# 浙江大学实验报告

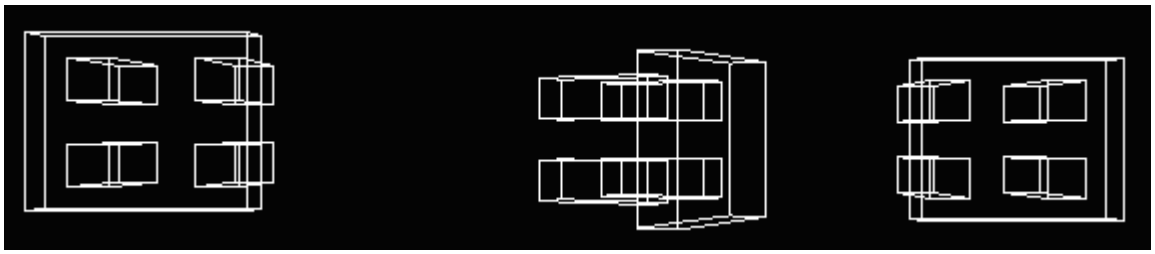
课程名称：____计算机图形学___ 指导老师：_____成绩：_____
实验名称：____OpenGL 矩阵_____ 实验类型：____基础实验____同组学生姓名：_____

## 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。

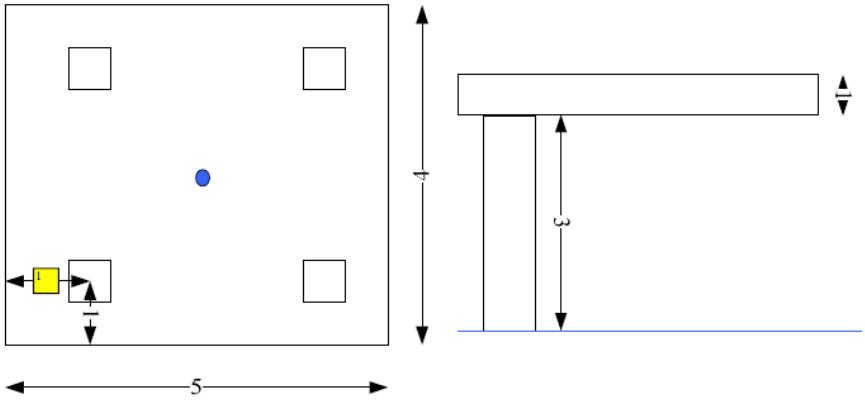
## 二、实验内容

使用 Visual Studio C++编译已有项目工程，并修改代码生成以下图形（参考示例答案）：



其中最左边的桌子循环上移（即匀速上移到一定位置后回到原点继续匀速上移），中间的桌子不断旋转，最右边的桌子循环缩小（即不断缩小到一定大小后回归原来大小继续缩小）。

桌子的模型尺寸如下：




## 三、主要仪器设备

Visual Studio C++
Ex2 工程

## 四、实验原理及实验步骤

一共分为两个步骤：绘制桌子和对桌子进行平移、旋转、缩放操作。

**1. 绘制桌子：**

（1） 绘制长方体：

每个桌子由五个长方体构成，分别为桌面和四条腿，每个长方体绘制的方式是相同的：首先给出长方体八个顶点的坐标（给出两个 x 轴坐标，两个 y 轴纵坐标，两个 z 轴坐标即可获得八个坐标），得到坐标后绘制出六个面，即可绘制出一个长方体。

实现代码如下：

```
void Draw_Cuboid(GLfloat x1, GLfloat x2, GLfloat y1, GLfloat y2, GLfloat z1, GLfloat z2)
 //Draw the cuboid
{
    int i, j;
    GLfloat vertex[8][3] = {//Eight vertices of the cuboid
        {x1,y1,z1},
        {x1,y2,z1},
        {x2,y2,z1},
        {x2,y1,z1},
        {x1,y1,z2},
        {x1,y2,z2},
        {x2,y2,z2},
        {x2,y1,z2}
    };

    GLint flat[6][4] = {//Six planes of the cuboid
        {0,1,2,3},
        {1,2,6,5},
        {0,4,7,3},
        {2,3,7,6},
        {0,1,5,4},
        {4,5,6,7}
    };

    glBegin(GL_QUADS);  //Draw the cuboid
    for (i = 0; i < 6; i++) {
        for (j = 0; j < 4; j++) {
            glVertex3fv(vertex[flat[i][j]]);
        }
    }
    glEnd();
}
```

（2） 绘制桌子：

分别给出绘制出五个长方体，拼凑起来就可以得到桌子。

实现代码如下：

```
1. void Draw_Table()  //Draw the table
2. {
3.     Draw_Cuboid(0.0, 1.0, 0.0, 0.8, 0.6, 0.8);  //draw the desktop
4.     //Draw four table legs
5.     Draw_Cuboid(0.1, 0.3, 0.1, 0.3, 0.0, 0.6);
6.     Draw_Cuboid(0.7, 0.9, 0.1, 0.3, 0.0, 0.6);
7.     Draw_Cuboid(0.1, 0.3, 0.5, 0.7, 0.0, 0.6);
8.     Draw_Cuboid(0.7, 0.9, 0.5, 0.7, 0.0, 0.6);
9. }
```

**2. 使桌子平移、旋转和缩放：**

（1） 平移：

此处运用的函数为glTranslatef()函数：void glTranslatef(GLfloat x, GLfloat y, GLfloat z);函数功能：沿 X 轴正方向平移 x 个单位(x 是有符号数)，沿 Y 轴正方向平移 y 个单位(y 是有符号数)，沿 Z 轴正方向平移 z 个单位(z 是有符号数)。

运用矩阵可以这样解释：

Translation

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

经过矩阵运算之后 $x=x+T_x$, $y=y+T_y$, $z=z+T_z$,这里的 $T_x$, $T_y$, $T_z$,即为函数中的 x, y, z，此处我运用 glTranslatef(0.0f, fTranslate, 0.0f)，使桌子在 Y 轴平移。

（2） 旋转：

此处运用的函数为 glRotatef()函数：void glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z);此处的旋转方向为：做出从(0,0,0)到(x,y,z)的向量，用右手握住这条向量，大拇指指向向量的正方向，四指环绕的方向就是旋转的方向；函数功能：以点(0,0,0)到点(x,y,z)为轴，旋转 angle 角度。

运用矩阵可以这样解释：

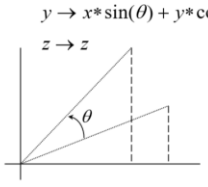
Rotation (around Z axis)

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$x \rightarrow x*\cos(\theta) - y*\sin(\theta)$

$y \rightarrow x*\sin(\theta) + y*\cos(\theta)$

$z \rightarrow z$




经过矩阵运算过后 x=x*cos(θ)-y*sin(θ)， y=x*sin(θ)+y*cos(θ),z=z，即绕 Z 轴旋转，在这里我运用 glRotatef(fRotate , 0 , 1.0f , 0)，即绕着 Y 轴旋转一定的角度。

（3） 缩放：

此处运用的函数为 glScalef()函数：void glScalef(GLfloat x, GLfloat y, GLfloat z);参数 x,y,z 分别为模型在 X，Y，Z 轴方向的缩放比。函数功能：沿 X 轴正方向拉伸 x 倍，沿 Y

轴正方向拉伸 y 倍，沿 Z 轴正方向拉伸 z 倍。

运用矩阵可以这样解释：

Scaling

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

经过矩阵运算过后 $x=x*S_x$， $y=y*S_y$, $z=z*S_z$，在这里我运用 glScalef(fScale, fScale, fScale)，即将桌子在 X, Y, Z 轴方向按相同比例缩小。

（4） 最后对 fTranslate, fRotate, fScale 三个因子进行更新，平移运动中当移动到一定位置时返回原位置，缩放运动中当缩小到一定程度时返回原大小。

（5） 实现代码如下：

```
1.  void redraw()
2.  {
3.      //display in wireframe mode
4.       glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
5.
6.      glClear(GL_COLOR_BUFFER_BIT);  //Clear window
7.      glLoadIdentity();  //Reset the current modelview matrix
8.
9.      //Translation
10.     glPushMatrix();
11.         glTranslatef(-2.7f, 0.0f, -6.0f);  //Place the table left
12.         glTranslatef(0.0f, fTranslate, 0.0f);  //Translate in Y direction
13.         Draw_Table();  //Draw table
14.     glPopMatrix();
15.
16.     //Rotation
17.     glPushMatrix();
18.         glTranslatef(0.0f, 0.4f, -6.0f);  //Place the table at center
19.         glRotatef(fRotate , 0 , 1.0f , 0);  //Rotate around Y direction
20.         glTranslatef(-0.5f, -0.4f, 0.0f);  //Rotate the table around the center
21.         Draw_Table();  //Draw table
22.     glPopMatrix();
23.
24.     //Scaling
25.     glPushMatrix();
26.         glTranslatef(1.7f, 0.0f, -6.0f);  //Place the table right
27.         glScalef(fScale, fScale, fScale);  //Scale
28.         Draw_Table();  //Draw table
29.     glPopMatrix();
```

```
30.
31.     //Update the factors
32.     fTranslate += 0.005f;
33.     fRotate += 0.5f;
34.     fScale -= 0.005f;
35.
36.     if (fTranslate > 0.5f) fTranslate = 0.0f;  //Return to the original position when mov
    ing to a certain position.
37.     if (fScale < 0.5f) fScale = 1.0f; //Change back to its original size when it is reduc
    ed to a certain extend.
38.     glutSwapBuffers();
39.
40. }
```

（6）

## 五、实验过程中的问题探讨。

1. 实验过程中一开始由于并不知道 idle()函数的作用是什么，于是没有调用 idle()函数，导致运行出来是三个静止的桌子，查询资料得知：调用 idle()函数可以实现连续动画。在程序中加上 idle()函数后运行正常，问题得到了解决。
2. 刚开始运行出来的结果左边两个桌子总是重叠在一起，找了很久代码方面的问题，最后发现其实只是初始给两个桌子安排的位置不合适，改了数据就不会重叠在一起了，总结了经验教训，之后再遇到类似问题知道该如何使画面显示正常。

## 六、实验结果及分析，心得

1. 实验结果详见压缩包 code\result 文件夹下的 Experiment2.exe，基本实现了试验所给样例的效果。

2.心得体会：

（1）知识方面：复习了上节课所讲述的内容，掌握了 OpenGL 的矩阵的使用，并理解了课程中矩阵变换的内容，了解了如何用矩阵去实现图形变换。

（2）其它启示：在遇到问题时可以仔细思考问题本身，从初始源头寻找问题出在哪里，而不是随便抓一处地方就开始检查代码的正确性，这样也许可以事半功倍。

## 七、源代码

```
1.  #include<gl/glut.h>
2.
3.  float fTranslate = 0.0f ;  //Translation factor
4.  float fRotate = 0.0f;  //Rotation factor
5.  float fScale = 1.0f;  //Scaling factor
6.
7.  void Draw_Cuboid(GLfloat x1, GLfloat x2, GLfloat y1, GLfloat y2, GLfloat z1, GLfloat z2)
     //Draw the cuboid
8.  {
9.      int i, j;
```

```
    GLfloat vertex[8][3] = {//Eight vertices of the cuboid
        {x1,y1,z1},
        {x1,y2,z1},
        {x2,y2,z1},
        {x2,y1,z1},
        {x1,y1,z2},
        {x1,y2,z2},
        {x2,y2,z2},
        {x2,y1,z2}
    };

    GLint flat[6][4] = {//Six planes of the cuboid
        {0,1,2,3},
        {1,2,6,5},
        {0,4,7,3},
        {2,3,7,6},
        {0,1,5,4},
        {4,5,6,7}
    };

    glBegin(GL_QUADS);  //Draw the cuboid
    for (i = 0; i < 6; i++) {
        for (j = 0; j < 4; j++) {
            glVertex3fv(vertex[flat[i][j]]);
        }
    }
    glEnd();
}

void Draw_Table()  //Draw the table
{
    Draw_Cuboid(0.0, 1.0, 0.0, 0.8, 0.6, 0.8);  //draw the desktop
    //Draw four table legs
    Draw_Cuboid(0.1, 0.3, 0.1, 0.3, 0.0, 0.6);
    Draw_Cuboid(0.7, 0.9, 0.1, 0.3, 0.0, 0.6);
    Draw_Cuboid(0.1, 0.3, 0.5, 0.7, 0.0, 0.6);
    Draw_Cuboid(0.7, 0.9, 0.5, 0.7, 0.0, 0.6);
}

void reshape(int width, int height)
{
    if (height == 0)                                        // Prevent A divide by zero

    {
```

```
53.         height = 1;                                         // Making height equal one
54.     }
55.
56.     glViewport(0, 0, width, height);                        // Reset the current viewport
57.
58.     glMatrixMode(GL_PROJECTION);                            // Select the projection matrix
59.     glLoadIdentity();                                       // Reset the projection matrix
60.
61.     // Calculate the aspect ratio of the window
62.     gluPerspective(45.0f, (GLfloat)width / (GLfloat)height, 0.1f, 100.0f);
63.
64.     glMatrixMode(GL_MODELVIEW);                             // Select the modelview matrix
65.     glLoadIdentity();                                       // Reset the modelview matrix
66. }
67.
68. void idle()  //Perform continuous animation
69. {
70.     glutPostRedisplay();
71. }
72.
73. void redraw()
74. {
75.     //display in wireframe mode
76.      glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
77.
78.     glClear(GL_COLOR_BUFFER_BIT);  //Clear window
79.     glLoadIdentity();  //Reset the current modelview matrix
80.
81.     //Translation
82.     glPushMatrix();
83.         glTranslatef(-2.7f, 0.0f, -6.0f);  //Place the table left
84.         glTranslatef(0.0f, fTranslate, 0.0f);  //Translate in Y direction
85.         Draw_Table();  //Draw table
86.     glPopMatrix();
87.
88.     //Rotation
89.     glPushMatrix();
90.         glTranslatef(0.0f, 0.4f, -6.0f);  //Place the table at center
91.         glRotatef(fRotate , 0 , 1.0f , 0);  //Rotate around Y direction
92.         glTranslatef(-0.5f, -0.4f, 0.0f);  //Rotate the table around the center
93.         Draw_Table();  //Draw table
94.     glPopMatrix();
95.
```

```
96.     //Scaling
97.     glPushMatrix();
98.         glTranslatef(1.7f, 0.0f, -6.0f);  //Place the table right
99.         glScalef(fScale, fScale, fScale);  //Scale
100.         Draw_Table();  //Draw table
101.     glPopMatrix();
102.
103.     //Update the factors
104.     fTranslate += 0.005f;
105.     fRotate += 0.5f;
106.     fScale -= 0.005f;
107.
108.     if (fTranslate > 0.5f) fTranslate = 0.0f;  //Return to the original position when mo
    ving to a certain position.
109.     if (fScale < 0.5f) fScale = 1.0f; //Change back to its original size when it is redu
    ced to a certain extend.
110.     glutSwapBuffers();
111.
112. }
113. int main(int argc, char *argv[])
114. {
115.     glutInit(&argc, argv);  //Initialize the glut library
116.     glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);  //Specify the window display mode that
     the function glutCreateWindow will create. RGB mode Double buffering
117.     glutInitWindowPosition(100, 100);  //Set the window position, which is the position
    of the top left corner of the window relative to the entire screen
118.     glutInitWindowSize(640, 480);  // Set the window size (may be covered by other windo
    ws)
119.     glutCreateWindow("Exercise2");  //Set the window title
120.     glutDisplayFunc(redraw);  //Register a draw callback function that specifies the fun
    ction to call when the window content needs to be redrawn
121.     glutReshapeFunc(reshape);  //the callback function when the registration window size
     changes.
122.     glutIdleFunc(idle);  //Perform continuous animation
123.     glutMainLoop();  //Glut event processing loop
124.     return 0;
125. }
```