**实验名称：Cohen-Sutherland直线段裁剪**

1. **具体内容**

**利用Cohen-Sutherland算法进行直线段裁剪**

1. **设计思路**

**利用Cohen-Sutherland算法进行直线段裁剪**

1. **主要函数及其简要说明**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 编码函数，输入点坐标，输出4位编码

\* px，py为输入点 clip\_polygon为裁剪多边形 clip\_polygon\_point\_count为裁剪多边形的顶点数

量

\* 4位编码的意义如下（从左到右)：

\* 第1位：如果端点在裁剪多边形的左侧，则为1，否则为0

\* 第2位：如果端点在裁剪多边形的右侧，则为1，否则为0

\* 第3位：如果端点在裁剪多边形的下侧，则为1，否则为0

\* 第4位：如果端点在裁剪多边形的上侧，则为1，否则为0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unsigned int enCode(double px, double py)

{

//用1 2 4 8 编码裁剪框的左、右、下、上区域

unsigned int RC = 0;

if (px < clip\_polygon[0].x)

{

RC = RC | 1;

}

else if (px > clip\_polygon[1].x)

{

RC = RC | 2;

}

if (py < clip\_polygon[0].y)

{

RC = RC | 4;

}

else if (py > clip\_polygon[3].y)

{

RC = RC | 8;

}

return RC;

}

//Cohen-Sutherland 裁剪函数

//根据线段的两个端点 point1 和 point2 的编码判断，该线段与裁剪多边形的位置，求出裁剪后的两个端点

void cilpping\_lines(my\_homogeneous\_point point1, my\_homogeneous\_point point2)

{

my\_homogeneous\_point test\_point11 = point1;//用test\_point11、test\_point22储存直线两个端点

my\_homogeneous\_point test\_point22 = point2;

float x = 0; //储存变化后的x，y坐标

float y = 0;

int code1 = enCode(test\_point11.x, test\_point11.y); //code1、code2记录顶点的编码

int code2 = enCode(test\_point22.x, test\_point22.y);

int code = 0;

while (code1 != 0 || code2 != 0) //两直线至少有一点在区域外

{

if ((code1 & code2) != 0)

{ //两点连线不过裁剪区域,跳出循环

return;

}

//将在区域外的点编码给code

code = code1;

if (code1 == 0) code = code2;

if ((1 & code) != 0) //直线段与左边界相交，求出直线与边界相交点

{

x = clip\_polygon[0].x;

y = test\_point11.y + (test\_point22.y - test\_point11.y) \* (clip\_polygon[0].x -test\_point11.x) / (test\_point22.x - test\_point11.x);

}

else if ((2 & code) != 0) //直线段与右边界相交，求出直线与边界相交点

{

x = clip\_polygon[1].x;

y = test\_point11.y + (test\_point22.y - test\_point11.y) \* (clip\_polygon[1].x -test\_point11.x) / (test\_point22.x - test\_point11.x);

}

else if ((4 & code) != 0) //直线段与下边界相交，求出直线与边界相交点

{

y = clip\_polygon[0].y;

x = test\_point11.x + (test\_point22.x - test\_point11.x) \* (clip\_polygon[0].y -test\_point11.y) / (test\_point22.y - test\_point11.y);

}

else if ((8 & code) != 0) //直线段与上边界相交，求出直线与边界相交点

{

y = clip\_polygon[3].y;

x = test\_point11.x + (test\_point22.x - test\_point11.x) \* (clip\_polygon[3].y -test\_point11.y) / (test\_point22.y - test\_point11.y);

}

//将求出的点重新赋值

if (code == code1)

{

test\_point11.x = x;

test\_point11.y = y;

code1 = enCode(x, y);

}

else

{

test\_point22.x = x;

test\_point22.y = y;

code2 = enCode(x, y);

}

}

//将新生成的点存入List中

List.push\_back(test\_point11);

List.push\_back(test\_point22);

}

1. **遇到的主要问题及解决思路：**

**无**

1. **程序运行结果**

 