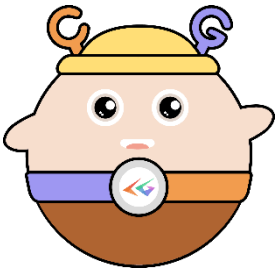
華中科技大學



计算机图形学课程

实验：Phong 模型

目录

1 Phong 模型

接下来，我们来介绍一下 Phong 模型。绘制效果如下，我们可以看到，中间的立方体是被照射的对象，有一个定向光源，六个点光源和一个聚光的光源。

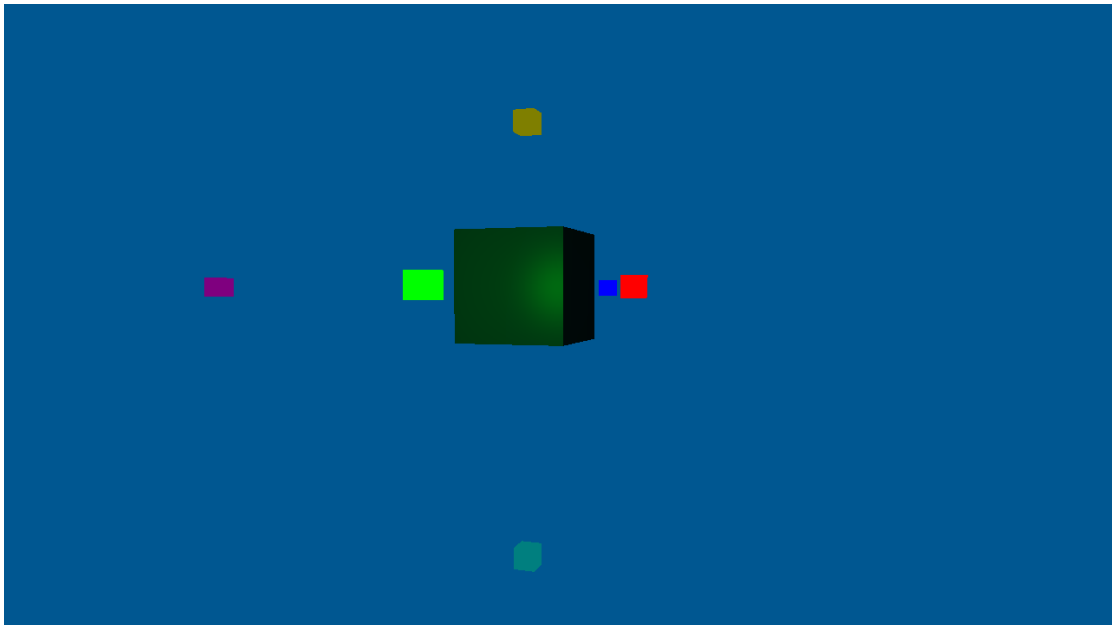


图 1 效果图

我们的程序流程主体还是跟前面的课程相同。因此我们这节的重点部分就是主要在这两个方面，一个是光源的设置，一个是光照计算。

1.1 光源设置

首先是光源的设置，我们这里的光源种类主要有三种，定向光，点光源和聚光，三种光源的效果叠加形成了我们刚才看到的效果。

那么我们在代码中是如何实现的呢？首先我们来看定向光，定向光就是类似太阳这种，它的属性包括定向光的方向，以及环境光，漫反射光，镜面反射光的强度参数。由于定向光是光源处于无限远处的平行光，因此我们无需指定出定向光光源具体的位置，只需指定其指向的方向即可。

接下来是点光源，我们定义了六个点光源，点光源就是类似灯泡这种，每一个点光源的属性都包括点光源的位置，环境光，漫反射光，和镜面反射光的

强度参数，这里每一个分量都要乘一个点光源的颜色。那么除此之外点光源与定向光有哪些不同呢？它与定向光不同的是它多了三个属性参数，这三个参数是用来计算衰减公式的三个系数，分别是 constant 常数项，linear 一次项和 quadratic 二次项，它会使得光线强度随距离的增加不断减小并且衰减的幅度也逐渐减小，这样更接近现实生活中点光源的效果。

我们的第三种光源是聚光，聚光就是类似手电筒的这种效果，聚光的属性也包括聚光光源的位置，方向，这里我们是用摄像机的位置和朝向来指定的，还有环境光，漫反射光，和镜面反射光的强度参数以及三个衰减系数，那么除此之外聚光光源与点光源又有哪些不同呢？这里我们又多了两个参数分别表示我们聚光内外圆锥的内外切光角。聚光的效果就相当于是一个圆锥的光效，我们这里通过使用两个圆锥来使我们的聚光效果看起来更加平滑。

当我们已经知道了我们是如何设置光源的，并且已经了解了三种光源的属性时，接下来我们要做的就是计算光源的光照，这里我们使用的是 Phong 模型来计算光照。

2 光照计算

具体计算光照的过程，我们是在立方体的片段着色器中进行的。由于我们在考虑物体颜色时才会考虑光照到物体产生的颜色影响，因此我们在立方体的片段着色器中计算光照。

2.1 定向光

接下来就是具体的计算过程，首先是计算定向光源，根据之前 Phong 模型的公式，我们分别计算环境光、漫反射光和镜面光。

我们在计算环境光的时候用的是漫反射光下的物体颜色而非环境光下的，是因为通常情况下漫反射光和环境光下的物体效果几近相同，因此我们无需存取两遍。

计算漫反射时我们对法向量和光线方向向量进行点乘，计算光源对当前片段实际的漫发射影响，即 L • N，结果值再乘以物体的材质颜色 Kd 和漫反射光因子 Ip。

```
// 漫反射
vec3 lightDir = normalize(-light.direction);
float diff = max(dot(normal, lightDir), 0.0);
```

计算镜面反射时我们对视线方向与反射方向的点乘（R • V）（并确保它不是负值），然后取它的 n 次幂。这个 n 是高光的反光度(Shininess)，结果值再乘以物体的材质颜色 Ks 和镜面光因子 Ip。

```
// 镜面反射
vec3 reflectDir = reflect(-lightDir, normal);
float spec = pow(max(dot(viewDir, reflectDir), 0.0),
material.shininess);
```

这里的高光的反光度 n(shininess)我们程序中取值为 32，那么为什么我们取 32 而不取其他的值呢？之所以选取这个值是因为我们不希望镜面光过于显眼，我们可以看到图中 n 取值不同时我们得出的光源效果，也可以自己尝试一下。

2.2 点光源

第二种光源效果是点光源，这里我们对于点光源的环境光、漫反射光和镜面光的计算同定向光相同，依然是根据 Phong 模型来计算，只不过我们引入了衰减这一属性。衰减与光源和物体的距离有关，它会使得光线强度随距离的增加不断减小并且衰减的幅度也逐渐减小。这里我们保留环境光，只考虑漫反射光和镜面光受到衰减的影响。

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

2.3 聚光

第三种光源效果是聚光光源，这里对于环境光、漫反射光和镜面光以及衰减的计算同点光源相同，不同点在于这里增加了聚光强度这一属性。

theta 就是光线方向同光源方向的夹角。

Epsilon 是内外圆锥之间的余弦差值，这里我们给出两个切光角，分别形成内外圆锥。

Intensity 则是我们最终得到的聚光强度，它通过 clamp()函数将 theta 值做了限制，在内圆锥内，强度大于 1.0，在内外圆锥之间，强度在 0.0-1.0 之间，在外圆锥外，强度为负值。

```
// 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);
```

我们同样保留环境光，对漫反射光和镜面反射增加聚光强度，就可以得到平滑的聚光效果。

最后我们对程序进行一个演示，我们这里设置了键盘响应事件，按 1 开启定向光，2~7 开启点光源，8 开启聚光效果。

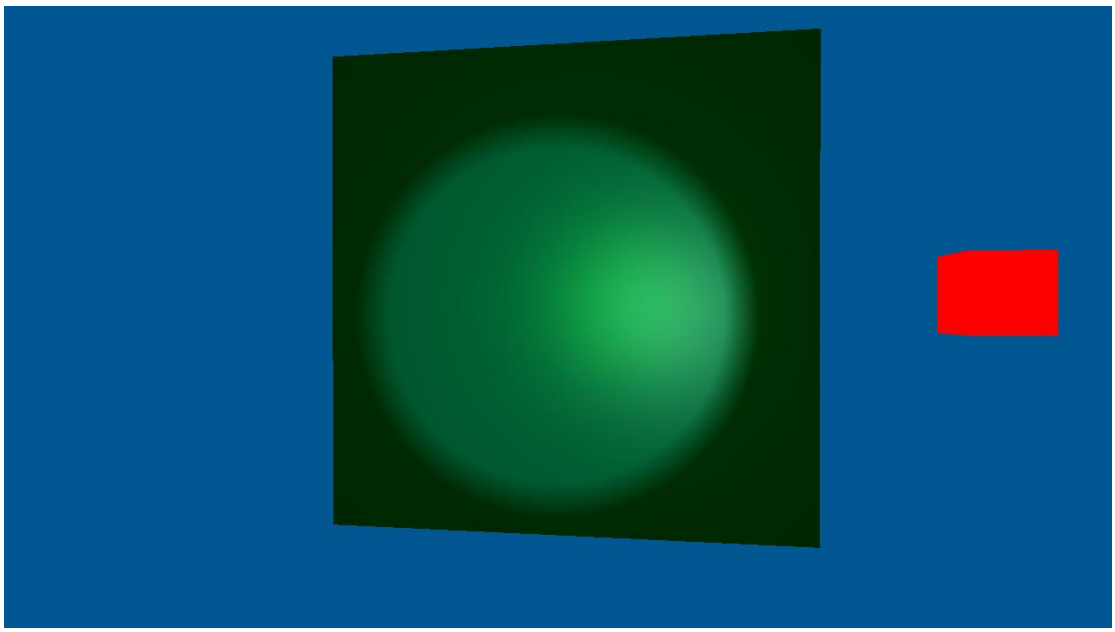


图 2.3 聚光效果图