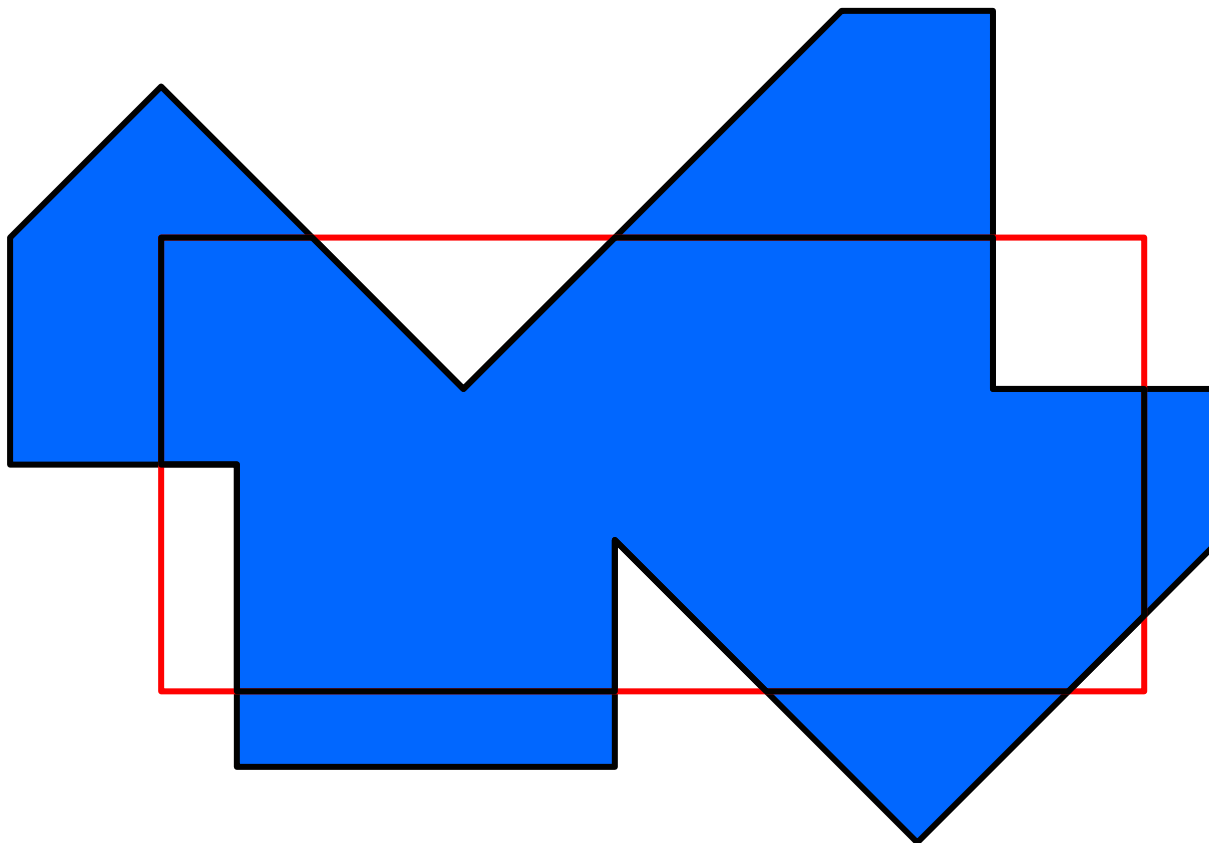
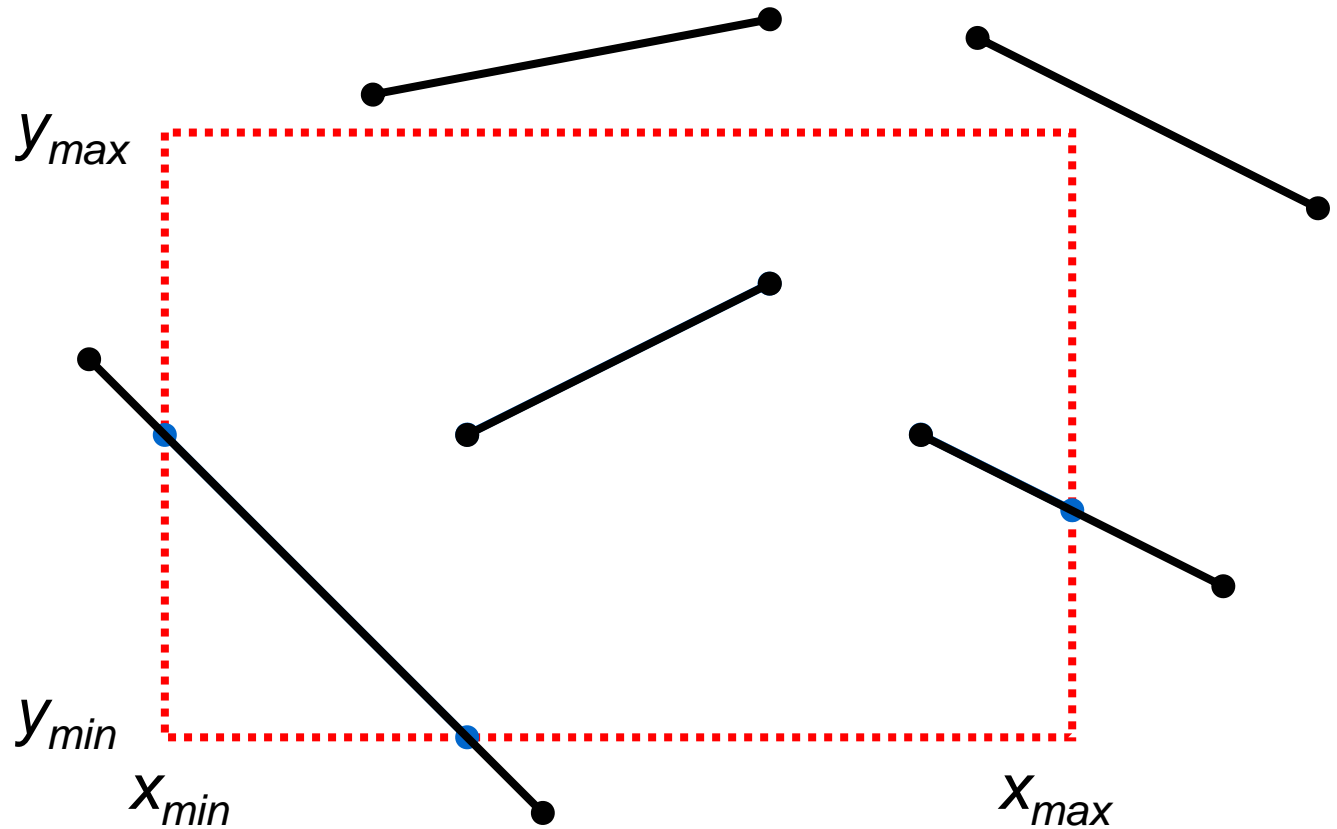


Kapitel 5: 2D-Clipping

2D-Clipping



Clipping von Linien



Region Code: Definition

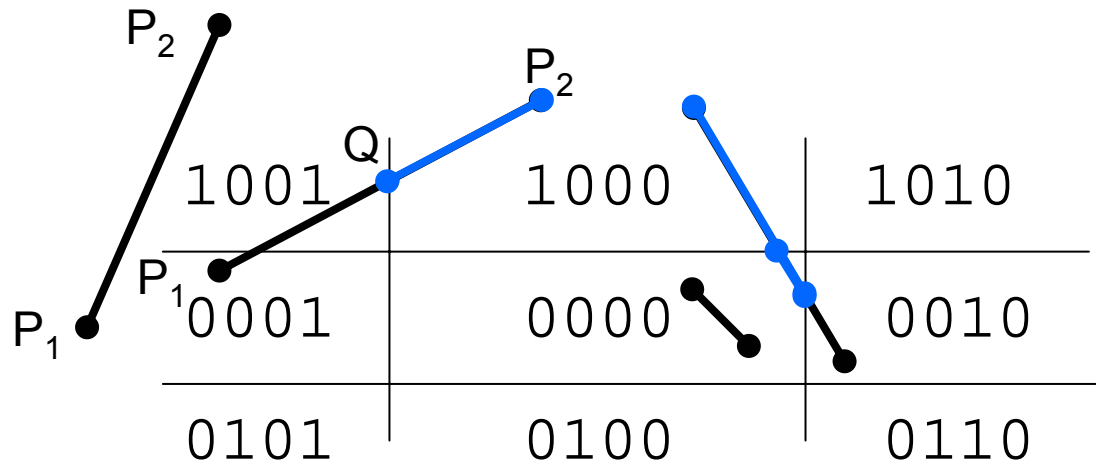
Bit 0: links Bit1:rechts Bit2: unten Bit3: oben

1001	1000	1010
y_{max}		
0001	0000	0010
y_{min}		
0101	x_{min} 0100	x_{max} 0110

Region Code: Berechnung

```
private static final byte CENTER = 0;
private static final byte LEFT   = 1;
private static final byte RIGHT  = 2;
private static final byte BOTTOM = 4;
private static final byte TOP    = 8;
public byte region_code (int x, int y) {
    byte c = CENTER;
    if (x < xmin) c = LEFT;
    if (x > xmax) c = RIGHT;
    if (y < ymin) c = c | BOTTOM;
    if (y > ymax) c = c | TOP;
    return c;
}
```

Cohen & Sutherland

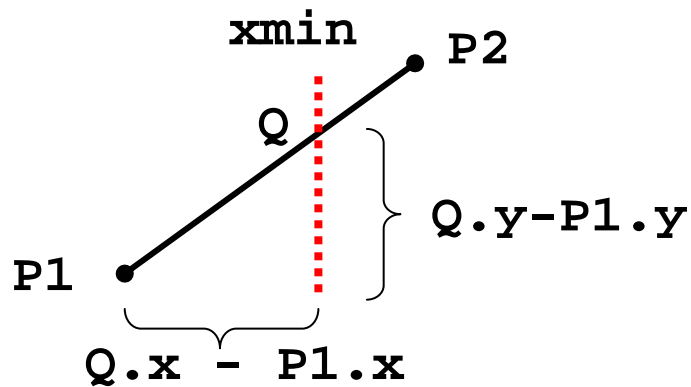


falls $\text{code}(P_1) \ \&\& \ \text{code}(P_2) \neq 0 \Rightarrow$ komplett außerhalb

falls $\text{code}(P_1) \ || \ \text{code}(P_2) = 0 \Rightarrow$ komplett innerhalb

sonst: berechne Schnittpunkt Q und teste Restlinie erneut

Schnittpunkte



$$\text{slope} = \frac{Q.y - P1.y}{Q.x - P1.x}$$

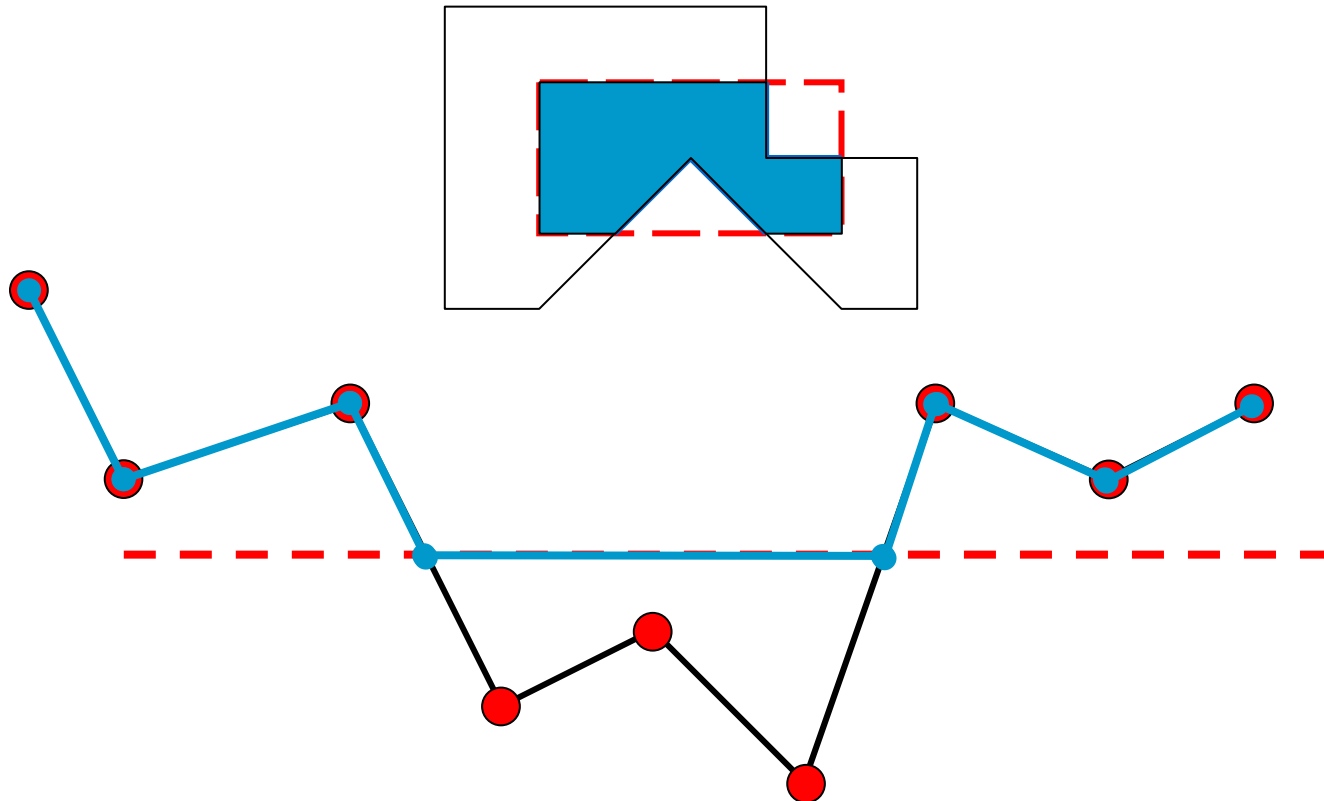
$$\text{slope} = \frac{P2.y - P1.y}{P2.x - P1.x}$$

```
slope = (double)(P2.y - P1.y)/(P2.x - P1.x);  
Q.x   = xmin  
Q.y   = (int)(Q.x - P1.x)*slope + P1.y
```

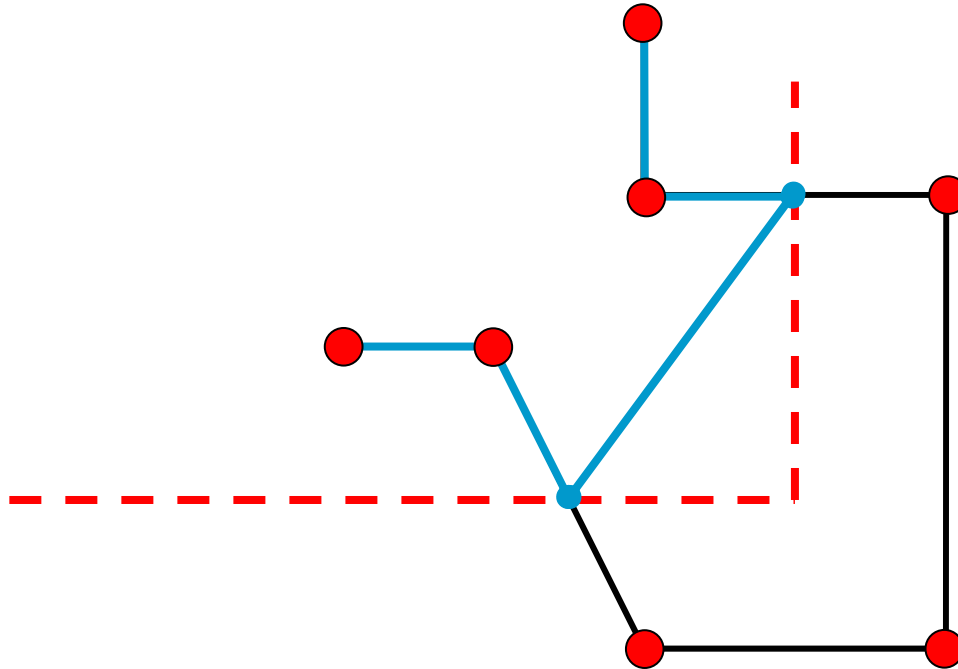
Cohen-Sutherland

```
boolean cohen_sutherland(Point P1,  
                        Point P2,  
                        Point Q1,  
                        Point Q2){  
    // clippt Gerade P1,P2 am Fenster  
    // liefert true, falls sichtbar  
    // liefert in Q1,Q2 den sichtbaren Teil  
    ...  
}
```


Clipping von Polygonen



Problem bei Clip-Fenster-Ecken



Sutherland & Hodgman

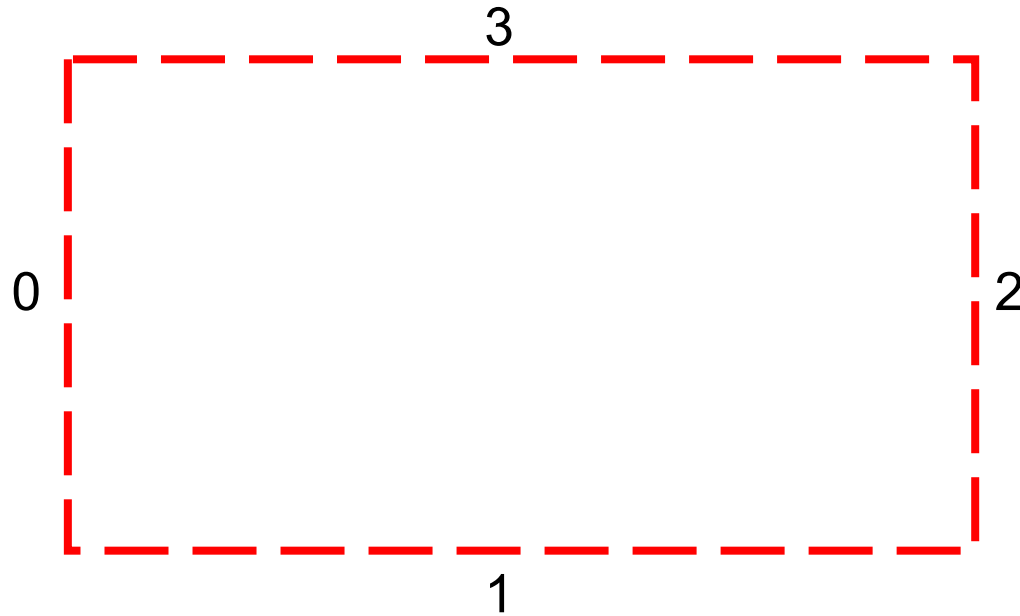
für eine Clipping-Gerade E

für jeden Polygonpunkt P_i :

falls P_i sichtbar: übernahm P_i

falls Kante von P_i zu P_{i+1} E schneidet:
übernahm Schnittpunkt

4 Clipping-Kanten



Sichtbarkeitstest

```
boolean On_Visible_Side(  
    Point P, int wert, int fall) {  
    switch (fall) {  
        case 0: return (P.x >= wert);  
        case 1: return (P.y >= wert);  
        case 2: return (P.x <= wert);  
        case 3: return (P.y <= wert);  
    }  
}
```

Schnittpunkt

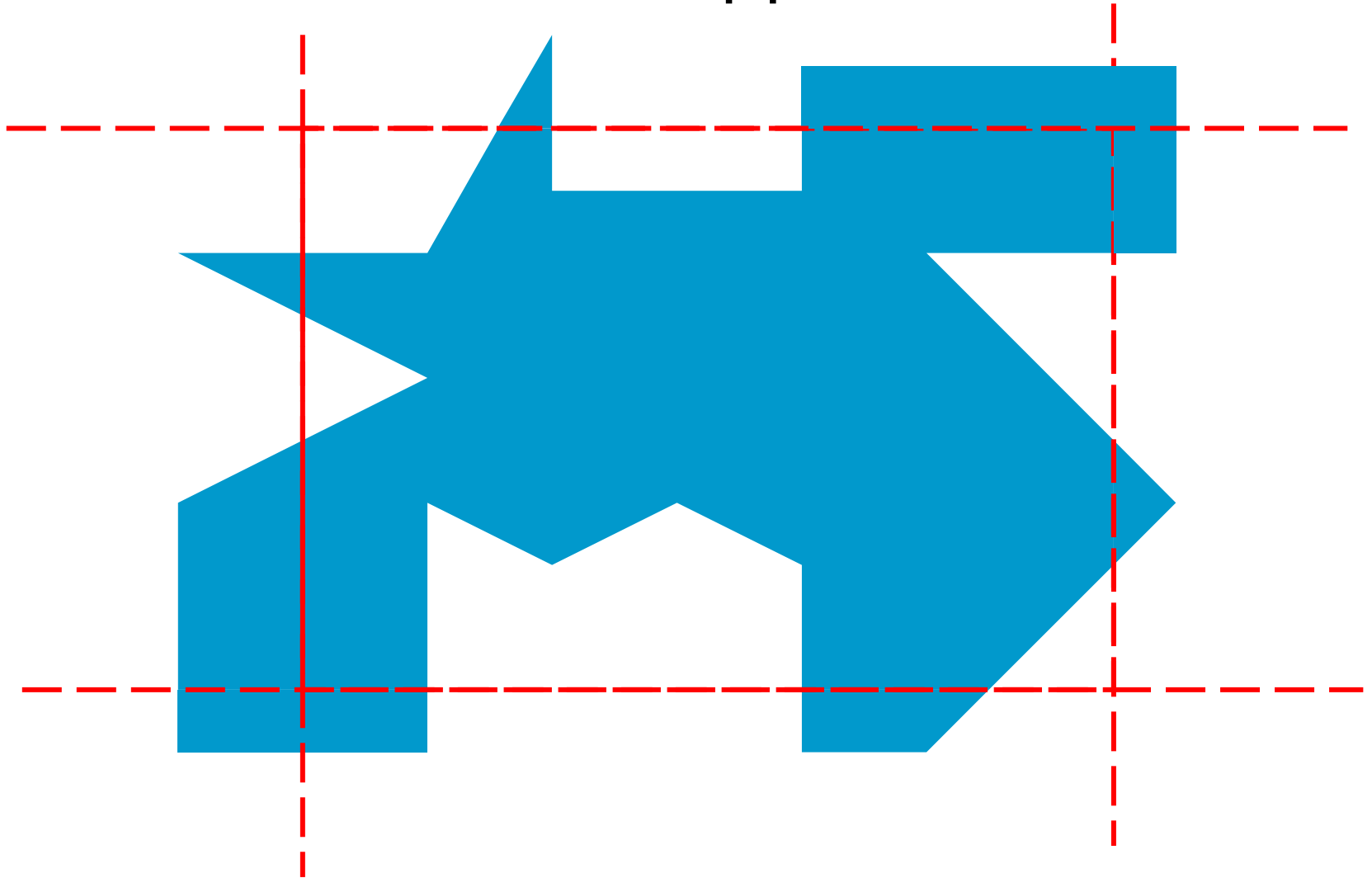
```
boolean intersection(  
    Point P1, Point P2,  
    int wert, int fall, Point I) {  
    ...  
    P1_vis = On_Visible_Side(P1,wert,fall);  
    P2_vis = On_Visible_Side(P2,wert,fall);  
    ...  
    slope =(double)(P2.y-P1.y)/  
            (double)(P2.x-P1.x);  
    if (fall %2 == 0) {  
        I.x = (int) wert;  
        I.y = (int)(wert-P1.x)*slope + P1.y;  
    }  
    ...  
}
```

Aufruf von Sutherland_hodgman

```
int n;           // Zahl der Eckpunkte
Point[] points; // Polygon

n = sutherland_hodgman(n, points, xmin, 0);
n = sutherland_hodgman(n, points, ymin, 1);
n = sutherland_hodgman(n, points, xmax, 2);
n = sutherland_hodgman(n, points, ymax, 3);
```

4 x Clippen



Clipping-Implementation

Java-Applet:

[~cg/2004/skript/node46.htm](#)

Clipping eines Kreises

