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 <stdlib.h>

#include <math.h>

#include <iostream>

#include<stdio.h>

#include <Windows.h>

#include <GL/GL.h>

#include <GL/glut.h>

//画线

void setPixel(GLint x,GLint y)

{

glBegin(GL\_POINTS);

glVertex2i(x,y);

glEnd();

}

void init(void)

{

glClearColor(1.0,1.0,1.0,0.0);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(-200,200.0,-200.0,200.0);

}

void LineDDA(int x0,int y0,int xEnd,int yEnd){

int dx=xEnd-x0, dy=yEnd-y0, steps, k;

float xIncrement,yIncrement,x=x0,y=y0;

if(fabs(dx)>fabs(dy))

steps=fabs(dx);

else

steps=fabs(dy);

xIncrement=float(dx)/float(steps);

yIncrement=float(dy)/float(steps);

setPixel(round(x),round(y));

for(k=0;k<steps;k++){

x+=xIncrement;

y+=yIncrement;

setPixel(round(x),round(y));

}

}

//裁剪

class wcPt2D

{

public:

GLfloat x,y;

};

const GLint winLeftBitCode = 0x1;

const GLint winRightBitCode = 0x2;

const GLint winBottomBitCode = 0x4;

const GLint winTopBitCode = 0x8;

inline GLint inside(GLint code)

{

return GLint(!code);

}

inline GLint reject(GLint code1,GLint code2)

{

return GLint(code1&code2);

}

inline GLint accept(GLint code1,GLint code2)

{

return GLint(!(code1|code2));

}

GLubyte encode(wcPt2D pt,wcPt2D winMin,wcPt2D winMax)

{

GLubyte code = 0x00;

if(pt.x<winMin.x)

code = code|winLeftBitCode;

if(pt.x>winMax.x)

code = code|winRightBitCode;

if(pt.y<winMin.y)

code = code|winBottomBitCode;

if(pt.y>winMax.y)

code = code|winTopBitCode;

return(code);

}

void swapPts(wcPt2D \*p1,wcPt2D \*p2)

{

wcPt2D tmp;

tmp = \*p1;

\*p1 = \*p2;

\*p2 = tmp;

}

void swapCodes(GLubyte \*c1,GLubyte \*c2)

{

GLubyte tmp;

tmp = \*c1;

\*c1 = \*c2;

\*c2 = tmp;

}

void lineClipCohSuth(wcPt2D winMin,wcPt2D winMax,wcPt2D p1,wcPt2D p2)

{

GLubyte code1,code2;

GLint done = false,plotLine = false;

GLfloat m;

while(!done)

{

code1 = encode(p1,winMin,winMax);

code2 = encode(p2,winMin,winMax);

if(accept(code1,code2))

{

done = true;

plotLine = true;

}//简取

else if(reject(code1,code2))//简弃

done = true;

else

{

if(inside(code1))

{

swapPts(&p1,&p2);

swapCodes(&code1,&code2);

}

if(p2.x!=p1.x)

m=(p2.y-p1.y)/(p2.x-p1.x);//计算k

if(code1 &winLeftBitCode)

{

p1.y+=(winMin.x-p1.x)\*m;

p1.x = winMin.x;

}

else if(code1& winRightBitCode){

p1.y += (winMax.x-p1.x)\*m;

p1.x = winMax.x;

}

else if(code1 &winBottomBitCode)

{

if(p2.x != p1.x)

p1.x+=(winMin.y - p1.y)/m;

p1.y=winMin.y;

}

else if(code1 &winTopBitCode)

{

if(p2.x != p1.x)

p1.x +=(winMax.y - p1.y)/m;

p1.y = winMax.y;

}

}//else

}//while

if(plotLine)

{

LineDDA(p1.x,p1.y,p2.x,p2.y);

}

}

void cutwindow(int xmin,int ymin,int xmax,int ymax)

{

LineDDA(xmin,ymin,xmin,ymax);

LineDDA(xmin,ymin,xmax,ymin);

LineDDA(xmin,ymax,xmax,ymax);

LineDDA(xmax,ymin,xmax,ymax);

}

void display() //DDA演示

{

glClear(GL\_COLOR\_BUFFER\_BIT);// 清空显示窗口

glColor3f(0.0,0.0,0.0);

wcPt2D min;

min.x = -100;//裁剪窗口左下角横坐标

min.y = -100;//裁剪窗口左下角纵坐标

wcPt2D max;

max.x=100;//裁剪窗口右上角横坐标

max.y=100;//裁剪窗口右上角纵坐标

wcPt2D p1;

p1.x=-80;//需要裁剪的直线的起点的横坐标

p1.y= 70;//需要裁剪的直线的起点的纵坐标

wcPt2D p2;

p2.x=110;//需要裁剪的直线的终点的横坐标

p2.y=-70;//需要裁剪的直线的终点的纵坐标

cutwindow(min.x,min.y,max.x,max.y);//绘制窗口

lineClipCohSuth(min,max,p1,p2);

glFlush();

}

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

{

glutInit(&argc,argv);// 初始 GLUT

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);//设定显示模式

glutInitWindowPosition(400,400);// 设定窗口位置

glutInitWindowSize(400,400);// 设定窗口大小

glutCreateWindow("直线裁剪");// 用前面指定参数创建窗口，并定义窗口名称

init();// 进行一些初始化工作

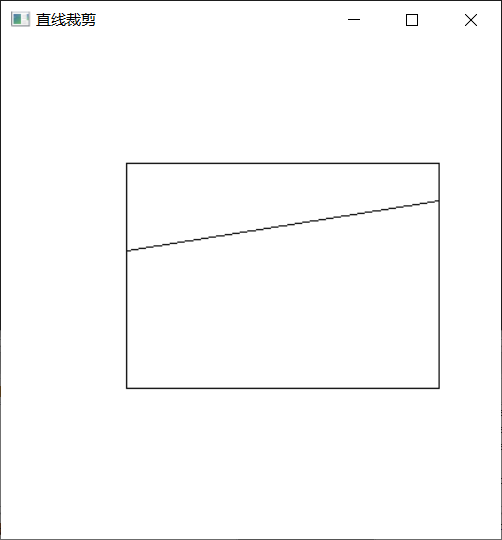
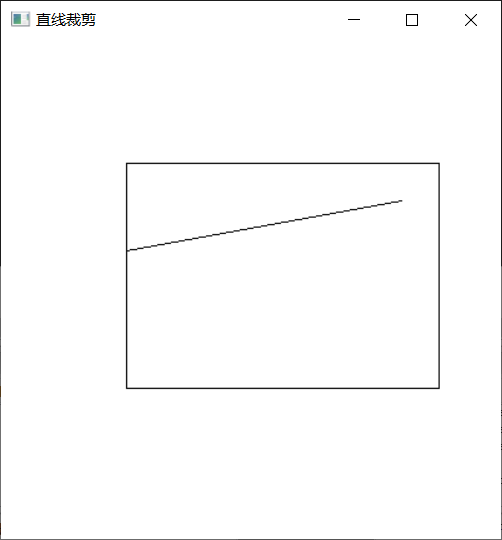
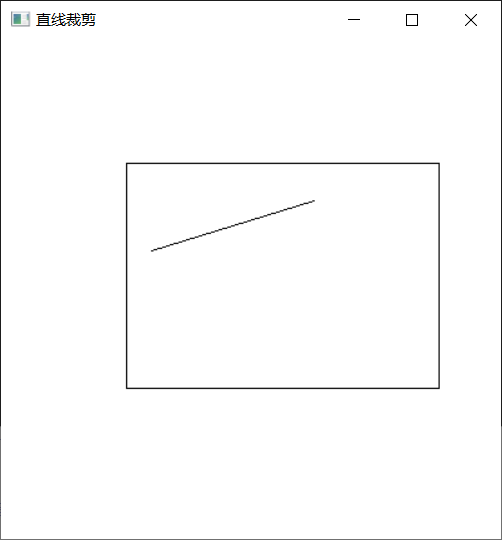
glutDisplayFunc(&display); // 指定绘制的回调函数

glutMainLoop();// 进入无穷循环，等待事件处理

return 0;

}

4、实验结果



5、实验总结

1.实验过程中使用了Sutherland编码算法，实验原理为先确定裁剪窗口并绘出来，裁剪窗口所在直线的延长线将窗口分为9个部分，9个部分分别用四位编码表示，通过四位编码的位于或者位或，快速确认线段是否在直线内，如果直线跨着裁剪窗口，则进行下一步的分析。

2.过程中调用了前面写过的DDA算法，在实验过程中，我遇见了编译没错但运行不出来的情况，检查了好长时间才发现是在setPixel和display函数里面都写了glBegin和glEnd，导致窗口显示不出来，这次的错误让我懂得了写代码需要学会统筹，不能几个函数里面都有相同的内容，否则很容易出问题。