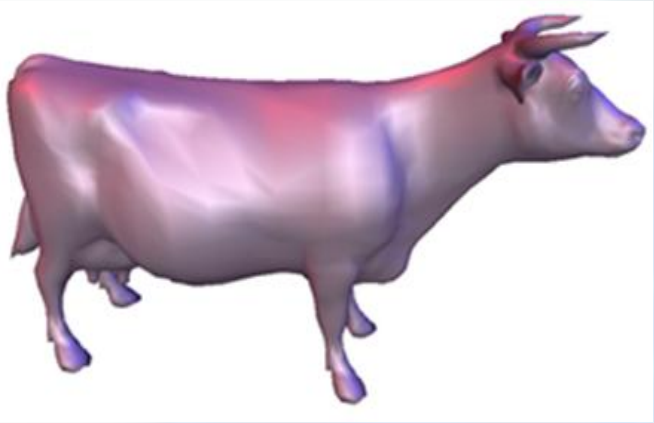


提高对三种光的蓝色分量反射系数

1 Phong模型的实现

◆再谈马赫带效应

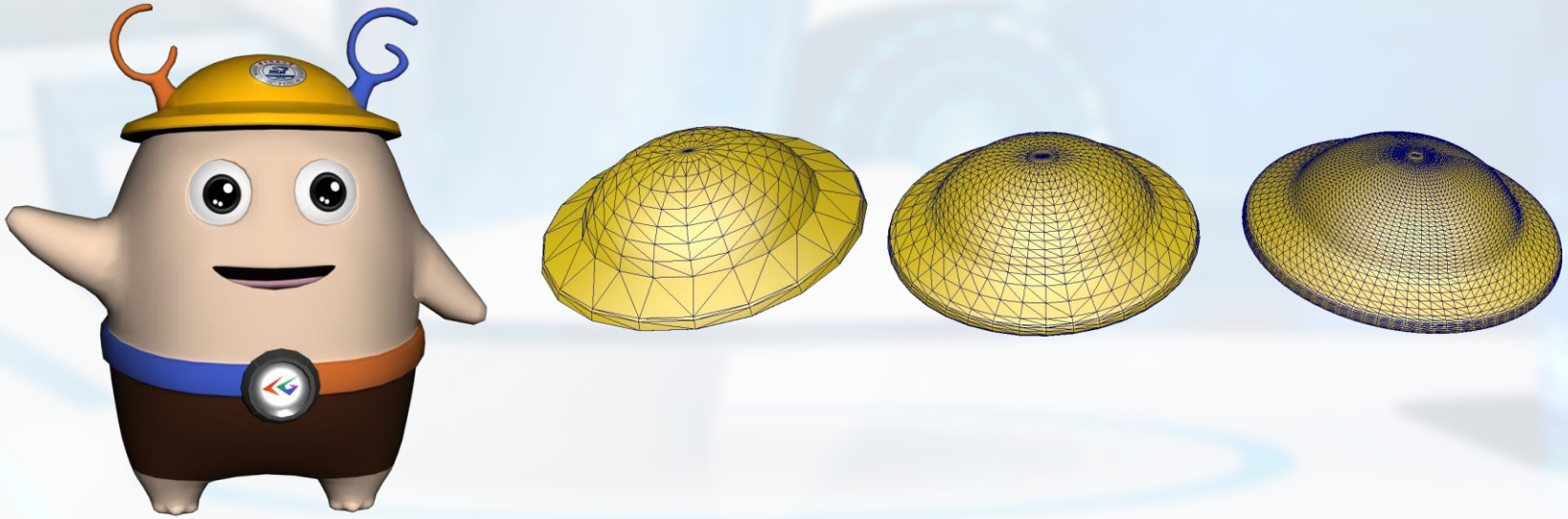




1 Phong模型的实现

◆如何消除或者减弱马赫带效应

方法一：多边形细分

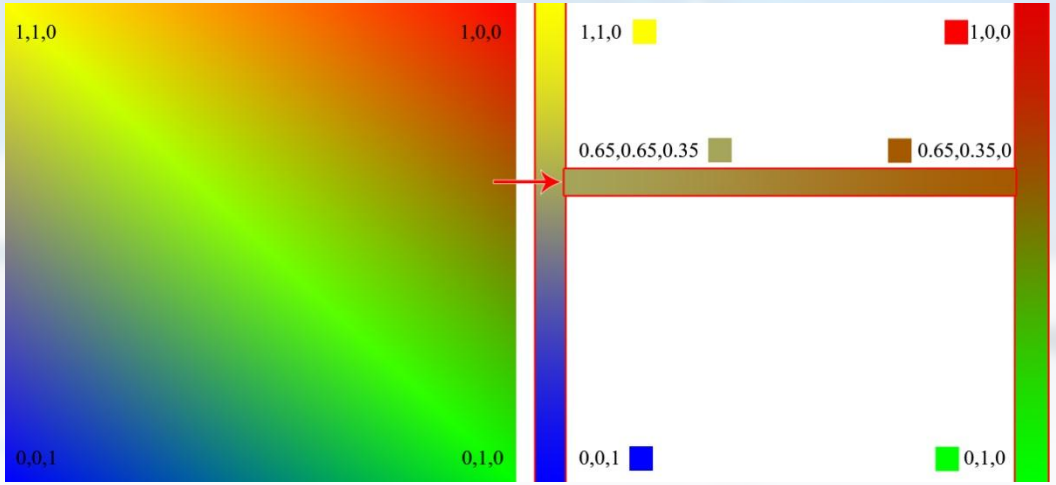


1 Phong模型的实现

◆如何消除或者减弱马赫带效应

方法二：明暗处理 ***Gouraud明暗处理***

多边形内部各点颜色的计算：对顶点颜色双线性插值

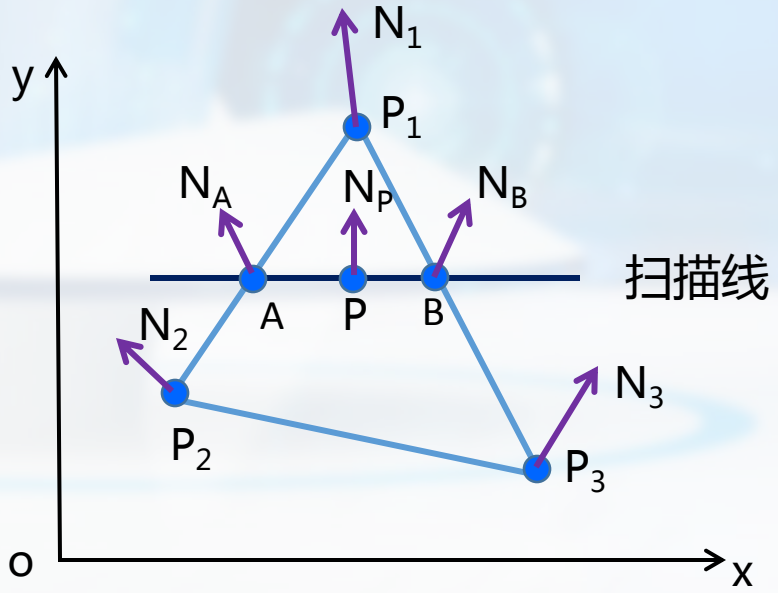


1 Phong模型的实现

◆如何消除或者减弱马赫带效应

方法二：明暗处理 ***Phong明暗处理***

多边形内部各点法矢量的获得：对顶点法矢量双线性插值

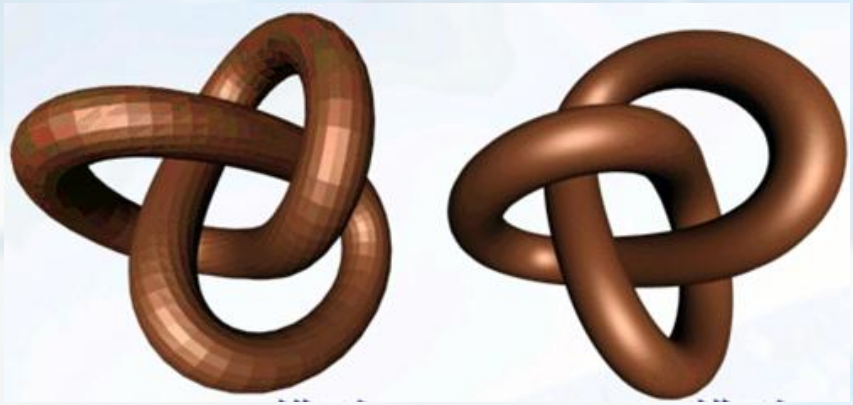


1 Phong模型的实现

◆如何消除或者减弱马赫带效应

方法二：明暗处理

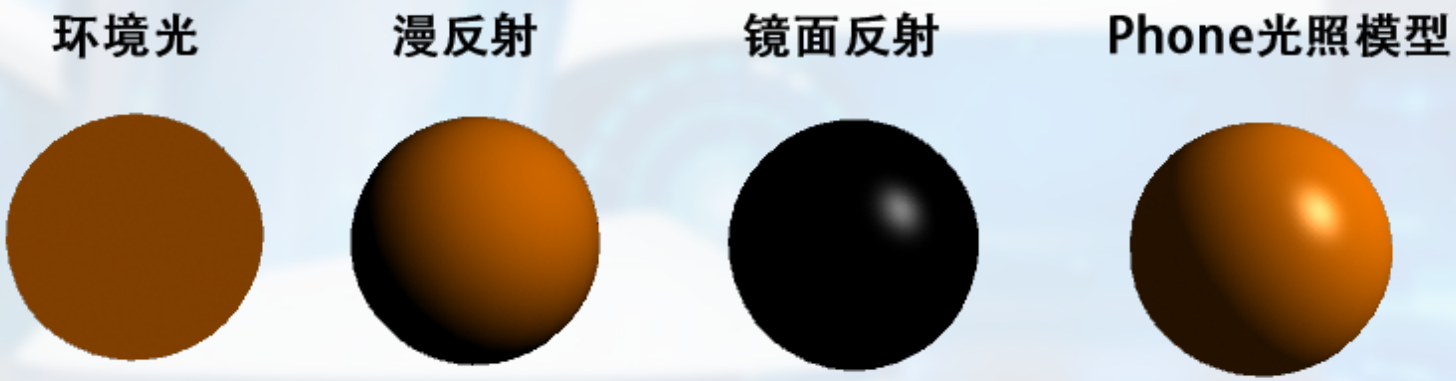
Phong明暗处理效果好于Gouraud明暗处理，但是计算量也更大



Gouraud明暗处理　　Phong明暗处理

2 OpenGL中的实现

◆Phong模型

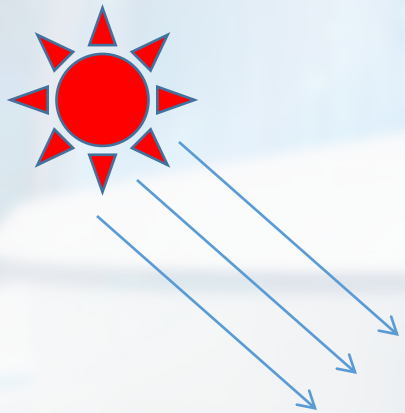


2 OpenGL中的实现

◆Phong模型

基于Phong模型模拟平行光、点光源和聚光灯效果：

➢平行光：不加衰减因子



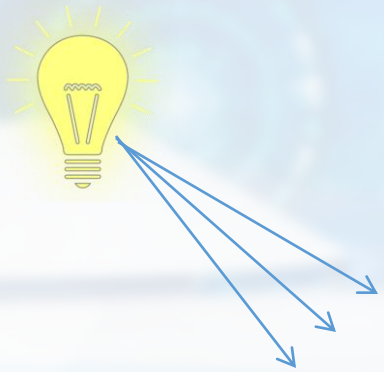
无穷远处照射过来
可以看作是平行光
而且没有衰减

2 OpenGL中的实现

◆Phong模型

基于Phong模型模拟平行光、点光源和聚光灯效果：

- 平行光：不加衰减因子
- 点光源：保留衰减因子



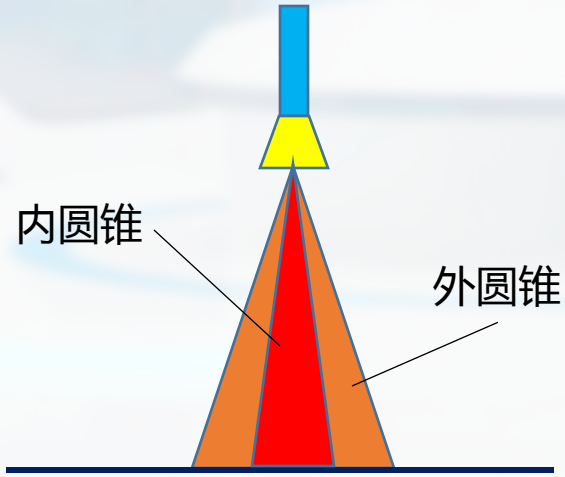
有穷远处照射过来
有衰减

2 OpenGL中的实现

◆Phong模型

基于Phong模型模拟平行光、点光源和聚光灯效果：

- 平行光：不加衰减因子
- 点光源：保留衰减因子
- 聚光灯：光锥的内圆锥和外圆锥



2 OpenGL中的实现

◆Phong模型

➢平行光：不加衰减因子

```
// 计算定向光
vec3 CalcDirLight(DirLight light, vec3 normal, vec3 viewDir)
{
    if(!light.on) {
        return vec3(0.0);
    }
    // 漫反射
    vec3 lightDir = normalize(-light.direction);
    float diff = max(dot(normal, lightDir), 0.0);
    // 镜面反射
    vec3 reflectDir = reflect(-lightDir, normal);
    float spec = pow(max(dot(viewDir, reflectDir), 0.0), material.shininess);
    // 在漫反射光下物体颜色
    vec3 diffuseColor = vec3(material.diffuse);
    // 计算环境光，漫反射光和镜面光
    vec3 ambient = light.ambient * diffuseColor;
    vec3 diffuse = light.diffuse * diff * diffuseColor;
    vec3 specular = light.specular * spec * vec3(material.specular);
    return ambient + diffuse + specular;
}
```

2 OpenGL中的实现

◆Phong模型

➢点光源：保留衰减因子

距离

```
// 距离和衰减
   float d = length(light.position - fragPos);
   float attenuation = 1.0 / (light.c + light.l * d + light.q * d * d);
```

衰减因子的计算

```
   ......

   return (ambient + diffuse + specular) * attenuation;
```

可以实现距离越近就会越亮，距离越远就越暗的效果

2 OpenGL中的实现

◆Phong模型

➢聚光灯：光锥的内外圆锥

```
// 聚光强度
  float theta = dot(lightDir, normalize(-light.direction));
  float epsilon = light.cutOff - light.outerCutOff;
  float intensity = clamp((theta - light.outerCutOff) / epsilon, 0.0, 1.0);

  ……

  return (ambient + (diffuse + specular) * intensity) * attenuation;
```

这里为了使聚光效果看起来边缘更加圆滑，我们需要模拟聚光有一个内圆锥和外圆锥，通过内外圆锥之间的余弦值差来计算聚光强度（intensity），clamp函数把聚光强度约束在了[0,1]之间。

漫反射和镜面反射需要乘以聚光强度

2 OpenGL中的实现

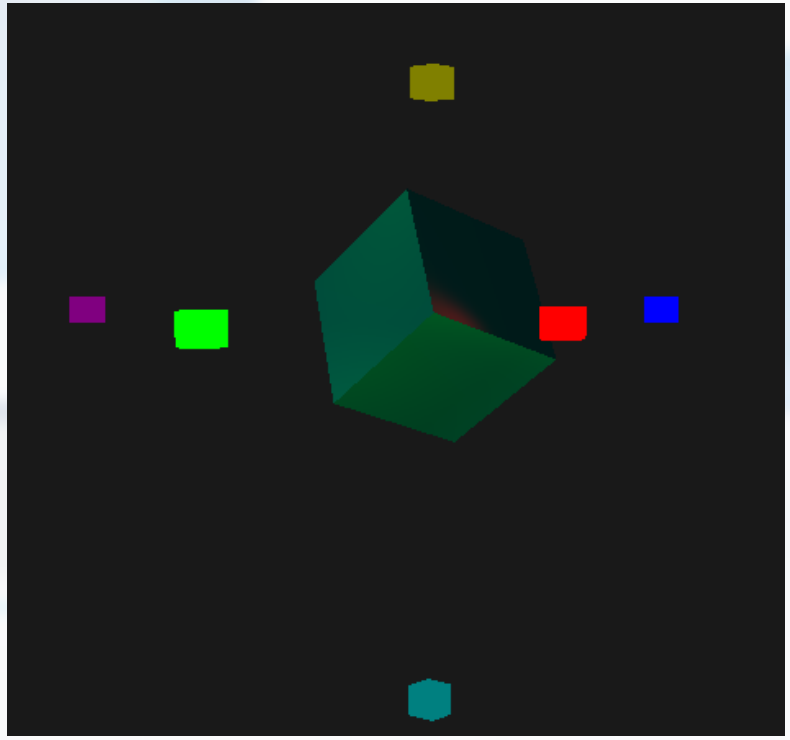
◆Phong模型

- ➢聚光灯：光锥的内外圆锥

2 OpenGL中的实现

◆实验

➢要求实现一个平行光、六个点光源和一个聚光灯




谢谢
软件学院　万琳