云南大学

本科实验报告

课程名称： 计算机图形学实验

实验名称： 实验五 OpenGL下的几何变换

学院（系）： 信息学院

专 业： 计算机科学与技术

年 级： 2020级

姓 名： 胡诚皓

学 号： 20201060330

指导教师： 吴昊

成 绩：

2022年 5 月 4 日

**一. 实验目的**

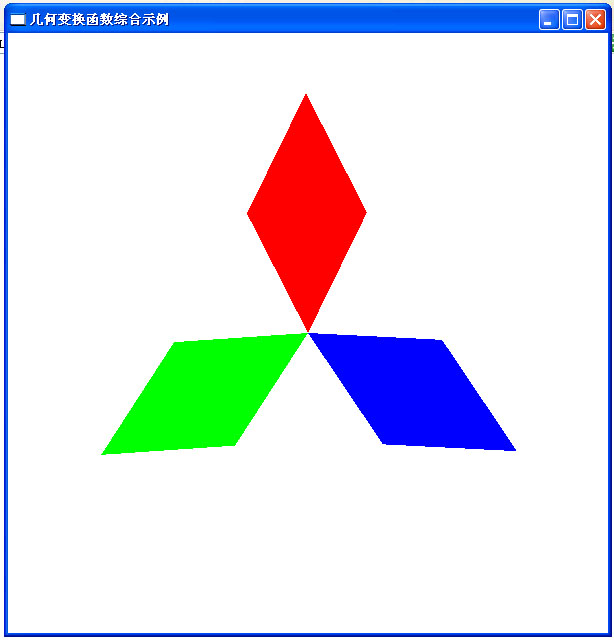
1. 理解变换的原理；

2. 掌握平移、旋转、缩放变换的方法；

3. 掌握以上方法的组合变换。

**二. 实验内容**

根据实验指导书中的提示，完成上机实做题，绘制指定图形：



要求通过先绘制一个菱形，再通过几何变换绘制三个不同位置、旋转角度、颜色的菱形。需要使用提供的绘制以原点为中心的函数drawDiamond。

**三. 实验代码**

#include <GL/glut.h>

#define PROJECTION\_X\_MIN -5.0

#define PROJECTION\_X\_MAX 5.0

#define PROJECTION\_Y\_MIN -5.0

#define PROJECTION\_Y\_MAX 5.0

void drawDiamond(void) //绘制中心在原点的菱形（宽>高）

{

glBegin(GL\_POLYGON); //顶点指定需要按逆时针方向

glVertex2f(0.0f, -1.0f);//下点

glVertex2f(2.0f, 0.0f);//右点

glVertex2f(0.0f, 1.0f);//上点

glVertex2f(-2.0f, 0.0f);//左点

glEnd();

}

void init(void) {

glClearColor(1.0, 1.0, 1.0, 0.0);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(PROJECTION\_X\_MIN, PROJECTION\_X\_MAX, PROJECTION\_Y\_MIN, PROJECTION\_Y\_MAX);

glMatrixMode(GL\_MODELVIEW);

}

void Draw(void) {

glClear(GL\_COLOR\_BUFFER\_BIT);

// 先画上方的菱形（红色）

glLoadIdentity();// 每次都从单位矩阵开始变换

glColor3f(1.0f, 0.0f, 0.0f);// 设置颜色为红色

glRotatef(0.0f, 0.0f, 0.0f, 1.0f);// 为了保持代码一致，不旋转写成旋转0°

glTranslatef(0.0f, 2.0f, 0.0f);// 沿y轴向上平移2个单位

glRotatef(90.0f, 0.0f, 0.0f, 1.0f);// 沿x轴旋转90°，使得菱形高>宽

drawDiamond();

glFlush();

// 画左侧的菱形（绿色）

glLoadIdentity();

glColor3f(0.0f, 1.0f, 0.0f);// 设置颜色为绿色

glRotatef(120.0f, 0.0f, 0.0f, 1.0f);// 沿x轴旋转120°

glTranslatef(0.0f, 2.0f, 0.0f);// 沿y轴向上平移2个单位

glRotatef(90.0f, 0.0f, 0.0f, 1.0f);// 沿x轴旋转90°，使得菱形高>宽

drawDiamond();

glFlush();

// 最后画右侧的菱形（蓝色）

glLoadIdentity();

glColor3f(0.0f, 0.0f, 1.0f);// 设置颜色为蓝色

glRotatef(240.0f, 0.0f, 0.0f, 1.0f);// 沿x轴旋转240°

glTranslatef(0.0f, 2.0f, 0.0f);// 沿y轴向上平移2个单位

glRotatef(90.0f, 0.0f, 0.0f, 1.0f);// 沿x轴旋转90°，使得菱形高>宽

drawDiamond();

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_SINGLE);

glutInitWindowPosition(100, 100);

glutInitWindowSize(500, 500);

glutCreateWindow("3-diamond");

init();

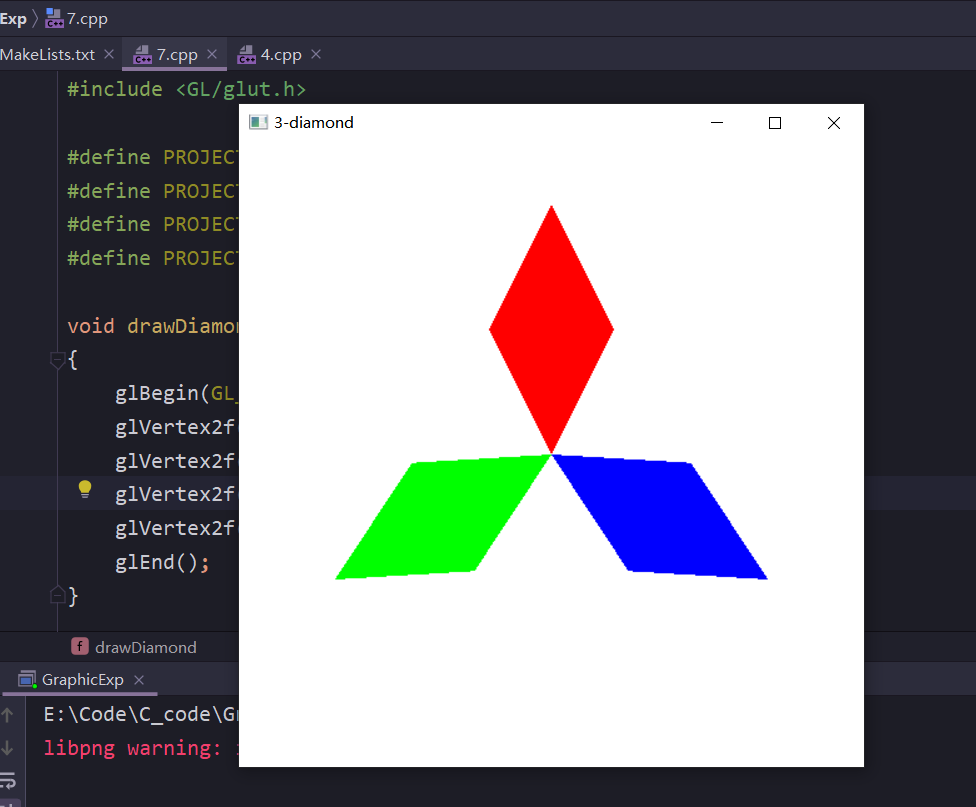
glutDisplayFunc(Draw);

glutMainLoop();

return 0;

}

**四. 实验结果**



**五. 实验分析**

本次实验要求画出的图形，是R、G、B的三个均匀分布在一个圆周上的菱形。要求中给出的画菱形的函数会得到一个横向的宽>高的菱形，也就是说，在实际绘制的时候要将其旋转90°以实现目标效果。此处采用的方式如下

（1）绘制每个菱形前都使用glLoadIdentity直接还原变换矩阵为初始状态

（2）分别旋转（坐标系）0°、120°、240°；

（3）向y轴方向平移2个单位；

（4）再旋转90°使得菱形是锐角朝外的；

（5）调用drawDiamond绘制。

经过试验中的尝试与观察，发现OpenGL自带的几个变换操作函数都是以“变换坐标系”的形式工作的。比如旋转90°，其实是把整个坐标系逆时针旋转了90°。并且，变换的最初始原点，是由gluOrtho2D函数指定的。下面是在实验中形成的对几个OpenGL自带变换函数的理解。

glTranslatef(GLfloat x, GLfloat y, GLfloat z)，沿着向量(x, y, z)进行平移，由于此次实验是在二维平面上进行操作，因此可以将z轴上的平移量设置为0。

glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z)，以向量(x, y, z)为旋转轴，根据右手系方向旋转angle度。们在二维平面进行旋转时，其实单纯以z轴为旋转轴即可。因此，将旋转轴直接设置为z轴方向的一个单位向量(0, 0, 1)。

glScalef(GLfloat x, GLfloat y, GLfloat z)，这个就比较简单，设置x、y、z轴上的缩放因子。

OpenGL中还有矩阵堆栈的功能，对于需要进行一些复杂模型的多次变换来说十分有用。glLoadIdentity函数可以用于将“当前”的变换矩阵直接置为单位矩阵，glPushMatrix函数会复制“当前”变换矩阵状态为一个副本，压入OpenGL内部的矩阵栈中；而glPopMatrix函数会从矩阵栈中弹出一个副本，覆盖“当前”变换矩阵，也就是说“当前”的变换矩阵的状态会丢失。总的来说，glPushMatrix和glPopMatrix的配对使用能够消除上一次的变换对本次变换的影响。