UNIVERSITÄT BONN

RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT BONN

INSTITUT FÜR INFORMATIK DER UNIVERSITÄT BONN

Einführung in die Computergrafik

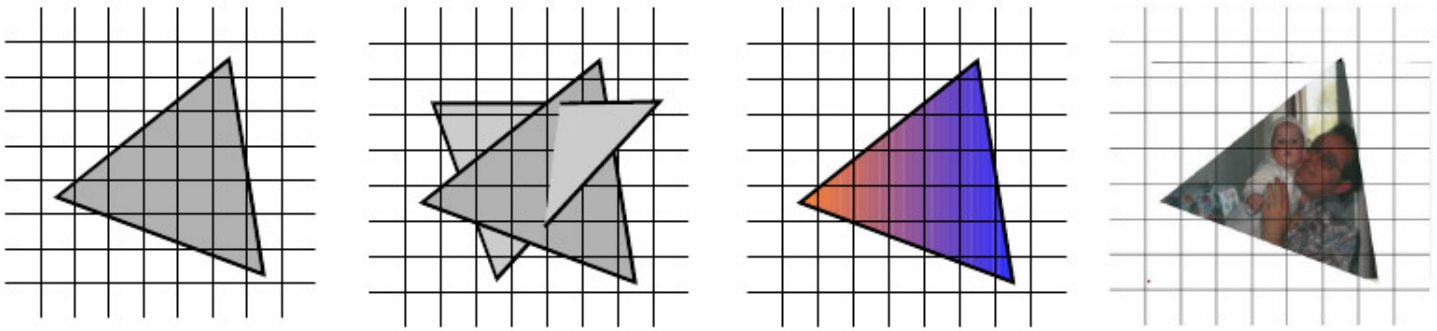
Kapitel 9: Rasteralgorithmen

Prof. Dr. Matthias Hullin

Institut für Informatik
Abteilung 2: Visual Computing
Universität Bonn

16. Mai 2020

Im Bilderzeugungsteil (ICS) des Rastergraphiksystems werden die einzelnen graphischen Primitiva (Dreiecke, Polygone), aus den die Szene zusammengesetzt ist, in Rasterpunkte (Pixel) zerlegt.



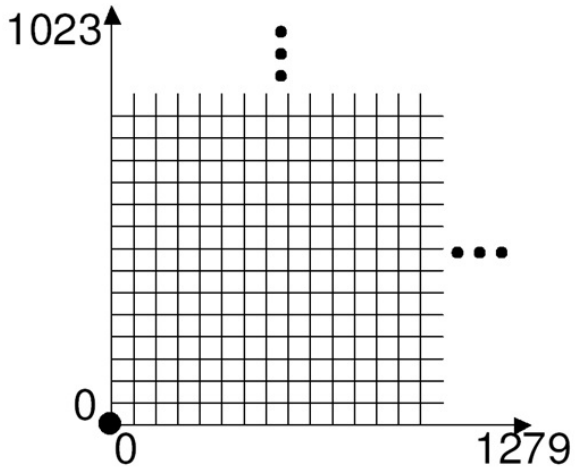
Für jedes Pixel werden dabei zusätzlich Operationen

- zur Verdeckungsrechnung (inklusive Transparenz)
- zum Shading
- und zur Texturierung

durchgeführt.

Nachdem ein Bild gerastert wurde, befindet es sich im sogenannten Framebuffer , von dem es aus dann durch die Hardware auf einen Monitor darstellt werden kann.

Ein Framebuffer ist dabei ein einfaches lineares Speicherarray, welches durch 2D-Adressen angesprochen werden kann.



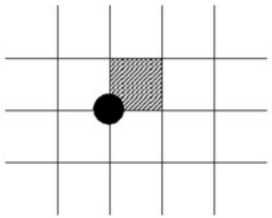
Die Startadresse $(0, 0)$ ist dabei

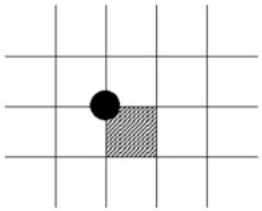
- links oben (X11, AWT)
- links unten (OpenGL)

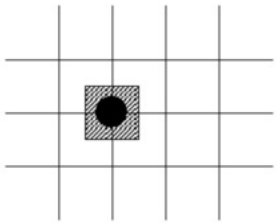
Der Framebuffer kann beschrieben und gelesen werden, indem man den gewünschten Wert einträgt bzw. abfragt:

setpixel(int x , int y , value)
value = readpixel(int x , int y)

Zu beachten ist, wo der Pixel relativ zur Scanline gesehen liegt, d.h. wo der Pixel im Verhältnis zum eigentlich angesprochenen Punkt liegt:

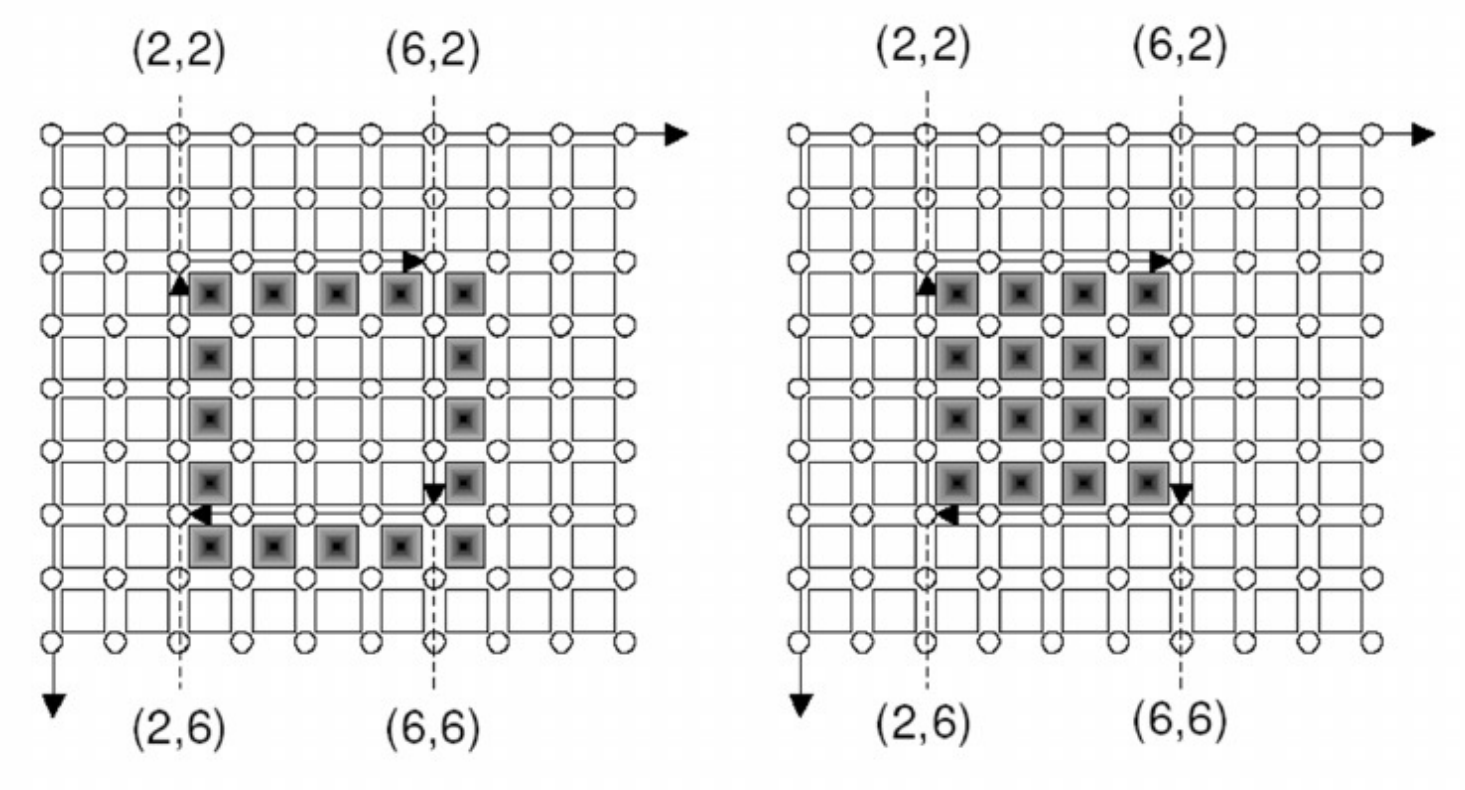

OpenGL


Java


X11 / Foley

Wie man sieht, ist dies bis auf eine Verschiebung von einem halben bzw. einem Pixel äquivalent.
Dies ist insofern wichtig, weil beim Zeichnen mit Floating Point Koordinaten entschieden werden muss, wo der Pixel gezeichnet wird, und dies **konsistent** sein muss.

Diese Konsistenz ist wichtig, wie man am folgenden Beispiel sieht:



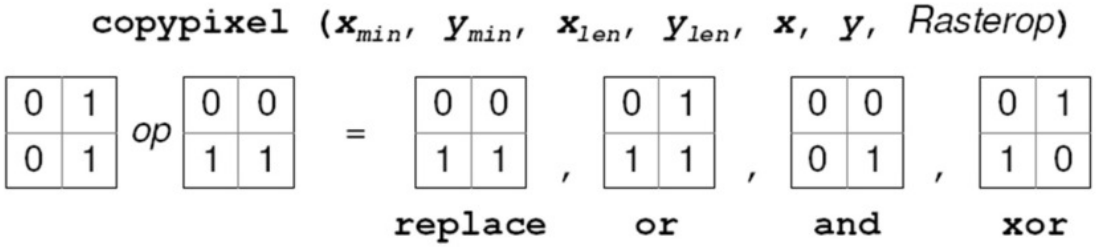
Es macht in diesem Fall einen großen Unterschied, ob man ein Quadrat umrandet oder füllt. Das umrandete Quadrat ist um einen Pixel breiter und höher als das gefüllte Quadrat.

Die Farbe für aktuelle Zeichenoperationen ist

- kein Aufrufparameter
- sondern einer Zustandsvariable (wie andere Attribute: Linienstil, Füllstil)

Möchte man rechteckige Bereiche im Framebuffer verschieben (z.B. Fenster, Icons, Scrolltext), so wird dies hardware-unterstützt durchgeführt:

- Die Adressumrechnung erfolgt meist per Hardware
- Dies geschieht z.B. durch einen Block-Transfer (mit DMA) bitBLT



Ziel: Rastern einer Linie, wobei Anfangs- und Endpunkt auf dem Raster liegen.

Annahme: Für die Steigung m der Linie bzw. Geraden gilt: $-1 \leq m \leq 1$

Die Steigung kann berechnet werden:

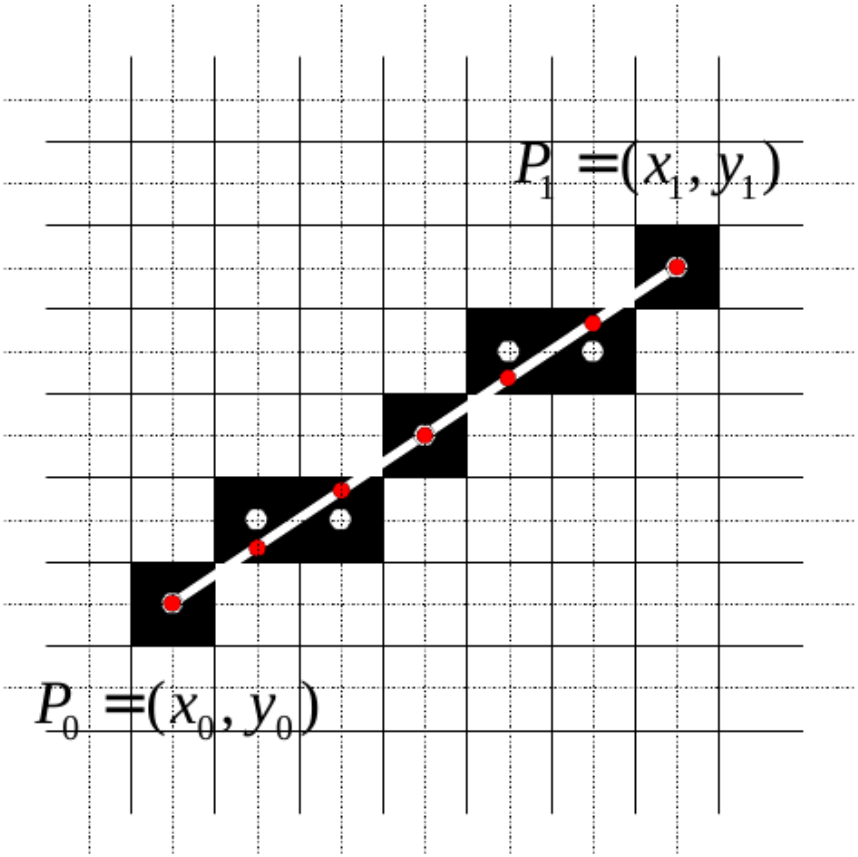
$$m = \frac{\Delta y}{\Delta x} = \frac{y_1 - y_0}{x_1 - x_0}$$

Dann ist der i-te y-Wert:

$$y_i = \frac{\Delta y}{\Delta x} \cdot x_i + b$$

für $i = 1, \ldots, \Delta x$ und $x_i = x_0 + i$.
Somit wird der nähere Pixel bezogen auf y_i gezeichnet, d.h. der Pixel

$$(x_i, round(y_i)) = \left(x_i, Floor\left(y_i + \frac{1}{2}\right)\right)$$



Problem: Diese Vorgehensweise ist ineffizient. Jede Operation benötigt

- Floating Point Multiplikation
- Additionen
- Rundung

1. Ansatz: Vermeide Multiplikation durch inkrementellen Algorithmus

$$\begin{aligned} y_i &= \frac{\Delta y}{\Delta x} \cdot x_i + b \\ &= \frac{\Delta y}{\Delta x} \cdot (x_{i-1} + (x_i - x_{i-1})) + b \\ &= y_{i-1} + \frac{\Delta y}{\Delta x} \cdot (x_i - x_{i-1}) \end{aligned}$$

Da der Abstand zweier Pixel bezüglich der x-Koordinate 1 ist, d.h. $x_i - x_{i-1} = 1 \; \forall i \in \mathbb{N}_0$, gilt:

$$y_i = y_{i-1} + \frac{\Delta y}{\Delta x}$$

Somit ergibt sich der i-te y-Wert aus seinem Vorgänger plus der Steigung der Geraden.

2. Ansatz: Verzichte zusätzlich auf Berechnung der Steigung (Floating Point Wert) und benutze Integer-Arithmetik

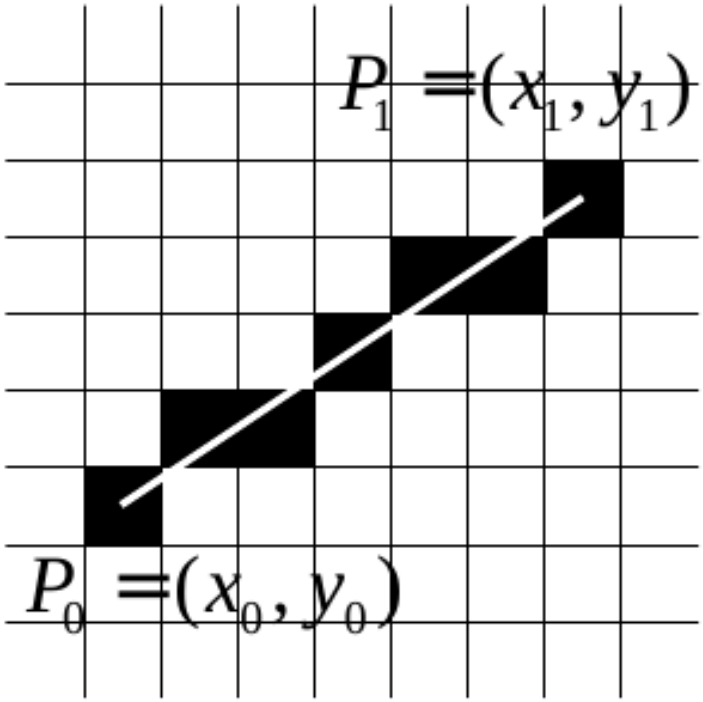
Annahme: Steigung m ist zwischen 0 und 1, d.h. $0 \leq m \leq 1$.

Dies ist keine Einschränkung, da ansonsten folgende Operation durchgeführt und am Ende rückgängig gemacht werden muss:

- $m < 0$: Spiegelung an der x-Achse
- $m > 1$: Spiegelung an der Achse $y = x$

Unter obiger Annahme gelten zudem besondere Eigenschaften:

$$\Delta x = x_1 - x_0 \geq 0$$
$$\Delta y = y_1 - y_0 \geq 0$$
$$\Delta x \geq \Delta y$$



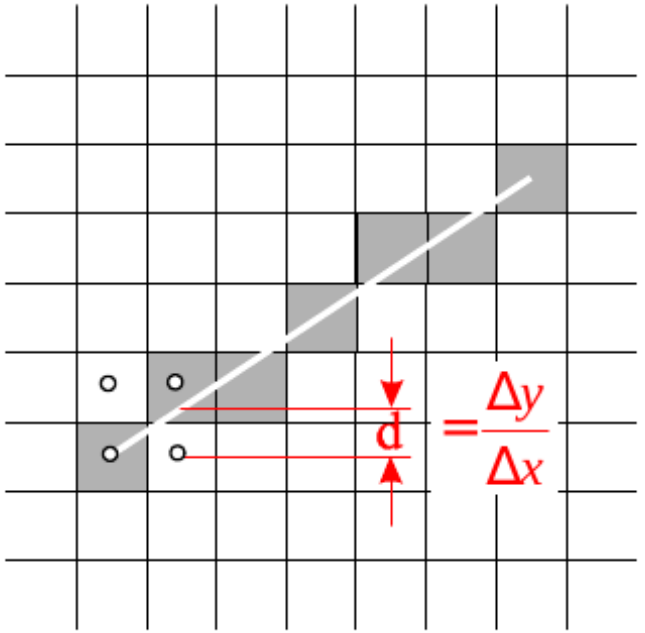
Frage: Ist Abstand d zur Geraden $\leq \frac{1}{2}$ oder $> \frac{1}{2}$?

Dazu definieren wir uns eine Entscheidungsvariable:

$$E := \frac{\Delta y}{\Delta x} - \frac{1}{2}$$

$$E' := 2\Delta x \cdot E = 2\Delta y - \Delta x$$

$E \leq 0$	$E > 0$
$x := x + 1$	$x := x + 1$ $y := y + 1$
$E := E + \frac{\Delta y}{\Delta x}$	$E := E + \frac{\Delta y}{\Delta x} - 1$
$E' := E' + 2\Delta y$	$E' := E' + 2\Delta y - 2\Delta x$



Integer-Arithmetik in blau

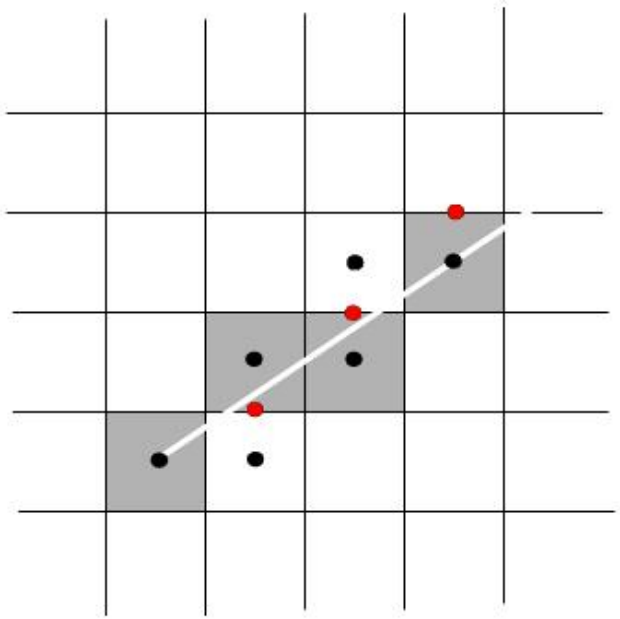
Alternative: Mittelpunktalgorithmus

- Annahmen wie beim Bresenham-Algorithmus
- Welches Pixel soll als nächstes gewählt werden?

$$(x+1, y) \qquad \text{oder} \qquad (x+1, y+1)$$

Zu entscheiden ist also, auf welcher Seite der Geraden der Mittelpunkt M liegt:

$$M = \left(x+1, y+\frac{1}{2}\right)$$



Verwende dazu die implizite Geradengleichung:

$$g(x,y) = \Delta y \cdot x - \Delta x \cdot y + (\Delta x \cdot y_1 - \Delta y \cdot x_1)$$

Nun entscheide wie folgt:

$$g\left(x+1, y+\frac{1}{2}\right) \leq 0 \qquad \text{wähle Pixel } (x+1, y)$$

$$g\left(x+1, y+\frac{1}{2}\right) > 0 \qquad \text{wähle Pixel } (x+1, y+1)$$

Inkrementelle Berechnung von $g\left(x+1, y+\frac{1}{2}\right)$:
Setzt man den Mittelpunkt in die implizite Gleichung ein, so ergibt sich

$$g\left(x+1, y+\frac{1}{2}\right) = \Delta y \cdot (x+1) - \Delta x \cdot \left(y+\frac{1}{2}\right) + C$$

mit $C = \Delta y \cdot x_1 - \Delta x \cdot y_1$. Nun gibt es die beiden Fälle:

- **Pixel $(x+1, y)$ gewählt:** Dann gilt für den nächsten zu betrachtenden Punkt $(x+1, y)$:

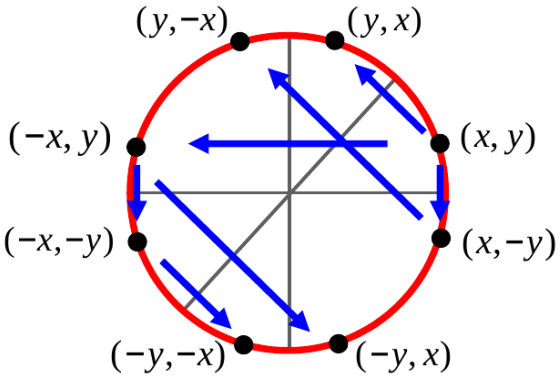
$$\begin{aligned} g\left(x+2, y+\frac{1}{2}\right) &= \Delta y \cdot (x+2) - \Delta x \cdot \left(y+\frac{1}{2}\right) + C \\ &= g\left(x+1, y+\frac{1}{2}\right) + \Delta y \end{aligned}$$

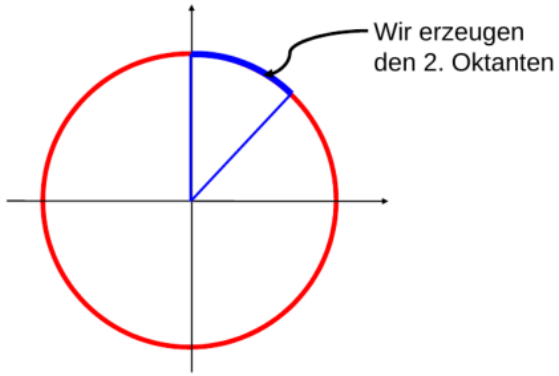
- **Pixel $(x+1, y+1)$ gewählt:** Dann gilt für den nächsten zu betrachtenden Punkt $(x+1, y+1)$:

$$\begin{aligned} g\left(x+2, y+\frac{3}{2}\right) &= \Delta y \cdot (x+2) - \Delta x \cdot \left(y+\frac{3}{2}\right) + C \\ &= g\left(x+1, y+\frac{1}{2}\right) + \Delta y - \Delta x \end{aligned}$$

Somit lässt sich der nächste Punkt aus dessen Vorgänger plus einer entsprechenden Addition berechnen.

Bei einem Kreis nutzen wir dessen Symmetrie aus und rastern nur den zweiten Oktanten. Die andere Oktanten werden durch Spiegelungen erzeugt.





Der Kreisbogen im zweiten Oktanten hat dabei folgende vorteilhafte Eigenschafen:

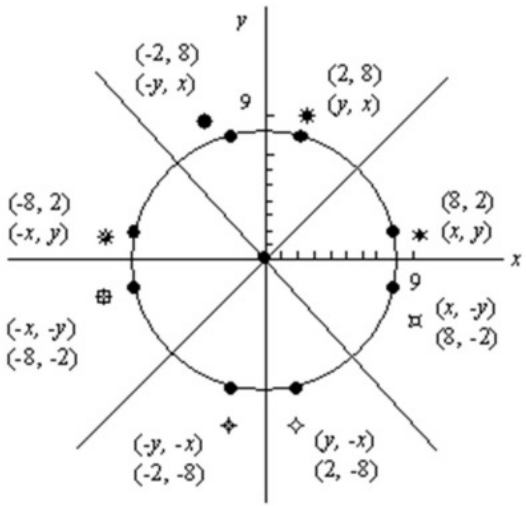
- Eindeutigkeit
- keine vertikalen Tangenten
- Betrag der Steigung beschränkt: $|m| \leq 1$

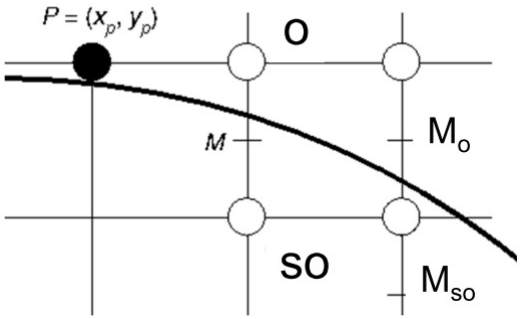
Somit kann der Bresenham- und der Mittelpunktalgorithmus angewendet werden.

Wir verwendet die implizite Kreisgleichung und betrachten den Punkt $\left(x+1, y-\frac{1}{2}\right)$:

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

$$f\left(x+1, y-\frac{1}{2}\right) = (x+1)^2 + \left(y-\frac{1}{2}\right)^2 - R^2$$





Nun verfahren wir analog zur obigen Variante:

- **O: Pixel** $(x+1, y)$ **gewählt:** Dann gilt für den nächsten zu betrachtenden Punkt $(x+1, y)$:

$$f\left(x+2, y-\frac{1}{2}\right) = (x+2)^2 + \left(y-\frac{1}{2}\right)^2 - R^2$$
$$= f\left(x+1, y-\frac{1}{2}\right) + 2x + 3$$

- **SO: Pixel** $(x+1, y-1)$ **gewählt:** Dann gilt für den nächsten zu betrachtenden Punkt $(x+1, y-1)$:

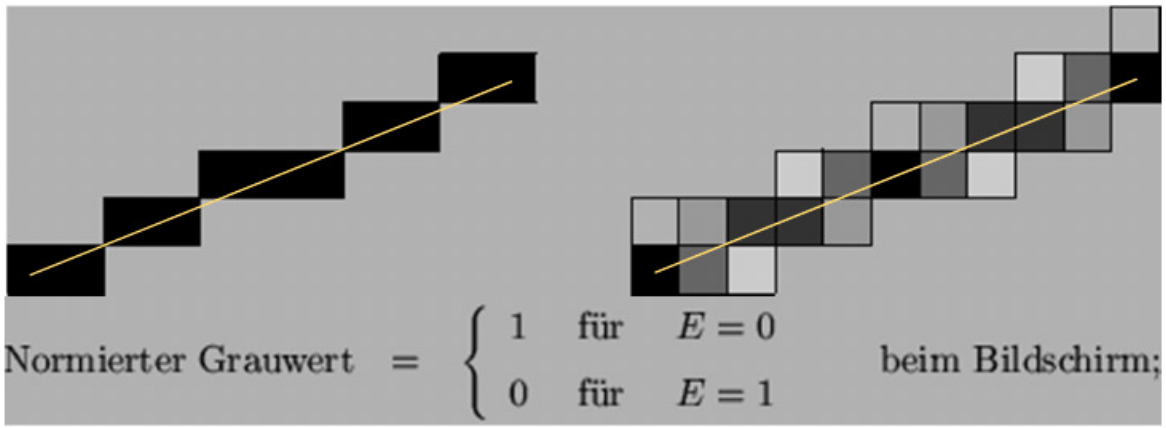
$$f\left(x+2, y-\frac{3}{2}\right) = (x+2)^2 + \left(y-\frac{3}{2}\right)^2 - R^2$$
$$= f\left(x+1, y-\frac{1}{2}\right) + 2x - 2y + 5$$

Glätten von gerasterten Linien

Beim Rastern kommt es bei niedriger Pixelauflösung ($\ll 300dpi$) zu Treppenbildung.

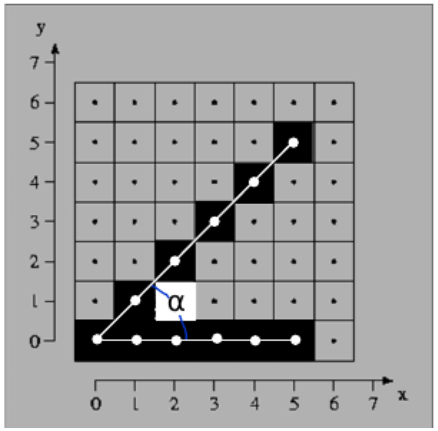
Um diesen Effekt zu minimieren, werden Kanten geglättet. Dies ist sowohl mit Geraden als auch mit gekrümmten Linien (wie beim Kreis) möglich.

Dabei wird die Helligkeit (in dem Fall Grauwerte) entsprechend des vertikalen Abstands der Pixel von der eigentlichen Linie linear skaliert:



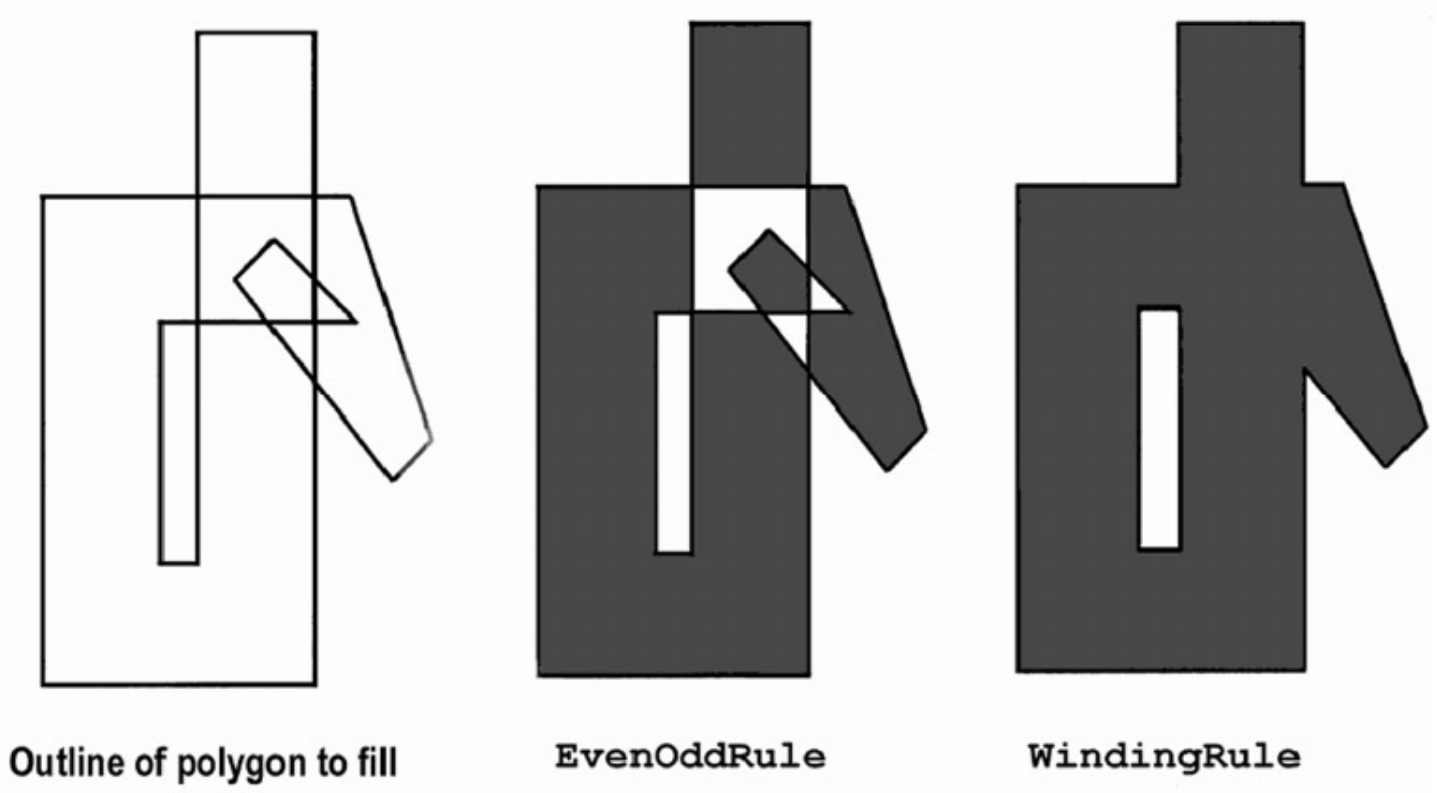
Dabei muss eine Gammakorrektur vorgenommen werden:

- Bei Strecken in geringem Abstand voneinander muss verhindert werden, dass hell gesetzte Pixel von einem benachbarten Pixel wieder dunkel gesetzt werden
  $\rightarrow$ Einfachste Lösung: Speichere neuen Helligkeitswert nur, falls er größer als der bereits gespeicherte ist
- Strecken mit Neigung $0 \leq \alpha \leq \pi/4$ sind länger als achsenparallele Strecken mit gleicher Pixelanzahl
  $\rightarrow$ Skaliere den Grauwert um den Faktor $1/\cos(\alpha)$, um denselben Helligkeitseindruck zu erhalten



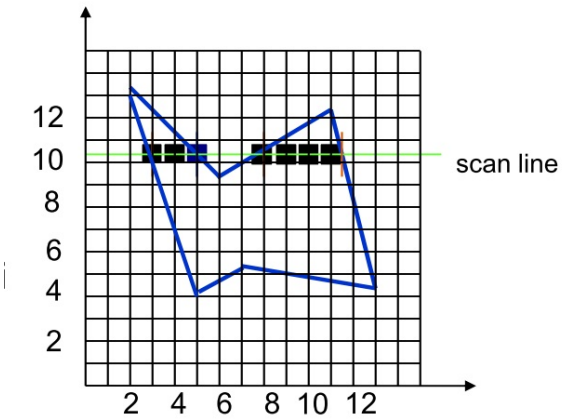
Weitere Verfahren später im Kapitel über Anti-Aliasing.

Füllen von Polygonen



Idee: Scan-Line-Algorithmus

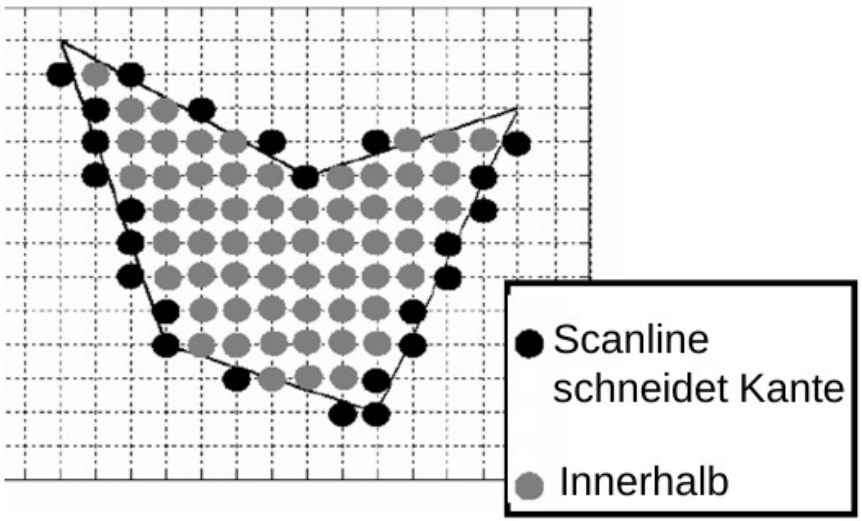
- Finde Schnittpunkte der Scan-Line mit allen Kanten des Polygons
- Sortiere Schnittpunkte nach wachsender x-Koordinate
- Fülle alle Pixel zwischen Paaren aufeinanderfolgender Schnittpunkte die im Inneren des Polygons liegen
- Nutze dazu die Regel der ungeraden Parität:
  - Parität zu Beginn ist gleich 0
  - Inkrementiere Parität mit jedem Schnittpunkt um Eins
  - Zeichne Pixel falls Parität ungerade

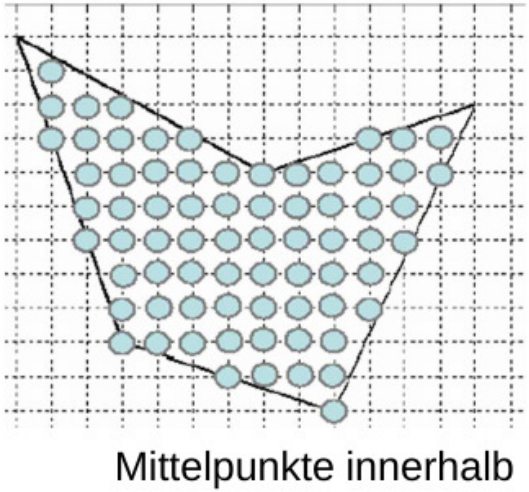


Bemerkung: Alle Polygoneckpunkte sind dabei auf Integerwerte gerundet.

Verbesserungen: Extrema der Teilstücke (Spans)

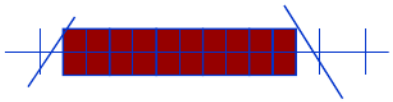
Verwende den Mittelpunktalgorithmus auf den Kanten und zeichne nur Pixel innerhalb des Polygons:



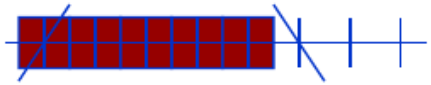


Sonderfälle:

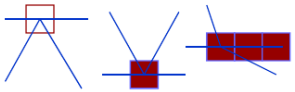
- Schnitt Floating-Point-Wert
  → Innen abrunden, Außen aufrunden



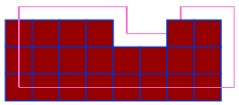
- Schnitt Integerwert
  → Linker Endpunkt eines Spans: Innen
  Rechter Endpunkt : Außen



- Gemeinsamer Kantenpunkt (Vertex) von Nachbarpolygonen
  → Nur y_{min}-Knoten einer Kante wird zur Berechnung der Parität gezählt

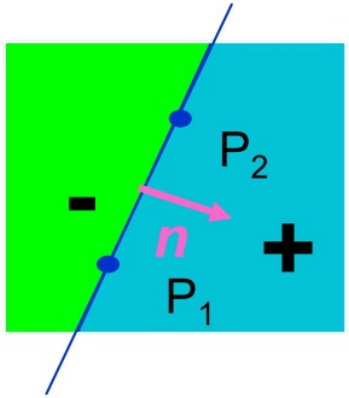


- Horizontale Kanten
  → Pixel horizontaler Kanten werden zur Berechnung der Parität nicht gezählt



Gegeben:

$$P_1 = (x_1, y_1) \qquad P_2 = (x_2, y_2)$$
$$n = (\Delta y, -\Delta x) = (y_2 - y_1, -(x_2 - x_1))$$

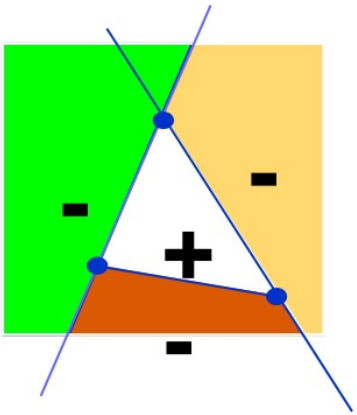


Dann ist die implizite Form der Kante:

$$E(x, y) = \langle (x, y) - P_1 | n \rangle$$

Setzt man nun einen Punkt ein, so liefert obige implizite Darstellung den Abstand zur Kante:

$$\begin{aligned} E(x, y) &= \langle (x, y) - P_1 | n \rangle \\ &= \langle (x - x_1, y - y_1) | (\Delta y, -\Delta x) \rangle \\ &= (x - x_1) \cdot \Delta y + (y - y_1) \cdot (-\Delta x) \\ &= x\Delta y - x_1 \Delta y - y\Delta x + y_1 \Delta x \end{aligned}$$



Damit der Abstand korrekt ist, muss die Normale n normiert sein, d.h.

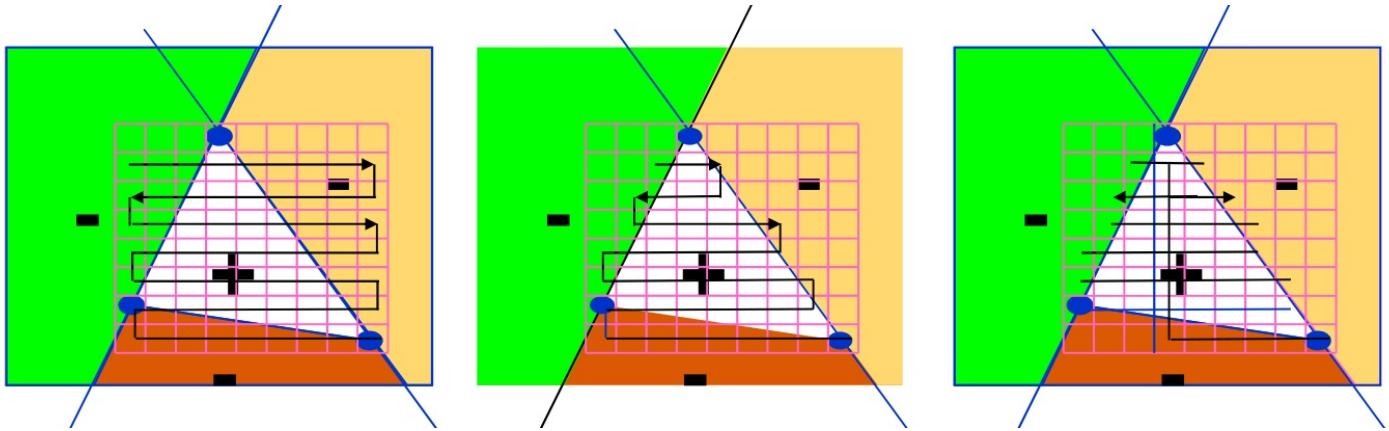
$$\|n\|_2 = \sqrt{\Delta x^2 + \Delta y^2} = 1$$

Füllen von Dreiecken (Pineda 1988)

Die Abstandsfunktion E_i der i-ten Kante kann inkrementell aufgebaut werden:

$$E_i(x+1, y) = E_i(x, y) + (\Delta y)_i$$
$$E_i(x, y+1) = E_i(x, y) - (\Delta x)_i$$

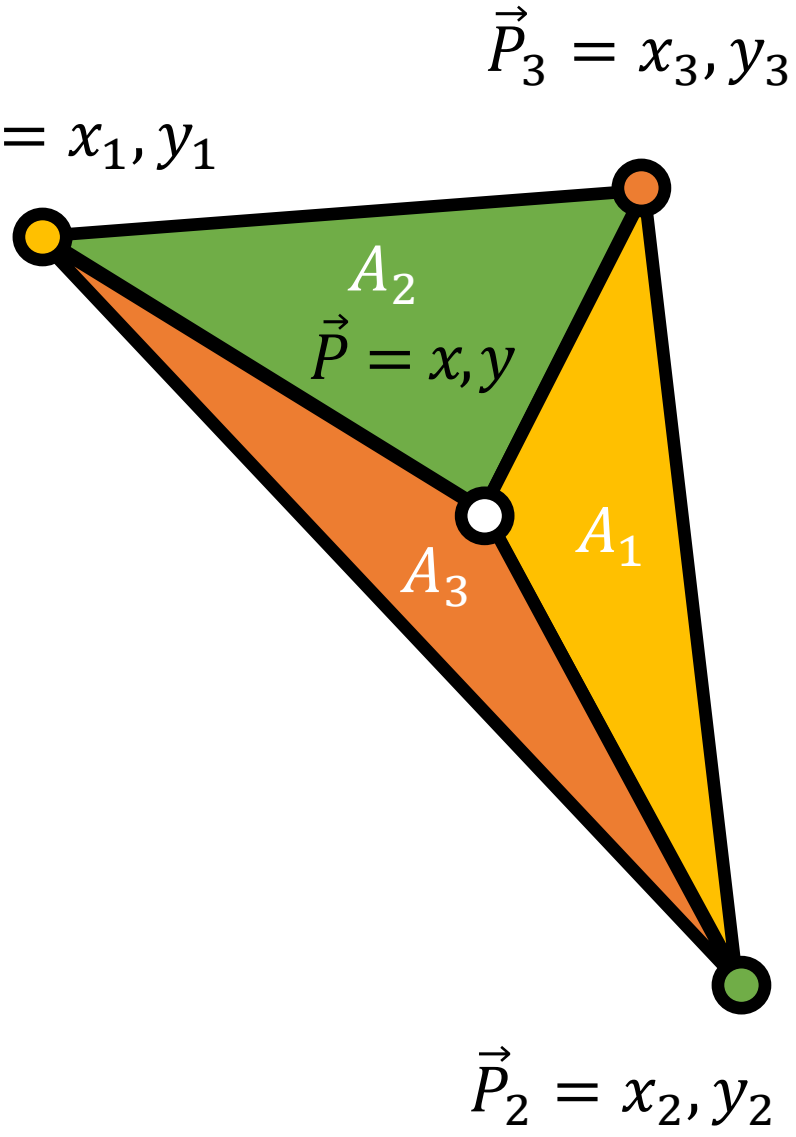
Anschließend kann auf verschiedene Arten traversiert werden:



Füllen von Dreiecken II: so machen es heutige GPUs

- Verwende baryzentrische Koordinaten (vgl. Kapitel “Euklidische Geometrie”)

Refresher: Baryzentrische Koordinaten



- Baryzentrische Koordinaten (Gewichte) von Punkt $\vec{P}$:

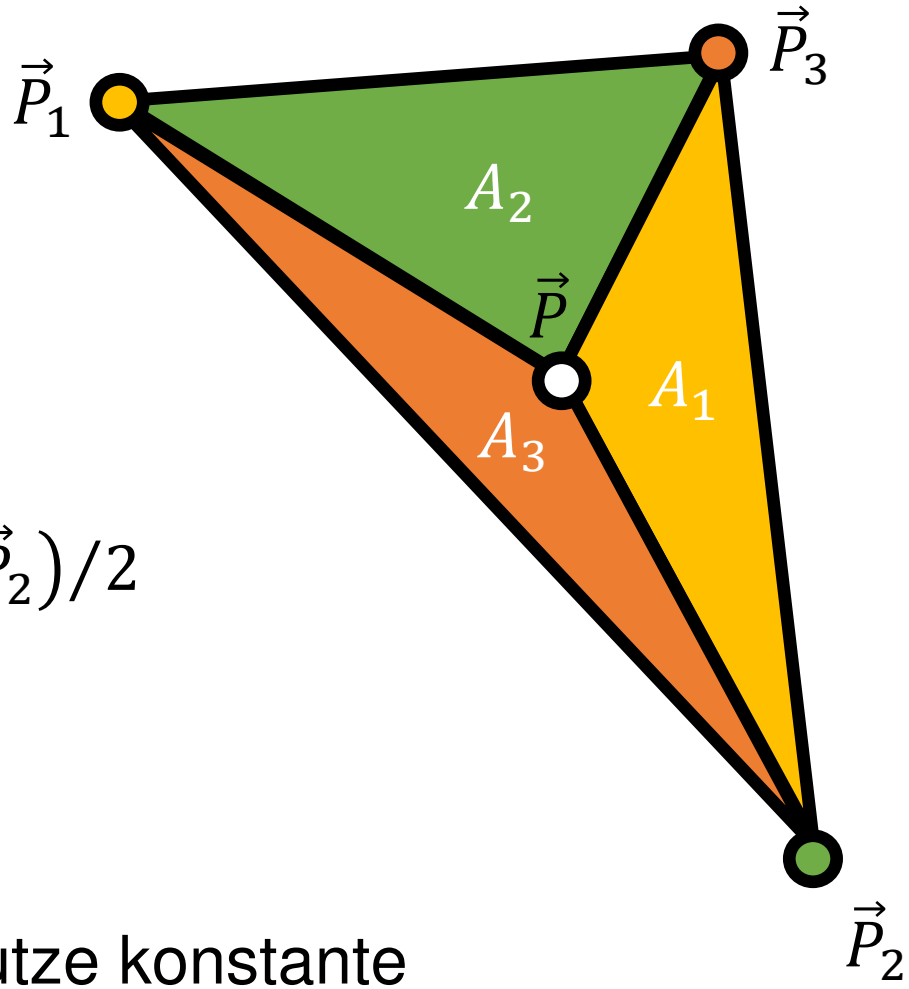
$$\lambda_i = {}^{A_i}/_{A_1 + A_2 + A_3}$$

$$\Rightarrow \vec{P} = \sum_i \lambda_i \vec{P}_i$$

- Wenn $0 \leq \lambda_i \leq 1$, ist $\vec{P}$ eine Konvexkombination von $\vec{P}_i$ und liegt daher im Dreieck

Schnelle baryzentrische Koordinaten im "screen space"

Schreibe Fläche A_1 als Kantengleichung für “2D-Ebene” P_2P_3:



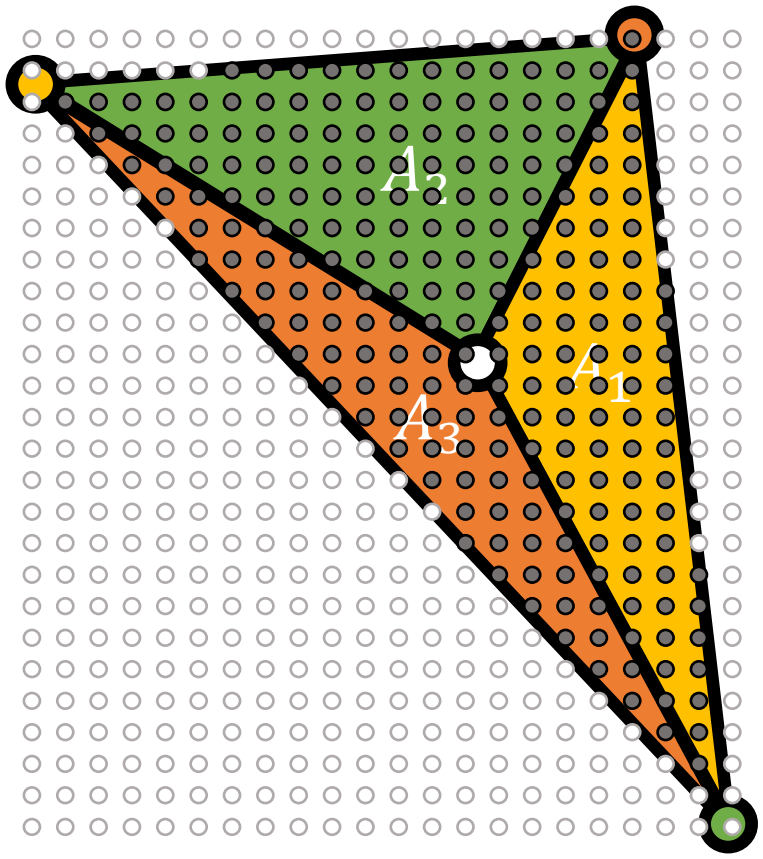
$$
\begin{aligned}
&A_1 \\
&= \left(\vec{P}_3 - \vec{P}_2\right) \times \left(\vec{P} - \vec{P}_2\right)/2 \\
&= \cdots \\
&= a_1 x + b_1 y + c_1
\end{aligned}
$$

Berechne $\{a, b, c\}_1$ vor und nutze konstante Inkremente übers Pixelgitter [Giesen 2011]

Rasterisierung von Dreiecken

- Für alle Pixel (x, y) in Bounding Box:
  - Berechne baryzentr. Koordinaten als $\lambda_{1,2} = a_{1,2}x + b_{1,2}y + c_{1,2}$
  - Wenn $\lambda_1 > 0, \lambda_2 > 0,\ \lambda_1 + \lambda_2 < 1$:
    - Zeichne Pixel

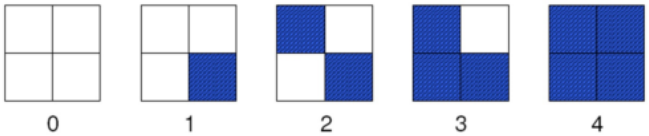
Beliebig parallelisierbar



Gegeben: Hochauflösendes Gerät (z.B. Drucker), das aber nur schwarz und weiß als Werte setzen kann

Ziel: Grautöne erzeugen

- Wir tauschen Intensitätsauflösung gegen Ortsauflösung
- Das Auge mittelt Intensitätsvariationen hoher Ortsauflösung
  - unterschiedlich große Punkte (Zeitung)
  - unterschiedlich gefüllte Muster gleich großer Punkte



- Allgemein stehen für ein quadratisches $(n \times n)$-Feld $n^2 + 1$ Graustufen zur Verfügung
- Dies wird durch eine $(n \times n)$-Matrix modelliert, in der die Reihenfolge der zu setzenden Pixel gespeichert werden, die sogenannte Dither-Matrix
- Um den gewünschten Grauton zu erzeugen, werden bei gegebener Intensität I alle Pixel (i, j) mit $D^n(i, j) < I$ gezeichnet

Beispiele für Dither-Matrizen:

$$D^2 = \begin{pmatrix} 1 & 3 \\ 2 & 0 \end{pmatrix} \qquad D^3 = \begin{pmatrix} 6 & 8 & 4 \\ 1 & 0 & 3 \\ 5 & 2 & 7 \end{pmatrix}$$

Dithering (Halbtonverfahren)

Wie werden die Dither-Matrizen konstruiert?

- vermeide horizontale bzw. vertikale Linien
- wachse von innen nach außen
- keine isolierten Punkte