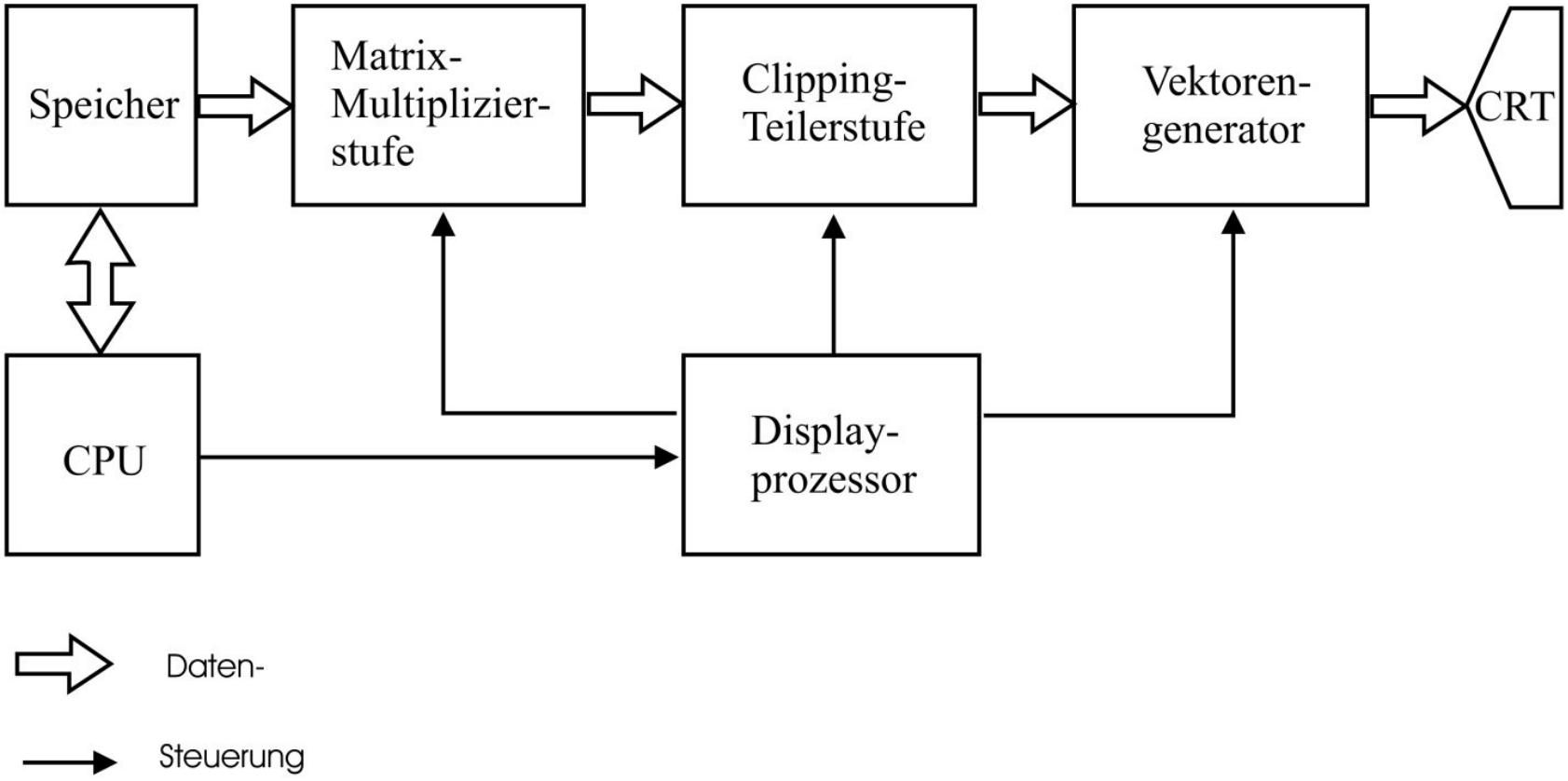
Die Rasterbildtechnik

Anfänge der Computergraphik: Vektordisplays



Vorteile von Vektordisplays:

- geringer Speicheraufwand (Display-Liste statt Pixelfeld)
- schnelles Umschalten der angezeigten Information (Animationen)

Nachteile:

- begrenzte Komplexität der darstellbaren Information (Vektoren)
- eingeschränkte Technologie

Heute:

Dominanz der Rastertechnik

Beispiele:

- Vektorbildschirm $\longrightarrow$ Rasterbildschirm
- Plotter $\longrightarrow$ Drucker
- Telex $\longrightarrow$ Telefax
- Buchdruck $\longrightarrow$ Offsettdruck
- analoge Kopie $\longrightarrow$ digitale (Raster-)Kopie

$\longrightarrow$ Algorithmen zur Rasterung kontinuierlicher Daten notwendig

Beispiele:

- geometrische Objekte (Linien, Kurven)
- Schriftdarstellung
- allgemeine Flächen (Dreiecke)
- Bilder

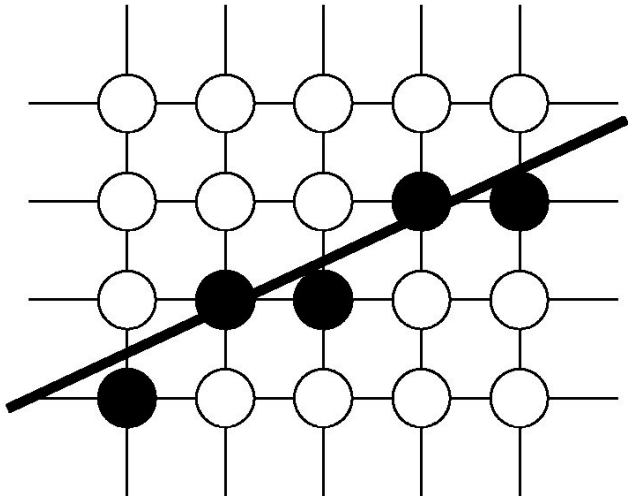
Probleme:

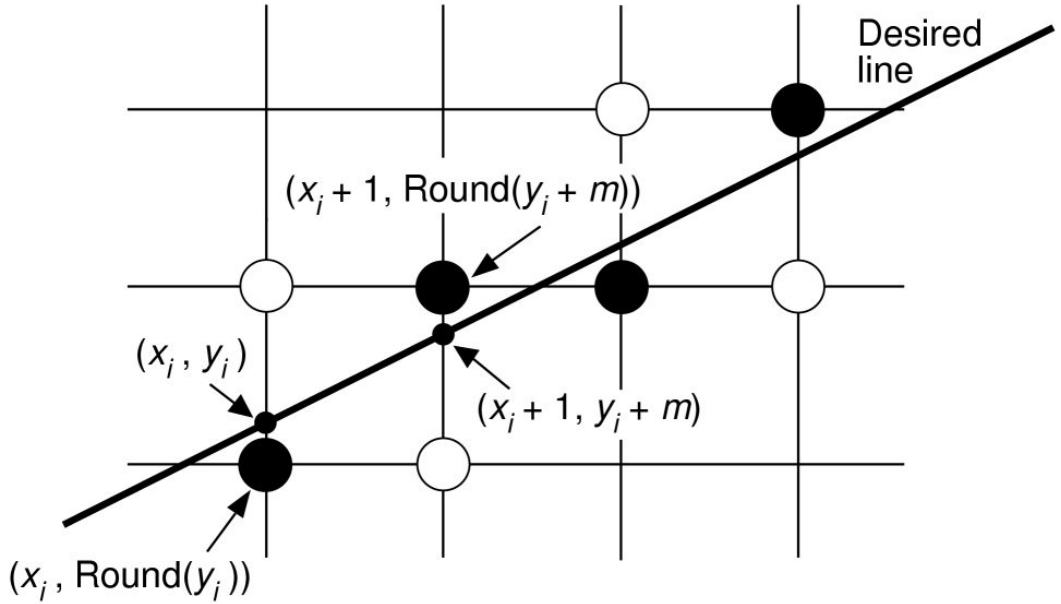
- Effizienz
- Aliasing (Treppenbildung, Moirémuster)
- Rauschen

Rasterung von Linien

ein erster Algorithmus:

```
void Line(                                  /* Annahme: -1 ≤ m ≤ 1, x0 < x1 */
    int x0, int y0,                         /* Linker Endpunkt */
    int x1, int y1,                         /* Recher Endpunkt */
    int value)                              /* Pixelfarbwert */
{
    int x;                                  /* x läuft von x0 nach x1 */
    double m = (y1 - y0) / (x1 - x0);       /* Steigung */
    double y = y0;
    for (x=x0; x<= x1; x++){
      WritePixel(x, Round(y), value);       /* Setze Pixel auf Farbwert */
      y += m;                               /* Erhöhe y um m */
    }
}
```





Probleme:

- Steigung $|m| <= 1$
- ineffizient durch komplizierte Fließkommar-Arithmetik

geht es besser ?

Bresenham- oder Midpoint-Algorithmus

$\longrightarrow$ Arbeitet mit Integerwerten

$\longrightarrow$ Schnelligkeitsargument heute nicht mehr so wesentlich

$\longrightarrow$ aber: Prototyp vieler inkrementeller Verfahren

Strecke:

$$y = \frac{dy}{dx}\, x + B$$

bzw.:

$$0 = dy \cdot x + B \cdot dx - dx \cdot y$$

Darstellung als implizite Funktion:

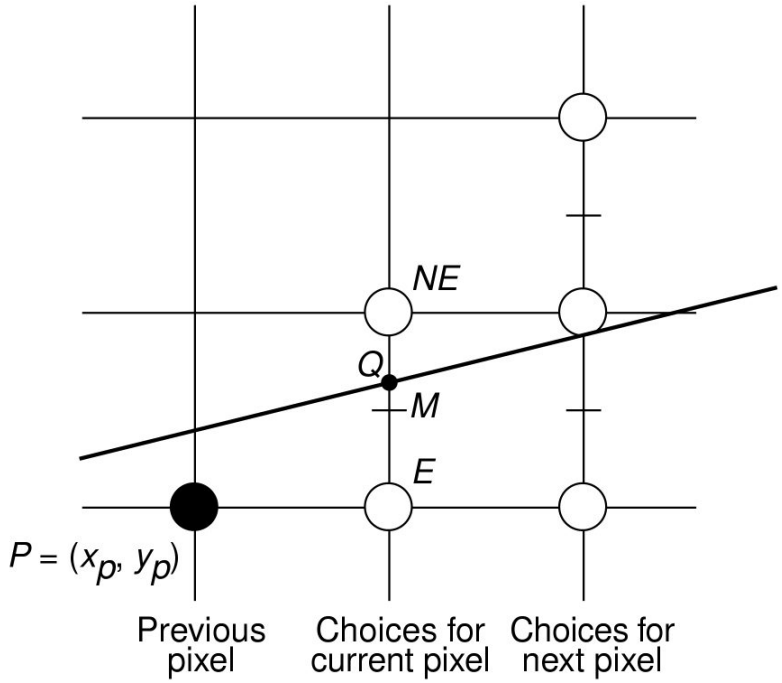
$$F(x, y) = ax + by + c = 0$$

mit $a = dy, b = -dx$ und $c = B \cdot\ dx$

$\longrightarrow$ $F(x, y)$ zerteilt Raum in zwei Bereiche:

$$F(x, y) > 0 \quad \text{bzw.} \quad F(x, y) \leq 0$$

$\longrightarrow$ Annahme für zu zeichnende Linie : $|m| \leq 1$



$\Rightarrow$ Entscheidung, welches Pixel als nächstes gesetzt wird: N oder NE

Entscheidungsvariable für N oder NE: Mittelpunkt zwschen N und NE

$\rightarrow$ in implizite Formulierung einsetzen:

$$d = F(x_p + 1, y_p + \frac{1}{2})$$

$\Rightarrow$ d $>$ 0: wähle NE sonst: wähle E

$$d_{old} = dy\ (x_p + 1) - dx\ (y_p + \frac{1}{2}) + B$$

Wird E gewählt, so gilt:

$$d_{new} = dy\ (x_p + 2) - dx\ (y_p + \frac{1}{2}) + B$$

und das Inkrement für d ist: $\Delta_E = d_{new} - d_{old} = dy$

Wird NE gewählt, so gilt:

$$d_{new} = dy\ (x_p + 2) - dx\ (y_p + \frac{3}{2}) + B$$

und das Inkrement für d ist: $\Delta_{NE} = d_{new} - d_{old} = dy - dx$

Bestimmung des initialen Wertes für d:

Das erste Pixel sitzt auf (x_0, y_0), daher gilt:

$$d_{start} = dy\ (x_0 + 1) - dx\ (y_0 + \frac{1}{2}) + B = F(x_0, y_0) + dy - \frac{dx}{2}$$

da (x_0, y_0) per Definition auf der Linie ist gilt $F(x_0, y_0) = 0$:

$$d_{start} = dy - \frac{dx}{2}$$

$\Rightarrow$ nehme alles mal zwei, um ganzzahlige Werte zu erhalten

(unproblematisch, da nur das Vorzeichen von d entscheidet)

$$\begin{aligned} d_{start} &= 2dy - dx \\ \Delta_{NE} &= 2(dy - dx) \\ \Delta_{E} &= 2dy \end{aligned}$$

Midpoint-Algorithmus

```
void MidpointLine(int x0, int y0, int x1, int y1, int value)
{
    int dx = x1 - x0;
    int dy = y1 - y0;
    d = 2 * dy - dx;                /* Initialer Wert von d */
    int incrE = 2 * dy;             /* Inkrement für Wahl von E */
    int incrNE = 2 * (dy - dx);     /* Inkrement für Wahl von NE */
    int x = x0;
    int y = y0;

    WritePixel(x, y, value);        /* Startpunkt */

    while (x < x1){
      if (d ≤ 0) {                  /* Wähle E */
        d += incrE;
        x++;
      }
```

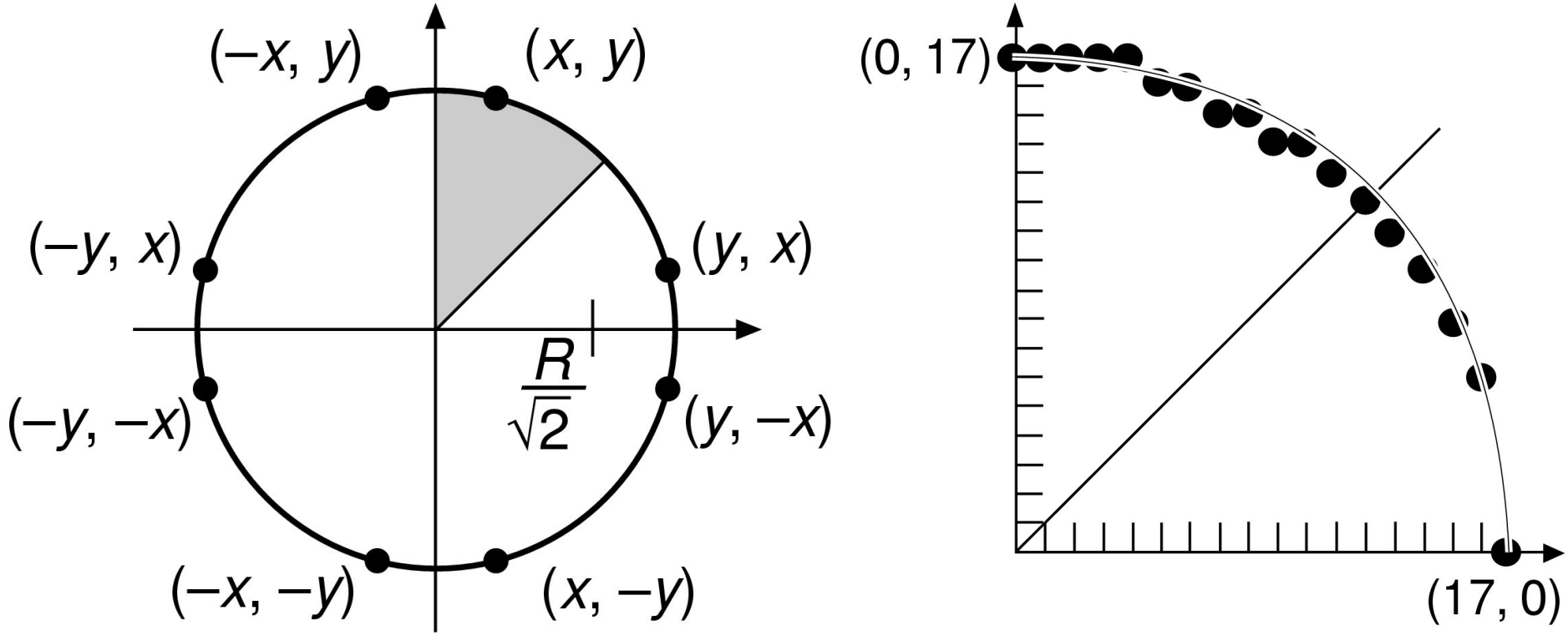
```
        else
      {                                  /* Wähle NE */
        d += incrNE;
        x++;
        y++;
      }
      WritePixel(x, y, value);
    }
}
```

Rasterung von Kreisen

→ Algorithmus zeichnet ein Achtel eines Kreises

→ daher: jeder Punkt wird sieben mal gespiegelt:

```
void CirclePoints(int x, int y, int value)
{   WritePixel(x, y, value);
    WritePixel(y, x, value);
    WritePixel(y, -x, value);
    WritePixel(x, -y, value);
    WritePixel(-x, -y, value);
    WritePixel(-y, -x, value);
    WritePixel(-y, x, value);
    WritePixel(-x, y, value);
}     /* Circle Points */
```



aus: Foley et al. Computer Graphics, Principles and Practice

Für die Funktion (R = Radius)

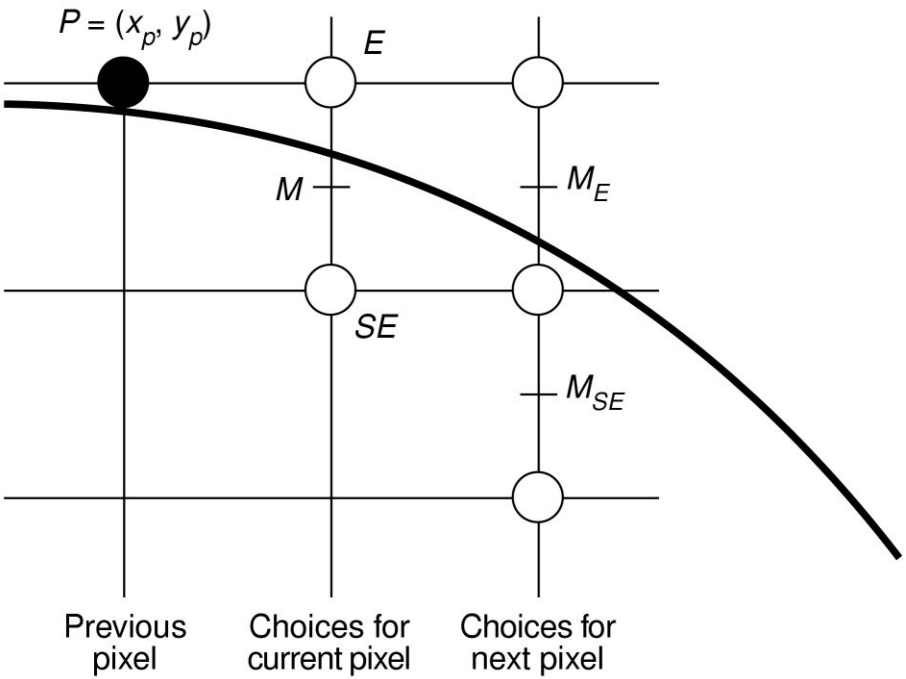
$$F(x, y) = x^2 + y^2 - R^2$$

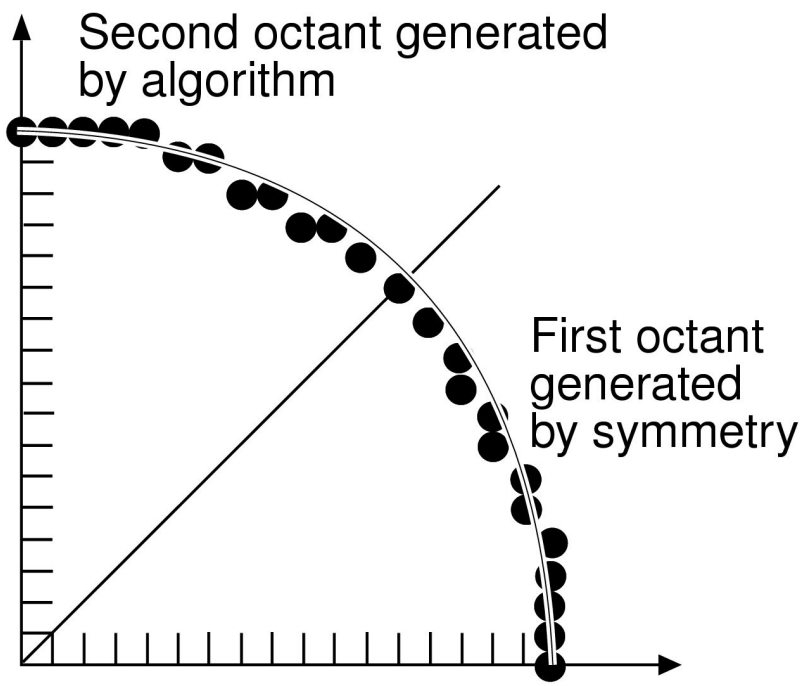
gilt:

- F(x,y) = 0: Punkt auf dem Kreis,
- F(x,y) < 0 Punkt innerhalb des Kreises
- F(x,y) > 0 Punkt außerhalb des Kreises.

Liegt der Mittelpunkt M zwischen E und SE

- außerhalb des Kreises, so wird SE gewählt
- innerhalb des Kreises, so wird E gewählt





aus: Foley et al. Computer Graphics, Principles and Practice

Berechnung der Entscheidungsvariable

Variable d im aktuellen Schritt:

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

- Falls $d_{alt} < 0$, wird E gewählt:

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

- Inkrement:

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

Falls $d_{alt} \geq 0$, wird SE gewählt:

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

Inkrement:

$$\Delta_{SE} = d_{neu} - d_{alt} = 2\,x_p - 2\,y_p + 5$$

Linienkonvertierung: Δ_E, Δ_{NE} waren Konstanten.

Hier: lineare Ausdrücke, abhängig vom aktuellen Punkt (x_p,y_p)

Anfangswert für d:

$$d = F(1, R - \frac{1}{2}) = 1 + (R^2 - R + \frac{1}{4}) - R^2 = \frac{5}{4} - R$$

```
void MidpointCircle (int radius, int value);    {Mittelpunkt im Ursprung}
int x, y;
float d;
begin
    x = 0;                                  /* Initialisierung */
    y = radius;
    d = 5/4 - radius;
    CirclePoints (x, y, value);
    while (y > x) {
            if (d < 0) {                        /* Auswahl von E */
                    d = d + 2 * x +3;
                    x = x + 1;
                }
            else
```

```
            {                          /* Auswahl von SE */
                d = d + 2 * (x-y) + 5;
                x = x + 1;
                y = y - 1;
            }
        CirclePoints (x, y, value)
    }
} /* MidpointCircle */
```

Ganzzahlige Kreiskonvertierung

bisher: d nicht ganzzahlig $\Rightarrow$ Fließkommar-Arithmetik

Neue Entscheidungsvariable:

$$h ::= d - \frac{1}{4}$$

$\Rightarrow$ Anfangswert: $h_{start} = 1 - R$

Entscheidungskriterium: $d < 0$ wird zu $h < -\frac{1}{4}$.

$\rightarrow$ da die Berechnung von h mit ganzzahligen Werten beginnt und h nur ganzzahlig (Δ_E, Δ_{SE}) inkrementiert wird, kann $h < -\frac{1}{4}$ durch $h < 0$ ersetzt werden.

Ganzzahliger Algorithmus

```
void MidpointCircle (int radius, int value)
{    int x = 0;                                  /* Initialisierung */
     int y = radius;
     int d = 1 - radius;
     CirclePoints (x, y, value);
     while (y > x) do {
          if (d < 0) {                           /* Auswahl von E */
               d = d + 2 * x +3;
               x = x + 1
          } else {                               /* Auswahl von SE */
               d := d + 2 * (x-y) + 5;
               x := x + 1;
               y := y - 1;
```

```
        }
        CirclePoints (x, y, value)
    }                                   {while}
}                                     {MidpointCircle}
```

Kreiskonvertierung mit Differenzen zweiter Ordnung

Vermeidung von Multiplikation: Differenzen zweiter Ordnung

$\rightarrow$ Wahl von E: Bewertungspunkt $(x_p, y_p) \rightarrow (x_p + 1, y_p)$

$\Rightarrow$ Differenz erster Ordnung:

$$\Delta_{E_{alt}}(x_p, y_p) = 2\ x_p + 3$$

Wegen

$$\Delta_{E_{neu}}(x_p + 1, y_p) = 2\ (x_p + 1) + 3$$

ist Differenz zweiter Ordnung:

$$\Delta_{E_{neu}} - \Delta_{E_{alt}} = 2$$

Andererseits

$$\Delta_{SE_{neu}}(x_p + 1, y_p) = 2\,(x_p + 1) - 2\,y_p + 5$$

und die Differenz zweiter Ordnung wird ebenfalls

$$\Delta_{SE_{neu}} - \Delta_{SE_{alt}} = 2$$

Wenn SE in der laufenden Iteration gewählt wurde, wandert der Bewertungspunkt von (x_p, y_p) nach $(x_p + 1, y_p - 1)$. Folglich wird

$$\Delta_{E_{neu}}(x_p + 1, y_p - 1) = 2\,(x_p + 1) + 3$$

und noch einmal

$$\Delta_{E_{neu}} - \Delta_{E_{alt}} = 2$$

Schließlich wird in diesem Fall

$$\Delta_{SE_{neu}}(x_p + 1, y_p - 1) = 2\,(x_p + 1) - 2\,(y_p - 1) + 5$$

also

$$\Delta_{SE_{neu}} - \Delta_{SE_{alt}} = 4$$

Zusammenfassung

- Rastertechnik ersetzte alle anderen Techniken
- Rasteralgorithmen sind Basis der Computergraphik
- Rasteralgorithmen: Beispiele für inkrementelle Verfahren
- bisher offene Probleme: Treppeneffekte