**// 直线段CS裁剪算法的响应函数**

void CLine\_ScanView::OnLineCs()

{

// TODO: 在此添加命令处理程序代码

Line\_Cut lc;

lc.DoModal();

int min\_clip\_x = 100 ,min\_clip\_y = 100 ,max\_clip\_x = 300,max\_clip\_y=300 ;

CClientDC dc(this);

dc.MoveTo(min\_clip\_x,min\_clip\_y);

dc.LineTo(min\_clip\_x,max\_clip\_y);

dc.MoveTo(min\_clip\_x,max\_clip\_y );

dc.LineTo(max\_clip\_x,max\_clip\_y);

dc.MoveTo(max\_clip\_x,max\_clip\_y );

dc.LineTo(max\_clip\_x,min\_clip\_y);

dc.MoveTo(max\_clip\_x,min\_clip\_y );

dc.LineTo(min\_clip\_x,min\_clip\_y);

if(CutLine(lc.x0,lc.y0,lc.x1,lc.y1)){

dc.MoveTo(lc.x0,lc.y0 );

dc.LineTo(lc.x1,lc.y1);

}

}

**// 直线段的CS裁剪算法**

int CLine\_ScanView::CutLine(int& x0, int& y0, int& x1, int& y1)

{

#define CLIP\_CODE\_C 0x0000

#define CLIP\_CODE\_N 0x0008

#define CLIP\_CODE\_S 0x0004

#define CLIP\_CODE\_E 0x0002

#define CLIP\_CODE\_W 0x0001

#define CLIP\_CODE\_NE 0x000a

#define CLIP\_CODE\_SE 0x0006

#define CLIP\_CODE\_NW 0x0009

#define CLIP\_CODE\_SW 0x0005

int xc0 = x0 ,yc0 = y0 , xc1=x1 , yc1=y1 ;

int min\_clip\_x = 100 ,min\_clip\_y = 100 ,max\_clip\_x = 300,max\_clip\_y=300 ;

int p0\_code = 0 ,p1\_code = 0 ;

//确定各个顶点所在的位置代码

if(y0<min\_clip\_y)

p0\_code|=CLIP\_CODE\_N;

else if(y0>max\_clip\_y)

p0\_code|=CLIP\_CODE\_S;

if(x0<min\_clip\_x)

p0\_code|=CLIP\_CODE\_W;

else if(x0>max\_clip\_x)

p0\_code|=CLIP\_CODE\_E;

if(y1<min\_clip\_y)

p1\_code|=CLIP\_CODE\_N;

else if(y1>max\_clip\_y)

p1\_code|=CLIP\_CODE\_S;

if(x1<min\_clip\_x)

p1\_code|=CLIP\_CODE\_W;

else if(x1>max\_clip\_x)

p1\_code|=CLIP\_CODE\_E;

//先检测一些简单的情况

if(p0\_code&p1\_code) //有相同的位置代码，表示在裁剪区外部

return 0 ;

if(p0\_code==0&&p1\_code==0) //表示两个点都在裁剪区内，不需要裁剪

return 1 ;

//判断第一个点的位置代码

switch(p0\_code)

{

case CLIP\_CODE\_C:

break;

case CLIP\_CODE\_N:

{

yc0 = min\_clip\_y ;

xc0 = x0 + 0.5 + (yc0-y0)\*(x1-x0)/(y1-y0);

break ;

}

case CLIP\_CODE\_S:

{

yc0 = max\_clip\_y;

xc0 = x0 + 0.5 + (yc0-y0)\*(x1-x0)/(y1-y0);

break ;

}

case CLIP\_CODE\_W:

{

xc0=min\_clip\_x;

yc0=y0+0.5+(xc0-x0)\*(y1-y0)/(x1-x0);

break;

}

case CLIP\_CODE\_E:

{

xc0=max\_clip\_x;

yc0=y0+0.5+(xc0-x0)\*(y1-y0)/(x1-x0);

break;

}

case CLIP\_CODE\_NE:

{

yc0 = min\_clip\_y;

xc0 = x0 + 0.5 + (yc0-y0)\*(x1-x0)/(y1-y0);

if(xc0<min\_clip\_x||xc0>max\_clip\_x)

{

xc0=max\_clip\_x;

yc0=y0+0.5+(xc0-x0)\*(y1-y0)/(x1-x0);

}

break;

}

case CLIP\_CODE\_SE:

{

yc0 = max\_clip\_y;

xc0 = x0 + 0.5 + (yc0-y0)\*(x1-x0)/(y1-y0);

if(xc0<min\_clip\_x||xc0>max\_clip\_x)

{

xc0=max\_clip\_x;

yc0=y0+0.5+(xc0-x0)\*(y1-y0)/(x1-x0);

}

break;

}

case CLIP\_CODE\_NW:

{

yc0=min\_clip\_y;

xc0 = x0 + 0.5 + (yc0-y0)\*(x1-x0)/(y1-y0);

if(xc0<min\_clip\_x||xc0>max\_clip\_x)

{

xc0=min\_clip\_x;

yc0=y0+0.5+(xc0-x0)\*(y1-y0)/(x1-x0);

}

break;

}

case CLIP\_CODE\_SW:

{

yc0=max\_clip\_y;

xc0 = x0 + 0.5 + (yc0-y0)\*(x1-x0)/(y1-y0);

if(xc0<min\_clip\_x||xc0>max\_clip\_x)

{

xc0=min\_clip\_x;

yc0=y0+0.5+(xc0-x0)\*(y1-y0)/(x1-x0);

}

break;

}

default:

break;

} // end switch(p0\_code)

//判断第二个点的位置代码

switch(p1\_code)

{

case CLIP\_CODE\_C:

break;

case CLIP\_CODE\_N:

{

yc1 = min\_clip\_y ;

xc1 = x1 + 0.5 + (yc1-y1)\*(x1-x0)/(y1-y0);

break ;

}

case CLIP\_CODE\_S:

{

yc1 = max\_clip\_y;

xc1 = x1 + 0.5 + (yc1-y1)\*(x1-x0)/(y1-y0);

break ;

}

case CLIP\_CODE\_W:

{

xc1=min\_clip\_x;

yc1=y1+0.5+(xc1-x1)\*(y1-y0)/(x1-x0);

break;

}

case CLIP\_CODE\_E:

{

xc1=max\_clip\_x;

yc1=y1+0.5+(xc1-x1)\*(y1-y0)/(x1-x0);

break;

}

case CLIP\_CODE\_NE:

{

yc1 = min\_clip\_y;

xc1 = x1 + 0.5 + (yc1-y1)\*(x1-x0)/(y1-y0);

if(xc1<min\_clip\_x||xc1>max\_clip\_x)

{

xc1=max\_clip\_x;

yc1=y1+0.5+(xc1-x1)\*(y1-y0)/(x1-x0);

}

break;

}

case CLIP\_CODE\_SE:

{

yc1 = max\_clip\_y;

xc1 = x1 + 0.5 + (yc1-y1)\*(x1-x0)/(y1-y0);

if(xc1<min\_clip\_x||xc1>max\_clip\_x)

{

xc1=max\_clip\_x;

yc1=y1+0.5+(xc1-x1)\*(y1-y0)/(x1-x0);

}

break;

}

case CLIP\_CODE\_NW:

{

yc1=min\_clip\_y;

xc1 = x1 + 0.5 + (yc1-y1)\*(x1-x0)/(y1-y0);

if(xc1<min\_clip\_x||xc1>max\_clip\_x)

{

xc1=min\_clip\_x;

yc1=y1+0.5+(xc1-x1)\*(y1-y0)/(x1-x0);

}

break;

}

case CLIP\_CODE\_SW:

{

yc1=max\_clip\_y;

xc1 = x1 + 0.5 + (yc1-y1)\*(x1-x0)/(y1-y0);

if(xc1<min\_clip\_x||xc1>max\_clip\_x)

{

xc1=min\_clip\_x;

yc1=y1+0.5+(xc1-x1)\*(y1-y0)/(x1-x0);

}

break;

}

default:

break;

}

//进行最后的检测

if(xc0>max\_clip\_x||xc0<min\_clip\_x||

yc0>max\_clip\_y||yc0<min\_clip\_y||

xc1>max\_clip\_x||xc1<min\_clip\_x||

yc1>max\_clip\_y||yc1<min\_clip\_y)

{

//表示全部在裁剪区外部

return 0 ;

}

//将裁减后的数据返回

x0 = xc0 ;

x1 = xc1 ;

y0 = yc0 ;

y1 = yc1 ;

return 1 ;

}

**// 直线段中点分割算法的响应函数**

void CLine\_ScanView::OnLineMidpoint()

{

// TODO: 在此添加命令处理程序代码

int x0,y0,x1,y1;

CClientDC dc(this);

int min\_clip\_x = 100 ,min\_clip\_y = 100 ,max\_clip\_x = 300,max\_clip\_y=300 ;

dc.MoveTo(min\_clip\_x,min\_clip\_y);

dc.LineTo(min\_clip\_x,max\_clip\_y);

dc.MoveTo(min\_clip\_x,max\_clip\_y );

dc.LineTo(max\_clip\_x,max\_clip\_y);

dc.MoveTo(max\_clip\_x,max\_clip\_y );

dc.LineTo(max\_clip\_x,min\_clip\_y);

dc.MoveTo(max\_clip\_x,min\_clip\_y );

dc.LineTo(min\_clip\_x,min\_clip\_y);

x0=120;y0=90;x1=310,y1=310;

Mid\_CutLine(x0,y0,x1,y1);

}

**// 直线段的中点分割算法**

void CLine\_ScanView::Mid\_CutLine(int x0, int y0, int x1, int y1)

{

#define CLIP\_CODE\_C 0x0000

#define CLIP\_CODE\_N 0x0008

#define CLIP\_CODE\_S 0x0004

#define CLIP\_CODE\_E 0x0002

#define CLIP\_CODE\_W 0x0001

#define CLIP\_CODE\_NE 0x000a

#define CLIP\_CODE\_SE 0x0006

#define CLIP\_CODE\_NW 0x0009

#define CLIP\_CODE\_SW 0x0005

int min\_clip\_x = 100 ,min\_clip\_y = 100 ,max\_clip\_x = 300,max\_clip\_y=300 ;

int p0\_code = 0 ,p1\_code = 0 ;

//确定各个顶点所在的位置代码

if(y0<min\_clip\_y)

p0\_code|=CLIP\_CODE\_N;

else if(y0>max\_clip\_y)

p0\_code|=CLIP\_CODE\_S;

if(x0<min\_clip\_x)

p0\_code|=CLIP\_CODE\_W;

else if(x0>max\_clip\_x)

p0\_code|=CLIP\_CODE\_E;

if(y1<min\_clip\_y)

p1\_code|=CLIP\_CODE\_N;

else if(y1>max\_clip\_y)

p1\_code|=CLIP\_CODE\_S;

if(x1<min\_clip\_x)

p1\_code|=CLIP\_CODE\_W;

else if(x1>max\_clip\_x)

p1\_code|=CLIP\_CODE\_E;

//先检测一些简单的情况

if(p0\_code&p1\_code) //有相同的位置代码，表示在裁剪区外部

{return ;}

else if(p0\_code==0&&p1\_code==0) //表示两个点都在裁剪区内，不需要裁剪

{

CClientDC dc(this);

dc.MoveTo(x0,y0 );

dc.LineTo(x1,y1);

return;

} else

{

int x2,y2;

x2=(x0+x1)/2;

y2=(y0+y1)/2;

if(x0-x2>1||x2-x0>1)

{

Mid\_CutLine(x0,y0,x2,y2);

}

if(x1-x2>1||x2-x1>1)

{

Mid\_CutLine(x2,y2,x1,y1);

}

}

return; }