Report

1. 实现 Phong 光照模型：
   - 场景中绘制一个 cube
     见 gif。两个 cube（物体，光源）使用同一组顶点数据，光源缩小移动后得到。

   - 自己写 shader 实现两种 shading: Phong Shading 和 Gouraud Shading，并解释两种 shading 的实现原理。
     两种 shader 的光照实现都是将环境光照、漫反射光照、镜面光照三个光照分量求和后再与物体颜色相乘，得到反射的颜色。两者的区别在于求光照分量的所在步骤不一样。Phong Shading 是在片段着色器中再计算光照颜色（包括三个光照分量以及最终的反射颜色），因此将对每个片段都计算光照颜色，运算量大但是显示内容更精确详细；Gouraud Shading 是在顶点着色器中就计算了三个光照分量，在片段着色器才与物体颜色相乘得到最终光照颜色，因此分量计算的对象只有少数的几个顶点，运算量小，但是缺少细节信息，顶点之间的光照颜色依靠插值补充，因此在顶点不足够多的情况下将显得不太真实。

   - 合理设置视点、光照位置、光照颜色等参数，使光照效果明显显示

     ```
     static float ambientStrength = 0.1;
     static float diffuseStrength = 1;
     static float specularStrength = 0.5;
     static int shininessStrength = 32;
     ```

     使用教程上的参数。

2. 使用 GUI，使参数可调节，效果实时更改：
   - GUI 里可以切换两种 shading
     见 gif，通过单选选择。

   - 使用如进度条这样的控件，使 ambient 因子、diffuse 因子、specular 因子、反光度等参数可调节，光照效果实时更改
3. Bonus：
   - 当前光源为静止状态，尝试使光源在场景中来回移动，光照效果实时更改。

     ```
     lightPos.x = sin(glfwGetTime()) * 2.0f;
     lightPos.y = cos(glfwGetTime()) * 2.0f;
     lightPos.z = sin(glfwGetTime()) * 2.0f;
     ```

     光源以椭圆的轨迹在三个不同维度上运动。在 xOy 平面为圆，在 yOz 为圆，在 xOz 平面为 z=x 线段。

程序可使用 W S A D 移动，↑ ↓ ← →方向键代替鼠标移动视角。