



Computergraphik I

Scan Conversion of Lines

G. Zachmann
Clausthal University, Germany
zach@tu-clausthal.de



Das Zeichnen von Linien

- Der Begriff **Scan-conversion** oder **Rasterisierung** bezeichnet allgemein das algorithmische Bestimmen, welche Pixel von dem Primitiv überdeckt werden
 - Der Name kommt von der Scan-Technik der Rasterdisplays
- Algorithmus für Linien ist grundlegend für 2D und 3D Computergraphiken
- (Linien zeichnen war einfacher bei Vektor Displays ;-)

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 2

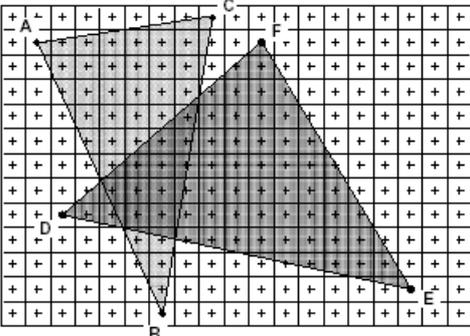


"La Linea"

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 3

Einordnung in die Pipeline

- Rasterisierung der Objekte in Pixel
- Ecken-Werte interpolieren (Farbe, Tiefenwert, ...)



Modell Transformation

Illumination (Shading)

Viewing Transformation (Perspective / Orthographic)

Clipping

Projektion (in Screen Space)

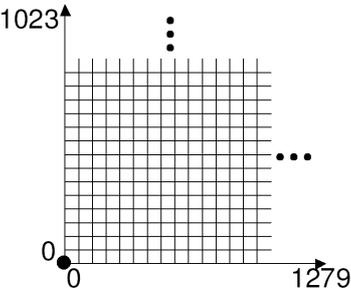
Scan Conversion (Rasterization)

Visibility / Display

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 4

Das Frame-Buffer-Modell

- Adressierung als 2D-Array

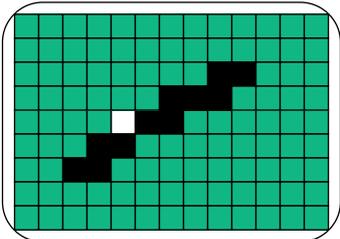


- Startadresse:
 - links oben (X11, Java AWT)
 - links unten (Open GL)

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 5

Was ist ein Pixel?

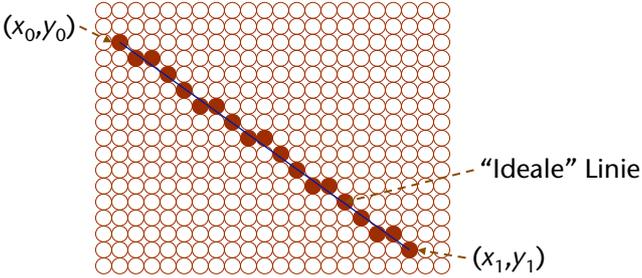
- Wir haben Ausgabegeräte mit endlicher, fester Auflösung und damit ein festes Raster
- In Wirklichkeit ist ein Pixel kein Punkt sondern ein kleines Quadrat
 - Daher der Name: "*pixel*" = "*picture cell*" oder "*picture element*"
- Und auch das ist nur eine Näherung
- Rasterisierung** bedeutet: ein graphisches Objekt, z.B. eine Linie, durch die Zellen eines Gitters annähern



G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 6

Die Ideale Linie

- Keine Unterbrechungen (diagonale Schritte sind erlaubt)
- Einheitliche Stärke und Helligkeit
- Fehlerfreiheit (setze nur den nächsten Pixel an der idealen Linie)
- Geschwindigkeit (wie schnell kann die Linie gezeichnet werden?)
- Invarianz gegenüber Zeichenrichtung



(x_0, y_0)

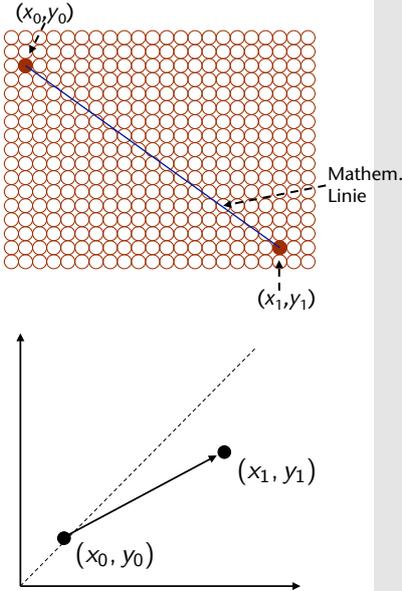
“Ideale” Linie

(x_1, y_1)

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 7

Die Aufgabe

- Gegeben: Endpunktkoordinaten einer Linie
- Vereinfachungen:
 - Ganzzahlige Koordinaten
 - Geradengleichung
 $y = mx + b$
 - Mit $0 \leq m \leq 1$
und $x_0 < x_1$
 - Alle übrigen Fälle bekommt man durch Vertauschen / Spiegeln
- Ausgabe: Folge von Pixeln (= rasterisierte Linie)



(x_0, y_0)

Mathem. Linie

(x_1, y_1)

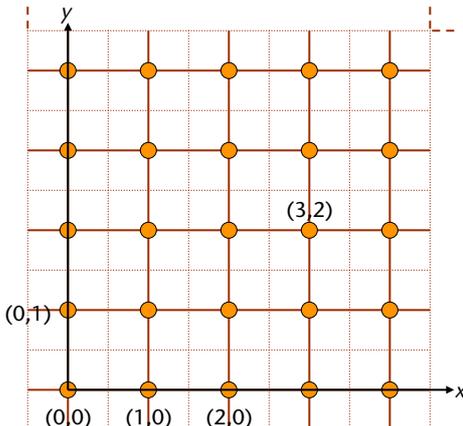
(x_0, y_0)

(x_1, y_1)

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 8

Bildschirmkoordinaten im Frame-Buffer-Modell

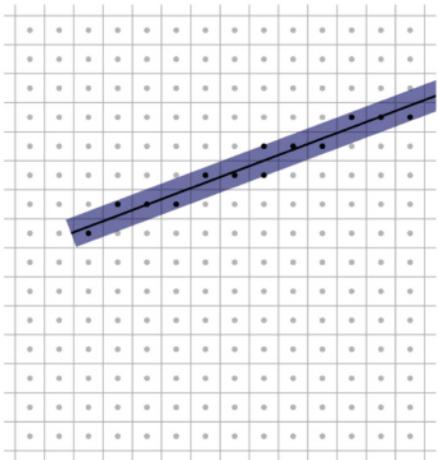
- Wir verwenden folgende 2D Bildschirmkoordinaten
 - Ganzzahlige Werte für Mittelpunkte der Pixel
 - Senkrecht = Y-Achse, Horizontal = X-Achse



G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 9

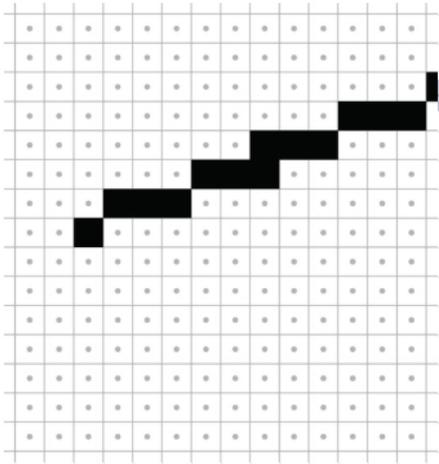
Erster Versuch der Scan-Konv. einer Linie

- Betrachte Linie als schmales Rechteck
- Zeichne alle Pixel, deren Zentrum im Inneren liegt



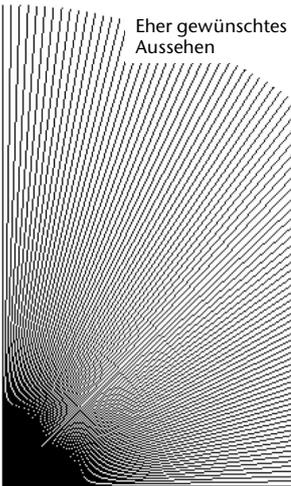
G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 10

- Betrachte Linie als schmales Rechteck
- Zeichne alle Pixel, deren Zentrum im Inneren liegt
- Problem: manchmal werden vertikal übereinander liegende Pixel gesetzt → unterschiedliche scheinbare Liniestärke

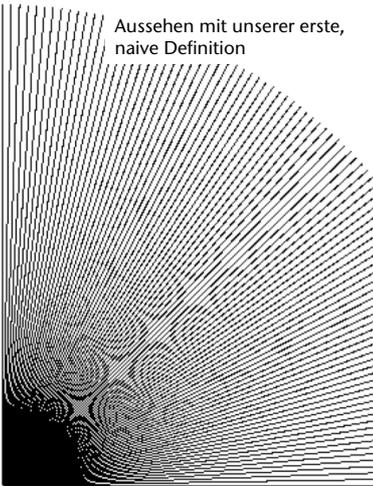


G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 11

- Problem:



Eher gewünschtes Aussehen



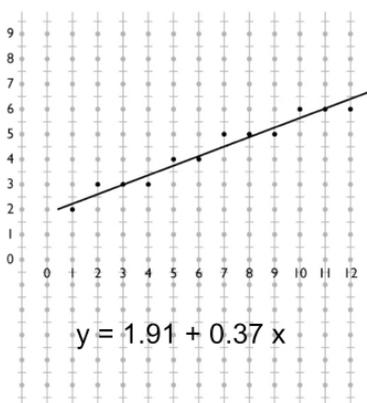
Aussehen mit unserer erste, naive Definition

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 12

Erster einfacher Algorithmus

- Einfacher Algorithmus: werte Gleichung der Linie pro Spalte (pro x-Koord.) ein Mal aus

```
for x = ceil(x0) .. floor(x1):
  y = b + m*x
  setPixel( x, round(y) )
```



- Problem: Floating-Point, Mult. und Runden sind (rel.) langsam

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 13

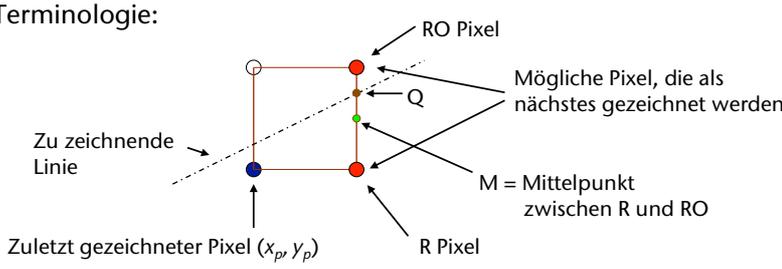
Enter Bresenham



Clip from Bresenham's Keynote at WSCG'03

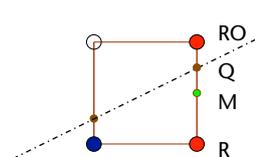
G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 14

Generelles Vorgehen und Terminologie

- Bei jedem X-Schritt gibt es nur zwei Möglichkeiten für die Y-Koordinate:
 - entweder bleibt die Y-Koord gleich;
 - oder die Y-Koord erhöht sich um genau 1 Pixel
- Terminologie:
 
- Q = Schnittpunkt der Linie mit Gridline $x_p + 1$

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 15

Zwei Varianten des Linien-Zeichens-Algorithmus

- Ursprünglich "**Bresenham-Algorithmus**" [1962]:
 - Bestimme Distanz zwischen RO und Q und zwischen R und Q
 - Bestimme Differenz zwischen den Distanzen
 - Vorzeichen des Ergebnisses legt fest, welcher Pixel eingefärbt wird
- Heute **Midpoint-Algo**:
 - Bestimme, auf welcher Seite der Linie liegt Mittelpunkt M
 - M = oberhalb → färbe Pixel R
 - M = unterhalb → färbe Pixel RO

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 16

Der Midpoint-Algorithmus

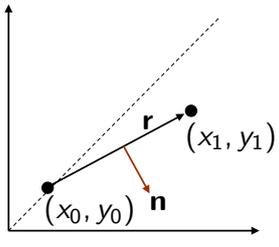
- Wie kann man einfach bestimmen, auf welcher Seite der Linie man sich befindet?
- Verwende implizite Form der Linie:

$$F(x, y) := \mathbf{n} \cdot \begin{pmatrix} x \\ y \end{pmatrix} - c = 0$$

$$\mathbf{r} = \begin{pmatrix} x_1 - x_0 \\ y_1 - y_0 \end{pmatrix} = \text{Richtungsvektor}$$

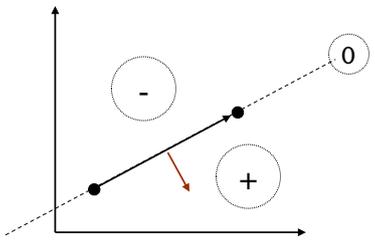
$$\mathbf{n} = \begin{pmatrix} n_1 \\ n_2 \end{pmatrix} = \begin{pmatrix} y_1 - y_0 \\ x_0 - x_1 \end{pmatrix} \text{ ist senkrecht dazu}$$

$$F(x_0, y_0) = 0 \text{ liefert } c = x_0 y_1 - y_0 x_1$$



G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 17

- Gegeben sei (x, y) . Dann ist
 - $F(x, y) = 0$, wenn (x, y) auf der Linie liegt
 - $F(x, y) < 0$, wenn (x, y) oberhalb der Linie liegt
 - $F(x, y) > 0$, wenn (x, y) unterhalb der Linie liegt



G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 18

- Für den Midpoint-Algorithmus, beachte das Vorzeichen von

$$F(M) = F(x_p + 1, y_p + \frac{1}{2})$$
- Definiere „Entscheidungsvariable“ d :

$$d = F(x_p + 1, y_p + \frac{1}{2})$$
- Wenn $d > 0$, färbe RO
- Wenn $d < 0$, färbe R
- Was ist mit dem nächsten Schritt?
- Annahme:
wir haben $d = F(M)$

x_p x_{p+1} x_{p+2}

y_p $y_p + \frac{1}{2}$ $y_p + 1$

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 19

1. Fall: R wurde ausgewählt \rightarrow nächstes M ist M_1

$$d_{old} = F(M)$$

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

$$= n_1(x_p + 1) + n_2(y_p + \frac{1}{2}) + c$$

und

$$d_{new} = F(M_1)$$

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

$$= n_1(x_p + 2) + n_2(y_p + \frac{1}{2}) + c$$

somit

$$d_{new} = d_{old} + n_1$$

x_p x_{p+1} x_{p+2}

y_p $y_p + \frac{1}{2}$ $y_p + 1$

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 20

2. Fall: RO wurde ausgewählt → nächstes M ist M_2

$$d_{old} = F(M)$$

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

$$= n_1(x_p + 1) + n_2(y_p + \frac{1}{2}) + c$$

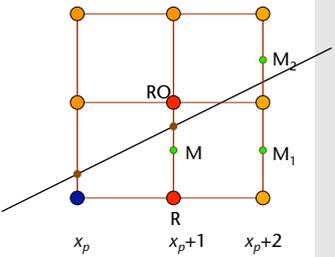
und

$$d_{new} = F(M_2)$$

$$= F(x_p + 2, y_p + \frac{3}{2})$$

$$= n_1(x_p + 2) + n_2(y_p + \frac{3}{2}) + c$$

somit

$$d_{new} = d_{old} + n_1 + n_2$$


G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 21

▪ Pseudo-Code des Midpoint-Algo:

```

berechne n1, n2, c
x, y ← x0, y0
d ← F(M) = F(x0 + 1, y0 + 1/2) = n1 + n2/2
setze d1 ← n1, d2 ← n1 + n2
while x ≤ x1:
  zeichne Pixel (x,y)
  x += 1
  if d > 0:
    y += 1
    d += d2
  else:
    d += d1

```

▪ Achtung: in obigen Pseudo-Code ist evtl. $d = \frac{k}{2}$

▪ Lösung: ...

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 22

Beispiel

- Zeichne Linie von (1,2) nach (5,5)

```

x, y ← x0, y0
d ← F(M) = F(x0 + 1, y0 + 1/2) = n1 + n2/2
setze d1 ← n1, d2 ← n1 + n2 while
x ≤ x1:
    zeichne Pixel (x,y)
    x += 1
    if d < 0:
        y += 1
        d += d2
    else:
        d += d1
        
```

n ₁	n ₂	d ₁	d ₂	d	x	y

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 23

Demo

Algorithm:

Midpoint Line Midpoint Circle Antialiased Line Antialiased Circle

```

WritePixel(x, y);
while (x < p2x) {
    if (d <= 0) {
        d += incMajor;
        x++;
    } else {
        d += incMinor;
        x++;
        y++;
    }
    WritePixel(x, y);
}
}

if the slope is (0, -1) {
    calculate decision variable d;
    calculate major axis (+X) increment 'incMajor';
    calculate minor axis increment 'incMinor';
        
```

Stepping Breakpoint Set Clear

p1x: -145 p1y: -113 p2x: 26 p2y: 21

dx: 171 dy: 134 d: 23

incMajor: 268 incMinor: -74 x: -144 y: -113

Pixel Size: 10X

Stop Run Loop Jog Step Rest

<http://www.cs.rit.edu/~ncs/whatsInALine/whatsInALine.html>

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 24

Historische Randnotiz

- Bekannt als DDA (digital differential analyzer)



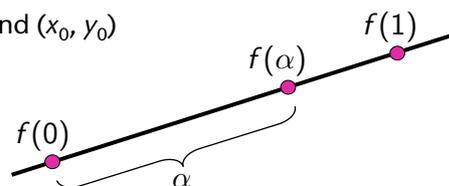
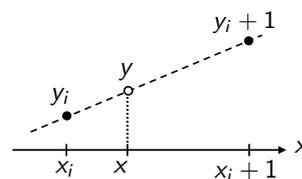
MADDIDA (Magnetic Drum Digital Differential Analyzer, Northrop Aircraft) 1952

Interpolation von Attributen

- Häufig haben Eckpunkte weitere Attribute (außer der Pos.)
 - Z.B. verschiedene Farben
- Ziel: ein gleichmäßiger Farbverlauf entlang der Linie
- Idee: lineare Interpolation

- Im 1D: $f(x) = (1 - \alpha)y_0 + \alpha y_1$
mit $\alpha = (x - x_0)/(x_1 - x_0)$

- Im 2D ist α gerade die normierte(!)
Distanz zwischen (x, y) und (x_0, y_0)



- Pixel liegen i.A. *nicht* genau auf der Linie
- Definiere 2D Funktion zur Projektion auf die Linie:

$$\alpha = \frac{\mathbf{r} \cdot (\mathbf{Q} - \mathbf{P}_0)}{l}$$

$$l = \|\mathbf{P}_1 - \mathbf{P}_0\|$$

$$f(\alpha) = (1 - \alpha) \begin{pmatrix} r_0 \\ g_0 \\ b_0 \end{pmatrix} + \alpha \begin{pmatrix} r_1 \\ g_1 \\ b_1 \end{pmatrix}$$
- Beobachtung: f ist linear in Q_x und Q_y
- Verwende DDA zur inkrementellen Berechnung von f

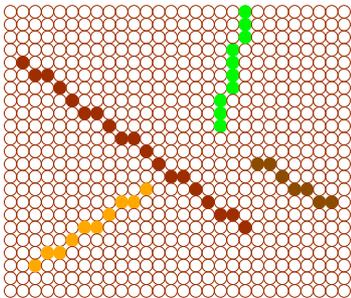
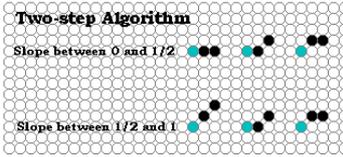
G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 27

- Resultat:

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 28

Geschwindigkeitssteigerung

- Sind rasterisierte Linien symmetrisch?
- Abhängig von der Länge:
 - Gerade # an Pixel → ja
 - Ungerade # an Pixel → ja, bis auf 1 Pixel
- Idee: zeichne von beiden Seiten [Rokne et al., 1990]
- Man kann 2 Pixel zeichnen mit:
 - 1 Vergleich
 - 1 Update der Entscheidungsvariable d
- Weiterhin: mit 1 Test kann man die nächsten 2 Pixel entscheiden [Wyvill et al., 1990]

Two-step Algorithm

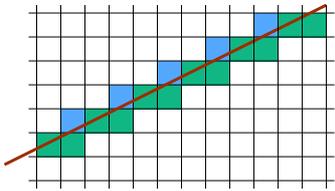
Slope between 0 and 1/2

Slope between 1/2 and 1

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 29

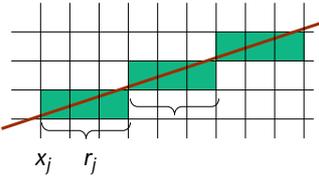
Das letzte(?) Quentchen Geschwindigkeit

- Vereinfachungen zunächst:
 1. Betrachte (unendliche) Linien mit $y = mx$, $0 \leq m \leq 1$
 2. Betrachte nur die Folge der grünen Zellen = Zellen, die an ihrer linken Kante von der Linie geschnitten werden
- Terminologie:
 - Zelle wird identifiziert durch deren linken unteren Eckpunkt (x_j, y_j)
 - **Span** := Folge von Zellen mit gleicher y-Koord.
 - Länge des j-ten Spans = r_j



G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 30

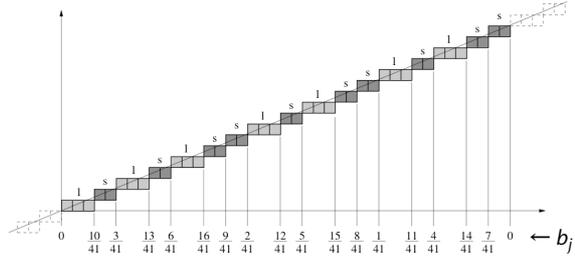
- Beobachtung: die diskrete Linie ist vollständig durch die Folge der Span-Längen definiert, denn

$$(x_{j+1}, y_{j+1}) = (x_j + r_j, y_j + 1)$$

- Satz (o. Bew.):
Alle Spans der diskretisierten Linie haben nur eine von höchstens zwei verschiedenen Längen, nämlich

$$\forall j: r_j = r \vee r_j = r + 1$$
- Klar ist:

$$\frac{1}{2} \leq m \leq 1 \Rightarrow r = 1$$

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 31

- Beispiel:
 
- Beobachtung: wenn wir ein seeehr langes Segment der Linie betrachten, dann gilt

$$\frac{\# \text{ Spans}}{\# \text{ Zellen}} = \frac{\Delta y}{\Delta x} \approx m$$

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 32

- Folge: aus der Steigung kann man die Span-Länge r (bzw. $r+1$) berechnen:

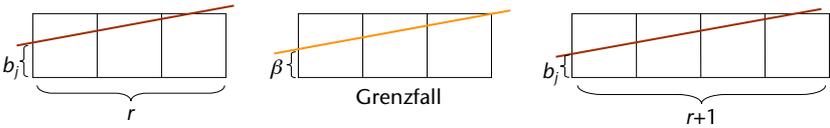
$$\frac{1}{m} = \text{mittlere Span-Länge}$$

$$= \text{Mittelwert von } r \text{ und } r+1 \Rightarrow$$

$$r = \left\lfloor \frac{1}{m} \right\rfloor, r+1 = \left\lceil \frac{1}{m} \right\rceil$$
- Im Folgenden: Berechnung von r_j , m.a.W., Methode zur Entscheidung, ob man einen "langen Span" oder einen "kurzen Span" hat

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 33

- Wovon hängt es ab, ob man einen langen / kurzen Span hat?



- Fazit: falls $b_j \geq \beta$, dann kurzer Span, sonst langer Span
- Bestimmung von β : $b_j = mx_j - y_j$

$$b_{j+1} - b_j = mr_j - 1$$

Im Grenzfall ist $b_{j+1} = 0$ und $b_j = \beta$, also

$$\beta = 1 - mr = 1 - m \left\lfloor \frac{1}{m} \right\rfloor$$

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 34

- Das nächste b_{j+1} ist also:
 - falls kurzer Span $\rightarrow b_{j+1} = b_j - \beta$
 - falls langer Span $\rightarrow b_{j+1} = b_j + m - \beta$
- Damit hat man einen iterativen, sehr effizienten Algo zur Aufzählung aller Zellen, die von einer Linie getroffen werden.
- Weiteres (lästiges) Detail:
 - Bei einem Strahl ist der erste Span i.A. gekürzt
 - Soll hier nicht weiter vertieft werden

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 35

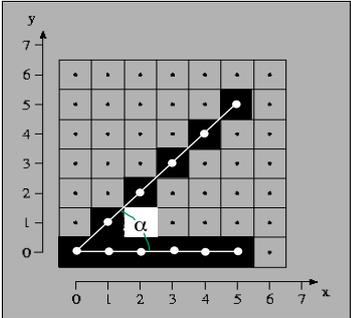
Speedup gegenüber einfachem DDA

- $O(n)$ bei DDA (z.B. Midpoint),
 $O(n/r)$ mit der Span-basierten Methode,
 $n = \text{Anzahl Zellen auf dem Strahl}, r = \text{mittlere Span-Länge}$
- In Zahlen:
 - Ca. Faktor 2 schneller über alle mögliche Orientierungen des Strahls

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 36

Weitere Überlegungen

- Einheitliche Stärke und Helligkeit
 - Bei gleicher Pixelzahl sind schräge Linien länger als horizontale
 - Ändere Intensität der Linie gemäß der Steigung
 - Skaliere den Grauwert um den Faktor $\cos(45^\circ - \alpha)$, $\alpha = 0^\circ \dots 45^\circ$

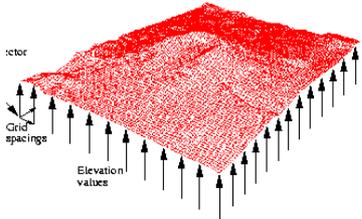
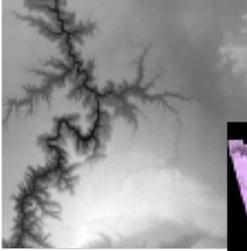


- Was ist bei gemusterten Linien? (gestrichelte Linie, etc.)

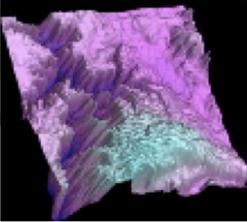
G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 37

Beispiel für eine Anwendung der DDA-Technik: Ray-Tracing von Height Fields [Henning & Stephenson, 2004]

- Height Field = Alle Arten von Flächen, die sich als Funktion $z = f(x, y)$ schreiben lassen
 - Z.B.: Terrain, Meßwerte über einer Ebene, 2D-Skalarfeld, ...

Height field (= Bitmap)



Rendered

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 38

Situation

- Die naive Methode, ein Height-Field zu raytracen:
 - Konvertiere das $n \times n$ Feld in $2n^2$ Dreiecke, teste Strahl gegen jedes
 - Probleme: langsam, benötigt viel Speicher
- Ziel: direktes Ray-Tracing des Height-Fields aus dem 2D-Array
- Gegeben:
 - Strahl
 - Feld $[0..n] \times [0..n]$ als Float-Array
 - Höhenwerte liegen auf den Gitterknoten vor

$z = f(x_j, y_j)$

(x_j, y_j)

$(0,0)$

x

y

Scan Conversion: Lines & Co 39

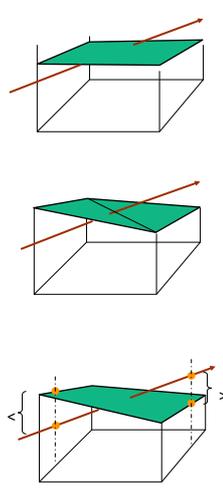
Algorithmus

1. Dimensionsreduktion
 - Projiziere Strahl in xy-Ebene
2. Alle Zellen der Reihe nach besuchen, die vom Strahl geschnitten werden (und nur diese)
 - Ähnlich zu Scan-Conversion, aber mit zusätzlichen Zellen
3. Strahl testen gegen das Flächenstück, das von den 4 Höhenwerten an den Ecken aufgespannt wird

Scan Conversion: Lines & Co 40

Schnitttest Strahl—Flächenstück in der Zelle

- Naive Methoden:
 - "Nearest-Neighbor":
 - Bestimme mittlere Höhe aus den 4 Höhenwerten an den Ecken
 - Schneide Strahl gegen horizontales Quadrat mit dieser mittleren Höhe
 - Sehr ungenau
 - "2 Dreiecke":
 - Konstruiere 2 Dreiecke aus den 4 Punkten über den Ecken
 - Knick innerhalb der Zelle, Aufteilung in Dreiecke nicht eindeutig
- Besser: "bilineare Interpolation"
 - Betrachte Fläche als parabolisches Hyperboloid
 - Bestimme Höhe am Rand über/unter dem Strahl durch lineare Interpolation
 - Vergleiche Vorzeichen
 - Bestimmt ggf. Schnittpunkt & Normale



G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 41

Beispiele für Terrain

Turtmann Valley Dataset

- **3 datasets** of **4k x 4k** height-samples each
@ 2m planar, 0.25m vertical inter-pixel spacing
- Normal-maps derived from input height-map
(**3x4096x4096**), mixed **JPEG** and **S3TC** compression
- Compressed dataset size: **33 MB**
- Flight speed is around 540 km/h ~ **Mach 0,5**

Bonn University

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 42



Valles Marineris, Mars - <http://mars.jpl.nasa.gov>

G. Zachmann Computer-Graphik 1 – WS 11/12 Scan Conversion: Lines & Co 43