实验4 交互与动画II

【实验目的】

1.掌握基本交互式程序的编程方法。

2.掌握基本动画程序的编程方法。

【实验原理】

介绍交互与动画相关的新的OpenGL函数（参考PPT和课本等资料）：

如窗口改变回调函数、重绘回调函数、单双缓存技术等。

【实验内容】

1.将正方形旋转的程序squareRotate.c改成正六边形旋转的程序。

将display函数中的顶点确定段改成下面这样：

|  |
| --- |
| glVertex2f(cos(theta \* DEGREES\_TO\_RADIANS),                 sin(theta \* DEGREES\_TO\_RADIANS))  glVertex2f(cos(pi / 3 + theta \* DEGREES\_TO\_RADIANS),                 sin(pi / 3 + theta \* DEGREES\_TO\_RADIANS))  glVertex2f(cos(2 \* pi / 3 + theta \* DEGREES\_TO\_RADIANS),                 sin(2 \* pi / 3 + theta \* DEGREES\_TO\_RADIANS))  glVertex2f(cos(2 \* pi / 3 + theta \* DEGREES\_TO\_RADIANS),                 sin(2 \* pi / 3 + theta \* DEGREES\_TO\_RADIANS))  glVertex2f(-cos(theta \* DEGREES\_TO\_RADIANS),                 -sin(theta \* DEGREES\_TO\_RADIANS))  glVertex2f(cos(-2 \* pi / 3 + theta \* DEGREES\_TO\_RADIANS),                 sin(-2 \* pi / 3 + theta \* DEGREES\_TO\_RADIANS))  glVertex2f(cos( -1 \* pi / 3 + theta \* DEGREES\_TO\_RADIANS),                 sin( -1 \* pi / 3 + theta \* DEGREES\_TO\_RADIANS)) |

即可实现正六边形的旋转

2.创建一个绘图程序，使得可用鼠标来创建一些简单的形状，如线段，三角形，矩形，并可通过菜单来实现下列功能。要求：

（1）可改变形状的颜色。

（2）可改变形状的大小。

（3）可移动形状。

（4）可旋转形状。

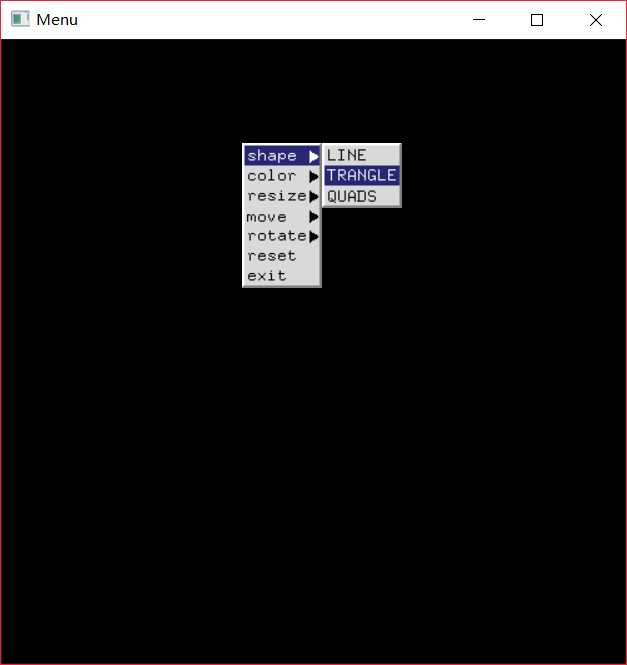
（5）你能想到的任何功能。

算法概括：

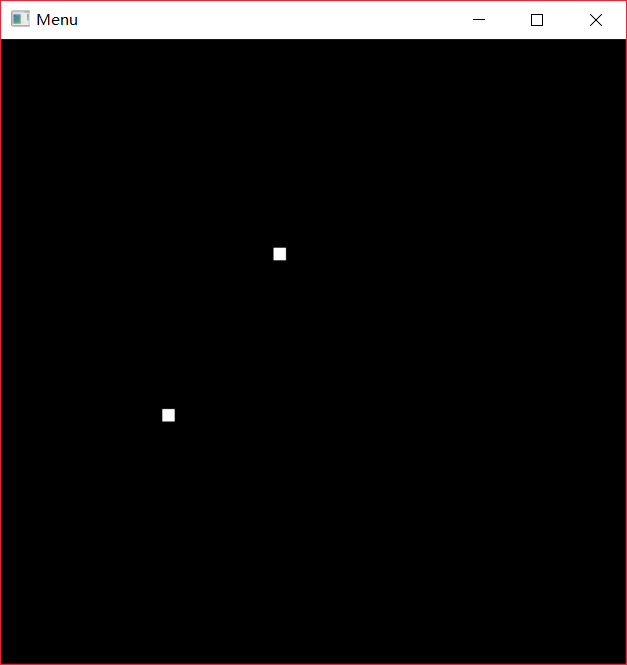
使用全局变量shape\_mode记录选择绘制的图形，全局变量vexs保存顶点信息，以完成旋转、平移后的重绘操作。

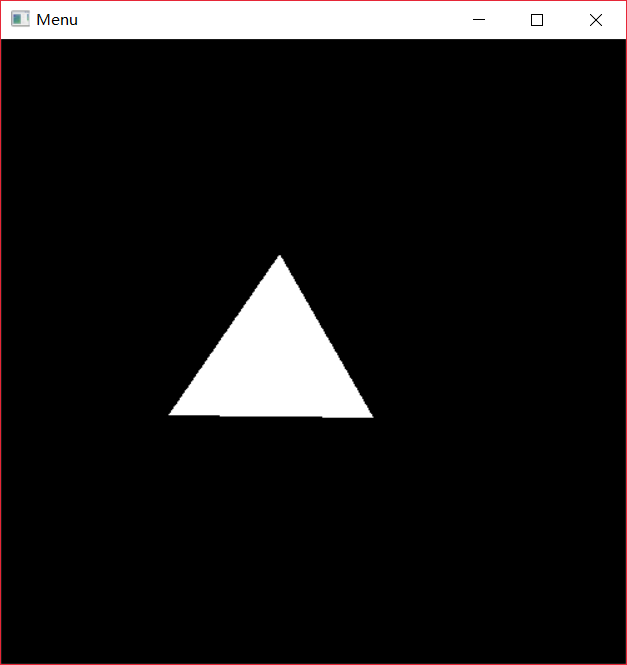
演示：

1、生成图形：

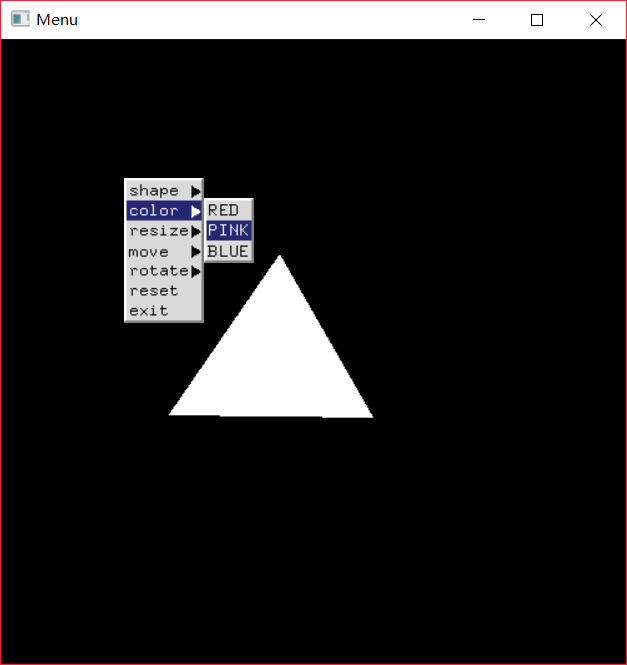


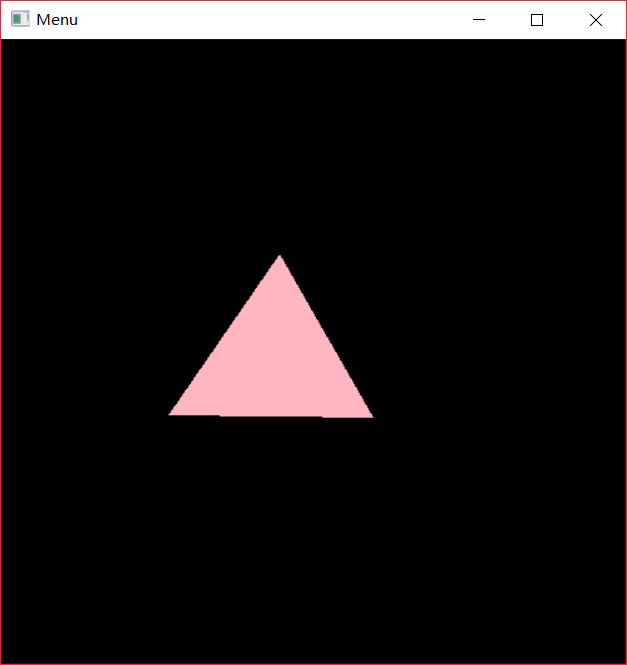
以三角形为例，使用鼠标依次确定三个顶点：



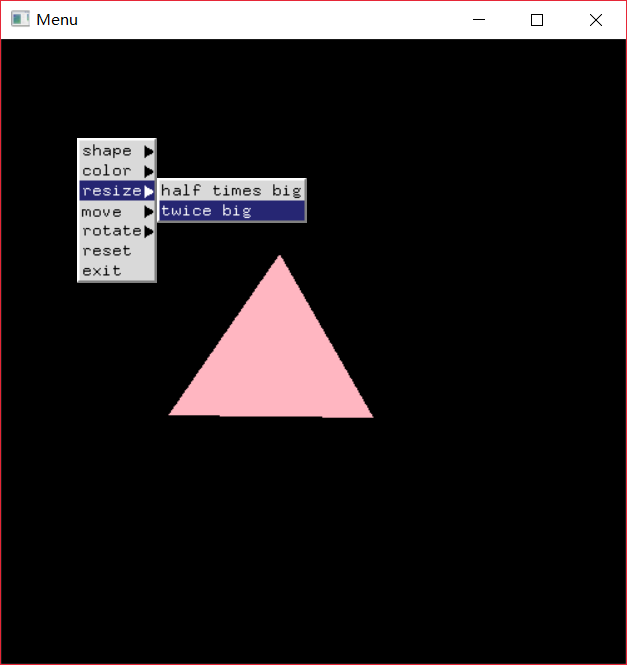


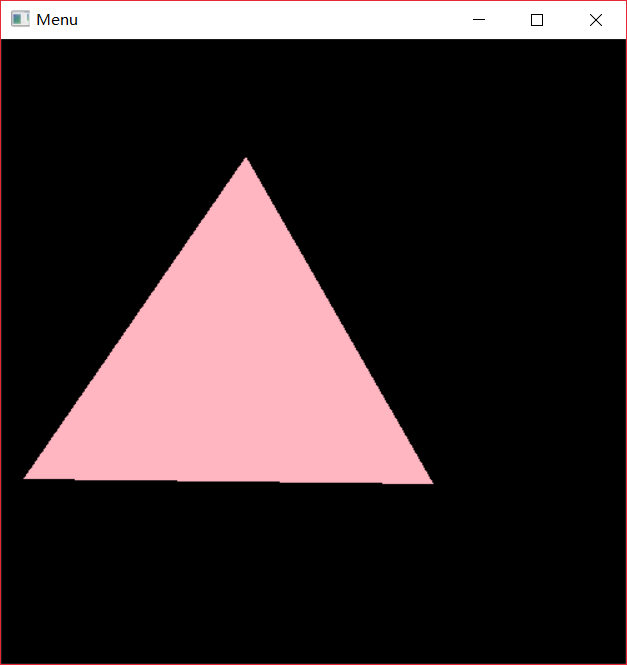
2、更改颜色（以粉色为例）：



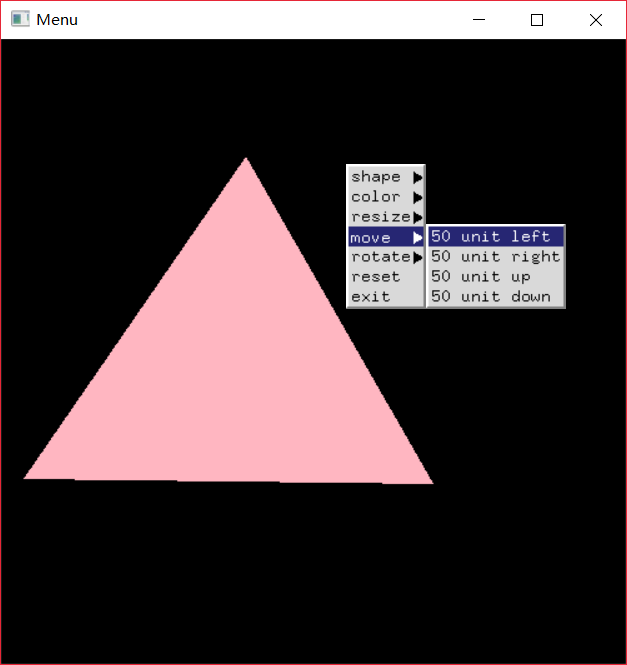


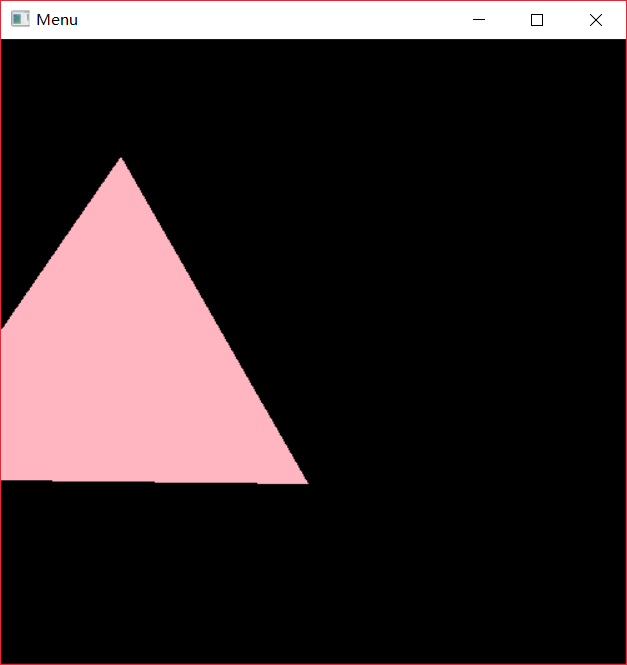
3、缩放（以放大两倍为例）：

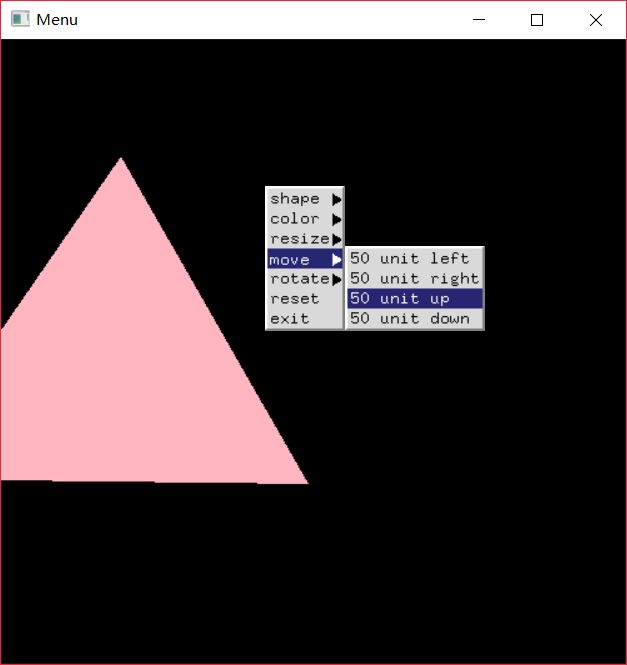


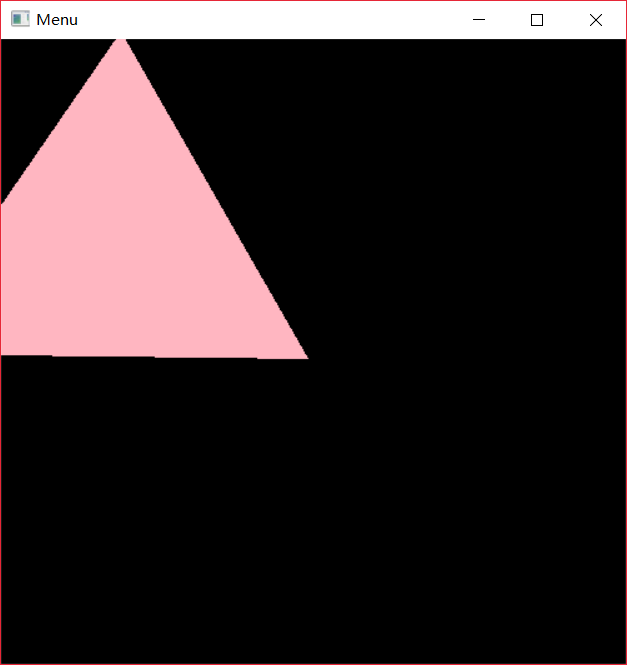


4、平移（以向左平移50个单位，再向上平移50单位为例）：

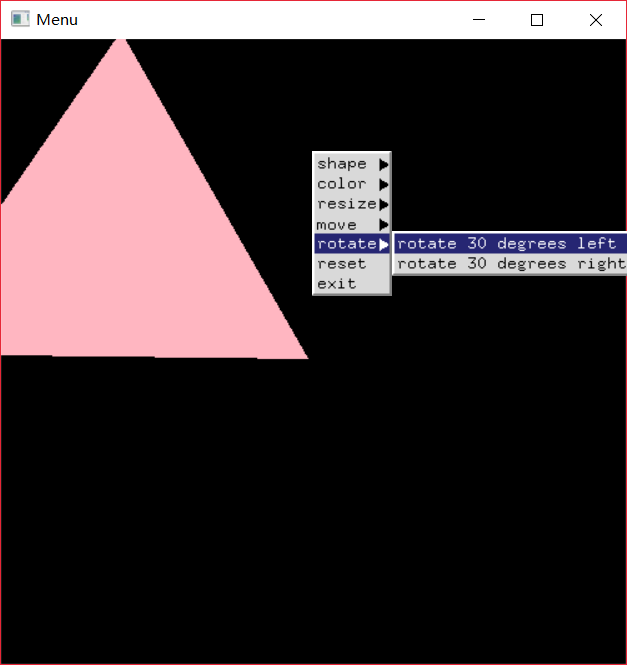


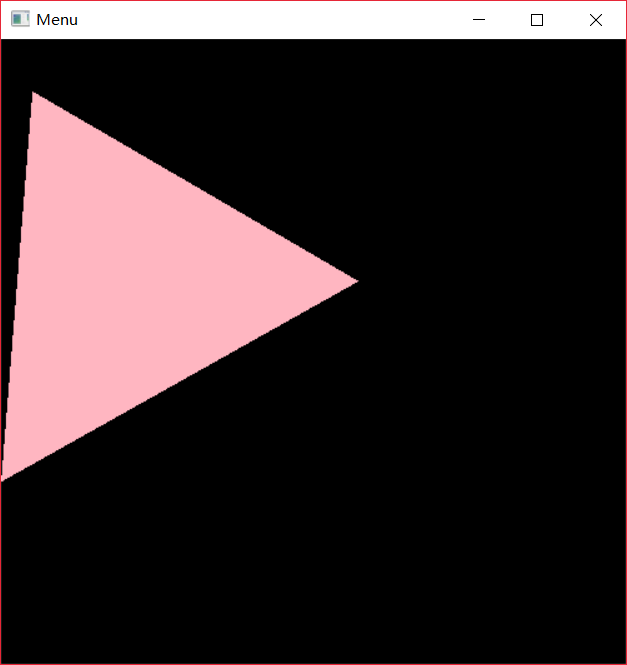


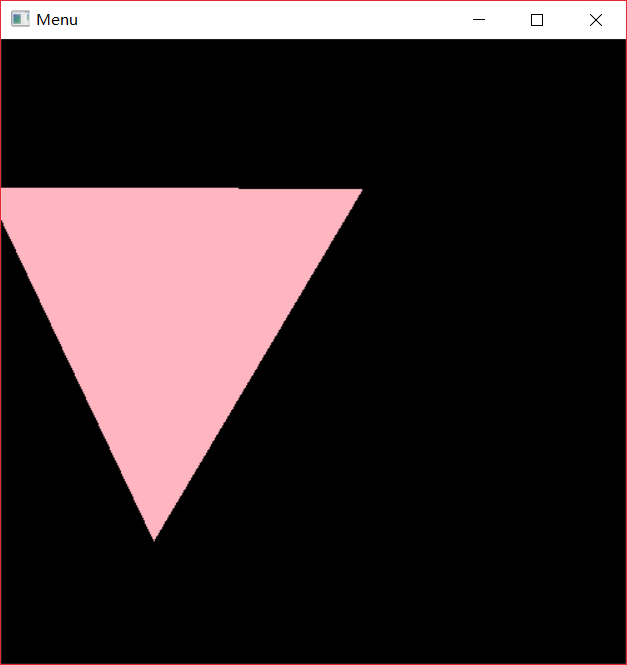




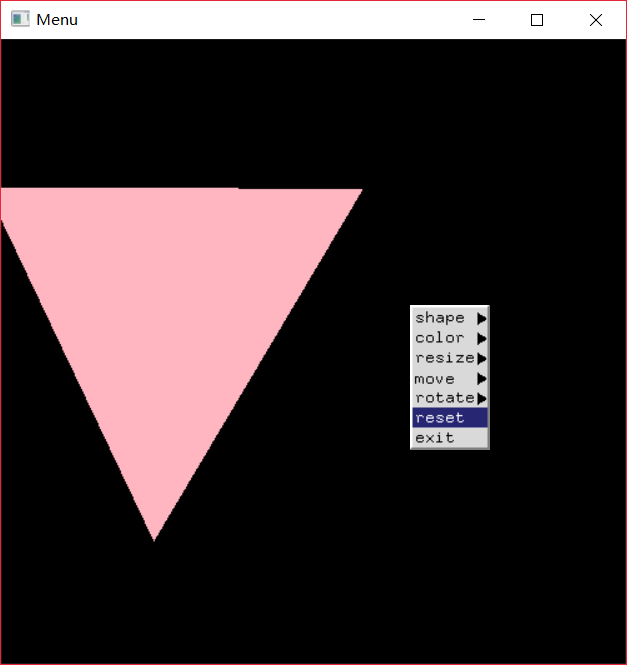
5、旋转（以连续两次向左旋转30度为例）：

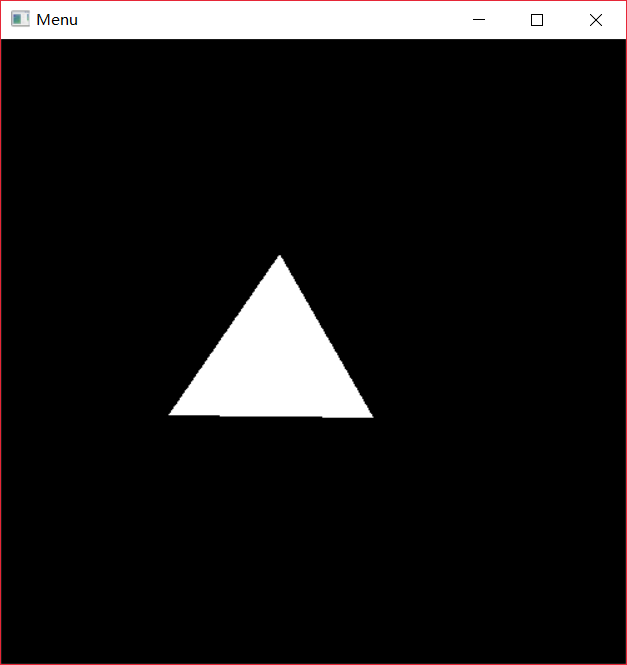




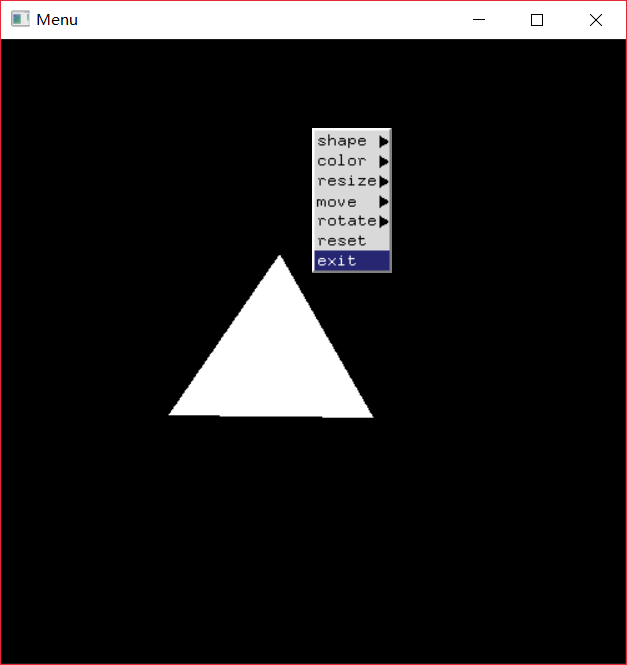


6、图形复位，恢复初始大小、位置和颜色：





7、终止程序：



选择该选项后程序将终止运行

其他效果（比如生成矩形和线段、缩小1/2、向右平移等）没有在此展示，但经测试，均可正常运行。

附：Python版源代码：

*from* OpenGL.GL *import* \*

*from* OpenGL.GLUT *import* \*

*# 全局变量记录形状选择*

shape\_mode = 0

*# status为0时不绘图，为1时绘图*

status = -1

vexs = []

def display():

    glClearColor(0.0, 0.0, 0.0, 1)

    glMatrixMode(GL\_PROJECTION)

    glLoadIdentity()

    glOrtho(-250, 250, -250, 250, -1, 1)

def process\_menu\_events(value):

    ''' 处理部分菜单命令 '''

    global color\_mode

*if* value == 0:

        exit()

*if* value == 1:

        glLoadIdentity()

        glOrtho(-250, 250, -250, 250, -1, 1)

        glColor3f(1.0, 1.0, 1.0)

        draw\_figure(vexs)

def draw\_point(x, y):

    ''' 绘制一个点 '''

    glPointSize(10)

    glBegin(GL\_POINTS)

    glVertex2f(x, y)

    glEnd()

    glFlush()

def draw\_figure(vexs):

    ''' 根据vexs列表和shape\_mode绘制图形 '''

    glClear(GL\_COLOR\_BUFFER\_BIT)

*if* shape\_mode == 1:

        glLineWidth(5)

        glBegin(GL\_LINES)

        glVertex2f(vexs[0]['x'], vexs[0]['y'])

        glVertex2f(vexs[1]['x'], vexs[1]['y'])

*elif* shape\_mode == 2:

        glBegin(GL\_TRIANGLES)

        glVertex2f(vexs[0]['x'], vexs[0]['y'])

        glVertex2f(vexs[1]['x'], vexs[1]['y'])

        glVertex2f(vexs[2]['x'], vexs[2]['y'])

*elif* shape\_mode == 3:

        glBegin(GL\_QUADS)

        glVertex2f(vexs[0]['x'], vexs[0]['y'])

        glVertex2f(vexs[1]['x'], vexs[0]['y'])

        glVertex2f(vexs[1]['x'], vexs[1]['y'])

        glVertex2f(vexs[0]['x'], vexs[1]['y'])

    glEnd()

    glFlush()

def mouse\_process(btn, state, x, y):

    ''' 根据鼠标左键确定图形的顶点 '''

    global vexs

    global status

    x = x - 250

    y = 250 - y

*if* btn == GLUT\_LEFT\_BUTTON and state == GLUT\_UP and status:

        draw\_point(x, y)

        vex = {}

        vex['x'] = x

        vex['y'] = y

        vexs.append(vex)

*if* len(vexs) == 2 and (shape\_mode == 1 or shape\_mode == 3):

            draw\_figure(vexs)

            status = 0

*elif* len(vexs) == 3 and shape\_mode == 2:

            draw\_figure(vexs)

            status = 0

def choose\_shape\_mode(value):

    ''' 选择绘图形状 '''

    global shape\_mode

    shape\_mode = value

    global status

    status = 1

    vexs.clear()

    glutMouseFunc(mouse\_process)

def figure\_color\_change(value):

    ''' 选择颜色 '''

    global vexs

*if* value == 1:

        glColor3ub(255, 48, 48)

*elif* value == 2:

        glColor3ub(255, 182, 193)

*elif* value == 3:

        glColor3ub(0, 191, 255)

    draw\_figure(vexs)

def figure\_resize(value):

    ''' 改变图形大小 '''

*if* value == 1:

*# 面积缩小为1/2*

        glScaled(0.5, 0.5, 0.0)

        draw\_figure(vexs)

*if* value == 2:

*# 面积扩大为2倍*

        glScaled(2, 2, 0)

        draw\_figure(vexs)

def figure\_move(value):

    ''' 平移图形 '''

*if* value == 1:

*# 向左平移0,2单位*

        glTranslatef(-50, 0.0, 1.0)

        draw\_figure(vexs)

*if* value == 2:

*# 向右平移50单位*

        glTranslatef(50, 0.0, 1.0)

        draw\_figure(vexs)

*if* value == 3:

*# 向上平移50单位*

        glTranslatef(0.0, 50, 1.0)

        draw\_figure(vexs)

*if* value == 4:

*# 向下平移50单位*

        glTranslatef(0.0, -50, 1.0)

        draw\_figure(vexs)

def figure\_rotate(value):

    ''' 旋转图形 '''

*if* value == 1:

*# 向左旋转30度*

        glRotated(30, 0, 0, 1)

        draw\_figure(vexs)

*if* value == 2:

*# 向右旋转30度*

        glRotated(-30, 0, 0, 1)

        draw\_figure(vexs)

def creat\_menu():

    ''' 创建菜单 '''

    glClear(GL\_COLOR\_BUFFER\_BIT)

*# 子菜单：选择绘图形状*

    shape\_menu = glutCreateMenu(choose\_shape\_mode)

    glutAddMenuEntry('LINE', 1)

    glutAddMenuEntry('TRANGLE', 2)

    glutAddMenuEntry('QUADS', 3)

*# 子菜单：选择颜色*

    color\_menu = glutCreateMenu(figure\_color\_change)

    glutAddMenuEntry('RED', 1)

    glutAddMenuEntry('PINK', 2)

    glutAddMenuEntry('BLUE', 3)

*# 子菜单：缩放*

    resize\_menu = glutCreateMenu(figure\_resize)

    glutAddMenuEntry('half times big', 1)

    glutAddMenuEntry('twice big', 2)

*# 子菜单：平移*

    move\_menu = glutCreateMenu(figure\_move)

    glutAddMenuEntry('50 unit left', 1)

    glutAddMenuEntry('50 unit right', 2)

    glutAddMenuEntry('50 unit up', 3)

    glutAddMenuEntry('50 unit down', 4)

*# 子菜单：旋转*

    rotate\_menu = glutCreateMenu(figure\_rotate)

    glutAddMenuEntry('rotate 30 degrees left', 1)

    glutAddMenuEntry('rotate 30 degrees right', 2)

*# 创建主菜单*

    main\_menu = glutCreateMenu(process\_menu\_events)

*# 将子菜单与主菜单关联*

    glutAddSubMenu('shape', shape\_menu)

    glutAddSubMenu('color', color\_menu)

    glutAddSubMenu('resize', resize\_menu)

    glutAddSubMenu('move', move\_menu)

    glutAddSubMenu('rotate', rotate\_menu)

    glutAddMenuEntry('reset', 1)

    glutAddMenuEntry('exit', 0)

*# 菜单调出绑定到鼠标右键*

    glutAttachMenu(GLUT\_RIGHT\_BUTTON)

def main():

    glutInit()

    glutInitDisplayMode(GLUT\_SINGLE or GLUT\_RGBA)

    glutInitWindowPosition(200, 200)

    glutInitWindowSize(500, 500)

    glutCreateWindow("Menu")

    glutDisplayFunc(display)

    creat\_menu()

    glutMainLoop()

main()

附：squareRotate.c：

/\*

\* double.c

\* This program demonstrates double buffering for

\* flicker-free animation. The left and middle mouse

\* buttons start and stop the spinning motion of the square.

\*/

#include <stdlib.h>

#ifdef \_\_APPLE\_\_

#include <GLUT/glut.h>

#else

#include <GL/glut.h>

#endif

#include <math.h>

#define DEGREES\_TO\_RADIANS 3.14159/180.0

GLfloat theta = 0.0; // 全局变量

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_POLYGON);

glVertex2f(cos(theta\*DEGREES\_TO\_RADIANS),sin(theta\*DEGREES\_TO\_RADIANS));

glVertex2f(-sin(theta\*DEGREES\_TO\_RADIANS),cos(theta\*DEGREES\_TO\_RADIANS));

glVertex2f(-cos(theta\*DEGREES\_TO\_RADIANS),-sin(theta\*DEGREES\_TO\_RADIANS));

glVertex2f(sin(theta\*DEGREES\_TO\_RADIANS),-cos(theta\*DEGREES\_TO\_RADIANS));

glEnd();

glutSwapBuffers ();

}

void idle()

{

theta += 2.0;

if (theta > 360.0) theta -= 360.0;

glutPostRedisplay(); // 请求重绘

}

void myinit ()

{

glClearColor (0.0, 0.0, 0.0, 1.0);

glColor3f (1.0, 1.0, 1.0);

glShadeModel (GL\_FLAT);

}

void mouse(int btn, int state, int x, int y)

{

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

glutIdleFunc(idle);

if(glutGetModifiers() == GLUT\_ACTIVE\_CTRL && btn==GLUT\_LEFT\_BUTTON && state==GLUT\_DOWN)

glutIdleFunc(NULL);

}

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

{

// 按下Q、q，终止程序

if(key == 'Q' || key == 'q') exit(0);

}

void myReshape(int w, int h)

{

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

if (w <= h)

glOrtho (-2.0, 2.0, -2.0\*(GLfloat)h/(GLfloat)w,

2.0\*(GLfloat)h/(GLfloat)w, -1.0, 1.0);

else

glOrtho (-2.0\*(GLfloat)w/(GLfloat)h,

2.0\*(GLfloat)w/(GLfloat)h, -2.0, 2.0, -1.0, 1.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity ();

}

/\* Main Loop

\* Open window with initial window size, title bar,

\* RGBA display mode, and handle input events.

\*/

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

{

glutInit(&argc,argv);

glutInitDisplayMode (GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowPosition(500,0);

glutCreateWindow("double buffered");

myinit ();

glutDisplayFunc(display);

glutReshapeFunc (myReshape);

glutIdleFunc (idle);

glutMouseFunc (mouse);

glutKeyboardFunc(mykey);

glutMainLoop();

}