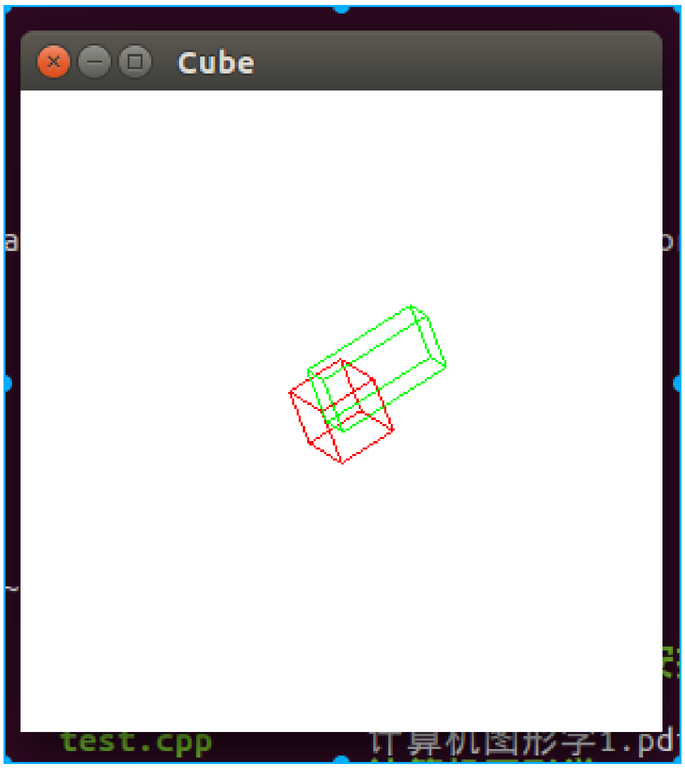
1. 实验目的

- 掌握图形的二维基本变换及其OpenGL实现。
- 掌握图形窗口到视区的变换
- 掌握图形的三维几何变换及其OpenGL实现

2. 题目一：利用OpenGL实现一个立方体关于参考点（10.0,20.0,10.0）进行放缩变换，放缩因子为（2.0,1.0,0.5）

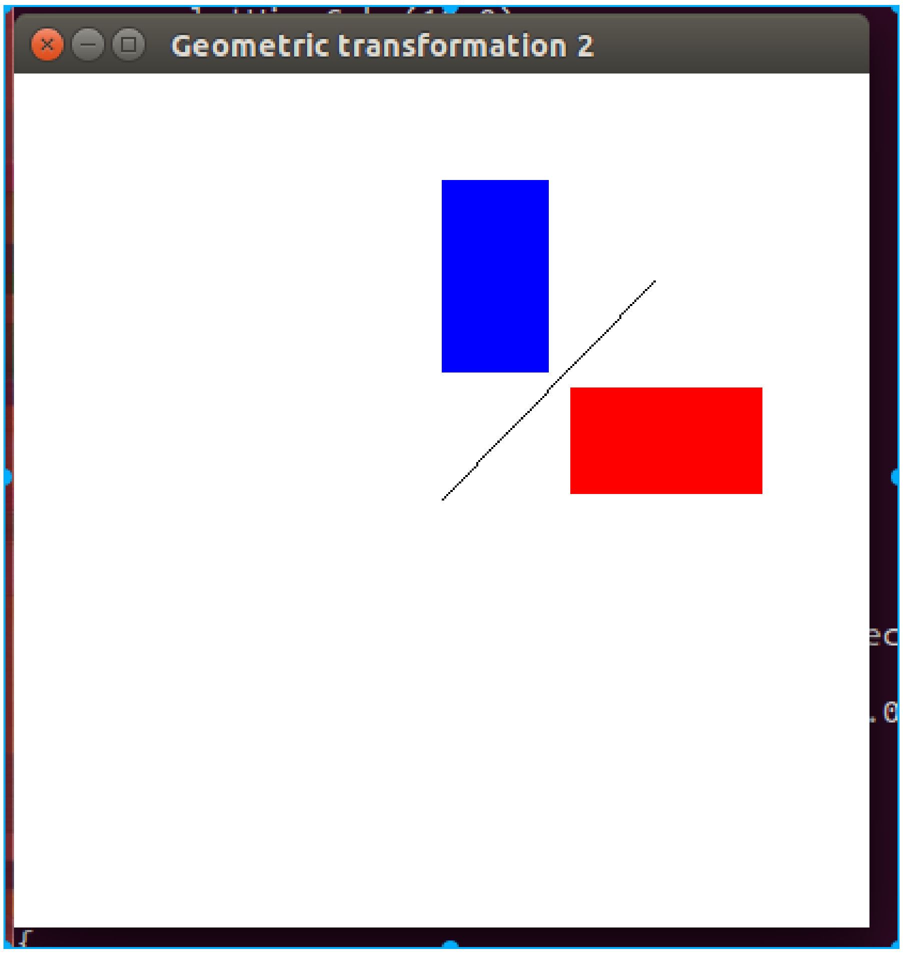
2.1. 实验结果



红色立方体为初始立方体绕着向量（1,1,1）旋转45°的图像，绿色立方体为变换后的结果

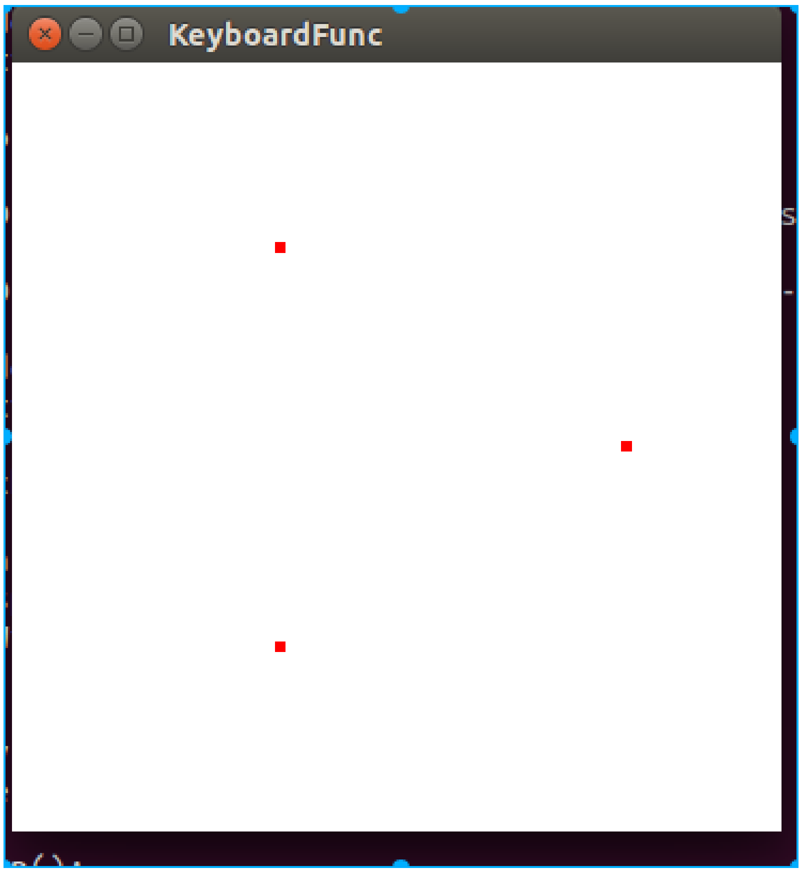
3. 题目二：利用OpenGL实现一个矩形关于y=x+5对称的新图形。

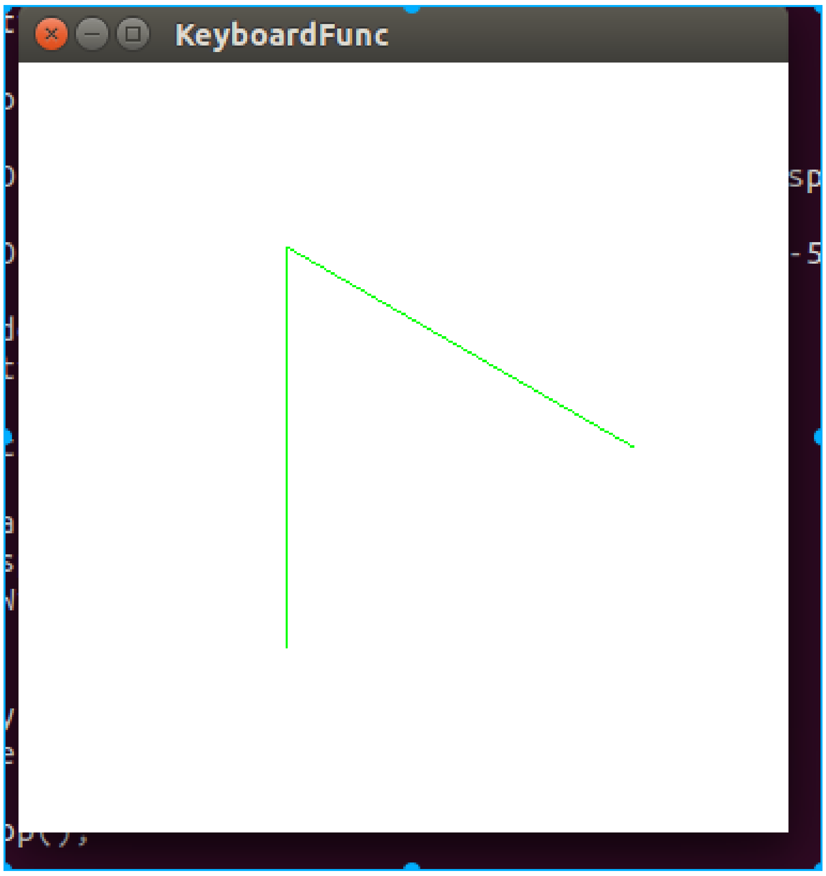
3.1. 实验结果

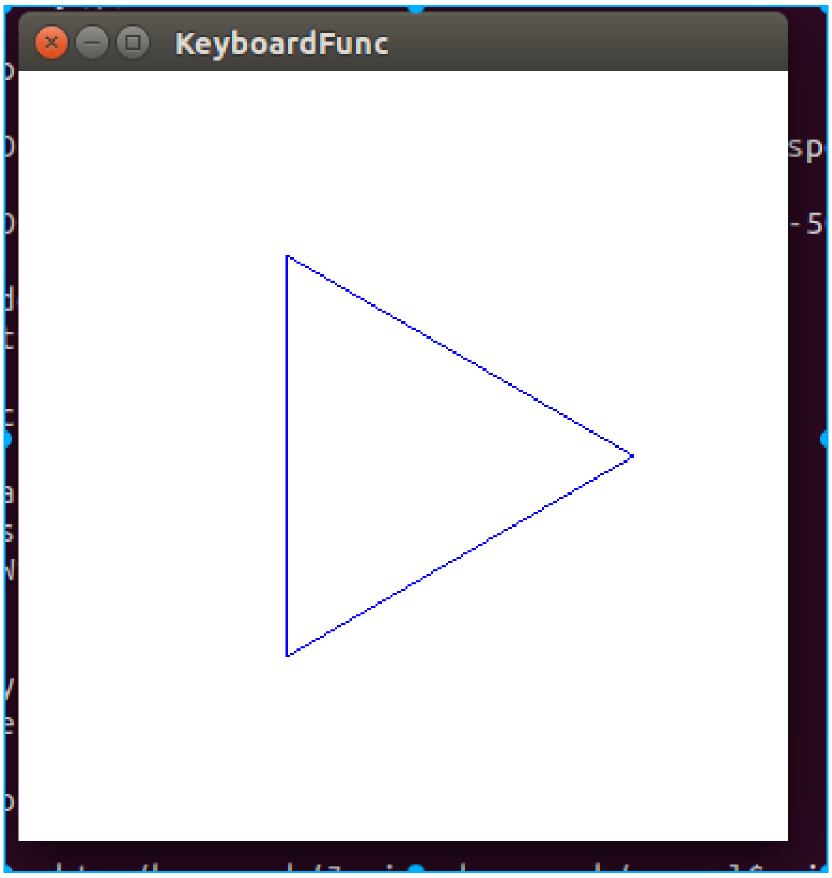


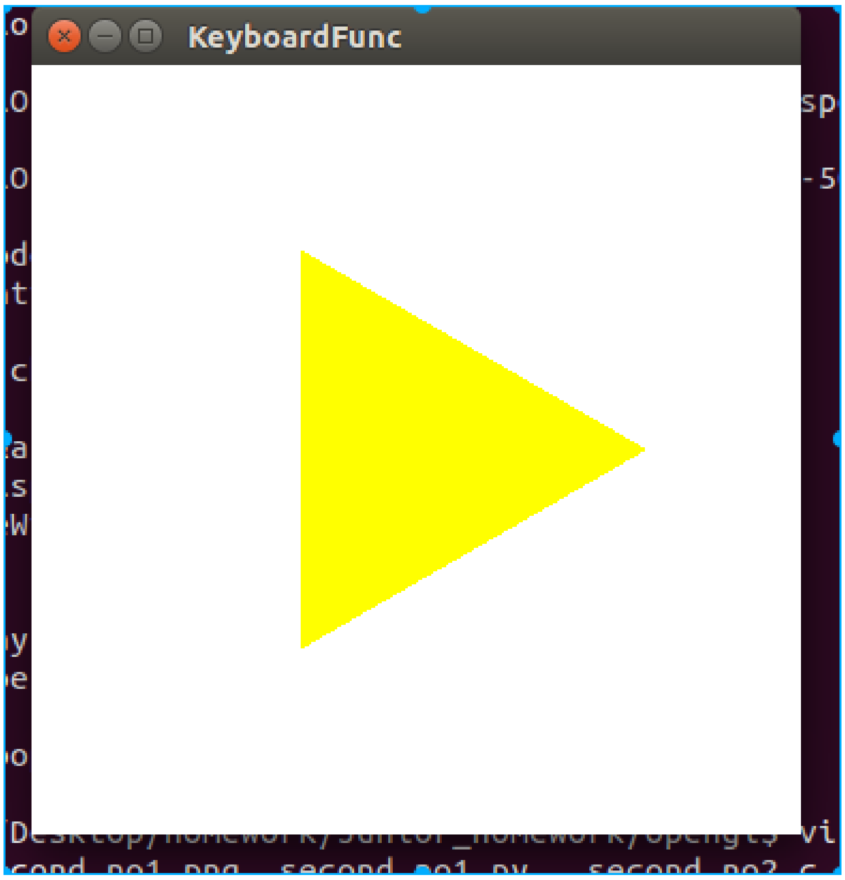
4. 题目三：通过定义键盘回调函数，每按一次空格键，让三个点依次完成画点、画线、画三角形、让三角形平移和缩放，并让三角形沿三角形中心旋转起来。

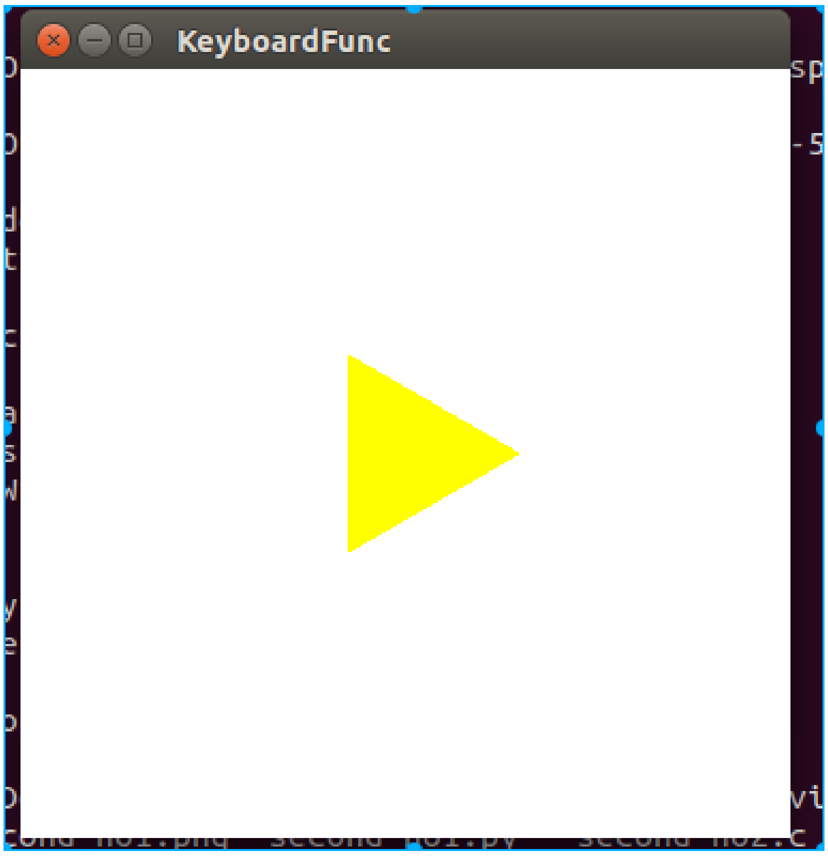
4.1. 实验结果

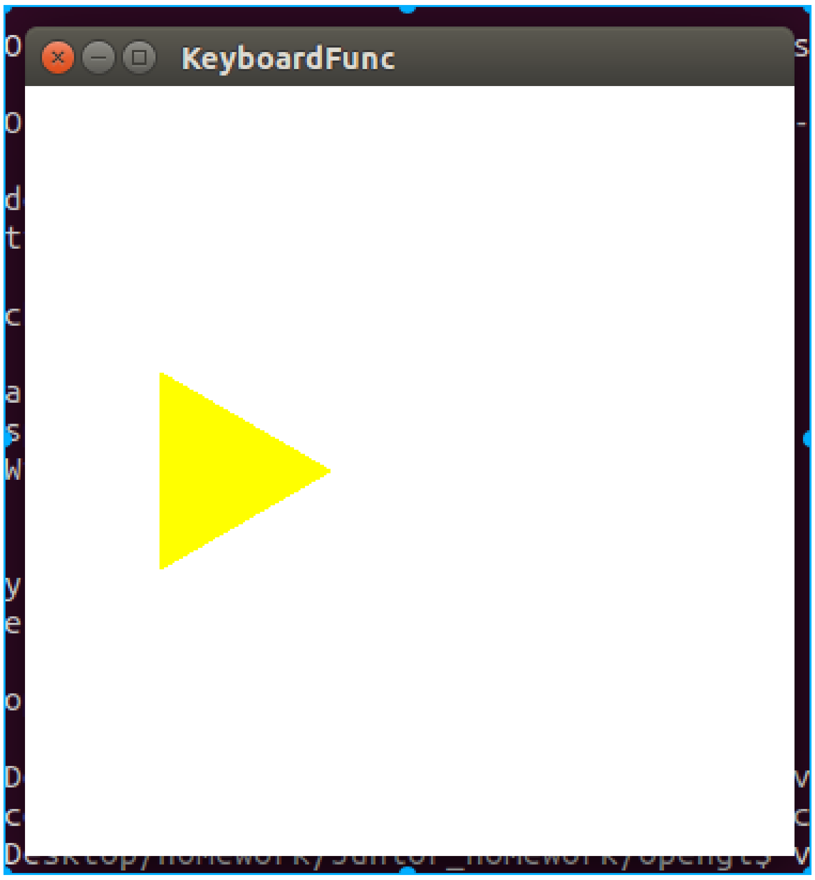

KeyboardFunc

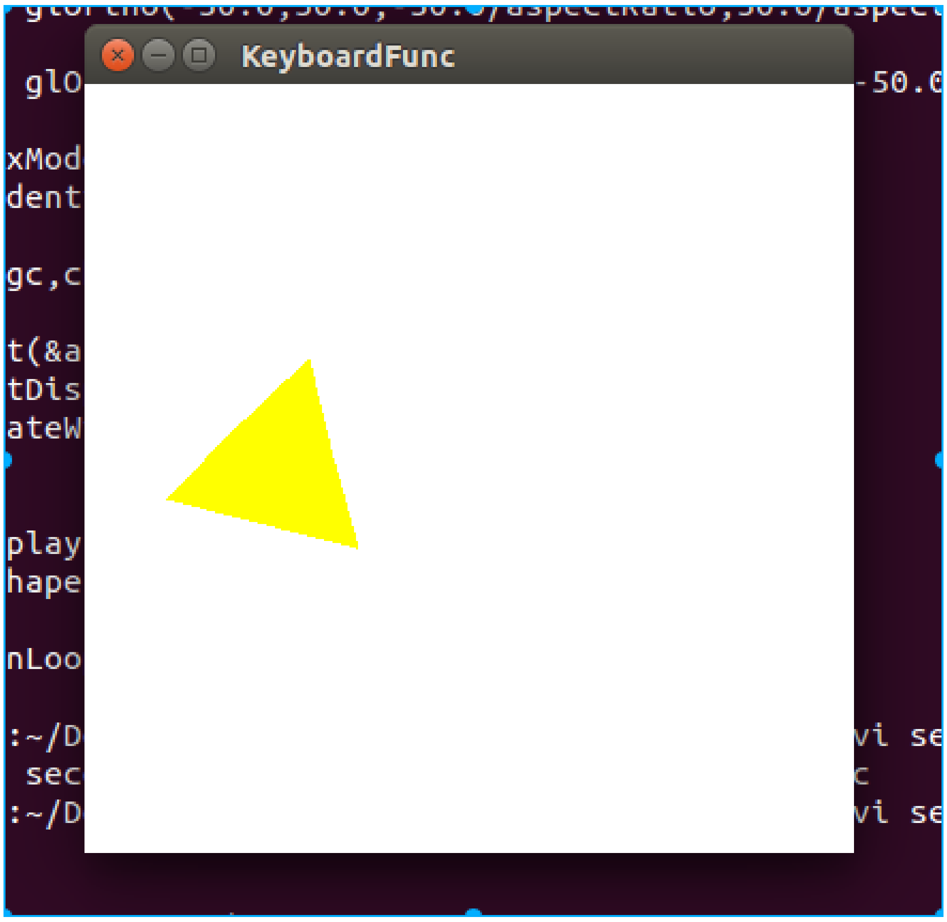

KeyboardFunc


KeyboardFunc


KeyboardFunc


KeyboardFunc


KeyboardFunc



5. 心得体会

通过本次上机实验，掌握了图形的二维变换和三维变换的算法实现，并且加深了课堂上学过的关于图形二维和三维变换的原理的理解，巩固了理论知识。

6. 源代码

6.1. 题目一

```
#include <GL/glut.h>
void init()
{
    glClearColor(1.0,1.0,1.0,1.0);
}
void RenderScene()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0,0.0,0.0);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    glRotatef(45.0,1.0,1.0,1.0);//绕着向量（1,1,1）所指定的轴旋转45°
    glPushMatrix();//保存矩阵状态
    glutWireCube(10.0);
//    glColor3f(0.0,0.0,1.0);
//    glTranslatef(0.0,20.0,0.0); //沿y轴正方向移动20个单位
//    glutWireCube(10.0);
    glPopMatrix();//恢复矩阵状态

    glColor3f(0.0,1.0,0.0);

    glTranslatef(-10.0,-20.0,-10.0);
    glScalef(2.0,1.0,0.5);  //在x和z轴放大2倍，而y轴保持原大小不变
    glTranslatef(10.0,20.0,10.0);
    glutWireCube(10.0);

    glFlush();
}

void ChangeSize(GLsizei w,GLsizei h)
{
    GLfloat aspectRatio;
    if(h==0)
        h = 1;
    glViewport(0,0,w,h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    aspectRatio = (GLfloat)w/(GLfloat)h;
    if(w<=h)
        glOrtho(-50.0,50.0,-50.0/aspectRatio,50.0/aspectRatio,-50.0,50.0);
    else
        glOrtho(-50.0*aspectRatio,50.0*aspectRatio,-50.0,50.0,-50.0,50.0);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
int main(int argc,char *argv[])
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_RGB|GLUT_SINGLE);
    glutCreateWindow("Cube");

    init();
    glutDisplayFunc(RenderScene);
    glutReshapeFunc(ChangeSize);

    glutMainLoop();
}
```

6.2. 题目二

```
#include <GL/glut.h>
void init()
{
    glClearColor(1.0,1.0,1.0,0.0);
//    gluOrtho2D(-300,300,-200,200);
}
void RenderScene()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_POLYGON);
    glColor3f(0.0,0.0,1.0);
    glVertex2f(0,60);
    glVertex2f(0,150);
    glVertex2f(50,150);
    glVertex2f(50,60);
    glEnd();
    glColor3f(0.0,0.0,0.0);

    glBegin(GL_LINES);
    glVertex2f(0,0);
    glVertex2f(100,103);
    glEnd();
//    glRectf(0.0,10.0,50.0,100.0);


//    glColor3f(1.0,0.0,0.0);
//    glTranslatef(0.0,-5.0,0.0);
//    glRectf(50.0,100.0,200.0,150.0);

//    glRectf(50.0,100.0,200.0,150.0);
//

//    glLoadIdentity();
//    glColor3f(0.0,1.0,0.0);
//    glTranslatef(0,50.0,0.0);
//    glRotatef(-90.0,0.0,0.0,1.0);
//
    glBegin(GL_POLYGON);
    glColor3f(1.0,0.0,0.0);
    glVertex2f(60,3);
    glVertex2f(150,3);
    glVertex2f(150,53);
    glVertex2f(60,53);
    glEnd();



//    glRectf(0.0,10.0,50.0,100.0);

//    glRotatef(-90,0,0,1);
//    glLoadIdentity();
//    glColor3f(1.0,1.0,0.0);
//    glScalef(0.5,1.0,1.0);
//    glRectf(50.0,100.0,200.0,150.0);

    glFlush();
}
void ChangeSize(GLsizei w,GLsizei h)
{
    float ratio;
    if(h==0)
        h = 1;
    ratio = (GLfloat)w/(GLfloat)h;
    glViewport(0,0,w,h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    if(w<=h)
        gluOrtho2D(-200.0,200.0,-200.0/ratio,200.0/ratio);
    else
        gluOrtho2D(-200.0*ratio,200.0*ratio,-200.0,200);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
void main(int argc,char *argv[])
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_RGB|GLUT_SINGLE);
    glutInitWindowPosition(50,50);
    glutInitWindowSize(400,400);
    glutCreateWindow("Geometric transformation 2");

    init();
    glutDisplayFunc(RenderScene);
    glutReshapeFunc(ChangeSize);
    glutMainLoop();
}
```

6.3. 题目三

```cpp
#include <stdlib.h>
#include <stdio.h>
#include <GL/glut.h>
#include "math.h"
#include "bits/stdc++.h"
using namespace std;
int currentMode = 0;
const int ModeNums = 7;
float theta=0.0;
float PI=3.1415926;
float R=6.0,n=3;
void init()
{
    glClearColor(1.0,1.0,1.0,1.0);
}
void myKey( unsigned char key, int x, int y) //��ÖASCII��Ö������������ĵ�ǰx��yλ��Ҳ������ء�
{
    switch(key)
    {
        case ' ': currentMode = (currentMode+1)%ModeNums;
                  glutPostRedisplay();
                  break;
        case 27:  exit(-1);
    }
}

void mydisplay(){
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_TRIANGLES);
    int i=0;
    for(int i=0;i<n;i++){
        glVertex2f(R*cos(theta+i*2*PI/n),R*sin(theta+i*2*PI/n));
    }
    glEnd();
    glFlush();
}

void myidle(){

    theta+=0.001;
    if(theta>=2*PI) theta-=2*PI;
    mydisplay();
}


void RenderScene()
{
    cout << currentMode << endl;
    glClear(GL_COLOR_BUFFER_BIT);
    switch(currentMode)
    {
        case 0:
                glutIdleFunc(NULL);
                glPointSize(5);
                glBegin(GL_POINTS);
                theta=0.0;
                glColor3f(1.0,0.0,0.0);
                break;
        case 1:
                glutIdleFunc(NULL);
                glBegin(GL_LINE_STRIP);
                theta=0.0;
                glColor3f(0.0,1.0,0.0);
                break;
        case 2:
                glutIdleFunc(NULL);
                glBegin(GL_LINE_LOOP);
                theta=0.0;
                glColor3f(0.0,0.0,1.0);
                break;
        case 3:
                glutIdleFunc(NULL);
                glBegin(GL_TRIANGLES);
                theta=0.0;
                glColor3f(1.0,1.0,0.0);
                break;
        case 4:
                glutIdleFunc(NULL);
                glScalef(0.5,0.5,0.5);
                glBegin(GL_TRIANGLES);
                break;


        case 5:
                glutIdleFunc(NULL);
                glTranslatef(-10,0,0);
                glBegin(GL_TRIANGLES);
                break;
        case 6:
                glutIdleFunc(myidle);
                return;

    }
    int i=0;
    for(int i=0;i<n;i++){
        glVertex2f(R*cos(theta+i*2*PI/n),R*sin(theta+i*2*PI/n));
    }

    glEnd();
    glFlush();
//    printf("%lf,%lf\n",R*(cos(theta)+i*2*PI/n),R*sin(theta+i*2*PI/n));
}
void ChangeSize(GLsizei w,GLsizei h)
{
    float ratio;
    if(h==0)
        h = 1;
    glViewport(0,0,w,h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    ratio = (float)w/(float)h;
    if(w<=h)
        gluOrtho2D(-10.0,10.0,-10.0/ratio,10.0/ratio);
    else
        gluOrtho2D(-10.0*ratio,10.0*ratio,-10.0,10.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}




int main(int argc,char *argv[])
{
    printf("%lf\n",sin(1/2.0*PI));
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowPosition(50,50);
    glutInitWindowSize(360,360);
    glutCreateWindow("KeyboardFunc");

    init();
    glutDisplayFunc(RenderScene);
    glutReshapeFunc(ChangeSize);
    glutKeyboardFunc(myKey);  //Ï��ǰ�������ü��ص�������
    printf("Press space to continue,press escape to exit!\n");
    glutMainLoop();
}
```