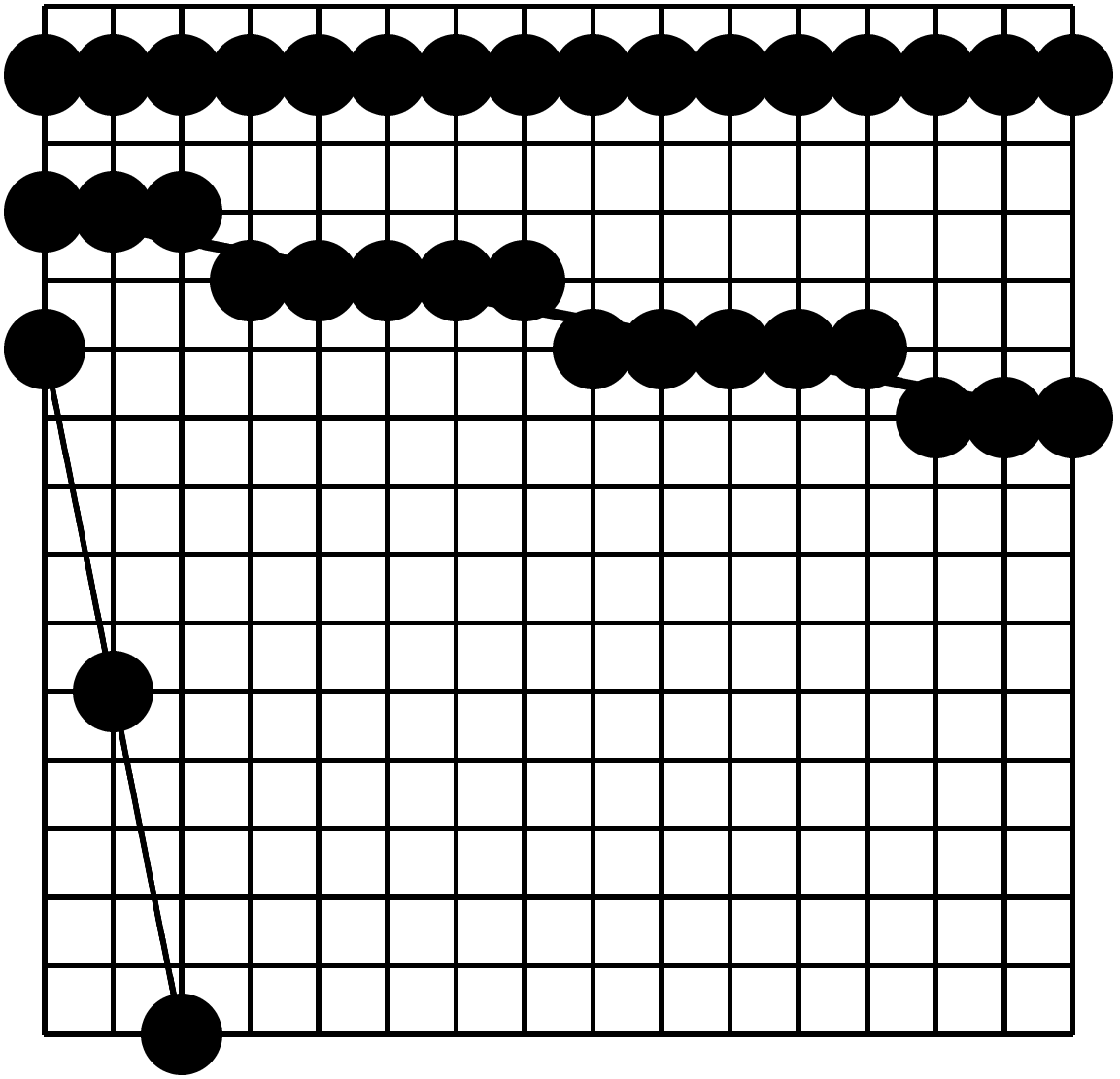
Zeichnen von Geraden

Naiver Geradenalgorithmus

```
// Verbinden der Gitterpunkte (x0,y0) und
// (x1,y1) durch ein Geradenstueck
void drawLine(int x0, int y0, int x1, int y1)
{
  int x;
  double dy = y1 - y0;
  double dx = x1 - x0;
  double m  = dy / dx;
  double y  = y0;

  for (x=x0; x<=x1; x++)
  {
    drawPixel(x, round(y));
    y = y + m;   //oder: y = y0 + m*(x - x0);
  }
}
```

Zeichnen von Geraden



Zeichnen von Geraden

Liegt die Steigung der Geraden dem Betrage nach nicht zwischen 0 und 1, so sollten bei der Berechnung die Rollen von x- und y-Achse vertauscht werden.

Pro Bild müssen i.A. sehr viele Geradensegmente gezeichnet werden.

Daher sollte der Algorithmus zum Geradenzeichnen möglichst effizient sein.

Der **Bresenham-** oder **Mittelpunktalgorithmus** verzichtet auf Fließkommaoperationen und zeichnet die Gerade allein auf der Basis von Integeroperationen.

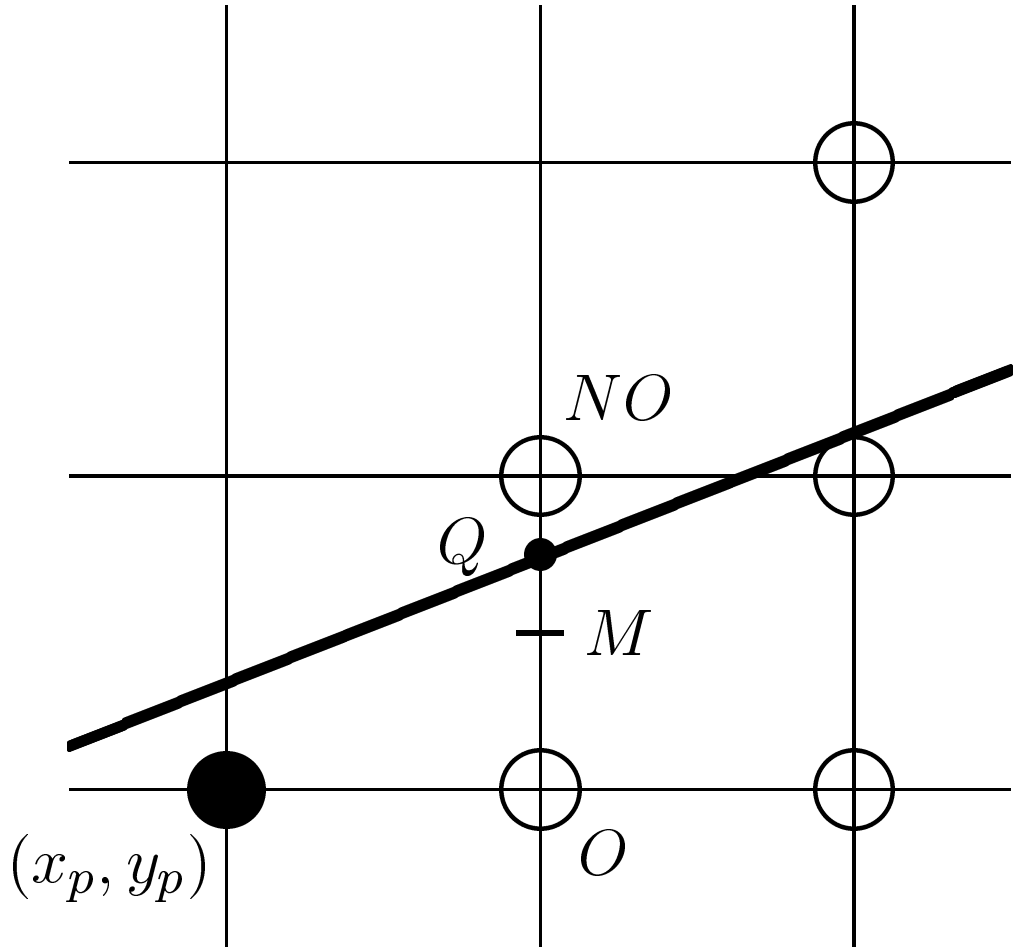
Der Mittelpunktalgorithmus

Wir betrachten im Folgenden zunächst nur Geraden mit einer Steigung zwischen 0 und 1.

Wurde bei einer derartigen Geraden ein Pixel gezeichnet, kommen für das nächste zu zeichnende Pixel jeweils nur zwei Pixel in Frage:

- das Pixel rechts daneben oder
- das Pixel diagonal darüber

Der Mittelpunktalgorithmus



Der Mittelpunktalgorithmus

Die Entscheidung, welches der beiden Pixel gezeichnet werden muss, wird anhand des Mittelpunktes zwischen den beiden Pixelkandidaten entschieden:

- Liegt der Mittelpunkt unterhalb der Geraden, muss das obere Pixel gezeichnet werden.
- Liegt der Mittelpunkt oberhalb der Geraden, muss das untere Pixel gezeichnet werden.

Der Mittelpunktalgorithmus

Geradengleichung:

$$y = f(x) = m \cdot x + b$$

in impliziter Form:

$$F(x, y) = Ax + By + C \qquad (a \geq 0)$$

bzw.

$$F(x, y) = m \cdot x - y + b = 0$$

Der Mittelpunktalgorithmus

- $F(x,y) = 0 \Leftrightarrow (x,y)$ liegt auf der Geraden
- $F(x,y) < 0 \Leftrightarrow (x,y)$ liegt oberhalb der Geraden
- $F(x,y) > 0 \Leftrightarrow (x,y)$ liegt unterhalb der Geraden

Setzt man für (x,y) die Koordinaten des entsprechenden Mittelpunktes (x_M, y_M) ein, so gilt:

- $F(x_M, y_M) < 0 \Leftrightarrow$ Das Pixel O ist zu zeichnen.
- $F(x_M, y_M) > 0 \Leftrightarrow$ Das Pixel NO ist zu zeichnen.

Im Falle $F(x_M, y_M) = 0$ muss eine (einheitliche) Rundungsentscheidung gefällt werden (z.B. immer das obere Pixel).

Der Mittelpunktalgorithmus

zu zeichnendes Geradenstück zwischen den Pixeln (x_0, y_0) und (x_1, y_1)

Geradengleichung:

$$\frac{y - y_0}{x - x_0} = \frac{y_1 - y_0}{x_1 - x_0}$$

$$y = \frac{dy}{dx}x + y_0 - \frac{dy}{dx}x_0$$

mit $dx = x_1 - x_0$ und $dy = y_1 - y_0$

Der Mittelpunktalgorithmus

implizite Form:

$$0 \;=\; \frac{dy}{dx}x - y + y_0 - \frac{dy}{dx}x_0$$

bzw.

$$F(x,y) \;=\; dy \cdot x - dx \cdot y + C \;=\; 0$$

mit

$$C \;=\; dx \cdot y_0 - dy \cdot x_0$$

Der Mittelpunktalgorithmus

Einsetzen des Mittelpunktes $M = (x_M, y_M)$ mit ganzzahligem Wert x_M und

$$y_M = y_M^{(0)} + \frac{1}{2}$$

mit ganzzahligem Wert $y_M^{(0)}$ erfordert Fließkommaarithmerik.

Multiplikation mit dem Faktor 2:

Der Mittelpunktalgorithmus

$$\tilde{F}(x,y) \;=\; 2 \cdot dy \cdot x - dx \cdot 2 \cdot y + 2 \cdot C \;=\; 0$$

Einsetzen des Mittelpunktes $M = (x_M, y_M)$ mit

$$y_M \;=\; y_M^{(0)} + \frac{1}{2},$$

wobei $x_M, y_M^{(0)}$ ganzzahlig sind:

$$\tilde{F}(x_M, y_M) \;=\; 2 \cdot dy \cdot x_M - dx \cdot (2 \cdot y_M^{(0)} + 1) + 2 \cdot C \;=\; 0$$

rein ganzzahlige Operationen!

Der Mittelpunktalgorithmus

Weitere Verbesserung in der Originalversion des Mittelpunktalgorithmus:

inkrementelle Berechnung der Werte $\tilde{F}(x_M, y_M)$ bzw. $F(x_M, y_M) \longrightarrow$
nur ganzzahlige Addition, keine Multiplikation erforderlich

Der Mittelpunktalgorithmus

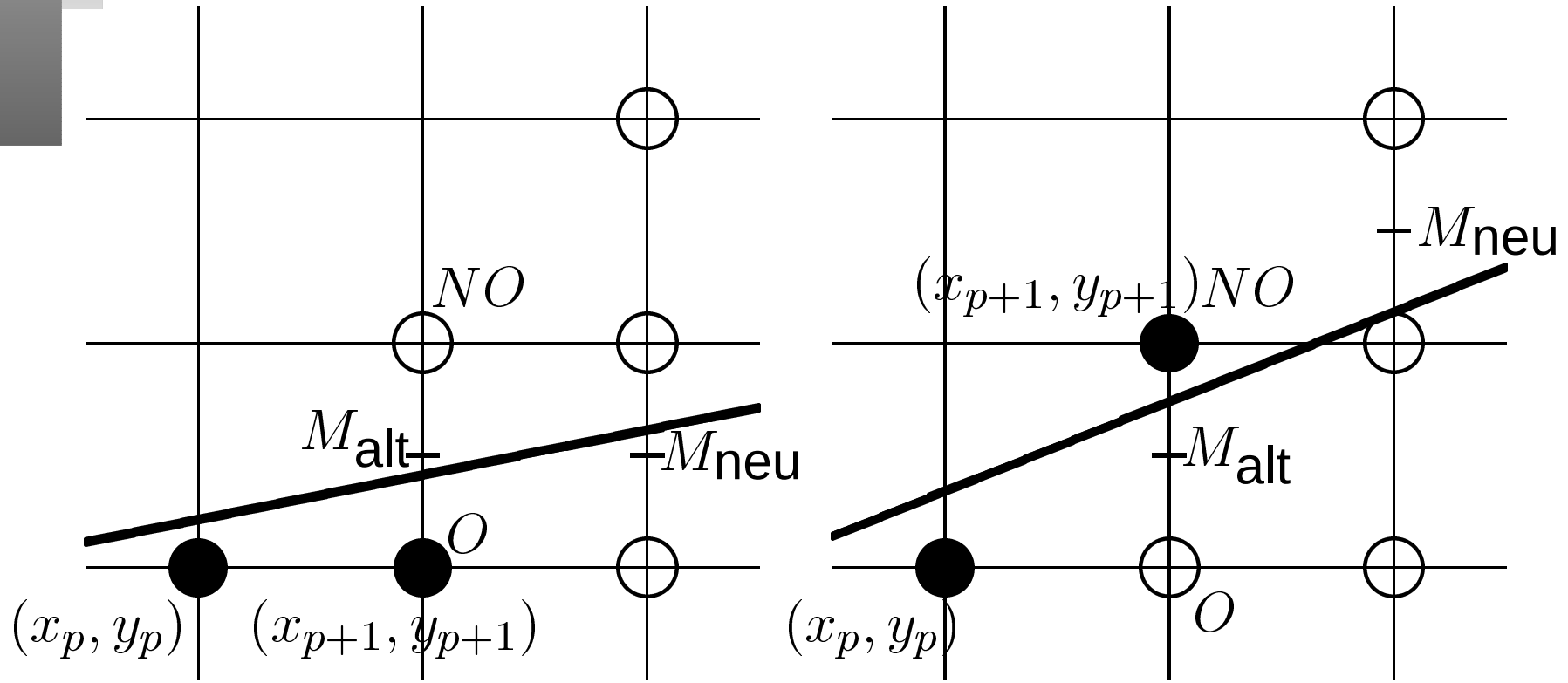
inkrementelle Berechnung der Entscheidungsvariablen

$$d = F(x_M, y_M) = dy \cdot x_M - dx \cdot y_M + C$$

Wie ändert sich d?

Der Mittelpunktalgorithmus

Fallunterscheidung:



Der Mittelpunktalgorithmus

Fall 1: O, d.h. $(x_{p+1}, y_{p+1}) = (x_p + 1, y_p)$ war das nach (x_p, y_p) zu zeichnende Pixel.

zum Zeichnen des Pixels (x_{p+2}, y_{p+2}) zu betrachtender Mittelpunkt:

$$M_{\text{neu}} = \left(x_p + 2, y_p + \frac{1}{2}\right)$$

Der Mittelpunktalgorithmus

$$d_{\text{neu}} = F\left(x_p + 2, y_p + \frac{1}{2}\right) = dy \cdot (x_p + 2) - dx \cdot \left(y_p + \frac{1}{2}\right) + C$$

$$d_{\text{alt}} = F\left(x_p + 1, y_p + \frac{1}{2}\right) = dy \cdot (x_p + 1) - dx \cdot \left(y_p + \frac{1}{2}\right) + C$$

$$\Delta_O = d_{\text{neu}} - d_{\text{alt}} = dy.$$

Der Mittelpunktalgorithmus

Fall 2: NO, d.h. $(x_{p+1}, y_{p+1}) = (x_p + 1, y_p + 1)$ war das nach (x_p, y_p) zu zeichnende Pixel.

zum Zeichnen des Pixels (x_{p+2}, y_{p+2}) zu betrachtender Mittelpunkt:

$$M_{\text{neu}} = \left(x_p + 2, y_p + \frac{3}{2}\right)$$

Der Mittelpunktalgorithmus

$$d_{\text{neu}} = F\left(x_p + 2, y_p + \frac{3}{2}\right) = dy \cdot (x_p + 2) - dx \cdot \left(y_p + \frac{3}{2}\right) + C$$

$$d_{\text{alt}} = F\left(x_p + 1, y_p + \frac{1}{2}\right) = dy \cdot (x_p + 1) - dx \cdot \left(y_p + \frac{1}{2}\right) + C$$

$$\Delta_{NO} = d_{\text{neu}} - d_{\text{alt}} = dy - dx$$

Der Mittelpunktalgorithmus

$$\Delta = \begin{cases} dy & \text{falls } O \text{ gewählt wurde} \\ dy - dx & \text{falls } NO \text{ gewählt wurde} \end{cases}$$

d.h.

$$\Delta = \begin{cases} dy & \text{falls } d_{\text{alt}} < 0 \\ dy - dx & \text{falls } d_{\text{alt}} > 0 \end{cases}$$

Δ ist immer ganzzahlig, so dass sich die Entscheidungsvariable d nur um ganzzahlige Werte ändert.

Der Mittelpunktalgorithmus

Initialisierung von d: Startpixel: (x_0, y_0)

erster zu betrachtender Mittelpunkt: $\left(x_0 + 1, y_0 + \frac{1}{2}\right)$

$$
\begin{aligned}
d_{\text{init}} &= F\left(x_0 + 1, y_0 + \frac{1}{2}\right) \\
&= dy \cdot (x_0 + 1) - dx \cdot \left(y_0 + \frac{1}{2}\right) + C \\
&= dy \cdot x_0 - dx \cdot y_0 + C + dy - \frac{dx}{2} \\
&= F(x_0, y_0) + dy - \frac{dx}{2} \\
&= dy - \frac{dx}{2}
\end{aligned}
$$

Der Mittelpunktalgorithmus

d_{init} ist i.A. nicht ganzzahlig.

Betrachte statt der Entscheidungsvariablen d die Entscheidungsvariable $D = 2 \cdot d$:

$$D = \hat{F}(x,y) = 2 \cdot F(x,y) = 2 \cdot dy \cdot x - 2 \cdot dx \cdot y + 2 \cdot C = 0$$

D ist immer ganzzahlig.

Der Mittelpunktalgorithmus

$$D_{\text{init}} = 2 \cdot dy - dx$$

$$D_{\text{neu}} = D_{\text{alt}} + \Delta \quad \text{mit}$$

$$\Delta = \begin{cases} 2 \cdot dy & \text{falls } D_{\text{alt}} < 0 \\ 2 \cdot (dy - dx) & \text{falls } D_{\text{alt}} > 0 \end{cases}$$

Der Mittelpunktalgorithmus

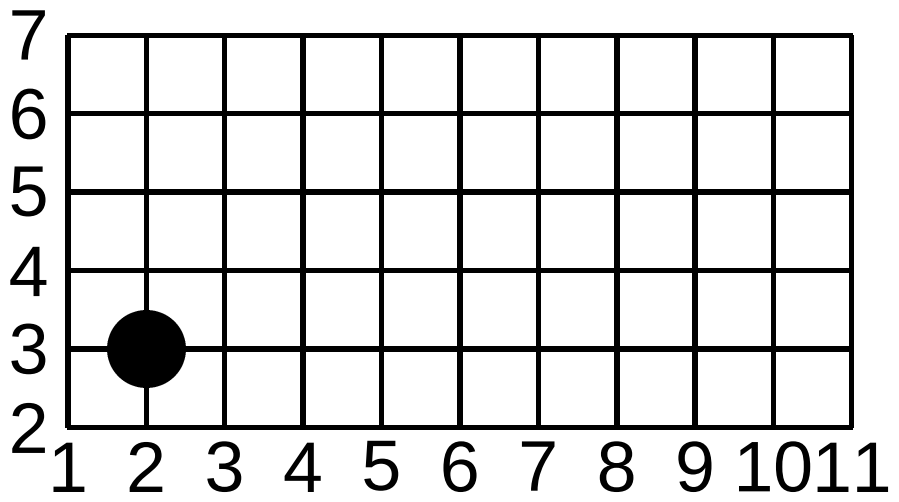
Beispiel: Geradensegment von (2,3) bis (10,6):

$$
\begin{aligned}
dx &= 10 - 2 &&= 8 \\
dy &= 6 - 3 &&= 3 \\
\\
\Delta_O &= 2 \cdot dy &&= 6 \\
\Delta_{NO} &= 2 \cdot (dy - dx) &&= -10
\end{aligned}
$$

Der Mittelpunktalgorithmus

$$\Delta_O = 6$$
$$\Delta_{NO} = -10$$

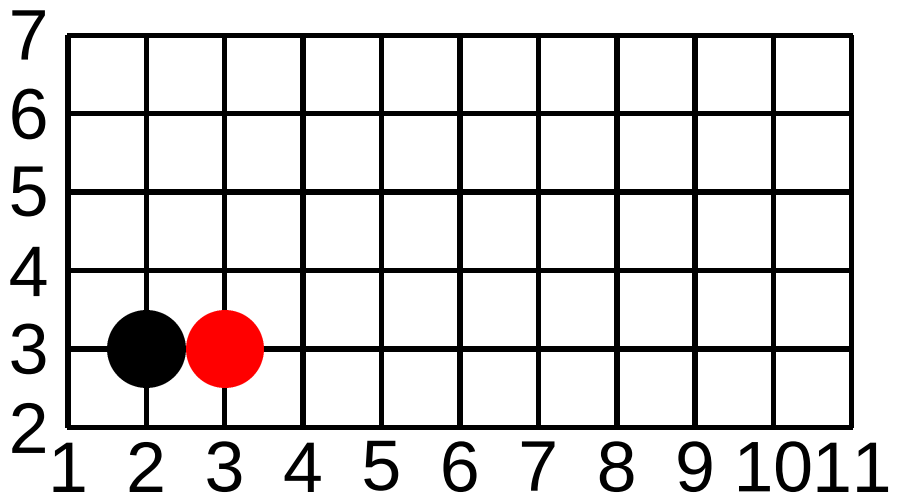
$$D_{\text{init}} = 2 \cdot dy - dx = -2 \quad (O)$$



Der Mittelpunktalgorithmus

$$\Delta_O = 6$$
$$\Delta_{NO} = -10$$

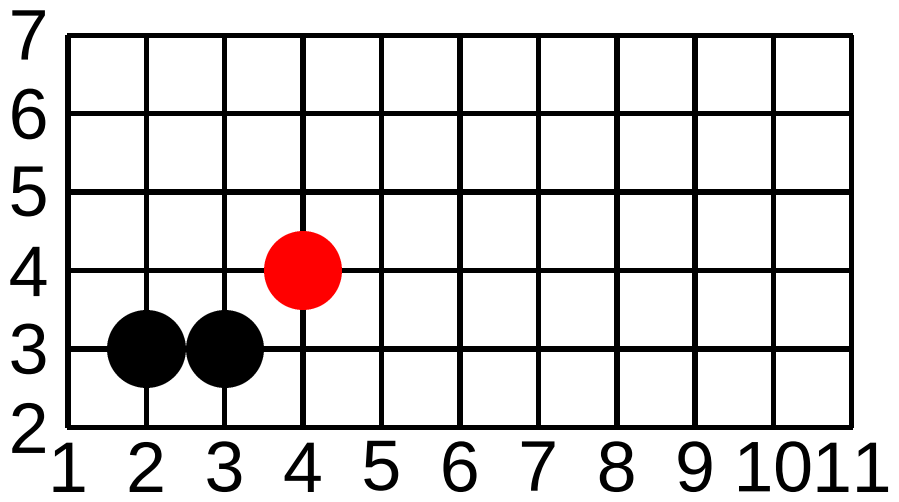
$$D_{\text{init}} = 2 \cdot dy - dx = -2 \quad (O)$$



Der Mittelpunktalgorithmus

$$\Delta_O = 6$$
$$\Delta_{NO} = -10$$

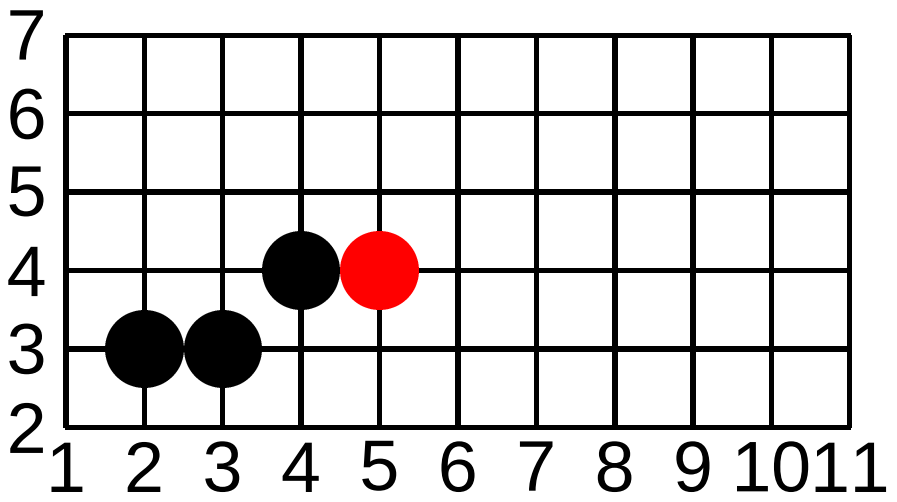
$$D_{\text{init}+1} = D_{\text{init}} + \Delta_O = 4 \quad (NO)$$



Der Mittelpunktalgorithmus

$$\Delta_O = 6$$
$$\Delta_{NO} = -10$$

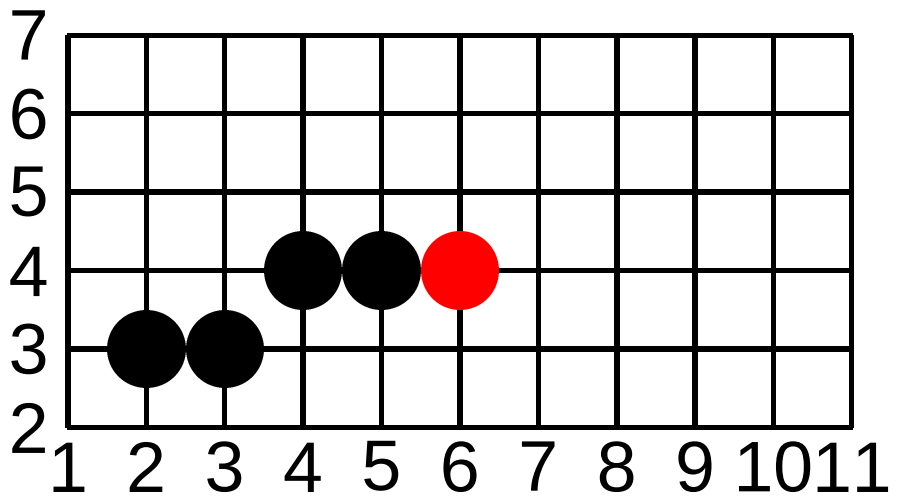
$$D_{\text{init}+2} = D_{\text{init}+1} + \Delta_{NO} = -6 \quad (O)$$



Der Mittelpunktalgorithmus

$$\Delta_O = 6$$
$$\Delta_{NO} = -10$$

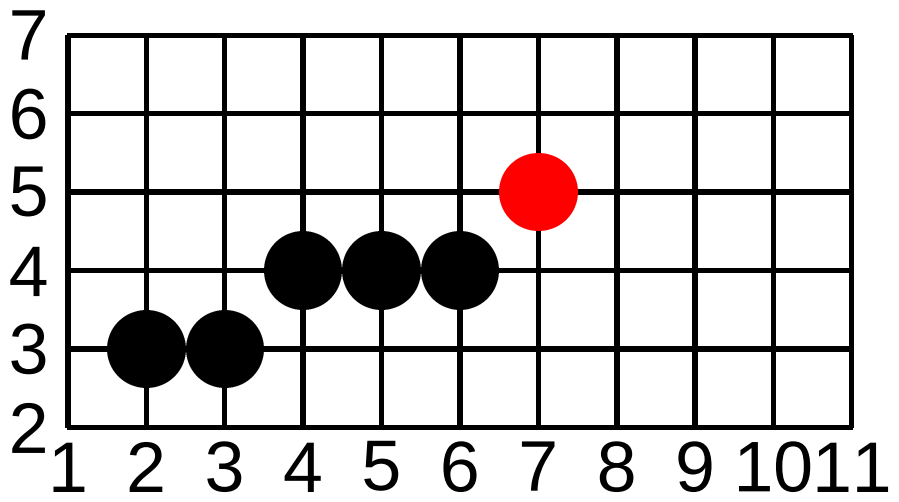
$$D_{\text{init}+3} = D_{\text{init}+2} + \Delta_O = 0 \quad (O?)$$



Der Mittelpunktalgorithmus

$$\Delta_O = 6$$
$$\Delta_{NO} = -10$$

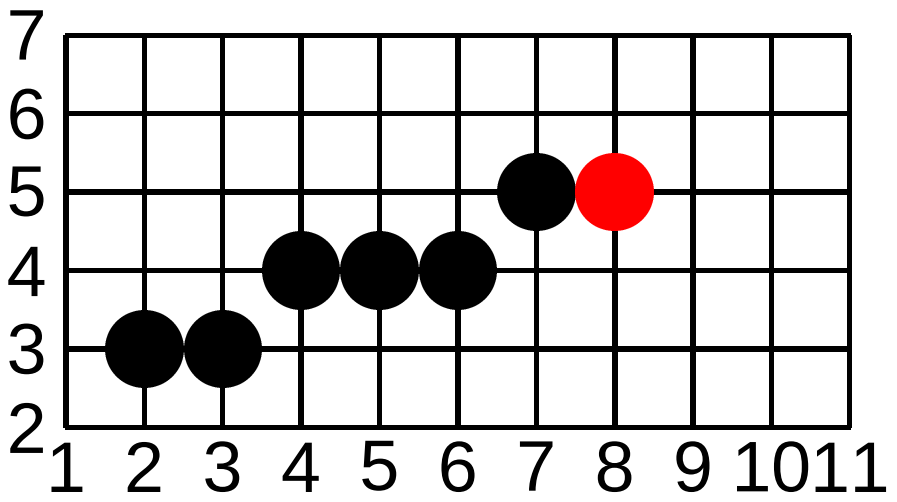
$$D_{\text{init}+4} = D_{\text{init}+3} + \Delta_O = 6 \quad (NO)$$



Der Mittelpunktalgorithmus

$$\Delta_O = 6$$
$$\Delta_{NO} = -10$$

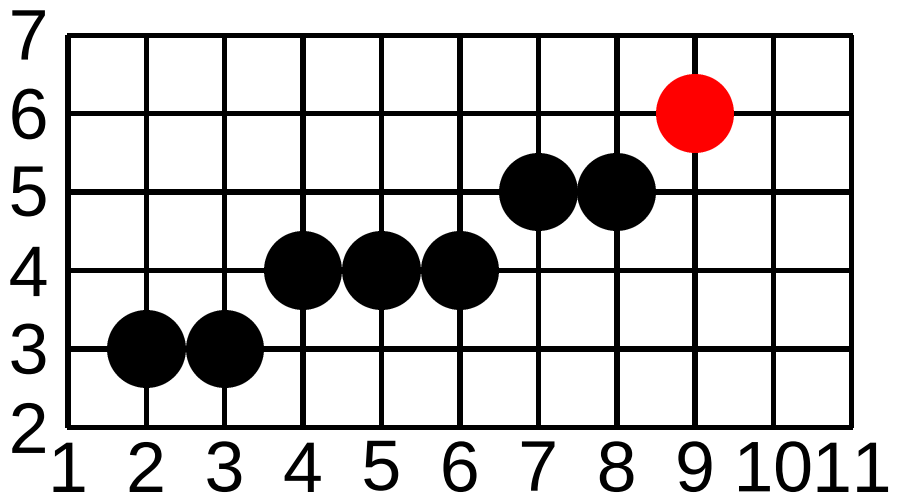
$$D_{\text{init}+5} = D_{\text{init}+4} + \Delta_{NO} = -4 \quad (O)$$



Der Mittelpunktalgorithmus

$$\Delta_O = 6$$
$$\Delta_{NO} = -10$$

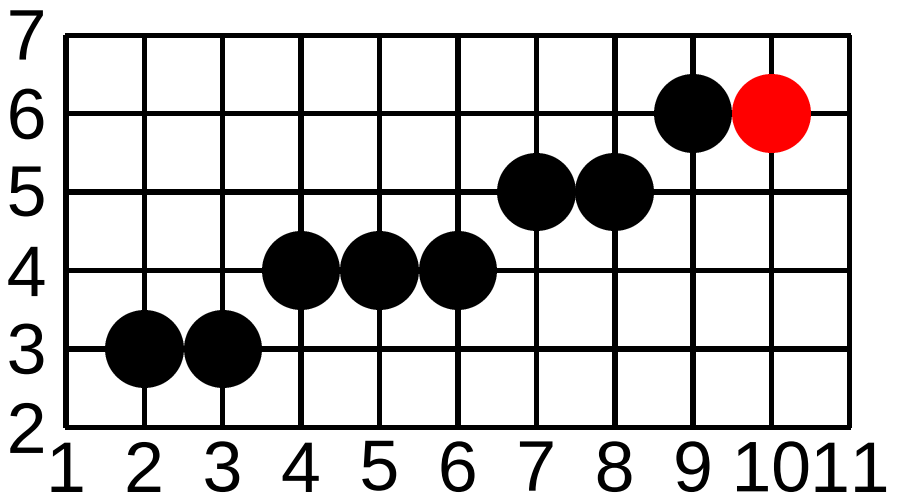
$$D_{\text{init}+6} = D_{\text{init}+7} + \Delta_O = 2 \quad (NO)$$



Der Mittelpunktalgorithmus

$$\Delta_O = 6$$
$$\Delta_{NO} = -10$$

$$D_{\text{init}+7} = D_{\text{init}+6} + \Delta_{NO} = -8 \quad (O)$$



Der Mittelpunktalgorithmus

- **Geraden, deren Steigung betragsmäßig größer als 1 ist:**

  Vertauschung der Rollen von x- und y-Achse bei der Durchführung des Algorithmus.

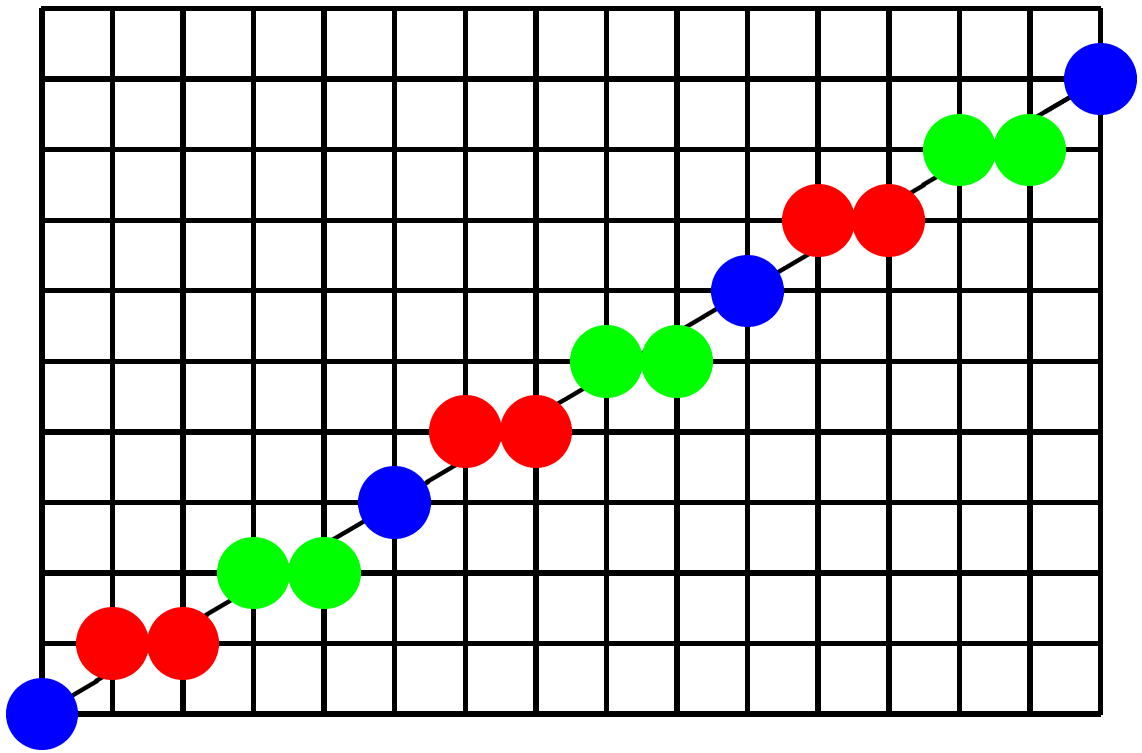
- **Geraden mit negativer Steigung (zwischen -1 und 0):**

  Führe die gleichen Überlegungen wie für die Gerade mit der gleichen positiven Steigung durch. Anstatt des „nordöstlichen“ Pixels muss nur jeweils das „südöstliche“ betrachtet werden.

- **Geradensegmente mit nichtganzzahligem Anfangs- und Endpunkt:**

  Runde die Koordinaten des Anfangs- und Endpunktes und zeichne die Verbindungsgerade.

Strukturelle Algorithmen



Beim Zeichnen einer Geraden treten sich wiederholende Steigungsmuster auf, z.B.

D, H, D, H, D

Strukturelle Algorithmen

Bezeichnet D einen Diagonal- (NO-Pixel) und H einen Horizontalschritt (O-Pixel), so lässt sich eine Gerade durch ein sich wiederholendes Muster aus D- und H-Schritten beschreiben.

Strukturelle Algorithmen berechnen diese Muster zum Zeichnen der Geraden.

Der Mittelpunktalgorithmus benötigt für ein Geradenstück aus n Pixeln $O(n)$ ganzzahlige Rechenoperationen.

Strukturelle Algorithmen benötigen nur logarithmischen Rechenaufwand.

Grundprinzip:

- gegeben: Start- und Endpunkt: (x_0, y_0) bzw. (x_1, y_1) (eines Geradenstücks mit Steigung zwischen 0 und 1)
- Berechne $dx = x_1 - x_0$ und $dy = y_1 - y_0$
- Es müssen (ohne das Startpixel) insgesamt dx Pixel gezeichnet werden. Dafür sind dy Diagonal- und $(dx - dy)$ Horizontalschritte durchzuführen.
- Wähle als erste Approximation die Sequenz $H^{dx-dy}D^{dy}$.
- Permutiere die Ausgangssequenz so, dass sich die korrekte Sequenz ergibt.

Der Algorithmus von Brons

- Sind dx und $(dx - dy)$ (bzw. dy) nicht teilerfremd (d.h. $g = \text{ggT}(dx, dy) > 1$), lässt sich das Zeichnen der Geraden durch g sich wiederholende Sequenzen der Länge dx/g realisieren.
- Ohne Beschränkung der Allgemeinheit nehmen wir daher an, dass dx und dy teilerfremd sind.
- Seien P und Q zwei beliebige Worte über dem Alphabet $\{D, H\}$.
- Aus einer Ausgangsfolge $P^p Q^q$ mit teilerfremden Häufigkeiten p und q (sowie ohne Beschränkung der Allgemeinheit $1 < q < p$) wird mittels ganzzahliger Division

Der Algorithmus von Brons

- $p = k \cdot q + r, \quad 0 < r < q$

  die permutierte Folge

  $$(P^k Q)^{q-r}(P^{k+1}Q)^r, \quad \text{falls } (q-r) > r$$

  $$(P^{k+1} Q)^{r}(P^{k}Q)^{q-r}, \quad \text{falls } r > (q-r)$$

  erzeugt.

- Fahre rekursiv mit den beiden Teilfolgen der Längen r bzw. $(q-r)$ fort, bis $r = 1$ oder $(q-r) = 1$ gilt.

Beispiel

$(x_0, y_0) = (0, 0)$, $(x_1, y_1) = (82, 34)$

$dx = 82$, $dy = 34$, ggT$(dx, dy) = 2$

daher: $\widetilde{dx} = dx/2 = 41$, $\widetilde{dy} = dy/2 = 17$

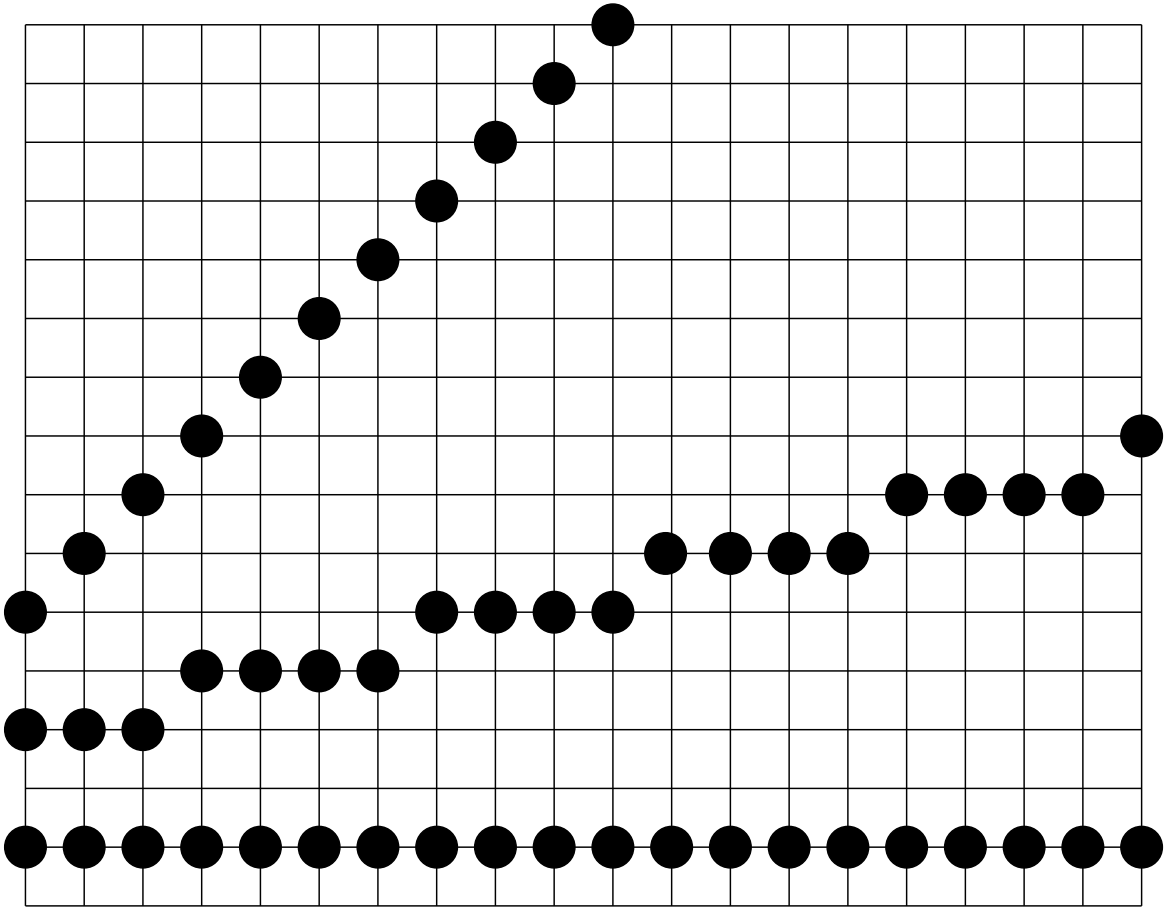
$$H^{24}D^{17} \qquad 24 = 1 \cdot 17 + 7$$

$$(HD)^{10}(H^2D)^7 \qquad 10 = 1 \cdot 7 + 3$$

$$(HDH^2D)^4((HD)^2H^2D)^3 \qquad 4 = 1 \cdot 3 + 1$$

$$(HDH^2D(HD)^2H^2D)^2((HDH^2D)^2(HD)^2((HD)^2H^2D))^1$$

Unterschiedliche Pixeldichte

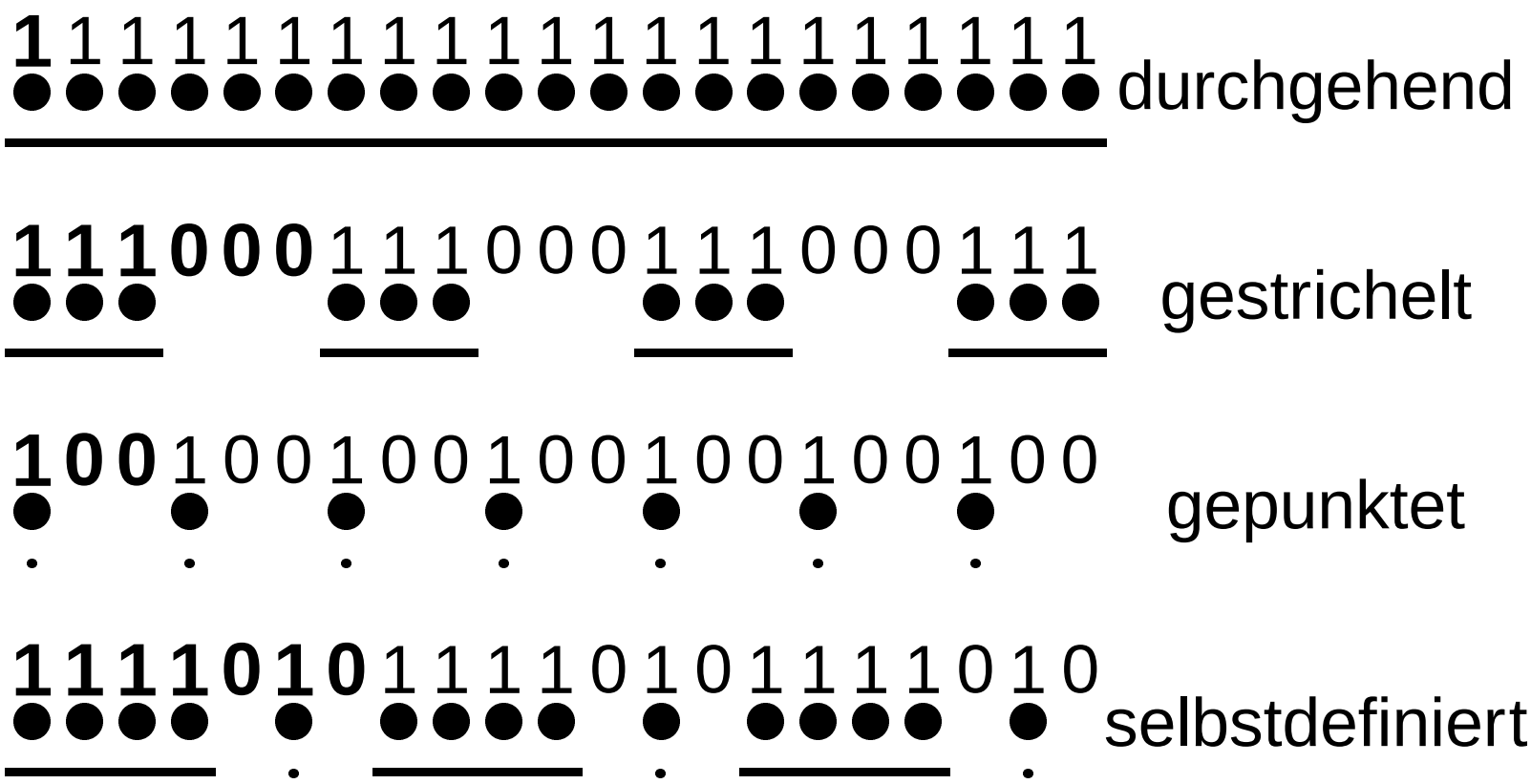


Unterschiedliche Pixeldichte

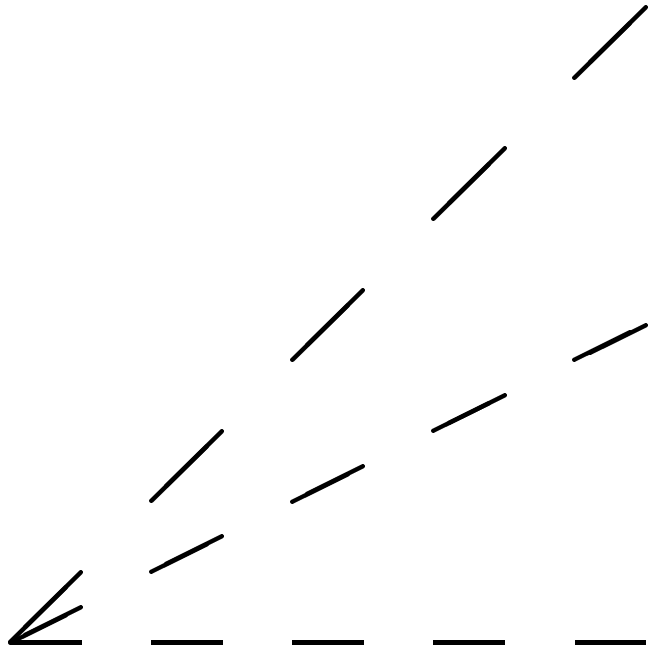
Gerade vom Punkt $(0,0)$ zum Punkt (n,m) mit $m \leq n$

m	Länge der Geraden	Pixeldichte
0	n	1
$\frac{n}{4}$	$n \cdot \sqrt{1+\frac{1}{16}}$	$\frac{1}{\sqrt{1+\frac{1}{16}}}$
n	$n \cdot \sqrt{2}$	$\frac{1}{\sqrt{2}}$
m	$n \cdot \sqrt{1+\left(\frac{m}{n}\right)^2}$	$\frac{1}{\sqrt{1+\left(\frac{m}{n}\right)^2}}$

Linienstile mit Bitmasken



Linienstile mit Bitmasken



Unterschiedliche Strichelungslängen bei Verwendung derselben Bitmaske

Linienstile in Java 2D

Liniendicke:

```
BasicStroke bsThickLine =
                  new BasicStroke(3.0f);
g2d.setStroke(bsThickLine);
```

Strichelungsmuster

```
BasicStroke bsDash =
  new BasicStroke(thickness,
                  BasicStroke.CAP_BUTT,
                  BasicStroke.JOIN_BEVEL,
                  2.0f,
                  dashPattern,dashPhase);
```

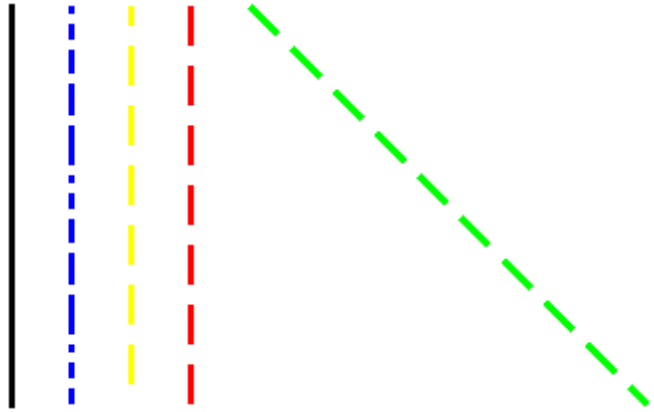
Linienstile in Java 2D

- `thickness`: Dicke der Linie
- `BasicStroke.CAP_BUTT, BasicStroke.JOIN_BEVEL, 2.0f` legen die Übergänge bei Linienzügen fest.
- `dashPattern`: Array, das die Längen der Linien und Lücken festlegt, z.B.

```
float[] dashPattern =
        new float[]{20,10};
```

- `dashPhase`: Stelle, an der das Strichelungsmuster beginnt

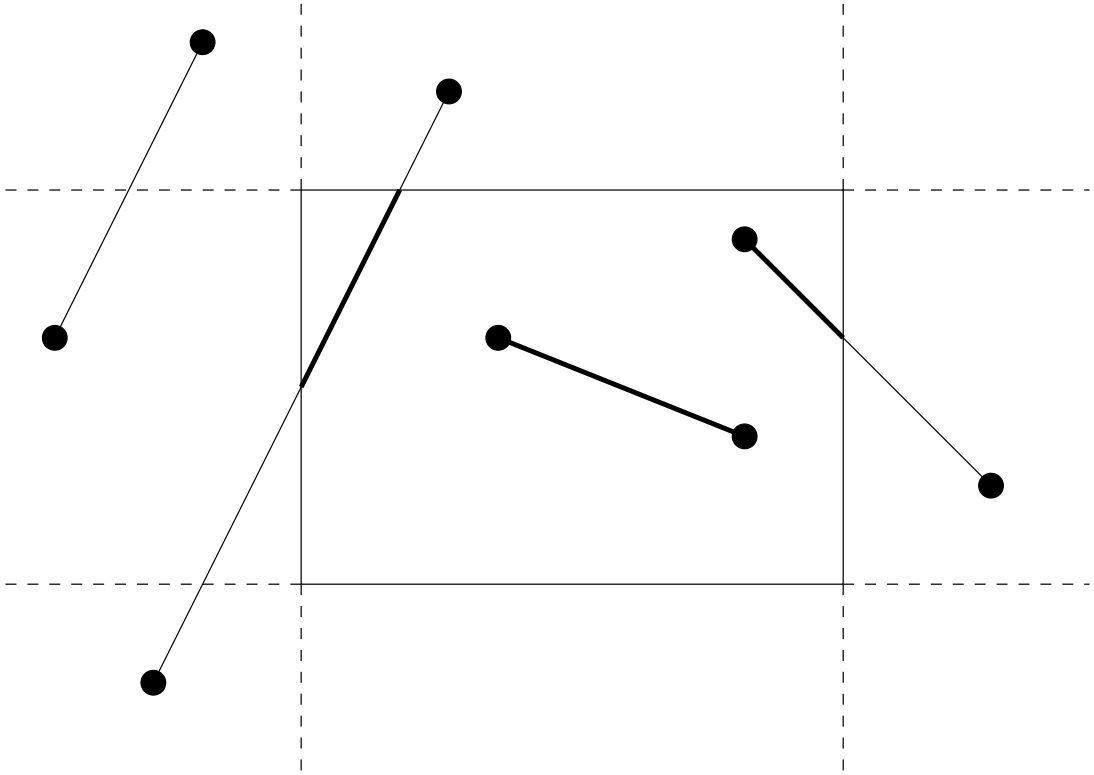
StrokingExample.java



dashPattern	dashPhase
4,5,8,5,12,5,16,5,20,5	0
20,10	10
20,10	0
20,10	0

Clipping

Unter **Clipping** versteht man allgemein das Abschneiden oder gänzliche Weglassen von Objekten, die nur teilweise bzw. gar nicht in den darzustellenden Bildbereich hineinragen.



2D-Geraden-Clipping

Einfache, aber rechenaufwendige Methode des Clippings von Geradensegmenten:

Berechnung der Schnittpunkte der betrachteten Geraden und den durch das Rechteck induzierten vier Geraden.

Darstellung eines Geradenstücks mit Startpunkt (x_0, y_0) und Endpunkt (x_1, y_1) als Konvexkombination von Start- und Endpunkt:

$$\mathbf{g}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = (1-t) \cdot \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + t \cdot \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$$

mit $0 \leq t \leq 1$.

2D-Geraden-Clipping

Berechnung der Schnittpunkte mit dem Rechteck, das durch die Punkte $(x_{\text{min}}, y_{\text{min}})$ und $(x_{\text{max}}, y_{\text{max}})$ definiert wird:

Bestimmung des Schnittpunktes mit der unteren Kante:

$$(1-t_1)\cdot\begin{pmatrix} x_0 \\ y_0 \end{pmatrix}+t_1\cdot\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = (1-t_2)\cdot\begin{pmatrix} x_{\text{min}} \\ y_{\text{min}} \end{pmatrix}+t_2\cdot\begin{pmatrix} x_{\text{max}} \\ y_{\text{min}} \end{pmatrix}$$

Liefert Werte für t_1 und t_2. (Gibt es keine eindeutige Lösung, sind die beiden Geraden parallel.)

2D-Geraden-Clipping

- $t_1 < 0$ und $t_2 < 0$: Der Schnittpunkt liegt außerhalb des Geradenstücks und vor x_{min}.
- $0 \leq t_1 \leq 1$ und $t_2 < 0$: Das Geradenstück schneidet die durch die untere Kante definierte Gerade vor x_{min}.
- $t_1 > 1$ und $t_2 < 0$: Der Schnittpunkt liegt außerhalb des Geradenstücks und vor x_{min}.
- $t_1 < 0$ und $0 \leq t_2 \leq 1$: Die Gerade schneidet die Kante vor (x_0, y_0).
- $0 \leq t_1 \leq 1$ und $0 \leq t_2 \leq 1$: Das Geradenstück schneidet die untere Kante.
- $t_1 > 1$ und $0 \leq t_2 \leq 1$: Die Gerade schneidet die Kante hinter (x_1, y_1).

2D-Geraden-Clipping

- $t_1 < 0$ und $t_2 > 1$: Der Schnittpunkt liegt außerhalb des Geradenstücks und hinter x_{max}.
- $0 \leq t_1 \leq 1$ und $t_2 > 1$: Das Geradenstück schneidet die durch die untere Kante definierte Gerade hinter x_{max}.
- $t_1 > 1$ und $t_2 > 1$: Der Schnittpunkt liegt außerhalb des Geradenstücks und hinter x_{max}.

Entsprechende Überlegungen für die übrigen Rechteckkanten

Cohen-Sutherland-Clipping

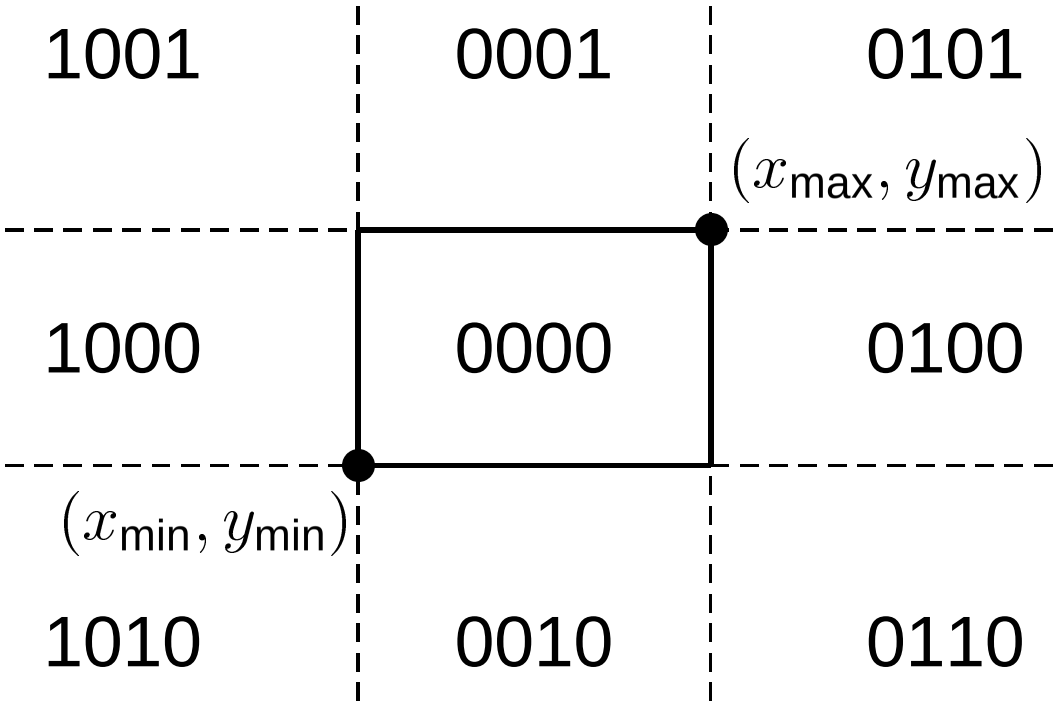
Ziel: möglichst Vermeidung der Berechnung von Geradenschnittpunkten

Einteilung der 2D-Welt in neun Teilbereiche, die durch einen 4-Bit-Code beschrieben werden:

Einem Punkt $P = (x_P, y_P)$ wird der Binärcode $b_1^{(P)} b_2^{(P)} b_3^{(P)} b_4^{(P)} \in \{0,1\}^4$ zugeordnet mit:

$$
\begin{aligned}
b_1^{(P)} = 1 \quad &\Leftrightarrow \quad x_p < x_{\text{min}} \\
b_2^{(P)} = 1 \quad &\Leftrightarrow \quad x_p > x_{\text{max}} \\
b_3^{(P)} = 1 \quad &\Leftrightarrow \quad y_p < y_{\text{min}} \\
b_4^{(P)} = 1 \quad &\Leftrightarrow \quad y_p > y_{\text{max}}
\end{aligned}
$$

Cohen-Sutherland-Clipping

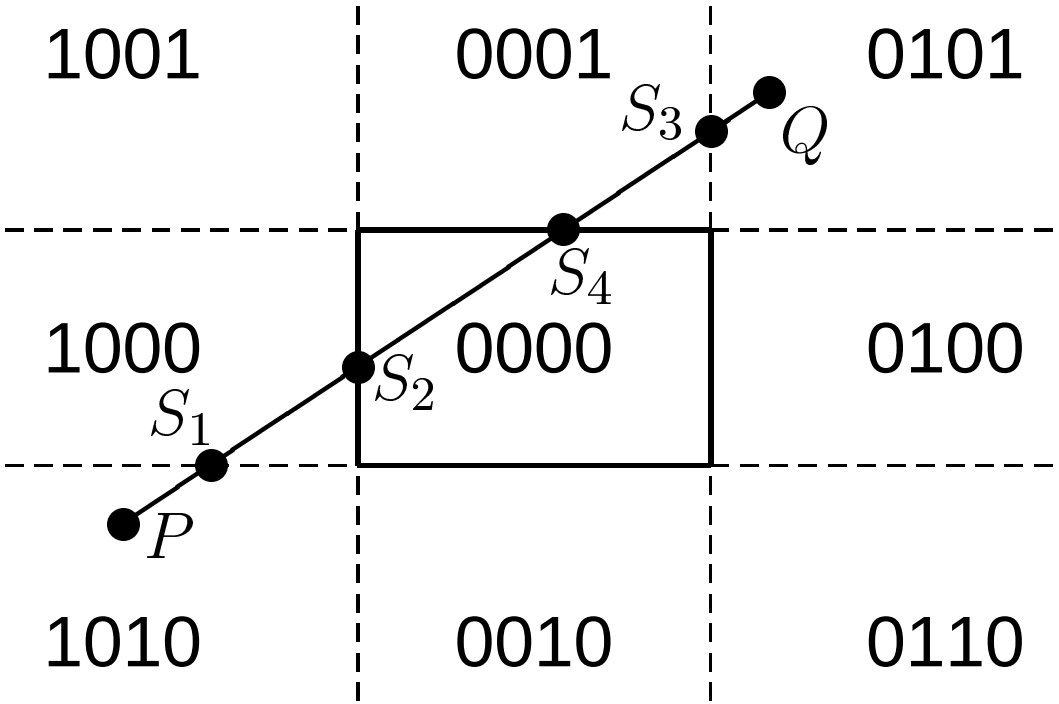


Cohen-Sutherland-Clipping

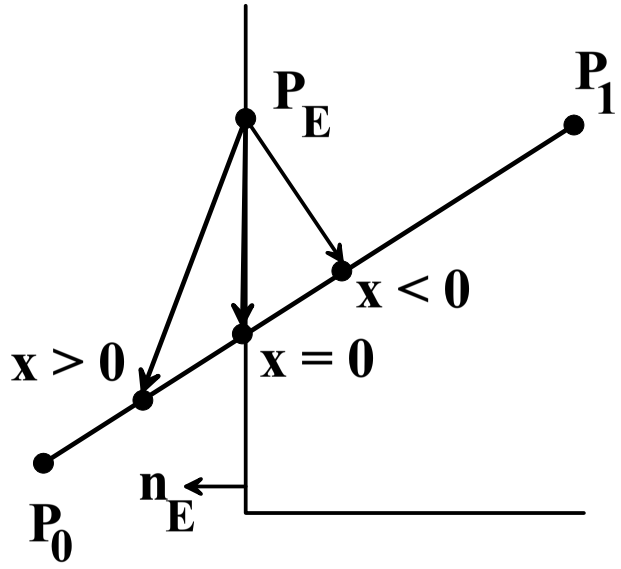
Geradensegment mit Startpunkt P und Endpunkt Q

- $b^{(P)} \vee b^{(Q)} = 0000 \rightarrow$ zeichne die Strecke $\overline{PQ}$ (fertig)
- $b^{(P)} \wedge b^{(Q)} \neq 0000 \rightarrow$ kein Zeichnen erforderlich (fertig)
- Es muss $b^{(P)} \neq 0000$ oder $b^{(Q)} \neq 0000$ gelten, ohne Einschränkung der Allgemeinheit gelte $b^{(P)} \neq 0000$. Berechne den oder die Schnittpunkte (maximal zwei) des Geradensegments mit den Rechteckgeraden, zu denen die Eins(en) in $b^{(P)} \neq 0000$ gehören. Ersetze P durch den (oder einen der beiden) Schnittpunkt(e) und fahre von vorne fort.

Cohen-Sutherland-Clipping



Cyrus-Beck-Clipping



$$\mathbf{g}(t) \;=\; (1-t)\cdot\mathbf{p}_0 + t\cdot\mathbf{p}_1 \;=\; \mathbf{p}_0 + (\mathbf{p}_1-\mathbf{p}_0)t \qquad (t\in[0,1])$$

Verbindungsvektor $\mathbf{p}_E$ mit einem Punkt auf der Strecke $\mathbf{p}_0$ und $\mathbf{p}_1$:

$$\mathbf{p}_0 + (\mathbf{p}_1-\mathbf{p}_0)t - \mathbf{p}_E$$

Cyrus-Beck-Clipping

Für den Schnittpunkt der Geraden mit der Kante des Clippingbereichs muss gelten:

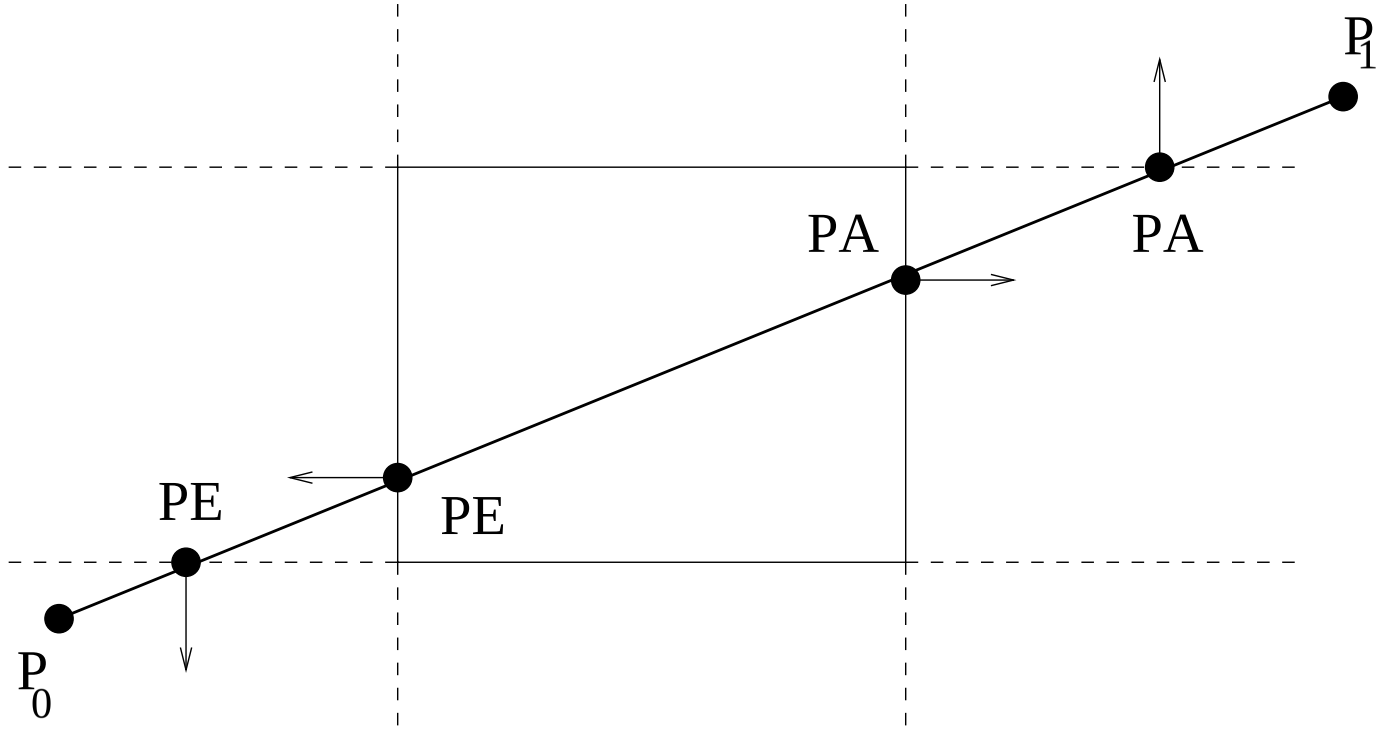
$$0 \;=\; \mathbf{n}_E^\top \cdot (\mathbf{p}_0 + (\mathbf{p}_1 - \mathbf{p}_0)t - \mathbf{p}_E) \;=\; \mathbf{n}_E^\top \cdot (\mathbf{p}_0 - \mathbf{p}_E) + \mathbf{n}_E^\top \cdot (\mathbf{p}_1 - \mathbf{p}_0)t$$

$$t \;=\; -\frac{\mathbf{n}_E^\top \cdot (\mathbf{p}_0 - \mathbf{p}_E)}{\mathbf{n}_E^\top \cdot (\mathbf{p}_1 - \mathbf{p}_0)}$$

$t \notin [0,1]$: kein Schnittpunkt der Kante mit dem Geradensegment

Cyrus-Beck-Clipping

Die noch verbleibenden potenziellen Schnittpunkte mit den Rechteckkanten werden als „potenziell austretend“ (PA) und „potentiell eintretend“ (PE) charakterisiert.



Cyrus-Beck-Clipping

Rechnerisch lässt sich die Entscheidung PA oder PE anhand des Winkels zwischen der Geraden $\overline{\mathbf{p}_0\mathbf{p}_1}$ und dem zur entsprechenden Rechteckkante gehörenden Normalenvektor $\mathbf{n}$ treffen:

- Ist der Winkel größer als $90°$, liegt der Fall PE vor.
- Ist der Winkel kleiner als $90°$, liegt der Fall PA vor.

Dazu: Bestimmung des Vorzeichen des Skalarprodukts

$$\mathbf{n}^\top \cdot (\mathbf{p}_1 - \mathbf{p}_0)$$

Cyrus-Beck-Clipping

Berechnung des im Clippingrechteck liegenden Geradenstücks:

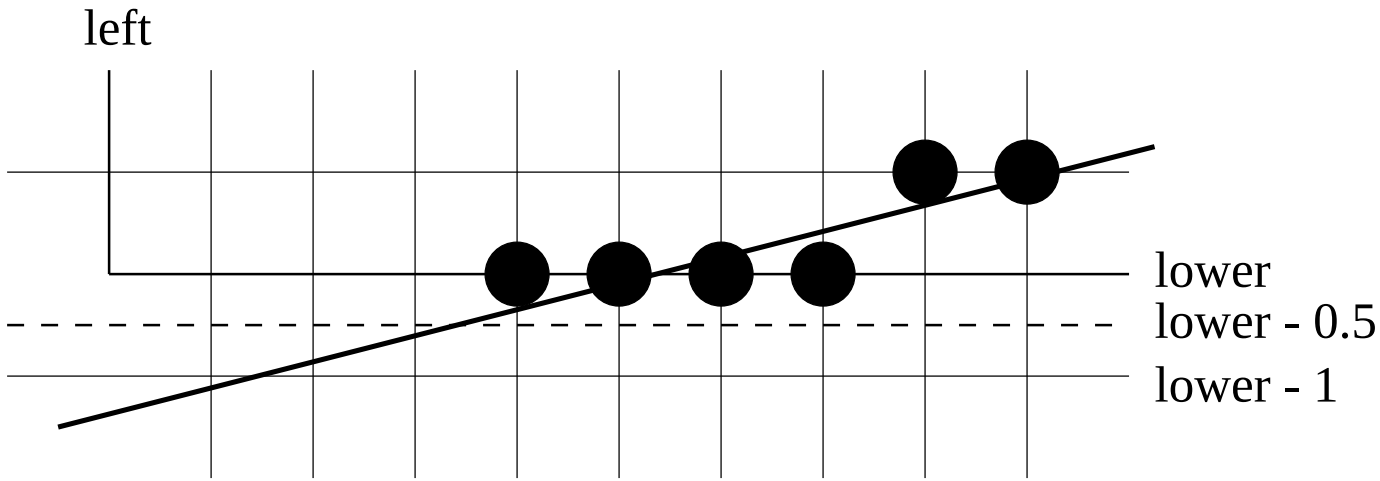
Bestimme den größten Wert t_E, der zu einem PE-Punkt gehört und den kleinste Wert t_A, der zu einem PA-Punkt gehört.

Gilt $t_E < t_A$ ist genau der Teil der Geraden zwischen den Punkten $P_0 + (P_1 - P_0)t_E$ und $P_0 + (P_1 - P_0)t_A$ in das Rechteck zu zeichnen.

Andernfalls liegt das Geradenstück außerhalb des Rechtecks.

Geraden-Clipping

Probleme bei der Bestimmung des Startpixels:

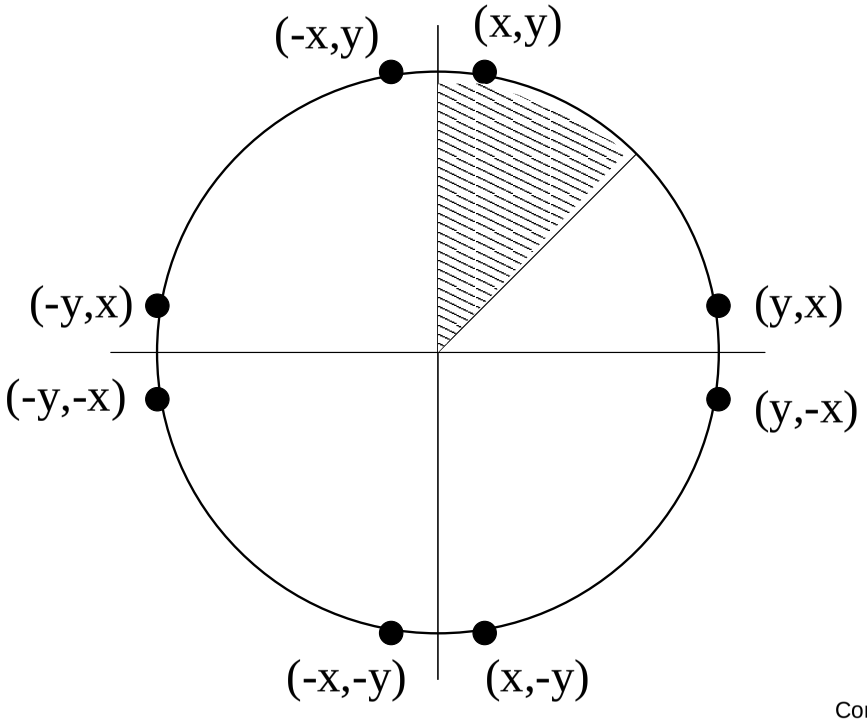


Bresenham für Kreise

Es soll ein Kreis mit Mittelpunkt (x_m, y_m) und Radius R gezeichnet werden.

Ist (x_m, y_m) ein Rasterpunkt, so reicht es, ein Verfahren zur Zeichnung eines Kreises mit Mittelpunkt $(0, 0)$ zu kennen und (x_m, y_m) als Offset zu verwenden.

Kreissymmetrie (Achtelkreis ist ausreichend):



Bresenham für Kreise

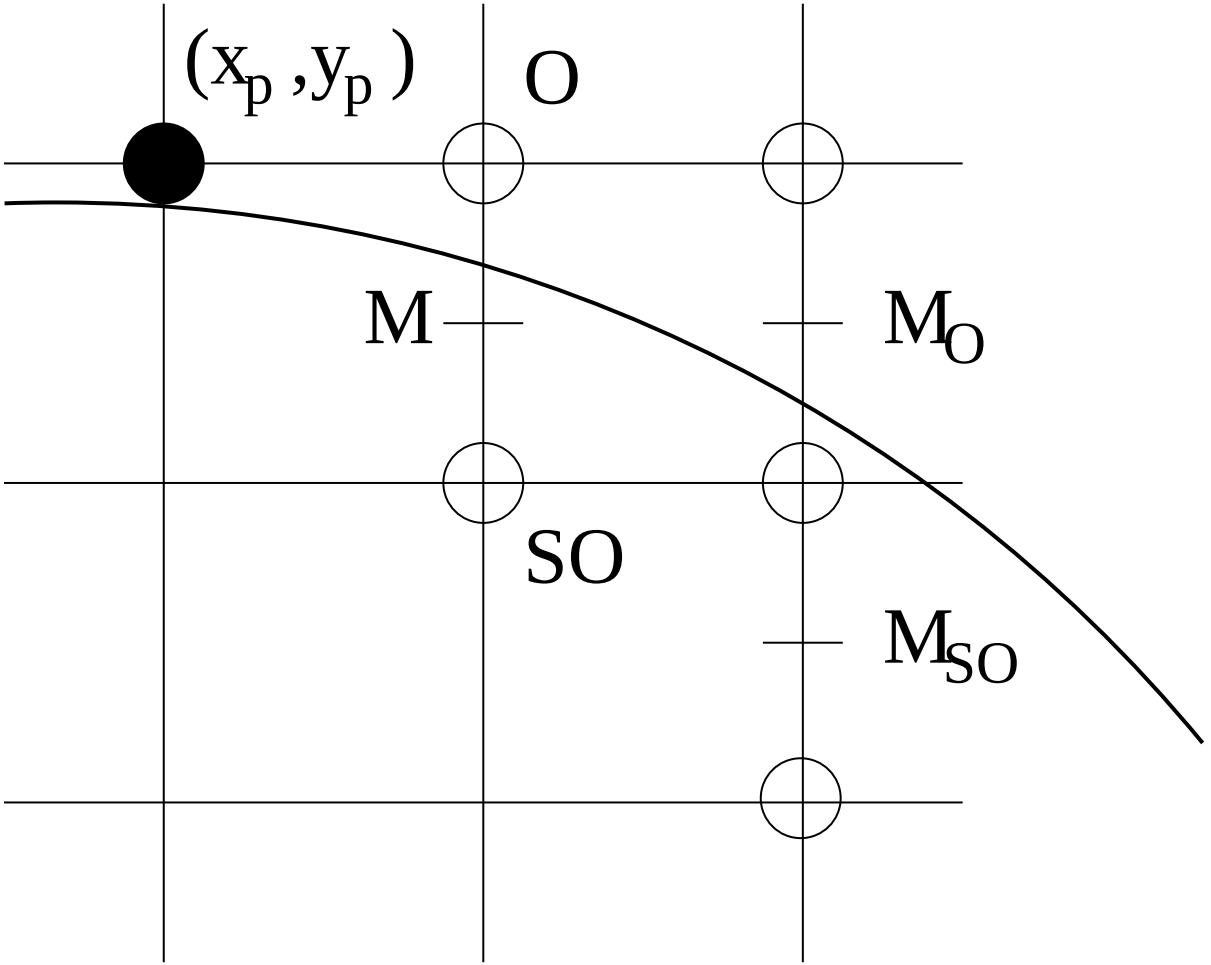
Kreisgleichung $x^2 + y^2 = R^2$ in impliziter Form:

$$d \;=\; F(x,y) \;=\; x^2 + y^2 - R^2$$

- $F(x,y) = 0 \;\Leftrightarrow\; (x,y)$ liegt auf dem Kreis.
- $F(x,y) > 0 \;\Leftrightarrow\; (x,y)$ liegt außerhalb des Kreises.
- $F(x,y) < 0 \;\Leftrightarrow\; (x,y)$ liegt innerhalb des Kreises.

Wurde der Punkt (x_p, y_p) in einem Schritt gezeichnet, kommt als nächster zu zeichnender Punkt nur einer der beiden Punkte O und SO in Frage.

Bresenham für Kreise



Bresenham für Kreise

Analog zu Geraden: implizite Darstellung des Kreises als Entscheidungsvariable

Einsetzen des Mittelpunkts liefert Entscheidung für den nächsten zu zeichnenden Punkt

- Ist $d > 0$, so muss SO gezeichnet werden.
- Ist $d < 0$, so muss O gezeichnet werden.

Um welchen Wert Δ ändert sich d in jedem Schritt?

Bresenham für Kreise

Fall 1: O, d.h. $(x_{p+1}, y_{p+1}) = (x_p + 1, y_p)$ war das nach (x_p, y_p) zu zeichnende Pixel

zum Zeichnen des Pixels (x_{p+2}, y_{p+2}) zu betrachtender Mittelpunkt:

$$M_{\text{neu}} = \left(x_p + 2, y_p - \frac{1}{2}\right)$$

Bresenham für Kreise

$$d_{\text{neu}} = F\left(x_p + 2, y_p - \frac{1}{2}\right) = (x_p + 2)^2 + \left(y_p - \frac{1}{2}\right)^2 - R^2$$

$$d_{\text{alt}} = F\left(x_p + 1, y_p - \frac{1}{2}\right) = (x_p + 1)^2 + \left(y_p - \frac{1}{2}\right)^2 - R^2$$

$$\Delta_O = d_{\text{neu}} - d_{\text{alt}} = 2x_p + 3$$

Bresenham für Kreise

Fall 2: SO, d.h. $(x_{p+1}, y_{p+1}) = (x_p + 1, y_p - 1)$ war das nach (x_p, y_p) zu zeichnende Pixel

zum Zeichnen des Pixels (x_{p+2}, y_{p+2}) zu betrachtender Mittelpunkt:

$$M_{\text{neu}} = \left(x_p + 2, y_p - \frac{3}{2}\right)$$

Bresenham für Kreise

$$d_{\text{neu}} = F\left(x_p + 2, y_p - \frac{3}{2}\right) = (x_p + 2)^2 + \left(y_p - \frac{3}{2}\right)^2 - R^2$$

$$d_{\text{alt}} = F\left(x_p + 1, y_p - \frac{1}{2}\right) = (x_p + 1)^2 + \left(y_p - \frac{1}{2}\right)^2 - R^2$$

$$\Delta_{SO} = d_{\text{neu}} - d_{\text{alt}} = 2x_p - 2y_p + 5$$

Bresenham für Kreise

$$\Delta = \begin{cases} 2x_p + 3 & \text{falls } O \text{ gewählt wurde} \\ 2x_p - 2y_p + 5 & \text{falls } SO \text{ gewählt wurde} \end{cases}$$

d.h.

$$\Delta = \begin{cases} 2x_p + 3 & \text{falls } d_{\text{alt}} < 0 \\ 2x_p - 2y_p + 5 & \text{falls } d_{\text{alt}} > 0, \end{cases}$$

Δ ist immer ganzzahlig, so dass sich die Entscheidungsvariable d nur um ganzzahlige Werte ändert.

Bresenham für Kreise

Initialisierung von d:

Der Startpunkt des Kreises ist $(0, R)$ mit $R \in \mathbb{N}$.

erster zu betrachtender Mittelpunkt: $\left(1, R - \frac{1}{2}\right)$.

$$F\left(1, R - \frac{1}{2}\right) = \frac{5}{4} - R$$

Initialisierung von d:

$$d = \frac{5}{4} - R$$

Bresenham für Kreise

Zusammenfassung:

$$d_{\text{init}} = \frac{5}{4} - R$$

$$d_{\text{neu}} = d_{\text{alt}} + \Delta$$

$$\Delta = \begin{cases} 2x_p + 3 & \text{falls } d_{\text{alt}} < 0 \\ 2x_p - 2y_p + 5 & \text{falls } d_{\text{alt}} > 0 \end{cases}$$

Bresenham für Kreise

bis auf $5/4$ bei der Initialisierung, nur ganzzahlige Operationen

ersetze d durch den Wert $D = d - \frac{1}{4}$.

Es müssten jetzt statt auf $d > 0$ bzw. $d < 0$ auf $D > -\frac{1}{4}$ bzw. $D < -\frac{1}{4}$ getestet werden.

Da D nur ganzzahlige Werte annimmt, genügt weiterhin die Abfrage $D > 0$ oder $D < 0$.

Bresenham für Kreise

$$D_{\text{init}} = 1 - R$$

$$D_{\text{neu}} = D_{\text{alt}} + \Delta$$

$$\Delta = \begin{cases} 2x_p + 3 & \text{falls } D_{\text{alt}} < 0 \\ 2x_p - 2y_p + 5 & \text{falls } D_{\text{alt}} > 0 \end{cases}$$

Bresenham für Kreise

Kreise mit nichtganzzahligem, aber rationalem Radius:

$$x^2 + y^2 = \left(\frac{m}{n}\right)^2 \quad \Leftrightarrow \quad n^2x^2 + n^2y^2 = m^2$$

Anwendung des gleichen Prinzips auf die implizite Form

$$F(x, y) = n^2x^2 + n^2y^2 - m^2$$

Prinzip des Bresenham-Algorithmus

- gegeben: Im Intervall $[x_0, x_1]$ zu zeichnende Kurve $y = f(x)$
- Voraussetzung: Entweder gilt $0 \leq f'(x) \leq 1$ oder $-1 \leq f'(x) \leq 0$ im gesamten zu zeichnenden Intervall.
- Schreibe $y = f(x)$ geeignet in impliziter Form $D = F(x, y) = 0$ und verwende D als Entscheidungsvariable.
- Berechne die Änderung Δ von D in Abhängigkeit des zuletzt gezeichneten Pixels (O/NO bzw. O/SO).

Prinzip des Bresenham-Algorithmus

- Voraussetzung: Δ ist ganzzahlig oder zumindest rational. Ist Δ (echt) rational, betrachte die Entscheidungsvariable nD anstelle von D, wobei n so gewählt ist, dass $n\Delta$ immer ganzzahlig ist.
- Bestimme den Startwert D_{init} durch Einsetzen des ersten Mittelpunktes (nach x_0).
- Ist D_{init} nicht ganzzahlig, können die Nachkommastellen ignoriert werden, wenn sich D nur ganzzahlig ändert.

Zeichnen beliebiger Kurven

Der Mittelpunkt- oder Bresenham-Algorithmus lässt sich z.B. aufgrund der Steigungsvoraussetzungen nicht auf beliebige Kurven anwenden.

Außerdem müssten die Berechnungsvorschriften für jeden Kurventyp individuell bestimmt werden.

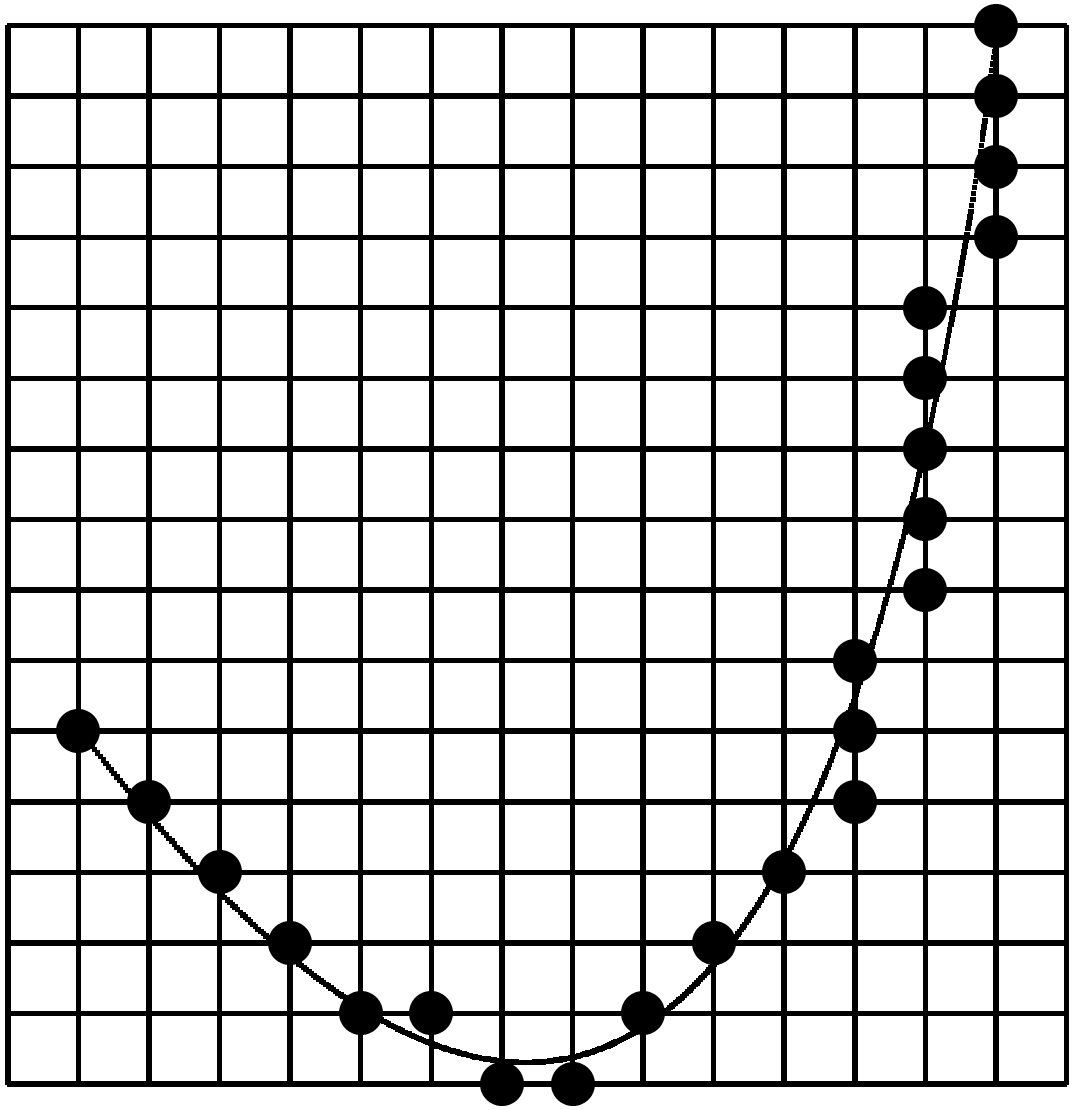
Beliebige Kurven werden daher nach folgendem Schema gezeichnet:

gegeben: Im Intervall $[x_0, x_1]$ zu zeichnende Kurve $y = f(x)$ (mit $x_0, x_1 \in \mathbb{Z}$)

Zeichnen beliebiger Kurven

```
int yRound1, yRound2;
yRound1 = round(f(x0));
for (int x=x0; x<x1; x++)
{
  yRound2 = round(f(x+1));
  drawLine(x,yRound1,x+1,yRound2);
  yRound1 = yRound2;
}
```

Zeichnen von Funktionen



Antialiasing

Aliasing-Effekte treten bei der diskreten Abtastung eines kontinuierlichen Signals auf.

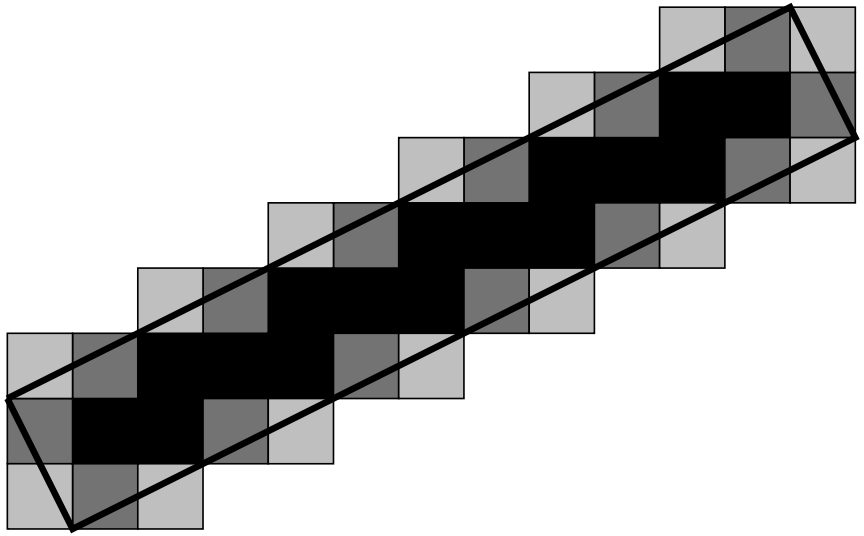
Beim Zeichnen von Linien (Abtastung einer kontinuierlichen Kurve durch ein diskretes Pixelgitter) kommt es daher zu Aliasing-Effekten (Ausfransungen, Stufen).

Antialiasing versucht, diese Effekte durch die Verwendung von verschiedenen Grauwerten oder Farbintensitäten entgegenzuwirken.

Pixel werden nicht entweder schwarz oder weiß gezeichnet, sondern mit unterschiedlicher Intensität, je nachdem, wie weit sie von der zu zeichnenden Linie entfernt sind oder abgedeckt werden.

Unweighted Area Sampling

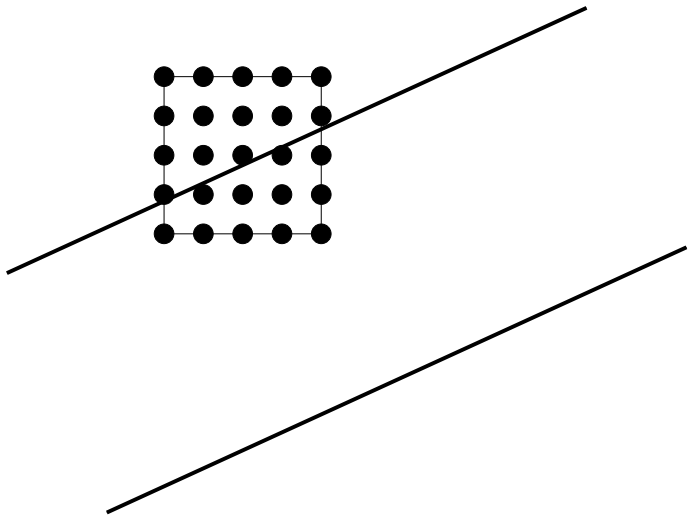
- Ein Geradenstück wird als (schmales, langes) Rechteck interpretiert.
- Jedem Pixel wird ein Quadrat zugeordnet. Die Intensität des Pixels wird proportional zur Schnittfläche des Pixelquadrats mit dem Geradenrechteck gewählt.



Unweighted Area Sampling

Einfache Heuristik zur Berechnung des Anteils des Pixelquadrats, der vom Geradenrechteck überdeckt wird:

Das Pixelquadrat wird mit einem feineren Pixelraster überzogen und der Anteil der verfeinerten Pixel innerhalb des Geradenrechtecks bestimmt die Intensität des Pixels.

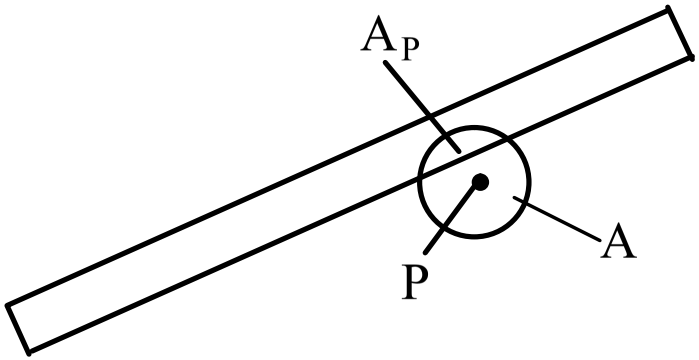


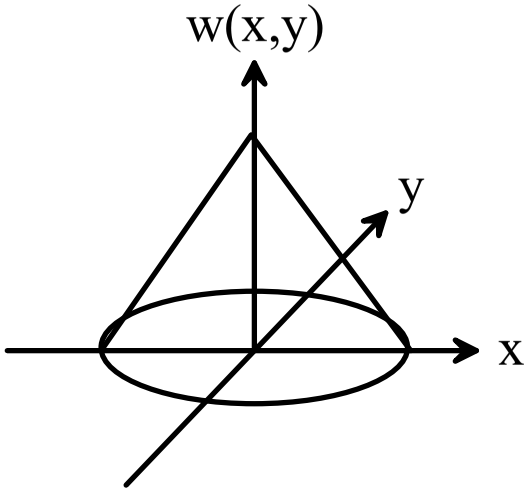
Intensität: 11/25

Weighted Area Sampling

Weighted Area Sampling: Verwendung einer Gewichtsfunktion $w(x, y)$

$$\text{Intensität für Pixel P} = \frac{\int_{A_P} w(x,y)dx\,dy}{\int_A w(x,y)dx\,dy}$$





Gupta-Sproull-Antialiasing

Beim Weighted Area Sampling müssen diskrete Intensitätswerte (maximal 256, i.A. verwendet man deutlich weniger) berechnet.

Durchläuft man die Pixel in geeigneter Weise, so ähnelt dieses Verfahren dem Zeichnen einer Kurve auf einem Pixelraster:

- Das Durchlaufen der Pixel entspricht dem Durchlaufen der x-Pixel beim Kurvenzeichnen.
- Das Bestimmen der (diskreten/gerundeten) Intensitätswerte entspricht dem Berechnen der gerundeten Funktionswerte beim Kurvenzeichnen.

Gupta-Sproull-Antialiasing

Der Gupta-Sproull-Antialiasing-Algorithmus verwendet eine geeignete Gewichtsfunktion und eine passende Menge von Intensitätsstufen, so dass sich die Berechnung der Intensitätswerte mit dem Bresenham-Algorithmus durchführen lässt.

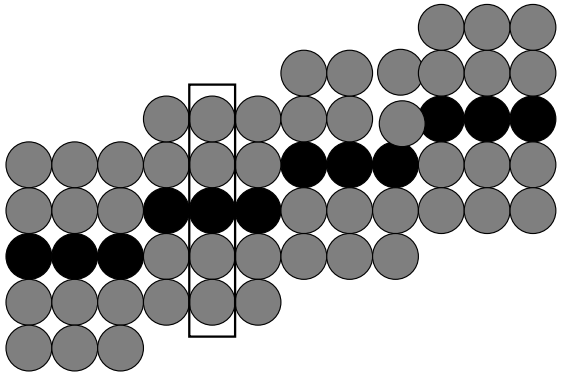
Dadurch müssen keine Integrale berechnet, nicht einmal Fließkommaarithmetik verwendet werden.

Antialiasing in Java2D:

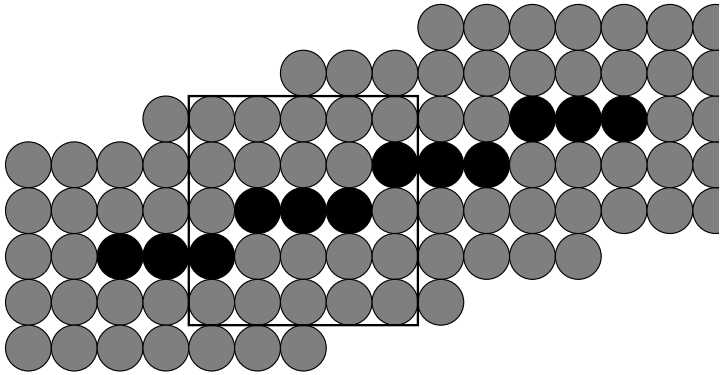
```
g2d.setRenderingHint(
        RenderingHints.KEY_ANTIALIASING,
        RenderingHints.VALUE_ANTIALIAS_ON);
```

Zeichnen dicker Linien

Bei einer hohen Auflösung sind Linien, die nur einen Pixel breit sind, i.A. sehr dünn.



Pixel-Replikation



Zeichenstift

Alternative: Linien als auszufüllende Polygone/Rechtecke betrachten

Polygonzüge mit dicken Linien

Wie sollen die Enden von dicken Linien aussehen?

Java 2D: Zeichnen dicker Linien

```
new BasicStroke(thickness,ending,join);
```

Werte für `ending`:

- `BasicStroke.CAP_BUTT`: Die Enden der Linien werden gerade, senkrecht zur Richtung der Gerade abgeschnitten.
- `BasicStroke.CAP_ROUND`: An die Enden wird ein Halbkreis angefügt.
- `BasicStroke.CAP_SQUARE`: An die Enden der Linien wird ein Rechteck angefügt, so dass die Linien um eine halbe Linienbreite verlängert werden.

Java 2D: Zeichnen dicker Linien

Werte für `join`:

- `BasicStroke.JOIN_MITER`: Die äußeren Kanten der breiten Linien werden bis zu ihrem Schnittpunkt verlängert, so dass eine spitze Verbindung entsteht.
  Bei sehr spitzem Winkel kann die Spitze weit über die eigentlichen Linien hinaus gehen. Vermeidung: weiterer `float`-Parameter, der die maximale Länge der Spitze begrenzt. Bei zu langer Spitze Verwendung des folgenden Verbindungsmodus:

Java 2D: Zeichnen dicker Linien

- `BasicStroke.JOIN_BEVEL`: Die Verbindungsstelle der beiden Liniensegmente wird gerade, sekrecht zur Winkelhalbierenden der beiden Linien abgeschnitten.
- `BasicStroke.JOIN_ROUND`: An das abgeschnittene Ende, das sich bei `BasicStroke.JOIN_BEVEL` ergibt, wird ein Kreisabschnitt anghängt. Der Öffnungswinkel des Kreisabschnitts wird so gewählt, dass die Enden der Liniensegmente die Tangenten an den Kreisabschnitt bilden.