浙江大学实验报告

课程名称：____计算机图形学____ 指导老师：_____成绩：_____
实验名称：____OpenGL 矩阵_____ 实验类型：_____基础实验_____同组学生姓名：_____

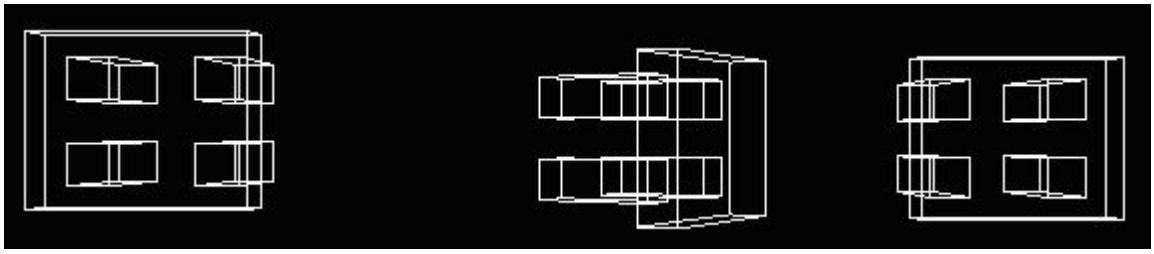
装 订 线

一、实验目的和要求

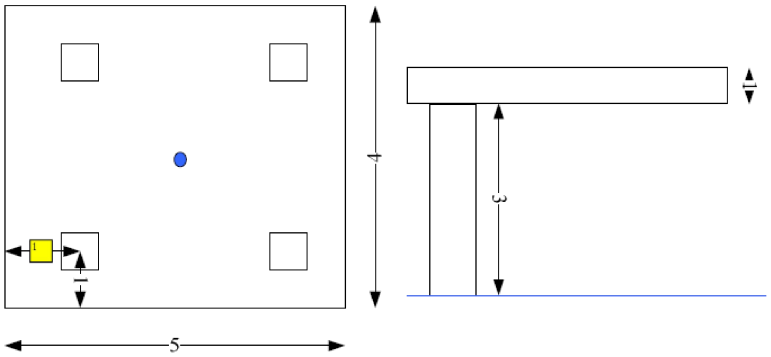
在 OpenGL 编程基础上，通过实现实验内容，掌握 OpenGL 的矩阵使用，并验证课程中矩阵变换的内容。

二、实验内容和原理

使用 Visual Studio C++编译已有项目工程，并修改代码生成以下图形：



模型尺寸如下：

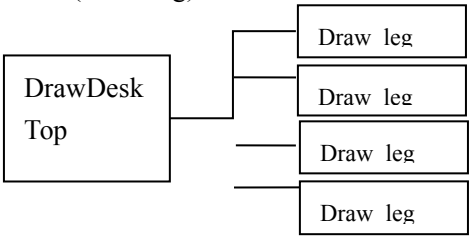


三、主要仪器设备

Visual Studio C++
glut-3.7.6-bin.zip
glutEx1-vs2005 工程

四、操作方法和实验步骤

本次实验的关键在于使用层次建模法绘制桌子，绘制一个桌子的过程可以分解为画桌面(DrawDesktop)和画桌腿(DrawLeg)，因此可以先进行伸缩，平移等几何变换，再使用 glutWireSolidCube 绘制长方体组成桌子。



1）绘制桌面的函数

注意到桌面的尺寸要求，需对 y 轴及 z 轴进行相应的伸缩变换，变换矩阵为：

$$\begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.2 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.8 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

```
// 绘制桌面
void DrawDesktop()
{
    glScalef(1.0f, 0.2f, 0.8f);    //进行伸缩变换
    glutWireCube(5.0f);
}
```

2）绘制桌腿的函数

类似桌面，对每条桌腿也要进行伸缩变换至合适的尺寸。但注意到我们是先画桌面再画桌腿，而画桌面时已经经过一次伸缩变换，所以应先将现有的坐标还原为未画桌面前的坐标，相应的变换矩阵为

$$\begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0/0.2 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0/0.8 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

之后再将桌腿变换为需要的尺寸，相应的变换矩阵为

$$\begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 3.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

```
// 绘制桌腿
void DrawLeg()
{
    glScalef(1.0f, 3.0f, 1.0f);
    glScalef(1.0f, 5.0f, 1.0f/0.8f);

    glutWireCube(1.0f);
}
```

3）使用层次建模法绘制桌子

注意到每条桌腿相对桌面的位置是对称的，因此我们每一条桌腿都可以相对桌面的坐标系进行绘制，即使

用层次建模法，使用 OpenGL 的矩阵堆栈可以实现保存以桌面为参考的坐标系（变换矩阵）
以一条桌腿为例，按照给出的桌子尺寸，桌腿的中心应相对桌子的中心（原点）平移向量 v1 = (-1.5, -2.0, 1.0) （模型中给出的尺寸有问题，我进行了稍微修改），相应的变换矩阵为：（因为平移之前进行了尺寸的缩放，所以平移矩阵的分量要乘以相应的系数）

$$\begin{bmatrix} -1.5 & 0.0 & 0.0 & 0.0 \\ 0.0 & -2.0*5.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0*1.0/0.8 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

```
// 使用层次建模法绘制桌子
void Draw_Desk()
{
    DrawDesktop();          // 画桌面
    glPushMatrix();         // 保存当前变换矩阵

    glTranslatef(-1.5, -2.0*5.0, 1.0*1.0/0.8);  // 相对桌面平移
    DrawLeg();              // 画桌腿 1
    glPopMatrix();          // 使用桌面坐标系
    glPushMatrix();         // 保存桌面坐标系

    glTranslatef(1.5, -2.0*5.0, 1.0*1.0/0.8);
    DrawLeg();              // 画桌腿 2
    glPopMatrix();
    glPushMatrix();

    glTranslatef(1.5, -2.0*5.0, -1.0*1.0/0.8);
    DrawLeg();              // 画桌腿 3
    glPopMatrix();
    glPushMatrix();

    glTranslatef(-1.5, -2.0*5.0, -1.0*1.0/0.8);
    DrawLeg();              // 画桌腿 4
    glPopMatrix();
    glPushMatrix();

    glPopMatrix();      // Don't forget this!
}
```

最后将 glutEx1-vs2005 工程中的 Draw_Triangle()函数替换为 Draw_Desk()函数并作一定的参数调整即可实现最终的显示效果。

五、实验数据记录和处理

实验中所有变换矩阵为：

1)修改桌面尺寸：

$$\begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.2 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.8 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

2)还原桌面尺寸：

$$\begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0/0.2 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0/0.8 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

3）修改桌腿尺寸：

$$\begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 3.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

4）平移桌腿：

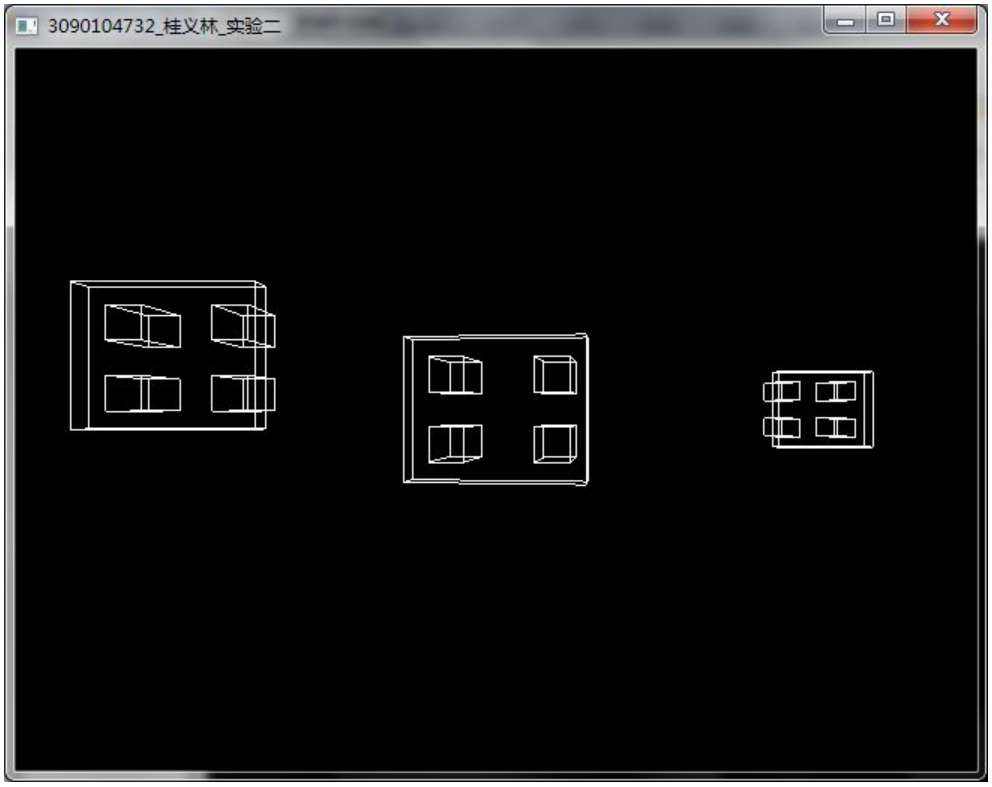
$$\begin{bmatrix} -1.5 & 0.0 & 0.0 & 0.0 \\ 0.0 & -2.0*5.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0*1.0/0.8 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

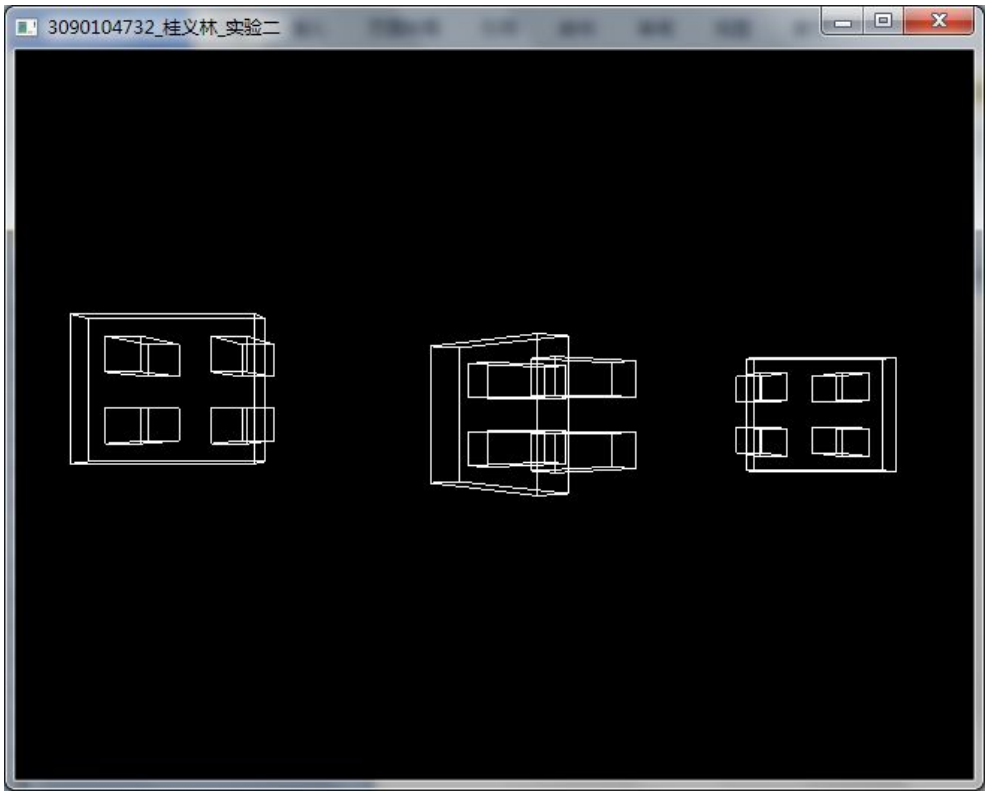
$$\begin{bmatrix} 1.5 & 0.0 & 0.0 & 0.0 \\ 0.0 & -2.0*5.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0*1.0/0.8 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

$$\begin{bmatrix} 1.5 & 0.0 & 0.0 & 0.0 \\ 0.0 & -2.0*5.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -1.0*1.0/0.8 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

$$\begin{bmatrix} -1.5 & 0.0 & 0.0 & 0.0 \\ 0.0 & 2.0*5.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -1.0*1.0/0.8 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

六、实验结果与分析





七、讨论、心得

(1) 通过本次实验我了解了 OpenGL 中的几何变换的原理和方法，明白了矩阵堆栈的概念，学会了通过矩阵堆栈进行参考坐标系的设置，成功使用层次建模法绘制出桌子。

(2) 实验中我对 OpenGL 中的矩阵做了一些研究，我发现如下两点

1) OpenGL 中的矩阵是按照**列优先存储**的，即 glLoadMatrix(m);/*m 是一个长度为 16 的数组*/ 参数中的 m 写成矩阵应为如下形式：

$$\begin{bmatrix} m[0] & m[4] & m[8] & m[12] \\ m[1] & m[5] & m[9] & m[13] \\ m[2] & m[6] & m[10] & m[14] \\ m[3] & m[7] & m[11] & m[15] \end{bmatrix}$$

2) OpenGL 中的矩阵是**右乘**的，但却使用了**列向量**的方式保存坐标向量，其实现方式是先将所有矩阵乘好，最后将得到的矩阵左乘列向量，这也解释了为什么 OpenGL 中**后指定的变换先执行**，如：

```
glLoadMatrix(m1);   //此时矩阵堆栈顶部为 m1
glMultiMatrix(m2);   //此时矩阵堆栈顶部为 m1*m2
/*  绘图代码  */        //假设原坐标为(x,y,z)^T，则变换后的新坐标为 (x1,y1,z1)^T = (m1*m2) * (x,y,z)^T
```