实验八 二维直线段裁剪实验

时间：2022年5月11日

地点：信息学院机房

1、实验内容

指定矩形框，输入直线段端点并显示直线，采用Cohen-Sutherland算法对直线段进行裁剪，显示裁剪之后结果

2、实验目的

验证Cohen-Sutherland直线段裁剪算法

3、实验代码

|  |
| --- |
| #include<windows.h>  #include<iostream>  #include<gl\glut.h>  #include<utility>  #include<vector>  #include<math.h>  using namespace std;  const int screenWidth = 800;  const int screenHeight = 600;  const int windowPositionX = 300;  const int windowPositionY = 150;  class Point {  public:  double x, y;  Point() {}  Point(double x, double y) {  this->x = x;  this->y = y;  }  };  vector<Point> vertice; //顶点  //线段  vector<pair<Point, Point> > V;  //用于裁剪的窗口  class Window {  public:  int x, y;  int width;  int height;  int l, r, t, b; //边界：左右上下  Window(int x, int y, int width, int height) {  this->x = x;  this->y = y;  this->width = width;  this->height = height;  l = x;  r = x + width;  t = y + height;  b = y;  }  };  Window myWindow(250, 200, 300, 200);  //画裁剪窗口  void draw() {  glBegin(GL\_LINE\_LOOP);  glVertex2i(myWindow.l, myWindow.b);  glVertex2i(myWindow.l, myWindow.t);  glVertex2i(myWindow.r, myWindow.t);  glVertex2i(myWindow.r, myWindow.y);  glEnd();  }  //画点函数  void draw\_a\_point(int x, int y)  {  glBegin(GL\_POINTS);  glColor3f(0, 0, 0);  glVertex2f(x, y);  glEnd();  glFlush();  }  //判断位置  void chopLine(Point& p, unsigned char code, double dely, double delx) {  if (code & 1) { //0001 左方，位与运算，结果不为0时证明在窗口左方有交点  p.y += (myWindow.l - p.x) \* dely / delx;  p.x = myWindow.l;  }  else if (code & 2) { //0010 右方  p.y += (myWindow.r - p.x) \* dely / delx;  p.x = myWindow.r;  }  else if (code & 4) { //0100 下方  p.x += (myWindow.b - p.y) \* delx / dely;  p.y = myWindow.b;  }  else { //1000 上方  p.x += (myWindow.t - p.y) \* delx / dely;  p.y = myWindow.t;  }  }  //按位或，生成编号  void generateCode(Point& p, unsigned char& code) {  if (p.x < myWindow.l) code |= 1;  if (p.y > myWindow.t) code |= 8;  if (p.x > myWindow.r) code |= 2;  if (p.y < myWindow.b) code |= 4;  }  //裁剪  int cut(pair<Point, Point>& tmp) {  unsigned char code1;  unsigned char code2;  int k = 0;  do {  code1 = 0;  code2 = 0;  generateCode(tmp.first, code1);  generateCode(tmp.second, code2);  if ((code1 | code2) == 0) { //完全在窗口里面（0000|0000）  return 1;  }  else if ((code1 & code2) != 0) { //在某条边界同侧，即完全在窗口外面  return 0;  }  if (code1 != 0) {  chopLine(tmp.first, code1, tmp.second.y - tmp.first.y, tmp.second.x - tmp.first.x);  }  if (code2 != 0) {  chopLine(tmp.second, code2, tmp.second.y - tmp.first.y, tmp.second.x - tmp.first.x);  }  k++;  } while (1);  }  void func() {  pair<Point, Point> tmp(Point(vertice[0].x, vertice[0].y), Point(vertice[1].x, vertice[1].y));  V.push\_back(tmp);  glBegin(GL\_LINES);  glColor3f(0.0f, 0.0f, 0.0f);  glVertex2f(V[0].first.x, V[0].first.y);  glVertex2f(V[0].second.x, V[0].second.y);  glColor3f(1.0f, 0.0f, 0.0f);  int a = cut(V[0]);  if (a == 1) {  glVertex2f(V[0].first.x, V[0].first.y);  glVertex2f(V[0].second.x, V[0].second.y);  }  glEnd();  glFlush();  //V.pop\_back();  //ertice.pop\_back();  V.clear();  vertice.clear();  }  void mydisplay(void)  {  glClear(GL\_COLOR\_BUFFER\_BIT); //clear the screen  glColor3f(0.0f, 0.0f, 0.0f);  glLineWidth(3.0);  draw();  //func();  glFlush();  }  void mymouse(int button, int state, int x, int y)  {  if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN)  {  draw\_a\_point(x, screenHeight - y);  Point p(x, screenHeight - y);  vertice.push\_back(p);  cout << "起点" << ": (" << x << ", " << y << ")" << endl;  }  if (button == GLUT\_RIGHT\_BUTTON && state == GLUT\_DOWN)  {  draw\_a\_point(x, screenHeight - y);  Point p(x, screenHeight - y);  vertice.push\_back(p);  cout << "终点" << ": (" << x << ", " << y << ")" << endl;  cout << "开始剪裁" << endl;  func();  }  }  int main(int argc, char\*\* argv)  {  cout << "先点击鼠标左键画起点；" << endl << "再点击鼠标右键画终点并开始裁剪线段；" << endl;  glutInit(&argc, argv); //initialize the toolkit  glutInitDisplayMode(GLUT\_RGB | GLUT\_SINGLE);//t the display mode  glutInitWindowSize(screenWidth, screenHeight); //t the window size  glutInitWindowPosition(windowPositionX, windowPositionY); //t the window position  glutCreateWindow("Cohen-Sutherland 裁剪算法"); //open the screen window  glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  gluOrtho2D(0, screenWidth, 0, screenHeight);  glClearColor(1, 1, 1, 1);  glClear(GL\_COLOR\_BUFFER\_BIT);  glutDisplayFunc(&mydisplay); //register the callback functions  glutMouseFunc(&mymouse);  glutMainLoop(); //get into a perpetual loop  } |

4、实验总结

本次实验主要进行的直线的裁剪，使用的是Cohen-Sutherland算法。主要过程是对于每条线段P1P2分为三种情况处理:（1）若P1P2完全在窗口内，则显示该线段P1P2。（2）若P1P2明显在窗口外，则丢弃该线段。（3）若线段不满足（1）或（2）的条件，则在交点处把线段分为两段。其中一段完全在窗口外，可弃之。然后对另一段重复上述处理。