**电 子 科 技 大 学**

**实 验 报 告**

**一、实验二：OpenGL建模与变换编程实验**

**二、实验室名称：主楼A2-412 实验时间：14：30~17：55**

**三、实验目的：**

1. 理解OpenGL视图变换、投影变换、模型-视图变换的相关原理；

2. 掌握与变换相关的OpenGL API函数及其用法。

**四、实验原理：OpenGl**

**五、实验内容：**

**（一）编程任务1——绘制钟表**

**（二）编程任务2——P96 作业2**

编写一个程序，实现交互式的移动摄像机。显示一个或多个对象（如茶壶等），并确保摄像机在移动时总是朝向场景中的同一点。

**六、实验器材（设备、元器件）：电脑**

**七、实验步骤：**

1. OpenGL建模的基本原理和思想：使用层次化建模，一个部件依赖于一个部件，这样在修改时可以减少工作量。
2. OpenGL中移动摄像机的基本原理和方式：使用gluLookAt函数改变摄像机的位置和朝向。

**八、实验数据及结果分析：**

编程任务1：

程序功能：

程序实现了根据本地时间绘制钟表。

程序结构：

表盘——void createDial();

刻度——void createScale1();void createScale2();

时针、分针、秒针——void createHourhand(); void createMinutehand(); void createSecondhand();

利用建模变换将钟表的基本构件组装在一起——void createWatch()；

控制钟表时针、分针和秒针的运动——void processWatch(int hour, int minute, int second);

依据当前时间绘制钟表——void displayWatch(void)；

源代码：

#include <glut.h>

#include <stdio.h>

#include <math.h>

#include <time.h>

#define huafen 100

#define clockR 80

void init();

void createDial();

void createScale1();

void createScale2();

void createWatch();

void createHourhand();

void createMinutehand();

void createSecondhand();

void displayWatch();

void processWatch(int hour, int minute, int second);

void gettime();

const double PI = 3.14159265;

struct tm\* localTime = new tm;

int hour, minute, second;

time\_t curTime;

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

glutInit(&argc, argv);

glutInitWindowSize(800, 800);

glutInitWindowPosition(0, 0);

glutCreateWindow("第二次实验");

glutDisplayFunc(displayWatch);

glutIdleFunc(displayWatch);

init();

glutMainLoop();

}

void createDial() {

glClear(GL\_COLOR\_BUFFER\_BIT);

//绘制黑色圆

glBegin(GL\_POLYGON);

glColor3f(0, 0, 0);

for (double i = 0; i <= 2 \* PI; i += 2 \* PI / huafen)

glVertex2d(clockR \* cos(i), clockR \* sin(i));

glEnd();

//绘制渐变色圆

glBegin(GL\_TRIANGLE\_FAN);

glVertex2d(0, 0);

glColor3f(0.7, 0.7, 0.7);

for (double i = 0; i <= 2 \* PI; i += 2 \* PI / huafen)

glVertex2d((clockR - 4) \* cos(i), (clockR - 4) \* sin(i));

glEnd();

//绘制横竖线

glBegin(GL\_LINES);

glColor3f(0, 0, 0);

glVertex2d(-clockR, 0);

glVertex2d(clockR, 0);

glVertex2d(0, clockR);

glVertex2d(0, -clockR);

glEnd();

}

void createScale1()

{

glRectd(-8, -2, 0, 2);

}

void createScale2()

{

glRectd(-15, -1, 0, 1);

}

void createWatch()

{

createDial();

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

//绘制大刻度

glColor3f(0, 0, 0);

for (int i = 0; i <= 360; i += 90) {

glPushMatrix();

glRotated(i, 0, 0, 1);

glTranslated(clockR, 0, 0);

createScale1();

glPopMatrix();

}

//绘制小刻度

for (int i = 30; i <= 330; i += 30) {

if (i % 90 == 0)

continue;

glPushMatrix();

glRotated(i, 0, 0, 1);

glTranslated(clockR, 0, 0);

createScale2();

glPopMatrix();

}

gettime();

processWatch(hour, minute, second);

//绘制最上面的圆

glColor3f(0.5, 0.5, 0.5);

glBegin(GL\_POLYGON);

for (double i = 0; i <= 2 \* PI; i += 2 \* PI / huafen)

glVertex2d(4 \* cos(i), 4 \* sin(i));

glEnd();

glColor3f(0, 0, 0);

glBegin(GL\_LINE\_LOOP);

for (double i = 0; i <= 2 \* PI; i += 2 \* PI / huafen)

glVertex2d(4 \* cos(i), 4 \* sin(i));

glEnd();

}

void createHourhand() {

glColor3f(1, 1, 1);

glBegin(GL\_POLYGON);

glVertex2f(-10, 0);

glVertex2f(0, 8);

glVertex2f(30, 0);

glVertex2f(0, -8);

glEnd();

glColor3f(0, 0, 0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(-10, 0);

glVertex2f(0, 8);

glVertex2f(30, 0);

glVertex2f(0, -8);

glEnd();

}

void createMinutehand() {

glColor3f(1, 1, 1);

glBegin(GL\_POLYGON);

glVertex2f(-10, 0);

glVertex2f(0, 5);

glVertex2f(50, 0);

glVertex2f(0, -5);

glEnd();

glColor3f(0, 0, 0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(-10, 0);

glVertex2f(0, 5);

glVertex2f(50, 0);

glVertex2f(0, -5);

glEnd();

}

void createSecondhand() {

glColor3f(1, 1, 1);

glBegin(GL\_POLYGON);

glVertex2f(-8, -1);

glVertex2f(60, -1);

glVertex2f(60, 1);

glVertex2f(-8, 1);

glEnd();

glColor3f(0, 0, 0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(-8, -1);

glVertex2f(60, -1);

glVertex2f(60, 1);

glVertex2f(-8, 1);

glEnd();

}

void processWatch(int hour, int minute, int second) {

//根据时间旋转指针

glPushMatrix();

if (hour >= 12)

hour -= 12;

glRotated((hour - 3) \* (-30), 0, 0, 1);

createHourhand();

glPopMatrix();

glPushMatrix();

glRotated((minute - 15) \* (-6), 0, 0, 1);

createMinutehand();

glPopMatrix();

glPushMatrix();

glRotated((second - 15) \* (-6), 0, 0, 1);

createSecondhand();

glPopMatrix();

}

void gettime()

{

time(&curTime);

localtime\_s(localTime, &curTime);

hour = localTime->tm\_hour;

minute = localTime->tm\_min;

second = localTime->tm\_sec;

}

void displayWatch()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

createWatch();

glFlush();

}

void init() {

glClearColor(1, 1, 1, 0);//设置屏幕颜色

glMatrixMode(GL\_PROJECTION);

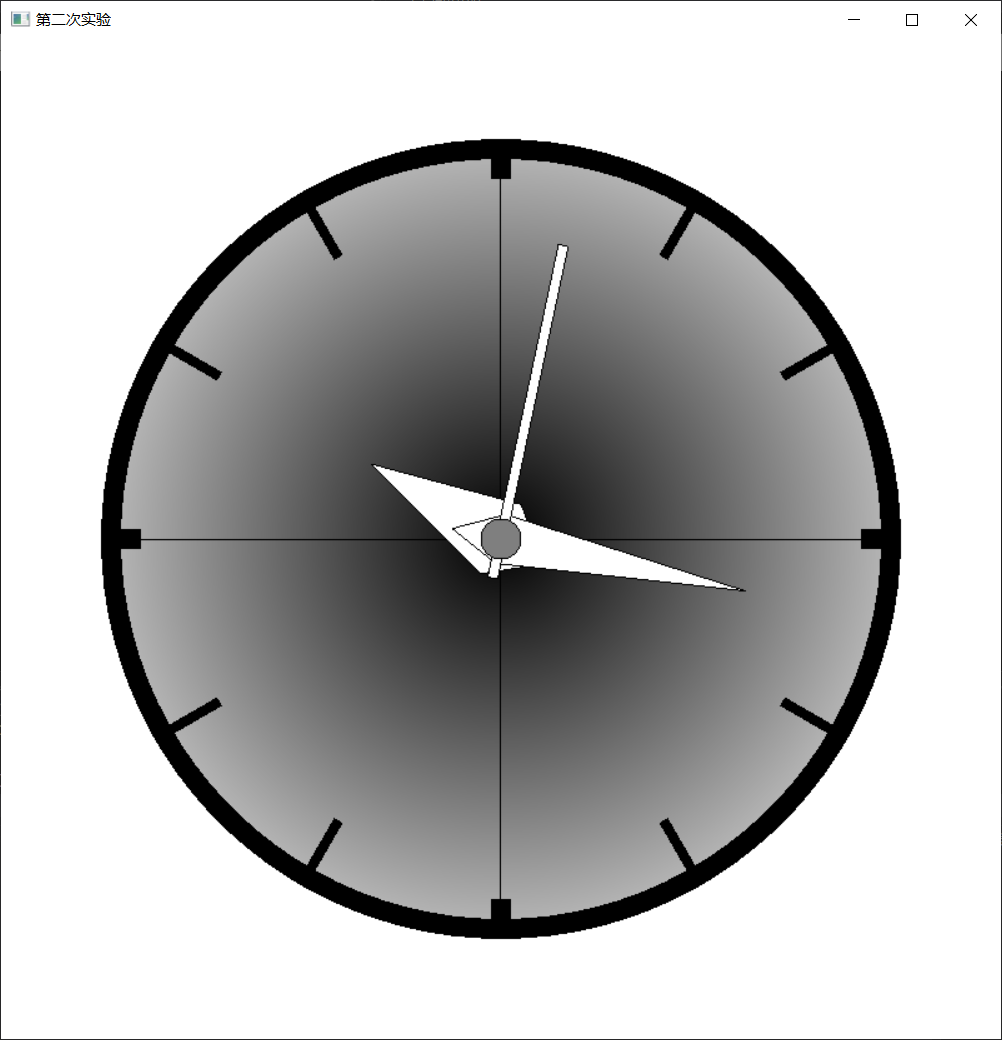
glLoadIdentity();

gluOrtho2D(-100, 100, -100, 100);

}

运行结果：

在22：17分时钟运行的结果，三个指针都能正确旋转。



编程任务2：

程序功能：

程序实现了显示一个茶壶，摄像机始终指向茶壶，用户通过键盘对摄像机的参数进行调整。

程序结构：

init函数实现了初始化，进行摄像机初始角度的设置。

display函数用来设置摄像机位置和显示茶壶。

注册了键盘的回调函数，可以对全局变量进行操作。

源代码：

#include <glut.h>

#include <stdio.h>

#include <math.h>

void init();

void display();

void mykey(unsigned char, int, int);

double x=0, y=0, z=10,n=5,f=20,fov=60;

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

glutInit(&argc, argv);

glutInitWindowSize(800, 800);

glutInitWindowPosition(0, 0);

glutCreateWindow("第二次实验");

glutDisplayFunc(display);

glutKeyboardFunc(mykey);

init();

glutMainLoop();

}

void display()

{

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

glColor3f(0, 0, 1);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(fov, 1, n, f);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

gluLookAt(x, y, z, 0, 0, 0, 0, 1, 0);

glutWireTeapot(1);

glFlush();

}

void init() {

glEnable(GL\_DEPTH\_TEST);

glClearColor(1, 1, 1, 0);//设置屏幕颜色

}

void mykey(unsigned char key, int a, int b) {

switch (key)

{

case '1': y++; break;

case '2': y--; break;

case '3': x++; break;

case '4': x--; break;

case '5': z++; break;

case '6': z--; break;

case '7': fov++; break;

case '8': fov--; break;

case '9': n++; break;

case '0': n--; break;

case '-': f++; break;

case '=': f--; break;

default:

break;

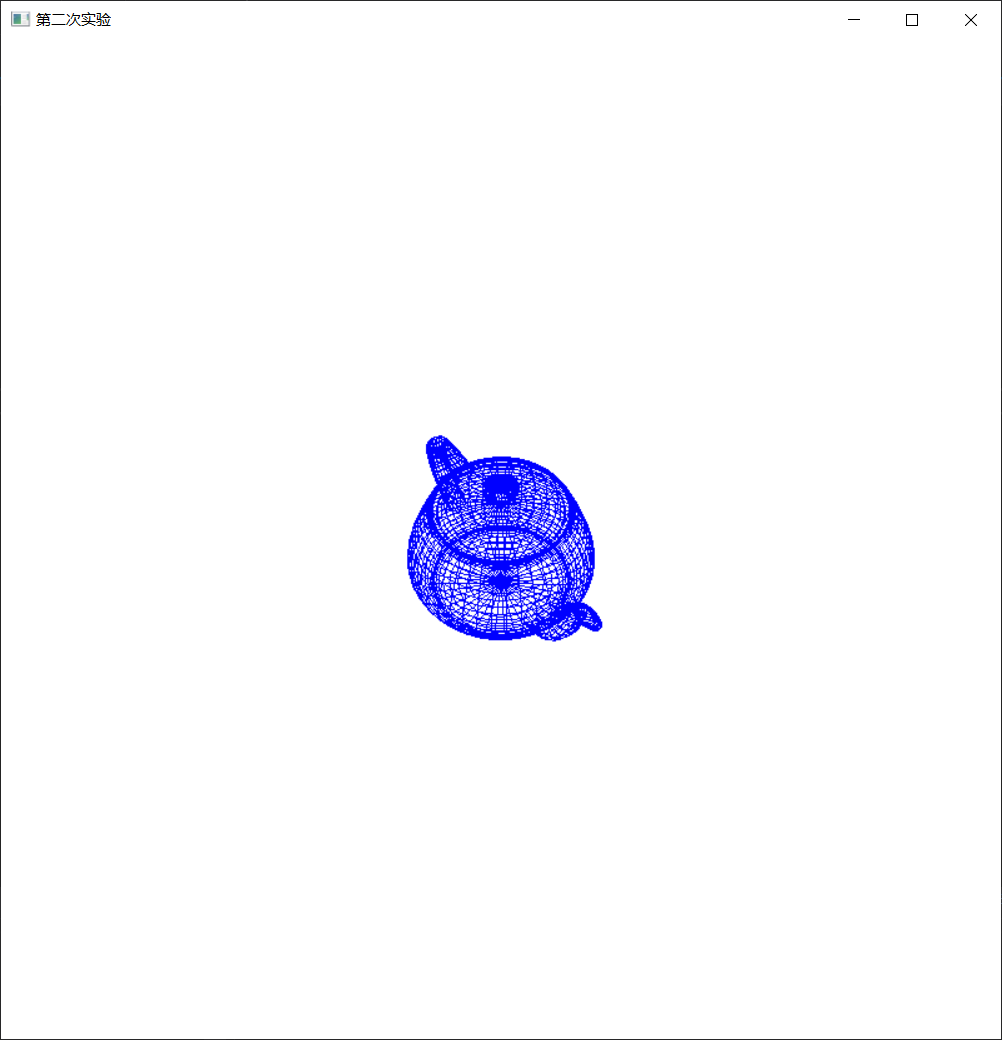
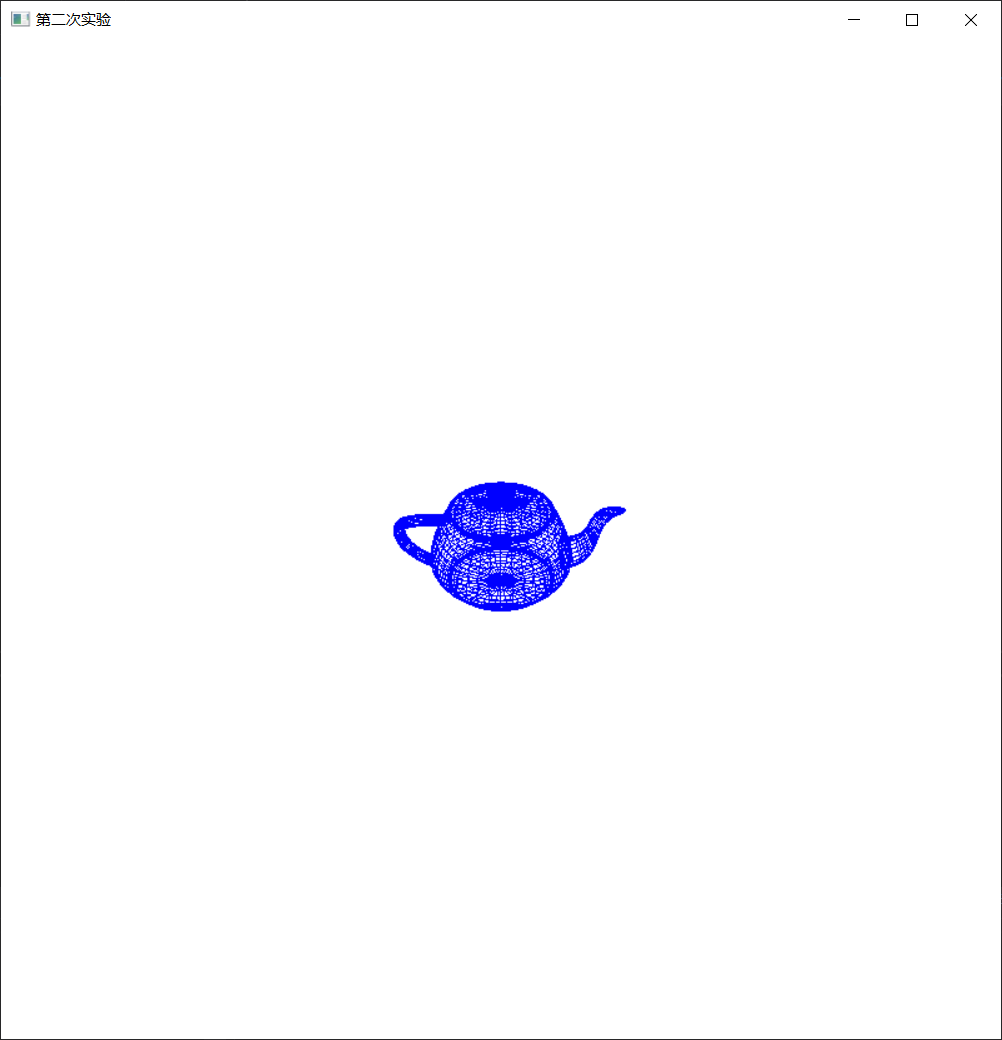
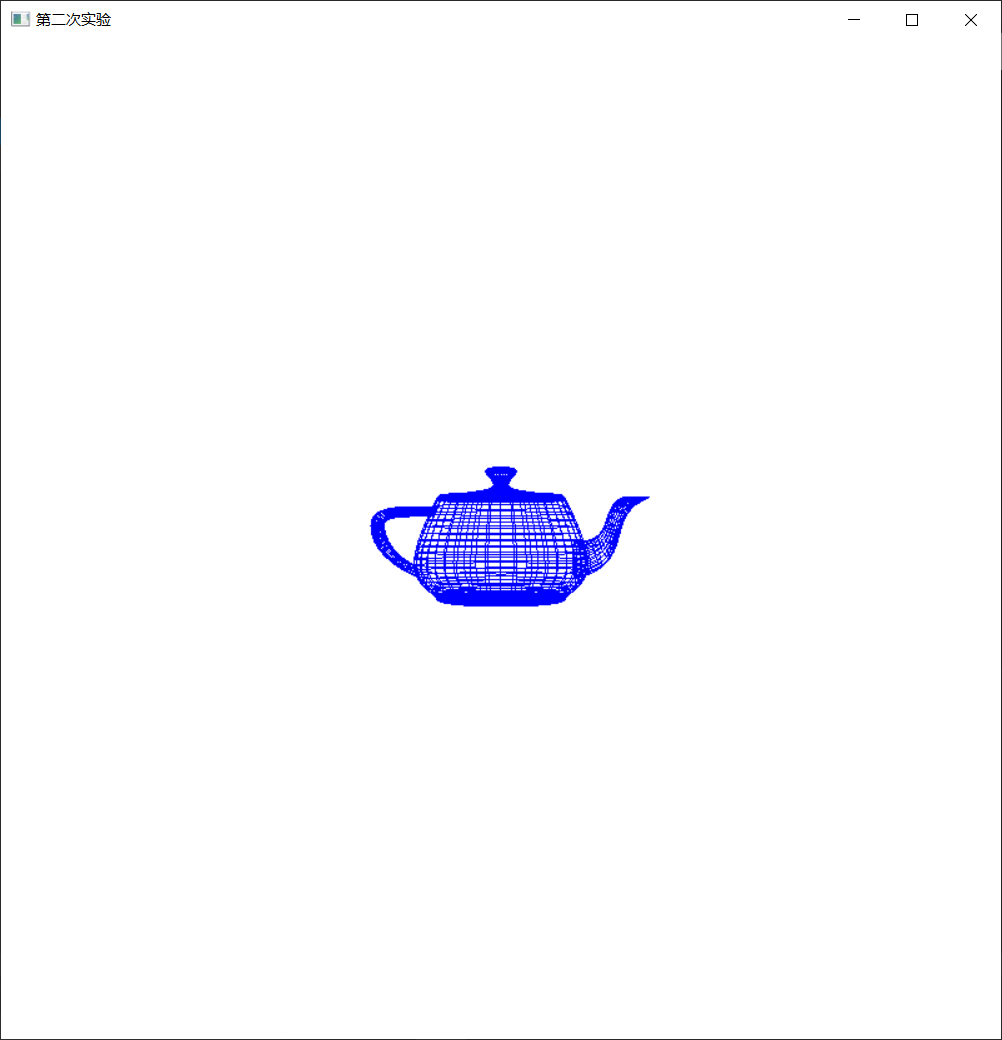
}

glutPostRedisplay();

}

运行结果：

初始界面为，通过键盘调整参数后为、



**总结、改进建议及心得体会：**对于实验一，绘制时钟时要注意各个部件的层次结构，在绘制单个物体时要使用glPushMatrix()和glPopMatrix()来保存之前的矩阵信息。在绘制圆形时用分割细致的矩形进行模拟，在制作渐变效果时要求中间点为黑色，其他为灰色，所以只能采用三角形来绘制。

对于实验二，在对摄像机进行操作时，需要注意MVP的顺序，同时在代码中矩阵操作是从下往上的，所以MVP的顺序也要颠倒。