**计算机图形学**

**实验报告**

**exp4**

**目录**

[**Task1** 1](#_Toc509145986)

[功能说明： 1](#_Toc509145987)

[效果预览： 1](#_Toc509145988)

[源代码： 1](#_Toc509145989)

[**Task2** 7](#_Toc509145990)

[功能说明： 7](#_Toc509145991)

[效果预览： 7](#_Toc509145992)

[源代码： 7](#_Toc509145993)

**Task1**

功能说明：

实现橡皮条画直线。

1. 添加菜单，使得可以改变线的颜色、线型、线宽等属性。

2. 允许画多条直线，前面画过的直线不会消失

效果预览：

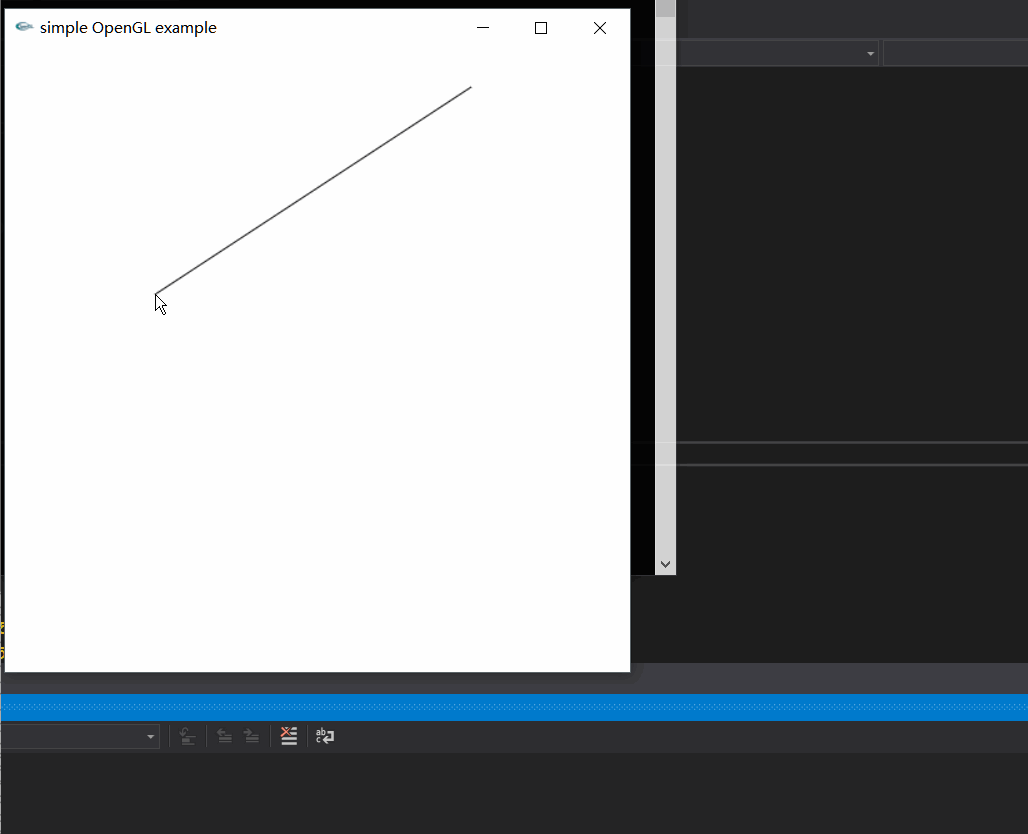


图1.1 Task1程序预览（查看附件动图）

源代码：

/\*

\* 实现橡皮条画直线

\* 1. 添加菜单，使得可以改变线的颜色、线型、线宽等属性。

\* 2. 允许画多条直线，前面画过的直线不会消失

\*\*\*\*\*\*\*

\* @author：宋灵冰

\*/

#include<vector>

#include<iostream>

#include <gl/glut.h>

using namespace std;

int height = 500;

int width = 500;

GLfloat xMax = 500;

GLfloat yMax = 500;

int Pattern[7] =

{

//直线

0xFFFF,

//点线 1000100010001000, 表示实际画线的点，反序后转换成16进制就是0x1111 dotted

//. . . . . . . . . . . . . .

//0x1111,

0x0101,

//点划线 1111111111100100 dot dash

//\_\_\_\_ . \_\_\_\_ . \_\_\_\_\_ . \_\_\_\_\_. \_\_\_\_\_

0x27FF,

//0x1C47,

//中心线 1111111111001100 centre line

//\_\_\_\_\_ \_ \_\_\_\_\_ \_ \_\_\_\_\_ \_ \_\_\_\_\_ \_ \_\_\_\_\_

0x33FF,

//虚线 1111110011111100 dashed

//\_\_\_\_ \_\_\_\_ \_\_\_\_ \_\_\_\_ \_\_\_\_ \_\_\_\_ \_\_\_\_

0x3F3F,

//双点划线 1111111100100100 double dot dash

// \_\_\_\_ . . \_\_\_\_ . . \_\_\_\_ . . \_\_\_\_ . . \_\_\_\_

0x24FF,

//三点划线 111111110101010 tri\_dot\_dash

// \_\_\_\_ . . \_\_\_\_ . . \_\_\_\_ . . \_\_\_\_ . . \_\_\_\_

0x55FF

};

typedef struct {

GLint x, y;

}Point2i;

vector<Point2i> pointVector;

Point2i curPoint;

GLfloat color[3] = { 0,0,0 };

int style = 0;

float lineWidth = 1.0;

void changeColor() {

color[0] = rand() % 255 / 255.0;

color[1] = rand() % 255 / 255.0;

color[2] = rand() % 255 / 255.0;

}

void changeStyle() {

style = (style + 1) % 7;

}

void addLineWidth() {

lineWidth += 1;

}

void minusLineWidth() {

if(lineWidth > 1)

lineWidth -= 1;

}

void menuFunc(int value) {

switch (value)

{

case 1:

changeColor();

break;

case 2:

changeStyle();

break;

case 3:

addLineWidth();

break;

case 4:

minusLineWidth();

break;

default:

break;

}

}

//设置初始属性

void initial() {

//设置背景为白色

glClearColor(1.0, 1.0, 1.0, 1.0);

//启用反走样

glEnable(GL\_BLEND);

glEnable(GL\_LINE\_SMOOTH);

glHint(GL\_LINE\_SMOOTH\_HINT, GL\_FASTEST);

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

//启用虚线

glEnable(GL\_LINE\_STIPPLE);

//设置观察参数

glMatrixMode(GL\_MODELVIEW);

//初始化坐标系

glLoadIdentity();

gluOrtho2D(0, 500.0, 0, 500.0);

glMatrixMode(GL\_MODELVIEW);

glShadeModel(GL\_FLAT);

glutCreateMenu(menuFunc);

glutAddMenuEntry("Change Color", 1);

glutAddMenuEntry("Change Style", 2);

glutAddMenuEntry("Add Line Width", 3);

glutAddMenuEntry("Minus Line Width", 4);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

}

//用于初始显示图像

void displayScene() {

//清空窗口

glClear(GL\_COLOR\_BUFFER\_BIT);

glLineStipple(1, Pattern[style]);

glLineWidth(lineWidth);

glColor3fv(color);

glBegin(GL\_LINE\_STRIP);

for (vector<Point2i>::iterator position = pointVector.begin();

position != pointVector.end(); position++) {

Point2i fixedPoint = \*position;

glVertex2d(fixedPoint.x, fixedPoint.y);

}

glVertex2d(curPoint.x, curPoint.y);

glEnd();

glFlush();

//交换通道

glutSwapBuffers();

}

//实现移动回调函数，使用鼠标控制移动

void drawFunc(int button, int state, int x, int y) {

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_UP) {

Point2i newPoint{ ((int)(x / (GLfloat)width \* xMax)), ((int)((height - y) / (GLfloat)height \* yMax)) };

pointVector.push\_back(newPoint);

}

glutPostRedisplay();

}

void slideFunc(int x, int y) {

curPoint = Point2i{ ((int)(x / (GLfloat)width \* xMax)), ((int)((height - y) / (GLfloat)height \* yMax)) };

glutPostRedisplay();

}

void reshape(int w, int h) {

glMatrixMode(GL\_MODELVIEW);

glViewport(0, 0, w, h);

width = w;

height = h;

GLfloat wScale = (GLfloat)width / (GLfloat)w;

GLfloat hScale = (GLfloat)height / (GLfloat)h;

//初始化坐标系

glLoadIdentity();

if (w <= h) {

yMax = 500.0\*(GLfloat)h / (GLfloat)w;

gluOrtho2D(0, 500,

0, yMax);

}

else {

xMax = 500.0\*(GLfloat)w / (GLfloat)h;

gluOrtho2D(0, 500.0\*(GLfloat)w / (GLfloat)h,

0, 500);

}

glutPostRedisplay();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(width, height);

glutInitWindowPosition(100, 100);

glutCreateWindow("simple OpenGL example");

initial();

glutMouseFunc(drawFunc);

glutPassiveMotionFunc(slideFunc);

glutDisplayFunc(displayScene);

glutReshapeFunc(reshape);

glutMainLoop();

return 0;

}

**Task2**

功能说明：

学习ply文件格式，读入文件内容，将模型显示出来，要求

1. 添加菜单，允许选择画点、画线、或者画多边形着色。

2. 设置鼠标左键、中键回调函数，使其分别实现对模型进行旋转、平移。

3 由于右键已被菜单占用，故缩放功能实用[]键完成，注意切换为英文模式

4 设置光照

效果预览：

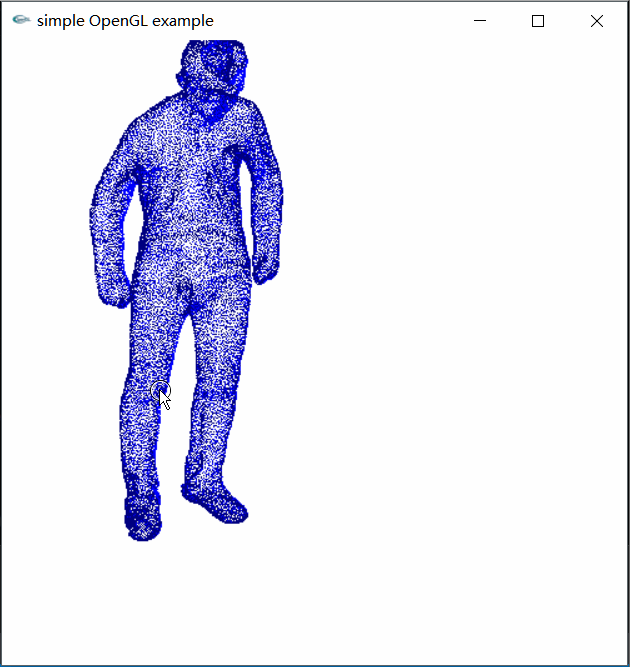


图2.1 Task2 程序预览（查看附件动图）

源代码：

/\*

\* 该程序实现以下功能

\* 读入ply文件内容，将模型显示出来

\* 添加菜单，允许选择画点、画线、或者画多边形着色。

\* 设置鼠标左键、中键回调函数，使其分别实现对模型进行旋转、平移。

\* 由于右键已被菜单占用，故缩放功能实用[]键完成，注意切换为英文模式

\* 设置光照。

\*\*\*\*\*\*\*

\* @author：宋灵冰

\*/

#include <iostream>

#include <vector>

#include <string.h>

#include <stdio.h>

#include <gl/glut.h>

using namespace std;

typedef struct {

GLdouble x, y, z;

}Point3d;

typedef struct {

int index1, index2, index3;

}FaceIndex;

enum mouseButton{LEFT\_CLICK, MIDDLE\_CLICK, NO\_CLICK};

GLfloat angleX = 0.0;

GLfloat angleY = 0.0;

GLfloat scaleLevel = 1.0;

GLfloat moveX = 0.0;

GLfloat moveY = 0.0;

const GLfloat moveZ = 0.0;

GLfloat tempX = 0.0;

GLfloat tempY = 0.0;

int orgX, orgY;

bool moveState = false;

int pointNum, faceNum;

vector<Point3d> pointContainer, normalContainer;

vector<FaceIndex> faceContainer;

GLenum mode = GL\_POINT;

mouseButton mouseMode;

void readPly(const char\* fileName) {

FILE\* f = fopen(fileName, "r");

if (!f)

return;

char strBuffer[255];

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

fgets(strBuffer, 255, f);

memset(strBuffer, '\0', 255);

}

fscanf(f, "element vertex %d\n", &pointNum);

for (int i = 0; i < 6; i++) {

fgets(strBuffer, 255, f);

memset(strBuffer, '\0', 255);

}

fscanf(f, "element face %d\n", &faceNum);

for (int i = 0; i < 2; i++) {

fgets(strBuffer, 255, f);

memset(strBuffer, '\0', 255);

}

Point3d tempPoint;

Point3d tempNormal;

FaceIndex tempFace;

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

fscanf(f, "%lf %lf %lf %lf %lf %lf\n",

&tempPoint.x, &tempPoint.y, &tempPoint.z,

&tempNormal.x, &tempNormal.y, &tempNormal.z);

pointContainer.push\_back(tempPoint);

normalContainer.push\_back(tempNormal);

}

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

fscanf(f, "3 %d %d %d\n",

&tempFace.index1, &tempFace.index2, &tempFace.index3);

faceContainer.push\_back(tempFace);

}

fclose(f);

}

void menuFunc(int value) {

switch (value)

{

case 1:

mode = GL\_POINT;

break;

case 2:

mode = GL\_LINE;

break;

case 3:

mode = GL\_FILL;

break;

default:

break;

}

glutPostRedisplay();

}

//设置初始属性

void myinit() {

//设置背景颜色

glClearColor(1.0, 1.0, 1.0, 1.0);

//设置观察参数

glMatrixMode(GL\_MODELVIEW);

glEnableClientState(GL\_NORMAL\_ARRAY);

//启用深度缓存

glEnable(GL\_DEPTH\_TEST);

//初始化坐标系

glLoadIdentity();

//设置当前相机的位置和方向

gluLookAt(

0.0, 0.0, 0.0,

0.0, 0.0, -1.0,

0.0, 1.0, 0.0);

glOrtho(-1000.0, 1000.0, -1000.0, 1000.0, -10000.0, 10000.0);

readPly("lizhenxiout-repaired.ply");

glutCreateMenu(menuFunc);

glutAddMenuEntry("Point Mode", 1);

glutAddMenuEntry("Line Mode", 2);

glutAddMenuEntry("Mesh Mode", 3);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

GLfloat ambient[] = { 0.3f, 0.3f, 0.3f, 1.0f }; //定义环境光颜色

GLfloat diffuse[] = { 0.8f, 0.8f, 0.8f, 1.0f }; //定义漫反射光颜色

GLfloat specular[] = { 0.0f, 0.0f, 0.0f, 1.0f }; //定义镜面反射光颜色

glLightfv(GL\_LIGHT0, GL\_AMBIENT, ambient);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, diffuse);

glLightfv(GL\_LIGHT1, GL\_SPECULAR, specular);

GLfloat lightPosition[] = { 0.0f, 0.0f, -500.0f, 1.0f }; //定义光源位置

glLightfv(GL\_LIGHT0, GL\_POSITION, lightPosition);

//开启灯光

glEnable(GL\_LIGHT0);

glEnable(GL\_LIGHTING);

glEnable(GLUT\_MULTISAMPLE);

glShadeModel(GL\_SMOOTH);

}

void drawModel() {

Point3d tempPoint;

Point3d tempVector;

FaceIndex tempFace;

Point3d tempFaceVertex[3];

Point3d tempFaceNormal[3];

GLfloat mat[] = { 0.0f, 0.0f, 1.0f, 1.0f }; //定义材质的反射光颜色

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_DIFFUSE, mat);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_AMBIENT, mat);

glPolygonMode(GL\_FRONT\_AND\_BACK, mode);

glBegin(GL\_TRIANGLES);

{

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

tempFace = faceContainer[i];

tempFaceVertex[0] = pointContainer[tempFace.index1];

tempFaceVertex[1] = pointContainer[tempFace.index2];

tempFaceVertex[2] = pointContainer[tempFace.index3];

tempFaceNormal[0] = normalContainer[tempFace.index1];

tempFaceNormal[1] = normalContainer[tempFace.index2];

tempFaceNormal[2] = normalContainer[tempFace.index3];

glNormal3d(tempFaceNormal[0].x, tempFaceNormal[0].y, tempFaceNormal[0].z);

glVertex3d(tempFaceVertex[0].x, tempFaceVertex[0].y, tempFaceVertex[0].z);

glNormal3d(tempFaceNormal[1].x, tempFaceNormal[1].y, tempFaceNormal[1].z);

glVertex3d(tempFaceVertex[1].x, tempFaceVertex[1].y, tempFaceVertex[1].z);

glNormal3d(tempFaceNormal[2].x, tempFaceNormal[2].y, tempFaceNormal[2].z);

glVertex3d(tempFaceVertex[2].x, tempFaceVertex[2].y, tempFaceVertex[2].z);

}

}

glEnd();

}

//用于初始显示图像

void displayScene() {

//清空窗口

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

glPushMatrix();

{

glTranslatef(moveX \* 3.0, moveY \* 3.0, moveZ);

glRotatef(angleX, 1.0, 0.0, 0.0);

glRotatef(angleY, 0.0, 1.0, 0.0);

glScalef(scaleLevel, scaleLevel, scaleLevel);

drawModel();

}

glPopMatrix();

glFlush();

//交换通道

glutSwapBuffers();

}

//实现缩放回调函数，使用[]控制缩放

void scaleFunc(unsigned char key, int x, int y) {

switch (key)

{

case '[':

if (scaleLevel > 0)

scaleLevel -= 0.1;

break;

case ']':

scaleLevel += 0.1;

break;

case 27:

exit(0);

break;

default:

break;

}

glutPostRedisplay();

}

//设置鼠标左键、中键回调函数，使其分别实现对模型进行旋转、平移

void mouseFunc(int button, int state, int x, int y) {

if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN && !moveState) {

mouseMode = LEFT\_CLICK;

orgX = x;

orgY = y;

tempX = angleY;

tempY = angleX;

moveState = true;

}

if (button == GLUT\_MIDDLE\_BUTTON && state == GLUT\_DOWN && !moveState) {

mouseMode = MIDDLE\_CLICK;

orgX = x;

orgY = y;

tempX = moveX;

tempY = moveY;

moveState = true;

}

if (state == GLUT\_UP && moveState) {

moveState = false;

mouseMode = NO\_CLICK;

}

glutPostRedisplay();

}

void moveFunc(int x, int y) {

switch (mouseMode) {

case LEFT\_CLICK:

angleY = tempX + (x - orgX);

angleX = tempY - (y - orgY);

break;

case MIDDLE\_CLICK:

moveX = tempX + x - orgX;

moveY = tempY - y + orgY;

break;

}

glutPostRedisplay();

}

void reshape(int w, int h) {

glMatrixMode(GL\_MODELVIEW);

glViewport(0, 0, w, h);

//初始化坐标系

glLoadIdentity();

if (w <= h)

glOrtho(-1000, 1000,

-1000.0\*(GLfloat)h / (GLfloat)w, 1000.0\*(GLfloat)h / (GLfloat)w,

-10000, 10000);

else

glOrtho(-1000.0\*(GLfloat)w / (GLfloat)h, 1000.0\*(GLfloat)w / (GLfloat)h,

-1000, 1000,

-10000, 10000);

glutPostRedisplay();

}

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

glutInit(&argc, argv);

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

glutInitWindowSize(500, 500);

glutInitWindowPosition(100, 100);

glutCreateWindow("simple OpenGL example");

myinit();

glutKeyboardFunc(scaleFunc);

glutMouseFunc(mouseFunc);

glutMotionFunc(moveFunc);

glutDisplayFunc(displayScene);

glutReshapeFunc(reshape);

glutMainLoop();

return 0;

}