OpenGL坐标变换　平移，缩放与旋转

OpenGL有内建的坐标系，事实上OpenGl有两套坐标系，一个坐标系被称为眼睛坐标（eye coordinate system）　简称ECS 。 OpenGL还有一套坐标，被称为（object coordinate system）　简称OCS ,而这个才是更为重要的，其实我们用来绘图的正是OCS。

两个坐标系中ECS 可以看成是一个现实存在的　基本不变的全局坐标系，而OCS则可以看成是用户自定义的坐标系，我们可以将这个坐标系任意的平移与缩放，在初始情况下他和ECS是重合的，也可以通过glLoadIdentity()强制复位，这样可以给我们的绘图带来极大的方便。这里有一点是要值得注意的是在使用一个函数时需要弄清它是使用什么坐标系的，刚刚我们用到的glVertex系列函数都是用的OCS

下面是一个平移和缩放例子：

# 简单旋转，平移、缩放

#include<GL/glut.h>

#include <windows.h>

static GLfloat spin = 0.0; //旋转量

static GLfloat move = 0.0; //平移量

static GLfloat size = 1.0; //缩放量

void init(void)

{

glClearColor(0.0, 0.0, 0.0, 0.0); //指定清除颜色（黑色），即背景颜色

glShadeModel(GL\_FLAT);

}

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT); //清除所有的像素

glPushMatrix();

glTranslatef(move, 0, 0); //移动，参数含义(x轴位移,y轴位移,z轴位移)

glRotatef(spin, 0, 0, 1); //旋转，参数含义(旋转量,x轴,y轴,z轴)

glScalef(size, size, 1); //缩放，参数含义(x轴倍数,y轴倍数,z轴倍数)

glColor3f(0.0, 1.0, 0.0); //绘制颜色RGB

glRectf(-25.0, -25.0, 25.0, 25.0); //绘制矩形

glPopMatrix();

glutSwapBuffers();

}

void reshape(int w, int h)

{

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-50.0, 50.0, -50.0, 50.0, -1.0, 1.0);

}

void spinAndSizeDisplay()

{

spin > 360 ? spin -= 360 : spin += 2;

size > 2 ? size -= 2 : size += 0.003;

glutPostRedisplay(); //标记当前窗口需要重绘，否则不会旋转

Sleep(100);

}

void moveDisplay()

{

move = move > 20 ? move -= 20 : move += 1;

glutPostRedisplay(); //标记当前窗口需要重绘，否则不会旋转

Sleep(100);

}

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

{

glutInit(&argc, argv); //初始化GLUT并处理命令行参数

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB); //指定模式：双缓存；RGB模式

glutInitWindowSize(250, 250); //指定窗口大小（像素）

glutInitWindowPosition(300, 200); //指定窗口左上角在屏幕上的位置

glutCreateWindow(argv[0]); //使用OpenGL场景创建一个窗口，参数为窗口名称

init(); //调用初始化函数

glutDisplayFunc(display); //显示

glutReshapeFunc(reshape); //重绘

//glutIdleFunc(spinAndSizeDisplay); //旋转&缩放

glutIdleFunc(moveDisplay); //移动(与上边函数只能有一个有效)

glutMainLoop(); //进入主循环并处理事件，此时创建的所有窗口都会显示出来，被渲染到这些窗口中的内容也将显示出来，程序开始处理事件，注册的显示回调函数被触发

return 0; //ANSI C要求函数main()返回一个int值

}

# 基于交互的图形平移-旋转-缩放

原文链接：https://blog.csdn.net/weixin\_57040827/article/details/121677602

//\*\*\*\*\*\*\*\*将头文件嵌入源文件中（2-6行）\*\*\*\*\*\*\*\*

#include<GL/glut.h>

#include<GL/freeglut.h>

#include<GL/gl.h>

#include<iostream>

#include <math.h>

using std::cout; //控制台输出

#define \_USE\_MATH\_DEFINES //使用数学定义

//\*\*\*\*\*\*\*\*定义窗口大小（10-11行）\*\*\*\*\*\*\*\*

GLfloat w = 700;

GLfloat h = 700;

//\*\*\*\*\*\*\*\*图形沿x和y轴旋转（13-14行）\*\*\*\*\*\*\*\*

double rotate\_x = 0.0;

double rotate\_y = 0.0;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*对颜色的宏定义（16-27行）\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# define white 1.0 , 1.0 , 1.0

# define green 0.0 , 0.502, 0.0

# define red 1.0 , 0.0 , 0.0

# define gray 0.502, 0.502,0.502

# define hgray 0.117, 0.180,0.227

# define blue 0.0 , 0.0 , 1.0

# define pi 3.14159265

# define gold 1.0,215.0/255.0,0.0

# define hgreen 0.0,100.0/255.0,0.0

# define brown 210.0/255.0, 105.0/255.0, 30.0/255.0

# define men 244.0/255.0 ,164.0/255.0 , 96.0/255.0

# define menba 139.0/255.0 ,69.0/255.0,19.0/255.0

double fang[8][3]; // 定义长方体的8个顶点，定义constract函数，当给定长方体的最里面的点时，计算其余的7个顶点 并赋值给fang

double san[8][3]; // 定义地面为梯形的长方体

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*绘制球体轮廓（31-55行）\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void drawSphere(double r, int lats, int longs, double x, double y, double z) {

int i, j;

for (i = 0; i <= lats; i++) {

double lat0 = pi \* (-0.5 + (double)(i - 1) / lats);

double z0 = sin(lat0);

double zr0 = cos(lat0);

double lat1 = pi \* (-0.5 + (double)i / lats);

double z1 = sin(lat1);

double zr1 = cos(lat1);

glBegin(GL\_QUAD\_STRIP); //以点绘制图形

glScaled(100, 100, 100); //当前图形沿x，y，z轴分别放大原来的100倍

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*在drawSphere-for循环中绘制法线，产生光照（45-54行）\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

for (j = 0; j <= longs; j++) {

double lng = 2 \* pi \* (double)(j - 1) / longs;

double x = cos(lng);

double y = sin(lng);

glNormal3f(x \* zr0, y \* zr0, z0);

glVertex3f(zr0, zr0, z0);

glNormal3f(x \* zr1, y \* zr1, z1);

glVertex3f(zr1, zr1, z1);

}

glEnd();

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*绘制梯形长方体（58-88行）\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void cons(double x, double y, double z, double x1, double y1, double z1) {

san[0][0] = x;

san[0][1] = y;

san[0][2] = z; // 第0个点

san[1][0] = x;

san[1][1] = y;

san[1][2] = z + z1; // 第1个点

san[4][0] = x;

san[4][1] = y + y1;

san[4][2] = z; // 第4个点

san[5][0] = x;

san[5][1] = y + y1;

san[5][2] = z + z1 / 2; // 第5个点

for (int i = 0; i < 3; i++) {

if (i == 0) {

san[3][i] = san[0][i] + x1;

san[2][i] = san[1][i] + x1;

san[6][i] = san[4][i] + x1;

san[7][i] = san[5][i] + x1;

}

else {

san[3][i] = san[0][i];

san[2][i] = san[1][i];

san[6][i] = san[4][i];

san[7][i] = san[5][i];

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*利用长方体的8个顶点绘制长方体（90-114行）\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void constract(double x, double y, double z, double x1, double y1, double z1) {

fang[0][0] = x;

fang[0][1] = y;

fang[0][2] = z; // 第0个点

fang[1][0] = x;

fang[1][1] = y;

fang[1][2] = z + z1; // 第一个点

fang[2][0] = x + x1;

fang[2][1] = y;

fang[2][2] = z + z1; // 第二个点

fang[3][0] = x + x1;

fang[3][1] = y;

fang[3][2] = z; // 第三个点

for (int i = 0; i < 4; i++) { // for()循环来完成其余的四个点

for (int j = 0; j < 3; j++) {

if (j == 1)

fang[i + 4][j] = fang[i][j] + y1;

else

fang[i + 4][j] = fang[i][j];

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*房顶（116-164行）\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void build2() {

glBegin(GL\_POLYGON); //填充凸多边形

glNormal3f(0.0, -1.0, 0.0);

glVertex3f(san[0][0], san[0][1], san[0][2]); // 调用 glVertex 时，当前颜色、法线和纹理坐标与顶点关联。

glVertex3f(san[1][0], san[1][1], san[1][2]);

glVertex3f(san[2][0], san[2][1], san[2][2]);

glVertex3f(san[3][0], san[3][1], san[3][2]);

glEnd(); // 下底

glBegin(GL\_POLYGON);

glNormal3f(-1.0, 0.0, 0.0);

glVertex3f(san[1][0], san[1][1], san[1][2]);

glVertex3f(san[0][0], san[0][1], san[0][2]);

glVertex3f(san[4][0], san[4][1], san[4][2]);

glVertex3f(san[5][0], san[5][1], san[5][2]);

glEnd(); // 左面

glBegin(GL\_POLYGON);

glNormal3f(1.0, 0.0, 0.0);

glVertex3f(san[7][0], san[7][1], san[7][2]);

glVertex3f(san[6][0], san[6][1], san[6][2]);

glVertex3f(san[3][0], san[3][1], san[3][2]);

glVertex3f(san[2][0], san[2][1], san[2][2]);

glEnd(); // 右面

glBegin(GL\_POLYGON);

glNormal3f(0.0, 0.0, 1.0);

glVertex3f(san[5][0], san[5][1], san[5][2]);

glVertex3f(san[6][0], san[6][1], san[6][2]);

glVertex3f(san[2][0], san[2][1], san[2][2]);

glVertex3f(san[1][0], san[1][1], san[1][2]);

glEnd(); // 前面

glBegin(GL\_POLYGON);

glNormal3f(0.0, 0.0, -1.0);

glVertex3f(san[0][0], san[0][1], san[0][2]);

glVertex3f(san[3][0], san[3][1], san[3][2]);

glVertex3f(san[7][0], san[7][1], san[7][2]);

glVertex3f(san[4][0], san[4][1], san[4][2]);

glEnd(); // 后面

glBegin(GL\_POLYGON);

glNormal3f(0.0, 1.0, 0.0);

glVertex3f(san[4][0], san[4][1], san[4][2]);

glVertex3f(san[7][0], san[7][1], san[7][2]);

glVertex3f(san[6][0], san[6][1], san[6][2]);

glVertex3f(san[5][0], san[5][1], san[5][2]);

glEnd(); // 上面

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*房子（166-214行）\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void build() {

glBegin(GL\_POLYGON);

glNormal3f(0.0, -1.0, 0.0);

glVertex3f(fang[0][0], fang[0][1], fang[0][2]);

glVertex3f(fang[1][0], fang[1][1], fang[1][2]);

glVertex3f(fang[2][0], fang[2][1], fang[2][2]);

glVertex3f(fang[3][0], fang[3][1], fang[3][2]);

glEnd(); // 下底

glBegin(GL\_POLYGON);

glNormal3f(-1.0, 0.0, 0.0);

glVertex3f(fang[1][0], fang[1][1], fang[1][2]);

glVertex3f(fang[0][0], fang[0][1], fang[0][2]);

glVertex3f(fang[4][0], fang[4][1], fang[4][2]);

glVertex3f(fang[5][0], fang[5][1], fang[5][2]);

glEnd(); // 左面

glBegin(GL\_POLYGON);

glNormal3f(1.0, 0.0, 0.0);

glVertex3f(fang[7][0], fang[7][1], fang[7][2]);

glVertex3f(fang[6][0], fang[6][1], fang[6][2]);

glVertex3f(fang[2][0], fang[2][1], fang[2][2]);

glVertex3f(fang[3][0], fang[3][1], fang[3][2]);

glEnd(); // 右面

glBegin(GL\_POLYGON);

glNormal3f(0.0, 0.0, 1.0);

glVertex3f(fang[5][0], fang[5][1], fang[5][2]);

glVertex3f(fang[6][0], fang[6][1], fang[6][2]);

glVertex3f(fang[2][0], fang[2][1], fang[2][2]);

glVertex3f(fang[1][0], fang[1][1], fang[1][2]);

glEnd(); // 前面

glBegin(GL\_POLYGON);

glNormal3f(0.0, 0.0, -1.0);

glVertex3f(fang[0][0], fang[0][1], fang[0][2]);

glVertex3f(fang[3][0], fang[3][1], fang[3][2]);

glVertex3f(fang[7][0], fang[7][1], fang[7][2]);

glVertex3f(fang[4][0], fang[4][1], fang[4][2]);

glEnd(); // 后面

glBegin(GL\_POLYGON);

glNormal3f(0.0, 1.0, 0.0);

glVertex3f(fang[4][0], fang[4][1], fang[4][2]);

glVertex3f(fang[7][0], fang[7][1], fang[7][2]);

glVertex3f(fang[6][0], fang[6][1], fang[6][2]);

glVertex3f(fang[5][0], fang[5][1], fang[5][2]);

glEnd(); // 上面

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*绘制场景内各种组件（216-419行）\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT); //加载场景，如果将这行代码注释掉，将什么都看不到

glMatrixMode(GL\_MODELVIEW); //控制输出图形

glLoadIdentity(); //增加M矩阵

//\*\*\*\*\*\*\*\*\*\*\*\*\*控制x，y轴方向转动（222-223行）\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

glRotatef(rotate\_x, 1.0, 0.0, 0.0);

glRotatef(rotate\_y, 0.0, 1.0, 0.0);

glTranslatef(-270, 0, -270); //平移变换

glScalef(1.5, 1.5, 1.5); //图像缩放

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*增加地面底板（颜色绿色228-231行）\*\*\*\*\*\*\*\*

glBegin(GL\_POLYGON);

constract(0, 0, 0, 500, 10, 500);//地板

glColor3f(1.0,1.0,1.0);

build();

constract(0, 0, 0, 10, 500, 500);//左墙

glColor3f(1.0, 1.0, 1.0);

build();

constract(0, 0, 0, 500,500, 10);//后墙

glColor3f(1.0, 1.0, 1.0);

build();

constract(10, 10, 10, 200, 300, 300);//床体

glColor3f(0.5,0.5,0);

build();

constract(10, 300, 10, 200, 30, 300);//床垫子

glColor3f(1.0, 0.75, 0.44);

build();

constract(20, 20, 310, 180,270, 10);//镜子

glColor4f(white, 0.75);

build();

constract(210, 10, 10, 250, 60, 100);//楼梯

glColor3f(0.69, 0.77, 0.87);

build();

constract(210, 60, 10, 200, 60, 100);//楼梯

glColor3f(0.69, 0.77, 0.87);

build();

constract(210, 110, 10, 150, 60, 100);//楼梯

glColor3f(0.69, 0.77, 0.87);

build();

constract(210, 160, 10, 100, 60, 100);//楼梯

glColor3f(0.69, 0.77, 0.87);

build();

constract(210, 210, 10, 50, 60, 100);//楼梯

glColor3f(0.69, 0.77, 0.87);

build();

constract(230, 100, 100, 130, 10, 80);//桌面

glColor3f(0.8, 0.34, 0.34);

build();

constract(270, 10, 200, 50, 60, 40);//椅子

glColor3f(1.0, 0.76, 0.06);

build();

constract(250, 110, 130, 50, 3, 40);//电脑

glColor3f(0.0,0.0,0.0);

build();

constract(250, 110, 130, 50, 40, 3);//电脑屏幕

glColor3f(0.0, 0.0, 0.0);

build();

glColor3f(hgreen);

glTranslatef(330, 140, 130);

glutSolidSphere(16.0, 10, 10); //盆栽球

constract(-5, 0, -5, 5, -30, 5);//树干

glColor3f(brown);

build();

constract(169, -130, 200, 12, 250, 130);//门

glColor3f(red);

build();

constract(159, 0, 200,10 , 10, 10);//门把手

glColor3f(0.0, 0.0, 0.0);

build();

constract(179, 0, 200, 10, 10, 10);//门把手

glColor3f(0.0, 0.0, 0.0);

build();

glTranslatef(-330, -140, -130);

glColor3f(red);

glTranslatef(30, 25, 420);

glutSolidSphere(16.0, 10, 10);//哑铃球

glTranslatef(-30, -25, -420);

glColor3f(red);

glTranslatef(30, 25, 380);

glutSolidSphere(16.0, 10, 10);//哑铃球

glTranslatef(-30, -25, -380);

constract(30, 25, 380, 5, 5, 40);//哑铃棍

glColor3f(0.0, 0.0, 0.0);

build();

glTranslatef(330, 140, 140);

glutSolidTeapot(1);

glFlush();

glTranslatef(-330, -140, -140);

constract(350, 400, 10, 10, 10, 5);//墙画1

glColor3f(red);

build();

constract(330, 415, 10, 10, 10, 5);//墙画2

glColor3f(red);

build();

constract(310, 400, 10, 10, 10, 5);//墙画3

glColor3f(red);

build();

constract(315, 380, 10, 10, 10, 5);//墙画4

glColor3f(red);

build();

constract(330, 360, 10, 10, 10, 5);//墙画5

glColor3f(red);

build();

constract(350, 350, 10, 10, 10, 5);//墙画6

glColor3f(red);

build();

constract(370, 415, 10, 10, 10, 5);//墙画8

glColor3f(red);

build();

constract(390, 400, 10, 10, 10, 5);//墙画9

glColor3f(red);

build();

constract(385, 380, 10, 10, 10, 5);//墙画10

glColor3f(red);

build();

constract(370, 360, 10, 10, 10, 5);//墙画11

glColor3f(red);

build();

glColor3f(hgray);

for (int i = 100; i < 300; i += 25)

{

constract(200, 320, i, 5, 30, 6);

build(); //床栏杆

}

for (int i = 25; i < 201; i += 25)

{

constract(i, 320, 300, 6, 30, 6);

build(); //床栏杆

}

for (int i = 100; i < 300; i += 1)

{

constract(200, 350, i, 6, 5, 6);

build(); //床栏杆

}

constract(0, 350, 300, 200, 5, 6);

glColor3f(hgray);

build();//床栏杆

constract(10, 400, 350, 2, 60, 80);//表

glColor3f(1.0,0.65,0);

build();

constract(12, 430, 390, 1, 1, 20);//时针

glColor3f(0, 0, 0);

build();

constract(12, 430, 390, 1, 26, 1);//分针

glColor3f(0, 0, 0);

build();

constract(0, 500, 0, 500, 10, 500);//棚顶

glColor4f(blue,0.25);

build();

constract(500, 0, 0, 10, 500, 500);//右墙

glColor4f(1.0, 1.0, 1.0, 0.3);

build();

constract(0, 0, 500, 500, 500, 10);//后墙

glColor4f(blue,0.25);

build();

glFlush(); // 刷新

glutSwapBuffers();//双缓冲

}

void init(void)

{

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*设置光照（422-438行）\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

GLfloat sun\_direction[] = { 250.0, 500.0, 250.0, 0.0 };//发光位置

GLfloat sun\_intensity[] = { 1.0, 1.0, 1.0, 1.0 };//强度

GLfloat ambient\_intensity[] = { 0.8, 0.5, 0.5, 1.0 };//颜色比例偏红

glEnable(GL\_LIGHTING);

glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, ambient\_intensity);

glEnable(GL\_LIGHT0);

glLightfv(GL\_LIGHT0, GL\_POSITION, sun\_direction);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, sun\_intensity);//漫射光

glLightfv(GL\_LIGHT0, GL\_SPECULAR, sun\_intensity);//镜面反光

glEnable(GL\_COLOR\_MATERIAL);

glColorMaterial(GL\_FRONT , GL\_AMBIENT\_AND\_DIFFUSE);//材质

cout << "The OpenGL version is: " << glGetString(GL\_VERSION) << "\n";

cout << glGetString(GL\_VENDOR) << "\n";

glLineWidth(5);

glClearColor(0.0, 0.0, 0.0, 0.0); //定义背景颜色为黑色

glEnable(GL\_BLEND); // 打开混合

glDisable(GL\_DEPTH\_TEST); // 关闭深度测试

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE); // 基于源象素alpha通道值的半透明混合函数

glMatrixMode(GL\_PROJECTION);

glOrtho(-w, w, -h, h, -w, w);

}

//\*\*\*\*\*\*\*\*\*\*定义键盘上下左右控制（449-459行）\*\*\*\*\*\*\*\*\*\*

void specialkeys(int key, int x, int y) {

if (key == GLUT\_KEY\_RIGHT)

rotate\_y += 5;

else if (key == GLUT\_KEY\_LEFT)

rotate\_y -= 5;

else if (key == GLUT\_KEY\_UP)

rotate\_x += 5;

else if (key == GLUT\_KEY\_DOWN)

rotate\_x -= 5;

glutPostRedisplay();

}

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

{

glMatrixMode(GL\_MODELVIEW);

glMatrixMode(GL\_PROJECTION);

switch (key)

{

case 'a': case 'A': glTranslated(-10, 0, 0); break; //向左平移

case 'd': case 'D': glTranslated(10, 0, 0); break; //向右平移

case 'w': case 'W': glTranslated(0, 10, 0); break; //向上平移

case 's': case 'S': glTranslated(0, -10, 0); break; //向下平移

case 'q': case 'Q': glTranslated(0, 0, 100); break; //向外平移

case 'e': case 'E': glTranslated(0, 0, -100); break; //向里平移

case 'j': case 'J': glRotated(10, 0, 1, 0); break; //绕y轴旋转10度

case 'l': case 'L': glRotated(10, 0, -1, 0); break;

case 'i': case 'I': glRotated(10, 1, 0, 0); break; //绕x轴旋转10度

case 'k': case 'K': glRotated(10, -1, 0, 0); break;

case 'u': case 'U': glRotated(10, 0, 0, 1); break; //绕z轴旋转10度

case 'o': case 'O': glRotated(10, 0, 0, -1); break;

case 'z': case 'Z': glScalef(1.5, 1.5, 1.5); break; //放大

case 'x': case 'X': glScalef(0.5, 0.5, 0.5); break; //缩小

default: break;

}

glutPostRedisplay(); //重新绘制

}

int main(int argc, char\*\* argv) // main函数 增加鼠标和键盘事件

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(w, h); //定义窗口大小

glutInitWindowPosition(100, 100); //窗口打开位置

glutCreateWindow("《温馨小窝》");

init();

glutKeyboardFunc(myKeyboard);

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);

glEnable(GL\_DEPTH\_TEST);

glutDisplayFunc(display);

glutSpecialFunc(specialkeys);

glutMainLoop();

return 0;

}