《计算机图形学实验》综合实验报告

题目 茶壶与光照

学 号 **20201050475**

姓 名 杨媛

指导教师 钱文华

日 期 **2022.6.21**

中文摘要

实验要求利用 Visual C++, OpenGL, Java 等工具，实现三维图形渲染，自定义三维图形，渲染过程须加入纹理、色彩、光照、阴影、透明等效果。而本次实验则主要聚焦于三维图形的光照。

OpenGL 可以设置至少 8 种光源，而在本次实验中使用了两种光源，一种是环境光，另一种是聚光灯。

关键词：OpenGL，茶壶，光照。

目录

正文

1、实验内容

利用 Visual C++, OpenGL, Java 等工具，实现三维图形渲染，自定义三维图形，三维图形不能仅仅是简单的茶壶、球体、圆柱体、圆锥体等图形，渲染过程须加入纹理、色彩、光照、阴影、透明等效果，可采用光线跟踪、光照明模型、纹理贴图、纹理映射等算法。

2、开发工具，程序设计及实现目的及基本模块介绍

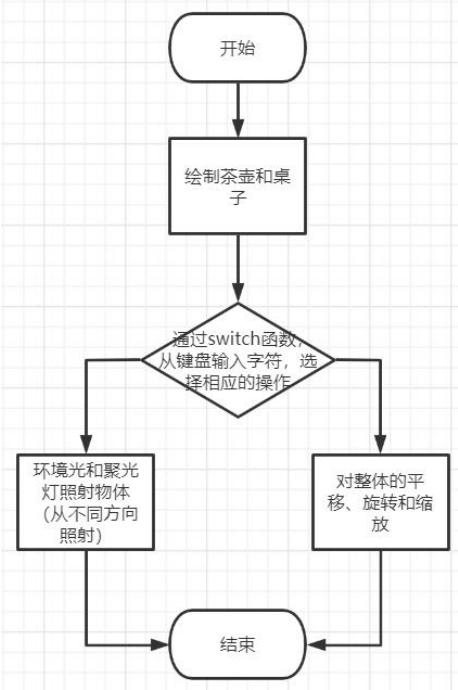
开发工具：Visual C++, OpenGL, Java 等工具，以及深度学习相关工具。

基本模块：茶壶和桌子的绘制，环境光和聚光灯带来的光照变化，物体整体的三维变化（平行、旋转、缩放），透视投影和正投影，键盘交互。

3、关键算法的理论介绍和程序实现步骤（可画流程图）

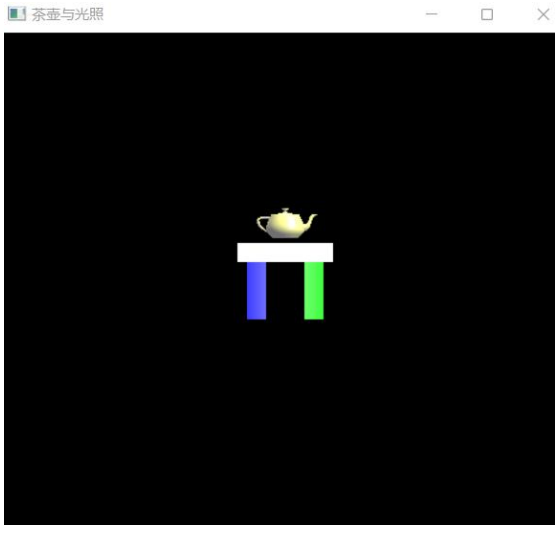
在计算机三维图像中，投影可以看作是一种将三维坐标变换为二维坐标的方法，常用到的有正交投影和透视投影。正交投影多用于三维健模，透视投影则由于和人的视觉系统相似，多用于在二维平面中对三维世界的呈现。

透视投影（Perspective Projection）是为了获得接近真实三维物体的视觉效果而在二维的纸或者画布平面上绘图或者渲染的一种方法，也称为透视图。它具有消失感、距离感、相同大小的形体呈现出有规律的变化等一系列的透视特性，能逼真地反映形体的空间形象。透视投影通常用于动画、视觉仿真以及其它许多具有真实性反映的方面。

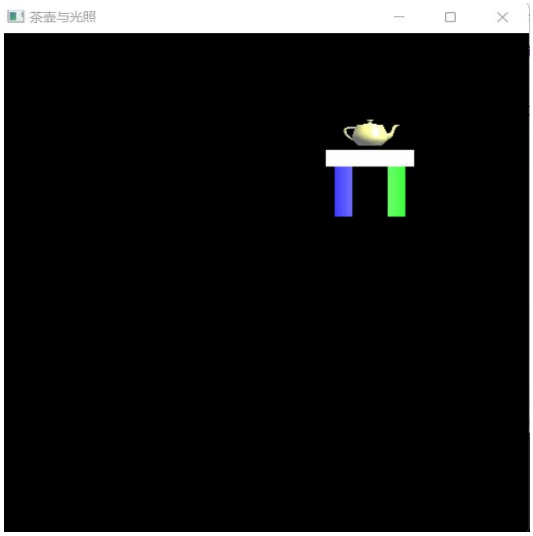


4、实验运行结果及存在问题

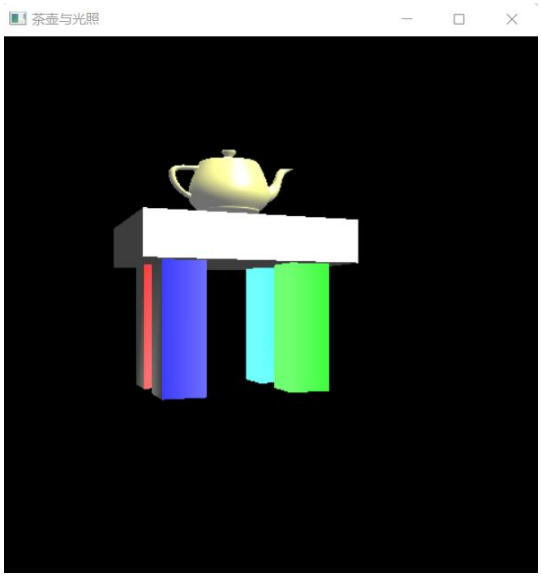
运行结果：



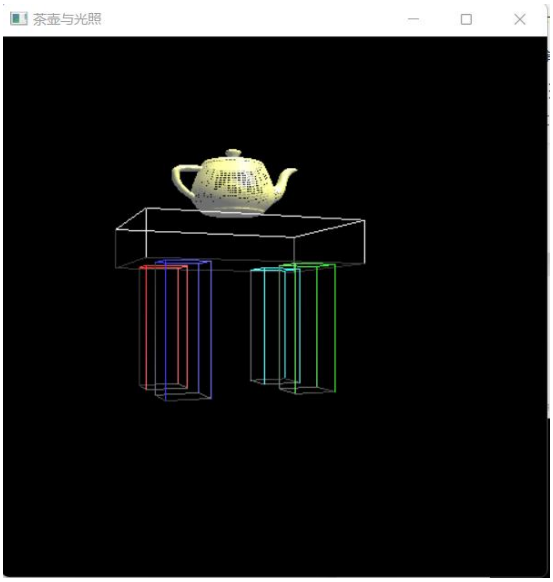
平移后



放大并旋转后

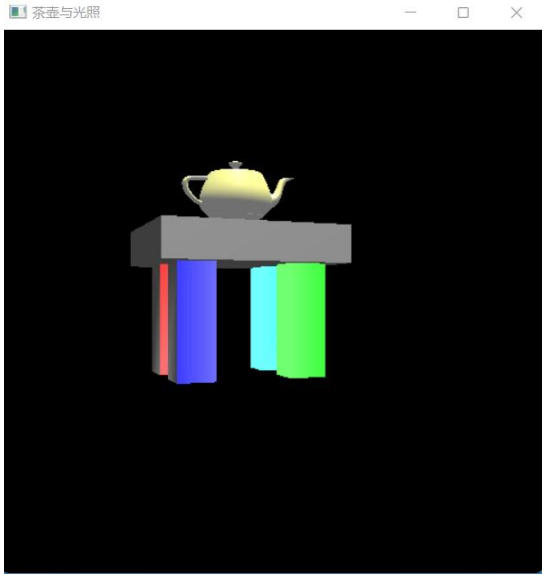


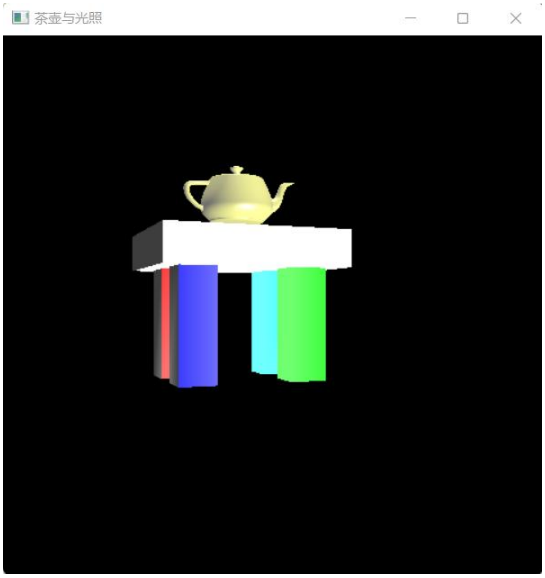
线框图



环境光（上移）

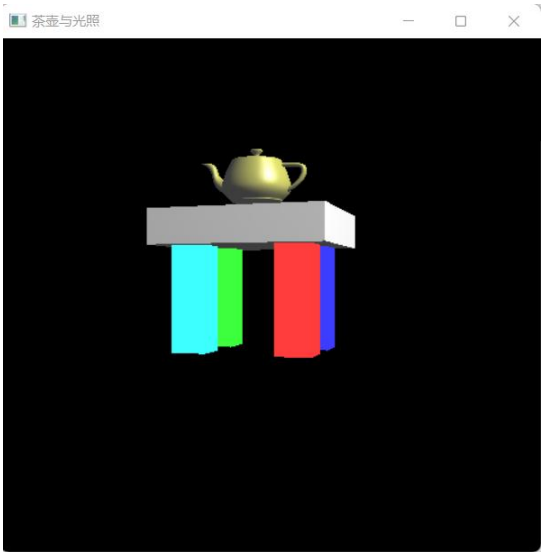
环境光（下移）

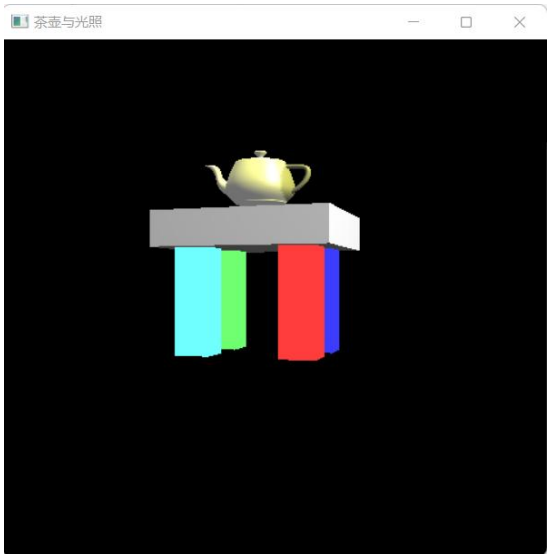




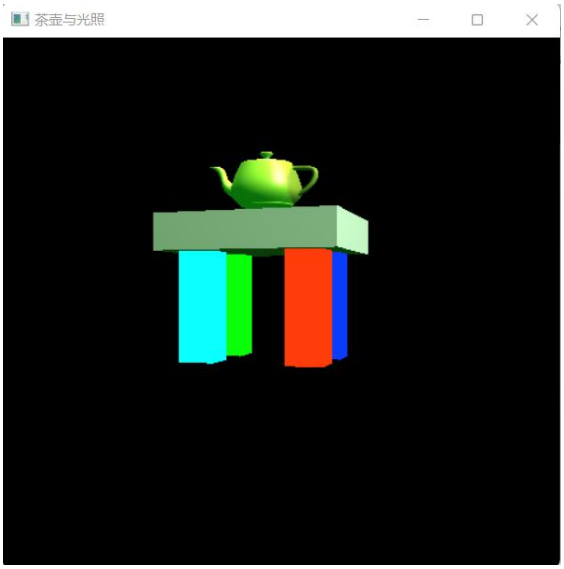
聚光灯（左移）

聚光灯（右移）

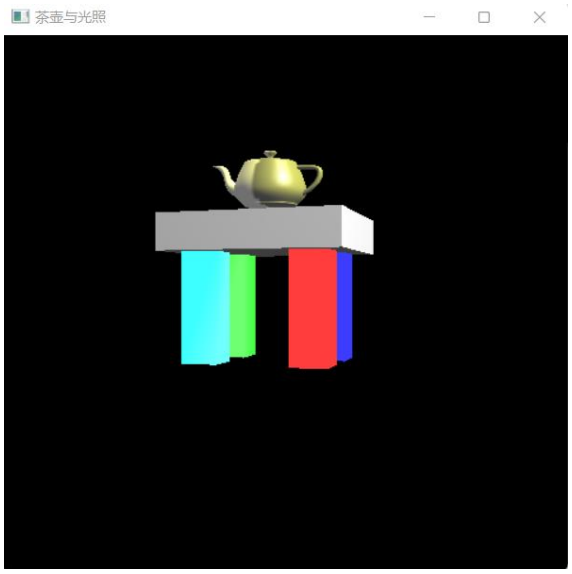




环境灯变色



聚光灯角度变小



存在问题：按下“p”后，才能通过前移和后移来控制物体缩放；光照变化的效果在某些时候并不明显。

5、实验体会及小结

环境光：影响因素为强度、颜色。这种类型的光源来自空间中的任何地方，并以相同的方式照亮所有物体。

聚光灯：影响因素为强度、颜色、位置、衰减因素、方向、夹角。

在编写代码的过程中，遇到了不少问题，通过在网上查阅资料，以及寻求帮助而解决了大部分问题。这不仅有助于我完成本次实验，而且还让我学到了更多与计算机图形学相关的知识。

对计算机图形学的认识：经过了一阶段计算机图形学的学习对于图形学中基本图形的生成算法有了一定的了解。深度研究图形学需要高深的数学知识且每一个细化的方向需要的知识也不一样。图形学是计算机科学与技术学科的活跃前沿学科被广泛的应用到生物学、物理学、化学、天文学、地球物理学、材料科学等领域。我深深感到这门学科涉及的领域之广是惊人的可以说博大精深。在这个计算机的时代什么都要用到，计算机技术图形也是我们生活中重要的部分，所以我们必须好好学习计算机图形学。

附录（所有代码，包括代码注释）

```
#include <windows.h>
#include <GL/glut.h>
#include <stdlib.h>
#include <stdio.h>
float fTranslate;
float fRotate;
float fScale= 1.0f;
float color_b = 1.0f;
```

```
bool bPersp = false;
bool bAnim = false;
bool bWire = false;
bool isWhite = true;
int wHeight = 0;
int wWidth = 0;
GLfloat color[] = { 1.0, 1.0, 1.0, 1.0 };
// 定义颜色
```

```
float eye[] = { 0, 0, 8 };
float center[] = { 0, 0, 0 };
GLfloat spotangle = 5.0f; //角度
//环境光位置
GLfloat light_x = 0.0f;
GLfloat light_y = 0.0f;
GLfloat light_z = 0.0f;
//聚光灯方向
GLfloat dir_x = 0.0f;
GLfloat dir_y = 0.0f;
GLfloat dir_z = 0.0f;
//绘制腿部
void Draw_Leg()
{
    glScalef(1, 1, 3);
    glutSolidCube(1.0);
}
void Draw_Triangle()
{
    GLfloat mat_specular[] = { 0.6f, 0.6f, 0.6f, 1.0f };//颜色
    GLfloat mat_diffuse0[] = { 0.65f, 0.65f, 0.2f, 1.0f };
    GLfloat mat_diffuse1[] = { 1.0f, 1.0f, 1.0f };
    GLfloat mat_diffuse2[] = { 0.0f, 1.0f, 1.0f };
    GLfloat mat_diffuse3[] = { 1.0f, 0.0f, 0.0f };
    GLfloat mat_diffuse4[] = { 0.0f, 1.0f, 0.0f };
    GLfloat mat_diffuse5[] = { 0.0f, 0.0f, 1.0f };
//画茶壶
    glPushMatrix();
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);// 设置多边形正面的镜面反射属性
    glMateriali(GL_FRONT_AND_BACK, GL_SHININESS, 50);// 指定镜面指数
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse0); //设置多边形正面漫反射属性
    glTranslatef(0, 0, 4+1);
    glRotatef(90, 1, 0, 0);
    glutSolidTeapot(1);
    glPopMatrix();
     //画桌面
    glPushMatrix();
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_diffuse1);// 设置多边形正面的镜面反射属性
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse1);//设置多边形正面漫反射属性
    glTranslatef(0, 0, 3.5);
    glScalef(5, 4, 1);
    glutSolidCube(1.0);
    glPopMatrix();
    //画四条腿
    glPushMatrix();
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_diffuse2);// 设置多边形正面的镜面反射属性
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse2);//设置多边形正面漫反射属性
    glTranslatef(1.5, 1, 1.5);
    Draw_Leg();
    glPopMatrix();

    glPushMatrix();
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_diffuse3);// 设置多边形正面的镜面反射属性
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse3);//设置多边形正面漫反射属性
    glTranslatef(-1.5, 1, 1.5);
    Draw_Leg();
    glPopMatrix();

    glPushMatrix();
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_diffuse4);// 设置多边形正面的镜面反射属性
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse4);//设置多边形正面漫反射属性
    glTranslatef(1.5, -1, 1.5);
    Draw_Leg();
    glPopMatrix();
```

```
    glPushMatrix();
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, mat_diffuse5);// 设置多边形正面的镜面反射属性
    glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, mat_diffuse5);// 设置多边形正面漫反射属性
    glTranslatef(-1.5, -1, 1.5);
    Draw_Leg();
    glPopMatrix();
}

void updateView(int width, int height)
{
    glViewport(0,0,width,height);//设置视窗大小
    glMatrixMode(GL_PROJECTION);//设置矩阵模式为投影
    glLoadIdentity();    //初始化矩阵为单位矩阵
    float whRatio = (GLfloat)width/(GLfloat)height;    //设置显示比例
    if (bPersp)
        gluPerspective(45.0f, whRatio,0.1f,100.0f); //透视投影
    else
        glOrtho(-3 ,3, -3, 3,-100,100);    //正投影
    glMatrixMode(GL_MODELVIEW);    //设置矩阵模式为模型
}

void reshape(int width, int height)
{
    if (height==0)        //如果高度为 0
        height=1; //让高度为 1（避免出现分母为 0 的现象）
    wHeight = height;
    wWidth = width;
    updateView(wHeight, wWidth); //更新视角
}
```

```
void idle()
{
    glutPostRedisplay(); //调用当前绘制函数
}

void key(unsigned char k, int x, int y)
{
    switch (k)
    {
    case 27:
    case 'q': {exit(0); break; }//退出程序
    case 'p': {bPersp = !bPersp; break; }//缩放
    case ' ': {bAnim = !bAnim; break; }//旋转
    case 'o': {bWire = !bWire; break; }//线框图
    case 'a': { //整体左移
        eye[0] += 0.2f;
        center[0] += 0.2f;
        break;
        }
    case 'd': {//整体右移
        eye[0] -= 0.2f;
        center[0] -= 0.2f;
        break;
        }
    case 'w': {//整体上移
        eye[1] -= 0.2f;
        center[1] -= 0.2f;
        break;
        }
    case 's': {//整体下移
        eye[1] += 0.2f;
        center[1] += 0.2f;
        break;
        }
    case 'z': {//整体前移
        eye[2] -= 0.2f;
        center[2] -= 0.2f;
        break;
        }
```

```
case 'c': {//整体后移
        eye[2] += 0.2f;
        center[2] += 0.2f;
        break;
        }
case 'j': {//环境光左移
        light_x = light_x - 0.2f;
        break;
        }
case 'l': {//环境光右移
        light_x = light_x + 0.2f;
        break;
        }
case 'i': {//环境光上移
        light_y = light_y + 0.2f;
        break;
        }
case 'k': {//环境光下移
        light_y = light_y - 0.2f;
        break;
        }
case 'n': {//环境光前移
        light_z = light_z + 0.2f;
        break;
        }
case 'm': {//环境光后移
        light_z = light_z - 0.2f;
        break;
        }
case 'r': {//环境光颜色切换
        isWhite = !isWhite;
        break;
        }
case 'f': {//聚光灯左移
        dir_x = dir_x - 0.05f;
        break;
        }
case 'h': {//聚光灯右移
        dir_x = dir_x + 0.05f;
        break;
        }
case 't':{//聚光灯上移
        dir_y = dir_y - 0.05f;
        break;
        }
case 'v': {//聚光灯后移
        dir_z = dir_z - 0.05f;
        break;
        }
case 'b': {//聚光灯前移
        dir_z = dir_z + 0.05f;
        break;
        }
case 'x': { //聚光灯角度变大
        if (spotangle <= 89.0f)
            spotangle = spotangle + 0.2f;
        break;
        }
case 'y': { //聚光灯角度变小
        if (spotangle >= 1.0f)
            spotangle = spotangle - 0.2f;
        break;
        }
    }
    updateView(wHeight, wWidth);
}
void redraw()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);  //清除颜色缓存和深度缓存
    glLoadIdentity(); //初始化矩阵为单位矩阵
    gluLookAt(eye[0],eye[1],eye[2],center[0], center[1], center[2],0, 1, 0);  // 场景（0，0，0）的视点中心(0,5,50)，Y 轴向上
    if (bWire) {
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
        //设置多边形绘制模式：正反面，线型
    }
    else {
    glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
        //设置多边形绘制模式：正反面，填充
    }
```

```
glEnable(GL_DEPTH_TEST);//开启深度测试
    glEnable(GL_LIGHTING);  //开启光照模式
    GLfloat light_pos[] = {5.0 + light_x ,5.0 + light_y,5.0 + light_z,1}; //定义环境光位置
    GLfloat light_pos1[] = { 0.0f, 5.0f, 0.0f, 1.0f }; //定义聚光灯位置
    GLfloat lightDir[] = { 0.0f + dir_x ,-1.0f + dir_y ,0.0f + dir_z }; //角度
    GLfloat white[] = { 1.0f, 1.0f, 1.0f, 1.0f }; //定义白色
    if (isWhite) {
        color[0] = 1.0f, color[1] = 1.0f, color[2] = 1.0f, color[3] = 1.0f;
    }
    else {
        color[0] = 0.0f, color[1] = 1.0f, color[2] = 0.0f, color[3] = 1.0f;
    }
    glLightfv(GL_LIGHT0, GL_POSITION, light_pos); //设置第0号光源的光照位置
    glLightfv(GL_LIGHT0, GL_SPECULAR, white); //设置镜面反射光照颜色
    glLightfv(GL_LIGHT0, GL_DIFFUSE, white);      //设置漫射光成分
    glLightfv(GL_LIGHT0, GL_AMBIENT, color);    //设置第0号光源多次反射后的光照颜色（环境光颜色）
    glEnable(GL_LIGHT0); //开启第0号光源
    glLightfv(GL_LIGHT1, GL_AMBIENT, color);        //设置环境光成分
    glLightfv(GL_LIGHT1, GL_SPECULAR, white);       //设置镜面光成分
    glLightfv(GL_LIGHT1, GL_DIFFUSE, white);        //设置漫射光成分
    glLightfv(GL_LIGHT1, GL_POSITION, light_pos1);
    glLightf(GL_LIGHT1, GL_SPOT_CUTOFF, spotangle); //裁减角度
    glLightfv(GL_LIGHT1, GL_SPOT_DIRECTION, lightDir); //光源方向
    glLightf(GL_LIGHT1, GL_SPOT_EXPONENT, 2.);          //聚集度
    glEnable(GL_LIGHT1);
    glRotatef(fRotate, 0, 1.0f, 0);  //旋转
    glRotatef(-90, 1, 0, 0); //旋转
```

```
    glScalef(0.2, 0.2, 0.2);   //缩放
    Draw_Triangle();    //绘制场景
    if (bAnim) fRotate+= 0.5f; //旋转因子改变
    glutSwapBuffers(); //交换缓冲区
}
int main (int argc, char *argv[])
{
    glutInit(&argc, argv);//对 glut 的初始化
    glutInitDisplayMode(GLUT_RGBA   |   GLUT_DEPTH   |
GLUT_DOUBLE);
    //初始化显示模式:RGB 颜色模型，深度测试，双缓冲
    glutInitWindowSize(480,480);//设置窗口大小
    int windowHandle = glutCreateWindow(" 茶 壶 与 光 照
");//设置窗口标题
    glutDisplayFunc(redraw); //注册绘制回调函数
    glutReshapeFunc(reshape);   //注册重绘回调函数
    glutKeyboardFunc(key); //注册按键回调函数
    glutIdleFunc(idle);//注册全局回调函数：空闲时调用
    glutMainLoop();   // glut 事件处理循环
    return 0;
}
```