**计算机图形学实验报告**

**学号：1004161221**

**姓名： 白云鹏\_\_**

**日期: 2018/12/21**

# 实验环境

|  |  |
| --- | --- |
| 操作系统 | Windows 10 64位 |
| 内存（可用/实际） | 7.78/8.00　GB |
| CPU | I5-7360U |

# Cohen-Sutherland线段裁剪算法

## 算法原理

其算法思想是用裁剪区域上下左右边界将线段分解为两部分，即可见区域和不可见区域，线段一共只有三种情况：

* 全部落在可见区域，全部保留
* 全部落在不可见区域，全部舍弃
* 部分落在可见区域，只保留这部分

然后按照左右下上的顺序对其进行裁剪，每次传递其落在窗口内或者与窗口相交的顶点。

## 源代码

class wcPt2D

{

public:

GLfloat x, y;

wcPt2D() { x = y = 0.0; }

wcPt2D(float a, float b)

{

x = a;

y = b;

}

void setCoords(GLfloat xCoord, GLfloat yCoord)

{

x = xCoord;

y = yCoord;

}

GLfloat getx() const { return x; }

GLfloat gety() const { return y; }

};

void drawLine(GLint x0, GLint y0, GLint x1, GLint y1)

{

glColor3f(1, 0, 0);

glBegin(GL\_LINES);

glVertex2i(x0, y0);

glVertex2i(x1, y1);

glEnd();

}

const GLint winLeftBitCode = 0x1;

const GLint winRightBitCode = 0x2;

const GLint winBottomBitCode = 0x4;

const GLint winTopBitCode = 0x8;

inline GLint round(const GLfloat a) { return GLint(a + 0.5); }

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.getx() < winMin.getx())

{

code = code | winLeftBitCode;

}

if (pt.getx() > winMax.getx())

{

code = code | winRightBitCode;

}

if (pt.gety() < winMin.gety())

{

code = code | winBottomBitCode;

}

if (pt.gety() > winMax.gety())

{

code = code | winTopBitCode;

}

return code;

}

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

{

wcPt2D tem;

tem = \*p1;

\*p1 = \*p2;

\*p2 = tem;

}

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

{

GLubyte tem;

tem = \*c1;

\*c1 = \*c2;

\*c2 = tem;

}

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;

}

else if (reject(code1, code2))

done = true;

else

{

if (inside(code1))

{

swapPts(&p1, &p2);

swapCodes(&code1, &code2);

}

if (p2.getx() != p1.getx())

{

m = (p2.gety() - p1.gety()) / (p2.getx() - p1.getx());

}

if (code1 & winLeftBitCode)

{

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

p1.x = winMax.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;

}

}

}

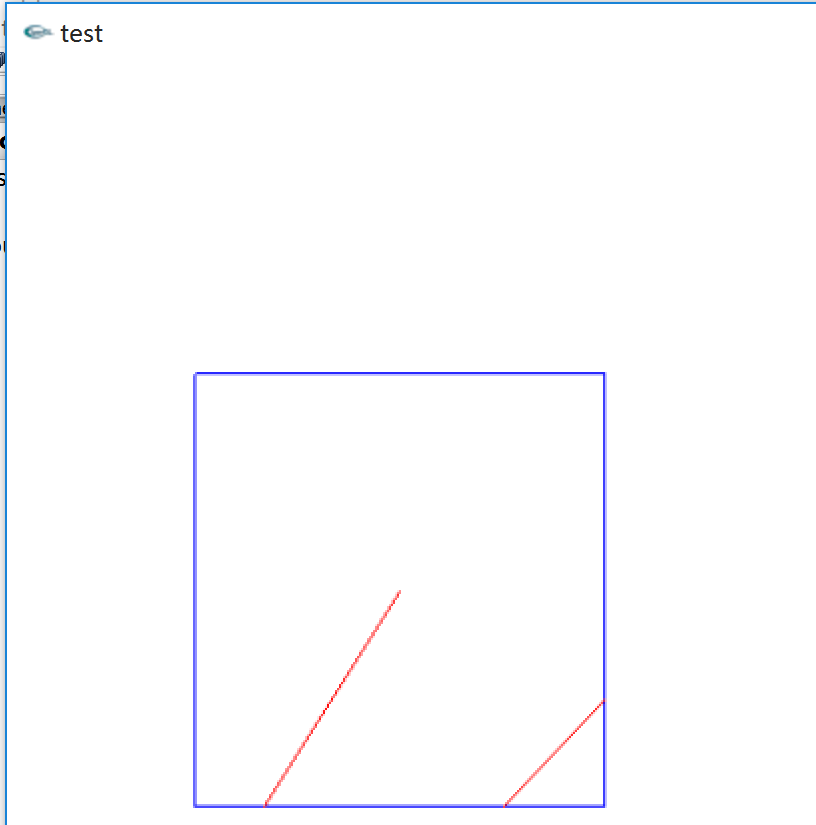
if(plotLine){

drawLine(round(p1.x),round(p1.y),round(p2.x),round(p2.y));

}

}

## 实验结果



# 梁友栋-Barsky线段裁剪算法

## 算法流程

## 源代码

class wcPt2D

{

public:

GLfloat x, y;

wcPt2D() { x = y = 0.0; }

wcPt2D(float a, float b)

{

x = a;

y = b;

}

void setCoords(GLfloat xCoord, GLfloat yCoord)

{

x = xCoord;

y = yCoord;

}

GLfloat getx() const { return x; }

GLfloat gety() const { return y; }

};

inline GLint round(const GLfloat a) { return GLint(a + 0.5); }

GLint clipTest(GLfloat p, GLfloat q, GLfloat \*u1, GLfloat \*u2)

{

GLfloat r;

GLint returnValue = true;

if (p < 0.0)

{

r = q / p;

if (r > \*u2)

returnValue = false;

else if (r > \*u1)

\*u1 = r;

}

else if (p > 0.0)

{

r = q / p;

if (r < \*u1)

returnValue = false;

else if (r < \*u2)

\*u2 = r;

}

else

{

if (q < 0.0)

returnValue = false;

}

return returnValue;

}

void drawLine(GLint x0, GLint y0, GLint x1, GLint y1)

{

glColor3f(1, 0, 0);

glBegin(GL\_LINES);

glVertex2i(x0, y0);

glVertex2i(x1, y1);

glEnd();

}

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

{

GLfloat u1 = 0.0, u2 = 1.0, dx = p2.getx() - p1.getx(), dy;

if (clipTest(-dx, p1.getx() - winMin.getx(), &u1, &u2))

{

if (clipTest(dx, winMax.getx() - p1.getx(), &u1, &u2))

{

dy = p2.gety() - p1.gety();

if (clipTest(-dy, p1.gety() - winMin.gety(), &u1, &u2))

if (clipTest(dy, winMax.gety() - p1.gety(), &u1, &u2))

{

if (u2 < 1.0)

{

p2.setCoords(p1.getx() + u2 \* dx, p1.gety() + u2 \* dy);

}

if (u1 > 0.0)

{

p1.setCoords(p1.getx() + u1 \* dx, p1.gety() + u1 \* dy);

}

drawLine(round(p1.getx()), round(p1.gety()), round(p2.getx()),

round(p2.gety()));

}

}

}

}

## 实验结果

## 梁