**《计算机图形学实验》综合实验报告**

**题目 三维图形渲染**

**学 号 20201060282**

**姓 名 单子奇**

**指导教师 钱文华**

**日 期 2022.6.22**

**摘要：**通过这一个学期的学习，利用Visual Studio Code，实现三维图形——茶壶的渲染，渲染过程须加入了色彩、光照等效果。学会了三维图形的绘制，添加颜色及光照等操作，为将来的深入学习奠定了基础。

**关键词：OpenGL、Visual Studio Code**

**目录**

[一、实验背景 2](#_Toc106726631)

[二、实验内容以及开发工具 2](#_Toc106726632)

[三、实验介绍 3](#_Toc106726633)

[四、实验结果与分析 3](#_Toc106726634)

[五、实验体会与小结 4](#_Toc106726635)

[六、参考文献 4](#_Toc106726636)

[七、附录 4](#_Toc106726637)

## 一、实验背景

当今，OpenGL是视频行业领域中用于处理2D/3D图形的最为广泛接纳的API，在此基础上，为了用于计算机视觉技术的研究，从而催生了各种计算机平台上的应用功能以及设备上的许多应用程序。

通过这一个学期的学习，我学到了好多相关知识，因此希望能够自己编写代码实现自定义三维图形渲染。

## 二、实验内容以及开发工具

**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。

**开发工具：**Visual Studio Code

## 三、实验介绍

程序的实现步骤

开始

初始化窗口

画茶壶

并添加颜色

确定光源位置，添加光照效果

设置反射系数、环境光系数、材料反射指数

调整视区大小保持与窗口的显示区域大小一致

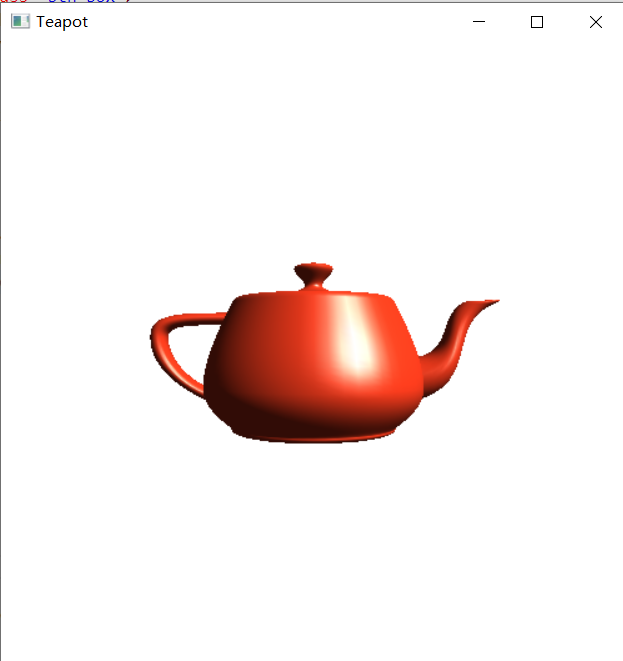
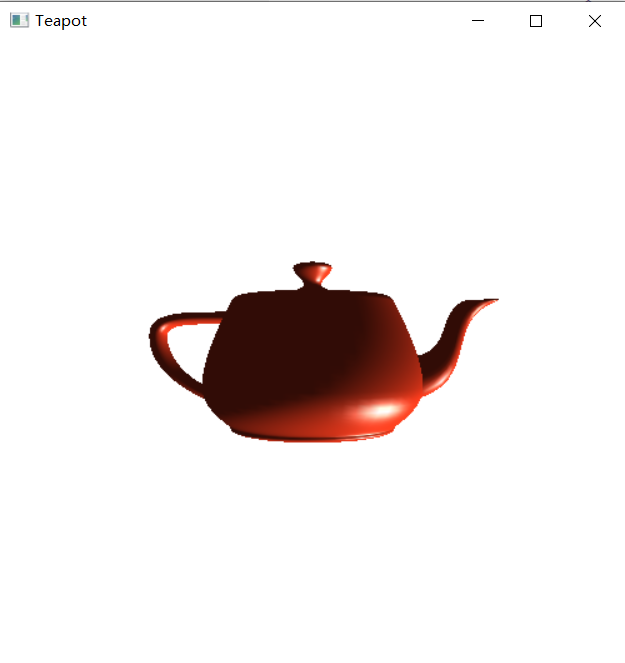
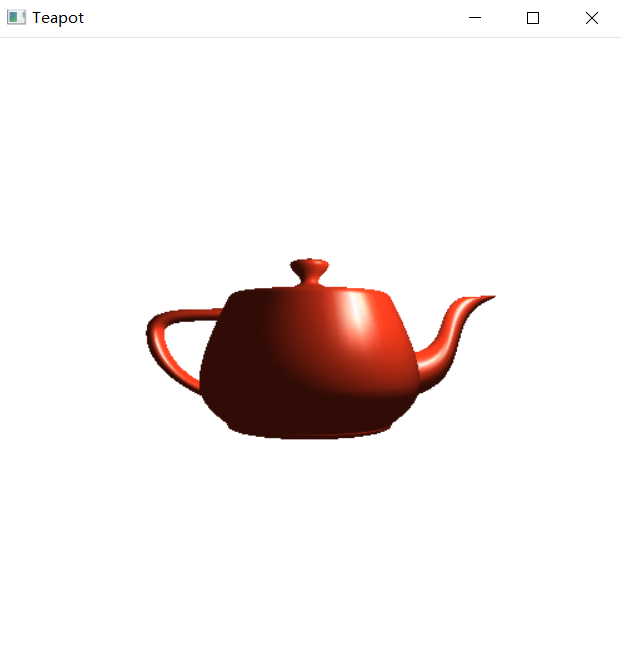
建立从世界坐标系到观察坐标系的转换的矩阵

在观察坐标系下根据透视投影矩阵进行投影变换

结束

## 四、实验结果与分析

**不同的光照角度**

**** ****

## 五、实验体会与小结

1.利用Visual Studio Code，实现了三维图形渲染，自定义三维图形，三维图形不仅仅是简单的茶壶，渲染过程须加入了色彩、光照等效果。

2.通过本次实验，我学会了三维图形的绘制，添加颜色及光照等操作，为将来的深入学习奠定了不可磨灭的基础。

3.真实感图形绘制是计算机图形学的一个重要组成部分。绘制的图形要尽可能立体，这时便需要我们加入光照，这样才能让我们做出的模型更加立体。

## 六、参考文献

[1]官酩杰. 基于OpenGL ES的移动平台图形渲染引擎研究与实现[D].北京交通大学,2010.

[2]章义来,张战成,吴敏纲,李程扬.建筑陶瓷三维装饰效果平台的研究与开发——实体建模与渲染模块设计[J].计算机辅助工程,2001(03):18-22.

[3]罗仁,李子轩.基于OpenGL的三维实时渲染引擎[J].信息与电脑(理论版),2021,33(18):112-115.

## 七、附录

#include <windows.h>

#include <GL/glut.h>

#include <stdlib.h>

static int year = 0, day = 0;

static int longitude = 200, latitude=200;

void init(void)

{

glClearColor (1.0, 1.0, 1.0, 0.0);

//多边形内部点的颜色通过对多边形顶点颜色插值产生

glShadeModel (GL\_SMOOTH);

glEnable(GL\_DEPTH\_TEST);

glDepthFunc(GL\_LESS);

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_FILL);

//漫反射系数，对应RGBA四个分量，A分量在混合开(glBlend)启后有效

GLfloat mat\_diffuse[] = { 0.96, 0.23, 0.11, 1.0 };

//镜面反射系数,对应RGBA四个分量，A分量在混合开启后有效

GLfloat mat\_specular[] = { 1.0, 1.0, 1.0, 1.0 };

GLfloat mat\_shininess[] = { 50.0 };

//设置环境光系数和漫反射光系数

glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, mat\_diffuse);

//set material镜面光反射

glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);

//设置材料反射指数

glMaterialfv(GL\_FRONT, GL\_SHININESS, mat\_shininess);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

}

void display(void)

{

glClear (GL\_COLOR\_BUFFER\_BIT |GL\_DEPTH\_BUFFER\_BIT );

//光源在无穷远处

GLfloat light\_position[] = { 1.0, 1.0, 1.0, 0.0 };

//光源位置

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);

//启用光源后，不在使用颜色函数对物体着色

glColor3f (1.0, 1.0, 1.0);

//使用模型视图矩阵栈保存后面的几何变换

glMatrixMode(GL\_MODELVIEW);

//保存世界坐标系到观察坐标系的变换矩阵

glPushMatrix();

glutSolidTeapot(1.0);

//恢复世界坐标系到观察坐标系的变换矩阵

glPopMatrix();

glutSwapBuffers();

}

void reshape (int w, int h)

{

//使视区大小保持与窗口的显示区域大小一致

glViewport (0, 0, (GLsizei) w, (GLsizei) h);

//使用模型视图矩阵栈

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

//设定观察视点，和观察方向

gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

//使用投影矩阵栈，准备设定投影矩阵

glMatrixMode (GL\_PROJECTION);

//初始化投影矩阵为单位矩阵

glLoadIdentity ();

// gluPerspective函数定义视景体（裁剪棱台），会根据四个参数计算用以投影变换的透视投影矩阵

gluPerspective(60.0, (GLfloat) w/(GLfloat) h, 0.5, 30.0);

}

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode (GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize (500, 500);

glutInitWindowPosition (100, 100);

glutCreateWindow("Teapot");

init ();

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutMainLoop();

return 0;

}