第一步，创建Light类，用来生成光源
详见Light.h light.vs light.frag

第二步，在main函数里包含头文件

```
#include <GL/glew.h>

#include <GLFW/glfw3.h>

#include "Shader.h"
#include "Camera.h"
#include "Light.h"

#include <glm/glm.hpp>
```

同时定义lightPos全局变量并初始化

```
GLfloat lastX = WIDTH/2;
GLfloat lastY = HEIGHT/2;

bool firstMouse = true;
glm::vec3 lightPos = glm::vec3(0.0f, 0.0f,
    0.0f);
```

为了使用Light类画光源，我们需要生成lightShader,并配置对应的VAO，VBO，我们在实例化shader前添加以下代码

```
Shader lightShader =
    Shader("res/shaders/light
    .vs","res/shaders/light.frag");

Light lightModel = Light();
```

绘制光源，需要在while循环里增加以下代码

```
glClearColor(0.2f, 0.3f, 0.3f, 1.0f);
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
glViewport(0, 0, screenWidth, screenHeight);
view = camera.GetViewMatrix();
glm::mat4 transform = glm::mat4(1.0f);
lightShader.Use();
lightPos = glm::vec3(40.0f, 0.0f, 0.0f);
lightPos = glm::rotate(lightPos, glm::radians(20.0f)*static_cast<GLfloat>(glfwGetTime()),
    glm::vec3(1.0f, 1.0f, 1.0f));
transform = glm::scale(transform, glm::vec3(0.1f, 0.1f, 0.1f));
transform = glm::translate(transform, lightPos);
glUniformMatrix4fv(glGetUniformLocation(lightShader.Program, "model"), 1, GL_FALSE,
    glm::value_ptr(transform));
glUniformMatrix4fv(glGetUniformLocation(lightShader.Program, "view"), 1, GL_FALSE,
    glm::value_ptr(view));
glUniformMatrix4fv(glGetUniformLocation(lightShader.Program, "projection"), 1, GL_FALSE,
    glm::value_ptr(projection));

lightModel.Draw(lightShader);
```

其中，我们通过移动lightPos，获得当前的光源的位置，然后讲transform矩阵修改为移到光源位置

对原来的立方体，我们去掉它的平移和旋转。
运行程序测试，能看到白色的光源在绕着物体旋转。

下面对立方体进行修改，首先是core.vs
其中我们需要导入物体表面法向量，同时将经过变换后的法向量及片元位置FragPos传给core.frag
其中需要注意的是，片元位置FragPos是物体在世界坐标系下的点，不需要与矩阵project和view相乘，法向量则需要进行修正。
片元着色器core.frag里，我们导入Normal， FragPos，通过uniform实时导入LightPos，ViewPos和p三个分量。修改一下参数，ambient的参数改为0.3f，spec后的参数改为3.0f，修改后效果更好
下面按照光照公式，分别写入环境光，漫反射，镜面反射。
相机类Camera.h里添加GetPosition函数
最后回到main.cpp文件里
我们将变量实时传入，这里课上有错误，LightPos以及ViewPos需要大写，图中已经修正

```
GLuint projLoc = glGetUniformLocation(shader.Program, "projection");
glUniformMatrix4fv(projLoc, 1, GL_FALSE, glm::value_ptr(projection));
glUniform3f(glGetUniformLocation(shader.Program, "LightPos"), lightPos.x,lightPos.y,lightPos.z);
glUniform3f(glGetUniformLocation(shader.Program, "ViewPos"),
    camera.GetPosition().x,camera.GetPosition().y,camera.GetPosition().z);
glUniform1f(glGetUniformLocation(shader.Program, "p"), 64.0f);
glBindVertexArray(VAO);
```

最后回到vertices[]里，添加法向量