BASIC：

动态效果见附视频文件 hw6.mp4。

1. 实现 Phong 光照模型：
   - 场景中绘制一个 cube
     效果见附视频文件
   - 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。
     效果见附视频文件
     解释：Phong Shading 是冯氏光照模型，主要结构由由 3 个分量组成：环境(Ambient)、漫反射(Diffuse)和镜面(Specular)光照。环境光照模拟的是环境中有多种光源组合成的在物体所有表面都存在的发散光线，只需要用一个环境光照参数乘光源，然后再乘以物体反射的颜色即可。漫反射模拟的是物体上与光线方向越接近的片段能从光源处获得更多的亮度，需要根据物体表面的点到光源的向量跟法向量的夹角大小决定，夹角越大，反射越弱。镜面光照也是依据光的方向向量和物体的法向量来决定的，但是它也依赖于观察方向，通过 Snell's Law 来计算反射向量，当观察方向与反射向量夹角越小，反射越强。Phong Shading 对这些光照分量的计算与处理都是在片段着色器中完成。而 Gouraud Shading 的主要结构与 Phong Shading 相同，只是计算放在顶点着色器中，所以其实物体表面只是顶点计算之后插值的结果，相对于 Phong Shading 效果要差一些。
   - 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
     效果见附视频文件

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
     效果见附视频文件
   - 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
     效果见附视频文件

Bonus

1. 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。
   效果见附视频文件
   当勾选 bonus 时，光源会在 XZ 平面上围绕着 cube 旋转，可以观察效果的实时变换。

代码思路解析

附代码文件（本次作业的核心代码在 hw6.cpp、camera.h、shader.h 文件内）

1. Shader 主要实现代码

```
//phong shader code
const char *phong_vs = "#version 330 core\n"
    "layout (location = 0) in vec3 aPos;\n"
    "layout (location = 1) in vec3 aColor;\n"
    "layout (location = 2) in vec3 aNormal;\n"
    "uniform mat4 model;\n"
    "uniform mat4 view;\n"
    "uniform mat4 projection;\n"
    "out vec3 ourColor;\n"
    "out vec3 ourFragPos;\n"
    "out vec3 ourNormal;\n"
    "void main()\n"
    "{\n"
    "    gl_Position = projection * view * model * vec4(aPos, 1.0f);\n"
    "    ourColor = aColor;\n"
    "    ourFragPos = vec3(model * vec4(aPos, 1.0f));\n"
    "    ourNormal = mat3(transpose(inverse(model))) * aNormal;\n"
    "}\0";

const char *phong_fs = "#version 330 core\n"
    "struct Material {\n"
    "    float shininess;\n"
    "    };\n"
    "struct Light {\n"
    "    vec3 ambient;\n"
    "    vec3 diffuse;\n"
    "    vec3 specular;\n"
    "    vec3 position;\n"
    "    };\n"
    "in vec3 ourColor;\n"
    "in vec3 ourFragPos;\n"
    "in vec3 ourNormal;\n"
    "uniform Material material;\n"
    "uniform Light light;\n"
    "uniform vec3 viewPos;\n"
    "out vec4 FragColor;\n"
    "void main()\n"
    "{\n"
```

```
"	//ambient\n"
"	vec3 ambient = light.ambient;\n"
"	//diffuse\n"
"	vec3 norm = normalize(ourNormal);\n"
"	vec3 lightDir = normalize(light.position - ourFragPos);\n"
"	float diff = max(dot(norm, lightDir), 0.0f);\n"
"	vec3 diffuse = diff * light.diffuse;\n"
"	//specular\n"
"	vec3 viewDir = normalize(viewPos - ourFragPos);\n"
"	vec3 reflectDir = reflect(-lightDir, norm);\n"
"	float spec = pow(max(dot(viewDir, reflectDir), 0.0f), material.shininess);\n"
"	vec3 specular = spec * light.specular;\n"
"	//result\n"
"	vec3 result = (ambient + diffuse + specular) * ourColor;\n"
"	FragColor = vec4(result, 1.0f);\n"
"}\n\0";
//gouraud shader code
static const char *gouraud_vs = "#version 330 core\n"
"layout (location = 0) in vec3 aPos;\n"
"layout (location = 1) in vec3 aColor;\n"
"layout (location = 2) in vec3 aNormal;\n"
"struct Material {\n"
"	float shininess;\n"
"	};\n"
"struct Light {\n"
"	vec3 ambient;\n"
"	vec3 diffuse;\n"
"	vec3 specular;\n"
"	vec3 position;\n"
"	};\n"
"uniform mat4 model;\n"
"uniform mat4 view;\n"
"uniform mat4 projection;\n"
"uniform Material material;\n"
"uniform Light light;\n"
"uniform vec3 viewPos;\n"
"out vec3 ourColor;\n"
"void main()\n"
"{\n"
"	gl_Position = projection * view * model * vec4(aPos, 1.0f);\n"
"	vec3 ourFragPos = vec3(model * vec4(aPos, 1.0f));\n"
"	vec3 ourNormal = mat3(transpose(inverse(model))) * aNormal;\n"
"	//ambient\n"
```

```
                "      vec3 ambient = light.ambient;\n"
                "      //diffuse\n"
                "      vec3 norm = normalize(ourNormal);\n"
                "      vec3 lightDir = normalize(light.position - ourFragPos);\n"
                "      float diff = max(dot(norm, lightDir), 0.0f);\n"
                "      vec3 diffuse = diff * light.diffuse;\n"
                "      //specular\n"
                "      vec3 viewDir = normalize(viewPos - ourFragPos);\n"
                "      vec3 reflectDir = reflect(-lightDir, norm);\n"
                "      float spec = pow(max(dot(viewDir, reflectDir), 0.0f),
material.shininess);\n"
                "      vec3 specular = spec * light.specular;\n"
                "      //result\n"
                "      ourColor = (ambient + diffuse + specular) * aColor;\n"
                "}\0";

            static const char *gouraud_fs = "#version 330 core\n"
                "in vec3 ourColor;\n"
                "out vec4 FragColor;\n"
                "void main()\n"
                "{\n"
                "      FragColor = vec4(ourColor, 1.0f);\n"
            "}\n\0";
```