北京邮电大学实验报告

计算机图形学实验报告

学院：计算机院

班级：2015211307

学号：2017526019

姓名：刘禾子

实验二

一、设计要求

试设计一个室内三维环境，并利用 Open GL 展示它的三维效果。

要求：

（1） 包含基本的实体元素：球，多面体，椎体，柱体，曲面等；

（2） 有全局光照和纹理功能

（3） 程序具有交互功能

二、交互说明

键盘方向键：

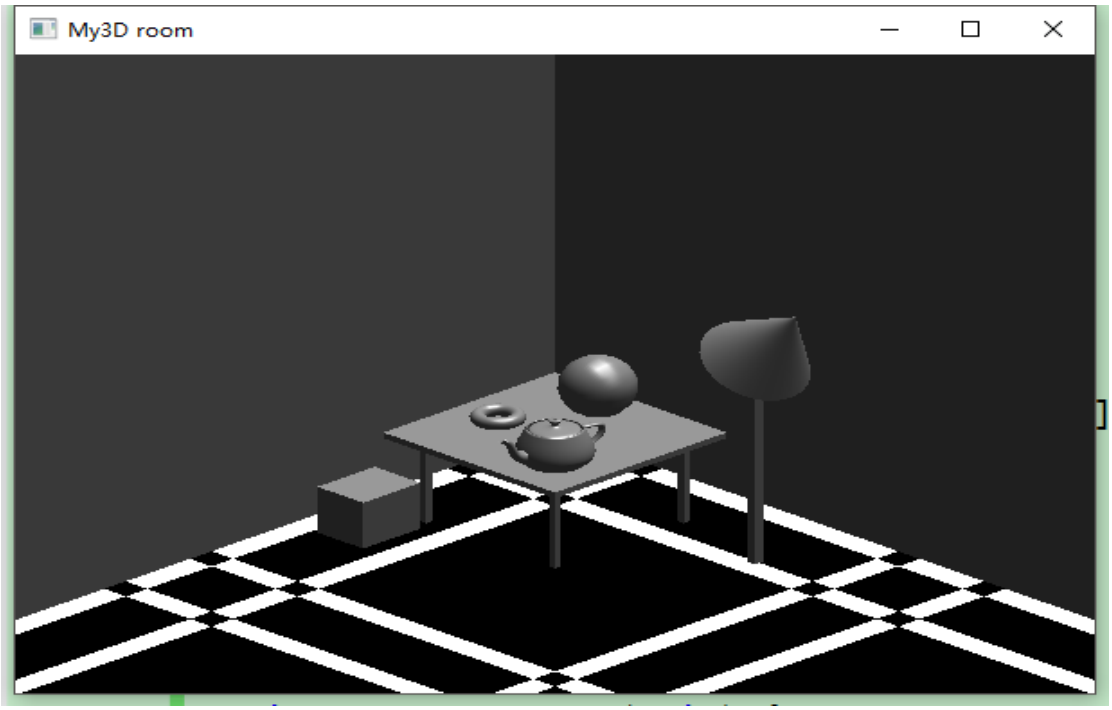
↑：视角上移

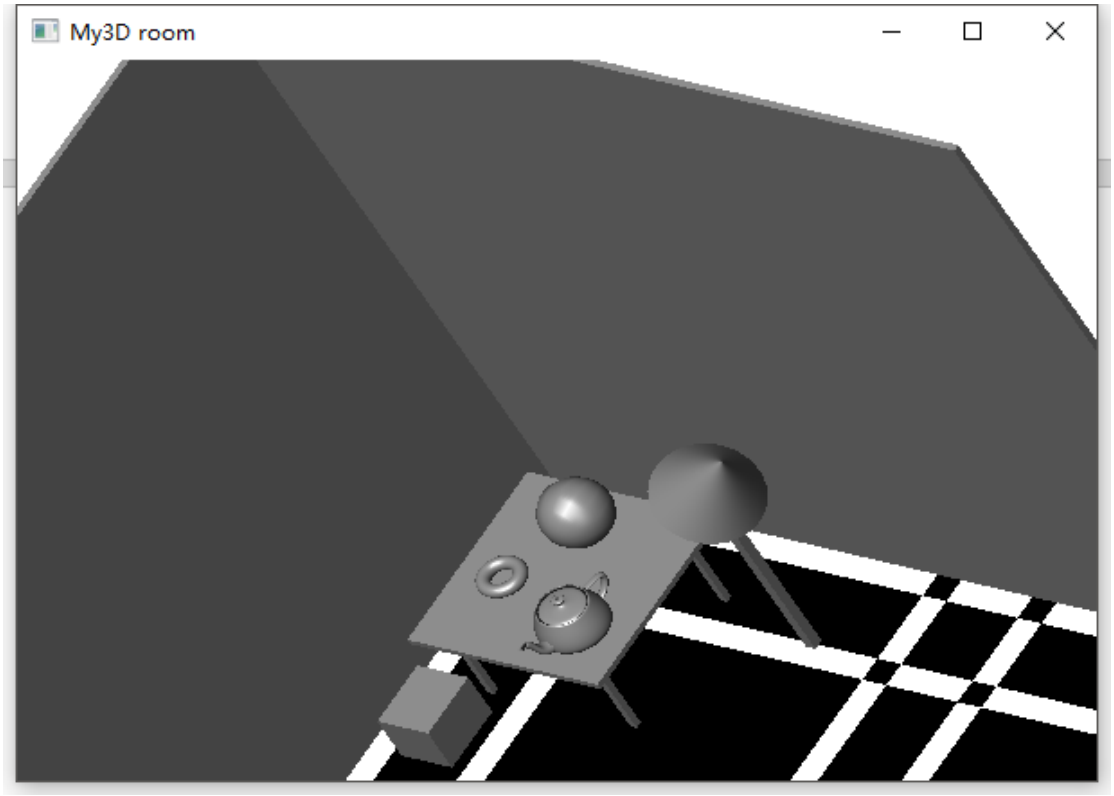
↓：视角下移

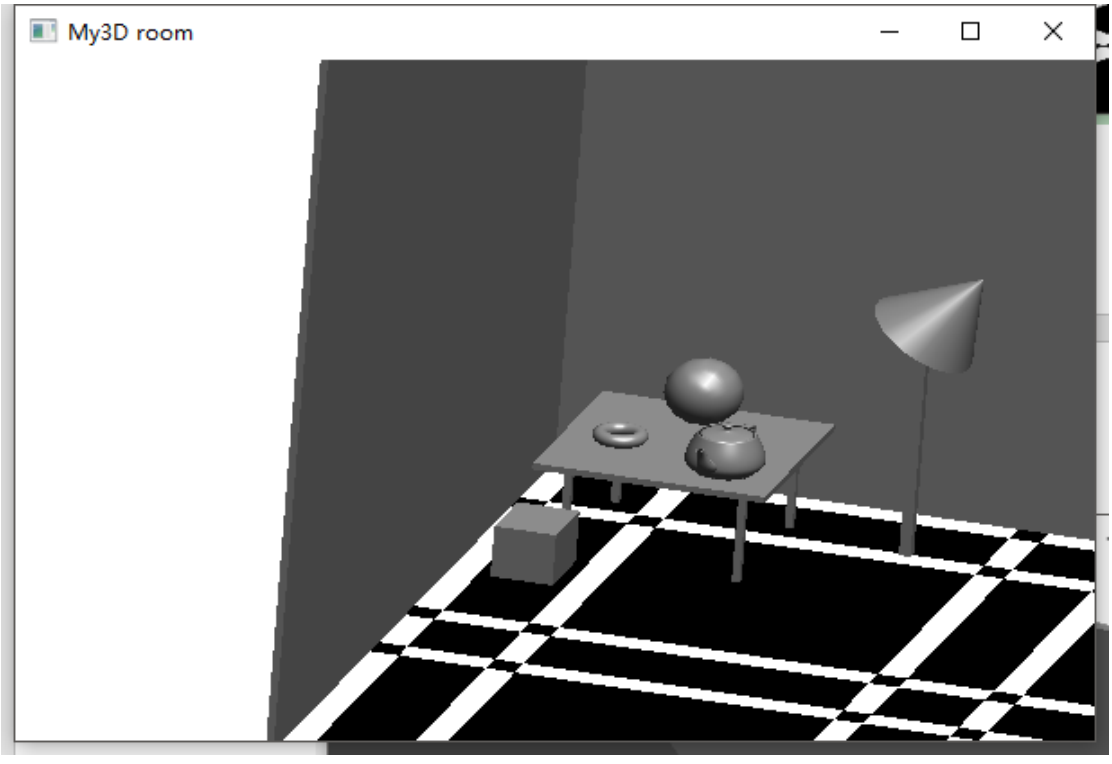
←：视角左移

→：视角右移

三、效果展示




My3D room


My3D room

四、程序代码

```
#include <stdlib.h>
#include <stdio.h>
#include <GL/glew.h>
#include <windows.h>
#include <gl/glut.h>
#pragma comment(lib,"glew32.lib")

#define checkImageWidth 64
#define checkImageHeight 64

static GLubyte checkImage[checkImageHeight][checkImageWidth][4];
static GLuint texName;

static GLfloat xrot = 0;
static GLfloat yrot = 0;

//绘制地面纹理图案
void makeCheckImage(void) {
    int i, j, c;
    for ( i = 0; i < checkImageHeight; i++)
    {
        for (j = 0; j < checkImageWidth; j++) {
            c = ((((i & 0x16) == 0) ^ ((j & 0x16)) == 0)) * 255;
            checkImage[i][j][0] = (GLubyte)c;
            checkImage[i][j][1] = (GLubyte)c;
            checkImage[i][j][2] = (GLubyte)c;
            checkImage[i][j][3] = (GLubyte)255;
        }
    }
}

//初始化
void SetupRC(void) {
    glEnable(GL_LIGHTING);          //启动光源
    glEnable(GL_LIGHT0);            //启动0号灯
    glShadeModel(GL_SMOOTH);        //设置光滑着色模式
    glEnable(GL_DEPTH_TEST);        //启动深度测试
    glEnable(GL_NORMALIZE);             //启动法向量
    glClearColor(1.0f, 1.0f, 1.0f, 0.0f);//为色彩缓冲区指定用于清除的值

    //设置表面材料的属性
    GLfloat mat_ambient[] = { 0.6f,0.6f,0.6f,1.0f };
```

```
    GLfloat mat_diffuse[] = { 0.5f,0.5f,0.5f,1.0f };
    GLfloat mat_specular[] = { 1.0f,1.0f,1.0f,1.0f };
    GLfloat mat_shininess[] = { 40.0f };
    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);//指定环境泛光的强度
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);//漫反射的强度
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);//镜面反射强度
    glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);//镜面反射光的汇聚强度

    //纹理
    makeCheckImage();                       //绘制地面纹理图案
    glPixelStorei(GL_UNPACK_ALIGNMENT, 1);//控制像素存储模式
    glGenTextures(1, &texName);             //用来生成纹理的数量为1
    glBindTexture(GL_TEXTURE_2D, texName);//绑定纹理

    //纹理滤波，图像从纹理图像空间映射到帧缓冲图像空间
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    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,GL_RGBA,checkImageWidth,checkImageHeight,
        0, GL_RGBA, GL_UNSIGNED_BYTE, checkImage);

}

//函数功能：绘制墙壁
void DrawWall(void)
{
    glPushMatrix();                 //矩阵堆栈压入
    glTranslated(1.2, 0.01, 1.2);   //将变换矩阵与当前矩阵相乘,使原点移动到（参数值坐标）
    glScaled(2.4, 0.02, 2.4);       //将比例矩阵与当前矩阵相乘,将当前图形沿x,y,z轴分别放大为原来的（参数值）倍
    glutSolidCube(1.0);             //size=1.0的实心立方体
    glPopMatrix();                  //矩阵堆栈弹出
}

//函数功能：绘制立方柱体
void DrawPillar(void)
{
    glPushMatrix();
    glTranslated(0, 0.15, 0);
    glScaled(0.02, 0.3, 0.02);
    glutSolidCube(1.0);
```

```
    glPopMatrix();
}

//函数功能：绘制桌子
void DrawTable(void)
{
    glPushMatrix();
    glTranslated(0.05, 0.3, 0.05);
    glScaled(0.6, 0.02, 0.6);
    glutSolidCube(1.0);    //绘制桌面
    glPopMatrix();

    glPushMatrix();
    glTranslated(0.275, 0, 0.275);
    DrawPillar();          //绘制桌腿
    glTranslated(0, 0, -0.45);
    DrawPillar();          //绘制桌腿
    glTranslated(-0.45, 0, 0.45);
    DrawPillar();          //绘制桌腿
    glTranslated(0, 0, -0.45);
    DrawPillar();          //绘制桌腿
    glPopMatrix();
}

//函数功能：绘图
void RenderScene(void)
{
    GLfloat light_position[] = { 2.0f, 6.0f, 3.0f, 0.0f };
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);  //指定0号光源的位置

    glMatrixMode(GL_PROJECTION);                        //对投影矩阵应用随后
的矩阵操作
    glLoadIdentity();                                   //将当前的用户坐标系
的原点移到了屏幕中心
    GLfloat h = 1.0;                                    //窗口的一半高度
    glOrtho(-h * 64 / 48.0, h * 64 / 48.0, -h, h, 0.1, 100.0); //将当前的可
视空间设置为正投影空间：左，右，下，上，近，远。

    glMatrixMode(GL_MODELVIEW);                         //对模型视景矩阵堆
栈应用随后的矩阵操作
    glLoadIdentity();
    gluLookAt(2, 1.6, 2, 0, 0.2, 0, 0, 1, 0);      //设置观察坐标系

                                                    //开始绘制
```

```
    glRotatef(xrot, 1.0f, 0.0f, 0.0f);              //旋转轴经过原点，方向为(1,0,0),旋转角度为xrot，方向满足右手定则
    glRotatef(yrot, 0.0f, 1.0f, 0.0f);                //旋转轴经过原点，方向为(0,1,0)，旋转角度为yrot
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glPushMatrix();
    glRotated(90.0, 0.0, 0.0, 1.0);
    DrawWall();             //绘制一面墙壁
    glPopMatrix();

    glPushMatrix();
    glRotated(-90.0, 1.0, 0.0, 0.0);
    DrawWall();             //绘制另一面墙壁
    glPopMatrix();

    glPushMatrix();
    glTranslated(0.4, 0, 0.4);
    DrawTable();            //绘制桌子
    glPopMatrix();

    glPushMatrix();
    glTranslated(0.2, 0.1, 0.85);
    glScaled(0.8, 0.8, 1.0);
    glutSolidCube(0.2);     //绘制箱子
    glPopMatrix();

    glPushMatrix();
    glTranslated(0.6, 0.38, 0.6);
    glRotated(-100, 0, 1, 0);
    glutSolidTeapot(0.1);  //绘制茶壶
    glPopMatrix();

    glPushMatrix();
    glTranslated(0.3, 0.33, 0.5);
    glRotated(180, 0, 180, 180);
    glutSolidTorus(0.02f, 0.05, 25, 50);//绘制手镯 【丝状花环】
    glPopMatrix();

    glPushMatrix();
    glTranslated(0.45, 0.42, 0.3);
    glutSolidSphere(0.1, 15, 50);      //绘制球体
    glPopMatrix();
```

```
    glPushMatrix();
    glTranslated(1.0, 0.35, 0.3);
    glScaled(0.03, 0.7, 0.03);
    glutSolidCube(1.0);                    //绘制灯柱
    glPopMatrix();

    glPushMatrix();
    glTranslated(1.0, 0.7, 0.3);
    glRotated(190, 180, 200, 270);
    glutSolidCone(0.15, 0.25f, 30, 25);    //绘制圆锥型灯罩
    glPopMatrix();

    //绘制纹理
    glEnable(GL_TEXTURE_2D);        //开启纹理
    glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);//映射方式

    glBegin(GL_QUADS);              //绘制地面
                                    //绘制四边形并给出当前顶点所对应的纹理坐
标
    glTexCoord2f(0.0, 0.0); glVertex3f(0.0, 0.0, 0.0);
    glTexCoord2f(0.0, 1.0); glVertex3f(0.0, 0.0, 2.4);
    glTexCoord2f(1.0, 1.0); glVertex3f(2.4, 0.0, 2.4);
    glTexCoord2f(1.0, 0.0); glVertex3f(2.4, 0.0, 0.0);
    glEnd();

    glDisable(GL_TEXTURE_2D);       //关闭纹理
    glFlush();                    //绘图结果显示到屏幕上
}

//函数功能：处理按键交互信息
void SpecialKeys(int key, int x, int y)
{
    switch (key) {
    case GLUT_KEY_UP:       //上键
        xrot += 1.0;
        break;
    case GLUT_KEY_DOWN:     //下键
        xrot -= 1.0;
        break;
    case GLUT_KEY_LEFT:     //左键
        yrot += 1.0;
        break;
    case GLUT_KEY_RIGHT:    //右键
        yrot -= 1.0;
```

```
        break;
    default:
        break;
    }
    glutPostRedisplay();        //标记当前窗口需要重新绘制
    glFlush();                  //绘图结果显示到屏幕上
}

//函数功能：改变窗口大小
void ChangeSize(int w, int h)
{
    GLfloat lightPos[] = { -50.f, 50.0f, 100.0f, 1.0f };
    GLfloat nRange = 1.9f;

    if (h == 0)
        h = 1;

    glViewport(0, 0, w, h);         //重新设置屏幕上的窗口大小

    glMatrixMode(GL_PROJECTION);//后继操作都在投影变换范围内
    glLoadIdentity();           //设置当前矩阵为单位矩阵

                                //正交投影
    if (w <= h)
        glOrtho(-nRange, nRange, -nRange*h / w, nRange*h / w, -nRange,
nRange);
    else
        glOrtho(-nRange*w / h, nRange*w / h, -nRange, nRange, -nRange,
nRange);

    glMatrixMode(GL_MODELVIEW);             //选择模型观察矩阵
    glLoadIdentity();                   //设置当前矩阵为单位矩阵

    glLightfv(GL_LIGHT0, GL_POSITION, lightPos);//重新定义光源
}

void main()
{
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);//双缓存
    glutInitWindowSize(600, 400);       //窗口大小
    glutInitWindowPosition(300, 100); //窗口位置
    glutCreateWindow("My3D room");      //创建窗口，名为xdy room

    SetupRC();                          //自定义的初始化函数
```

```
    glutDisplayFunc(RenderScene);       //显示回调函数，用于绘图

    glutReshapeFunc(ChangeSize);        //处理改变窗口大小
    glutSpecialFunc(SpecialKeys);       //处理按键交互信息
    glutMainLoop();                     //让GLUT框架开始运行，处理交互事件
}
```