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。

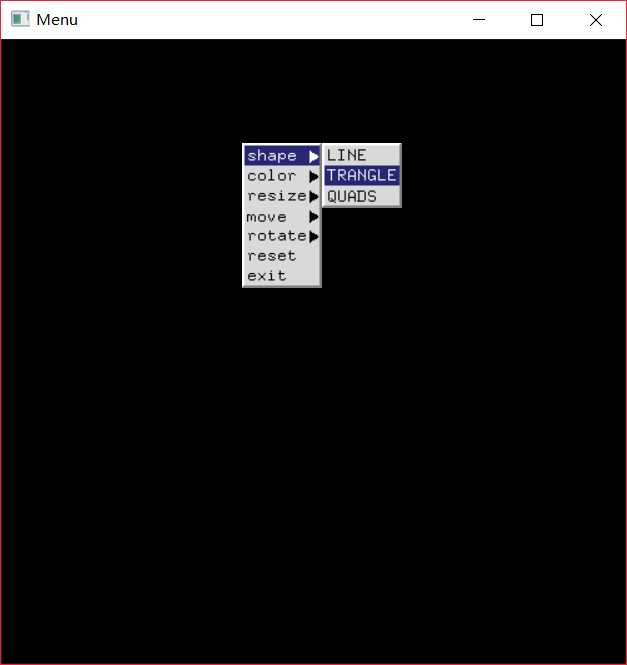
（5）你能想到的任何功能。

算法概括：

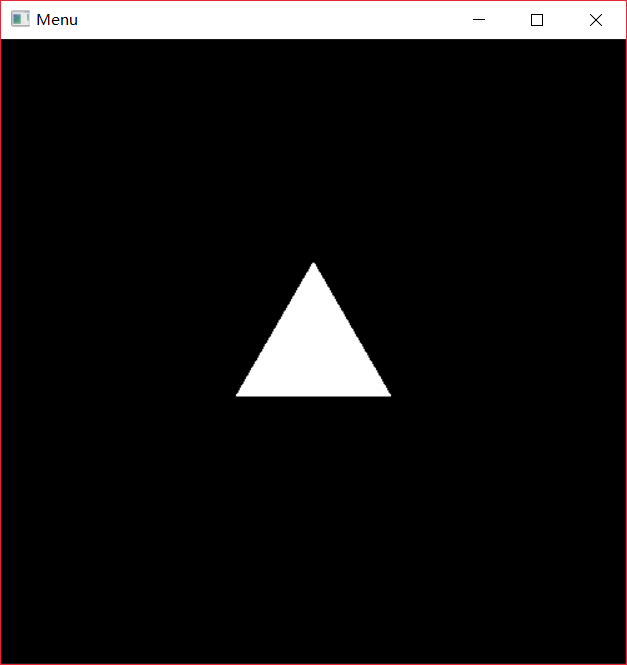
使用两个全局变量记录选择绘制的图形及颜色，以完成旋转、平移等操作。

演示：

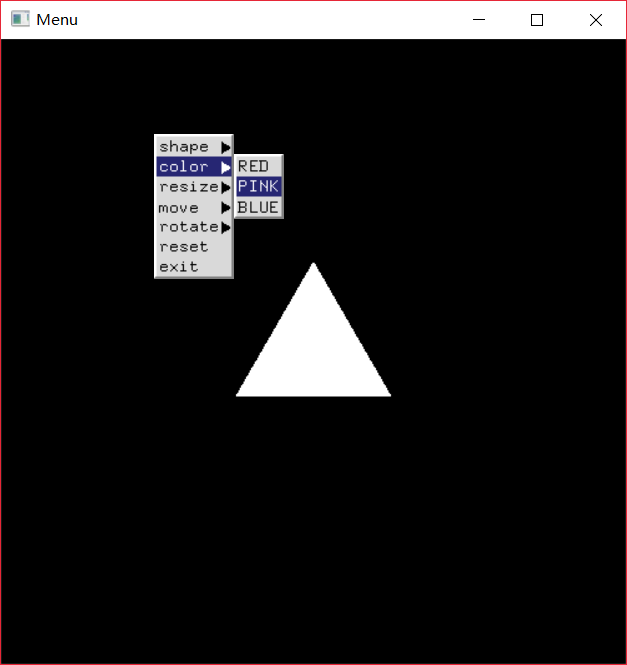
1、生成图形：

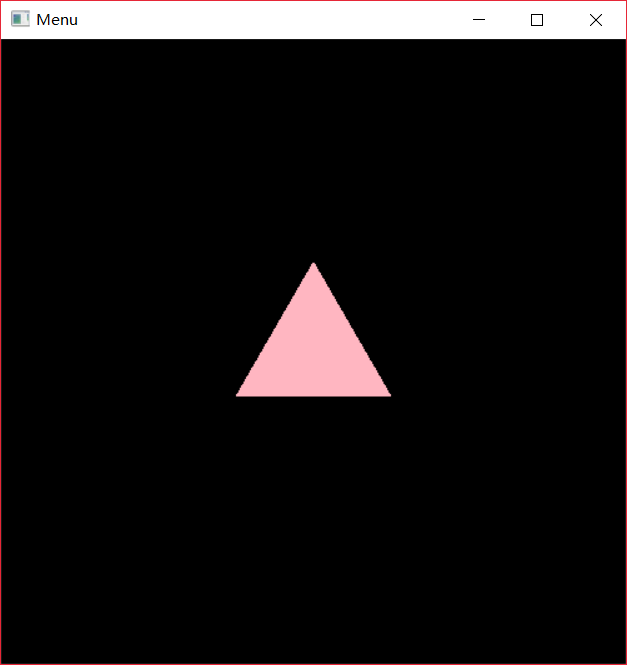


以三角形为例：

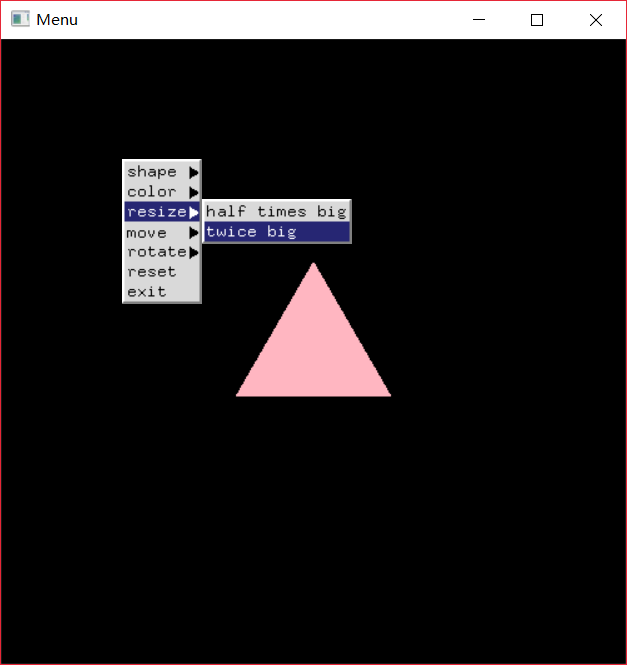


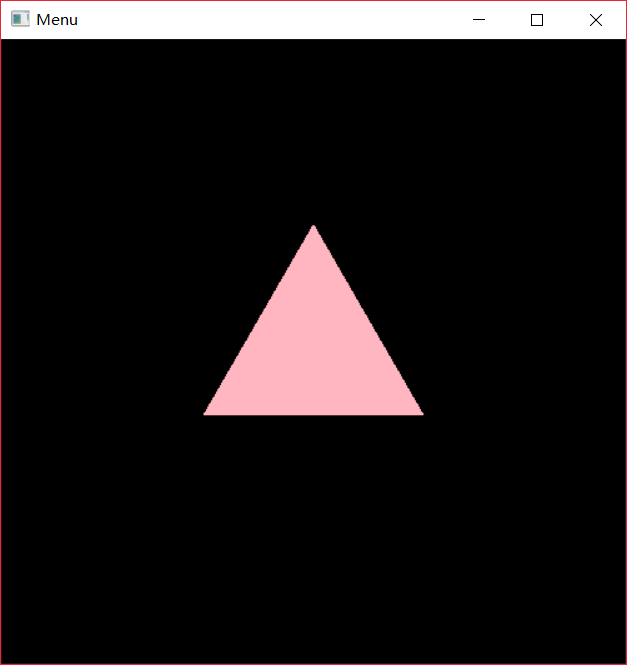
2、更改颜色（以粉色为例）：



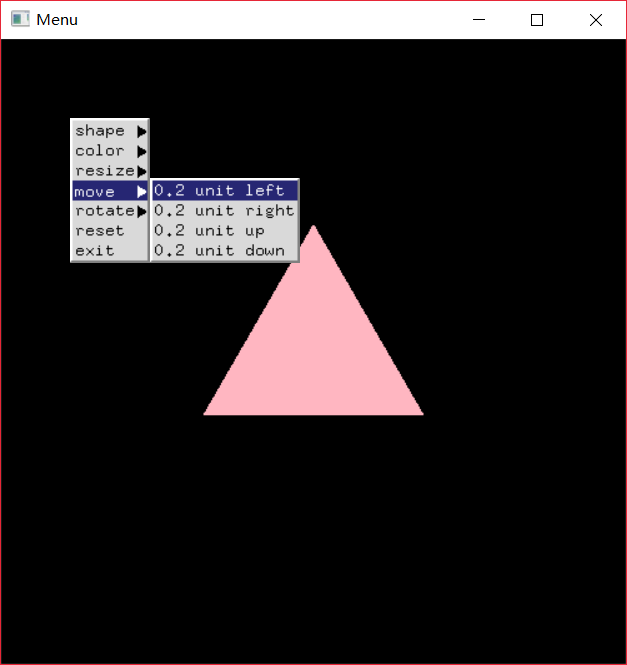


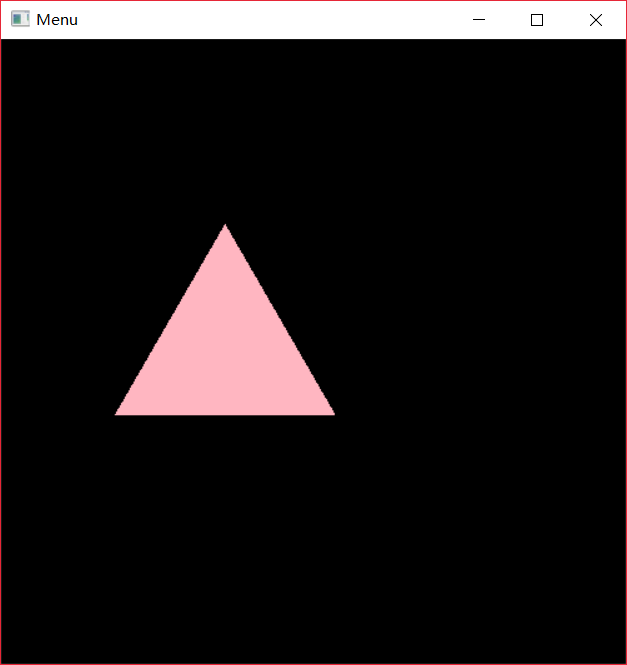
3、缩放（以放大两倍为例）：

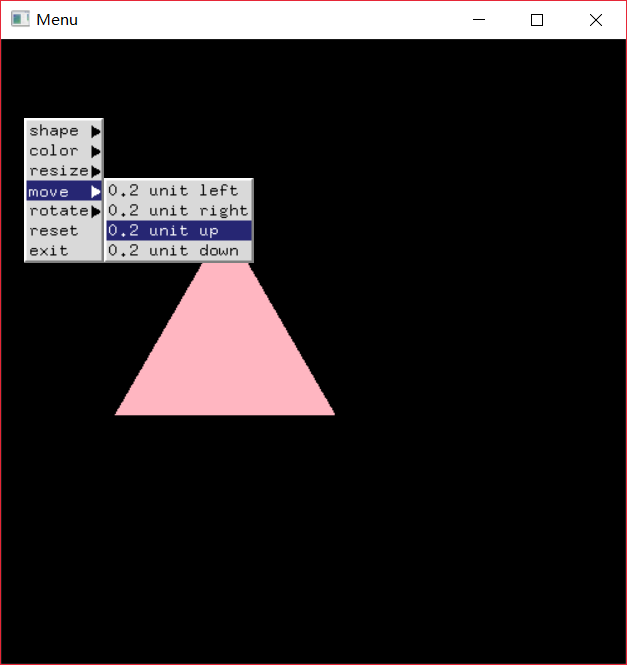


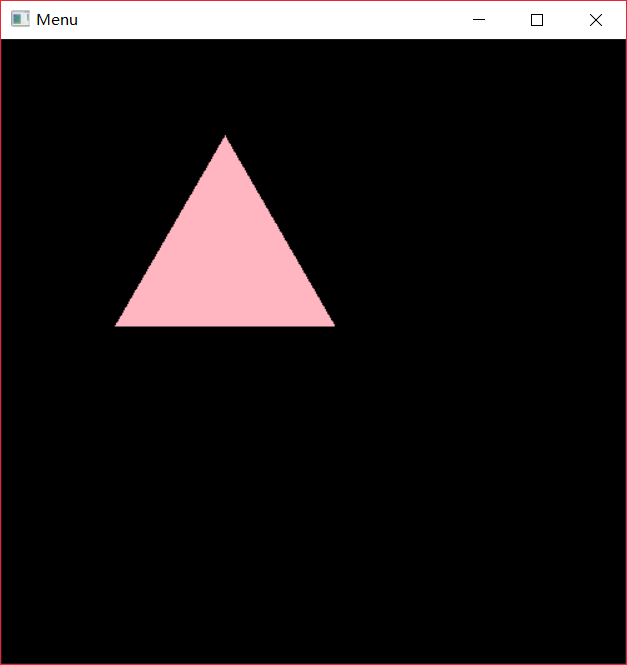


4、平移（以向左平移0.2个单位，再向上平移0.2单位为例）：

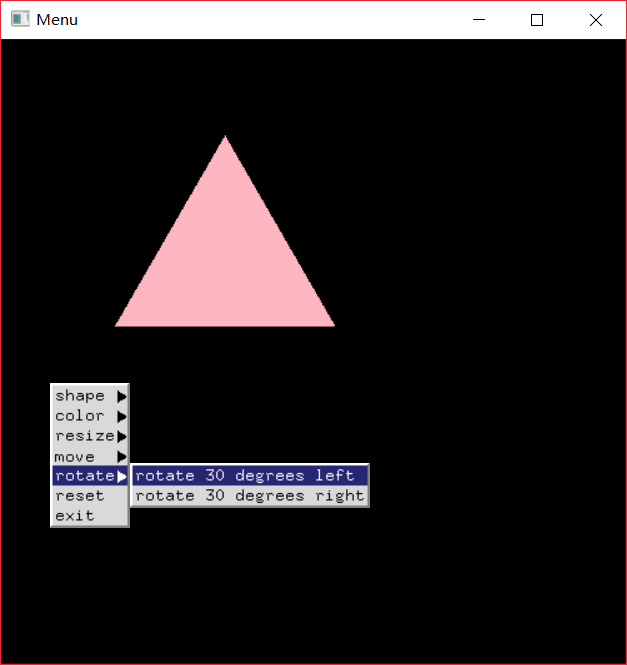


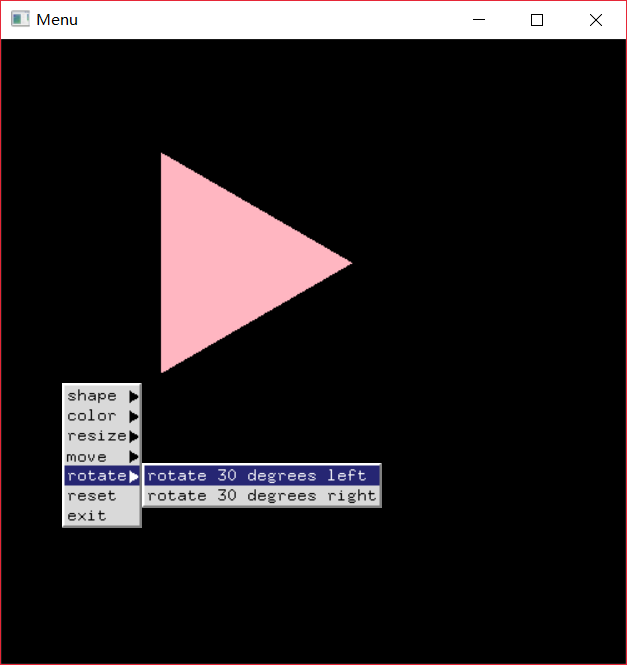


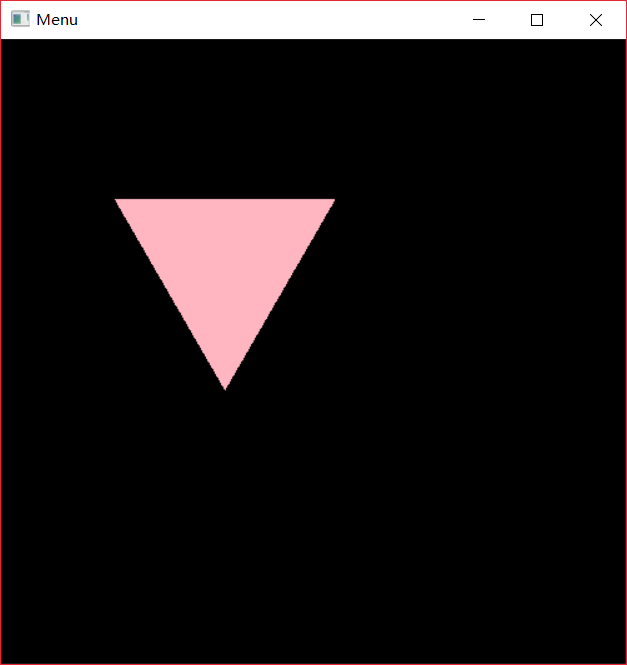




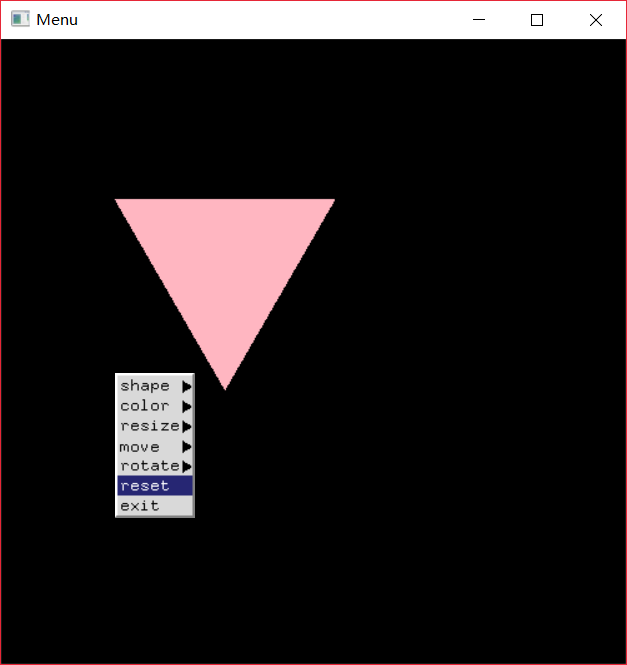
5、旋转（以连续两次向左旋转30度为例）：

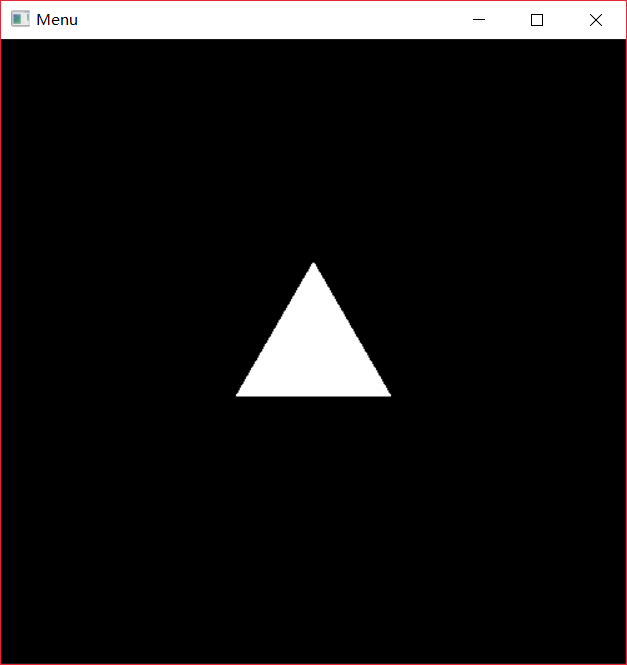




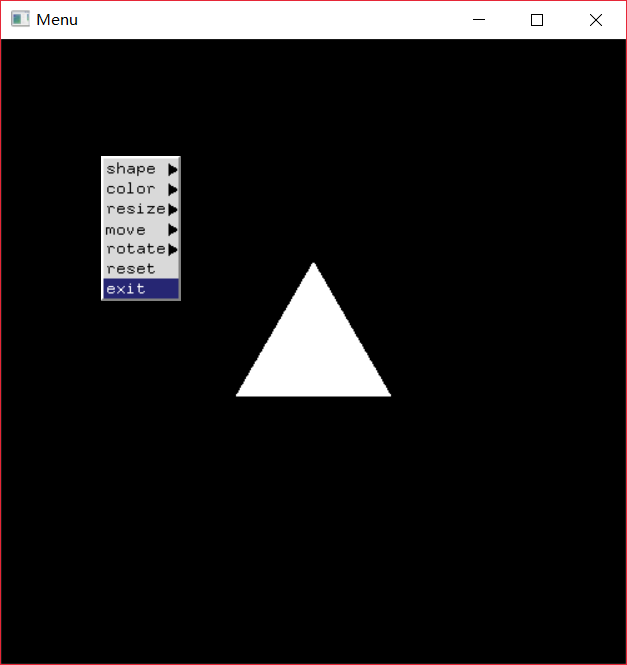


6、图形复位，恢复初始大小、位置和颜色：





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：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();

}