**《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 20201060336**

**姓 名 许豪**

**指导教师 钱文华**

**日 期 2022.6.15**

**摘要**

本实验利用Visual Studio进行开发，使用OpenGL实现三维图形渲染，基于一个茶壶的三维图形进行渲染，渲染过程加入纹理、光照、旋转、键鼠交互等效果，采用了光照明模型、纹理贴图、键鼠交互等算法，实现了程序。

**关键词：**OpenGL；三维图形；茶壶；渲染；键鼠交互

目录

**[1.](#_Toc105612602)****[实验背景](#_Toc105612602)** [4](#_Toc105612602)

**[2.](#_Toc105612603)****[实验内容](#_Toc105612603)** [4](#_Toc105612603)

**[3.](#_Toc105612604)****[关键代码及算法理论](#_Toc105612604)** [6](#_Toc105612604)

**[4.](#_Toc105612605)****[运行结果](#_Toc105612605)** [7](#_Toc105612605)

**[5.](#_Toc105612606)****[实验心得及小结](#_Toc105612606)** [9](#_Toc105612606)

**[参考文献：](#_Toc105612607)** [9](#_Toc105612607)

**[附录：](#_Toc105612608)** [1](#_Toc105612608)0

1. **实验背景**

三维场景渲染是计算机从三维场景内获取模型、材质和光照等基本信息并通过复杂计算输出真实感高图像的过程。三维场景渲染是三维图形平台内最重要的模块之一，主要负责对图形的组织、管理与显示，实际上是三维真实感图形的再现过程，具体表现在光线处理、纹理处理和图形显示处理。在许多应用领域对场景的实时性有较高的要求，如果三维场景显示达不到一定的显示速率，就会丧失实际应用的价值，如在设计过程中，输入的基本信息无法在视图内实时展现，则无法满足后续计算、仿真的要求。

随着计算机图形的的不断进化，三维制作技术的应用也越来越广泛。目前在影视动画，游戏，教育，建筑动画，视觉可视化等行业都在使用这种技术。并且随着科技的发展，越来越多的新技术被研发并应用到新的行业中。

**二、实验内容**

**1.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，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**

开发工具：Visual C++, OpenGL, Java等

目的:设计一个茶壶。运用计算机图形学理论、算法、技术等按照要求设计并实现程序，撰写实验报告。

**3.实现成果**

本实验利用Visual Studio进行开发，使用OpenGL实现三维图形渲染，基于一个茶壶的三维图形进行渲染，渲染过程加入纹理、光照、旋转、键鼠交互等效果，采用了光照明模型、纹理贴图、键鼠交互等算法，实现了程序。在这个基础上我还建立了一个较为复杂的模型，但是对其的渲染很复杂，所只做了个基本功能实现的程序，并没有实现对其进行交互式操作。

**三、关键代码和算法理论**

void glutMouseFunc(void(\*func)(int button,int state,int x,int y)); 要想在OpenGL中处理鼠标事件非常的方便，GLUT已经为我们的注册好了函数，只要我们提供一个方法。使用函数glutMouseFunc,就可以帮我们注册我们的函数，这样当发生鼠标事件时就会自动调用我们的方法。

void glRotatef(GLfloat angle,GLfloat x,GLfloat y,GLfloat z);先解释一下旋转方向，做(0,0,0)到(x,y,z)的向量，用右手握住这条向量，大拇指指向向量的正方向，四指环绕的方向就是旋转的方向；函数功能，以点(0,0,0)到点(x,y,z)为轴，旋转angle角度； 至于其他的旋转平移也只是调用函数即可。

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);

其实也注意到实验中存在光照，所以我们有必要指定纹理和光照是什么关系，在这里我们用了如下两个参数，一个保证纹理和光照混合，一个单纯绘制纹理而不考虑光照。glTexEnvf( GL\_TEXTURE\_ENV , GL\_TEXTURE\_ENV\_MODE , GL\_MODULATE ); // 设置纹理受光照影响

glTexEnvf(GL\_TEXTURE\_ENV,GL\_TEXTURE\_ENV\_MODE,GL\_DECAL);//设置纹理不受光照影响

至于键鼠交互部分，调用函数对鼠标进行绑定即可。

glutInit()是用glut来初始化OpenGL的，我将把所有的事情都留给这个函数，这基本上是无关紧要的，尽管它有参数，这基本上是无用的。

颜色、光照和材质的函数。如设置颜色模式函数glColor\*()、glIndex\*()，设置光照效果的函数glLight\*() 、glLightModel\*()和设置材质效果函数glMaterial()等等。

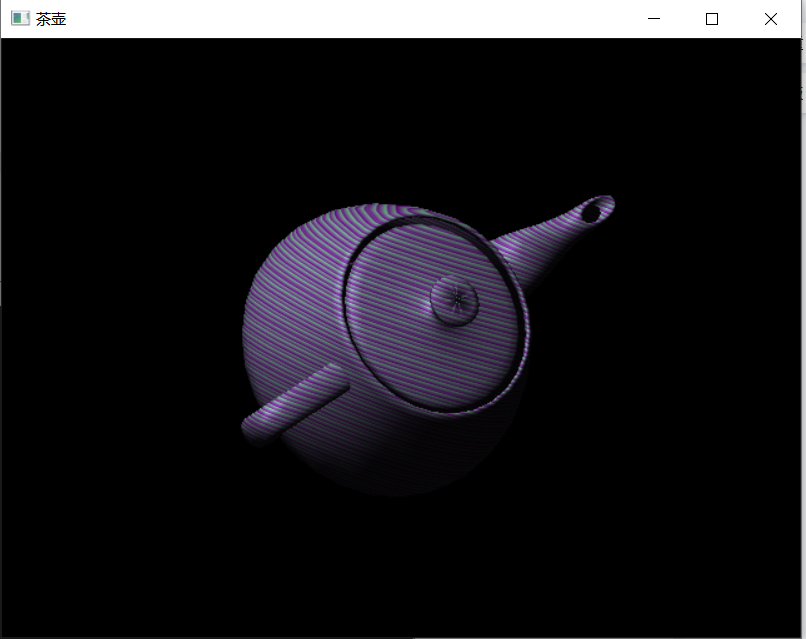
glutInitDisplayMode设置显示模式，其中的GLUT\_DOUBLE表示双缓冲。双缓冲指的是在绘制图形的时候先在一个缓冲区把它绘制好，绘图指令完成后再通过交换指令把它呈现到屏幕上，这样绘制效率较高。OpenGL一般采用双缓冲，如果只是显示静态图片，也可以用单缓冲模式。但我们设置了双缓冲模式以后，在绘图的函数里一定会加上这个函数。glutReshapeFunc是用来调整图形比例的。

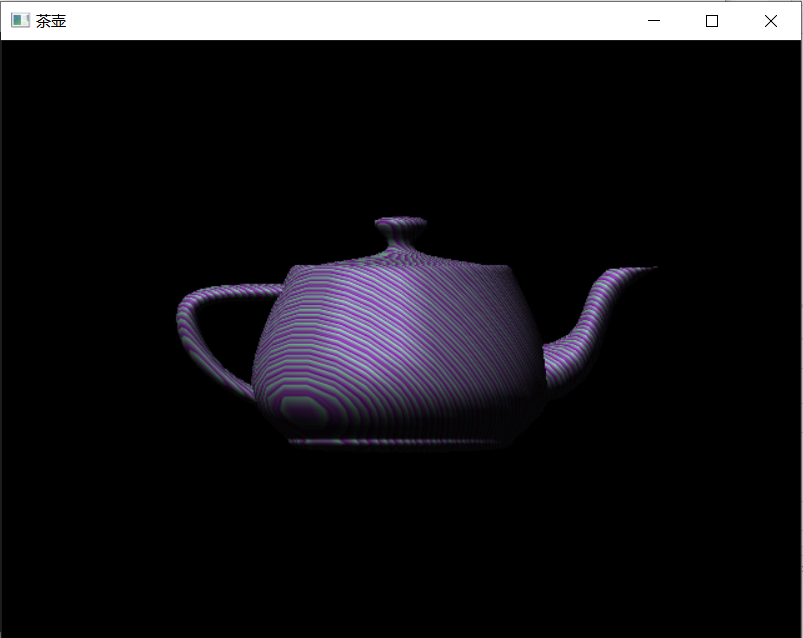
**四、运行结果**

**1.光照效果**



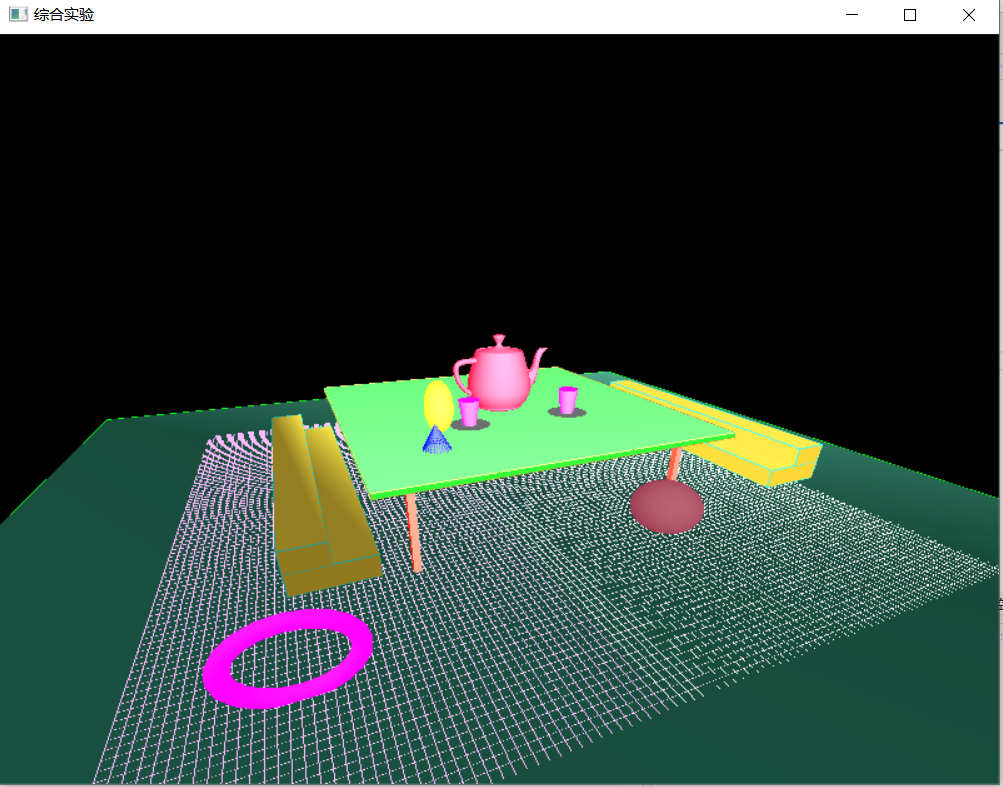
**2.纹理映射（更改参数变化）**





****

**拓展**



**五、实验心得和小结**

这个实验的流程大体是，首先初始化窗口，定义坐标与视口坐标，利用之前实验的知识内容绘制三维立体（茶壶）模型，定义每一个模型的坐标值以及利用变换函数实现相对位置的放置；利用之前实验中的知识给整个场景进行灯光的设置（光照）；然后实现模型材质的纹理贴图；最后进行鼠标与键盘的交互，实现鼠标可以控制茶壶的选择移动。在这个实验中，还是借鉴了老师和网上的资源，关键在于算法的实验，这个还好，有现成的资源，其实自己关键就在于的各个函数的理解级运用。

本实验利用Visual Studio进行开发，使用OpenGL实现三维图形渲染，基于一个茶壶的三维图形进行渲染，渲染过程加入纹理、光照、旋转、键鼠交互等效果，采用了光照明模型、纹理贴图、键鼠交互等算法，实现了程序。在这个基础上我还建立了一个较为复杂的模型，但是对其的其他可变性的渲染很复杂，只做了个基本功能实现的程序。由于目前还是技术和知识不够，茶壶的纹理虽然有艺术感，但还是做不出非常有高级感真实感的材质，以后还需要继续深入学习。但是这始终是我学习OpenGL后第一次完整的写出的一个比较大的完整的程序，值得纪念。

参考文献

1. 徐凤雪.OpenGL中三维图形的显示过程[J].电子技术与软件工程,2020(02):66-68.
2. 程飞.基于OpenGL的交互式科普平台设计研究[J].西昌学院学报(自然科学版),2018,32(04):78-81.DOI:10.16104/j.issn.1673-1891.2018.04.019.
3. 褚宏光. 基于OpenGL的羽毛双向纹理采样[D].广西师范大学,2018.

附录（详细见github）

#include <iostream>

#include<GL/glaux.h>

#include<gl/glut.h>

#include<stdio.h>

#include<math.h>

GLuint drawcube, drawsphere, drawteapot;

static GLfloat sx = 0, sy = 0, sz = 0;

static int lflag = GL\_TRUE, rflag = GL\_TRUE;

void reshape(int w, int h)

{

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

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(90, w / h, 0.5, 200); //透视效果

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

gluLookAt(-30, 30, 70, 0, 0, 0, 0, 10, 0);//设置观察点

}

void init()

{

glClearColor(0.0, 0.0, 0.0, 0.0);

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

//画出正方体显示列表

drawcube = glGenLists(1);

glNewList(drawcube, GL\_COMPILE);

glutSolidCube(1);

glEndList();

//画出球体显示列表

drawsphere = glGenLists(1);

glNewList(drawsphere, GL\_COMPILE);

glutSolidSphere(1, 20, 20);

glEndList();

//画出茶壶显示列表

drawteapot = glGenLists(1);

glNewList(drawteapot, GL\_COMPILE);

glutSolidTeapot(1);

glEndList();

//设置普通灯光照0位置及参数；

GLfloat position0[] = { 30,5,30,1 };

GLfloat light0s[] = { 0.10,0.10,0.10,0.0 };

GLfloat light0d[] = { 0.6,0.7,0.7 };

GLfloat light0a[] = { 0.9,0.9,0.9 };

glLightfv(GL\_LIGHT0, GL\_POSITION, position0);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, light0s);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light0d);

glLightfv(GL\_LIGHT0, GL\_AMBIENT, light0a);

//设置探照灯光照1位置及参数

GLfloat position1[] = { -60,40,0,1 };

GLfloat light1s[] = { 1.0,1.0,1.0,1.0 };

GLfloat light1d[] = { 0.06,0.1,0.1 };

GLfloat light1a[] = { 0.91,0.99,0.96 };

GLfloat direction[] = { 0,-60,0,1 };

glLightfv(GL\_LIGHT1, GL\_SPOT\_DIRECTION, direction);

glLightf(GL\_LIGHT0, GL\_SPOT\_CUTOFF, 30.0);

glLightfv(GL\_LIGHT1, GL\_POSITION, position1);

glLightfv(GL\_LIGHT1, GL\_SPECULAR, light1s);

glLightfv(GL\_LIGHT1, GL\_DIFFUSE, light1d);

glLightfv(GL\_LIGHT1, GL\_AMBIENT, light1a);

glEnable(GL\_LIGHTING);

glEnable(GL\_DEPTH\_TEST);

//glEnable(GL\_LIGHT0);

//glEnable(GL\_LIGHT1);

}

void SpecialKeys(int key, int x, int y)

{

if (key == GLUT\_KEY\_UP)

{

sy -= 5.50f;

glutPostRedisplay();

}

if (key == GLUT\_KEY\_DOWN)

{

sy += 5.50f;

glutPostRedisplay();

}

if (key == GLUT\_KEY\_LEFT)

{

sx -= 0.50f;

glutPostRedisplay();

}

if (key == GLUT\_KEY\_RIGHT)

{

sx += 0.50f;

// sx= (GLfloat)((const int)sx % 360);

// sy = (GLfloat)((const int)sy % 360);

glutPostRedisplay();

}

}

void draw()

{

if (lflag)

glEnable(GL\_LIGHT0);

if (rflag)

glEnable(GL\_LIGHT1);

glPushMatrix();

glTranslatef(sx, sy, sz);

//绘制地面；

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColorMaterial(GL\_FRONT, GL\_AMBIENT);

glColor4f(0.05, 0.2, 0.15, 0.0);

glTranslatef(0, -20, 0);

glScalef(150, 1, 150);

glCallList(drawcube);

glColor4f(0.0, 1.0, 0.0, 0.0);

glutWireCube(1);

glDisable(GL\_COLOR\_MATERIAL);

glPopMatrix();

//装饰地面

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColorMaterial(GL\_FRONT, GL\_AMBIENT);

glColor4f(0.9, 0.6, 0.99, 0.0);

glBegin(GL\_LINES);

float y = -19.5, z1 = 50;

for (float x1 = -50; x1 < 50; ++x1)

{

glVertex3f(x1, y, z1);

glVertex3f(x1, y, -z1);

}

float x2 = 50;

for (float z2 = -50; z2 < 50; ++z2)

{

glVertex3f(x2, y, z2);

glVertex3f(-x2, y, z2);

}

glEnd();

glPopMatrix();

//绘制四个桌腿：

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColorMaterial(GL\_FRONT, GL\_AMBIENT);

glColor4f(1.0, 0.05, 0.0, 0.0);

glTranslatef(20, -10, -20);

glScalef(1, 20, 1);

glCallList(drawcube);

glDisable(GL\_COLOR\_MATERIAL);

glPopMatrix();

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColorMaterial(GL\_FRONT, GL\_AMBIENT);

glColor4f(1.0, 0.05, 0.0, 0.0);

glTranslatef(-20, -10, -20);

glScalef(1, 20, 1);

glCallList(drawcube);

glDisable(GL\_COLOR\_MATERIAL);

glPopMatrix();

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColorMaterial(GL\_FRONT, GL\_AMBIENT);

glColor4f(1.0, 0.05, 0.0, 0.0);

glTranslatef(-20, -10, 20);

glScalef(1, 20, 1);

glCallList(drawcube);

glDisable(GL\_COLOR\_MATERIAL);

glPopMatrix();

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColorMaterial(GL\_FRONT, GL\_AMBIENT);

glColor4f(1.0, 0.05, 0.0, 0.0);

glTranslatef(20, -10, 20);

glScalef(1, 20, 1);

glCallList(drawcube);

glDisable(GL\_COLOR\_MATERIAL);

glPopMatrix();

//画出桌面：

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColorMaterial(GL\_FRONT, GL\_AMBIENT);

glColor4f(0.1, 1.0, 0.1, 0.0);

glScalef(50, 1, 50);

glCallList(drawcube);

glColor4f(0.39, 0.30, 0.1, 0.0);

glutWireCube(1);

// glFlush();

glDisable(GL\_COLOR\_MATERIAL);

glPopMatrix();

//绘制两边台阶；

glPushMatrix();//左边第一级台阶

glEnable(GL\_COLOR\_MATERIAL);

glColorMaterial(GL\_FRONT, GL\_AMBIENT);

glColor4f(0.51, 0.40, 0.1, 0.0);

glTranslatef(-30, -12, 0);

glScalef(10, 4, 50);

glCallList(drawcube);

glColorMaterial(GL\_FRONT, GL\_AMBIENT);

glColor4f(0.1, 0.50, 0.51, 0.0);

glutWireCube(1);

glDisable(GL\_COLOR\_MATERIAL);

glPopMatrix();

glPushMatrix();// 右边第一级台阶

glEnable(GL\_COLOR\_MATERIAL);

glColorMaterial(GL\_FRONT, GL\_AMBIENT);

glColor4f(0.51, 0.40, 0.1, 0.0);

glTranslatef(40, -12, 0);

glScalef(10, 4, 50);

glCallList(drawcube);

glColorMaterial(GL\_FRONT, GL\_AMBIENT);

glColor4f(0.1, 0.50, 0.51, 0.0);

glutWireCube(1);

glDisable(GL\_COLOR\_MATERIAL);

glPopMatrix();

glPushMatrix();//左边第二级台阶

glEnable(GL\_COLOR\_MATERIAL);

glColorMaterial(GL\_FRONT, GL\_AMBIENT);

glColor4f(0.51, 0.40, 0.1, 0.0);

glTranslatef(-32.5, -8, 0);

glScalef(5, 4, 50);

glCallList(drawcube);

glColorMaterial(GL\_FRONT, GL\_AMBIENT);

glColor4f(0.1, 0.50, 0.51, 0.0);

glutWireCube(1);

glDisable(GL\_COLOR\_MATERIAL);

glPopMatrix();

glPushMatrix();//右边第二级台阶

glEnable(GL\_COLOR\_MATERIAL);

glColorMaterial(GL\_FRONT, GL\_AMBIENT);

glColor4f(0.51, 0.40, 0.1, 0.0);

glTranslatef(42.5, -8, 0);

glScalef(5, 4, 50);

glCallList(drawcube);

glColorMaterial(GL\_FRONT, GL\_AMBIENT);

glColor4f(0.1, 0.50, 0.51, 0.0);

glutWireCube(1);

glDisable(GL\_COLOR\_MATERIAL);

glPopMatrix();

//桌面上摆放物品，茶壶一只，鸡蛋一个

glPushMatrix();//茶壶

glEnable(GL\_COLOR\_MATERIAL);

glColorMaterial(GL\_FRONT, GL\_AMBIENT);

glColor4f(0.51, 0.0, 0.1, 0.0);

glTranslatef(0, 8, 0);

glScalef(5, 10, 5);

glCallList(drawteapot);

glDisable(GL\_COLOR\_MATERIAL);

glPopMatrix();

//glEnable(GL\_BLEND);

//glBlendFunc(GL\_SRC\_ALPHA,GL\_ONE\_MINUS\_SRC\_ALPHA);

glPushMatrix();//鸡蛋

glEnable(GL\_COLOR\_MATERIAL);

glColorMaterial(GL\_FRONT, GL\_AMBIENT);

glColor4f(1.00, 0.91, 0.05, 0.8);

glTranslatef(-13, 6, 10);

glScalef(2, 5, 2);

glCallList(drawsphere);

glDisable(GL\_COLOR\_MATERIAL);

glPopMatrix();

glPushMatrix();//地面上球体

glEnable(GL\_COLOR\_MATERIAL);

glColorMaterial(GL\_FRONT, GL\_AMBIENT);

glColor4f(0.30, 0.11, 0.15, 0.8);

glTranslatef(15, -13, 25);

glScalef(5, 5, 5);

glCallList(drawsphere);

glDisable(GL\_COLOR\_MATERIAL);

glPopMatrix();

glPushMatrix();//圆环一个

glEnable(GL\_COLOR\_MATERIAL);

glColorMaterial(GL\_FRONT, GL\_AMBIENT);

glColor4f(0.90, 0.01, 0.95, 0.8);

glRotatef(1, 0, 0, 45);

glTranslatef(-35, -13, 35);

glutSolidTorus(1, 6, 20, 20);

glDisable(GL\_COLOR\_MATERIAL);

glPopMatrix();

GLUquadricObj\* pObj;

glPushMatrix();//第一个杯子 左边

pObj = gluNewQuadric();

gluQuadricNormals(pObj, GLU\_SMOOTH);

glTranslatef(-8, 6, 8);

glRotatef(90, 1, 0, 0);

gluCylinder(pObj, 1.50f, 1.0f, 5.0f, 26, 13);

glPopMatrix();

glPushMatrix(); //第2个杯子 右边

glTranslatef(8, 6, 8);

glRotatef(90, 1, 0, 0);

gluCylinder(pObj, 1.50f, 1.0f, 5.0f, 26, 13);

glPopMatrix();

glPushMatrix(); //桌面上垫圈

glEnable(GL\_COLOR\_MATERIAL);

glColorMaterial(GL\_FRONT, GL\_AMBIENT);

glColor4f(0.21, 0.21, 0.21, 0.0);

pObj = gluNewQuadric();

//gluQuadricDrawStyle(pObj3,GLU\_LINE);

glTranslatef(-8, 1, 8);

glRotatef(90, 1, 0, 0);

gluDisk(pObj, 0.50f, 3.0f, 10, 10);

glPopMatrix();

glPushMatrix(); //桌面上垫圈

pObj = gluNewQuadric();

glTranslatef(8, 1, 8);

glRotatef(90, 1, 0, 0);

gluDisk(pObj, 0.50f, 3.0f, 30, 30);

glPopMatrix();

glPushMatrix(); //地面上放置 圆锥体；

glEnable(GL\_COLOR\_MATERIAL);

glColorMaterial(GL\_FRONT, GL\_AMBIENT);

glColor4f(0.01, 0.01, 0.81, 0.0);

pObj = gluNewQuadric();

gluQuadricDrawStyle(pObj, GLU\_LINE);

glTranslatef(-15.0, 5.0, 15);

glRotatef(90, 1, 0, 0);

gluCylinder(pObj, 0.0f, 2.0f, 5.0f, 26, 13);

glPopMatrix();

glPopMatrix();

glutSwapBuffers();

}

void main(int argc, char\* argv[])

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize(800, 600);

glutInitWindowPosition(100, 100);

glutCreateWindow("综合实验");

init();

glutReshapeFunc(reshape);

glutSpecialFunc(SpecialKeys);

// glutMouseFunc(mouse);

glutDisplayFunc(draw);

glutMainLoop();

}