CG_HW6

实现思路

Gouraud Shading

- 计算出每个顶点的法线为所有邻接面法线的均值
- 通过顶点的法线和冯氏光照模型计算出每个顶点的颜色值
- 多边形内部的其他像素值通过顶点颜色值的线性插值得到
- 具体到实现：

```
// 根据描述可以知道，冯氏光照模型是应用到顶点着色器中的
/*   顶点着色器   */
// 计算出顶点像素的位置和法向量
"vec3 fragment_position = vec3(model * vec4(vertex_position, 1.0));\n"
"vec3 fragment_normal = mat3(transpose(inverse(model))) * vertex_normal;\n"

// 对顶点像素应用冯氏光照模型计算顶点像素颜色
// 环境光
"vec3 ambient = ambientStrength * light_color;"

// 漫反射
"vec3 norm = normalize(fragment_normal);\n"
"vec3 light_direction = normalize(light_position - fragment_position);\n"
"float diff = max(dot(norm, light_direction), 0.0);\n"
"vec3 diffuse = diffuseStrength * diff * light_color;\n"

// 镜面反射
"vec3 view_direction = normalize(view_position - fragment_position);\n"
"vec3 reflect_direction = reflect(-light_direction, norm); \n"
"float spec = pow(max(dot(view_direction, reflect_direction), 0.0), shininess);\n"
"vec3 specular = specularStrength * spec * light_color;\n"

// 加起来得到最终效果
"vertex_color =  ambient + diffuse + specular;\n"


/*   片段着色器   */
// 对于每一个插值像素，颜色值为顶点颜色与物体颜色积的插值
"fragment_color = vec4(vertex_color * object_color, 1.0);\n"
```

Phong Shading

- 计算出每个顶点的法线为所有邻接面法线的均值
- 对于每个多面体内部的每个像素，利用顶点法线进行线性插值获得该像素的法线
- 利用像素的法线和冯氏光照模型计算出每个像素值
- 具体到实现

```
`// 根据描述可以知道，冯氏光照模型是应用到片段着色器中的
/*   顶点着色器   */
 // 计算出顶点像素位置和法线
"fragment_position = vec3(model * vec4(vertex_position, 1.0));\n"
"fragment_normal = mat3(transpose(inverse(model))) * vertex_normal; \n"
```

```
/*  片段着色器  */
// 输入顶点像素位置和法线，着色器会自动对其线性插值
// 对于每一个插值像素，利用法线和冯氏光照模型计算出颜色值
"vec3 ambient = ambientStrength * light_color;"

"vec3 norm = normalize(fragment_normal);\n"
"vec3 light_direction = normalize(light_position - fragment_position);\n"
"float diff = max(dot(norm, light_direction), 0.0);\n"
"vec3 diffuse = diffuseStrength * diff * light_color;\n"

"vec3 view_direction = normalize(view_position - fragment_position);\n"
"vec3 reflect_direction = reflect(-light_direction, norm); \n"
"float spec = pow(max(dot(view_direction, reflect_direction), 0.0), shininess);\n"
"vec3 specular = specularStrength * spec * light_color;\n"

"vec3 result = (ambient + diffuse + specular) * object_color;\n"

"fragment_color = vec4(result, 1.0);\n"
```

实现效果见视频

可以看到，前者不能很好地表现物体表面的高光，只能在顶点处表现高光，而后者就不存在这个问题。