**计算机图形学**

**实验报告**

**exp3**

**目录**

[**Task1** 1](#_Toc509145986)

[功能说明： 1](#_Toc509145987)

[效果预览： 1](#_Toc509145988)

[源代码： 1](#_Toc509145989)

[**Task2** 5](#_Toc509145990)

[功能说明： 5](#_Toc509145991)

[效果预览： 5](#_Toc509145992)

[源代码： 5](#_Toc509145993)

**Task1**

功能说明：

1. 设置合适的相机位置和相机投影矩阵，使得OpenGL相机能够“看到”需要绘制的物体。

2. 简单地绘制一个正方体，要求

a. 总共6个面，每个面用2个三角形来表示

b. 为每个顶点指定颜色

c. 渲染模式分别采用smooth模式和flat模式 （请查glShadeMode函数）

d. 理解深度缓冲区的作用、用法，比较开启/不开启深度缓冲区（ glEnable(GL\_DEPTHTEST) ）的效果

效果预览：

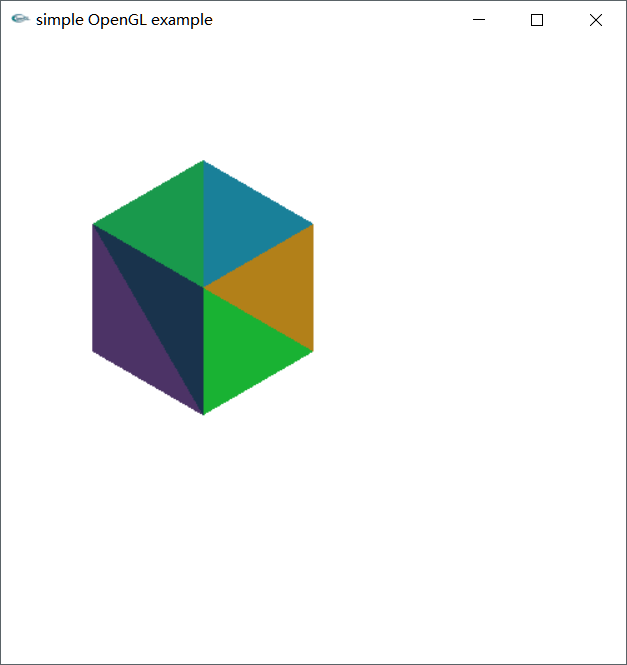


图1.Task1程序预览

源代码：

/\*

1. 设置合适的相机位置和相机投影矩阵，使得OpenGL相机能够“看到”需要绘制的物体。

2. 简单地绘制一个正方体，要求

a. 总共6个面，每个面用2个三角形来表示

b. 为每个顶点指定颜色

c. 渲染模式分别采用smooth模式和flat模式 （请查glShadeMode函数）

d. 理解深度缓冲区的作用、用法，比较开启/不开启深度缓冲区（ glEnable(GL\_DEPTHTEST) ）的效果

\*\*\*\*\*\*\*

\* @author：宋灵冰

\*/

#include <cmath>

#include <cstdlib>

#include <gl/glut.h>

//设置初始属性

void myinit() {

//设置背景为白色

glClearColor(1.0, 1.0, 1.0, 1.0);

//启用深度缓存

glEnable(GL\_DEPTH\_TEST);

//设置观察参数

glMatrixMode(GL\_MODELVIEW);

//初始化坐标系

glLoadIdentity();

//设置当前相机的位置和方向

gluLookAt(

0.0, 0.0, 0.0,

1.0, 1.0, 1.0,

-1.0, 1.0, -1.0);

glOrtho(-100.0, 100.0, -100.0, 100.0, -100.0, 100);

glMatrixMode(GL\_MODELVIEW);

glShadeModel(GL\_SMOOTH);

}

void displayScene() {

//清空窗口

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

GLdouble vex[8][3] = { { 25, 25, 25 },{ 25, -25, 25 },{ 25, 25, 75 },{ 25, -25, 75 },{ -25, 25, 25 },{ -25, 25, 75 },{ -25, -25, 25 },{ -25, -25, 75 } };

glPushMatrix();

{

glBegin(GL\_TRIANGLES);

glColor3f(0.1, 0.2, 0.3);

glVertex3dv(vex[0]);

glVertex3dv(vex[1]);

glVertex3dv(vex[2]);

glColor3f(0.3, 0.2, 0.4);

glVertex3dv(vex[3]);

glVertex3dv(vex[1]);

glVertex3dv(vex[2]);

glColor3f(0.8, 0.2, 0.4);

glVertex3dv(vex[3]);

glVertex3dv(vex[1]);

glVertex3dv(vex[7]);

glColor3f(0.2, 0.8, 0.7);

glVertex3dv(vex[7]);

glVertex3dv(vex[1]);

glVertex3dv(vex[6]);

glColor3f(0.8, 0.4, 0.3);

glVertex3dv(vex[7]);

glVertex3dv(vex[3]);

glVertex3dv(vex[2]);

glColor3f(0.3, 0.8, 0.6);

glVertex3dv(vex[7]);

glVertex3dv(vex[5]);

glVertex3dv(vex[2]);

glColor3f(0.7, 0.6, 0.8);

glVertex3dv(vex[5]);

glVertex3dv(vex[7]);

glVertex3dv(vex[6]);

glColor3f(0.5, 0.6, 0.7);

glVertex3dv(vex[4]);

glVertex3dv(vex[5]);

glVertex3dv(vex[6]);

glColor3f(0.1, 0.6, 0.3);

glVertex3dv(vex[5]);

glVertex3dv(vex[2]);

glVertex3dv(vex[0]);

glColor3f(0.1, 0.5, 0.6);

glVertex3dv(vex[4]);

glVertex3dv(vex[5]);

glVertex3dv(vex[0]);

glColor3f(0.1, 0.7, 0.2);

glVertex3dv(vex[6]);

glVertex3dv(vex[1]);

glVertex3dv(vex[0]);

glColor3f(0.7, 0.5, 0.1);

glVertex3dv(vex[4]);

glVertex3dv(vex[6]);

glVertex3dv(vex[0]);

glEnd();

}

glPopMatrix();

glFlush();

//交换通道

glutSwapBuffers();

}

//按下Esc退出

void pressToExit(unsigned char key, int x, int y) {

if (key == 27)

exit(0);

}

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

glutInit(&argc, argv);

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

glutInitWindowSize(500, 500);

glutInitWindowPosition(100, 100);

glutCreateWindow("simple OpenGL example");

myinit();

glutKeyboardFunc(pressToExit);

glutDisplayFunc(displayScene);

glutIdleFunc(displayScene);

glutMainLoop();

return 0;

}

**Task2**

功能说明：

1. 设置合适的相机位置和相机投影矩阵，使得OpenGL相机能够“看到”需要绘制的物体。

2. 用glutWireTeapot函数绘制茶壶。

3. 使用glRotate，glTranslate，glScale，对这个茶壶进行变换

4. 使用鼠标拖动茶壶可以实现移动，使用方向键可以控制旋转，使用 [ ]键（英文键盘）可以控制缩放

效果预览：

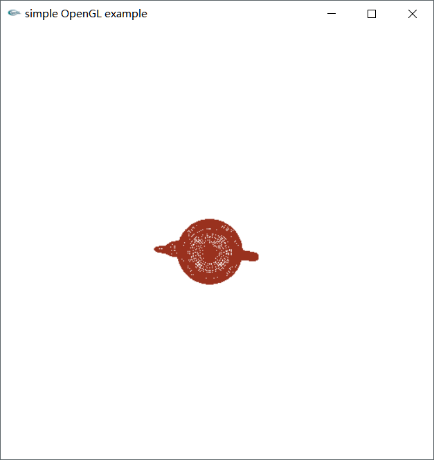
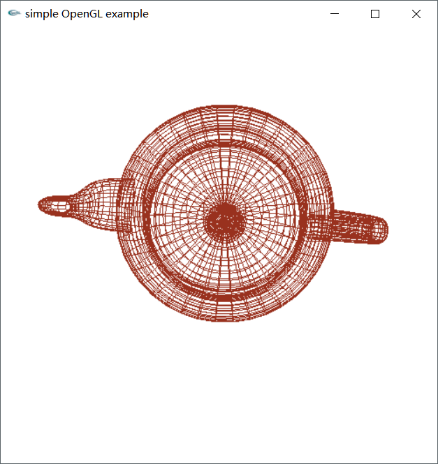
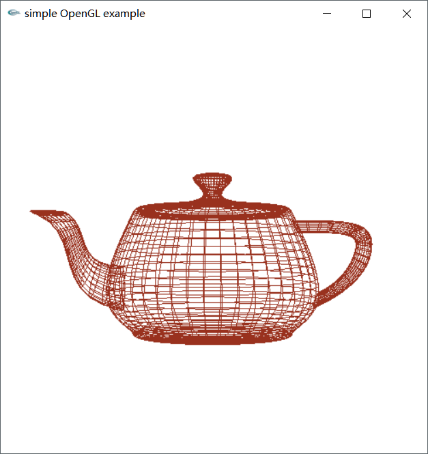


图2 图3 图4

详见视频

源代码：

/\*

\* 该程序实现以下功能

\* 1. 设置合适的相机位置和相机投影矩阵，使得OpenGL相机能够“看到”需要绘制的物体。

\* 2. 用glutWireTeapot函数绘制茶壶。

\* 3. 使用glRotate，glTranslate，glScale，对这个茶壶进行变换

\* 4. 使用鼠标拖动茶壶可以实现移动，使用方向键可以控制旋转，使用 [ ]键（英文键盘）可以控制缩放

\*\*\*\*\*\*\*

\* @author：宋灵冰

\*/

#include <gl/glut.h>

GLfloat angleX = 0.0;

GLfloat angleY = 0.0;

GLfloat scaleLevel = 1.0;

GLfloat moveX = 0.0;

GLfloat moveY = 0.0;

const GLfloat moveZ = 0.0;

GLfloat tempMoveX = 0.0;

GLfloat tempMoveY = 0.0;

int orgX, orgY;

bool move = false;

//设置初始属性

void myinit() {

//设置背景为白色

glClearColor(1.0, 1.0, 1.0, 1.0);

//启用深度缓存

glEnable(GL\_DEPTH\_TEST);

//设置观察参数

glMatrixMode(GL\_MODELVIEW);

//初始化坐标系

glLoadIdentity();

//设置当前相机的位置和方向

gluLookAt(

0.0, 0.0, 0.0,

0.0, 0.0, 1.0,

0.0, 1.0, 0.0);

glOrtho(-100.0, 100.0, -100.0, 100.0, -100.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

//用于初始显示图像

void displayScene() {

//清空窗口

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

glPushMatrix();

{

//设置茶壶颜色

glColor3d(153.0/255.0, 49.0/255.0, 30.0/255.0);

glutWireTeapot(20);

}

glPopMatrix();

glFlush();

//交换通道

glutSwapBuffers();

}

//设置空闲回调函数，用于响应状态变化

void changeView() {

//清空窗口

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

glPushMatrix();

{

glTranslatef(moveX / 3.0, moveY / 3.0, moveZ);

glRotatef(angleX, 1.0, 0.0, 0.0);

glRotatef(angleY, 0.0, 1.0, 0.0);

glScalef(scaleLevel, scaleLevel, scaleLevel);

glColor3d(153.0 / 255.0, 49.0 / 255.0, 30.0 / 255.0);

glutWireTeapot(50);

}

glPopMatrix();

glFlush();

//交换通道

glutSwapBuffers();

}

//实现旋转回调函数，使用方向键实现旋转

void rotateFunc(int key, int x, int y) {

switch (key)

{

case GLUT\_KEY\_UP:

angleX += 5.0;

break;

case GLUT\_KEY\_DOWN:

angleX -= 5.0;

break;

case GLUT\_KEY\_LEFT:

angleY += 5.0;

break;

case GLUT\_KEY\_RIGHT:

angleY -= 5.0;

break;

default:

break;

}

}

//实现缩放回调函数，使用[]控制缩放

void scaleFunc(unsigned char key, int x, int y) {

switch (key)

{

case '[':

if(scaleLevel > 0)

scaleLevel -= 0.1;

break;

case ']':

scaleLevel += 0.1;

break;

case 27:

exit(0);

break;

default:

break;

}

}

//实现移动回调函数，使用鼠标控制移动

void translateFunc(int button, int state, int x, int y) {

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN && !move) {

orgX = x;

orgY = y;

tempMoveX = moveX;

tempMoveY = moveY;

move = true;

}

if (state == GLUT\_UP && move) {

move = false;

}

}

void moveFunc(int x, int y) {

moveX = tempMoveX - x + orgX;

moveY = tempMoveY - y + orgY;

}

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

glutInit(&argc, argv);

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

glutInitWindowSize(500, 500);

glutInitWindowPosition(100, 100);

glutCreateWindow("simple OpenGL example");

myinit();

glutKeyboardFunc(scaleFunc);

glutSpecialFunc(rotateFunc);

glutMouseFunc(translateFunc);

glutMotionFunc(moveFunc);

glutDisplayFunc(displayScene);

glutIdleFunc(changeView);

glutMainLoop();

return 0;

}