实验三　正交投影和透视投影的实现

姓名 刘子言　学号 20002462　专业班级 计 203　成绩_____

实验日期 2022.5.5　实验地点 线上　指导教师(签名) 李建华

一．实验目的

1、　定义多视区。

2、　定义三维正交观察可视体，设置照相机的位置，实现“正视图 XOZ(V)”、“侧视图 YOZ(W)”和“俯视图 XOY(H)” 的四面体三视图。

3、　定义三维透视观察可视体，交互控制照相机的位置，透视观察四面体。

二．实验工具与设备

计算机，开发软件为 CodeBlocks （配置 OpenGL）

三、实验内容

1、 在本科学习平台（s.ecust.edu.cn）资料栏下，下载以下 3 个文件

① setLookAt.h：给出三维坐标点point3D的定义和建立视角视觉效果函数setLookAt ()。

② getTetrahedron.h：给出4个三维坐标点ABCD，定义四面体图段。

③ getAxis.h：给出轴长，定义三维坐标轴图段，x轴为红色、y轴为绿色、z轴为蓝色。

2、 给定的数据定义以及 main()函数（见附件 1），在此基础上（如图 1）完成以下要求：

```
GLuint objectID,axisID;//四面体对象ID、坐标轴ID、
int winWidth = 400, winHeight = 200; //窗口的宽度和高度
static float angle = 45.0;      //angle绕坐标轴的旋转角
static GLfloat xRot = 0.0f;  //旋转参数
static GLfloat yRot = 0.0f;  //旋转参数
static GLfloat zRot = 0.0f;  //旋转参数
static GLsizei iMode = 1;       //菜单值
static point3D s_eye(5.0,5.0,5.0); //观察点位置
static point3D s_at(0.0,0.0,0.0);  //视点中心
static point3D A(2.0f,0.0f,0.0),B(2.0f,1.0f,0.0),C(0.0f,1.0f,0.0),D(1.0f,1.0f,1.0);//如图7-41
```

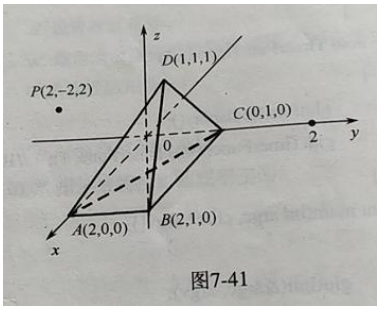


图7-41

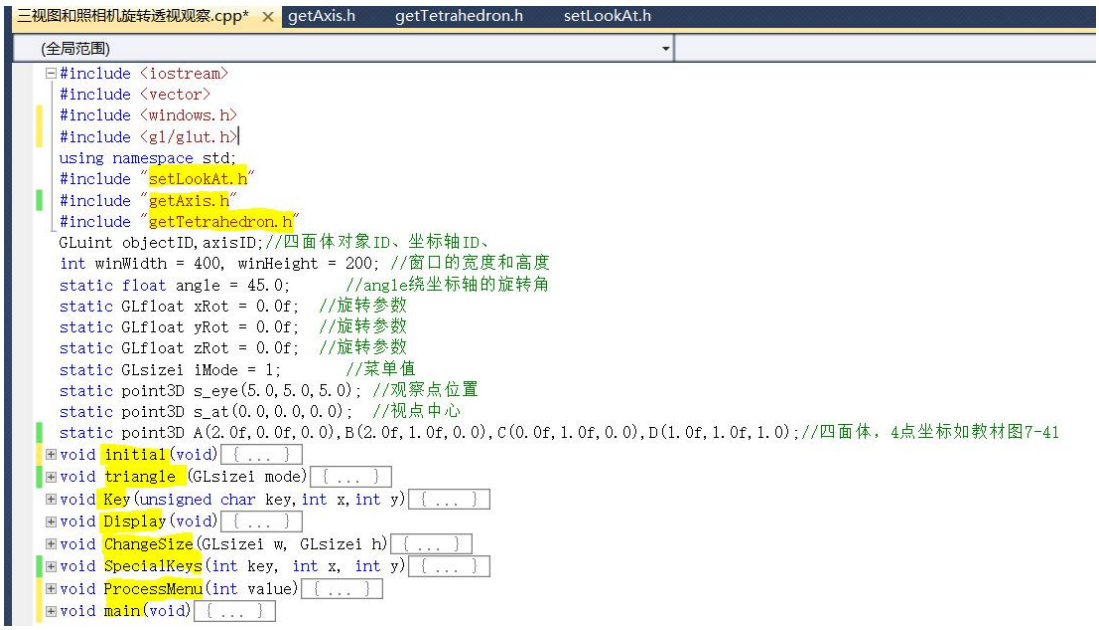


图1 实验三

① 参考p171程序6-2，使得窗口显示左右2个视区，左视区是正交投影，右视区是透视投影。

a) 利用GLint vp[4]; glGetIntegerv(GL_VIEWPORT ,vp); 获得视区的左下角坐标、宽度和高度信息，

b) 利用glOrtho(gldouble left, gldouble right, gldouble bottom, gldouble top,gldouble zNear, gldouble zFar)，定义左视区的三维正交观察可视体；

c) 利用gluPerspective(gldouble fovy,gldouble aspect, gldouble zNear, gldouble zFar)，定义右视区的三视维透视观察可体。

② 定义主菜单，菜单项为“正视图XOZ(V)”、“侧视图YOZ(W)”和“俯视图XOY(H)”，菜单项值分别为1、2和3。

a) 根据菜单选择，通过给定函数setLookAt(iMode,s_eye,s_at)，控制左视区的照相机的位置，使得左视区正交投影四面体的三视图。

b) 根据菜单选择，根据右手原则，确定右视区照相机围绕某坐标轴旋转，利用键盘上（↑）、下（↓）功能键（GLUT_KEY_UP、GLUT_KEY_DOWN），控制照相机绕坐标轴的旋转的角度的递减或递增，通过给定函数setLookAt(int menuValue,float angle,point3D eye,point3D at)，建立右视区的照相机的移动目坐标，使得右视区透视投影四面体。

实验代码如下：

1、头文件

（1）"setLookAt.h"

（2）"getAxis.h"

（3）"getTetrahedron.h"

2、main.cpp

```
#include<iostream>
#include<vector>
#include<windows.h>
#include<gl/glut.h>
using namespace std;
```

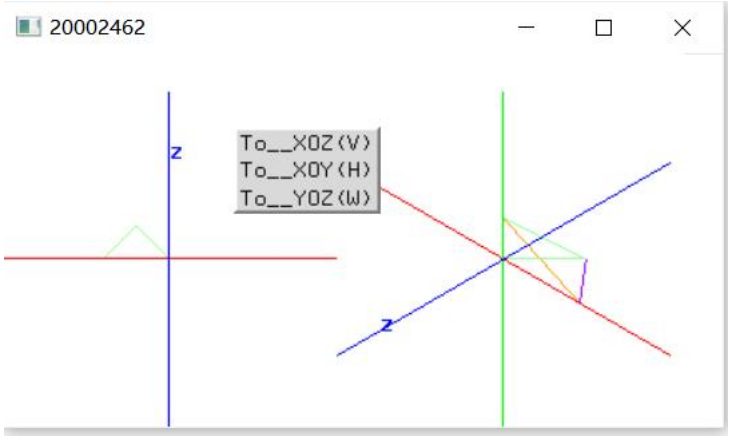
```
#include "setLookAt.h"       //给出三维坐标点point3D的定义和建立视角视觉效果函数
#include "getAxis.h"         //给出轴长，定义三维坐标轴图段，x轴为红色、y为绿、z为蓝
#include "getTetrahedron.h"  //给出4个三维坐标点ABCD，定义四面体图段

static GLsizei iMode = 1;                //三视图菜单项
float axisLength = 20.0;                 //三维坐标轴长
GLuint objectID,axisID;                  //四面体对象ID、坐标轴ID、
int winWidth = 400, winHeight = 200;     //窗口的宽度和高度
static float angle = 45.0;               //绕坐标轴的旋转角
static point3D s_eye(5.0,5.0,5.0);       //观察点位置
static point3D s_at(0.0,0.0,0.0);        //视点中心
static point3D A(2.0f,0.0f,0.0),B(2.0f,1.0f,0.0),C(0.0f,1.0f,0.0),D(1.0f,1.0f,1.0);
                                         //四面体四点坐标
void initial(void)
{
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);      //设置窗口背景颜色为白色
}
void ChangeSize(GLsizei w, GLsizei h)          //改变窗口大小
{
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);//多边形模式为线框
       winWidth = w; winHeight = h;
       glViewport(0, 0, w, h);                 //指定窗口显示区域
       glMatrixMode(GL_PROJECTION);            //设置投影参数
       glLoadIdentity();
}

void triangle(GLsizei mode)     //左右视图的绘制，mode决定选左or选右
{
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);//多边形模式为线框
    GLint vp[4];
    glGetIntegerv(GL_VIEWPORT ,vp); //获得视区的左下角坐标、宽度和高度信息
    int w = vp[2], h = vp[3];
    float aspect;
    if(w < h){
        aspect = (float)w/(float)h;
    }
    else{
        aspect = (float)h/(float)w;
    }
       glMatrixMode(GL_MODELVIEW);             //指定当前操作模型视图矩阵堆栈
       glLoadIdentity();                       //恢复初始坐标系，将当前点移动到屏幕中心

    if(mode==1)  //左视区
    {
```

```
        setLookAt(iMode, s_eye, s_at);//根据菜单iMode选择三视图，建立视角视觉效果
        glMatrixMode(GL_PROJECTION);  //设置投影参数
        glLoadIdentity();
        //定义左视区的三维正交观察可视体，创建一个正射投影矩阵
        glOrtho(-5.0, 5.0, -5.0, 5.0, 5.0, -5.0);

    }
    else    //右视区
    {
        //根据菜单iMode选择三视图，根据右手原则围绕某坐标轴旋转angle，建立视角视觉效果
        setLookAt(iMode, angle, s_eye, s_at);
        glMatrixMode(GL_PROJECTION);            //设置投影参数
        glLoadIdentity();
        gluPerspective(45.0f, aspect, 0.0f, 10.0f);//定义右视区的三维透视观察可视体
    }
    glCallList(objectID);                    //显示列表所存储的函数
    glCallList(axisID);
}

void SpecialKeys(int key, int x, int y)      //功能键上/下键控制旋转角度
{
    if(key == GLUT_KEY_UP)                   //上键递减
    {
        angle = angle - 1;
    }
    else if(key == GLUT_KEY_DOWN)            //下键递增
    {
        angle = angle + 1;
    }
    glutPostRedisplay();                     //刷新
}

void ProcessMenu(int value)//处理菜单响应
{
    iMode = value;
    glutPostRedisplay();
}
void Display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 0.0, 0.0);
    axisID = getAxis(axisLength);             //根据轴长axisLength定义XYZ三维坐标系
    objectID = getTetrahedron( A, B, C, D);   //4点ABCD定义四面体图元
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);//多边形模式为线框
```

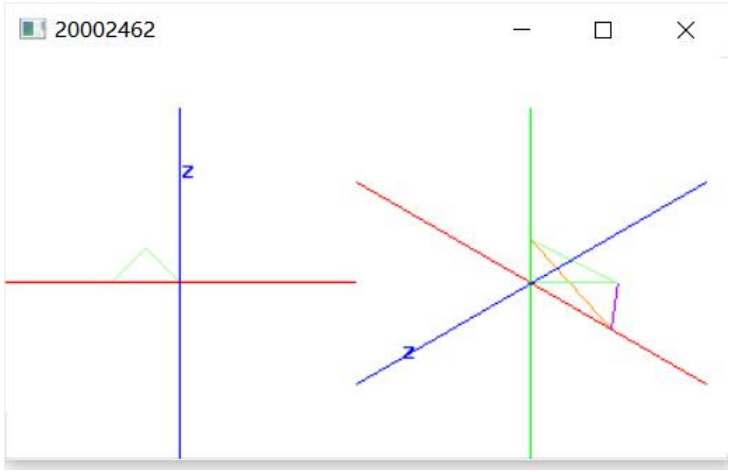
```
    //以下为左视区
    glViewport(0, 0, 200, 200);//指定从0，0开始，长宽均为200的视区
    triangle(1);
    //以下为右视区
    glViewport(200, 0, 200, 200);//指定从200，0开始，长宽均为200的视区
    triangle(2);
    glFlush();
}

int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE  | GLUT_RGB);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(winWidth, winHeight);
    glutCreateWindow("20002462");//多视区
    //创建菜单并定义菜单回调函数
    glutCreateMenu(ProcessMenu);
    glutAddMenuEntry("To__XOZ(V)",1);
    glutAddMenuEntry("To__XOY(H)",2);
    glutAddMenuEntry("To__YOZ(W)",3);
    glutAttachMenu(GLUT_RIGHT_BUTTON);//主菜单与鼠标右键关联
    initial();
    glutDisplayFunc(Display);        //指定窗口重绘响应函数
    glutReshapeFunc(ChangeSize);     //指定窗口大小改变响应函数
    glutSpecialFunc(SpecialKeys);    //指定功能键响应函数
    glutMainLoop();
    return 0;
}
```

运行截图如下：

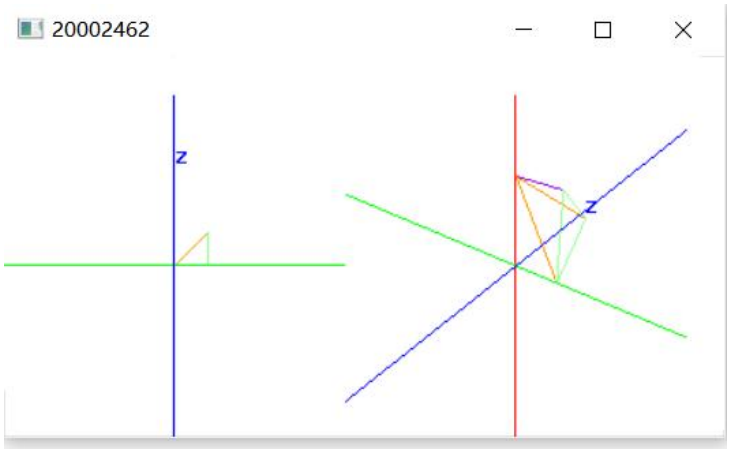
1、鼠标右击，选择菜单正视图 XOZ(V)、侧视图 YOZ(W)或俯视图 XOY(H)



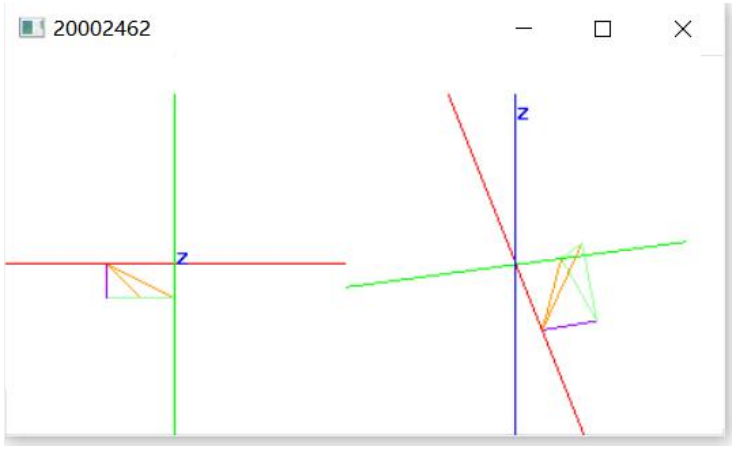
• 正视图



• 侧视图

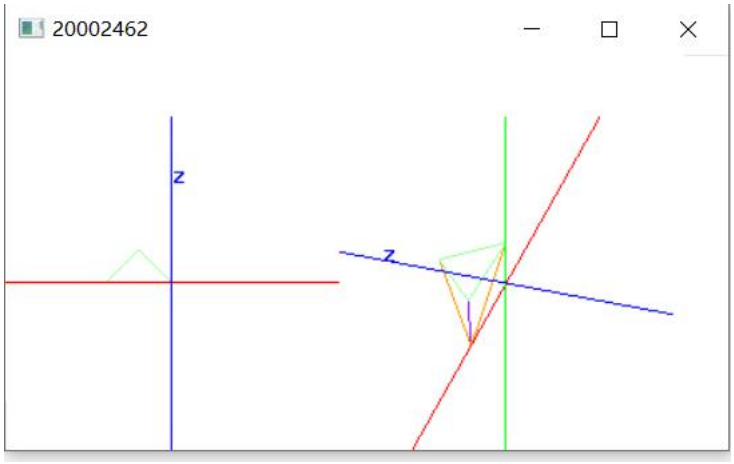


• 俯视图

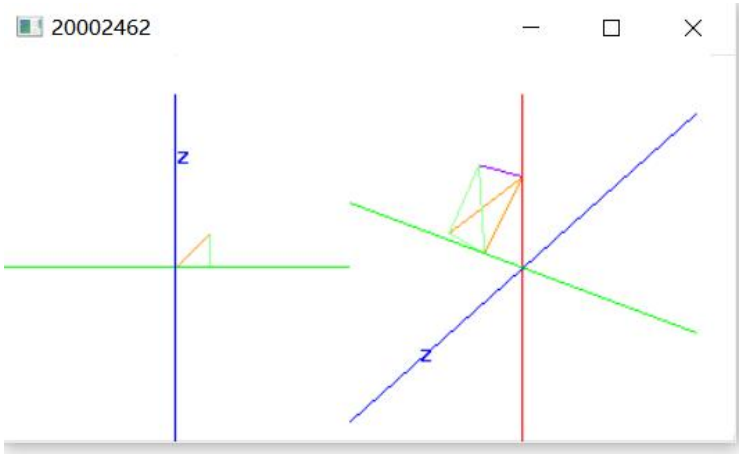


2、利用键盘上（↑）、下（↓）功能键，结合菜单选项控制照相机绕坐标轴旋转

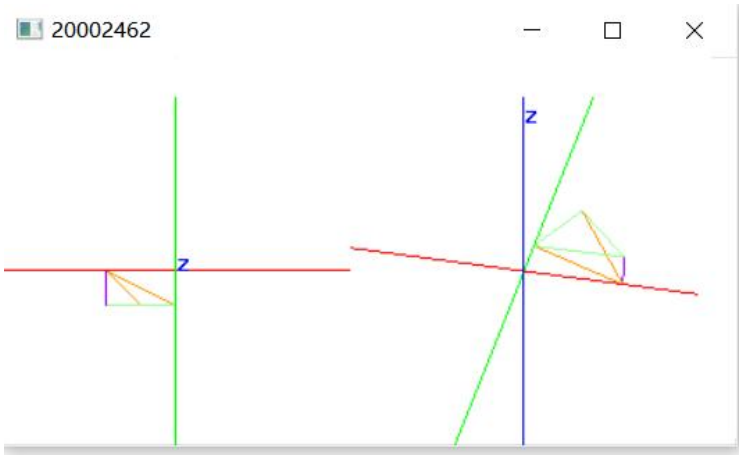
• 绕 y 轴转



• 绕 x 轴转



• 绕 z 轴转



四、拓展实验

1、在本科学习平台（s.ecust.edu.cn）资料栏下，有 1 个 ppt 文件“建立移动眼坐标，实现第一人称视角视觉效果”，利用键盘’x’、’y’、’z’，控制右视区的照相机的三维旋转角度，通过给定函数 setLookAt(const GLfloat x_angle,const GLfloat y_angle,const GLfloat z_angle,point3D eye,point3D at)，建立移动眼坐标，实现第一人称视角视觉效果。

已知 $R_x(\theta_x)\ R_y(\theta_y)R_z(\theta_z) =$ $\begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

实验代码如下：

1、头文件

（1）"setLookAt.h"

（2）"getAxis.h"

（3）"getTetrahedron.h"

2、main.cpp

在之前的 main.cpp 文件中做如下部分的修改：

（1）添加全局变量：

```
static GLfloat xRot = 0.0f;   //旋转参数
static GLfloat yRot = 0.0f;   //旋转参数
static GLfloat zRot = 0.0f;   //旋转参数
```

（2）增添函数：

```
void Key(unsigned char key, int x, int y)       //键盘响应函数
{
    if(key == 'x'){
        xRot = xRot + 1;
    }
    else if(key == 'y'){
        yRot = yRot + 1;
    }
    else if(key == 'z'){
        zRot = zRot + 1;
    }
    glutPostRedisplay();//刷新
}
```

（3）修改函数 void triangle(GLsizei mode)：

将语句 setLookAt(iMode, angle, s_eye, s_at) 注释

增添语句 setLookAt(xRot, yRot, zRot, s_eye, s_at)

```
//控制照相机的三维旋转角度，将场景坐标系映射到 eye 的观察坐标系，建立第一人称视角视觉效果
```

（4）修改函数 int main(int argc, char* argv[])：

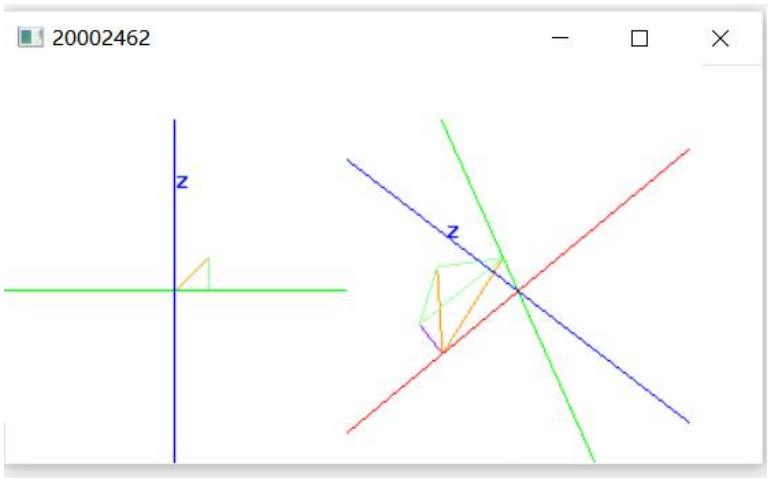
将语句 glutSpecialFunc(SpecialKeys) 注释

增添语句 glutKeyboardFunc(Key) //指定键盘响应函数

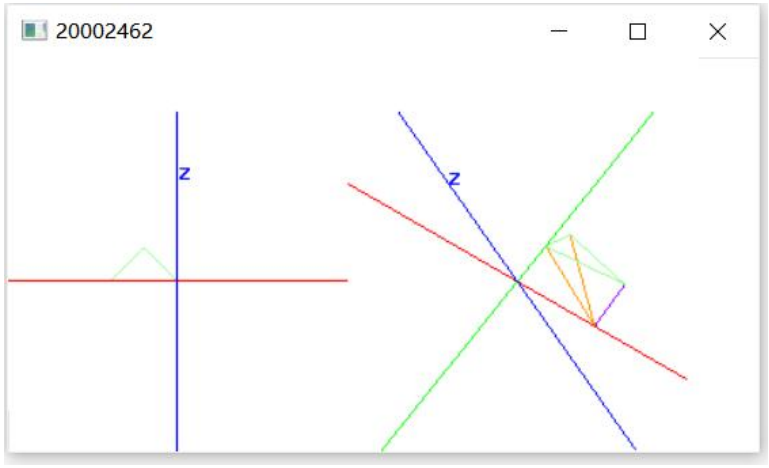
运行截图如下：

利用键盘’x’、’y’、’z’，控制右视区照相机的三维旋转角度

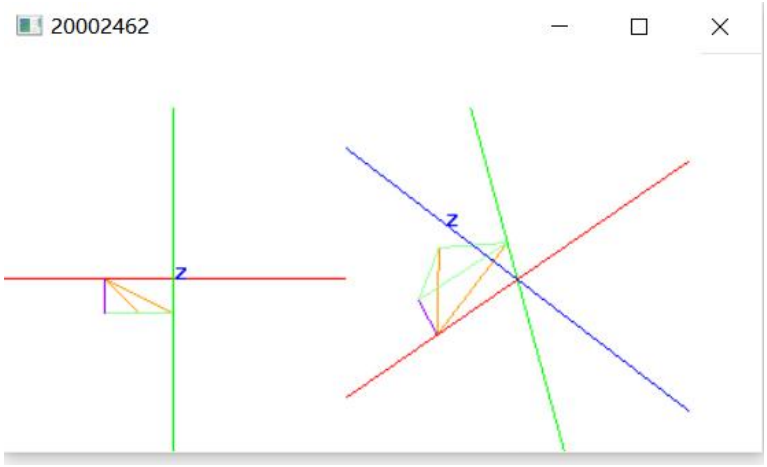
• 绕 x 轴转



• 绕 y 轴转



• 绕 z 轴转



五．思考题

1．透视投影和平行投影的用途在哪里？

答：

平行投影能精确地反映物体的实际尺寸，即不具有透视缩小性。另外平行线经过平行投影变换后仍保持平行。

透视投影的投影中心到投影面的距离是有限的，空间中任意一点的透视投影是投影中心到空间点构成的投影线与投影平面的交点。

平行投影和透视投影的本质区别在于透视投影的投影中心到投影面之间的距离是有限的，平行投影的投影中心到投影面之间的距离是无限的。与平行投影相比，透视投影的深度感更强，看上去更加真实，但透视投影不能真实地反映物体的精确尺寸和形状，透视投影的大小与物体到投影中心的距离有关。

附件 1：main()函数

```
void main(void){
    glutInitDisplayMode(GLUT_SINGLE  | GLUT_RGB);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(winWidth, winHeight);
    glutCreateWindow("多视区");
    int nGlutPolyMenu = glutCreateMenu(ProcessMenu);//创建菜单并定义菜单回调函数
    glutAddMenuEntry("正视图XOZ(V)",1);
    glutAddMenuEntry("俯视图XOY(H)",2);
    glutAddMenuEntry("侧视图YOZ(W)",3);
    glutAttachMenu(GLUT_RIGHT_BUTTON);//主菜单与鼠标右键关联
    initial();
    glutDisplayFunc(Display);       //指定窗口重绘响应函数
    glutReshapeFunc(ChangeSize);    //指定窗口大小改变响应函数
    //glutKeyboardFunc(Key);        //指定键盘响应函数
    glutSpecialFunc(SpecialKeys);   //指定功能键响应函数
    glutMainLoop();
}
```