Interactive Visual Computing (IVC)

bzw.

Computergrafik und Bildsynthese (CGB)

(Wintersemester 2011/12)

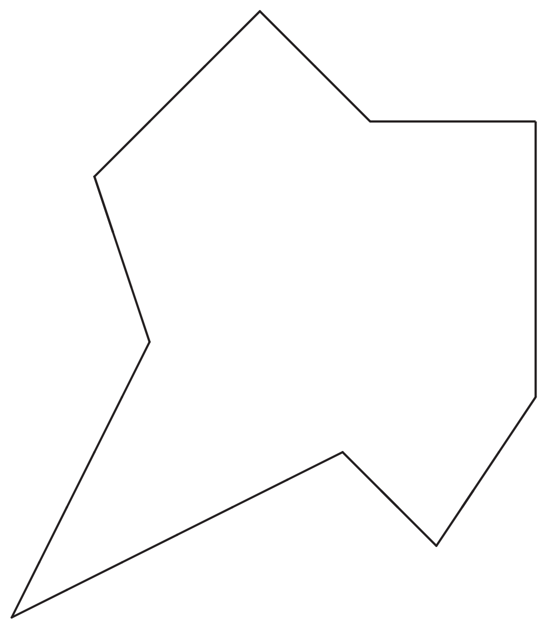
Werner Hansmann

Grafische Algorithmen

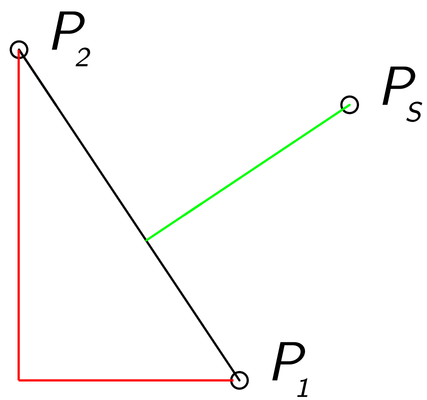
Grafische Algorithmen

Unter "Polygon" sei nachfolgend ein geschlossener Polygonzug im 2 D verstanden.

- Abstand zwischen zwei Punkten bzw. zwischen Punkt und Gerade
- Flächeninhalt eines Polygons
- Konvexe Hülle eines Polygons
- Punkt in Polygon
- Schnittpunkt von zwei Polygonkanten
- Überlappen von Polygonen
- Schraffieren von Polygonen
- Ausschnittsbildung und Kappen



Abstand zwischen zwei Punkten bzw. zwischen Punkt und Gerade



Abstand zwischen zwei Punkten P_1 und P_2:

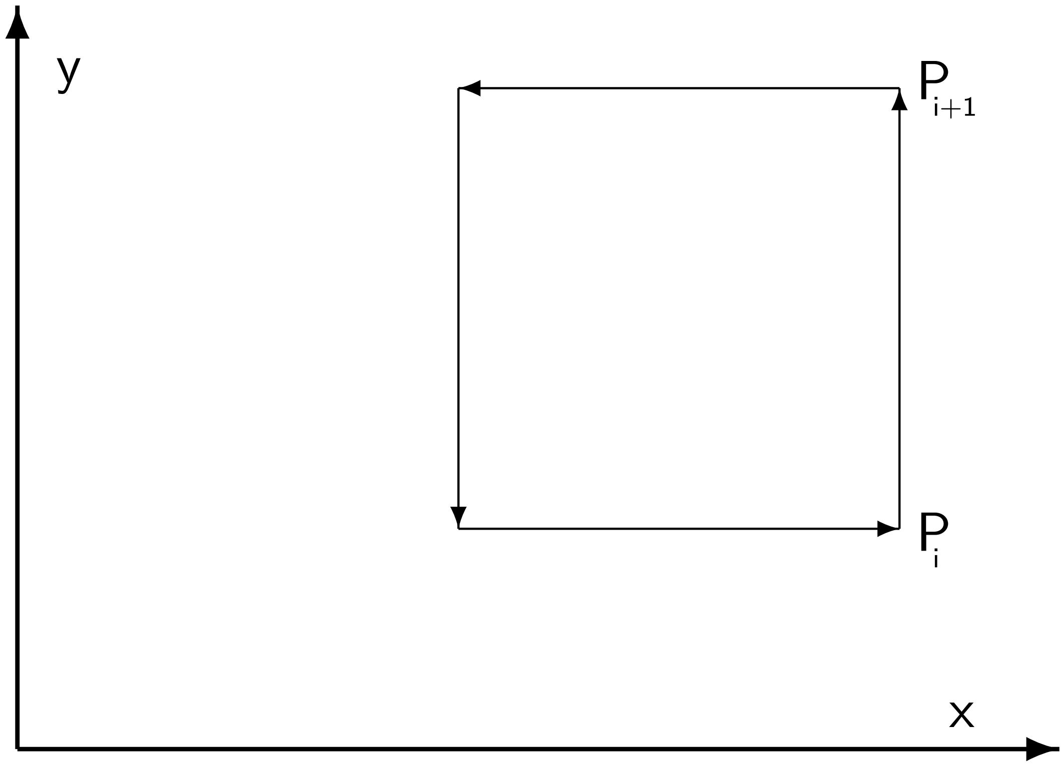
$$A = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (Pythagoras)$$

Abstand zwischen einem Punkt P_s und einer Geraden $ax + by + c = 0$:

$$A = \frac{a}{\sqrt{a^2 + b^2}} x_s + \frac{b}{\sqrt{a^2 + b^2}} y_s + \frac{c}{\sqrt{a^2 + b^2}} \quad (Hessesche\ Normalform)$$

$A > 0$, wenn Koordinatenursprung und P_s auf verschiedenen Seiten der Geraden liegen.

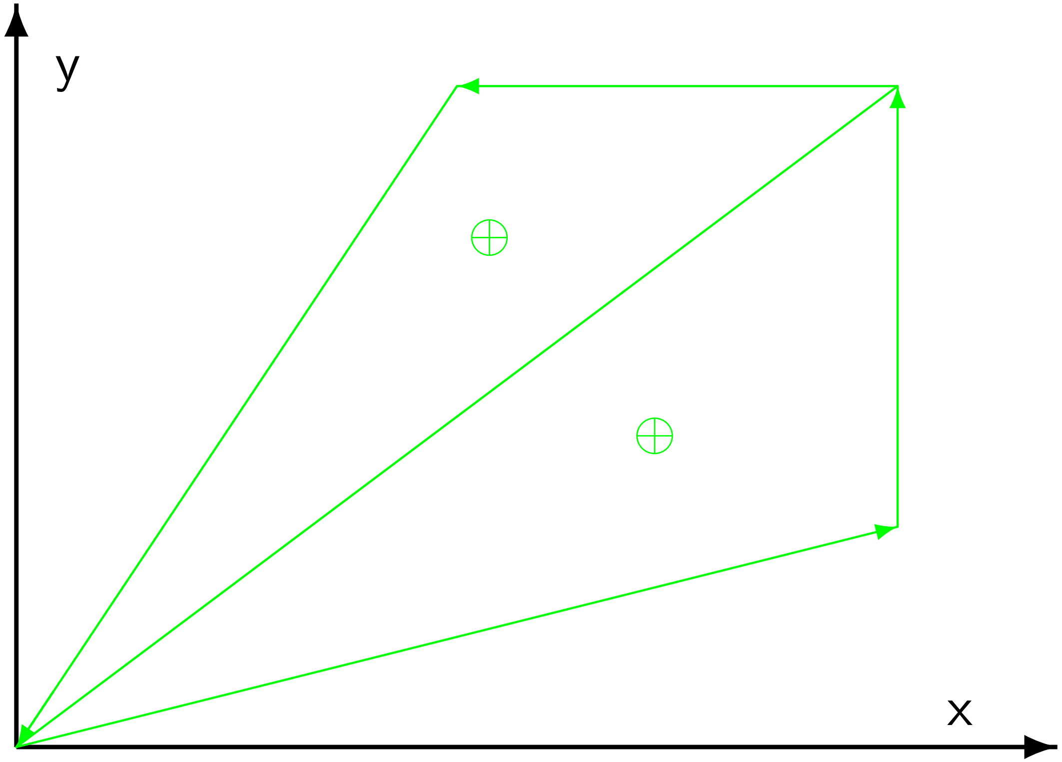
Flächeninhalt eines Polygons (Polygon Area)



bei orientierten Polygonkanten (mathem. positiver Umlaufsinn) und mit $P_n = P_1$ wird

$$F = \frac{1}{2} \sum_{i=1}^{n} (x_i y_{i+1} - x_{i+1} y_i)$$

Flächeninhalt eines Polygons (Polygon Area)

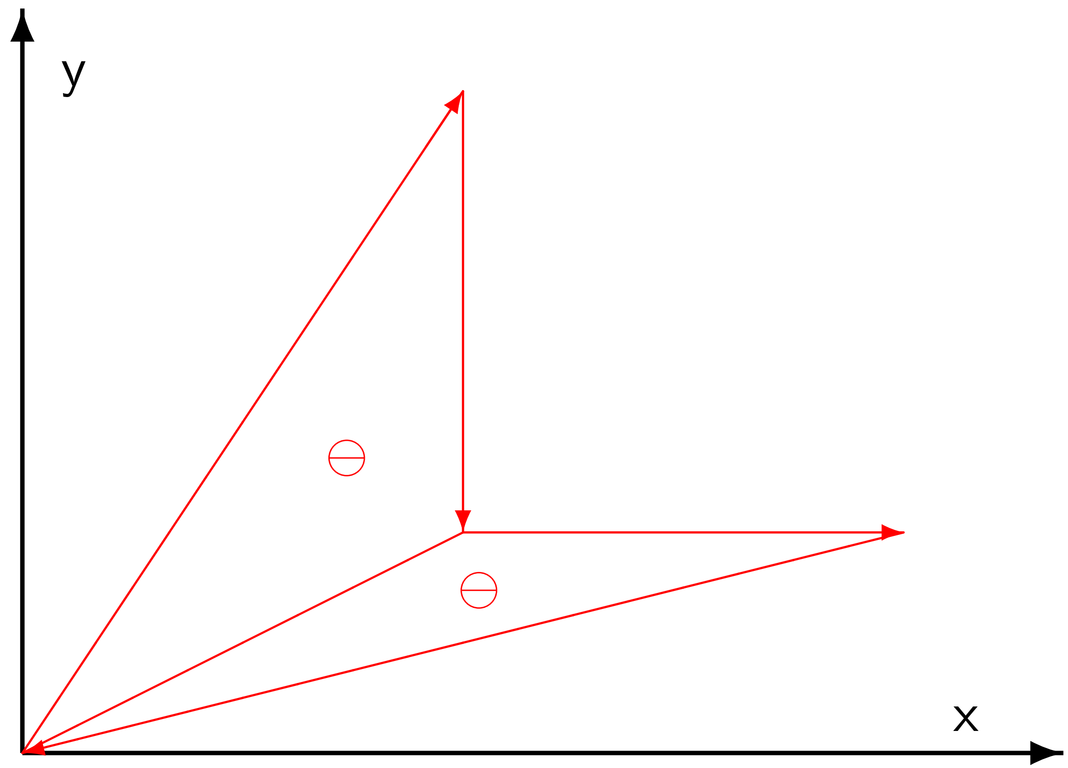


Dreiecke
mit positivem
Flächeninhalt

$$je\,Dreieck:\;\; F = \frac{1}{2}(x_i y_{i+1} - x_{i+1} y_i)$$

Flächeninhalt eines Polygons (Polygon Area)

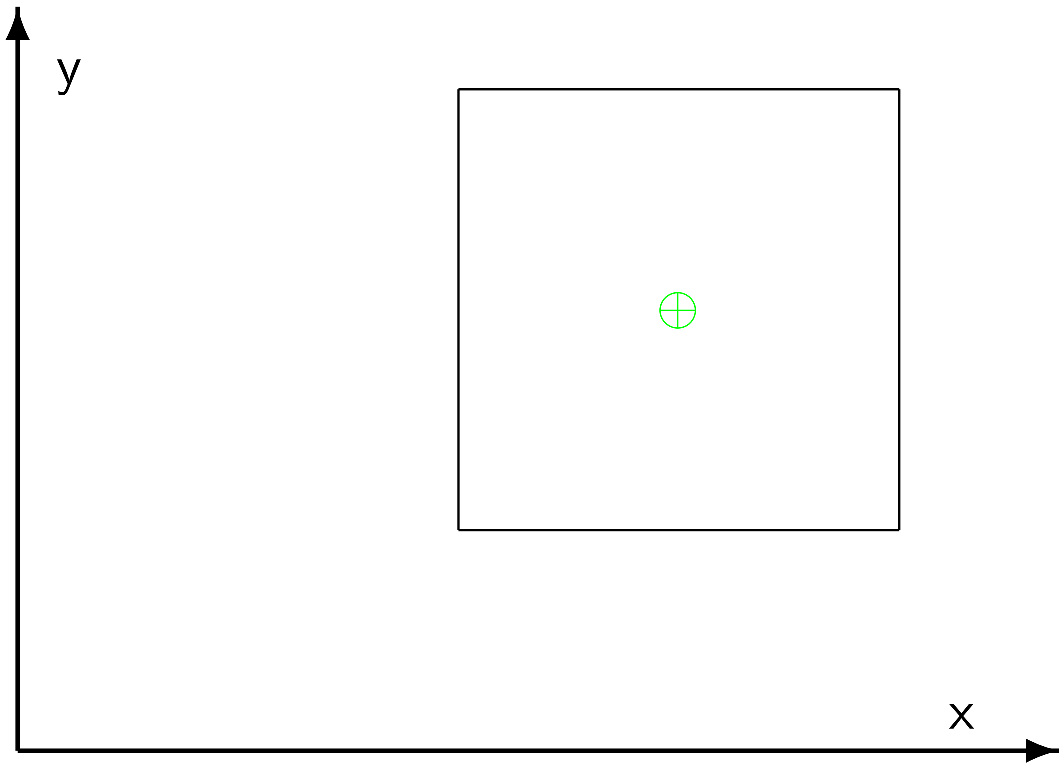
Dreiecke
mit negativem
Flächeninhalt



$$je\,Dreieck:\ F = \frac{1}{2}(x_i y_{i+1} - x_{i+1} y_i)$$

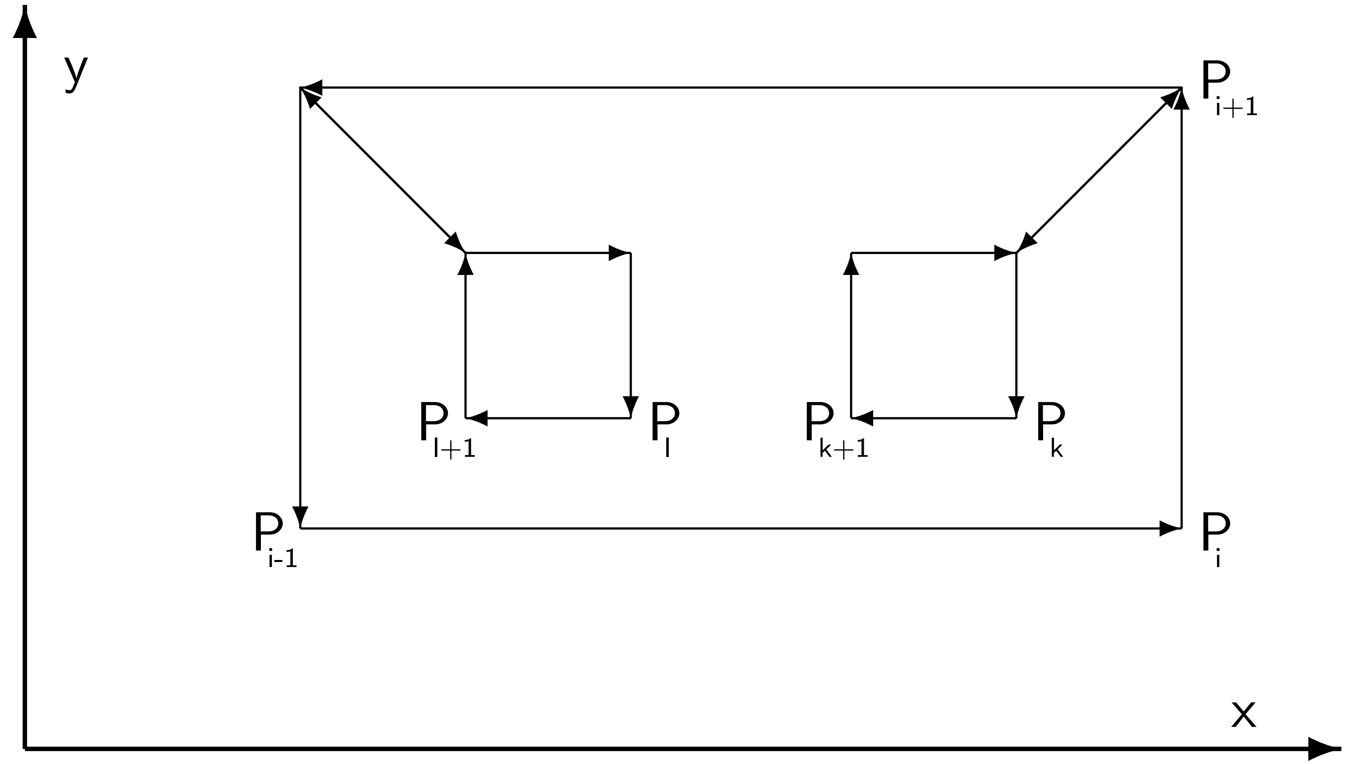
Flächeninhalt eines Polygons (Polygon Area)

verbleibender
positiver
Flächeninhalt



Flächeninhalt eines Polygons (Forts.)

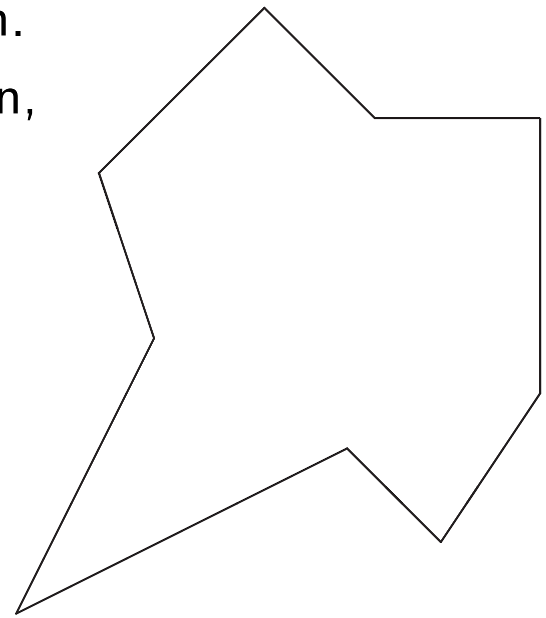
mehrfach zusammenhängende Flächen



Konvexe Hülle eines Polygons (Convex Hull)

Gegeben sei ein beliebig gestaltetes Polygon. Die „konvexe Hülle“ ist das kleinste Polygon, das

- das gegebene Polygon völlig beinhaltet
- keine „Beulen“ nach innen hat, d.h. jede beliebige Gerade schneidet dieses Polygon nur zweimal



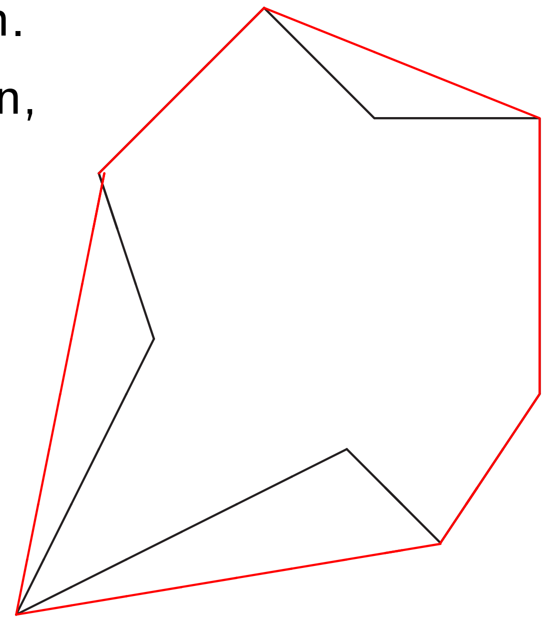
(

Konvexe Hülle eines Polygons (Convex Hull)

Gegeben sei ein beliebig gestaltetes Polygon.
Die „konvexe Hülle“ ist das kleinste Polygon, das

- das gegebene Polygon völlig beinhaltet
- keine „Beulen“ nach innen hat, d.h. jede beliebige Gerade schneidet dieses Polygon nur zweimal

(„Gummibandprinzip“)



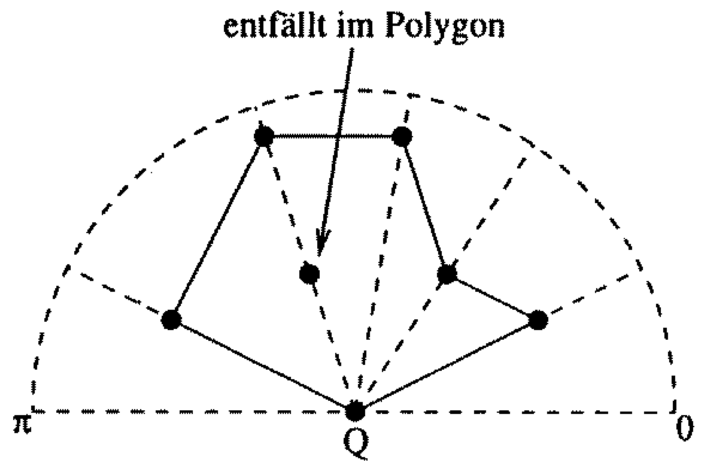
Konvexe Hülle einer Punktmenge (Convex Hull)

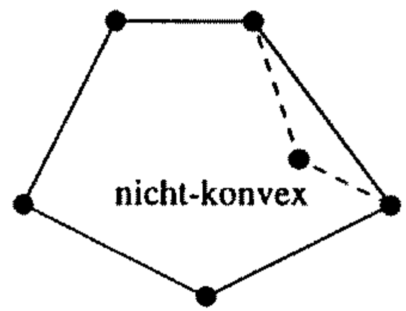
der Algorithmus:

1. auswählen des Referenzpunktes Q (Punkt mit kleinster Ordinate, bei Mehrdeutigkeit zusätzlich kleinster Abszisse)
2. sortieren der Punkte P_i nach Winkel, den sie mit Q und der positiven Abszissenrichtung einschließen (gleicher Winkel für mehrere Punkte: festhalten nur des von Q entferntesten)
3. verbinden der sortierten Punkte zu einem geschlossenen Polygon
4. aussortieren aller nicht-konvexen Eckpunkte

Ist bereits ein geschlossenes Polygon vorgegeben, können sofort die nicht-konvexen Eckpunkte aussortiert werden.

Konvexe Hülle einer Punktmenge (Forts.)





Aussortieren aller nicht-konvexen Eckpunkte

Teile das Polygon an den Eckpunkten P_{min} und P_{max} mit der kleinsten und der größten Abszisse (und ggf. kleinsten Ordinate) in zwei Hälften. Definiere P_{min} zum Ausgangspunkt der Untersuchung der unteren Polygonhälfte und speichere ihn als ersten Punkt des Hüllpolygons.

a) verbinde den Ausgangspunkt mit seinem Nachfolger; ist damit P_{max} erreicht $\longrightarrow$ **fertig**

b) liegen alle auf diesen Nachfolger folgenden Eckpunkte links von der gerichteten Verbindung? – **ja:** speichere diesen Nachfolger als nächsten Punkt des Hüllpolygons und mache ihn zum neuen Ausgangspunkt; fahre fort bei a) – **nein:** verbinde den Ausgangspunkt mit dem Nachfolger des Nachfolgers; fahre fort bei b)

Verfahre analog mit der oberen Polygonhälfte.

Schnittpunkt von zwei Polygonkanten (Edge Intersection)

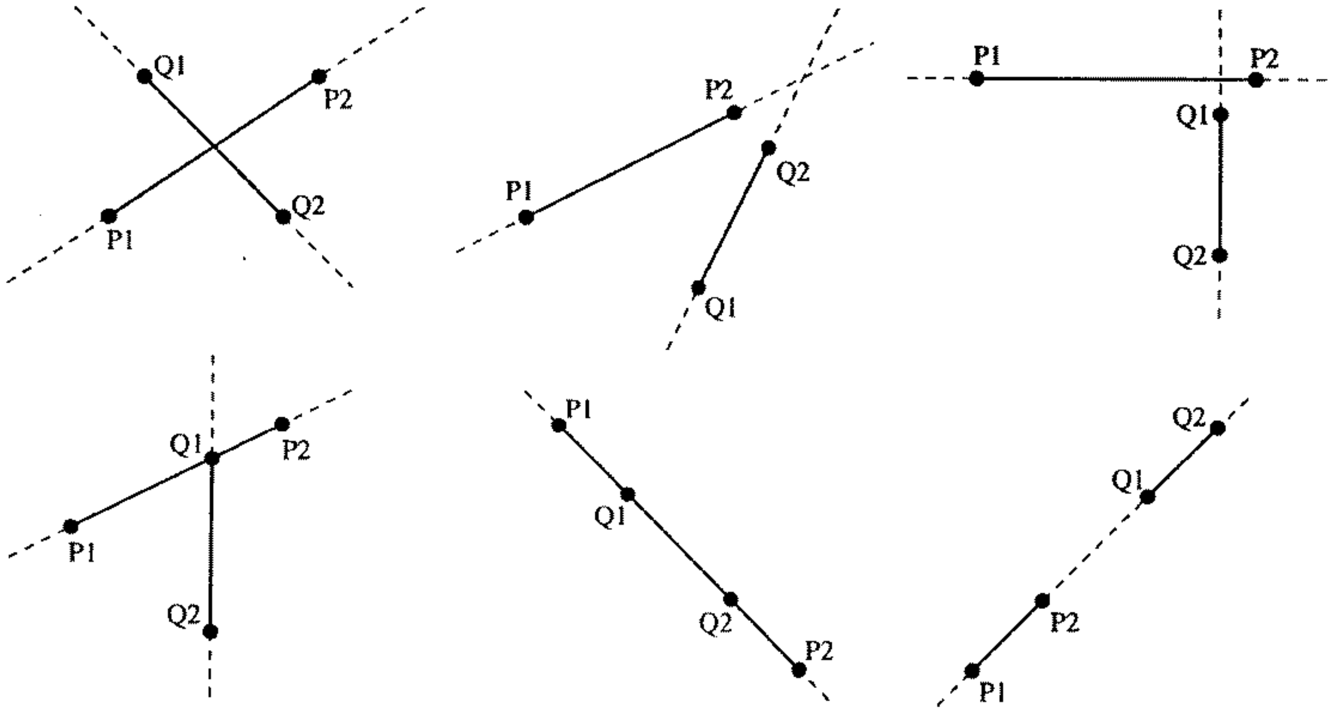
Der Schnittpunkt zweier Geraden

$a_1x + b_1y + c_1 = 0$ und $a_2x + b_2y + c_2 = 0$ ergibt sich zu

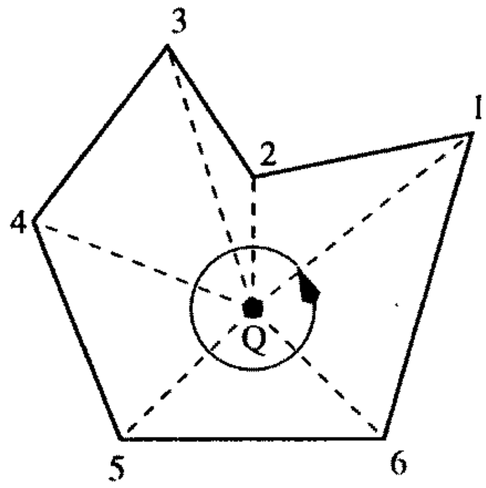
$$x_s = \frac{b_1c_2 - b_2c_1}{a_1b_2 - a_2b_1}, \quad y_s = \frac{c_1a_2 - c_2a_1}{a_1b_2 - a_2b_1}$$

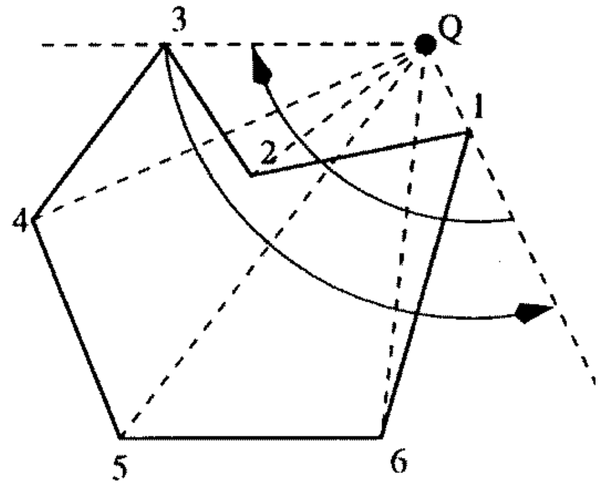
Eine Punktüberschneidung liegt dann vor, wenn bez. beider Kanten die Endpunkte der jew. anderen Kante auf verschiedenen Seiten liegen.

Schnittpunkt von zwei Polygonkanten (mögliche Fälle)



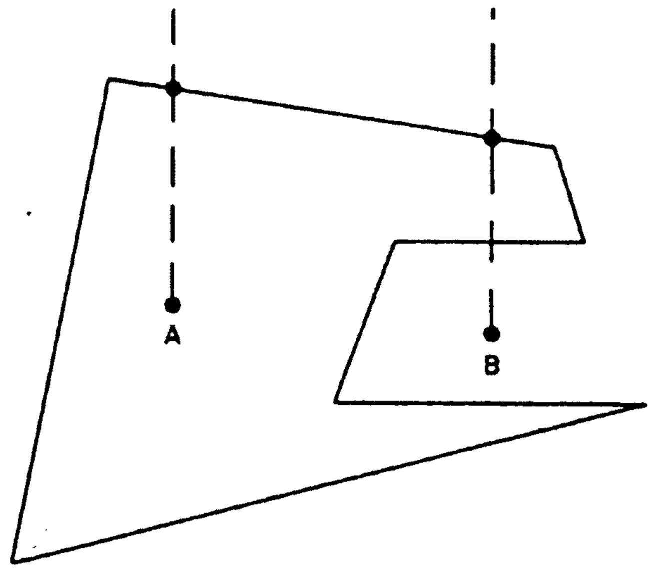
Punkt in Polygon (Point in Polygon) – Winkelsummenmethode





$$Winkelsumme \simeq 2\pi \Rightarrow innerhalb, \;\; Winkelsumme \simeq 0 \Rightarrow außerhalb$$

Punkt in Polygon (Point in Polygon) – Schnittpunktemethode



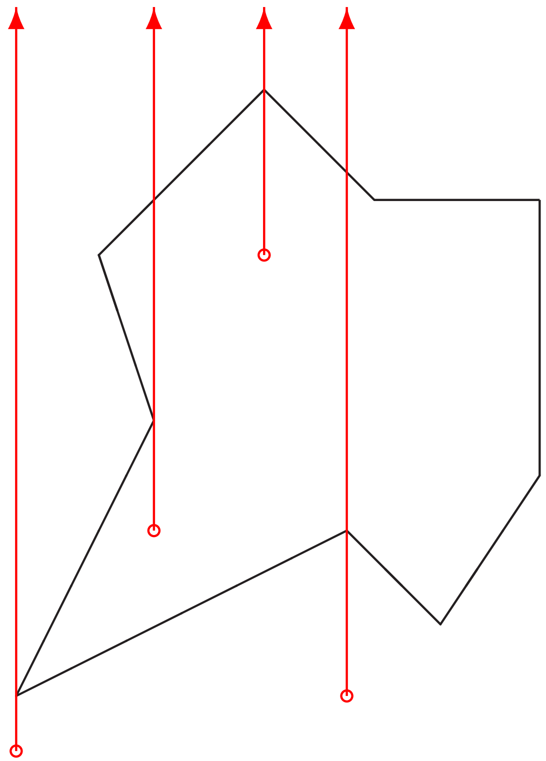
$$ungeradeAnzahlSchnittpkte. \Rightarrow innerhalb, \quad geradeAnz. \Rightarrow außerh.$$

<u>Sonderbehandlung</u> nötig, wenn <u>Halbgerade durch Polygonecke</u> verläuft !

Punkt in Polygon – Schnittpunktemethode (Point in Polygon) – Sonderfälle

Wenn die Halbgerade das Polygon in einem Eckpunkt schneidet, muss entschieden werden, ob dieser Schnittpunkt einfach oder gar nicht zählt:

- liegen beide Polygonkanten auf derselben Seite der Halbgeraden, zählt der Schnittpunkt nicht,
- liegen sie auf unterschiedlichen Seiten, zählt er einfach.



Überlappen von Polygonen (Polygon Overlap)

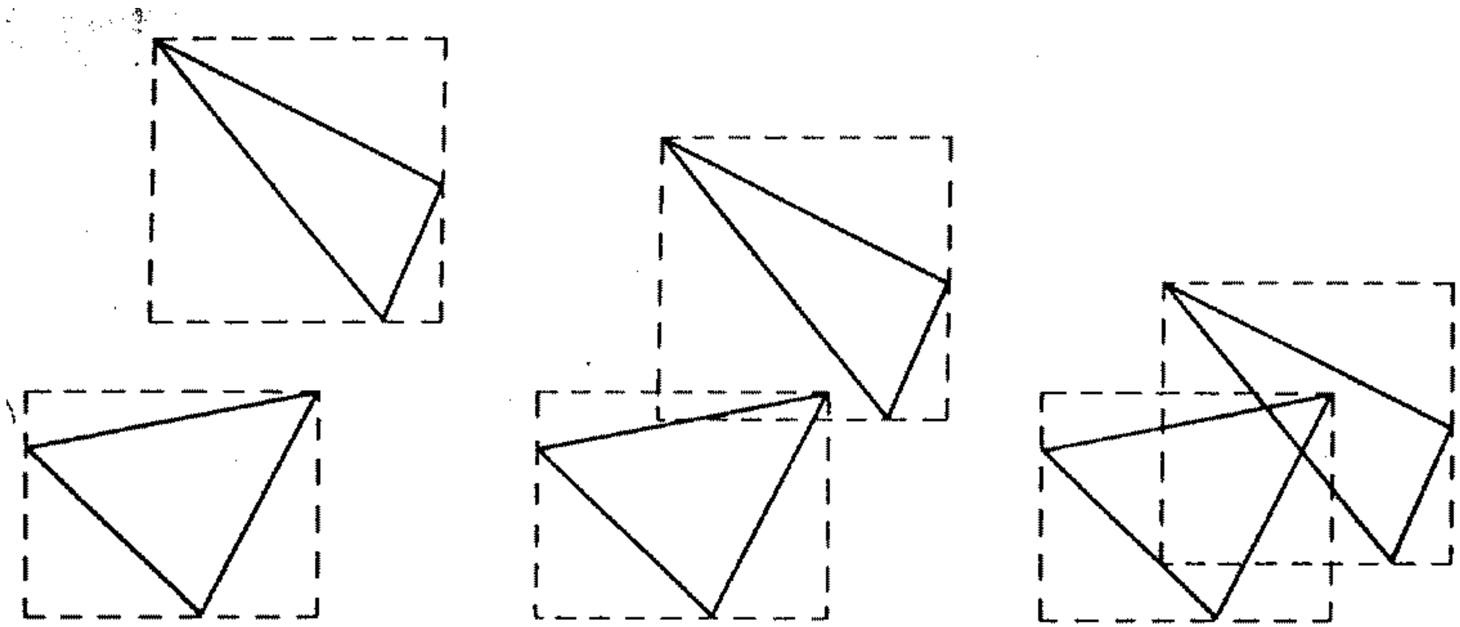
Vorauswahl durch **Minimax-Test** (minimum box test):
Ist

1. x_{min} des einen Polygons größer als x_{max} des anderen oder
2. y_{min} des einen Polygons größer als y_{max} des anderen,

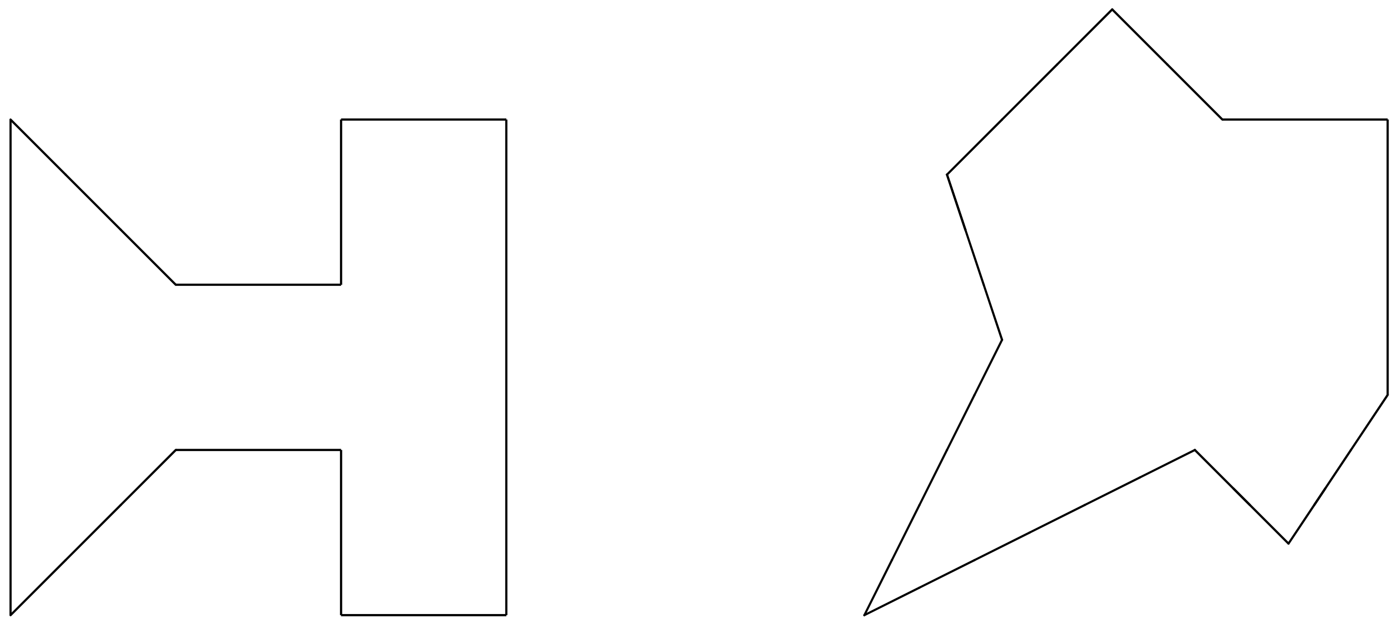
dann überlappen sich die Polygone nicht.

Andernfalls müssen die Polygonkanten jeweils direkt auf Überschneiden geprüft werden (paarweiser Minimax-Test für die Kanten);
ggf. muß der Schnittpunkt ermittelt werden.

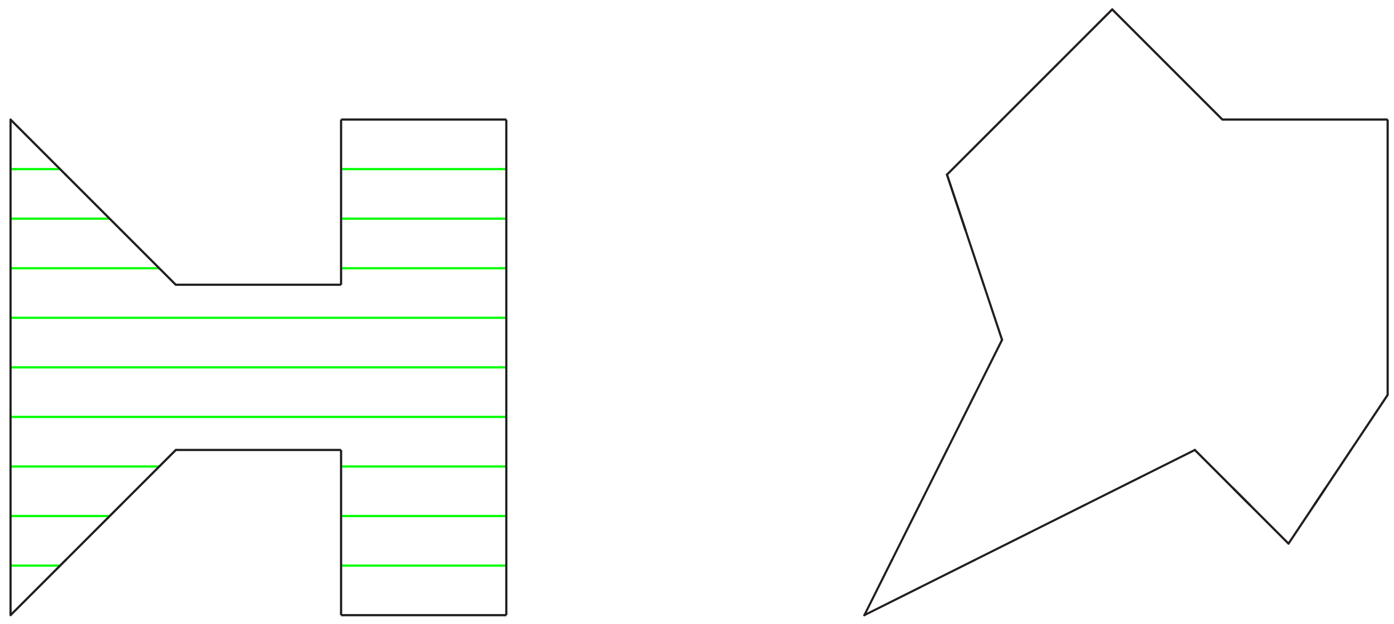
Überlappen von Polygonen (mögliche Fälle)



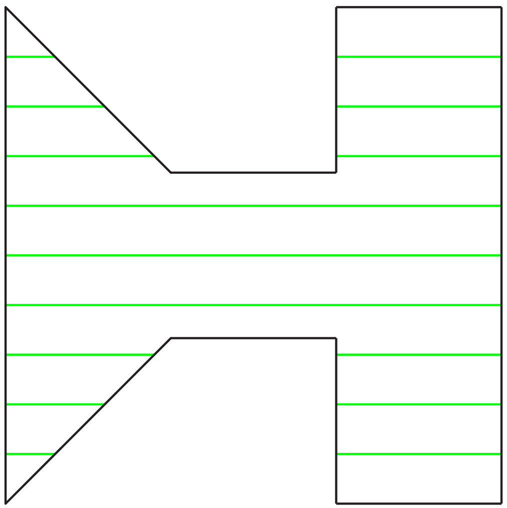
Schraffieren von Polygonen (Crosshatching)

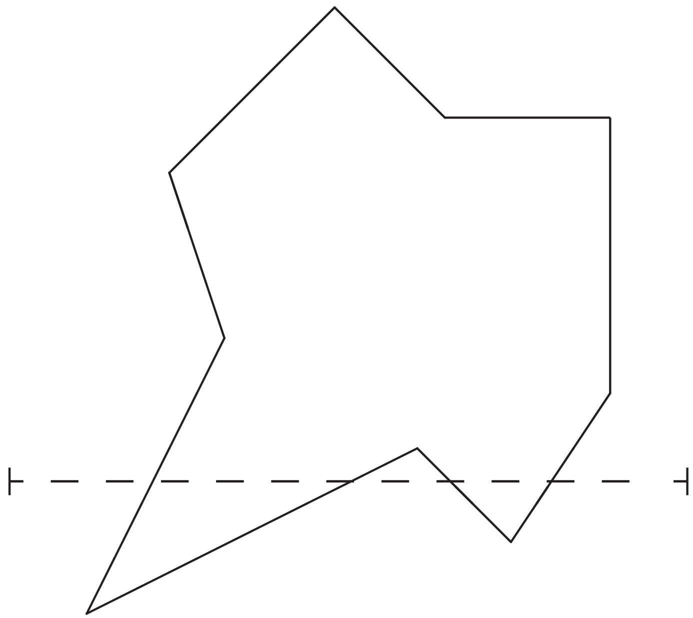


Schraffieren von Polygonen (Crosshatching)



Schraffieren von Polygonen (Crosshatching)





Schraffieren von Polygonen (der Algorithmus)

1. ermitteln der extremen Polygon-Ordinaten y_{min} und y_{max}
2. beginnend bei y_{min}:
   (a) inkrementieren der Schraffurlinienordinate um den Schraffurlinienabstand
   (b) bestimmen der Schnittpunkte der Schraffurlinie mit den Polygonkanten
   (c) sortieren der Schnittpunkte nach aufsteigender Abszisse
   (d) zeichnen einer Verbindung der sortierten Schnittpunkte in aufsteigender Reihenfolge, und zwar jeweils
   - "zum ungeraden Punkt" unsichtbar und
   - "zum geraden Punkt" sichtbar

   fortsetzen bei (a) bis y_{max} erreicht

Schraffieren von Polygonen (der Algorithmus)

Schraffur unter einem Winkel α:

1. drehen der Polygon-Eckpunktkoordinaten um den Winkel $-\alpha$
2. ermitteln der extremen Polygon-Ordinaten y_{min} und y_{max}
3. beginnend bei y_{min}:
   (a) inkrementieren der Schraffurlinienordinate
   (b) bestimmen der Schnittpunkte der (jetzt horizontalen) Schraffurlinie mit den Polygonkanten
   (c) sortieren der Schnittpunkte nach aufsteigender Abszisse
   (d) drehen der sortierten Schnittpunkte um α
   (e) zeichnen einer Verbindung der Schnittpunkte in aufsteigender Reihenfolge (s.o.)

   fortsetzen bei (a) bis y_{max} erreicht

Ausschnittsbildung und Kappen (Windowing and Clipping)

- **Window:** abzubildender Ausschnitt aus dem Weltkoordinatensystem
- **Viewport:** Teil des Zielkoordinatensystems, auf den das Window abgebildet werden soll (Geräteseite)
- Transformation zwischen Koordinatensystemen (im GKS zweistufig:
  1. Normierungstransformation:
     Weltkoordinaten $\Rightarrow$ Normalized Device Coordinates (NDC)
  2. Gerätetransformation:
     Normalized Device Coordinates (NDC) $\Rightarrow$ Geräte-Viewport)

  Für jede Anwendung können mehrere solche Transformationen spezifiziert werden

Ausschnittsbildung und Kappen (Windowing and Clipping) (Forts.)

Ziele des Windowing:

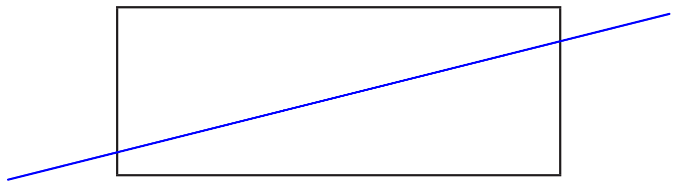
- Geräteunabhängigkeit
  (Festlegung des Formats für das Ausgabegerät ist völlig entkoppelt)
- ein Window kann in unterschiedlicher Weise auf mehrere Geräte ausgegeben werden
- mehrere Windows können einen Viewport nutzen

Window und Viewport werden in der Regel als achsparallele Rechtecke realisiert.

Ausschnittsbildung und Kappen (Windowing and Clipping) (Forts.)

Clipping-Algorithmen:

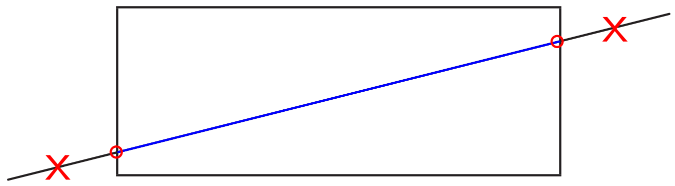
- Algorithmen, die Darstellungselemente, die ganz oder teilweise außerhalb des Viewports liegen, abschneiden
  (setzen meist schon bei der Window-Viewport-Transformation an; die Gerätetransformation für so „abgeschnittene“ Objektteile kann dann entfallen)



Ausschnittsbildung und Kappen (Windowing and Clipping) (Forts.)

Clipping-Algorithmen:

- Algorithmen, die Darstellungselemente, die ganz oder teilweise außerhalb des Viewports liegen, abschneiden
  (setzen meist schon bei der Window-Viewport-Transformation an; die Gerätetransformation für so „abgeschnittene“ Objektteile kann dann entfallen)



Ausschnittsbildung und Kappen (Windowing and Clipping) (Forts.)

Cohen-Sutherland-Algorithmus:

- Aufteilung der Bildebene in neun Regionen (Viewport im Zentrum)

1001	1000	1010
0001	0000	0010
0101	0100	0110

- Zuordnung eines 4-Bit-Schlüssels zu jeder Region; Bedeutung:
  1. Bit: Punkt liegt links vom linken Rand
  2. Bit: Punkt liegt rechts vom rechten Rand
  3. Bit: Punkt liegt unterhalb der unteren Randlinie
  4. Bit: Punkt liegt über der oberen Randlinie des Viewports

Ausschnittsbildung und Kappen (Windowing and Clipping) (Forts.)

Cohen-Sutherland-Algorithmus: Auswertung 1. Schritt

eine Gerade liegt völlig

- innerhalb des Viewports, wenn der Code für beide Endpunkte 0000 ist
- außerhalb des Viewports, wenn der Durchschnitt (logisches UND) der Codes beider Endpunkte verschieden von Null ist

andernfalls wird der Schnittpunkt zwischen Gerade und der kreuzenden Randlinie berechnet

Ausschnittsbildung und Kappen (Windowing and Clipping) (Forts.)

Cohen-Sutherland-Algorithmus: Auswertung 2. Schritt

ist m die Steigung der Geraden, ergibt sich der Schnittpunkt mit

- dem linken Rand zu: $x_s = x_l$ und $y_s = y_1 + (x_l - x_1)m$
- dem rechten Rand zu: $x_s = x_r$ und $y_s = y_1 + (x_r - x_1)m$
- dem unteren Rand zu: $x_s = x_1 + \frac{y_u - y_1}{m}$ und $y_s = y_u$
- dem oberen Rand zu: $x_s = x_1 + \frac{y_o - y_1}{m}$ und $y_s = y_o$

der Schnittpunkt wird als neuer Endpunkt der Geraden entsprechend codiert; danach wird beim 1. Schritt fortgefahren

horizontale bzw. vertikale Geraden erfahren Sonderbehandlung

Ausschnittsbildung und Kappen (Windowing and Clipping) (Forts.)

Liang-Barsky-Algorithmus (Annahmen):

- die (gerichteten) Begrenzungskanten des Viewport zerlegen die Bildebene jeweils in zwei Halbebenen, wobei der **sichtbare** Teil auf einer Seite liegt.
- der **Durchschnitt** der Halbebenen mit dem sichtbaren Teil ergibt den (sichtbaren) Viewport

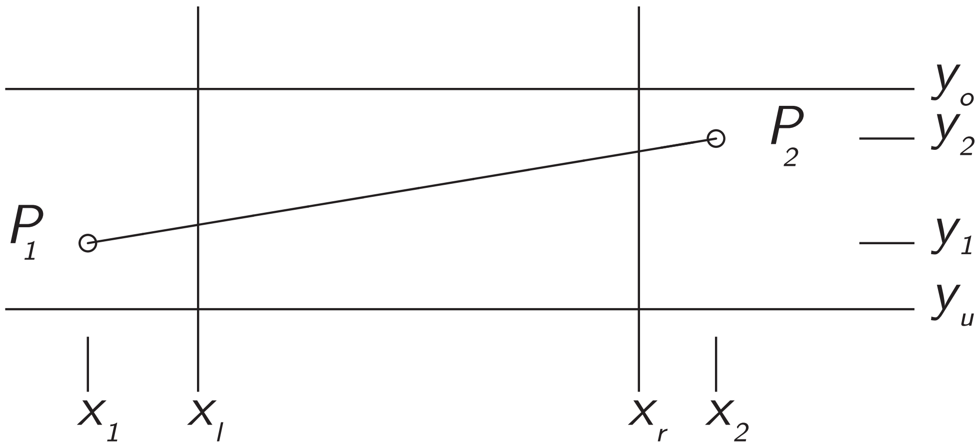
Ausschnittsbildung und Kappen (Windowing and Clipping) (Forts.)

Liang-Barsky-Algorithmus (Forts.):

- eine Gerade zwischen zwei Punkten ist gegeben in Parameterform:

$$x = x_1 + t \cdot (x_2 - x_1)$$
$$y = y_1 + t \cdot (y_2 - y_1) \quad \text{und} \quad 0 \leq t \leq 1$$

- die Viewportgrenzen sind x_l, x_r, y_u und y_o



Ausschnittsbildung und Kappen (Windowing and Clipping) (Forts.)

Liang-Barsky-Algorithmus (Forts.):

- im Viewport gilt mit

$$p_1 = x_1 - x_2, \qquad q_1 = x_1 - x_l,$$

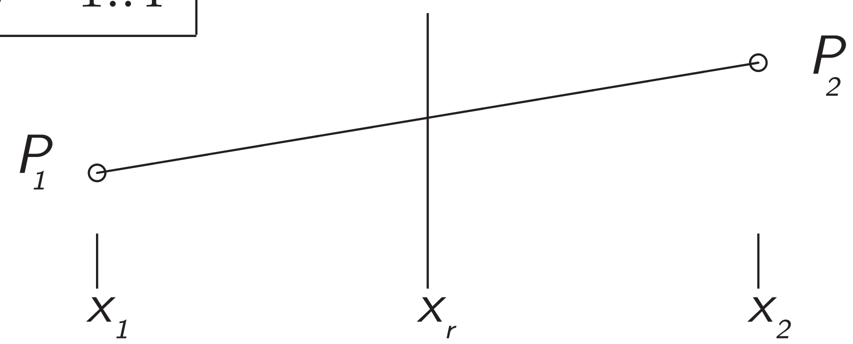
$$p_2 = x_2 - x_1, \qquad q_2 = x_r - x_1,$$

$$p_3 = y_1 - y_2, \qquad q_3 = y_1 - y_u,$$

$$p_4 = y_2 - y_1 \quad \text{und} \quad q_4 = y_o - y_1:$$

$$\boxed{p_i \cdot t \ \leq \ q_i \ \text{ mit } \ i = 1..4}$$

Beispiel:



Ausschnittsbildung und Kappen (Windowing and Clipping) (Forts.)

Liang-Barsky-Algorithmus (Forts.):

Die Ungleichungen $p_i \cdot t \leq q_i$ beschreiben je eine der sichtbaren Halbebenen, die durch die Begrenzungskanten des Viewports definiert sind. Es gilt:

- wenn $q_i \geq 0 \quad \forall i$ liegt P_1 im sichtbaren Bereich
- wenn $p_i > q_i \quad \forall i$ verläßt die Gerade den sichtbaren Bereich
- wenn $p_i < q_i \quad \forall i$ geht die Gerade in den sichtbaren Bereich
- wenn $p_i = 0$ verläuft die Gerade parallel zur i-ten Begrenzungskante
  (mit $i = 1$: links, $i = 2$: rechts, $i = 3$: unten, $i = 4$: oben)

Ausschnittsbildung und Kappen (Windowing and Clipping) (Forts.)

Liang-Barsky-Algorithmus (Forts.):

Gesucht: die Parameterwerte t_1 und t_2 für den sichtbaren Teil der Geraden; Schnittpunkte der Geraden mit den Begrenzungskanten ergeben sich für $p_i \neq 0$ jeweils für den Parameterwert $t = q_i/p_i$

Für die beiden gesuchten Parameterwerte folgt somit:

$$t_1 = max(\{q_i/p_i \mid p_i < q_i,\ i = 1, .., 4\} \cup \{0\})$$

$$t_2 = min(\{q_i/p_i \mid p_i > q_i,\ i = 1, .., 4\} \cup \{1\})$$

Einsetzen dieser Werte in die Parametergleichung der Geraden ergibt die Schnittpunkte mit dem Rand des Viewport

Ausschnittsbildung und Kappen (Windowing and Clipping) (Forts.)

Vergleich der Algorithmen:

- Cohen-Sutherland kommt mit deutlich weniger Gleitkomma-Operationen aus
- Liang und Barsky haben gemessen (4 Mio zufällige Clip-Operationen), daß ihr Verfahren um 1/3 schneller ist