

计算机图形学实验

实验四

学 号： 20191130038

姓 名： 郑广敏

专 业： 智能科学与技术

年 级： 2019级

任课教师： 吴昊

2020 年 11 月 27 日

**一、目的和要求：**

通过实验掌握下列知识:

1. 熟悉OpenGL Glut中常用的回调函数；
2. 掌握用鼠标对图形进行交互控制的方法；
3. 掌握用键盘对图形进行交互控制的方法；

**二、实验内容：**

## 常用回调函数的介绍

**1、用于显示的回调函数**

(1) void glutDisplayFunc(void (\*func)(void));

注册当前窗口的显示回调函数

参数:

func:形为void func()的函数,完成具体的绘制操作

这个函数告诉GLUT当窗口内容必须被绘制时,那个函数将被调用.当窗口改变大小或者从被覆盖的状态中恢复,或者由于调用glutPostRedisplay()函数要求GLUT更新时,执行func参数指定的函数.

(2) void glutReshapeFunc(void (\*func)(int width, int height));

指定当窗口的大小改变时调用的函数

参数:

func:形如void func(int width, int height)

处理窗口大小改变的函数.

width,height:为窗口改变后长宽.

这个函数确定一个回调函数,每当窗口的大小或形状改变时(包括窗口第一次创建),GLUT将会调用这个函数,这个回调函数接受这个窗口新的长宽作为输入参数.

**2、用于鼠标操作的回调函数**

(1) void glutMouseFunc(void (\*func)(int button, int state, int x, int y));

注册当前窗口的鼠标点击回调函数

参数:

func:形如void func(int button, int state, int x, int y);

button:鼠标的按键,为以下定义的常量

GLUT\_LEFT\_BUTTON:鼠标左键

GLUT\_MIDDLE\_BUTTON:鼠标中键

GLUT\_RIGHT\_BUTTON:鼠标右键

state:鼠标按键的动作,为以下定义的常量

GLUT\_UP:鼠标释放

GLUT\_DOWN:鼠标按下

x,y:鼠标按下式,光标相对于窗口左上角的位置

当点击鼠标时调用.

(2) void glutMotionFunc(void (\*func)(int x, int y));

当鼠标在窗口中按下并移动时调用glutMotionFunc注册的回调函数

(3) void glutPassiveMotionFunc(void (\*func)(int x, int y));

当鼠标在窗口中移动时调用glutPassiveMotionFunc注册的回调函数

参数:

func:形如void func(int x, int y);

x,y:鼠标按下式,光标相对于窗口左上角的位置,以像素为单位

(4) void glutEntryFunc(void (\*func)(int state));

设置鼠标进出窗口的回调函数

参数:

func:形如void func(int state);注册的鼠标进出回调函数

state:鼠标的进出状态,为以下常量之一

GLUT\_LEFT 鼠标离开窗口

GLUT\_ENTERED 鼠标进入窗口

当窗口取得焦点或失去焦点时调用这个函数,当鼠标进入窗口区域并点击时,state为GLUT\_ENTERED,当鼠标离开窗口区域点击其他窗口时,state为GLUT\_LEFT.

**3、用于键盘操作的回调函数**

(1) void glutKeyboardFunc(void (\*func)(unsigned char key, int x, int y));

注册当前窗口的键盘回调函数

参数:

func:形如void func(unsigned char key, int x, int y)

key:按键的ASCII码

x,y:当按下键时鼠标的坐标,相对于窗口左上角,以像素为单位

当敲击键盘按键时调用. ( 除了特殊按键,即glutSpecialFunc()中处理的按键,详见glutSpecialFunc() )

(2)void glutSpecialFunc(void (\*func)(int key, int x, int y));

设置当前窗口的键盘特定键的回调函数

参数:

Func:形如void func(int key, int x, int y);

key:按下的特定键,为以下定义的常量

GLUT\_KEY\_F1:F1功能键

GLUT\_KEY\_F2:F2功能键

GLUT\_KEY\_F3:F3功能键

GLUT\_KEY\_F4:F4功能键

GLUT\_KEY\_F5:F5功能键

GLUT\_KEY\_F6:F6功能键

GLUT\_KEY\_F7:F7功能键

GLUT\_KEY\_F8:F8功能键

GLUT\_KEY\_F9:F9功能键

GLUT\_KEY\_F10:F10功能键

GLUT\_KEY\_F11:F11功能键

GLUT\_KEY\_F12:F12功能键

GLUT\_KEY\_LEFT:左方向键

GLUT\_KEY\_UP:上方向键

GLUT\_KEY\_RIGHT:右方向键

GLUT\_KEY\_DOWN:下方向键

GLUT\_KEY\_PAGE\_UP:PageUp键

GLUT\_KEY\_PAGE\_DOWN:PageDown键

GLUT\_KEY\_HOME:Home键

GLUT\_KEY\_END:End键

GLUT\_KEY\_INSERT:Insert键

注意特别：上述键盘和鼠标的回调函数中，都返回光标的当前坐标x和y，注意这个坐标的原点是在窗口左上角，向右为X轴的正方向，向下为Y轴的正方向。而glut中的函数是把窗口的左下角设为原点，向右为X轴的正方向，向上为Y轴的正方向。所以假设窗口的高为winHeight，那么回调函数中光标（x，y）的位置换算到glut中实际上是（x，winHeight - y）。

注意:ESC,回车和delete键由ASCII码产生,即可以用glutKeyboardFunc()处理。当在键盘上敲击上述按键时调用该函数。注意与glutKeyboardFunc()的区别.

**4、其它常用的回调函数**

(8) void glutIdleFunc(void (\*func)(void));

设置空闲回调函数

参数:

func:形如void func(void);

当系统空闲时调用.

**5、实例**

**实例1、一个旋转的三角形――用空闲回调函数实现**

#include <GL/glut.h>

#include <math.h>

#define DEG\_TO\_RAD 0.017453 //角度转为弧度的参数，即 2\*PI/360

float theta=30.0; //直线与X轴正方向的夹角

float length=200.0; //直线的长度

float x=300.0, y=200.0; //直线的第一个端点

void init (void)

{

glClearColor (1.0, 1.0, 1.0, 0.0);

glMatrixMode (GL\_PROJECTION);

gluOrtho2D (0.0, 640.0, 0.0, 480.0);

}

void display (void)

{

glClear (GL\_COLOR\_BUFFER\_BIT);

glColor3f (1.0, 0.0, 0.0);

glBegin (GL\_POLYGON);

glVertex2f (x, y);

glVertex2f ( x + length\*cos(DEG\_TO\_RAD\*theta),

y + length\*sin(DEG\_TO\_RAD\*theta) );

glVertex2f ( x + length\*cos(DEG\_TO\_RAD\* (theta+30) ),

y + length\*sin(DEG\_TO\_RAD\* (theta+30)) );

glEnd ( );

glFlush ( );

}

void idleFunc()

{

theta += 0.1;

if (theta>360) theta -=360;

glutPostRedisplay(); //重新调用绘制函数

}

void main (int argc, char\*\* argv)

{

glutInit (&argc, argv);

glutInitDisplayMode (GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowPosition (50, 100);

glutInitWindowSize (640, 480);

glutCreateWindow ("Draw Triangle with Single Buffer");

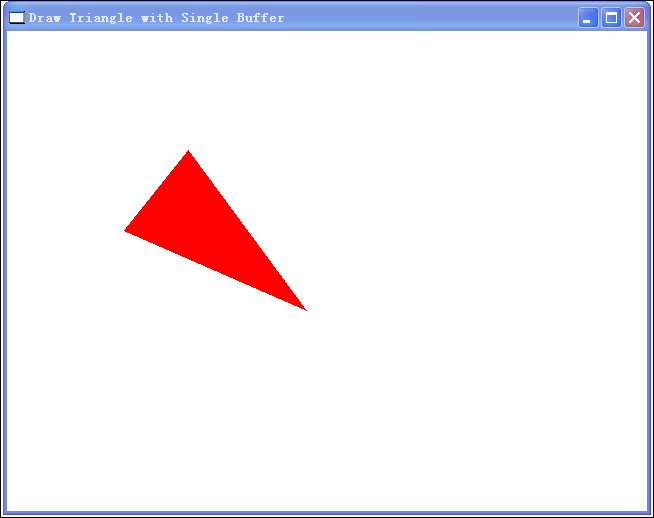
init ( );

glutDisplayFunc (display);

glutIdleFunc(idleFunc); //指定空闲回调函数

glutMainLoop ( );

}



该例子中，我们使红色三角形围绕其中的一个顶点旋转，该顶点就是（300，200），三角形的另外两个顶点由两个参数length和theta确定，length是这两个顶点离固定顶点（300，200）的距离，这两个顶点与（300，200）连线的夹角分别为theta和theta+30度（相对于X轴正方向），所以，当改变theta的值时，三角形的位置会围绕（300，200）旋转。

该程序中改变theta的值是在函数idleFunc中，每次增加0.1度，当大于360度时，又使之变为0度，始终使theta在）0～360度之间循环变化。该函数我们用glutIdleFunc(idleFunc)指定为空闲回调函数，当事件队列中没有事件需要处理时，该空闲回调函数得到执行。

执行后，会看到红色的三角形在旋转中有闪烁的现象，这是由于计算机的显示屏工作时都以固定的频率进行刷新，刷新时从显示卡的内存里取颜色缓存的内容，表现在显示屏上。对于应用程序来说，屏幕的刷新无法控制。该程序绘制时，实际上是对显示卡内存里的颜色缓存进行操作，所以程序对颜色缓存的写和显示屏刷新时对颜色缓存的读是不同步的，所以在某个时刻，颜色缓存的所有颜色值可能不是针对某个确定的theta值，所以三角形看起来不完整。

虽然应用程序无法将两个读写过程同步，但借助于双缓存技术，可以保证绘制一个完整的红色三角形。双缓存技术使用两个颜色缓存――前台缓存和后台缓存，前台缓存用于刷新显示屏，后台缓存用于应用程序的绘制。在绘制完成一个完整的三角形后，我们只要交换前台缓存和后台缓存，就能输出完整的图形。交换前台缓存和后台缓存的函数是void glutSwapBuffers() ,用它取代显示回调函数中的glFlush()即可。

要用双缓存技术，需要在初始化时就指定，把原来的

glutInitDisplayMode (GLUT\_SINGLE | GLUT\_RGB)

改为glutInitDisplayMode (GLUT\_DOUBLE | GLUT\_RGB)。

#include <GL/glut.h>

#include <math.h>

#define DEG\_TO\_RAD 0.017453 //角度转为弧度的参数，即 2\*PI/360

float theta=30.0; //直线与X轴正方向的夹角

float length=200.0; //直线的长度

float x=300.0, y=200.0; //直线的第一个端点

void init (void)

{

glClearColor (1.0, 1.0, 1.0, 0.0);

glMatrixMode (GL\_PROJECTION);

gluOrtho2D (0.0, 640.0, 0.0, 480.0);

}

void display (void)

{

glClear (GL\_COLOR\_BUFFER\_BIT);

glColor3f (1.0, 0.0, 0.0);

glBegin (GL\_POLYGON);

glVertex2f (x, y);

glVertex2f ( x + length\*cos(DEG\_TO\_RAD\*theta),

y + length\*sin(DEG\_TO\_RAD\*theta) );

glVertex2f ( x + length\*cos(DEG\_TO\_RAD\* (theta+30) ),

y + length\*sin(DEG\_TO\_RAD\* (theta+30)) );

glEnd ( );

glutSwapBuffers ( ); //交换前后台缓存

}

void idleFunc()

{

theta += 0.1;

if (theta>360) theta -=360;

glutPostRedisplay(); //重新调用绘制函数

}

void main (int argc, char\*\* argv)

{

glutInit (&argc, argv);

glutInitDisplayMode (GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowPosition (100, 100);

glutInitWindowSize (640, 480);

glutCreateWindow ("Draw Triangle with Double Buffer");

init ( );

glutDisplayFunc (display);

glutIdleFunc(idleFunc); //指定空闲回调函数

glutMainLoop ( );

}

该程序改用双缓存后，避免的闪烁的发生。

**实例2、一个旋转的三角形――用鼠标回调函数实现**

假如要把上例改为用鼠标控制三角形的旋转，例如单击鼠标左键一次，三角形顺时针旋转一定角度，单击鼠标右键一次，三角形逆时针旋转一定角度。这时，要借助鼠标操作回调函数来完成。

例如鼠标回调函数是void myMouse(int button, int state, int x, int y)，那么该鼠标回调函数在主函数中用glutMouseFunc(myMouse)注册该鼠标回调函数。myMouse可写为：

void myMouse(int button, int state, int x, int y)

{

//按下鼠标左键

if(button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)

theta += 5.0;

//按下鼠标右键

if(button == GLUT\_RIGHT\_BUTTON && state == GLUT\_DOWN)

theta -= 5.0;

if (theta>360) theta -=360;

if (theta<0) theta +=360;

glutPostRedisplay(); //重新调用绘制函数

}

**实例3、一个旋转的三角形――用键盘回调函数实现**

假如要把上例改为用键盘控制三角形的旋转，例如单击键盘a键，三角形顺时针旋转一定角度，单击键盘s键，三角形逆时针旋转一定角度，单击键盘c键，程序退出。这时，要借助键盘回调函数来完成。

例如键盘回调函数是void myKeyboard(unsigned char key, int x, int y)，那么该键盘回调函数在主函数中用glutKeyboardFunc( myKeyboard)注册该回调函数。myKeyboard可写为：

void myKeyboard(unsigned char key, int x, int y)

{

if(key == 'a' || key == 'A')

theta += 5.0;

if(key == 's' || key == 'S')

theta -= 5.0;

if(key == 'c' || key == 'C')

exit(0);

if (theta>360) theta -=360;

if (theta<0) theta +=360;

glutPostRedisplay(); //重新调用绘制函数

}

exit(0)表示退出程序，返回到操作系统，该函数在stdlib.h 中定义，所以程序开头加上#include <stdlib.h>。

加入要用键盘的一些特殊键来控制，例如F1和左方向键控制顺时针旋转，F2和右方向键控制逆时针旋转，则该代码就必须放到用glutSpecialFunc()注册的一个特殊键回调函数中。例如：

void mySpecialKeyboard(int key, int x, int y)

{

if(key == GLUT\_KEY\_F1 || key == GLUT\_KEY\_LEFT)

theta += 5.0;

if(key == GLUT\_KEY\_F2 || key == GLUT\_KEY\_RIGHT)

theta -= 5.0;

if (theta>360) theta -=360;

if (theta<0) theta +=360;

glutPostRedisplay(); //重新调用绘制函数

}

该函数在主函数中注册：

glutSpecialFunc( mySpecialKeyboard);

## 编程练习

1、编制一个OpenGL的程序，生成一个600\*600大小的窗口，上面绘制一个200\*300大小的矩形，用键盘的四个方向键控制矩形朝上下左右进行移动，按c键退出程序。

2、编制一个OpenGL的程序，生成一个600\*600大小的窗口，上面用鼠标画线。点击左键生成第一点，再点击左键生成第二点，用这两点画一条直线，然后再点击左键两次，再画另外一条直线。点击右键把屏幕清空。

**三 关键步骤的源码和注释**

1.

void display (void)

{

glClear (GL\_COLOR\_BUFFER\_BIT);

glColor3f (1.0, 0.0, 0.0);

glBegin (GL\_POLYGON);

glVertex2f (xx, yy);

glVertex2f (xx+length, yy);

glVertex2f (xx+length, yy+width);

glVertex2f (xx, yy+width);

glEnd ( );

glFlush ( );

}

void myKeyboard(unsigned char key, int x, int y)

{

if(key == 'a' || key == 'A')

xx -= 10.0;

if(key == 's' || key == 'S')

yy -= 10.0;

if(key == 'w' || key == 'W')

yy+=10.0;

if(key == 'd' || key == 'D')

xx+=10.0;

if(key == 'c' || key == 'C')

exit(0);

glutPostRedisplay(); //重新调用绘制函数

}

void mySpecialKeyboard(int key, int x, int y)

{

if(key == GLUT\_KEY\_UP) yy+=10.0;//上

if(key == GLUT\_KEY\_DOWN) yy-=10.0;//下

if(key == GLUT\_KEY\_LEFT) xx-=10.0;//左

if(key == GLUT\_KEY\_RIGHT) xx+=10.0;//右

glutPostRedisplay(); //重新调用绘制函数

}

int main (int argc, char\*\* argv)

{

glutInit (&argc, argv);

glutInitDisplayMode (GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowPosition (50, 100);

glutInitWindowSize (600, 600);

glutCreateWindow ("移动矩形");

init ( );

glutDisplayFunc(display);

glutKeyboardFunc(myKeyboard);

glutSpecialFunc(mySpecialKeyboard);

glutMainLoop ( );

return 0;

}

2. #include <GL/glut.h>

#include <math.h>

#include<windows.h>

#include<stdio.h>

float width=300.0; //宽

float length=200.0; //长

int xx[2], yy[2]; //点

int cnt=0;

void init (void)

{

glClearColor (1.0, 1.0, 1.0, 0.0);

glMatrixMode (GL\_PROJECTION);

gluOrtho2D (0.0, 600.0, 0.0, 600.0);

glClear (GL\_COLOR\_BUFFER\_BIT);

}

void display (void)

{

//glClear (GL\_COLOR\_BUFFER\_BIT);

glColor3f (1.0, 0.0, 0.0);

if(cnt==2){

glBegin (GL\_LINES);

glVertex2f (xx[0], yy[0]);

glVertex2f (xx[1], yy[1]);

glEnd ( );

cnt=0;

}else{

glBegin(GL\_POINT);

glEnd();

}

glFlush();

}

void myMouse(int button, int state, int x, int y)

{

//按下鼠标左键

if(button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN){

xx[cnt]=x,yy[cnt]=600-y;

cnt++;

}

//按下鼠标右键

if(button == GLUT\_RIGHT\_BUTTON && state == GLUT\_DOWN){

glClearColor (1.0, 1.0, 1.0, 0.0);

glClear (GL\_COLOR\_BUFFER\_BIT);

cnt=0;

// glFlush ( );

}

glutPostRedisplay(); //重新调用绘制函数

}

int main (int argc, char\*\* argv)

{

glutInit (&argc, argv);

glutInitDisplayMode (GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowPosition (50, 100);

glutInitWindowSize (600, 600);

glutCreateWindow ("画线");

init ( );

glutDisplayFunc(display);

glutMouseFunc(myMouse);

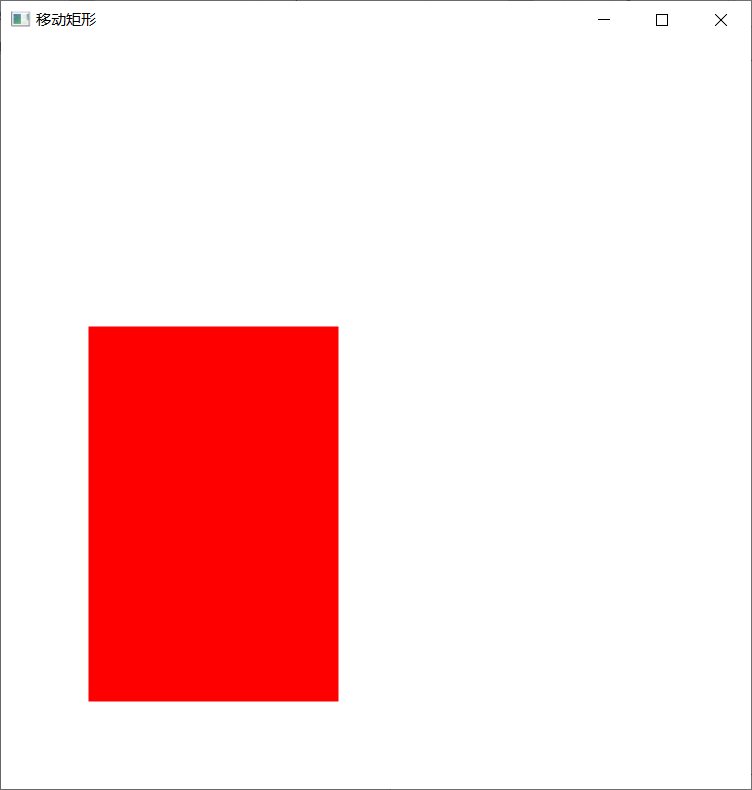
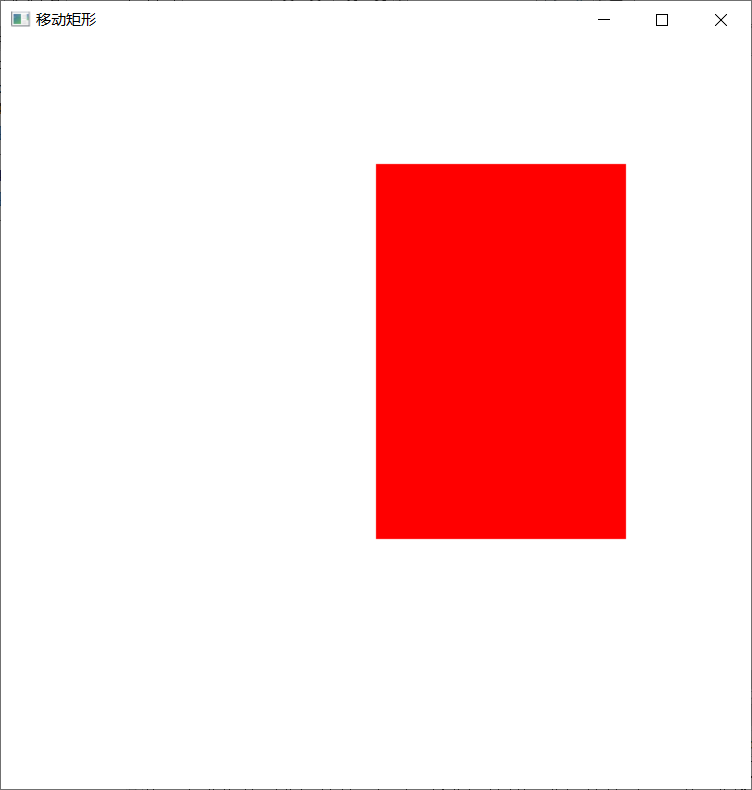
glutMainLoop ( );

return 0;

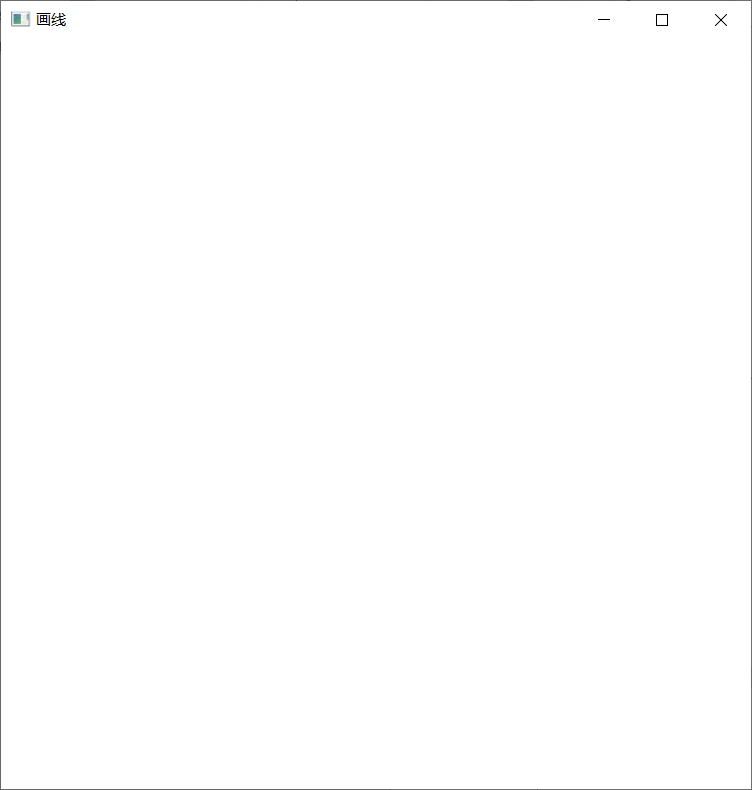
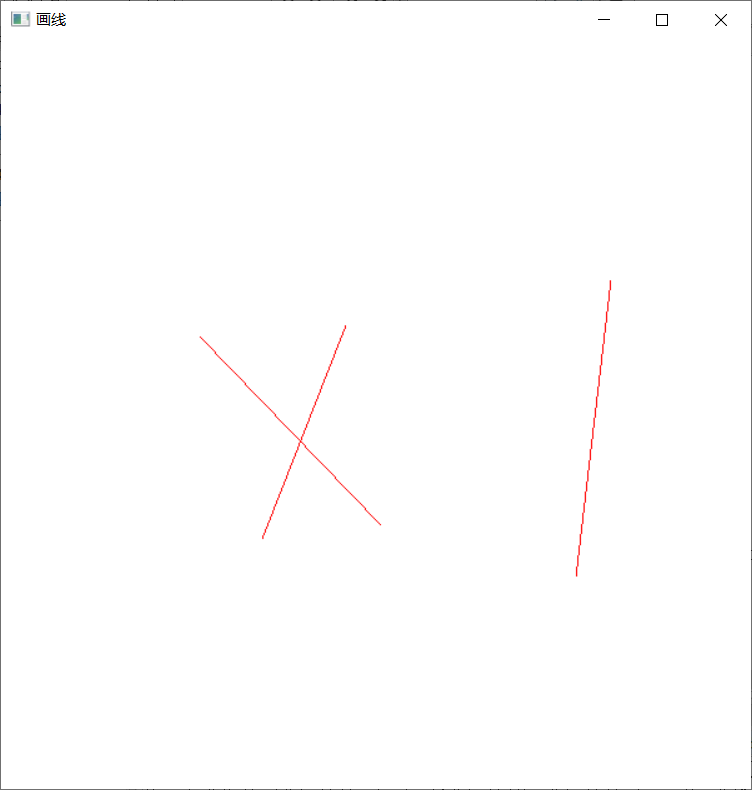
}

四 **实验结果（截图）**

1. **初始 按键移动**



**2. 左键画线 右键清空**



**五、实验分析**

在实现右键清空时发现，右键不能立即清空，后来查明是因为OpenGL的glClear函数需要执行一次画图后才能发挥作用，因此对于右键清空的功能，可以建一个空的glBegin(GL\_POINT);glEnd();来使glClear生效。另外，对于特殊键的捕获也不能与普通键放在一起，特殊键的编码会与普通键的编码冲突