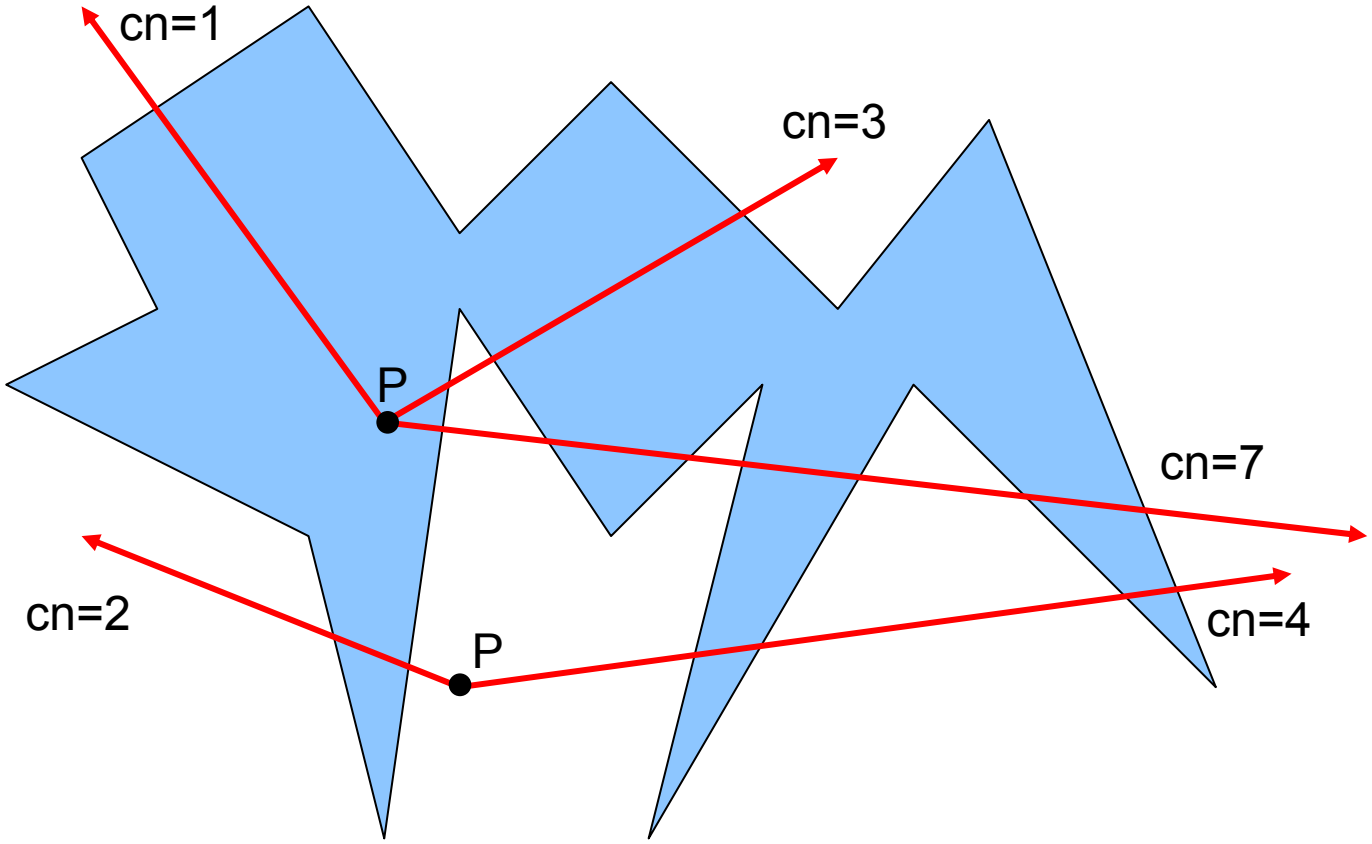
Computergrafik SS 2010

Oliver Vornberger

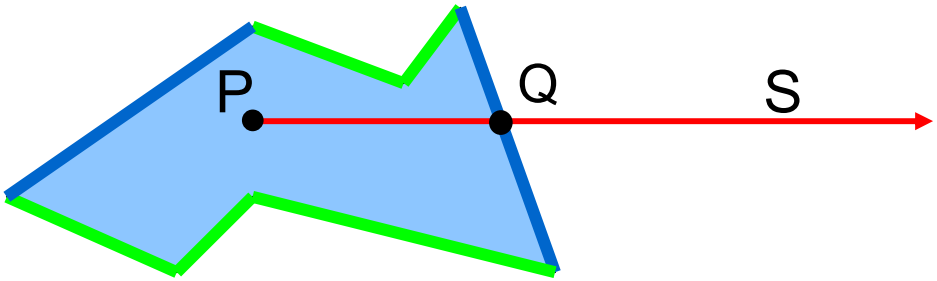
Kapitel 4:
noch 2D-Füllen

Vorlesung vom 20.04.2010

Punkt in Polygon



Kreuzungszahl berechnen



Sei *S* der von *P* nach rechts gehende Strahl

Für jede Polygonkante von P_1 nach P_2:

falls P_1 und P_2 oberhalb: kein Schnittpunkt

falls P_1 und P_2 unterhalb: kein Schnittpunkt

falls P_1 und P_2 auf verschiedenen Seiten:

berechne Schnittpunkt *Q* mit S

falls rechts von *P*: erhöhe Kreuzungszahl

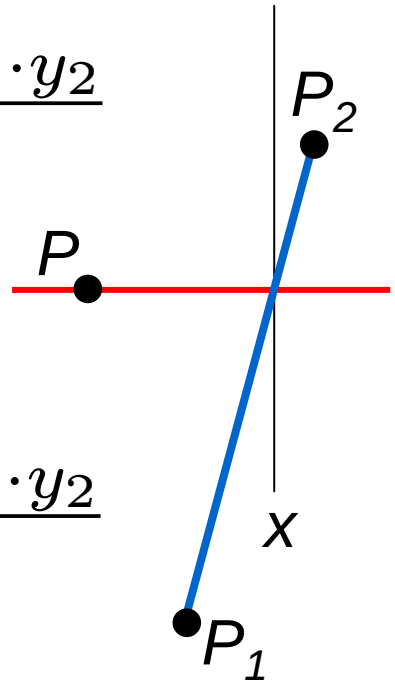
Schnittpunkt berechnen

$$f(x) = \frac{y_2 - y_1}{x_2 - x_1} \cdot x + \frac{x_2 \cdot y_1 - x_1 \cdot y_2}{x_2 - x_1}$$

$$f(x) = y$$

$$y = \frac{y_2 - y_1}{x_2 - x_1} \cdot x + \frac{x_2 \cdot y_1 - x_1 \cdot y_2}{x_2 - x_1}$$

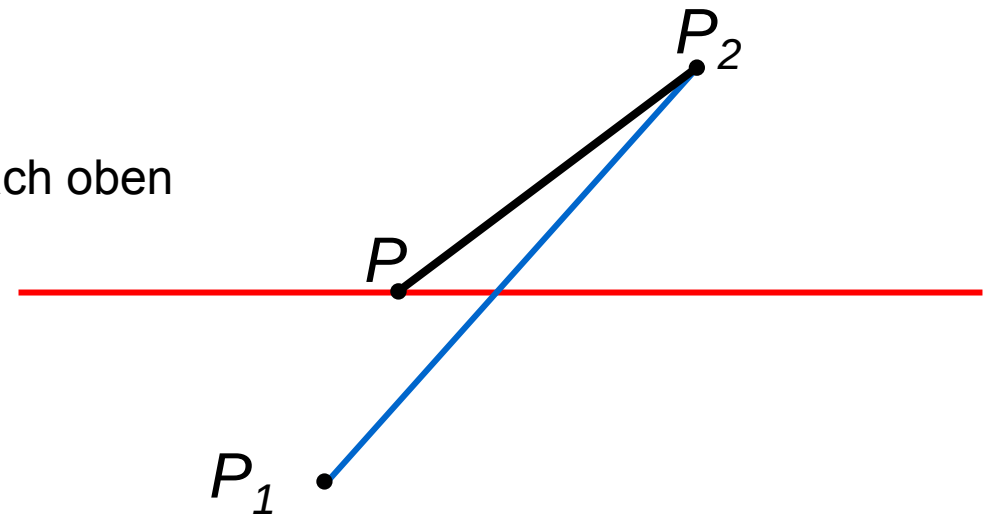
$$x = \frac{y \cdot (x_2 - x_1) - x_2 \cdot y_1 + x_1 \cdot y_2}{y_2 - y_1}$$



public boolean contains (int x, int y)

```
x1 = xpoints[n-1]; y1 = ypoints[n-1];
x2 = xpoints[0];   y2 = ypoints[0];
boolean inside = false;
boolean startUeber = y1 >= y ? true : false;
for (i=1; i<n; i++) {
  boolean endUeber = y2 >= y ? true : false;
  if ((startUeber != endUeber &&
   (double)(y*(x2-x1)- x2*y1 + x1*y2)/(y2-y1)>=x))
      inside = !inside;
  startUeber = endUeber;
  y1=y2; x1=x2; x2=xpoints[i]; y2=ypoints[i];
}
return inside;
```

Steigung statt Schnittpunkt

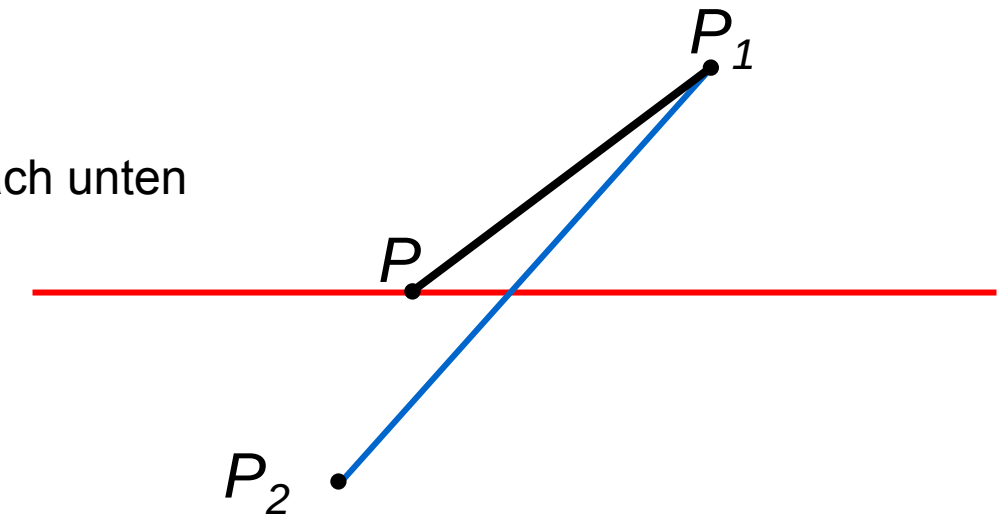


Kante von P_1 nach P_2 schneidet Strahl rechts von P, falls

$$\frac{y_2 - y_1}{x_2 - x_1} \geq \frac{y_2 - y}{x_2 - x}$$

$$(y_2 - y_1) \cdot (x_2 - x) \geq (y_2 - y) \cdot (x_2 - x_1)$$

Steigung statt Schnittpunkt



Kante von P_1 nach P_2 schneidet Strahl rechts von P, falls

$$\frac{y_2 - y_1}{x_2 - x_1} \geq \frac{y_1 - y}{x_1 - x}$$

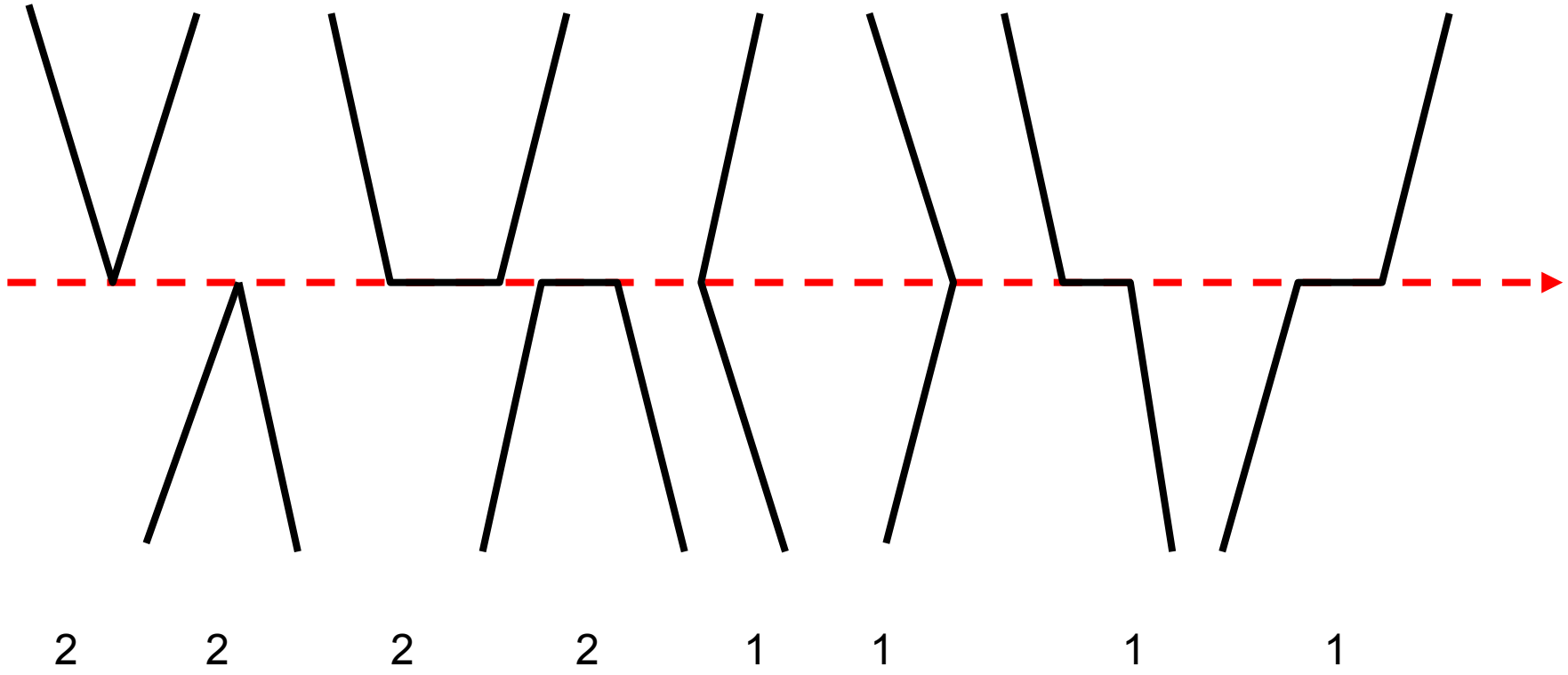
$$(y_2 - y_1) \cdot (x_1 - x) \geq (y_1 - y) \cdot (x_2 - x_1)$$

public boolean contains (int x, int y)

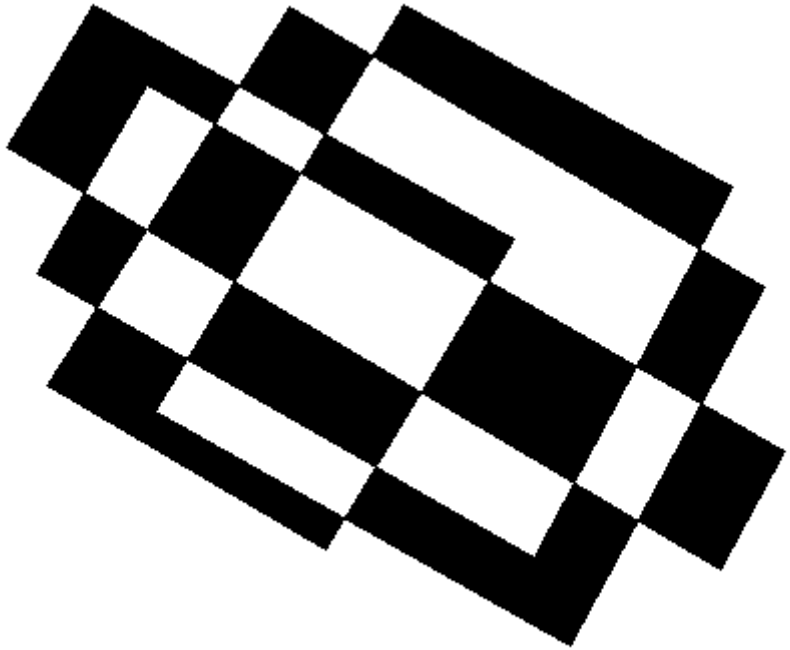
```
...
for (i=1; i<n; i++) {
  boolean endUeber = y2 >= y ? true : false;
  if (startUeber != endUeber && endUeber &&
      (y2-y1)*(x2-x) >= (y2-y)*(x2-x1))
      inside = !inside;
  if (startUeber != endUeber && startUeber &&
     (y2-y1)*(x1-x) >= (y1-y)*(x2-x1))
      inside = !inside;

  ...
```

Sonderfälle



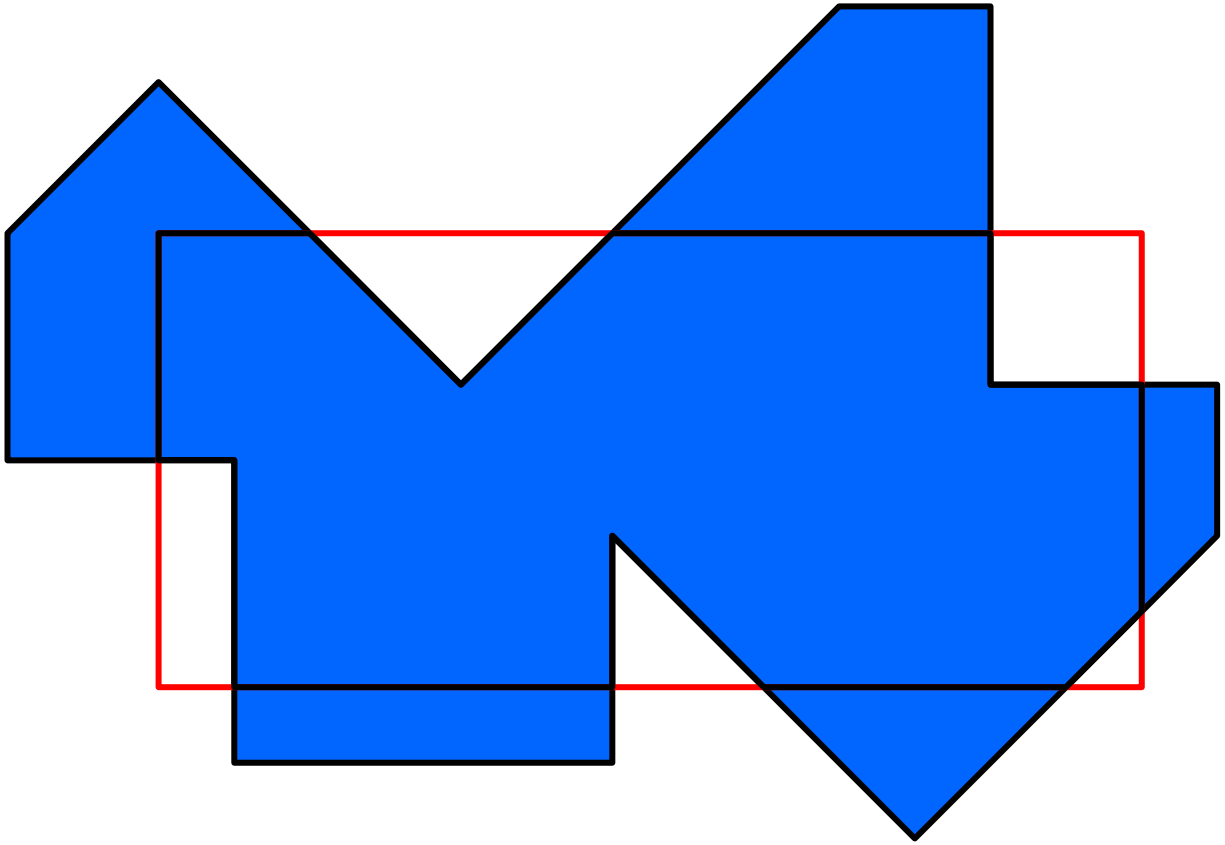
sich überschneidende Polygone



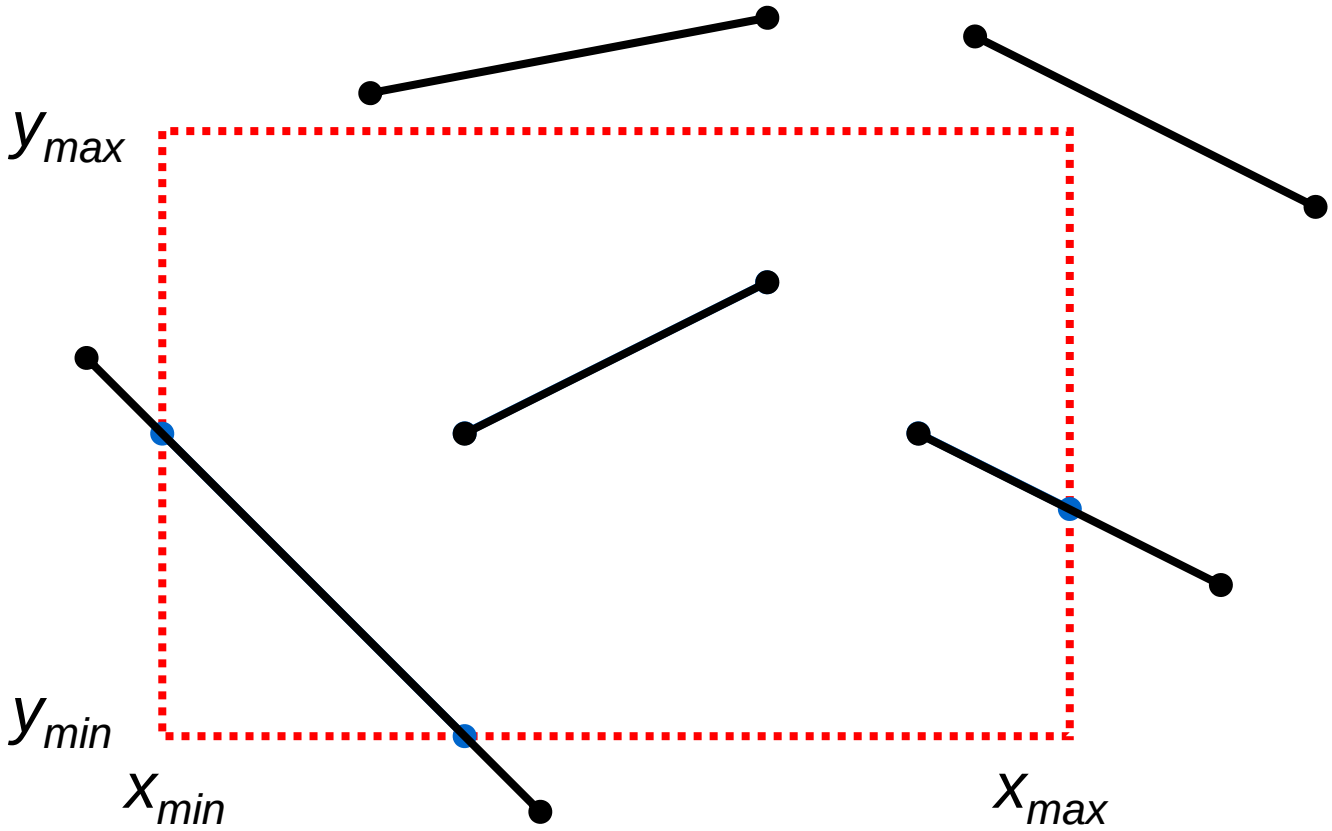
~cg/2010/skript/Applets/2D-basic/App.html

Kapitel 5:
2D-Clipping

2D-Clipping



Clipping von Linien



Region Code: Definition

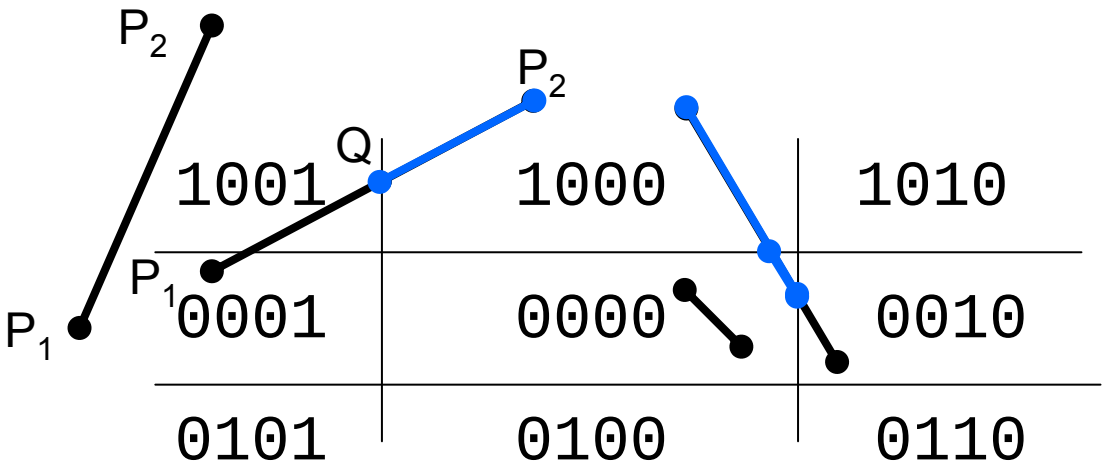
Bit 0: links Bit1:rechts Bit2: unten Bit3: oben

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 code(P_1) && code(P_2) ≠ 0 $\Rightarrow$ komplett außerhalb

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

sonst: berechne Schnittpunkt Q und teste Restlinie erneut

Quiz

P1 und P2 liegen komplett außerhalb, falls

1001	1000	1010
0001	0000	0010
0101	0100	0110

A Code(P1) || code(P2) = 0
| 0,0 %

B Code(P1) || code(P2) ≠ 0
| 0,0 %

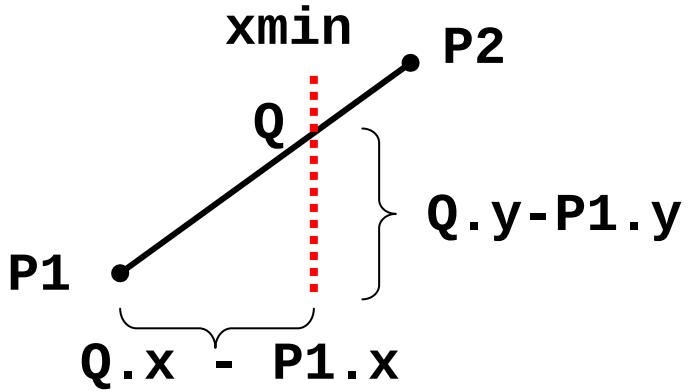
C Code(P1) && code(P2) = 0
| 0,0 %

D Code(P1) && code(P2) ≠ 0
| 0,0 %

Anzahl der Stimmen: 0

Abstimmung läuft ...

Schnittpunkte



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

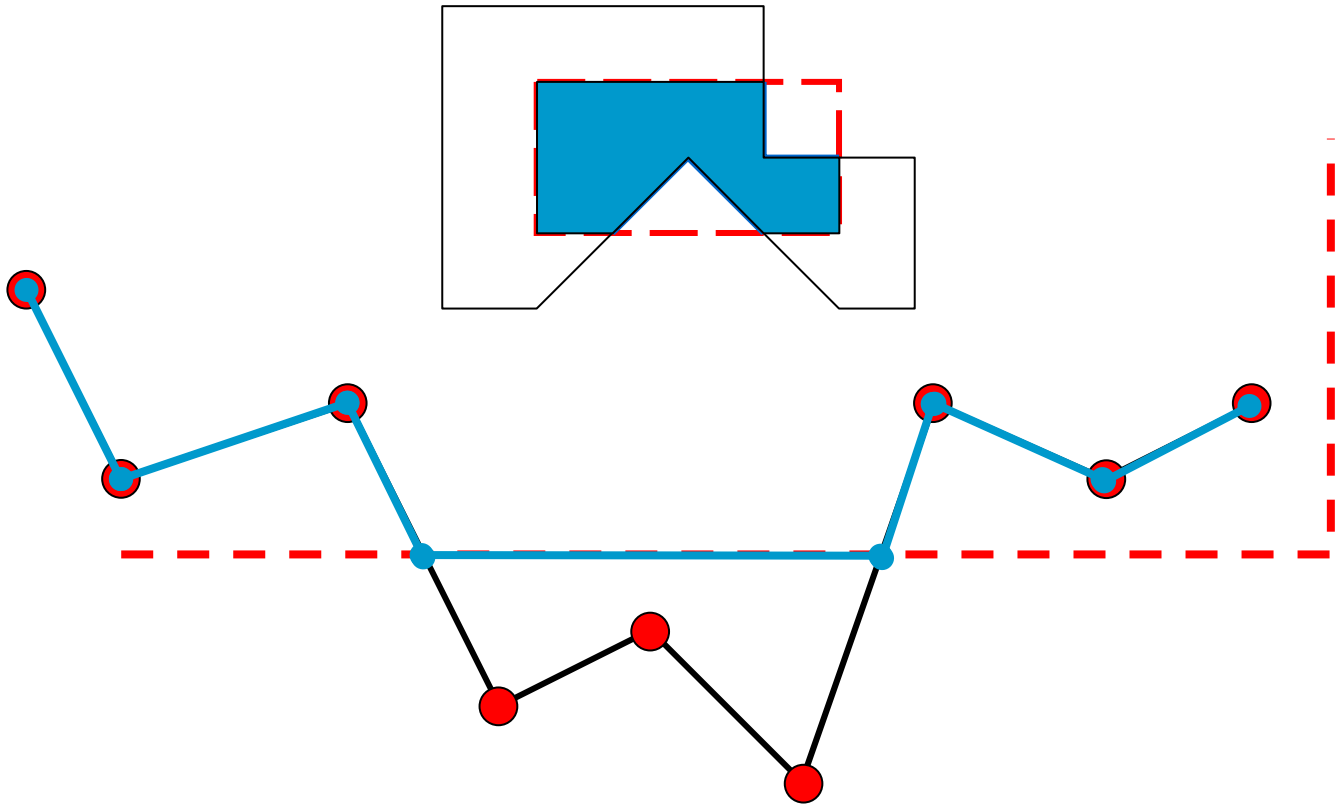
$$slope = \frac{P2.y - P1.y}{P2.x - P1.x}$$

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

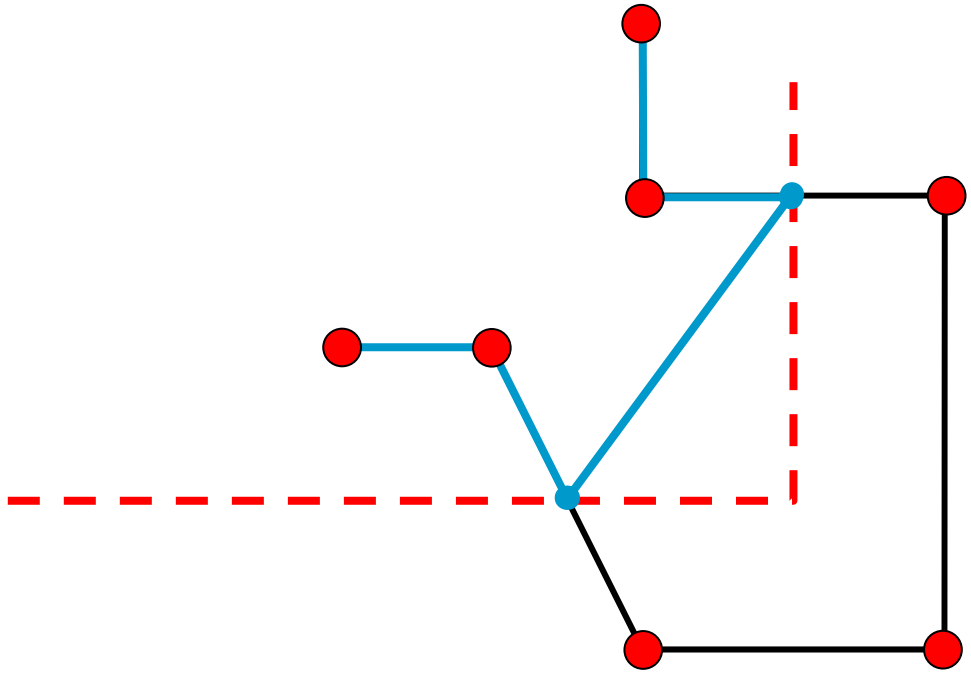
Cohen-Sutherland

```
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

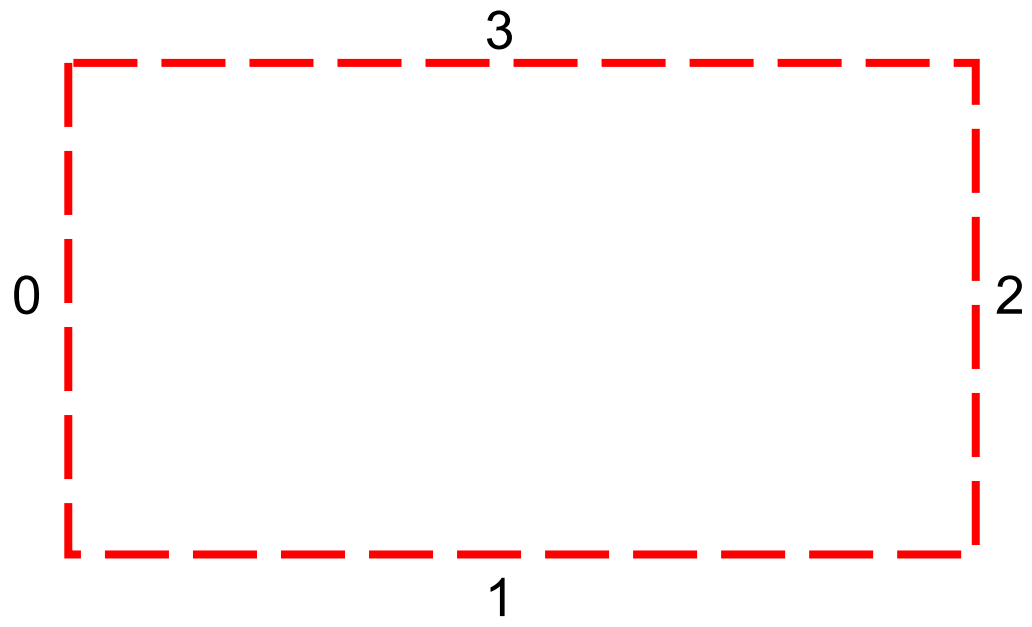
für eine Clipping-Gerade E

für jeden Polygonpunkt P_i:

falls P_i sichtbar: übernimm P_i

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

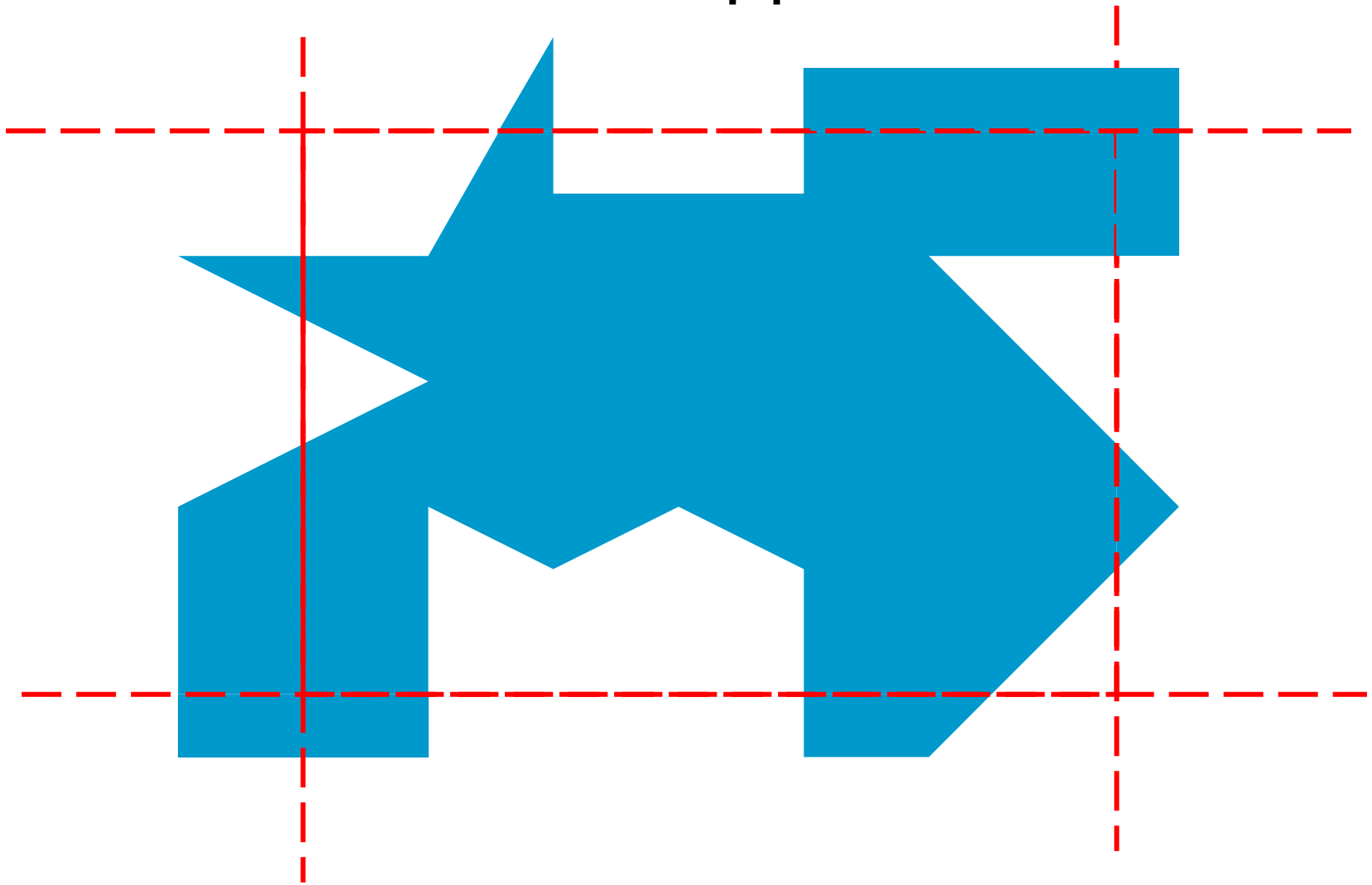
```
boolean intersection(
    Point P1, Point P2,
    int wert, int fall, Point I) {
...
P1_vis = On_Visible_Side(P1,wert,fall);
P2_vis = On_Visible_Side(P2,wert,fall);
...
slope =(double)(P2.y-P1.y)/
       (double)(P2.x-P1.x);
if (fall %2 == 0) {
    I.x = (int) wert;
    I.y = (int)(wert-P1.x)*slope + 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/2010/skript/Applets/2D-basic/App.html

Quiz

Verglichen mit einem Polygon mit 8 Ecken
ist der Rechenaufwand zum Clippen eines Kreises

A niedriger
0,0 %

B gleich hoch
0,0 %

C höher
0,0 %

D manchmal niedriger und manchmal höher
0,0 %

Anzahl der Stimmen: 0

Abstimmung läuft ...

Clipping eines Kreises