1. 证明如下的两条三次曲线段具有C1连续性，但没有G1连续性，并画出两条曲线段。

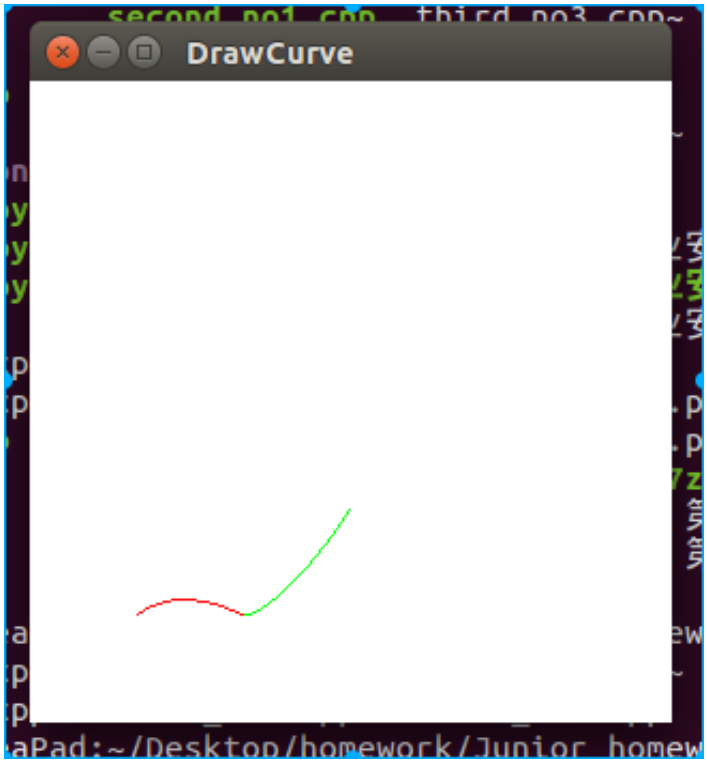
$$P_1 = \left[t^2 - 2t + 1, t^3 - 2t^2 + t\right] \tag{1}$$

$$P_2 = \left[t^2 + 1, t^3\right] \tag{2}$$

1.1. 证明过程：

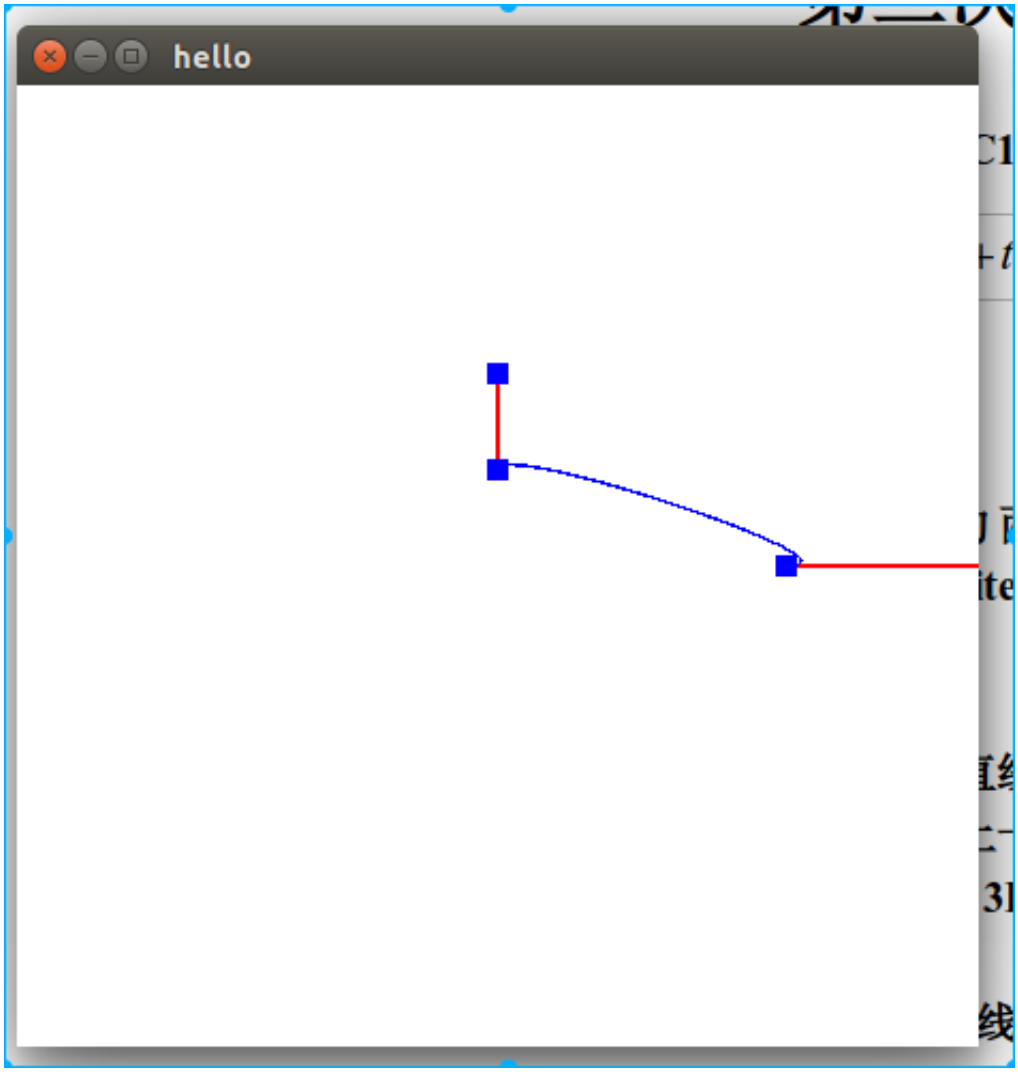
略

1.2. 实验结果



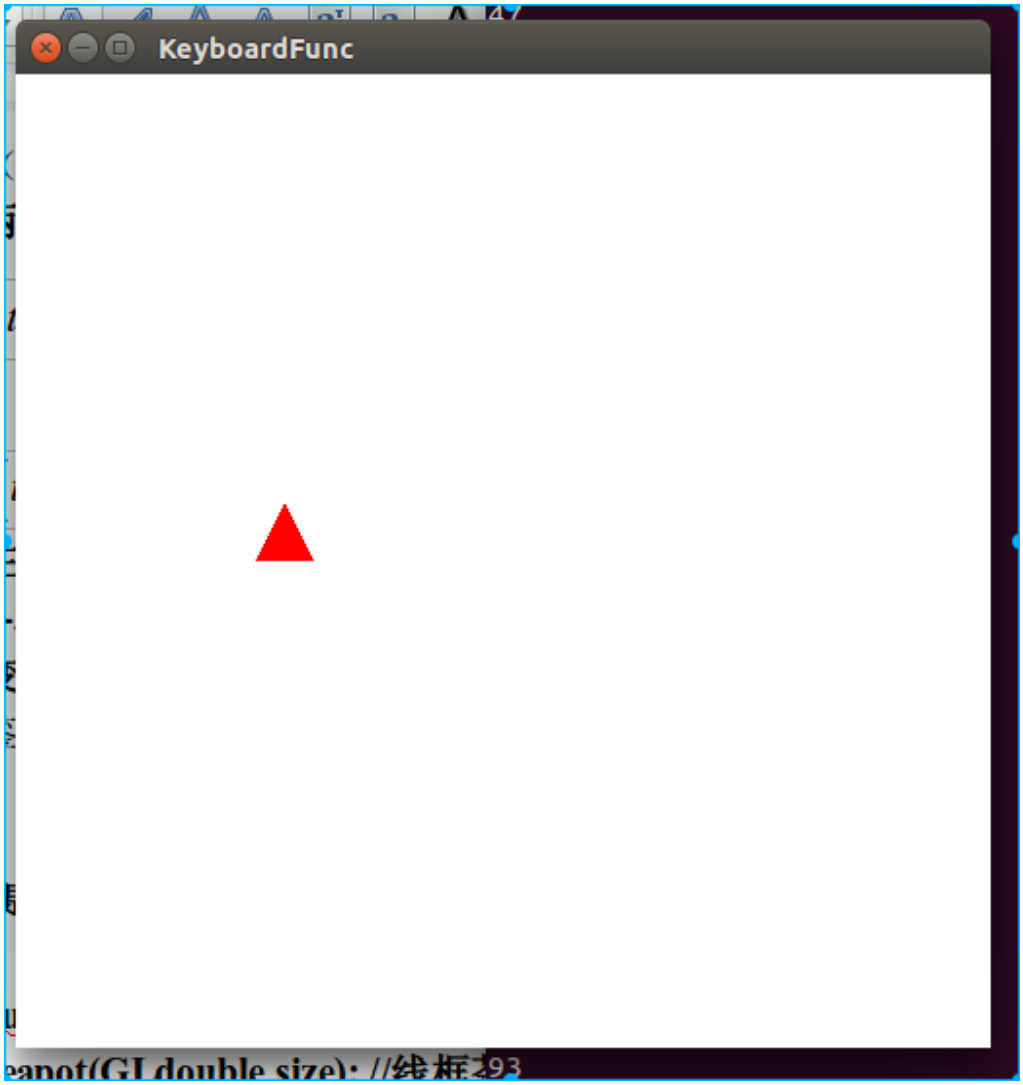
2. 假定一条三次Hermite曲线的两个端点P1=<0,1>,P4=<3,0>,端点处切向量R1=<0,1>,R4=<-3,0>，请写出Hermite多项式形式，并绘出最后曲线，改变切向量，观察曲线形状变化

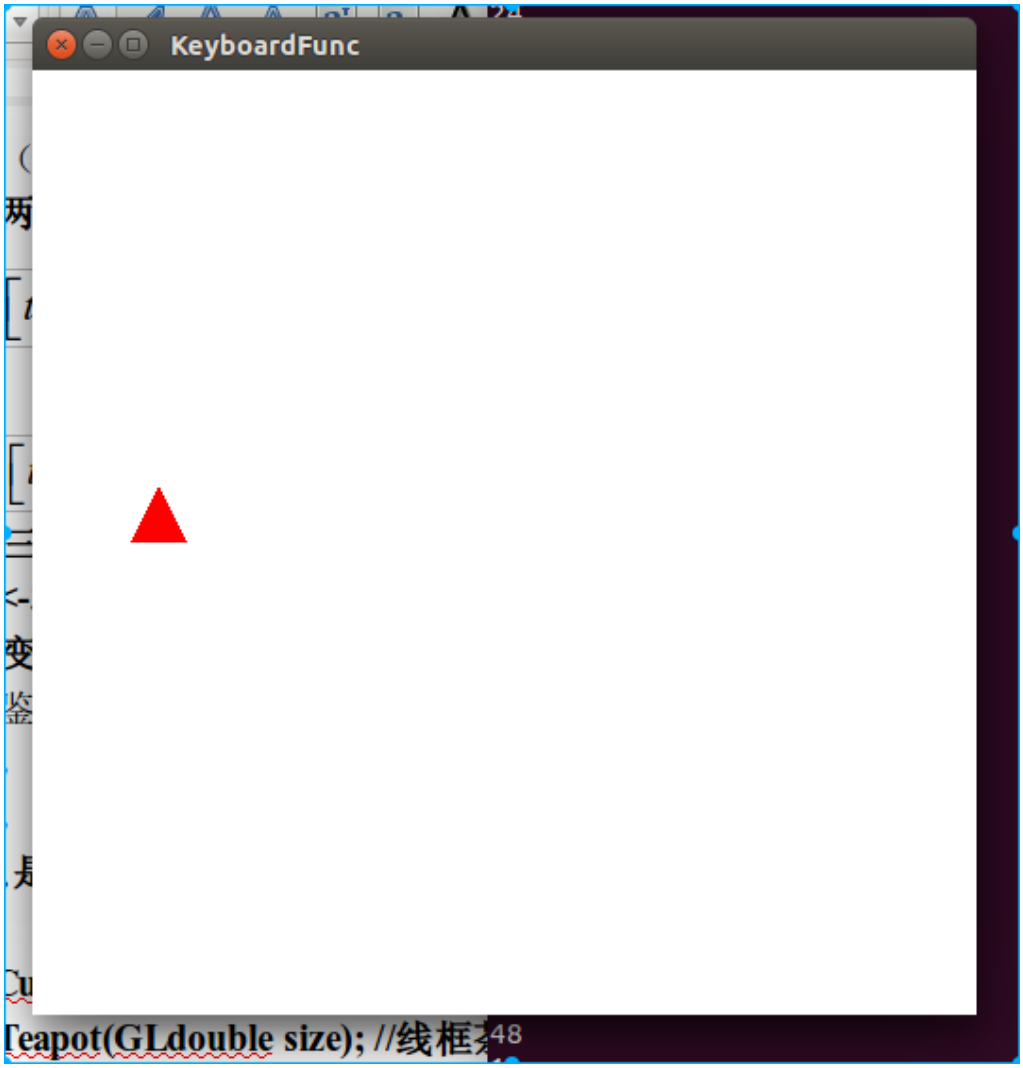
2.1. 实验结果



3. 编写程序，使一物体沿着一条直线匀速移动。

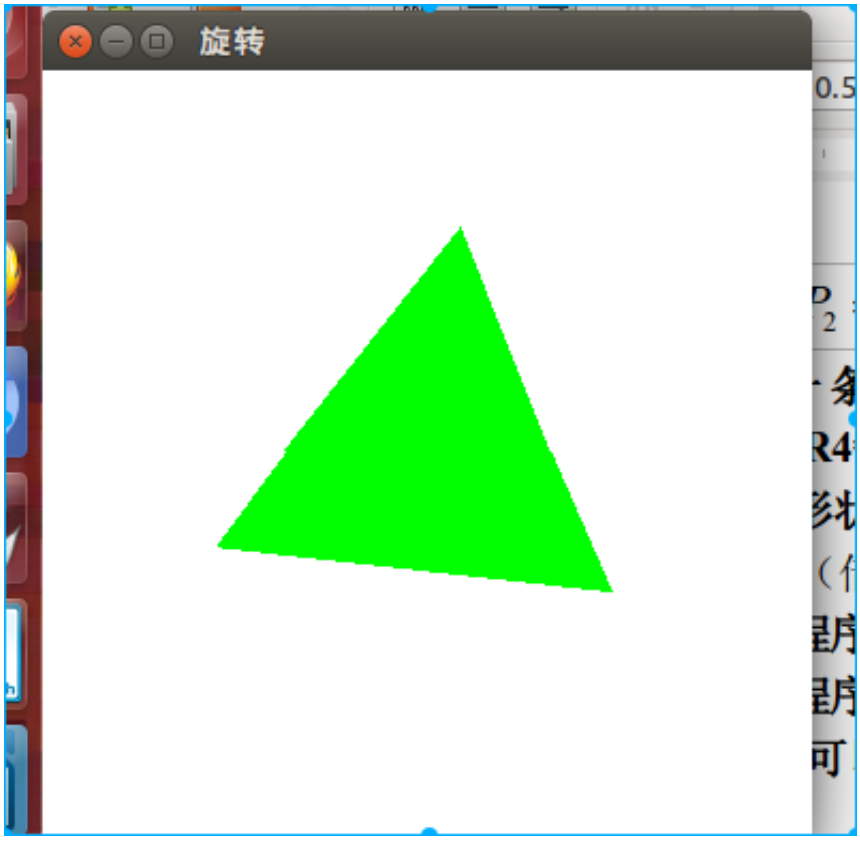
3.1. 实验结果

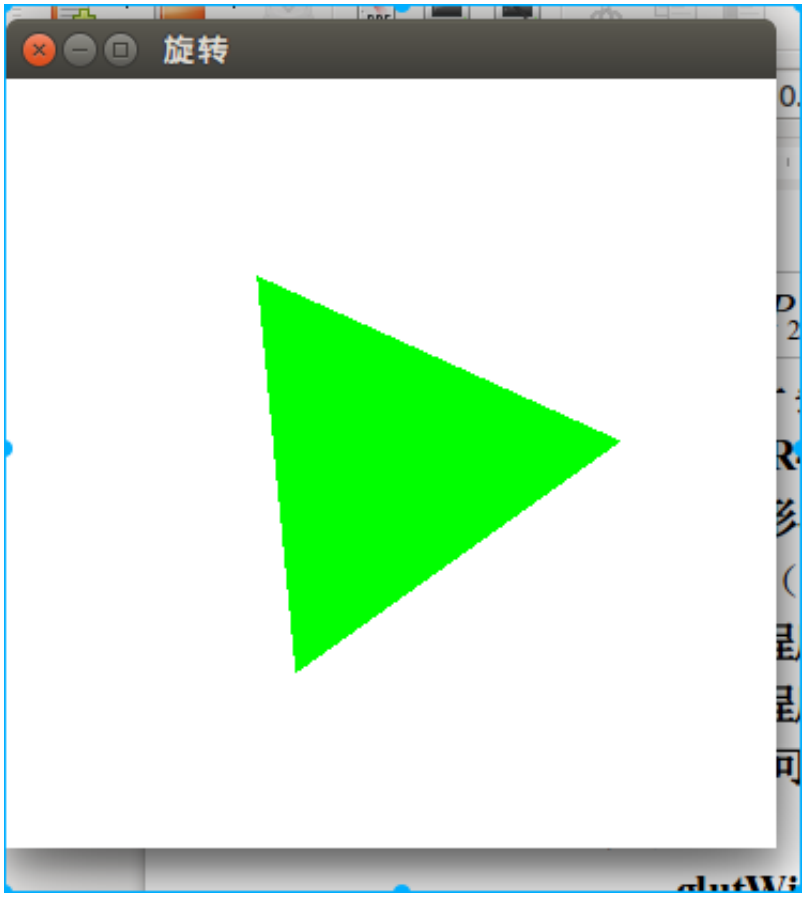

KeyboardFunc



4. 编写程序，使一物体围绕屏幕上一点匀速旋转。

4.1. 实验结果


旋转


旋转

5. 源代码

5.1. 题目一

```
#include <GL/glut.h>
typedef struct{
    float x,y;
}point;
void init()
{
    glClearColor(1.0,1.0,1.0,1.0);
}
void drawCurve1(int n)
{
    point pixels[100];
    float delta,t,t2,t3;
    int i;

    delta = 1.0/(n-1);
    glBegin(GL_LINE_STRIP);
        for(i=0;i<n;i++)
        {
            t = i*delta;
            t2 = t*t;
            t3 = t2*t;
            pixels[i].x = t2-2*t+1;
            pixels[i].y = t3-2*t2+t;
            glVertex2f(pixels[i].x,pixels[i].y);
        }
    glEnd();
}
void drawCurve2(int n)
{
    point pixels[100];
    float delta,t,t2,t3;
    int i;

    delta = 1.0/(n-1);
    glBegin(GL_LINE_STRIP);
        for(i=0;i<n;i++)
        {
            t = i*delta;
            t2 = t*t;
            t3 = t2*t;
            pixels[i].x = t2+1;
            pixels[i].y = t3;
            glVertex2f(pixels[i].x,pixels[i].y);
        }
    glEnd();
}
void RenderScene()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0,0.0,0.0);
    drawCurve1(100);
    glColor3f(0.0,1.0,0.0);
    drawCurve2(100);
    glFlush();
}
void ChangeSize(GLsizei w,GLsizei h)
{
    GLfloat aspectRatio;
    if(h==0)
        h = 1;
    glViewport(0,0,w,h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    aspectRatio = (GLfloat)w/(GLfloat)h;
```

```
    if(w<=h)
        glOrtho(-1.0,5.0,-1.0/aspectRatio,5.0/aspectRatio,1.0,-1.0);
    else
        glOrtho(-1.0*aspectRatio,5.0*aspectRatio,-1.0,5.0,1.0,-1.0);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
int main(int argc,char *argv[])
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_RGB|GLUT_SINGLE);
    glutCreateWindow("DrawCurve");
    init();
    glutDisplayFunc(RenderScene);
    glutReshapeFunc(ChangeSize);
    glutMainLoop();
    return 0;
}
```

5.2. 题目二

```
#include <math.h>
#include <GL/glut.h>
#include <iostream>
using namespace std;

struct  Point2
{
    double x;
    double y;

    Point2(int px, int py) { x = px; y = py; }
};

Point2 P0(0, 1);
Point2 P1(3, 0);
Point2 derP0(0, 1);
Point2 derP1(-3, 0);

bool mouseLeftDown = false;
bool mouseRightDown = false;

/*计算Hermite曲线*/
void Hermit(int n)
{
    float f1, f2, f3, f4;

    double deltaT = 1.0 / n;

    glBegin(GL_LINE_STRIP);
    for (int i = 0; i <= n; i++) {

        double T = i * deltaT;

        f1 = 2.0*pow(T, 3)  - 3.0*pow(T, 2) + 1.0;
        f2 = -2.0*pow(T, 3)  + 3.0*pow(T, 2);
        f3 =  pow(T, 3)   - 2.0*pow(T, 2) + T;
        f4 =  pow(T, 3)  - pow(T, 2);

        glVertex2f( f1*P0.x + f2*P1.x + f3*derP0.x + f4*derP1.x,
            f1*P0.y + f2*P1.y + f3*derP0.y + f4*derP1.y );
    }
    glEnd();
}

/*用鼠标进行绘制，完成后可改变控制点，拖动即可*/
void display(){
    glClear(GL_COLOR_BUFFER_BIT);
```

```
    glLineWidth(1.5);
    glColor3f (1.0, 0.0, 0.0);
    glBegin(GL_LINES);
    glVertex2f(P0.x, P0.y);
    glVertex2f(P0.x + derP0.x, P0.y + derP0.y);
    glVertex2f(P1.x, P1.y);
    glVertex2f(P1.x - derP1.x, P1.y - derP1.y);
    glEnd();

    glColor3f (0.0, 0.0, 1.0);
    glPointSize(10.0f);

    glBegin(GL_POINTS);
    glVertex2f(P0.x, P0.y);
    glVertex2f(P0.x + derP0.x, P0.y + derP0.y);
    glVertex2f(P1.x, P1.y);
    glVertex2f(P1.x - derP1.x, P1.y - derP1.y);
    glEnd();

    Hermit(20);

    glFlush();
    glutSwapBuffers();
}

void init()
{
    glClearColor(1.0, 1.0, 1.0, 0.0);
    glShadeModel(GL_FLAT);
//    gluOrtho2D(-5,5,-5,5);
}

void myReshape(int w, int h)
{
    glViewport(0, 0, (GLsizei)w, (GLsizei)h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-(GLsizei)w, (GLsizei)w, (GLsizei)h, (GLsizei)h);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void mouse(int button, int state, int x, int y)
{
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
    {
        mouseLeftDown = true;
    }

    if (button == GLUT_LEFT_BUTTON && state == GLUT_UP)
    {
        mouseLeftDown = false;
    }

    if (button == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
    {
        mouseRightDown = true;
    }

    if (button == GLUT_RIGHT_BUTTON && state == GLUT_UP)
    {
        mouseRightDown = false;
    }
}

double distance(int x1, int y1, int x2, int y2)
{
    return sqrt((x1-x2) * (x1 -x2) + (y1-y2) * (y1-y2));
}

void motion(int x, int y)
{
```

```
    if (mouseLeftDown)
    {
        if (distance(P0.x + derP0.x/4, P0.y + derP0.y/4, x, y) < 20)
        {
            derP0.x = (x - P0.x)*4;
            derP0.y = (y - P0.y)*4;
        }

        if (distance(P1.x - derP1.x/4, P1.y - derP1.y/4, x, y) < 20)
        {
            derP1.x = (P1.x - x)*4;
            derP1.y = (P1.y - y)*4;
        }
    }

    glutPostRedisplay();
}

int  main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB );
    glutInitWindowSize (450, 450);
    glutInitWindowPosition (200, 200);
    glutCreateWindow ("hello");
    init ();
    gluOrtho2D(-5,5,-5,5);
    glutDisplayFunc(display);
//    glutReshapeFunc(myReshape);
    glutMouseFunc(mouse);
    glutMotionFunc(motion);

    glutMainLoop();
    return 0;
}
```

5.3. 题目三

```
#include <stdlib.h>
#include <stdio.h>
#include <GL/glut.h>
#include "math.h"
#include "bits/stdc++.h"
using namespace std;

int width=500,height=500;
float xx1=0.0,yy1=0.0,xx2=15.0,yy2=30.0,xx3=30.0,yy3=0;
int n=500;
int flag=1;
void init()
{
    glClearColor(1.0,1.0,1.0,1.0);
    gluOrtho2D(0,width,-height/2,height/2);
}
void mydisplay(){
//    cout << xx1 << yy1 << endl;
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_TRIANGLES);
    glColor3f(1.0,0.0,0.0);
    glVertex2f(xx1,yy1);
    glVertex2f(xx2,yy2);
    glVertex2f(xx3,yy3);
    glEnd();
    glFlush();
}


void myidle(){
    float det=1.0*width/n;
    cout << det << endl;
    if(flag==1){
```

```
        xx1+=det;
        xx2+=det;
        xx3+=det;
        if(xx3+det>width){
            flag=-1;
        }
    }else if(flag==-1){
        xx1-=det;
        xx2-=det;
        xx3-=det;
        if(xx1-det<0){
            flag=1;
        }
    }
    cout << xx1 << endl;
    mydisplay();
}

int main(int argc,char *argv[]){
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowPosition(50,50);
    glutInitWindowSize(height,width);
    glutCreateWindow("KeyboardFunc");
    init();
    glutDisplayFunc(mydisplay);
    glutIdleFunc(myidle);
//    glutReshapeFunc(ChangeSize);
//    glutKeyboardFunc(myKey);  //Ï��ǰ�������ü��ɣ������
//    printf("Press space to continue,press escape to exit!\n");
    glutMainLoop();
}
```

5.4. 题目四

```
#include <stdlib.h>
#include <stdio.h>
#include <GL/glut.h>
#include "math.h"
#include "bits/stdc++.h"
using namespace std;
int currentMode = 0;
const int ModeNums = 7;
float theta=0.0;
float PI=3.1415926;
float R=6.0,n=3;
void init()
{
    glClearColor(1.0,1.0,1.0,1.0);
}

void mydisplay(){
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_TRIANGLES);
    glColor3f(0.0,1.0,0.0);
    int i=0;
    for(int i=0;i<n;i++){
        glVertex2f(R*cos(theta+i*2*PI/n),R*sin(theta+i*2*PI/n));
    }
    glEnd();
    glFlush();
}

void myidle(){
    theta+=0.001;
    if(theta>=2*PI) theta-=2*PI;
    mydisplay();
}
```

```
void ChangeSize(GLsizei w,GLsizei h)
{
    float ratio;
    if(h==0)
        h = 1;
    glViewport(0,0,w,h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    ratio = (float)w/(float)h;
    if(w<=h)
        gluOrtho2D(-10.0,10.0,-10.0/ratio,10.0/ratio);
    else
        gluOrtho2D(-10.0*ratio,10.0*ratio,-10.0,10.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}



int main(int argc,char *argv[])
{
    printf("%lf\n",sin(1/2.0*PI));
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowPosition(50,50);
    glutInitWindowSize(360,360);
    glutCreateWindow("旋转");

    init();
    glutDisplayFunc(mydisplay);
    glutReshapeFunc(ChangeSize);
    glutIdleFunc(myidle);
    printf("Press space to continue,press escape to exit!\n");
    glutMainLoop();
}
```