《计算机图形学》实验报告

裁剪算法实验

姓名 王雨朦

学号 3012216083

专业 计算机科学与技术

班级 3 班

天津大学计算机科学与技术学院

2014 年 12 月 25 日

一、实验目的

1. 实现 Cohen-Sutherland 直线裁剪算法（选做）
2. 实现 Sutherland-Hodgman 多边形裁剪算法

二、实验内容

Sutherland-Hodgman 多边形裁剪算法：

1、采用 MFC 基于对话框实现；

2、多边形绘制：鼠标在 picture control 控件中单击鼠标画边，画完后右击鼠标，即可的带封闭对变形；

3、裁剪线框绘制：在控件中某位置按下鼠标，作为线框左上角，拖拽鼠标到线框右下角，然后松开鼠标即可绘制一个矩形裁剪线框。

4、算法核心思想：

(1) 按照矩形线框左、右、下、上的顺序裁剪多边形。

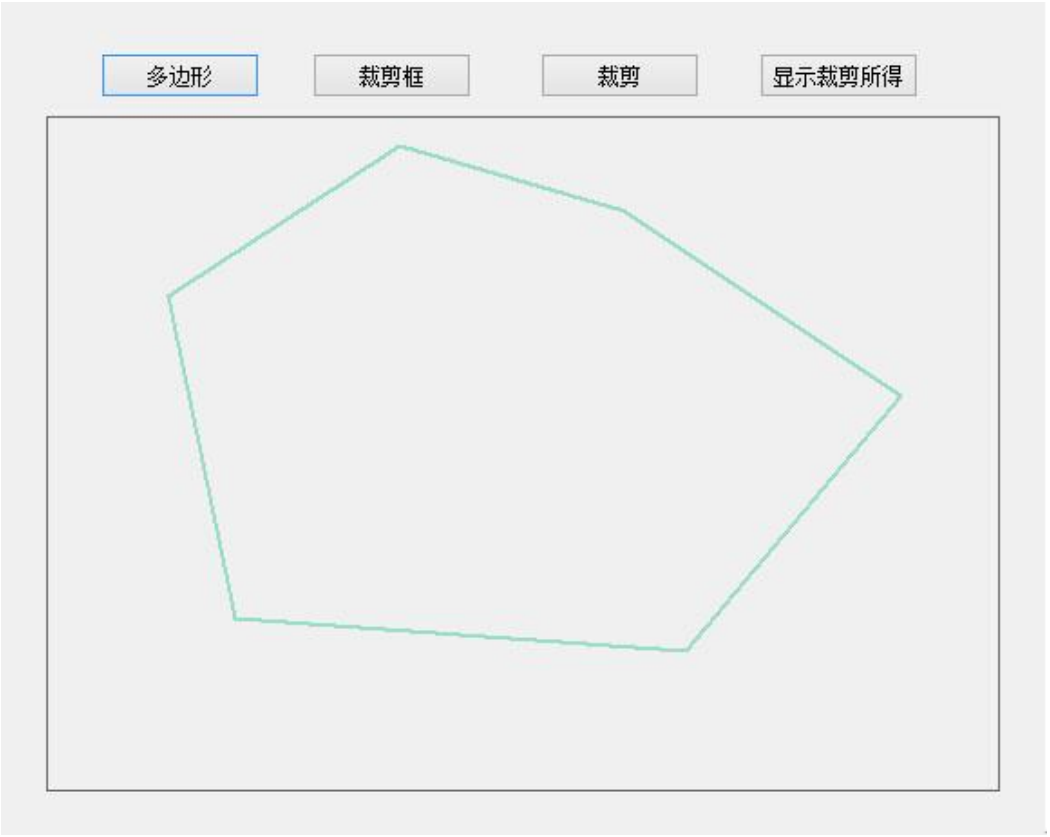
(2) 自绘制多边形起获取单击所得点存入数组中。

(3) 对于每一条裁剪边界，从多边形的第二个点开始遍历，如果当前点和前一个点所在的直线与边界有交点，则输出交点到新数组中。并且如果当前点在线框内侧，则再输出当前点到新数组中。

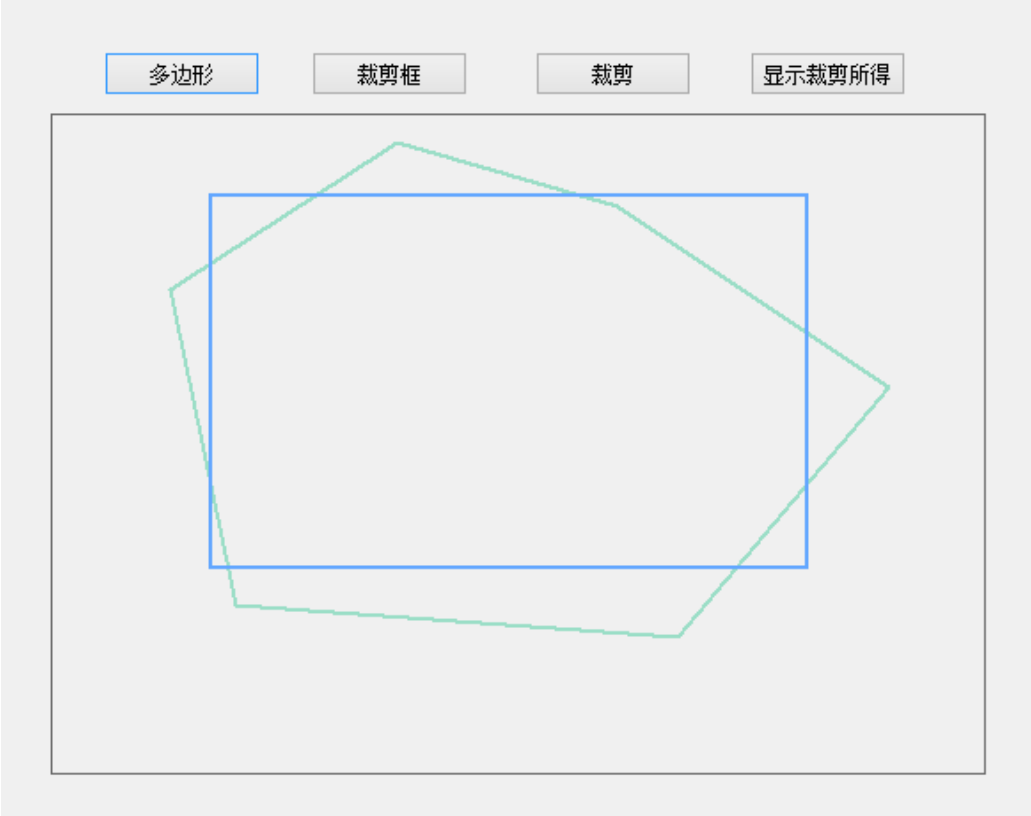
(4) 裁剪完成后根据新数组绘制出新多边形

三、实验结果

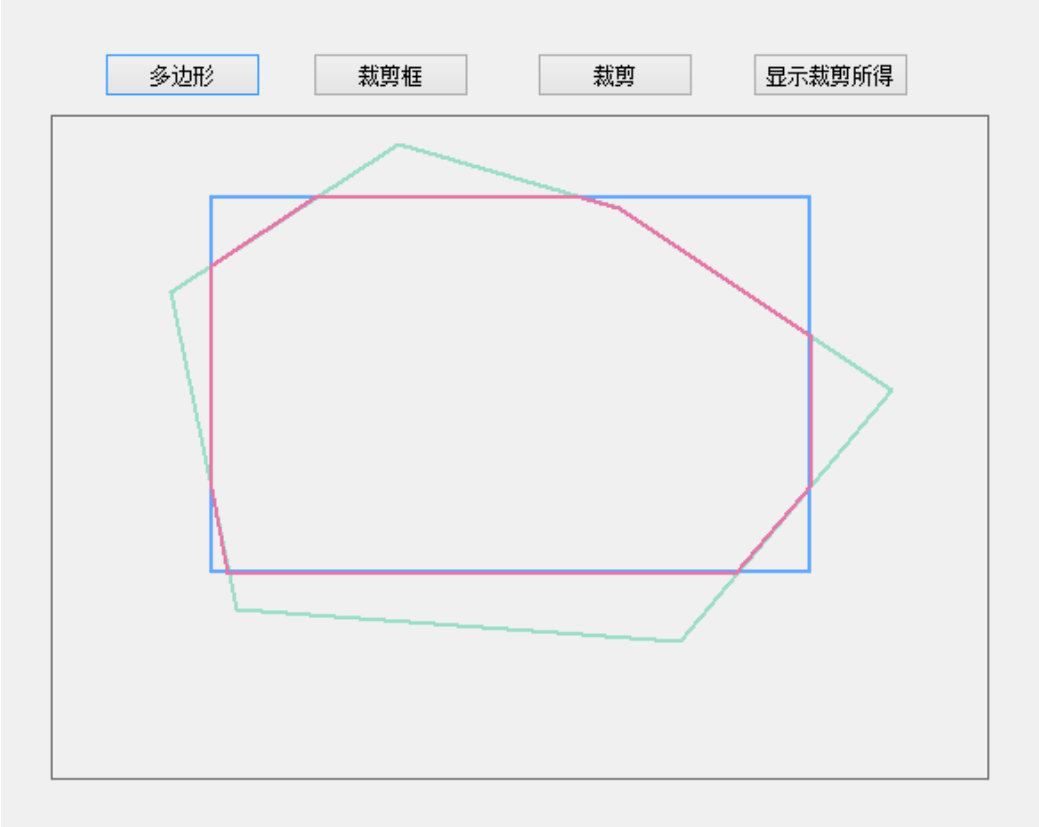
1. 绘制多边形



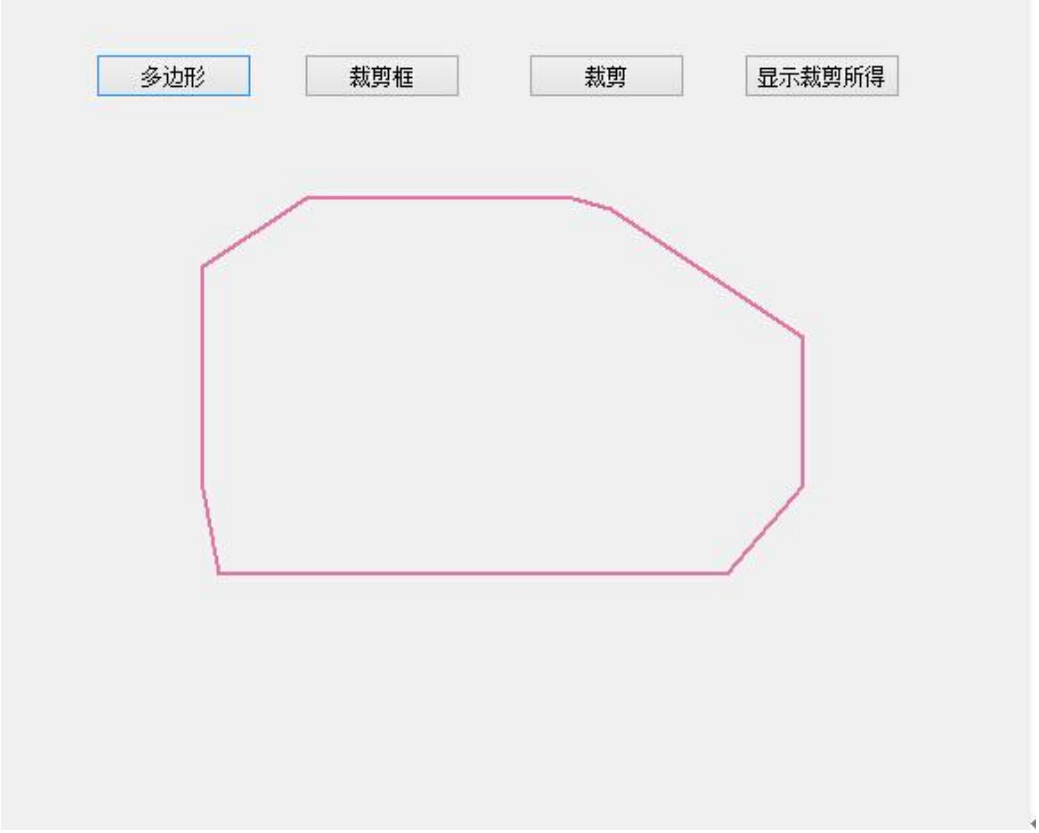
2. 绘制裁剪线框



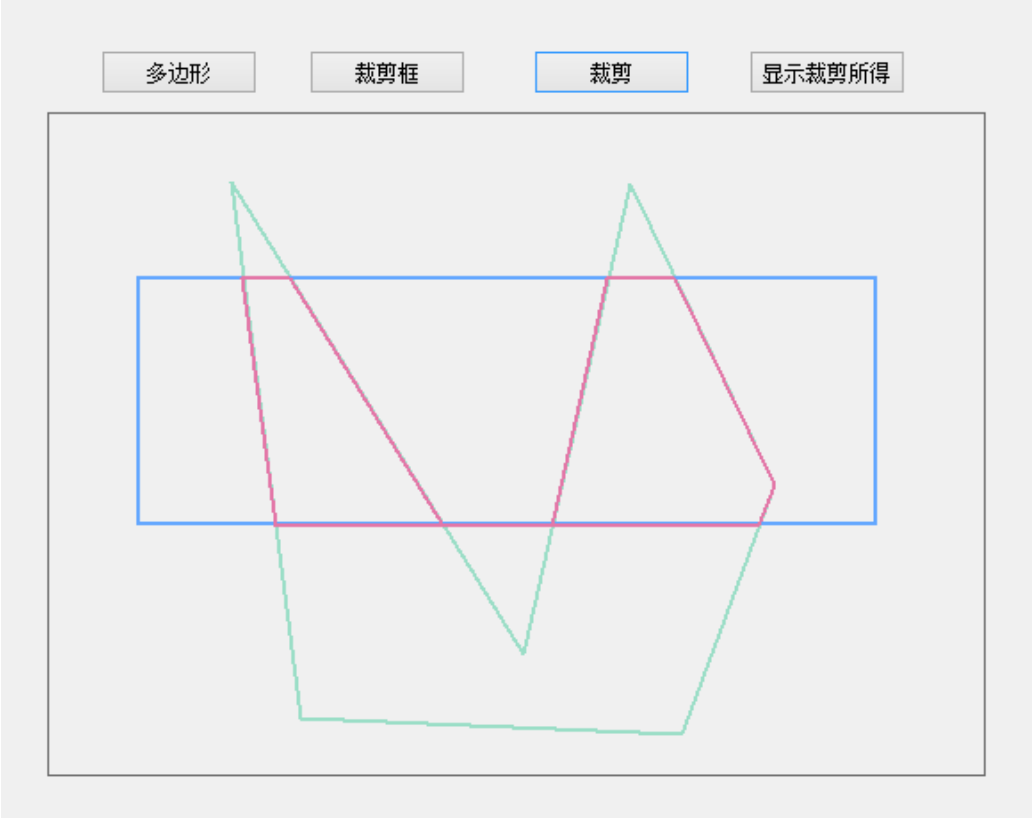
3. 点击裁剪，红色的线框即为裁剪所得。



4. 可以将裁减结果单独显示



5. 由此图可得该多边形有一定的裁剪缺陷，不能够裁剪凹多边形。



四、实验分析和总结

1. 本次实验较好的完成了 Sutherland-Hodgman 多边形裁剪算法，实现了可以对一个基本的凸多边形进行裁剪的功能。
2. 从裁剪凹多边形的实验结果来看，Sutherland-Hodgman 多边形裁剪算法在进行凹多边形裁剪时确实会出现一些问题。
3. 此次实验的优点是还可以进行裁剪后多边形的单独显示，缺点是没有实现清空功能，每次裁剪的时候需要重启窗口。

五、源代码

1. 头文件

```
public:
    afx_msg void OnBnClickedButton1();//多边形
    afx_msg void OnBnClickedButton2();//裁剪框
    afx_msg void OnBnClickedButton3();//裁剪
    bool flag1 = false;//绘制多边形
    bool flag2 = false;//绘制裁剪线框
    bool flag3 = false;//裁剪
    bool draw_aFlag = false;//多边形前面的点是否单击响应
    bool last_aFlag = false;//封闭多边形
    bool draw_bFlag = false;//裁剪框的绘制是否单击响应
    bool swapFlag = false;//标志是否有交换两点
    afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
    afx_msg void OnRButtonDown(UINT nFlags, CPoint point);
    afx_msg void OnLButtonUp(UINT nFlags, CPoint point);
    CArray <CPoint, CPoint&> aArray;//存储类型CPoint,访问类型CPoint
    CArray <CPoint, CPoint&> bArray; //存储左裁剪结果
    CArray <CPoint, CPoint&> cArray;
    CArray <CPoint, CPoint&> dArray;
    CArray <CPoint, CPoint&> eArray;
    CPoint edge1, edge2;//鼠标点击画裁剪框的端点
    CPoint intersectP;
    void drawPolygonA(CDC *pDc);
    void drawPolygonE(CDC *pDc, CArray <CPoint, CPoint&> Array);
    void drawCuttingFrame(CDC *pDc);
    afx_msg void OnRButtonUp(UINT nFlags, CPoint point);
    void intersectPointx(CPoint p1, CPoint p2, int edgex);//x,左右边界
    void intersectPointy(CPoint p1, CPoint p2, int edgey);//y,上下边界
    void xySwap(CPoint &p1, CPoint &p2);
    afx_msg void OnBnClickedButton4();//显示裁剪所得
```

2. Cpp 文件

```
void CpolygonClippingDlg::OnPaint()
{
    if ……
    else
    {
        CWnd *pWin = GetDlgItem(IDC_STATIC);//获取该控件的指针
        CDC *pDc = pWin->GetDC();//获取该控件的画布
        if (flag1){ drawPolygonA(pDc); }
```

```
        if (flag2){ drawCuttingFrame(pDc); }
        if (flag3){ drawPolygonA(pDc); }
    }
}
//画多边形
void CpolygonClippingDlg::drawPolygonA(CDC *pDc)
{
    if (flag1)
    {
        CPen myPen;
        myPen.CreatePen(PS_SOLID, 2, RGB(155, 222, 199));//清新绿
        CPen *Oldpen = pDc->SelectObject(&myPen);
        if (draw_aFlag)//画线
        {
            int size = aArray.GetSize();
            pDc->MoveTo(aArray.GetAt(size - 2));
            pDc->LineTo(aArray.GetAt(size - 1));
        }
        else if (last_aFlag)
        {
            int size = aArray.GetSize();
            pDc->MoveTo(aArray.GetAt(size - 2));
            pDc->LineTo(aArray.GetAt(size - 1));
            pDc->MoveTo(aArray.GetAt(size - 1));
            pDc->LineTo(aArray.GetAt(0));
        }
    }
    if (flag3)//输出最终裁剪所得多边形
    {
        CPen myPen;
        myPen.CreatePen(PS_SOLID, 2, RGB(227, 119, 168));//粉紫色
        CPen *Oldpen = pDc->SelectObject(&myPen);
        int size = eArray.GetSize();
        for (int i = 0; i < size; i++)
        {
            if (i == size - 1)
            {
                pDc->MoveTo(eArray.GetAt(i));
                pDc->LineTo(eArray.GetAt(0));
            }
            else
            {
                pDc->MoveTo(eArray.GetAt(i));
                pDc->LineTo(eArray.GetAt(i + 1));
```

```
            }
        }
    }
}
//画裁剪框
void CpolygonClippingDlg::drawCuttingFrame(CDC *pDc)
{
    CPen myPen;
    myPen.CreatePen(PS_SOLID, 2, RGB(99, 167, 255));//清新蓝
    int nSavedDC = pDc->SaveDC();
    pDc->SelectObject(&myPen);
    (CBrush*)pDc->SelectStockObject(NULL_BRUSH);//空心的笔
    if (flag2)//点击画框按钮
    {
        if (draw_bFlag)//画裁剪框的鼠标抬起
        {
            pDc->Rectangle(edge1.x, edge1.y, edge2.x, edge2.y);
        }
    }
}
//求边与左右边界交点
void CpolygonClippingDlg::intersectPointx(CPoint p1, CPoint p2,int edgex )
{
    if (p1.x<p2.x||p1.x>p2.x)//不与边界平行
    {
        int x = edgex;
        double  k = (p2.y - p1.y)*1.0 / (p2.x - p1.x);
        double y = k *(x - p1.x) + p1.y;
        intersectP.x = x;
        intersectP.y = y;
    }
}
//求边与上下边界交点
void CpolygonClippingDlg::intersectPointy(CPoint p1, CPoint p2, int edgey)
{
    if (p1.y<p2.y||p1.y>p2.y)//不与边界平行
    {
        int y = edgey;
        double k, x;
        if (p1.x<p2.x || p1.x>p2.x)//斜率存在
        {
            k = (p2.y - p1.y)*1.0 / (p2.x - p1.x);
            x = (y - p1.y) / k + p1.x;
        }
```

```
            intersectP.x = x;
            intersectP.y = y;
        }
    }

    //多边形
    void CpolygonClippingDlg::OnBnClickedButton1()
    {
        flag1 = true;
        flag2 = false;
    }
    //裁剪框
    void CpolygonClippingDlg::OnBnClickedButton2()
    {
        flag2 = true;
        flag1 = false;
    }
    //裁剪算法。核心算法。
    void CpolygonClippingDlg::OnBnClickedButton3()
    {
        flag1 = false;
        flag2 = false;
        //左裁剪
        int size = aArray.GetSize();
        CPoint p1, p2;//前一个点和当前点
        int edge = edge1.x;
        for (int i = 0; i < size; i++)
        {
            if (i == 0)//遇到零就是令最后一个点为当前点的前一个点
            {
                p1 = aArray.GetAt(size -1);
                p2 = aArray.GetAt(0);
            }
            else{
                //前一个点和当前点
                 p1 = aArray.GetAt(i - 1);
                 p2 = aArray.GetAt(i);
            }
            if ((p1.x<=edge&&p2.x>=edge) || (p1.x>=edge&&p2.x<=edge))//如果有交点
            {
                intersectPointx(p1, p2, edge);//求与左边界交点，保存在intersectP中
                bArray.Add(intersectP);//交点加入数组
            }
            if (p2.x >= edge)//如果第二个点在线框内侧，保留第二个点
```

```
        {
            bArray.Add(p2);//内侧的第二个点加入数组
        }
    }
    //右裁剪
     size = bArray.GetSize();
     edge = edge2.x;
    for (int i = 0; i < size; i++)
    {
        if (i == 0)//遇到零就是令最后一个点为当前点的前一个点
        {
            p1 = bArray.GetAt(size - 1);
            p2 = bArray.GetAt(0);
        }
        else{
            //前一个点和当前点
            p1 = bArray.GetAt(i - 1);
            p2 = bArray.GetAt(i);
        }
        if ((p1.x<=edge&&p2.x>=edge) || (p1.x>=edge&&p2.x<=edge))//如果有交点
        {
            intersectPointx(p1, p2, edge);//求与左边界交点，保存在intersectP中
            cArray.Add(intersectP);
        }
        if (p2.x <= edge)//如果第二个点在线框内侧，保留第二个点，加入数组
        {
            cArray.Add(p2);
        }
    }
    //下裁剪
    size = cArray.GetSize();
    edge = edge2.y;
    for (int i = 0; i < size; i++)
    {
        if (i == 0)//遇到零就是令最后一个点为当前点的前一个点
        {
            p1 = cArray.GetAt(size - 1);
            p2 = cArray.GetAt(0);
        }
        else{
            //前一个点和当前点
            p1 = cArray.GetAt(i - 1);
            p2 = cArray.GetAt(i);
        }
```

```
            if ((p1.y<=edge&&p2.y>=edge) || (p1.y>=edge&&p2.y<=edge))//如果有交点
            {
                intersectPointy(p1, p2, edge);//求与左边界交点，保存在intersectP中
                dArray.Add(intersectP);
            }
            if (p2.y <= edge)//如果第二个点在线框内侧，保留第二个点，加入数组
            {
                dArray.Add(p2);
            }
        }
        //上裁剪
        size = dArray.GetSize();
        edge = edge1.y;
        for (int i = 0; i < size; i++)
        {
            if (i == 0)//遇到零就是令最后一个点为当前点的前一个点
            {
                p1 = dArray.GetAt(size - 1);
                p2 = dArray.GetAt(0);
            }
            else{
                //前一个点和当前点
                p1 = dArray.GetAt(i - 1);
                p2 = dArray.GetAt(i);
            }

            if ((p1.y <= edge&&p2.y >= edge) || (p1.y >= edge&&p2.y <= edge))//如果有交点
            {
                intersectPointy(p1, p2, edge);//求与左边界交点，保存在intersectP中
                eArray.Add(intersectP);
            }
            if (p2.y >= edge)//如果第二个点在线框内侧，保留第二个点，加入数组
            {
                eArray.Add(p2);
            }
        }
        flag3 = true;//所有裁剪完成，输出点
    }
    //单独显示裁剪所得
    void CpolygonClippingDlg::OnBnClickedButton4()
    {
        //>GetDC()->FillSolidRect(lRect.left, lRect.top, lRect.Width(), lRect.Height(),
    RGB(240, 240, 240));
```

```
    CStatic* pStatic = (CStatic*)GetDlgItem(IDC_STATIC);
    CRect rect;
    pStatic->GetClientRect(&rect);
    pStatic->GetDC()->FillSolidRect(rect.left, rect.top, rect.Width(), rect.Height(),
RGB(255, 255, 255));//背景色置白色
}

//单击画多边形
void CpolygonClippingDlg::OnLButtonDown(UINT nFlags, CPoint point)
{
    CRect rectP;//获取控件的大小
    GetDlgItem(IDC_STATIC)->GetWindowRect(&rectP);//获取控件相对于屏幕的位置
    ScreenToClient(rectP);//转化为对话框上的相对位置
    if ((point.x >= rectP.left && point.x <= rectP.right) && (point.y >= rectP.top && point.y
<= rectP.bottom))
    {
        point.x = point.x - rectP.left;
        point.y = point.y - rectP.top;
    }
    if (flag1)//绘制多边形
    {
        aArray.Add(point);
        draw_aFlag = false;//如果是第一个点点下时不画线
        last_aFlag = false;
        if (aArray.GetSize() > 1)//从第二个点开始连线
        {
            draw_aFlag = true;
            last_aFlag = false;
        }
    }
    if (flag2)
    {
        edge1 = point;
        edge2 = point;//初始化
        draw_bFlag = false;
    }
    CDialogEx::OnLButtonDown(nFlags, point);
}

//鼠标响应函数
void CpolygonClippingDlg::OnLButtonDown(UINT nFlags, CPoint point)
{
    CRect rectP;//获取控件的大小
    GetDlgItem(IDC_STATIC)->GetWindowRect(&rectP);//获取控件相对于屏幕的位置
```

```
    ScreenToClient(rectP);//转化为对话框上的相对位置
    if ((point.x >= rectP.left && point.x <= rectP.right) && (point.y >= rectP.top && point.y
<= rectP.bottom))
    {
        point.x = point.x - rectP.left;
        point.y = point.y - rectP.top;
    }
    if (flag1)//绘制多边形
    {
        aArray.Add(point);
        draw_aFlag = false;//如果是第一个点点下时不画线
        last_aFlag = false;
        if (aArray.GetSize() > 1)//从第二个点开始连线
        {
            draw_aFlag = true;
            last_aFlag = false;
        }
    }
    if (flag2)
    {
        edge1 = point;
        edge2 = point;//初始化
        draw_bFlag = false;
    }
    CDialogEx::OnLButtonDown(nFlags, point);
}

void CpolygonClippingDlg::OnLButtonUp(UINT nFlags, CPoint point)
{
    CRect rectP;//获取控件的大小
    GetDlgItem(IDC_STATIC)->GetWindowRect(&rectP);//获取控件相对于屏幕的位置
    ScreenToClient(rectP);//转化为对话框上的相对位置
    if ((point.x >= rectP.left && point.x <= rectP.right) && (point.y >= rectP.top && point.y
<= rectP.bottom))
    {
        point.x = point.x - rectP.left;
        point.y = point.y - rectP.top;
    }
    if (flag2)
    {
        edge2 = point;
        if (edge1.x > edge2.x)//保证裁剪线框是从左上角到右下角
        {
            xySwap(edge1, edge2);
```

```
        }
        draw_bFlag = true;
    }
    CDialogEx::OnLButtonUp(nFlags, point);
}
void CpolygonClippingDlg::OnRButtonDown(UINT nFlags, CPoint point){
    if (flag1)//绘制多边形
    {
        CRect rectP;//获取控件的大小
        GetDlgItem(IDC_STATIC)->GetWindowRect(&rectP);//获取控件相对于屏幕的位置
        ScreenToClient(rectP);//转化为对话框上的相对位置

        if (aArray.GetSize() > 2)//可构成多边形
        {
            if ((point.x >= rectP.left && point.x <= rectP.right) && (point.y >= rectP.top
&& point.y <= rectP.bottom))
            {//保证在控件内
                point.x = point.x - rectP.left;
                point.y = point.y - rectP.top;
                //aArray.Add(point);
                draw_aFlag = false;
                last_aFlag = true;//点下时封闭多边形
            }
        }
    }
    CDialogEx::OnRButtonDown(nFlags, point);
}
//绘制完一个多边形后不能再绘制了
void CpolygonClippingDlg::OnRButtonUp(UINT nFlags, CPoint point)
{
    flag1 = false;
    CDialogEx::OnRButtonUp(nFlags, point);
}
```