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

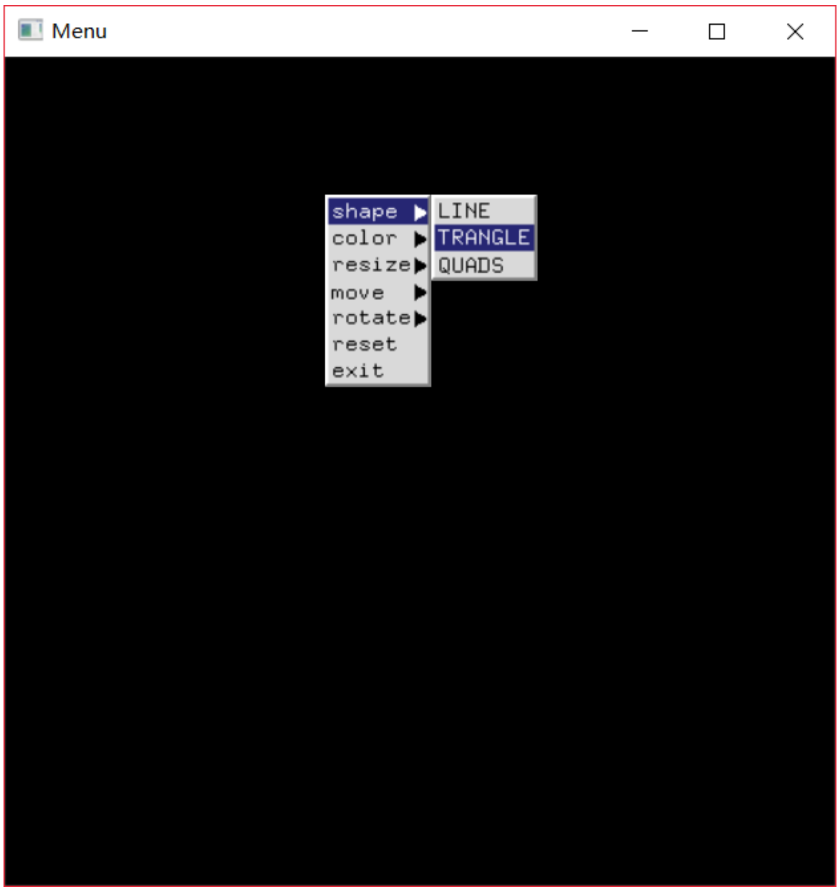
2.创建一个绘图程序，使得可用鼠标来创建一些简单的形状，如线段，三角形，矩形，并可通过菜单来实现下列功能。要求：
（1）可改变形状的颜色。
（2）可改变形状的大小。
（3）可移动形状。
（4）可旋转形状。
（5）你能想到的任何功能。

算法概括：

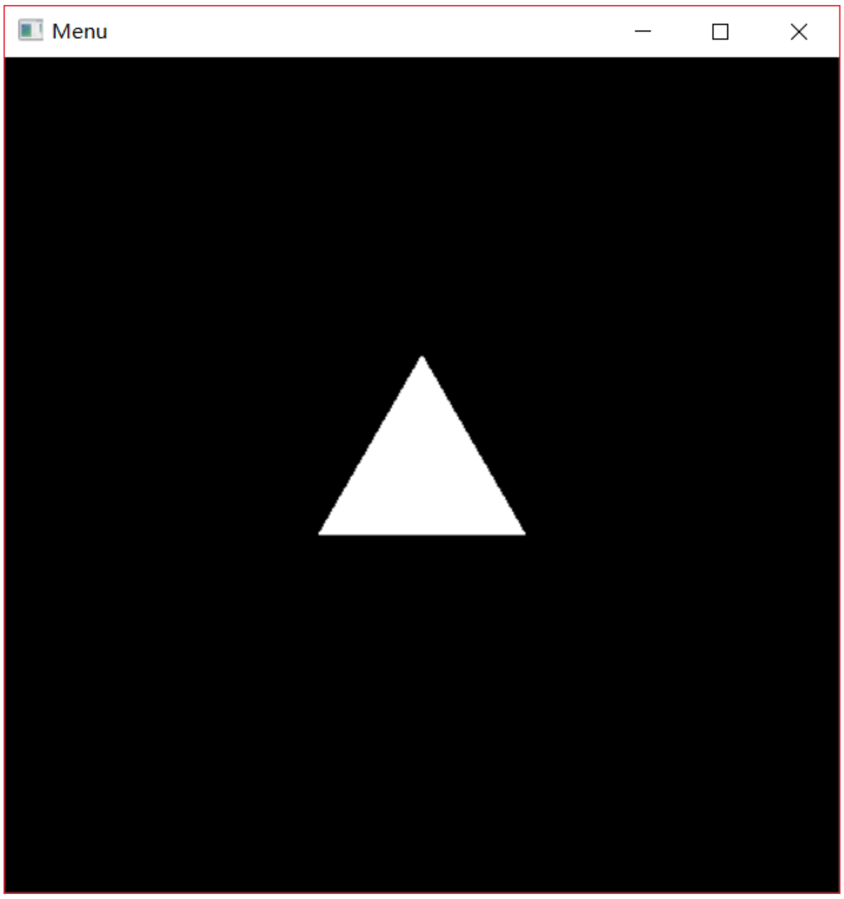
使用全局变量记录选择绘制的图形，以完成旋转、平移等操作。

演示：

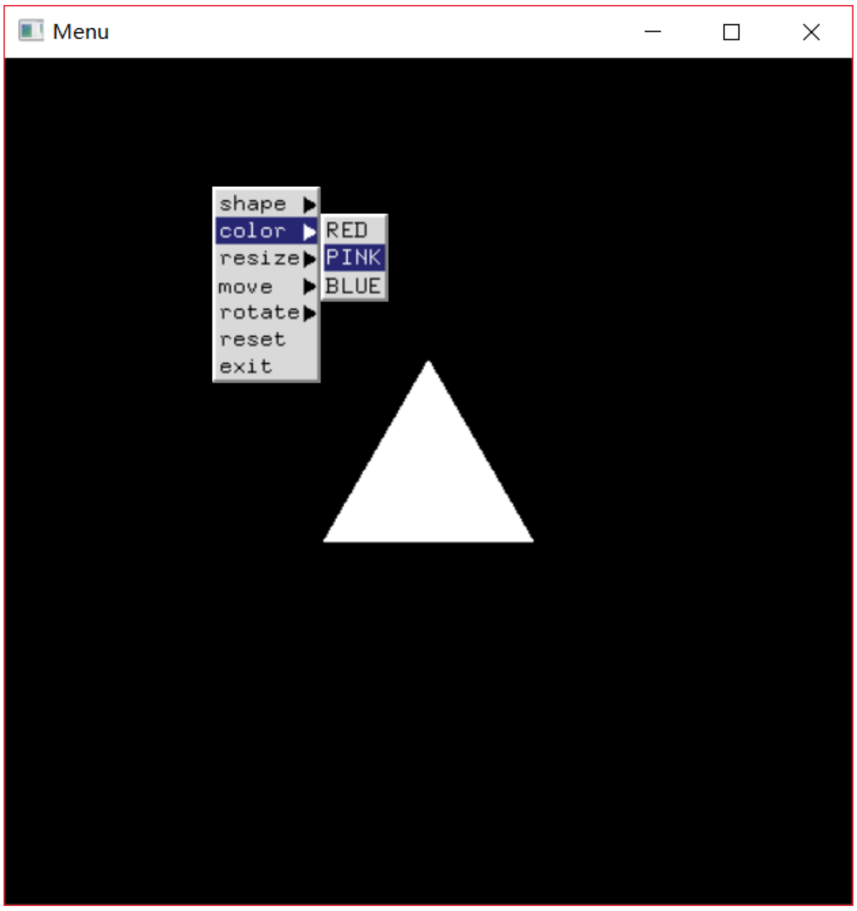
1、生成图形：

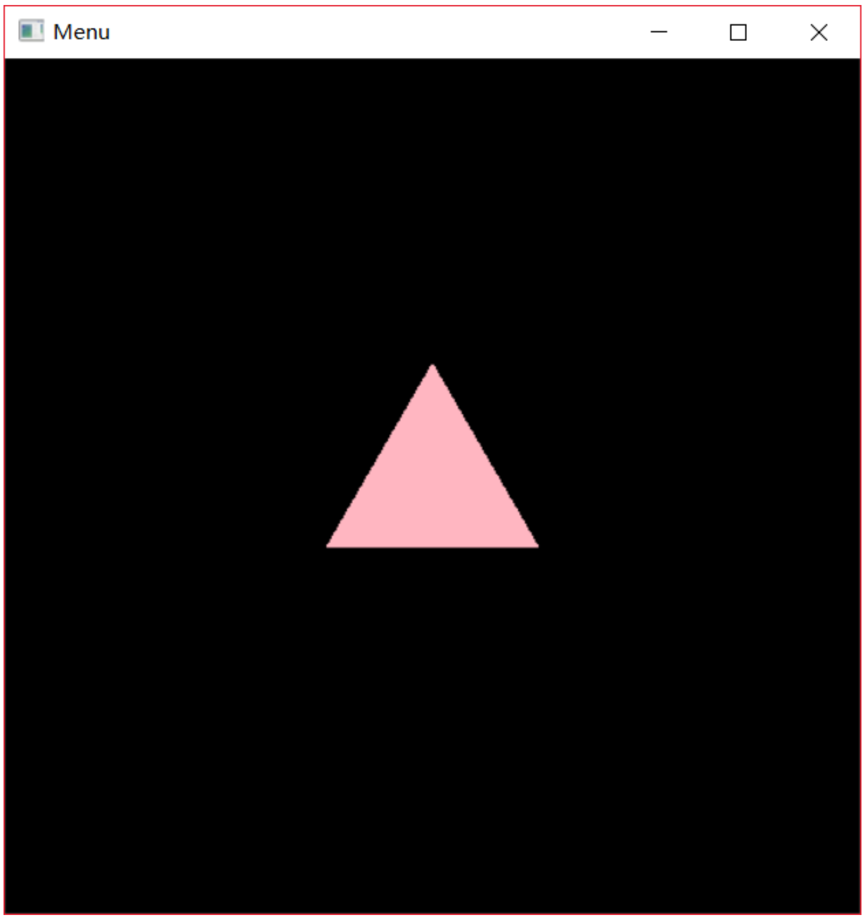


以三角形为例：

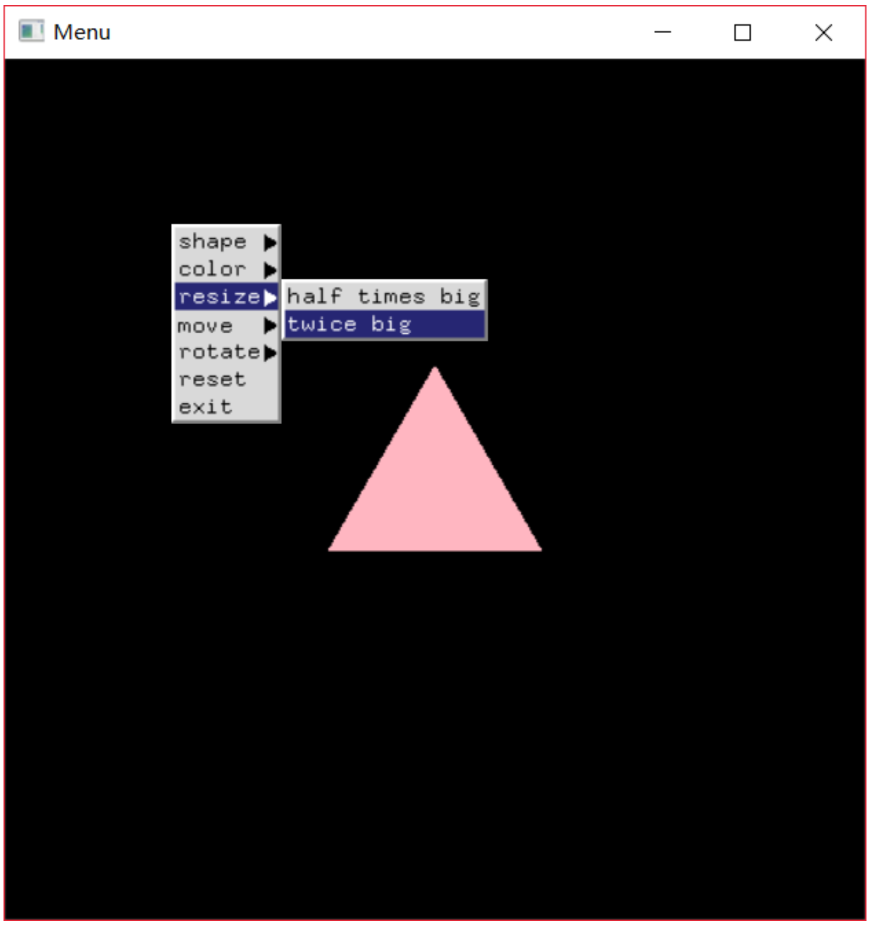


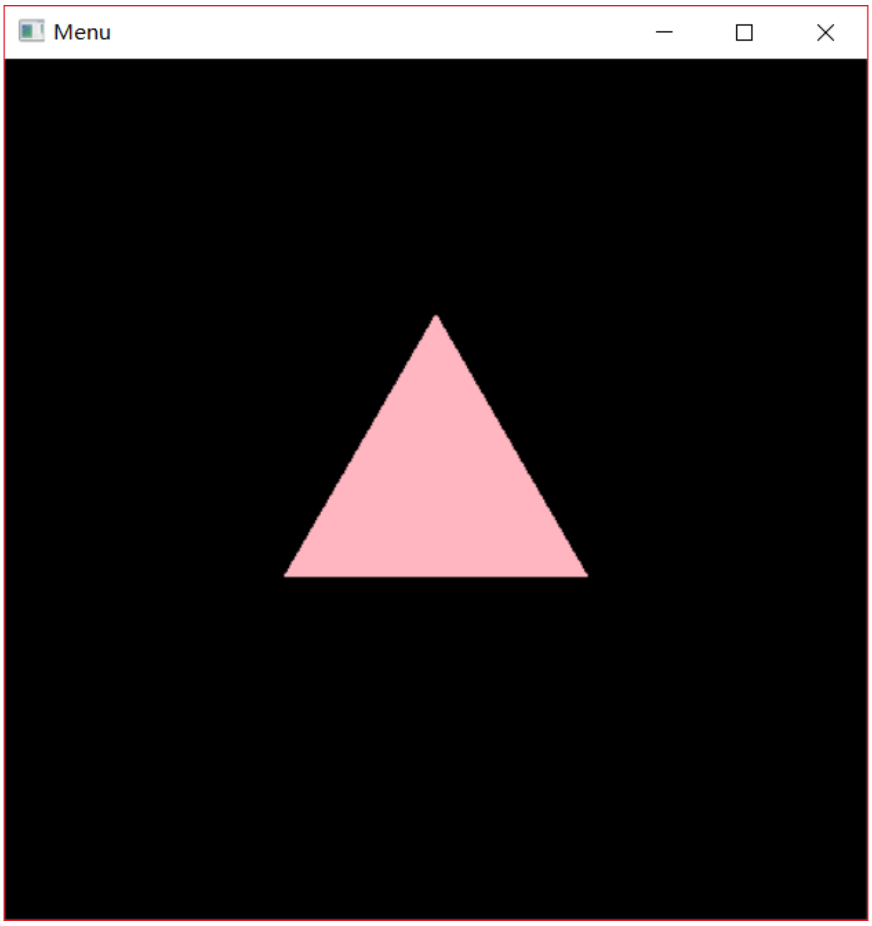
2、更改颜色（以粉色为例）：


Menu
shape
color
resize
move
rotate
reset
exit
RED
PINK
BLUE

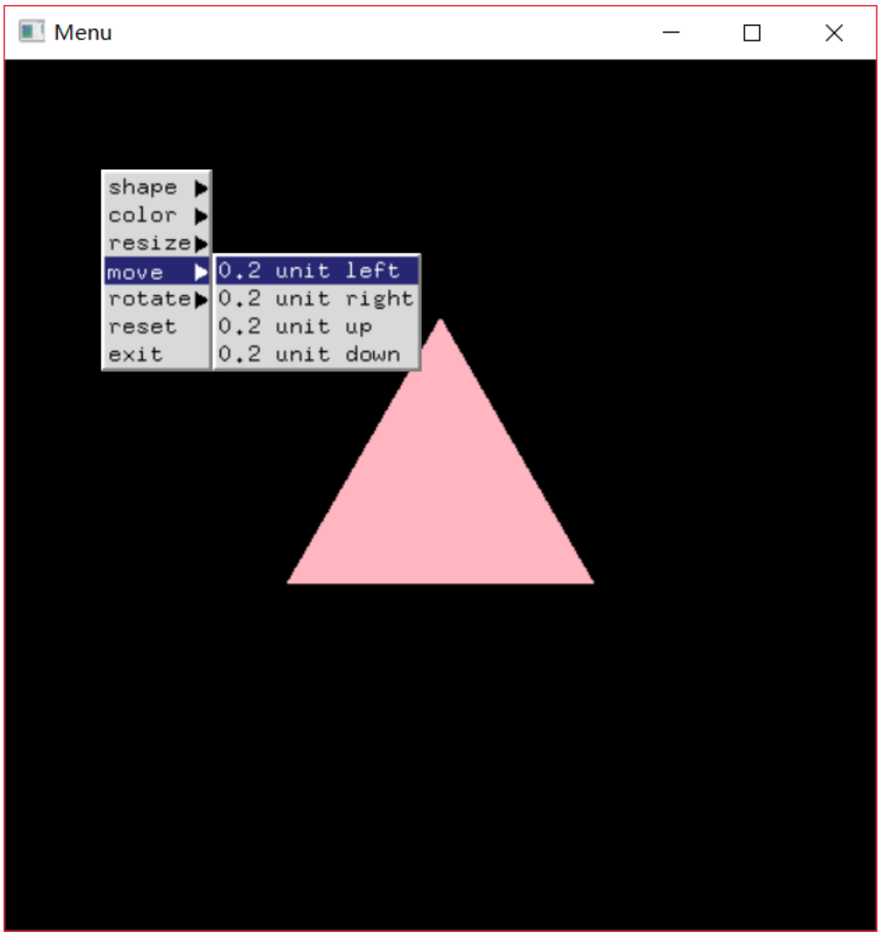

Menu

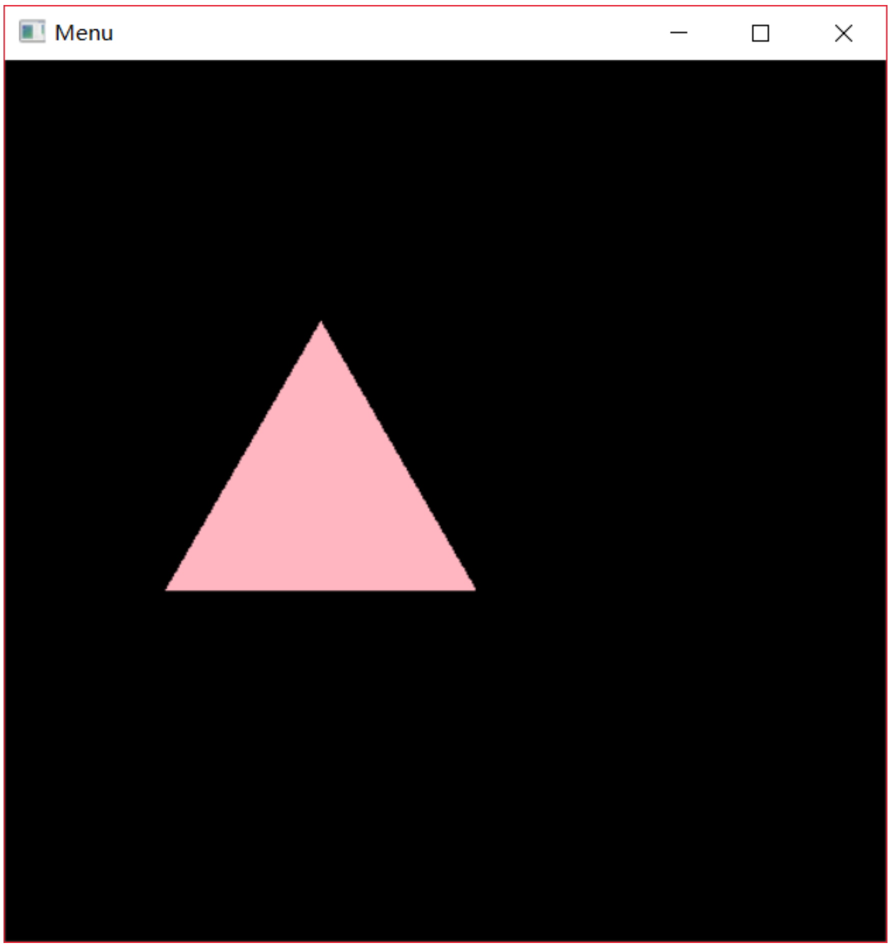
3、缩放（以放大两倍为例）：

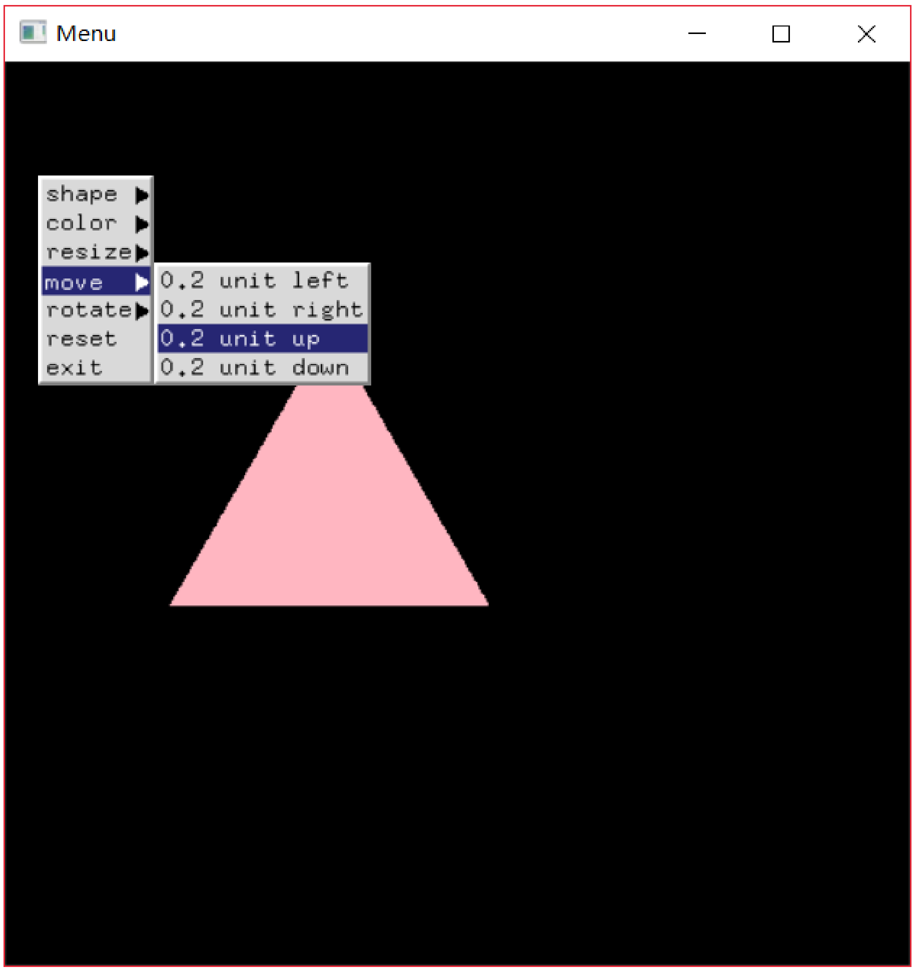

Menu
shape
color
resize
move
rotate
reset
exit
half times big
twice big

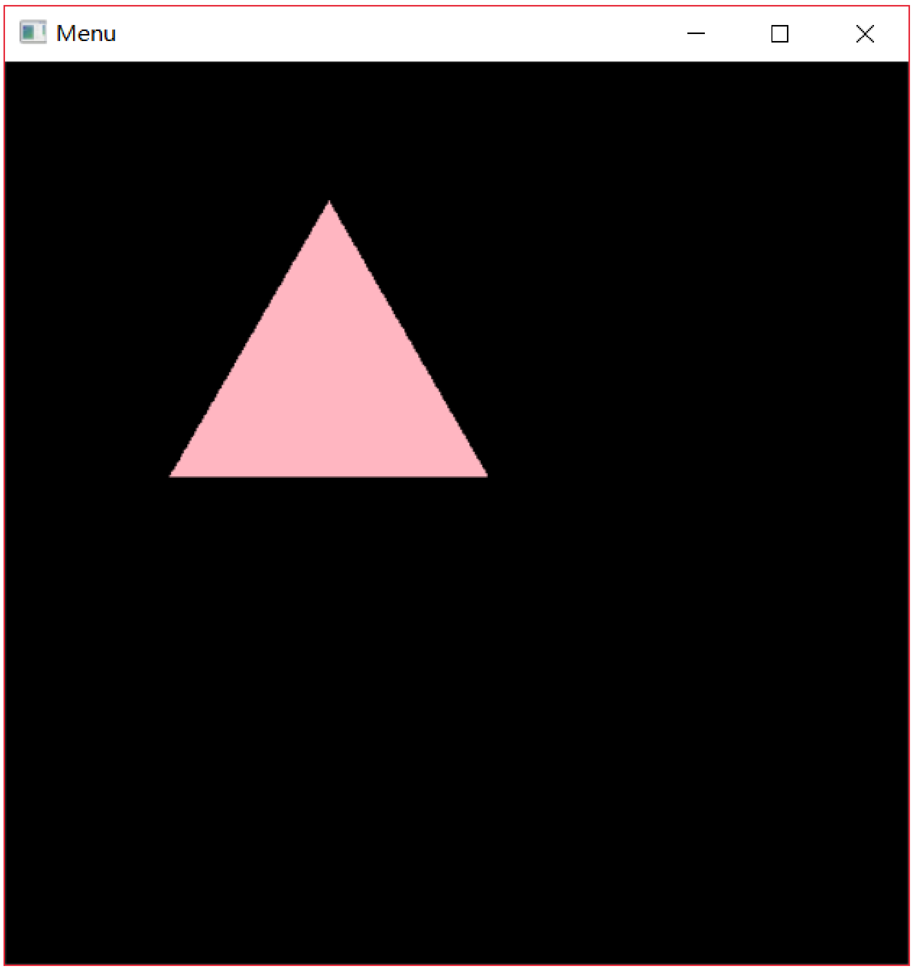

Menu

4、平移（以向左平移0.2个单位，再向上平移0.2单位为例）:

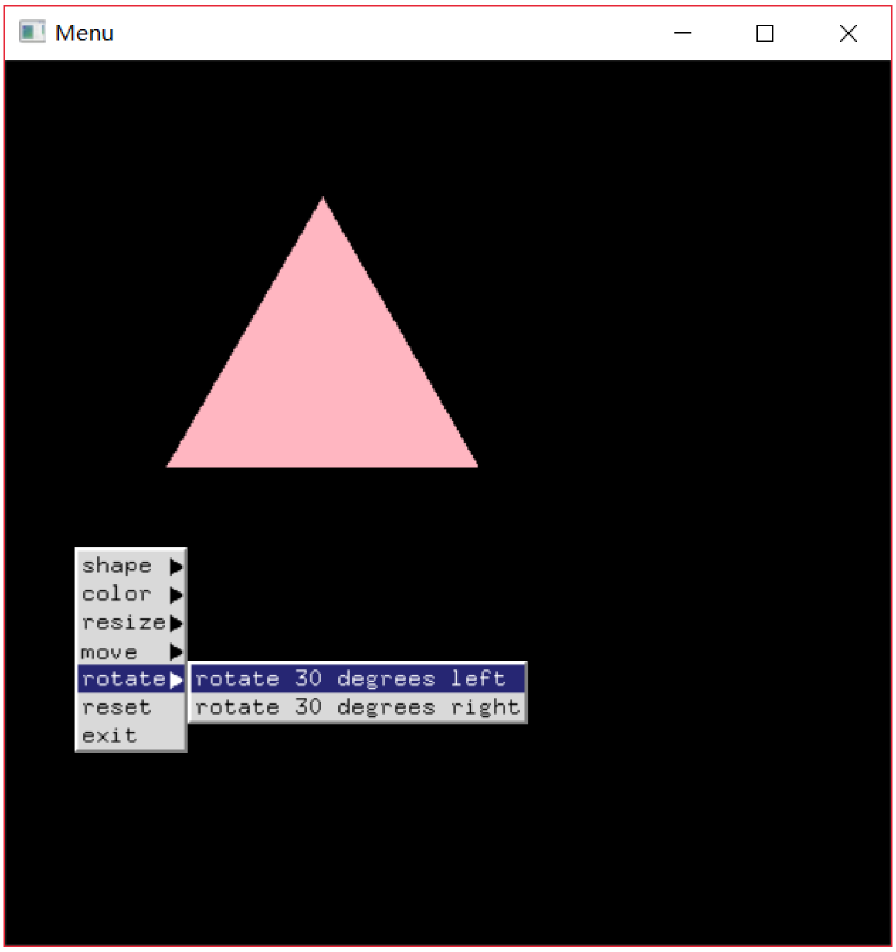


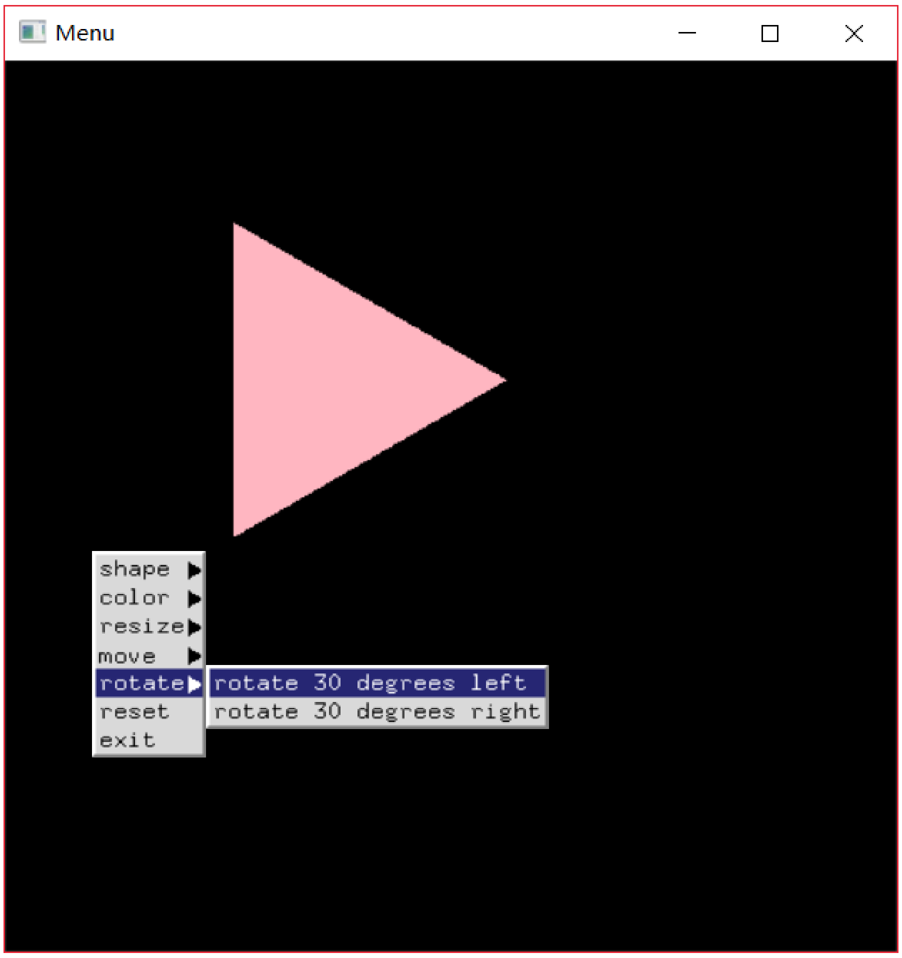


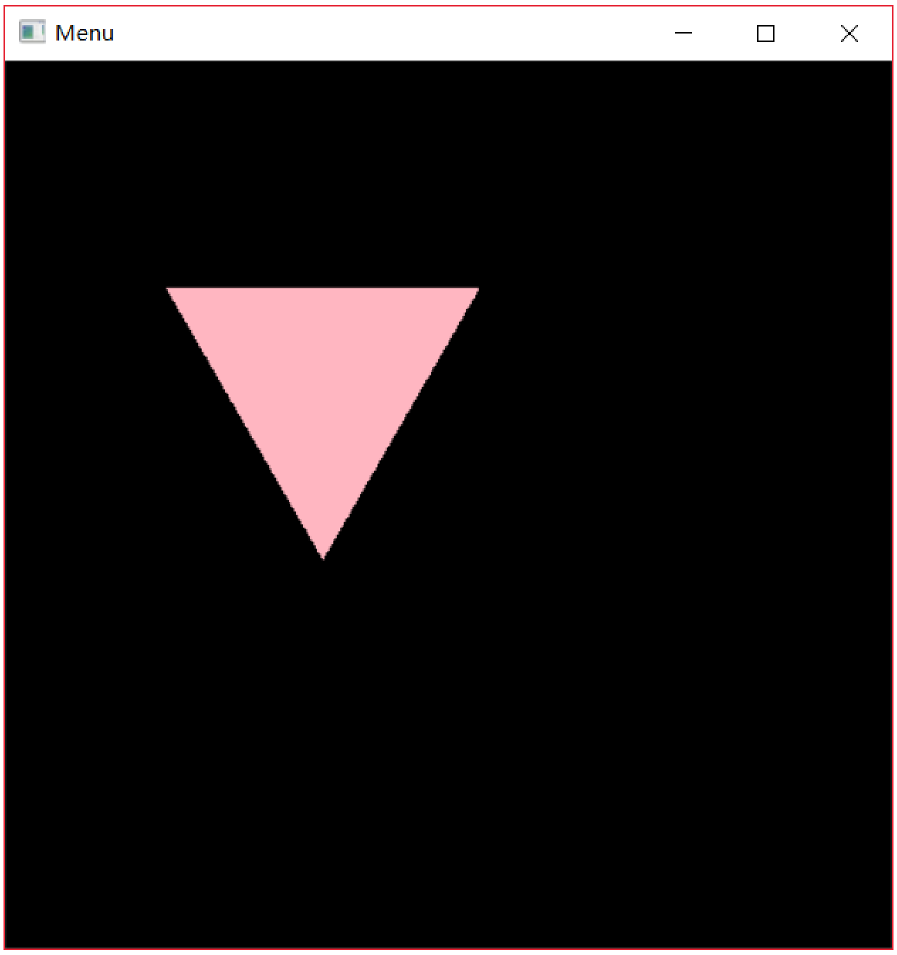

Menu
shape
color
resize
move
rotate
reset
exit
0.2 unit left
0.2 unit right
0.2 unit up
0.2 unit down


Menu

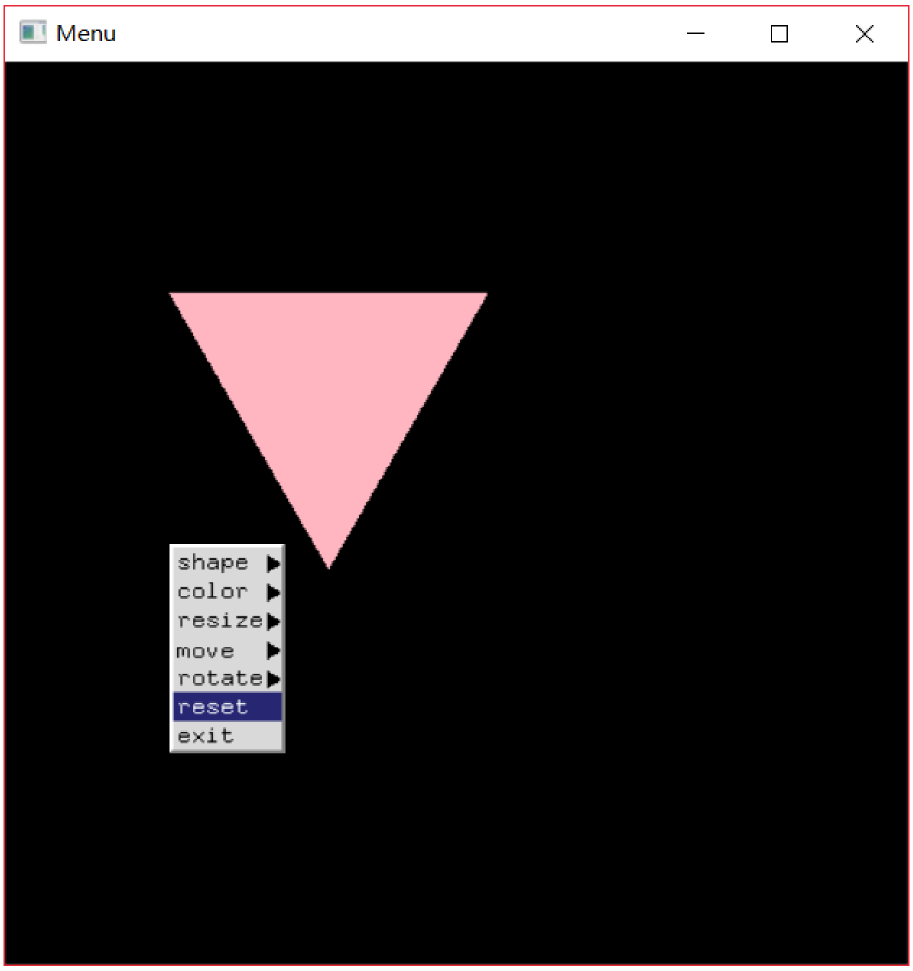
5、旋转（以连续两次向左旋转30度为例）：

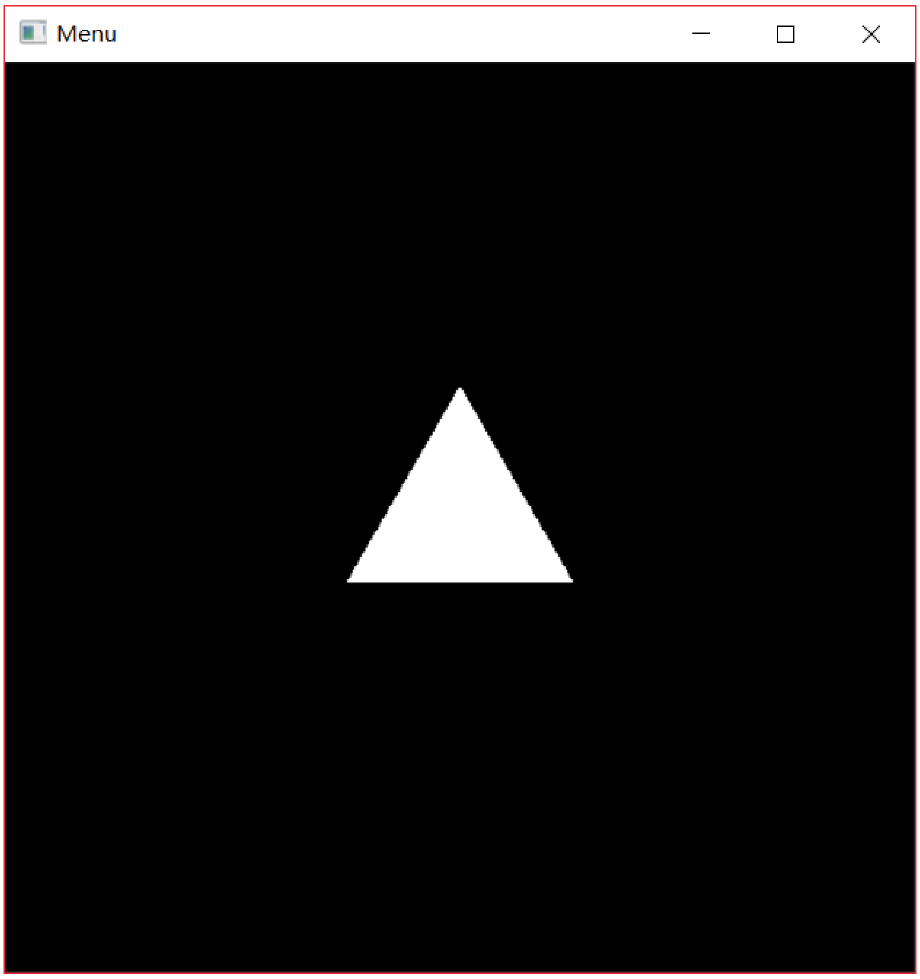




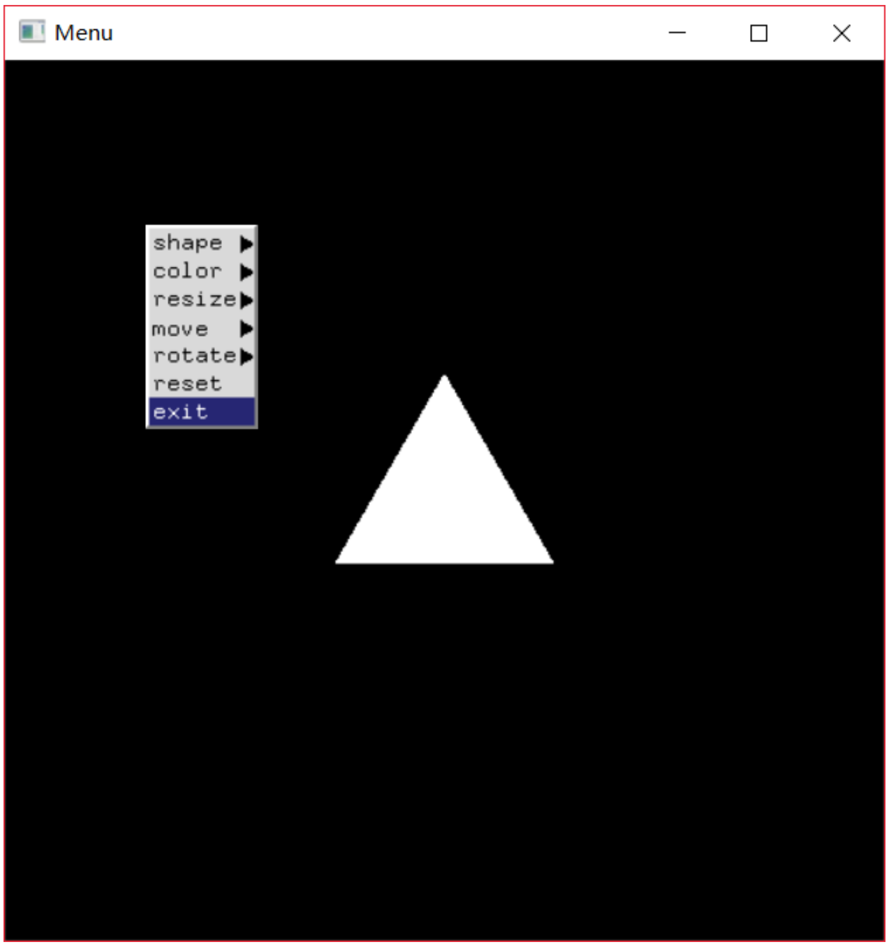


6、图形复位，恢复初始大小、位置和颜色：





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：

```python
from OpenGL.GL import *
from OpenGL.GLUT import *

# 全局变量记录形状选择
shape_mode = 0

def display():

    glClearColor(0.0, 0.0, 0.0, 1)
    glClear(GL_COLOR_BUFFER_BIT)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()


def process_menu_events(value):

    global color_mode
    if value == 0:

        exit()

    if value == 1:

        glLoadIdentity()
        glColor3f(1.0, 1.0, 1.0)
        draw_figure(shape_mode)


def draw_figure(value):
    ''' 根据选择的图元绘制相应的图形，并记录选择 '''
    global shape_mode
    shape_mode = value

    if value == 1:
        # 线段
        glClear(GL_COLOR_BUFFER_BIT)
        glLineWidth(5)
        glBegin(GL_LINES)
        glVertex2f(-0.5, 0.0)
        glVertex2f(0.5, 0.0)
        glEnd()
    if value == 2:
        # 三角形
        glClear(GL_COLOR_BUFFER_BIT)
        glShadeModel(GL_SMOOTH)
        glBegin(GL_TRIANGLES)
        glVertex2f(-0.25, -0.144)
        glVertex2f(0.25, -0.144)
        glVertex2f(0.0, 0.289)
        glEnd()


    if value == 3:
        # 矩形
        glClear(GL_COLOR_BUFFER_BIT)
        glShadeModel(GL_SMOOTH)
        glBegin(GL_POLYGON)
        glVertex2f(-0.5, -0.5)
        glVertex2f(-0.5, 0.5)
        glVertex2f(0.5, 0.5)
        glVertex2f(0.5, -0.5)
        glEnd()

    glFlush()


def figure_color_change(value):
    ''' 选择颜色 '''
    if value == 1:
        glColor3ub(255, 48, 48)

    elif value == 2:
        glColor3ub(255, 182, 193)

    elif value == 3:
        glColor3ub(0, 191, 255)

    draw_figure(shape_mode)


def figure_resize(value):
    ''' 改变图形大小 '''
    if value == 1:
        # 面积缩小为1/2
        glScaled(0.707, 0.707, 0.707)
        draw_figure(shape_mode)
    if value == 2:
        # 面积扩大为2 倍
        glScaled(1.414, 1.414, 1.414)
        draw_figure(shape_mode)


def figure_move(value):
    ''' 平移图形 '''
    if value == 1:
        # 向左平移0,2 单位
        glTranslatef(-0.2, 0.0, 0.0)
        draw_figure(shape_mode)

    if value == 2:
        # 向右平移0.2 单位
        glTranslatef(0.2, 0.0, 0.0)
        draw_figure(shape_mode)

    if value == 3:
        # 向上平移0.2 单位
        glTranslatef(0.0, 0.2, 0.0)
        draw_figure(shape_mode)

    if value == 4:
        # 向下平移0.2 单位
        glTranslatef(0.0, -0.2, 0.0)
        draw_figure(shape_mode)


def figure_rotate(value):
    ''' 旋转图形 '''
    if value == 1:
        # 向左旋转30 度
        glRotated(30, 0, 0, 1)
        draw_figure(shape_mode)

    if value == 2:
        # 向右旋转30 度
        glRotated(-30, 0, 0, 1)
        draw_figure(shape_mode)


def creat_menu():
    ''' 创建菜单 '''
    # 子菜单：选择绘图形状
    shape_menu = glutCreateMenu(draw_figure)
    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('0.2 unit left', 1)
    glutAddMenuEntry('0.2 unit right', 2)
    glutAddMenuEntry('0.2 unit up', 3)
    glutAddMenuEntry('0.2 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();
}
```