

**课程名称** 计算机图形学

**学院** 计算机学院

**专业** 计算机科学与技术

**班级**  2014211306

**学号** 2014211292

**姓名** 张有杰

1. 实验目的

试设计一个室内三维环境, 并利用openGL展示它的三维效果。要求：

（1）包含基本的实体元素：球、多面体、锥体、 柱体等；

（2）有全局光照效果和纹理功能；

（3）程序具有交互功能；

熟悉 Windows openGL API 的使用。

1. 实验环境

1. 操作系统：Windows 10

2. 开发环境：Visual Studio 2017

1. 实验内容

1.纹理功能和全局光照功能

2.交互功能

3.球体、锥体、多面体、柱体的绘制。

1. 实验步骤

1. 完成锥体、多面体、柱体、球体的绘制。

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();

glTranslated(1.0, 0, 1.0);

DrawTable(); //画桌子

glPopMatrix();

glPushMatrix();

glTranslated(0.4, 0.35, 1.7);

glScaled(0.03, 0.7, 0.03);

glutSolidCube(2.0); //绘制柱体

glPopMatrix();

glPushMatrix();

glTranslated(1.2, 0.4, 1.3);

glRotated(180, 0, 200, 180);

glutSolidCone(0.15, 0.25f, 30, 25); //绘制圆锥

glPopMatrix();

glPushMatrix();

glTranslated(0.2, 0.1, 0.85);

glScaled(0.8, 0.8, 1.0);

glutSolidCube(0.2); //绘制多面体

glPopMatrix();

//glPushMatrix();

//glTranslated(0.5, 0.8, 0.85);

//glScaled(0.8, 0.8, 1.0);

//glutWireCube(1.0); //绘制多面体框架

//glPopMatrix();

glPushMatrix();

glTranslated(0.9, 0.4, 0.9);

glutSolidSphere(0.1, 15, 50); //绘制球体

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

}

2. 使用方向键可以调整角度进行观察。

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

}

3. 通过缩放、扩大窗口大小可以改变观察点的远近。

//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

}

4. 纹理功能

void makeCheckImage(void)

{

int i, j, c;

float ti, tj;

for (i = 0; i < checkImageHeight; i++) {

ti = 2.0\*3.14159265\*i / checkImageHeight;

for (j = 0; j < checkImageWidth; j++) {

tj = 2.0\*3.14159265\*j / checkImageWidth;

c = ((((i & 0x16) == 0) ^ ((j & 0x16)) == 0)) \* 255;

checkImage[i][j][0] = (GLubyte)127 \* (1.0 + sin(i));

checkImage[i][j][1] = (GLubyte)127 \* (1.0 + cos(ti));

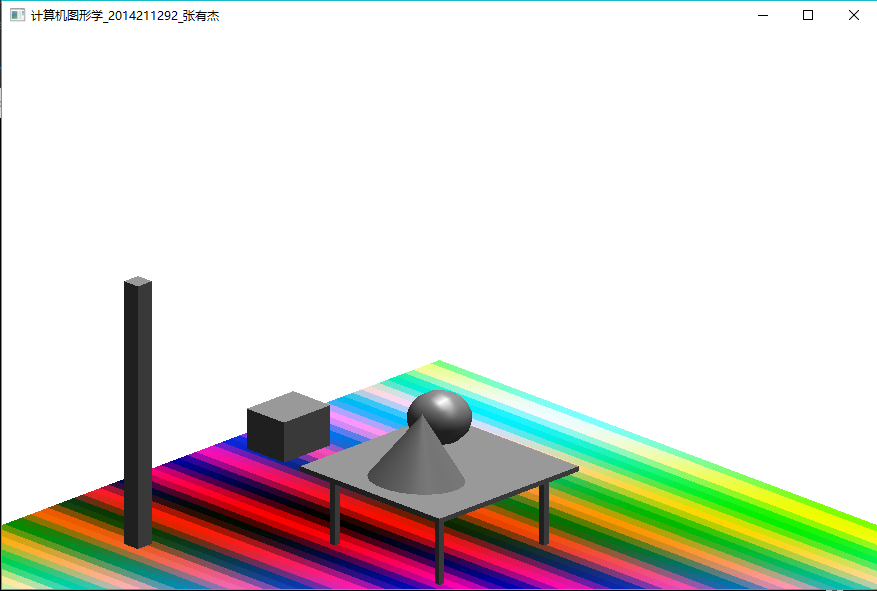
checkImage[i][j][2] = (GLubyte)127 \* (1.0 + sin(ti+tj));

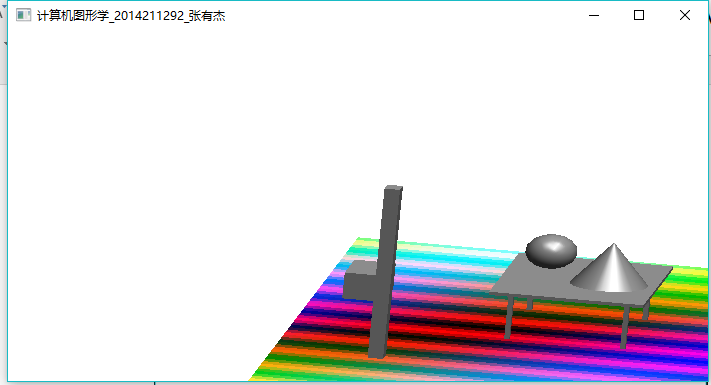
checkImage[i][j][3] = (GLubyte)255;

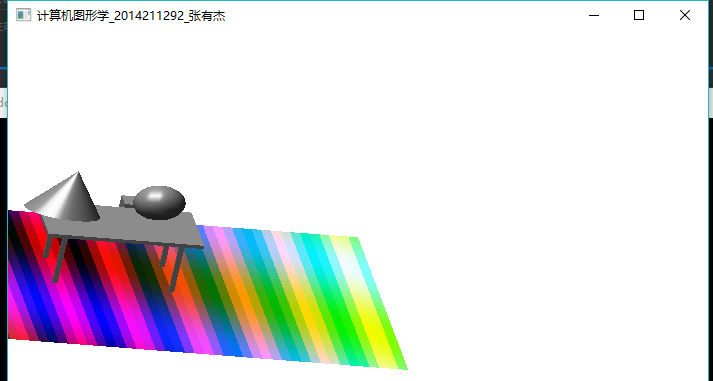
}

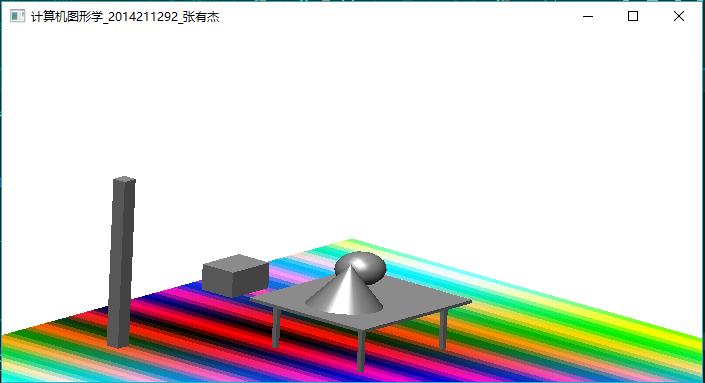
}

}

实验结果：







五、实验总结

在此次实验中，我通过使用 OpenGL 中的实现了 3D 场景下的柱体、球体、锥体等形状的绘制，完成了光照功能的设置，同时实现了纹理功能。

通过此次实验，我熟悉了 OpenGL 的基本使用，并实现了一个基本的室内环境。这次实验中遇到了许多问题，好在网上有许多 openGL 相关的教程可以参考，所以最好都顺利解决，这锻炼了我的动手能力和解决问题的能力，也提高了我的编程能力。