代码附录

1.多边形代码

#pragma comment(lib,"OpenGL32.lib")

#include <GLFW/glfw3.h>

#include <math.h>

void myDisplay1(void)

{

glClearColor(0.0, 0.0, 0.0, 0.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

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

int sides = 6; // 多边形的边数

float radius = 0.5; // 多边形的半径

glBegin(GL\_POLYGON);

glColor3f(1.0, 0.0, 0.0);

for (int i = 0; i < sides; i++)

{

float theta = 2.0 \* 3.1415926 \* float(i) / float(sides);

float x = radius \* cos(theta);

float y = radius \* sin(theta);

glVertex2f(x, y);

}

glEnd();

glFlush();

}

void myDisplay2(void)

{

glClearColor(0.0, 0.0, 0.0, 0.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

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

glBegin(GL\_POLYGON);

// 绘制梯形的四个顶点

glColor3f(1.0, 0.0, 0.0);

glVertex2f(-0.5, -0.5);

glVertex2f(-0.2, -0.5);

glVertex2f(0.2, 0.5);

glVertex2f(-0.7, 0.5);

glEnd();

glFlush();

}

void myDisplay(void)

{

glClearColor(0.0, 0.0, 0.0, 0.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

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

glBegin(GL\_TRIANGLE\_FAN);

// 绘制凹凸多边形的顶点

glColor3f(1.0, 0.0, 0.0);

glVertex2f(-0.5, -0.5);

glVertex2f(-0.2, -0.2);

glVertex2f(0.5, -0.2);

glVertex2f(0.2, 0.2);

glVertex2f(0.5, 0.5);

glVertex2f(-0.5, 0.5);

glEnd();

glFlush();

}

int main()

{

// 初始化GLFW

if (!glfwInit())

{

return -1;

}

// 创建窗口

GLFWwindow\* window = glfwCreateWindow(800, 600, "OpenGL Window", NULL, NULL);

if (!window)

{

glfwTerminate();

return -1;

}

// 设置当前窗口上下文

glfwMakeContextCurrent(window);

// 设置垂直同步

glfwSwapInterval(1);

// 主循环

while (!glfwWindowShouldClose(window))

{

// 处理输入事件

glfwPollEvents();

// 渲染指令

myDisplay();

// 交换缓冲区

glfwSwapBuffers(window);

}

// 终止GLFW

glfwTerminate();

return 0;

}

2.缩放图形代码

#include <stdio.h>

#include <Windows.h>

#include <gl/glut.h>

#define OUTPUT\_MODE 1

float scale = 0.0;

bool add = true;

unsigned char axle = 'x';

void init(void)

{

//glClearColor函数设置好清除颜色，glClear利用glClearColor函数设置好的当前清除颜色设置窗口颜色

glClearColor(1.0, 1.0, 0.8, 1.0);

}

void display(void)

{

printf("scale=%f, axle=%c\n", scale, axle);

glClear(GL\_COLOR\_BUFFER\_BIT);

//配置缩放比例大小scale

scale = (add ? scale + 0.1 : scale - 0.1);

if (scale >= 2.0)

{

add = false;

}

else if (scale <= 0.0) {

add = true;

}

//开始绘画

glPushMatrix();

{

if (axle == 'x') {

glScalef(scale, 1, 1);

}

else if (axle == 'y') {

glScalef(1, scale, 1);

}

else if (axle == 'z') {

glScalef(1, 1, scale);

}

else {

glScalef(scale, scale, scale);

}

glColor3f(1.0, 0.0, 1.0); //画笔梅红色

glBegin(GL\_POLYGON);

glVertex2f(-0.2, -0.2);

glVertex2f(-0.2, 0.2);

glVertex2f(0.2, 0.2);

glVertex2f(0.2, -0.2);

glEnd();

}

glPopMatrix();

glLoadIdentity(); //加载单位矩阵

glColor3f(0.0, 0.0, 1.0); //画笔蓝色

//--------画直线START--------

//画直线

glBegin(GL\_LINES);

glVertex2f(-0.5, 0);

glVertex2f(0.5, 0);

glVertex2f(0, 0.5);

glVertex2f(0, -0.5);

glEnd();

//--------画直线E N D--------

if (OUTPUT\_MODE == 0) {

glFlush();

}

else {

glutSwapBuffers();

}

Sleep(50);

}

void reshape(int w, int h)

{

int offset = 50;

int dis = (w > h ? h : w) - offset \* 2;

//配置显示物体屏幕的大小

glViewport(offset, offset, (GLsizei)dis, (GLsizei)dis);

printf("reshape: w=%d, h=%d, dis=%d\n", w, h, dis);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-1.5, 1.5, -1.5, 1.5, 0, 10);

//gluOrtho2D(-1.5, 1.5, -1.5, 1.5);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

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

{

switch (key) {

case 'x':

case 'X':

axle = 'x';

glutPostRedisplay();

break;

case 'y':

case 'Y':

axle = 'y';

glutPostRedisplay();

break;

case 'z':

case 'Z':

axle = 'z';

glutPostRedisplay();

break;

case 'a':

case 'A':

axle = 'a';

glutPostRedisplay();

break;

default:

break;

}

printf("按键%c\n", key);

}

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

{

printf("可通过按键xyza控制图形按哪一轴缩放\n");

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | (OUTPUT\_MODE == 0 ? GLUT\_SINGLE : GLUT\_DOUBLE));

glutInitWindowPosition(100, 100);

glutInitWindowSize(400, 400);

glutCreateWindow("第一个 OpenGL 程序");

init();

glutDisplayFunc(&display);

glutIdleFunc(display); //设置不断调用显示函数

glutReshapeFunc(reshape);

glutKeyboardFunc(keyboard);

glutMainLoop();

return 0;

}

3.旋转图形代码

#include <stdio.h>

#include <gl/glut.h>

/\*

功能描述：使用OpenGL简单画一个旋转的三角形

旋转变换函数glRotatef

\*/

#define OUTPUT\_MODE 1

//角度

int angle = 0;

int xyz[3] = { 0, 0, 0 };

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glLoadIdentity();

glRotatef(angle, xyz[0], xyz[1], xyz[2]);

glBegin(GL\_TRIANGLES);

{

glVertex3f(0.0f, 0.0f, 0.0f);

glVertex3f(0.5f, 0.0f, 0.0f);

glVertex3f(0.0f, 0.5f, 0.0f);

}

glEnd();

angle += 1;

angle %= 360;

if (OUTPUT\_MODE == 0) {

glFlush();

}

else {

glutSwapBuffers();

}

}

//按键

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

{

switch (key)

{

case 'x':

case 'X':

xyz[0] = 1;

xyz[1] = xyz[2] = 0;

break;

case 'y':

case 'Y':

xyz[1] = 1;

xyz[0] = xyz[2] = 0;

break;

case 'z':

case 'Z':

xyz[2] = 1;

xyz[0] = xyz[1] = 0;

break;

case 'a':

case 'A':

xyz[0] = xyz[1] = xyz[2] = 0;

break;

case 'b':

case 'B':

xyz[0] = xyz[1] = xyz[2] = 1;

break;

default:

break;

}

printf("当前绕%c轴旋转\n", key);

glutPostRedisplay();

}

void reshape(int w, int h)

{

int offset = 50;

int dis = (w > h ? h : w) - offset \* 2;

//配置显示物体屏幕的大小

glViewport(offset, offset, (GLsizei)dis, (GLsizei)dis);

printf("reshape: w=%d, h=%d, dis=%d\n", w, h, dis);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-1.5, 1.5, -1.5, 1.5, 0, 10);

//gluOrtho2D(-1.5, 1.5, -1.5, 1.5);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

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

{

printf("可通过按键xyzab控制图形旋转原点/轴\n");

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | (OUTPUT\_MODE == 0 ? GLUT\_SINGLE : GLUT\_DOUBLE));

glutInitWindowPosition(100, 100);

glutInitWindowSize(400, 400);

glutCreateWindow("第一个 OpenGL 程序");

glutDisplayFunc(&display);

glutIdleFunc(display); //设置不断调用显示函数

glutReshapeFunc(reshape);

glutKeyboardFunc(&keyboard);

glutMainLoop();

return 0;

}

4.茶壶代码

#include <GL/glut.h>

void init(void)

{

glClearColor(0.0, 0.0, 0.0, 0.0);

glMatrixMode(GL\_PROJECTION);

glOrtho(-5, 5, -5, 5, 5, 15);

glMatrixMode(GL\_MODELVIEW);

gluLookAt(0, 0, 10, 0, 0, 0, 0, 1, 0);

return;

}

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 0, 0);

glutWireTeapot(3);

glFlush();

return;

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_SINGLE);

glutInitWindowPosition(0, 0);

glutInitWindowSize(300, 300);

glutCreateWindow("OpenGL 3D View");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

添加光照的代码

#include <GL/glut.h>

#include <stdlib.h>

//自定义初始化opengl函数

void init(void)

{

//材质反光性设置

GLfloat mat\_specular[] = { 1.0, 1.0, 1.0, 1.0 }; //镜面反射参数

GLfloat mat\_shininess[] = { 50.0 }; //高光指数

GLfloat light\_position[] = { 1.0, 1.0, 1.0, 0.0 };

GLfloat white\_light[] = { 1.0, 1.0, 1.0, 1.0 }; //灯位置(1,1,1), 最后1-开关

GLfloat Light\_Model\_Ambient[] = { 0.2, 0.2, 0.2, 1.0 }; //环境光参数

glClearColor(0.0, 0.0, 0.0, 0.0); //背景色

glShadeModel(GL\_SMOOTH); //多变性填充模式

//材质属性

glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);

glMaterialfv(GL\_FRONT, GL\_SHININESS, mat\_shininess);

//灯光设置

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, white\_light); //散射光属性

glLightfv(GL\_LIGHT0, GL\_SPECULAR, white\_light); //镜面反射光

glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, Light\_Model\_Ambient); //环境光参数

glEnable(GL\_LIGHTING); //开关:使用光

glEnable(GL\_LIGHT0); //打开0#灯

glEnable(GL\_DEPTH\_TEST); //打开深度测试

}

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glutSolidTeapot(0.5);

/\*

glBegin(GL\_QUADS);

glVertex3f(0, 0, 10);

glVertex3f(0, 0, 10);

glVertex3f(20, 5, 10);

glVertex3f(30, 40, -10);

glEnd();

\*/

glFlush(); //glSwapBuffers();

}

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

//设置投影参数

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

//正交投影

if (w <= h)

glOrtho(-1.5, 1.5, -1.5 \* (GLfloat)h / (GLfloat)w, 1.5 \* (GLfloat)h / (GLfloat)w, -10.0, 10.0);

else

glOrtho(-1.5 \* (GLfloat)w / (GLfloat)h, 1.5 \* (GLfloat)w / (GLfloat)h, -1.5, 1.5, -10.0, 10.0);

//设置模型参数--几何体参数

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize(500, 500);

glutInitWindowPosition(100, 100);

glutCreateWindow("茶壶");

init();

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutMainLoop();

return 0;

}