实验3.4 Phong反射模型（2）

1. 实验内容
2. 了解OpenGL中基本的光照模型
3. 掌握OpenGL中实现基于片元的光照计算（Phong着色）
4. 理论背景

程序中有三个地方可以执行光照计算：在OpenGL应用程序代码中、在顶点着色器中或者在片元着色器中。无论在何处执行光照计算，都使用相同的基本光照模型，主要区别是在于绘制的效率和外观。

如果在顶点着色器中执行光照计算，那么在光栅化模块中将对顶点的颜色进行插值计算从而得到每个片元的颜色。而本周主要内容是基于片元的光照计算，区别在于片元着色器的输入数据不是顶点着色器计算之后的每个顶点的颜色，而是每个顶点的法向量，在渲染过程中对法向量进行插值而计算片元的颜色，从而输出到光栅化模块中。需要注意的是，输入的每个顶点必须是经过模-视变换和投影变换之后位于裁剪体内部的顶点，因为只有这些顶点最后在光栅化模块中才能得到渲染并显示在绘制窗口中。

如下图所示，左边是采用逐顶点的光照计算的效果，右边采用逐片元光照计算的效果，从结果上来看，因为逐片元计算会对三维物体表面法向量进行插值，从而使得物体表面的法向量场更加均匀，所以产生的光照效果也更加均匀。

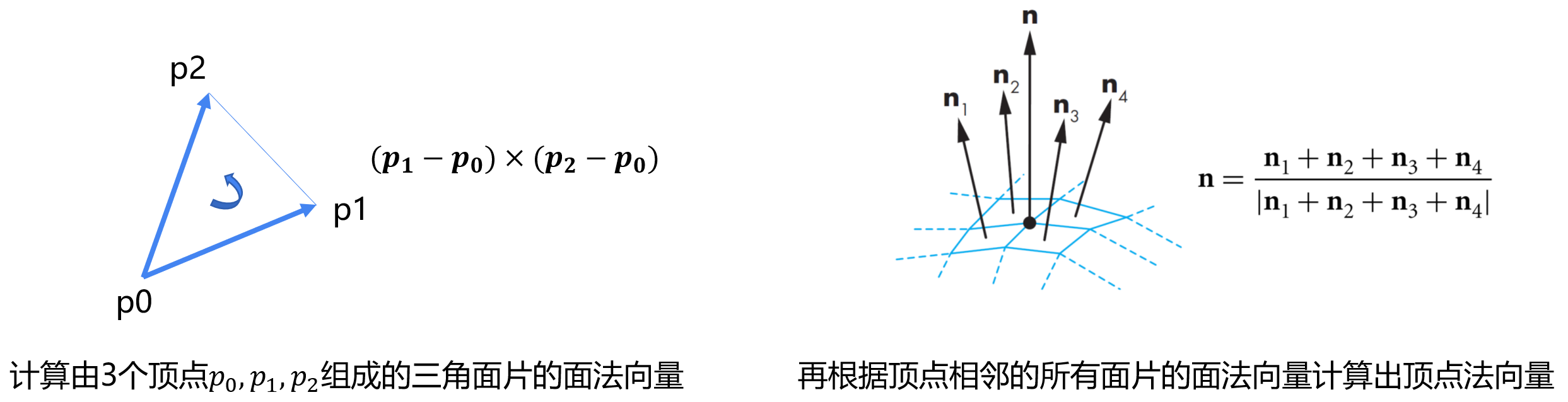


1. 实验内容

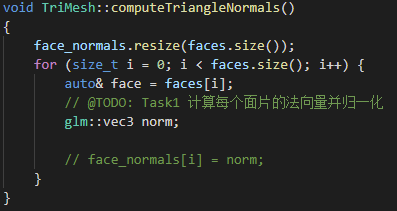
本次实验与3.3实验的内容基本一样，主要区别在于光照的计算，3.3在顶点着色器中计算，本次是在片元着色器中实现。

1. 法向量计算

法向量对光照计算有着重要作用，如果法向量计算有误，会造成许多绘制结果的缺陷。Task1中我们使用球作为模型，球的法向量可以精确求解，而对于其他的模型，我们需要先计算面片的法向量，然后再通过面片法向量计算顶点的法向量均值。



* 1. 计算面片法向量：给定三角形的三个顶点，和，法向量为：

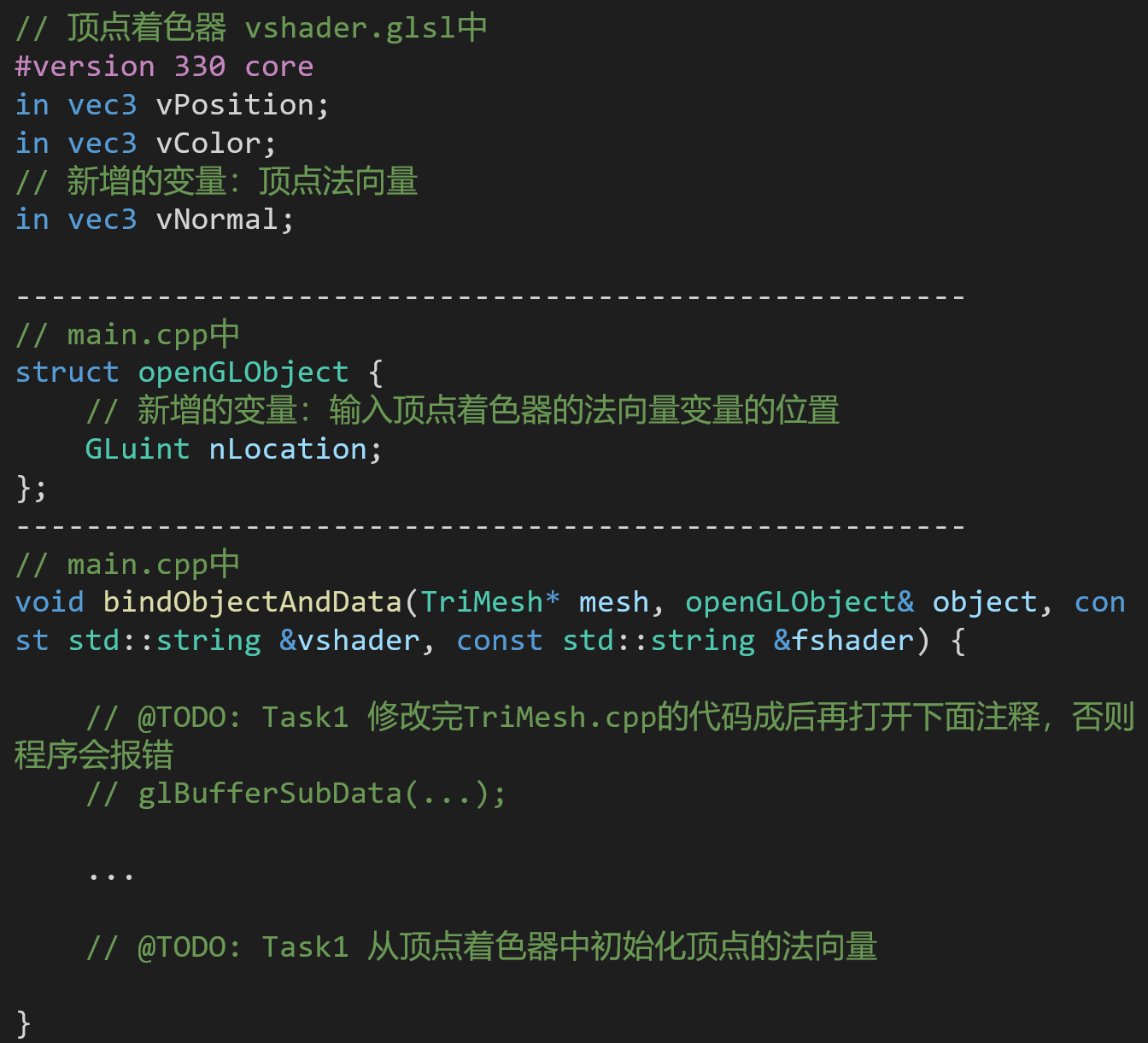


* 1. 计算顶点法向量：给定顶点所在面片的法向量，顶点的平均法向量为法向量的和。



上面的函数将会在storeFacesPoints函数中调用。

* 1. 我们计算好法向量后，和顶点坐标类似，我们需要将其数据传递给着色器，为此我们在顶点着色器文件内新加了一个法向量变量。同时我们给保存缓存对象的 openGLObject结构体内新加了一个 nLocation。参照顶点坐标vPosition的写法，在bindObjectAndData函数中将传递法向量的代码补全，注意BUFFER\_OFFSET的数值在bindObjectAndData函数中我们需要将法向量数据传递给着色器，具体写法参考vPosition的代码。



1. 在片元着色器中实现Phong光照模型

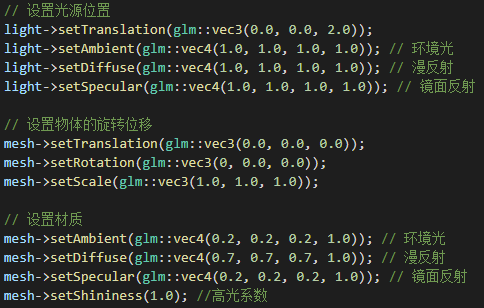
需要说明的是，在OpenGL绘制流水线中，光照计算需要等到三维物体经过相机变换和投影变换之后才能进行，因为此时物体在三维空间中的位置才完全确定下来。另外，通常情况下，我们在相机坐标系下来计算所有向量，因为在相机坐标系下，原点就是相机位置/眼睛位置。

具体实验过程如下：

1. 使用封装在TriMesh.h中的TriMesh类，读入球模型。



1. 设置光源，光源对象由光源位置环境光、漫反射光、镜面反射光组成。物体材质着由高光系数、环境光、漫反射光、镜面反射光参数组成。

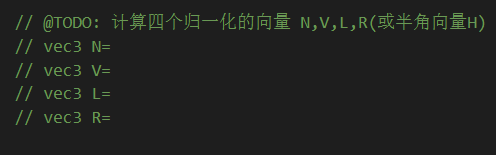


1. 为了传递这些材质数据，我们实现了一个bindLightAndMaterial函数用于传递数据给着色器。

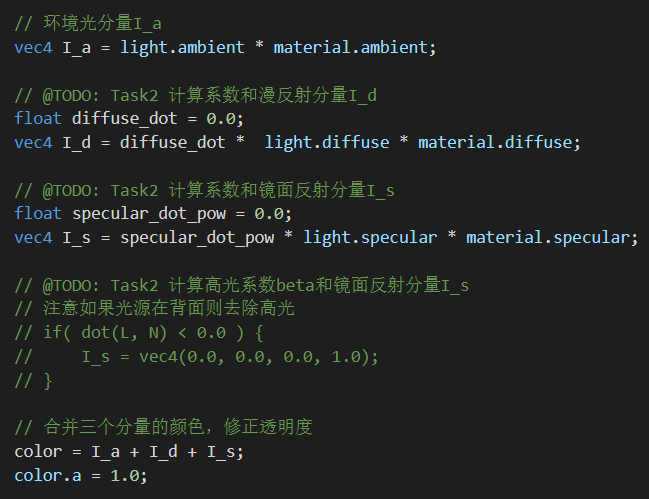


在顶点着色器中执行为每个顶点执行光照计算，为了简单考虑，我们这里假设衰减系数。

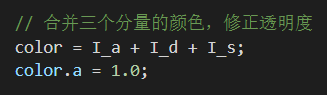
计算Phong反射模型涉及到的四个向量，并归一化，其中使用的normalize(a)函数和reflect(in,norm)函数均为GLSL语言内置函数，分别是归一化向量和依据入射向量和法向量计算反射向量。请根据图示计算N、L、V、R四个分量：



再依据公式，计算漫反射分量和镜面反射分量。代码中会使用到GLSL语言内置函数dot(a,b)，max(x,y)，作用分别是向量点积和取两者最大值，具体使用方法请参考相关文档。



最后为累加三个部分的颜色分量，得到每个顶点的颜色如下，颜色相加后最后一维的透明度需要进行修正。



1. 材质应用

曲面上某一点的颜色是光照和材质共同作用的结果，相同颜色的光照射在不同材质的物体上会得到不同的颜色，在代码中创建材质对象并应用到光照的计算中

设置材质对象，材质的属性有环境反射率、漫反射率、镜面反射率、（自发光系数）、高光系数等组成。可以参考这些网页提供的材质参数进行绘制。

<http://www.it.hiof.no/~borres/j3d/explain/light/p-materials.html>

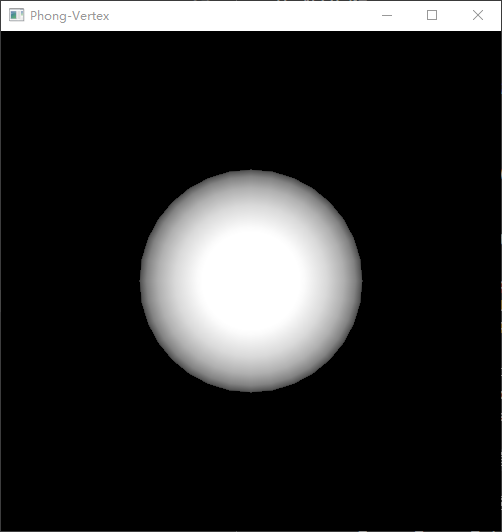
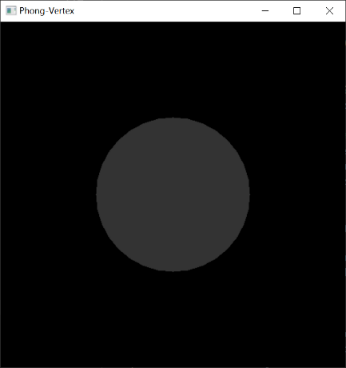
<http://devernay.free.fr/cours/opengl/materials.html>

1. 添加更多交互

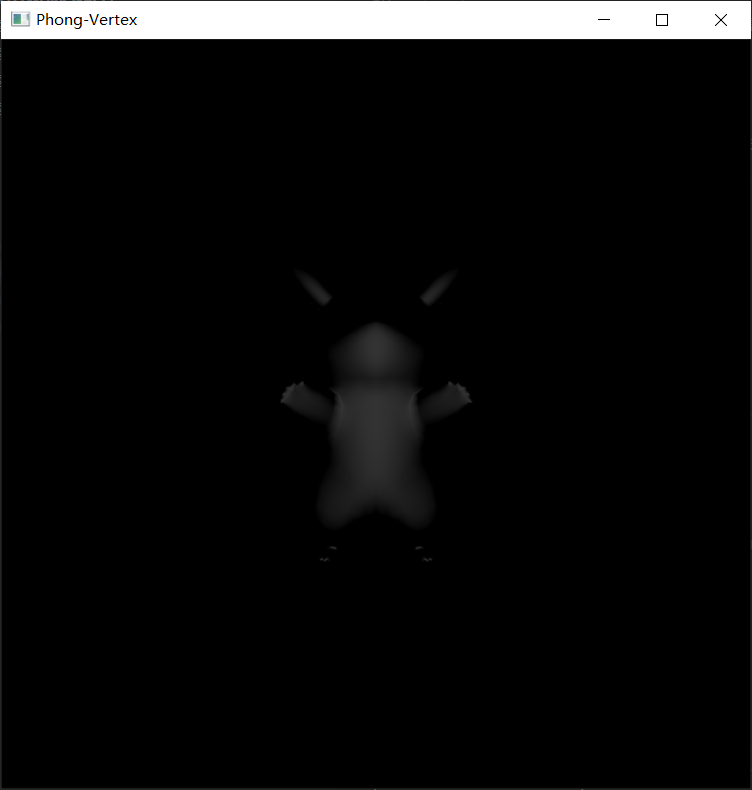
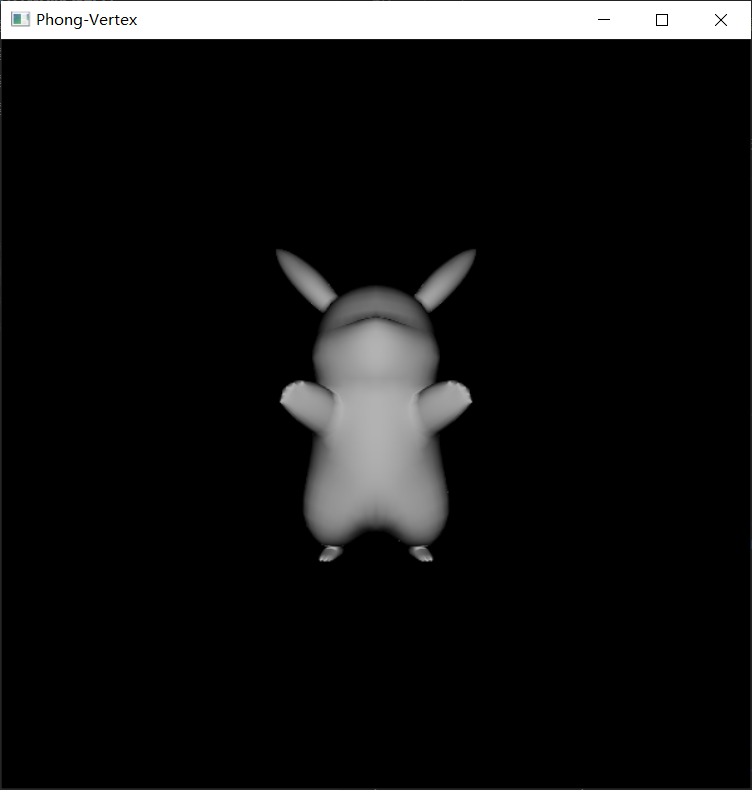
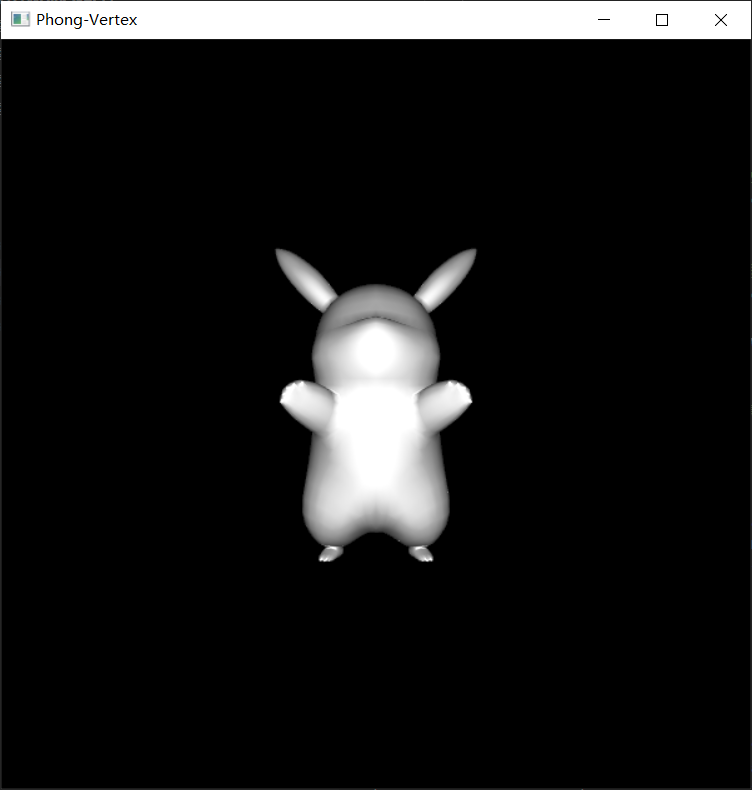
通过交互地改变物体、光源、材质、相机等参数进一步了解使用Phong反射模型实现的光照效果。

* 1. 尝试为光源light设置交互：如鼠标设置光源x,y位置（已实现）
  2. 尝试为材质material设置交互：1~4增减反射系数（已实现），5~7设置控制三种控制变量，来控制着色器单独使用环境光反射、漫反射、镜面反射（未实现）
  3. 尝试旋转或平移物体相机，从不同角度观察光照效果（已实现）

1. 示例结果
2. 具体修改内容如下：
   1. Task-1：在Trimesh.cpp中完善computeTriangleNormals()和computeVertexNormals()，接着在main.cpp中完善bindObjectAndData()
   2. Task-2：在fhsader.glsl中完善main()
   3. Task-3：在main.cpp中mainWindow\_key\_callback ()函数完善键盘交互
3. 下面分别为初始结果、添加漫反射结果、添加镜面反射结果



1. 下面分别为相同光源不同材质的结果



1. 下图左边是Phong反射模型在片元着色器上的实现（Phong着色），右图是上次实验在顶点着色器上的实现（Gouraud着色）

