# 《计算机图形学实验》综合实验报告

题目 基于 OpenGL 的三维图形渲染

学 号 20201060276

姓 名 宋佳轩

指导教师 钱文华

日 期 2022.6.21

# 摘要

本次实验通过 opengl 实现了对于茶壶添加光照效果与关闭灯光效果，以及实现了对茶壶的贴图以及贴图的转换。

**关键词：**Opengl 光照 纹理 三维图形 视角变换

# abstract

In this experiment, OpenGL is used to add lighting effect and turn off lighting effect to the teapot, and to realize the mapping and mapping conversion of the teapot.

**Key words:** OpenGL lighting texture 3D graphics perspective transformation

# 目录

# 一、实验背景与目的

经过一学期对计算机图形学的学习，我学到了许多关于图形学的知识以及OpenGL 的用法，对于相关的语法有了一定的了解。而本次实验就是对自己一学期计算机图形学实验的总结，通过之前所学习到的知识与技巧实现对一个茶壶的渲染。

# 二、实验内容与工具

通过 Visual Studio, OpenGL, Java 等工具，利用课程所学知识实现自定义三维图形与三维图形渲染，，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。

# 三、程序设计与基本模块介绍

程序分为五个模块：

1、贴图模块：实现对图片的采集与处理形成能够使用的贴图

2、光照模块，实现光源的定义与图形光照性质的定义

3、三维图形绘制模块：实现茶壶的绘制与贴图的生成

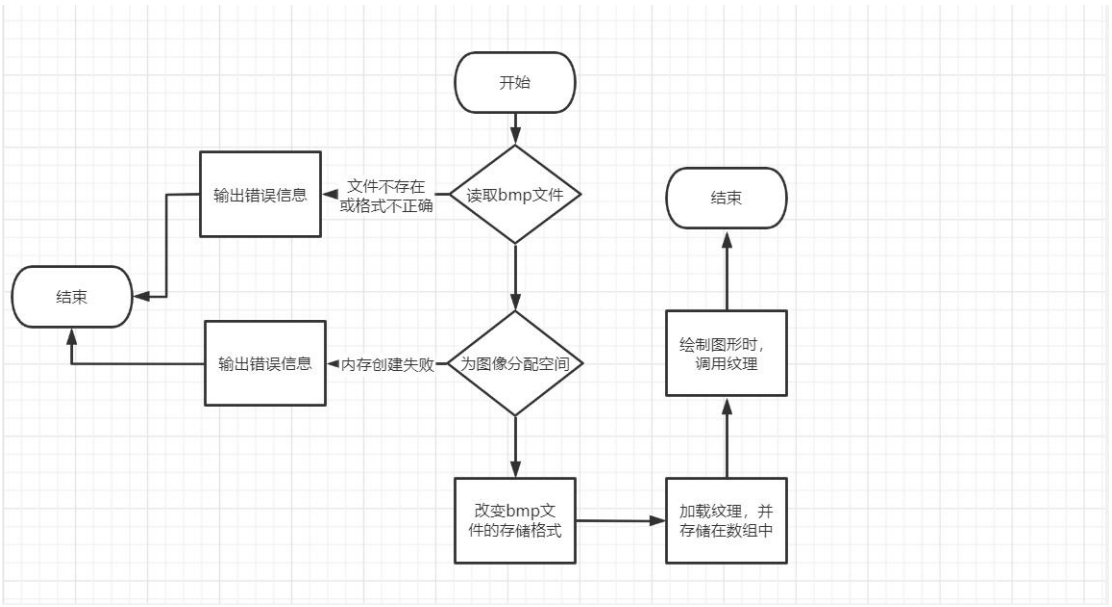
4、鼠标交互模块：实现图形的旋转，贴图的转换与光照的开关

5、主程序模块：启动程序

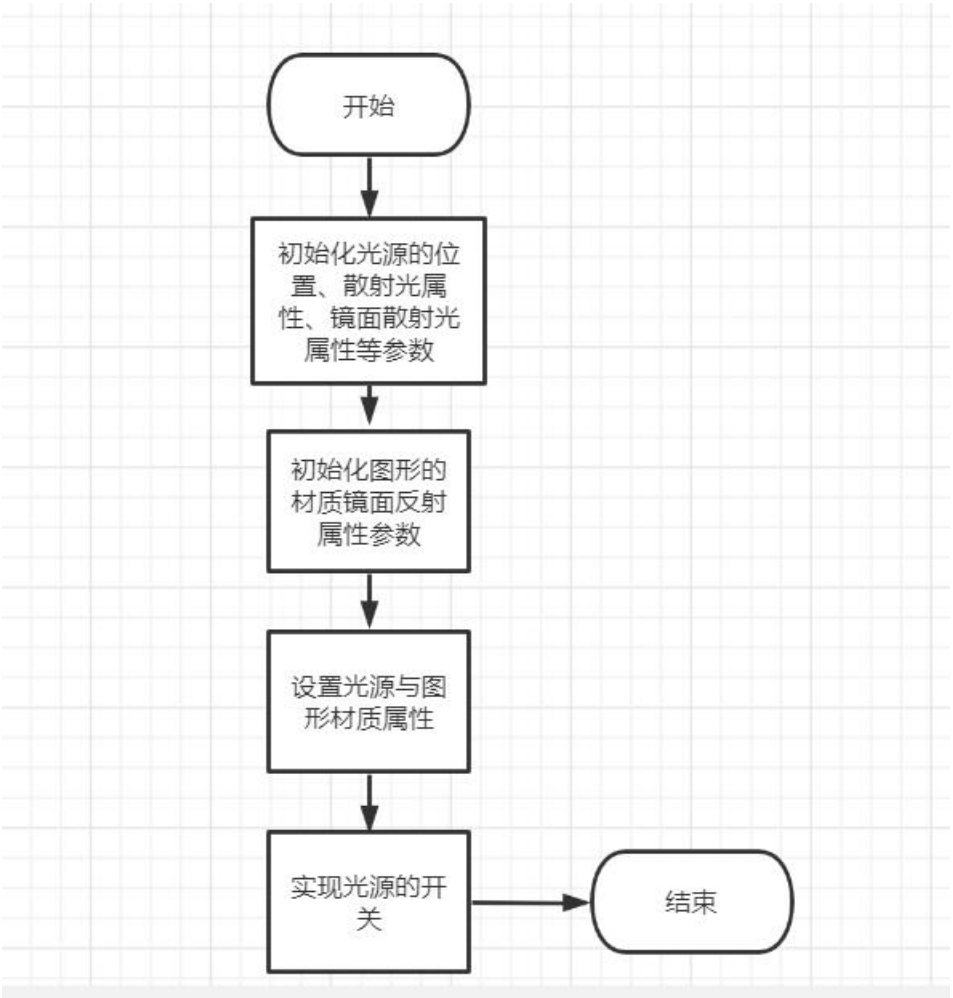
# 四、关键算法的介绍

纹理图片的读取：

主要关键点为 bmp 文件的读取与处理以及纹理的加载




光照与图形光照性质的生成：



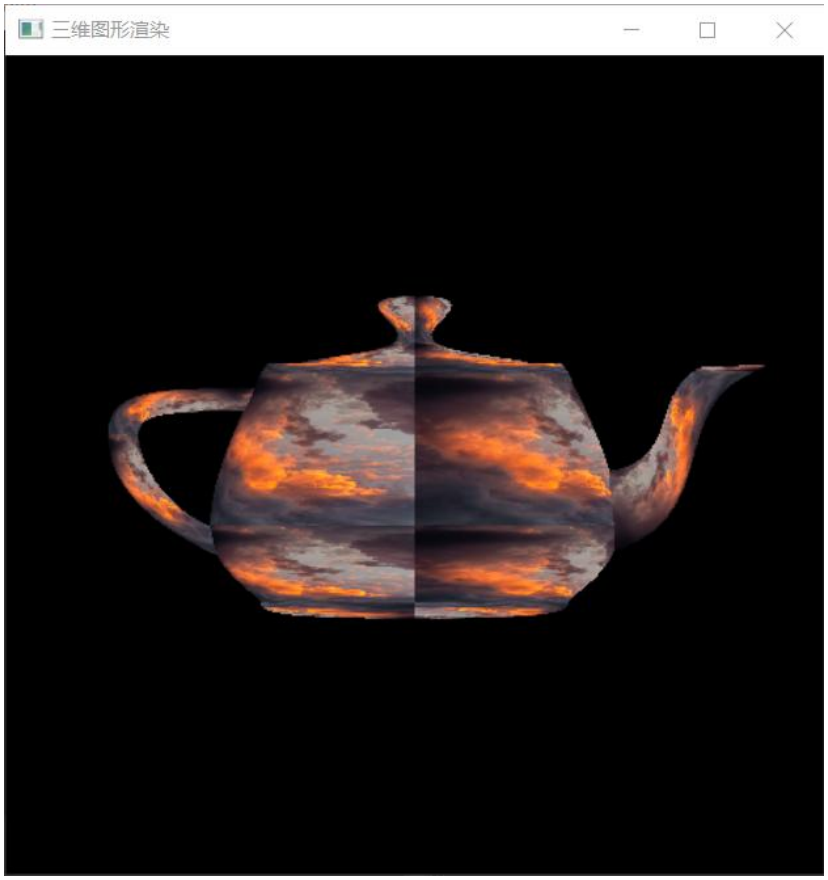

关键点在于光源的设置

鼠标交互实现图形的旋转：

当按下左键开始根据当前的 x、y 的坐标实现对 x,y,z 三个方向的旋转。

## 五、实验运行情况

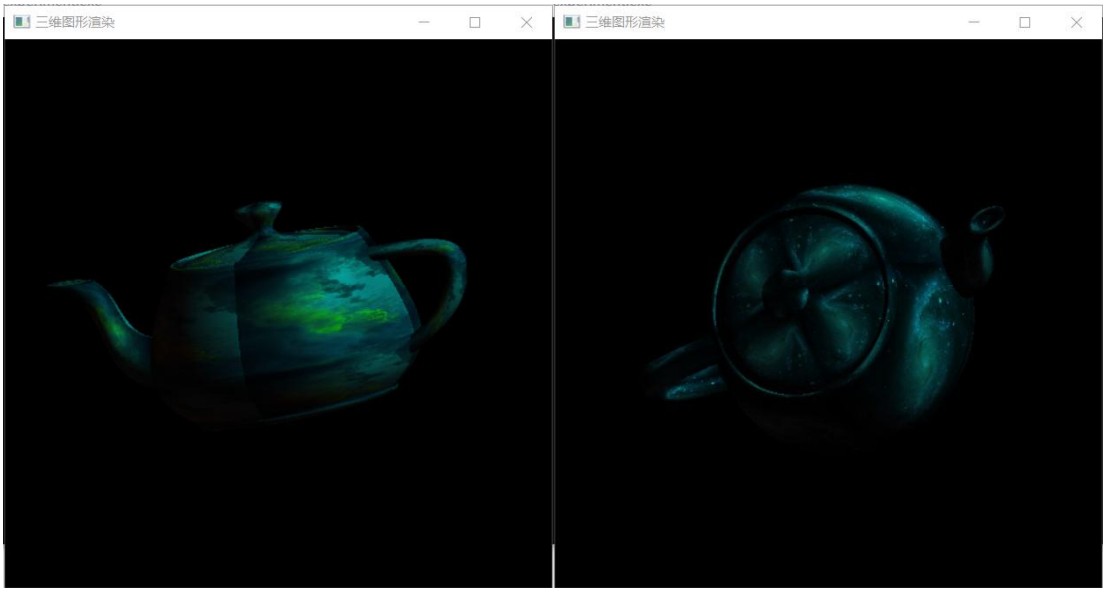
初始图像:

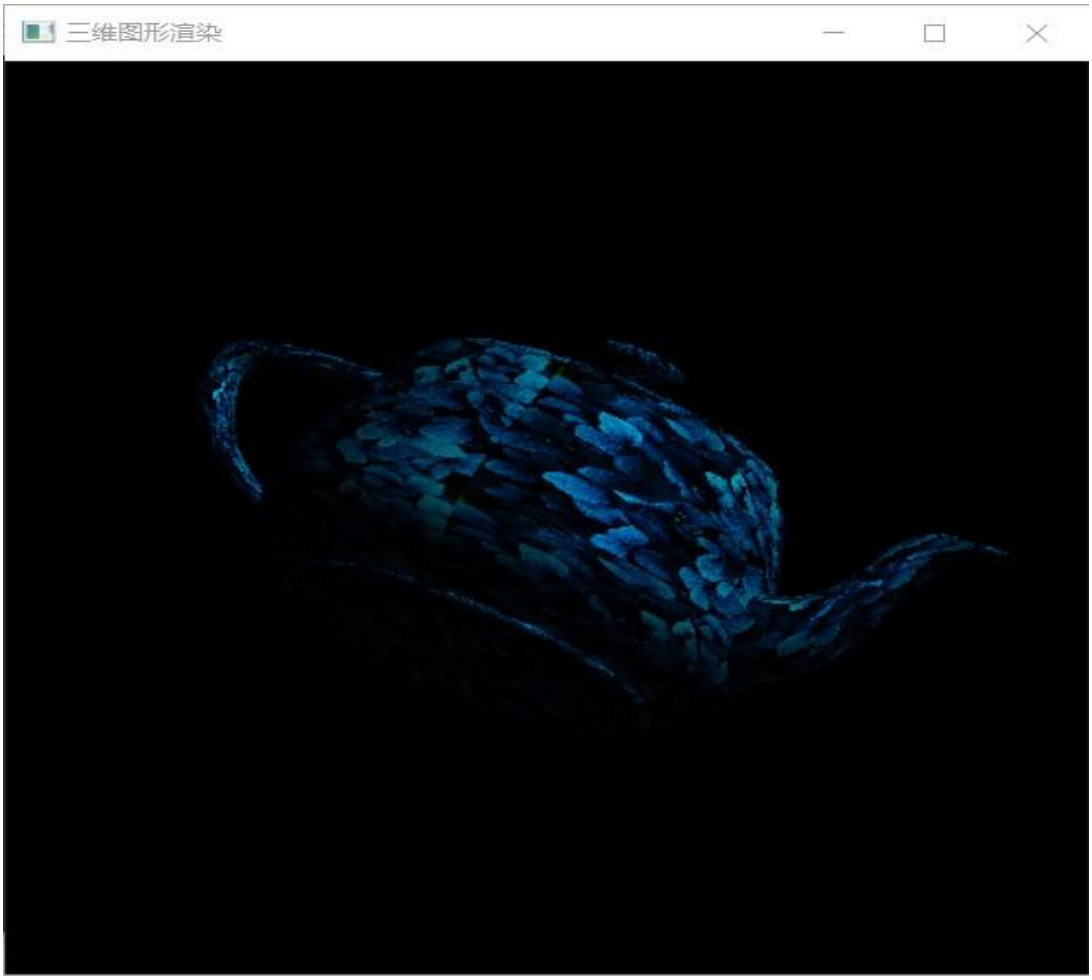



旋转:


纹理转换:


开启光照效果:




存在的问题：旋转有时较为别扭，而贴图效果不太好看

## 六、实验体会与总结

本次实验采用了光照、纹理贴图与三位观察等图形学知识。同时，在本次实验中实现了鼠标交互用鼠标的拖拽来旋转图像，让观察更为方便。而我也通过本次实验学习到了图形学光照相关知识以及贴图的方法。

当然，在实验中总是会出一些小问题，面对这些问题，我通过网上搜索的方式将错误一个一个排除，最后实现了程序的正常运行。

最后通过这次试验，我不仅加深了对图形学知识的巩固与提升，同时也知道了所想与所做天差地别，纸上谈兵不可取，要多动手。

## 参考文献

1、chy19911123.OpenGL glLightfv 函数的应用以及光源的相关知识.CSDN.2015-06-08.https://blog.csdn.net/chy19911123/article/details/46413121.

2、虚坏叔叔.《高效学习 OpenGL》 之 选择光照类型 glLightModel(), glEnable().CSDN.2014-02-20.https://blog.csdn.net/biggbang/article/details/19544405.

3、ZJU_fish1996.[OpenGL]茶壶与纹理.CSDN.2016-05-15.https://blog.csdn.net/ZJU_fish1996/article/details/51419541.

## 附录

```
#include <stdlib.h>
#include <windows.h>
#include <stdio.h>
#include <gl/glut.h>
#include <math.h>
#include<iostream>
using namespace std;
#define BITMAP_ID 0x4D42
#define Height 16
#define Width 16
float theta = 0.0, Xtheta = 0, Ytheta = 0, Ztheta = 0;//定义旋转角度、绕 x 轴旋转、绕 y 轴旋转
float oldx = 0, oldy = 0;//记录旧的点
float Rate = 1;//缩放比例
float XTran = 0, YTran = 0, ZTran = 0;//在 x 轴和 y 轴方向平移量
int winMax = 500, winMin = 500;//窗口大小
```

```
GLuint texture[4];// 纹理标示符数组，保存两个纹理的标示符
GLint ch = 0;//纹理选择
GLint light_ch = 0;//光照开关
//读纹理图片
unsigned char* LoadBitmapFile(char* filename, BITMAPINFOHEADER* bitmapInfoHeader)
{
    FILE* filePtr;    // 文件指针
    BITMAPFILEHEADER bitmapFileHeader;    // bitmap 文件头
    unsigned char* bitmapImage;        // bitmap 图像数据
    int    imageIdx = 0;        // 图像位置索引
    unsigned char    tempRGB;    // 交换变量

                                // 以“二进制+读”模式打开文件 filename
    filePtr = fopen(filename, "rb");
    if (filePtr == NULL) {
        printf("file not open\n");
        return NULL;
    }
    // 读入 bitmap 文件图
    fread(&bitmapFileHeader, sizeof(BITMAPFILEHEADER), 1, filePtr);
    // 验证是否为 bitmap 文件
    if (bitmapFileHeader.bfType != BITMAP_ID) {
        fprintf(stderr, "Error in LoadBitmapFile: the file is not a bitmap file\n");
        return NULL;
    }
    // 读入 bitmap 信息头
    fread(bitmapInfoHeader, sizeof(BITMAPINFOHEADER), 1, filePtr);
    // 将文件指针移至 bitmap 数据
    fseek(filePtr, bitmapFileHeader.bfOffBits, SEEK_SET);
    // 为装载图像数据创建足够的内存
    bitmapImage = new unsigned char[bitmapInfoHeader->biSizeImage];
    // 验证内存是否创建成功
    if (!bitmapImage) {
        fprintf(stderr, "Error in LoadBitmapFile: memory error\n");
        return NULL;
    }
    // 读入 bitmap 图像数据
    fread(bitmapImage, 1, bitmapInfoHeader->biSizeImage, filePtr);
    // 确认读入成功
    if (bitmapImage == NULL) {
        fprintf(stderr, "Error in LoadBitmapFile: memory error\n");
        return NULL;
    }
```

```
    //由于 bitmap 中保存的格式是 BGR，下面交换 R 和 B 的值，得到 RGB 格式
    for (imageIdx = 0; imageIdx < bitmapInfoHeader->biSizeImage; imageIdx += 3) {
        tempRGB = bitmapImage[imageIdx];
        bitmapImage[imageIdx] = bitmapImage[imageIdx + 2];
        bitmapImage[imageIdx + 2] = tempRGB;
    }
    // 关闭 bitmap 图像文件
    fclose(filePtr);
    return bitmapImage;
}
//加载纹理的函数
void texload(int i, char* filename)
{
    BITMAPINFOHEADER bitmapInfoHeader;                                 // bitmap 信息头
    unsigned char* bitmapData;                                        // 纹理数据

    bitmapData = LoadBitmapFile(filename, &bitmapInfoHeader);
    glBindTexture(GL_TEXTURE_2D, texture[i]);
    // 指定当前纹理的放大/缩小过滤方式
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
    glTexImage2D(GL_TEXTURE_2D,
        0,          //mipmap 层次(通常为，表示最上层)
        GL_RGB,     //我们希望该纹理有红、绿、蓝数据
        bitmapInfoHeader.biWidth, //纹理宽带，必须是 n，若有边框+2
        bitmapInfoHeader.biHeight, //纹理高度，必须是 n，若有边框+2
        0, //边框(0=无边框, 1=有边框)
        GL_RGB,     //bitmap 数据的格式
        GL_UNSIGNED_BYTE, //每个颜色数据的类型
        bitmapData);    //bitmap 数据指针
}
//初始化光源性质，初始化贴图
void init()
{
    GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat mat_shininess[] = { 50.0 };
    GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };
    GLfloat white_light[] = { 0.0, 1.0, 1.0, 1.0 };
    GLfloat Light_Model_Ambient[] = { 0.2 , 0.2 , 0.2 , 1.0 }; //
    GLfloat light_position1[] = { 0.0, -3.0, 0.0, 0.0 };
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glShadeModel(GL_SMOOTH);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);//材质属性中的镜面反射光
    glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);//材质属性的镜面反射指数
```

```
    /*glLightfv（光源编号，光源特性，参数数据）
    GL_AMBIENT（设置光源的环境光属性，默认值(0,0,0,1)）、GL_DIFFUSE（设置光源的散射光属性，默认值(1,1,1,1)）
    GL_SPECULAR（设置光源的镜面反射光属性，默认值(1,1,1,1)）、GL_POSITION（设置光源的位置，默认值(0,0,1,0)）*/
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);//设置 0 号光源位置
    glLightfv(GL_LIGHT0, GL_DIFFUSE, white_light);//设置 0 号光源散射光属性
    glLightfv(GL_LIGHT0, GL_SPECULAR, white_light);//设置 0 号镜面反射光属性
    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, Light_Model_Ambient);
    glEnable(GL_DEPTH_TEST);

    glGenTextures(4, texture); // 第一参数是需要生成标示符的个数, 第二参数是返回标示符的数组
    texload(3, "4.bmp");
    texload(1, "2.bmp");
    texload(2, "3.bmp");
}
//绘制茶壶
void drawTeapot()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glEnable(GL_TEXTURE_2D);
    glPushMatrix();
    if(ch!=0){
    glBindTexture(GL_TEXTURE_2D, texture[ch]);  //选择纹理 texture[status]
    glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);//设置纹理受光照影响
    glutSolidTeapot(1);
    glPopMatrix();
    glDisable(GL_TEXTURE_2D);    //关闭纹理 texture[status]
    }
    else
        glutSolidTeapot(1);
    glFlush();
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); //清空颜色和深度缓存
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    glTranslatef(0.0f, 0.0f, -4.0f);//平移三维裁剪窗口，让窗口能完全包含住物体
    glRotatef(Xtheta, 0.0f, 1.0f, 0.0f);//根据鼠标移动距离旋转物体
    glRotatef(Ytheta, 1.0f, 0.0f, 0.0f);
```

```
    glRotatef(Ztheta, 0.0f, 0.0f, 1.0f);
    drawTeapot();
    glutSwapBuffers();
}

void reshape(int w, int h) //重绘回调函数，在窗口首次创建或用户改变窗口尺寸时被调用
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    if (w <= h)

        glOrtho(-1.5, 1.5, -1.5 * (GLfloat)h / (GLfloat)w,

            1.5 * (GLfloat)h / (GLfloat)w, -10.0, 10.0);

    else

        glOrtho(-1.5 * (GLfloat)w / (GLfloat)h,

            1.5 * (GLfloat)w / (GLfloat)h, -1.5, 1.5, -10.0, 10.0);

    glLoadIdentity();
    glOrtho(-2.0, 2.0, -2.0, 2.0, 0.0, 10.0);
}

void motionROt(int x, int y)//实现摁住时，计算旋转量，实现旋转
{
    GLint deltax = x - oldx;
    GLint deltay = y - oldy;
    Xtheta += 360 * (GLfloat)deltax / (GLfloat)winMax/5;//根据屏幕上鼠标滑动的距离来设置
旋转的角度
    Ytheta += 360 * (GLfloat)deltay / (GLfloat)winMin / 5;
    Ztheta += 360 * (GLfloat)deltay / (GLfloat)winMin / 5;
    oldx = x;//记录此时的鼠标坐标，更新鼠标坐标
    oldy = y;//若是没有这两句语句，滑动是旋转会变得不可控
    display();
}

void mouseMove(int button, int state, int x, int y)//实现摁住鼠标移动观察茶壶
{
    oldx = x;//当左键按下时记录鼠标坐标
    oldy = y;
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
    {
```

```
            glutMotionFunc(motionROt);
        }
        if (button == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
        {
            if (light_ch == 1) {
                glDisable(GL_LIGHTING);
                cout << "关闭光照" << endl;
                    light_ch = 0;
            }
            else {
                glEnable(GL_LIGHTING);//开启光照效果
                cout << "开启光照" << endl;
                light_ch = 1;
            }

            glEnable(GL_LIGHT0);//开启 0 号光源
            display();
        }
        if (button == GLUT_MIDDLE_BUTTON && state == GLUT_DOWN)
        {
            if (ch == 3)
                ch = 0;
            else
               ch++;
            display();
        }
    }

    int main(int argc, char** argv)
    {
        cout << "鼠标左键实现茶壶的旋转" << endl;
        cout << "鼠标右键实现灯光的开关" << endl;
        cout << "鼠标中键实现贴图的转换" << endl;
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
        glutInitWindowSize(winMax, winMin);
        glutInitWindowPosition(100, 200);
        glutCreateWindow("三维图形渲染");
        glutReshapeFunc(reshape); //指定重绘回调函数
        glutDisplayFunc(display);
        glutMouseFunc(mouseMove);
        init();
        glutMainLoop();
    }
```