

Graphics Output Primitives

Hearn/Baker
3.1, 3.5, 3.14, 3.15, 3.20



A scene from the wolfman video. The animated figure of this primitive lycanthrope is modeled with 61 bones and eight layers of fur. Each frame of the computer animation contains 100,000 surface polygons. (Courtesy of the NVIDIA Corporation.)

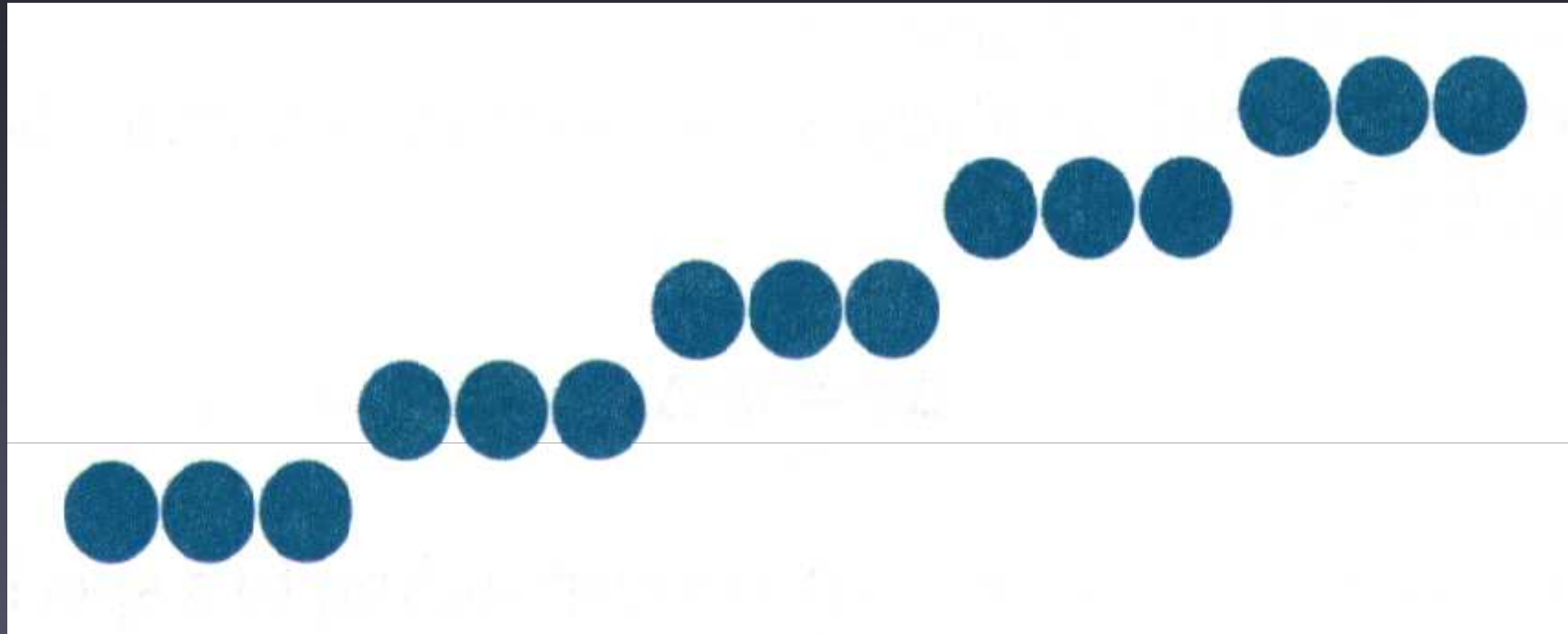
Grundlegende grafische Output-Primitive

- Punkte, Linien und parallele Linien
- Frame-Buffer Werte bestimmen
- Kreisbildung
- Ellipsenbildung
- andere Bögen
- Füllen von Flächen
- Pixel Arrays
- Characters
- ...

Punkte und Linien

- Rasterkonversion eines Punktes
 - ◆ Anordnung in Befehlsliste (Random Scan)
 - ◆ Eintrag im Bildschirmspeicher (Raster Scan)
- Rasterkonversion einer Linie
 - ◆ Anordnung in Befehlsliste (Random Scan)
 - ◆ Berechnet dazwischen liegende diskrete Pixel-Positionen (Raster Scan)
 - Treppeneffekte (“jaggies”), Aliasing

Linien: Treppenstufen Effekt



Treppenstufen-Effekt (jaggies) entsteht bei der Generierung einer Linie aus einer Folge von Pixel-Positionen

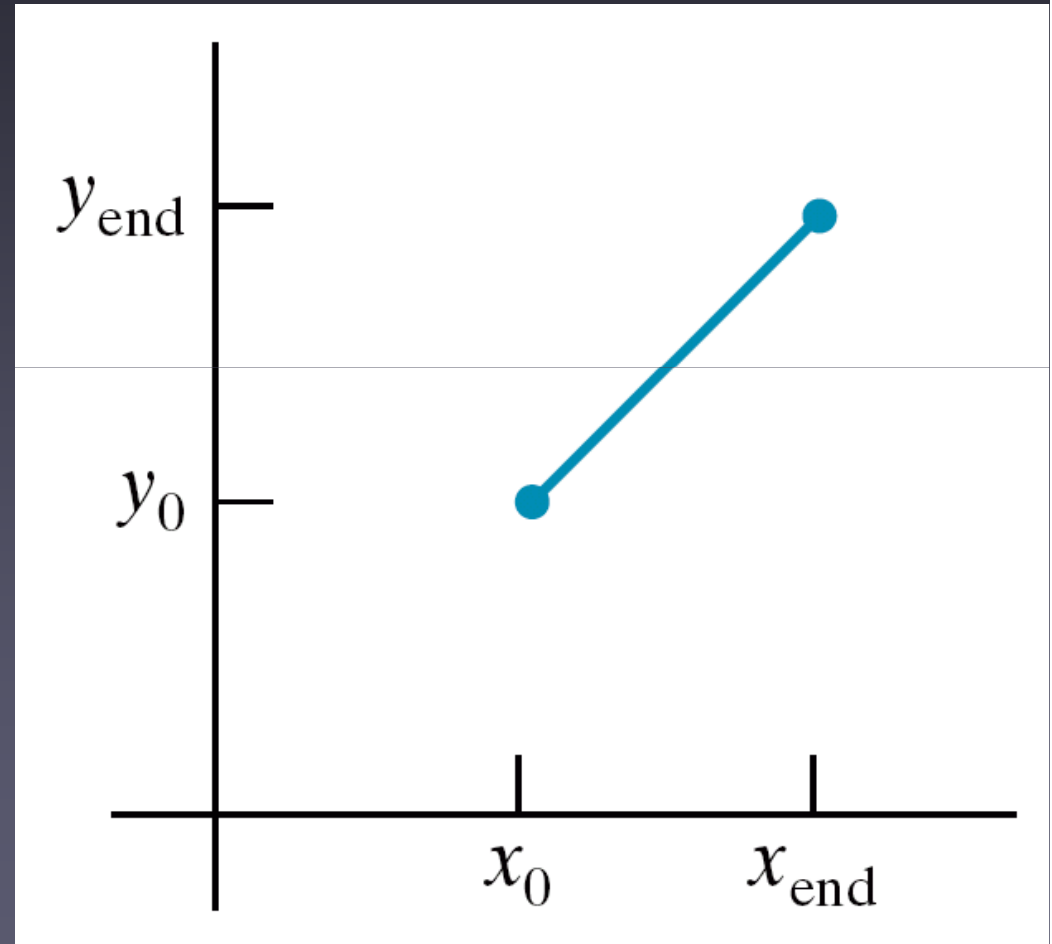
Linienalgorithmus

Geradengleichung: $y = m \cdot x + b$

Linie zwischen 2
Punkten:

$$m = \frac{y_{\text{end}} - y_0}{x_{\text{end}} - x_0}$$

$$b = y_0 - m \cdot x_0$$

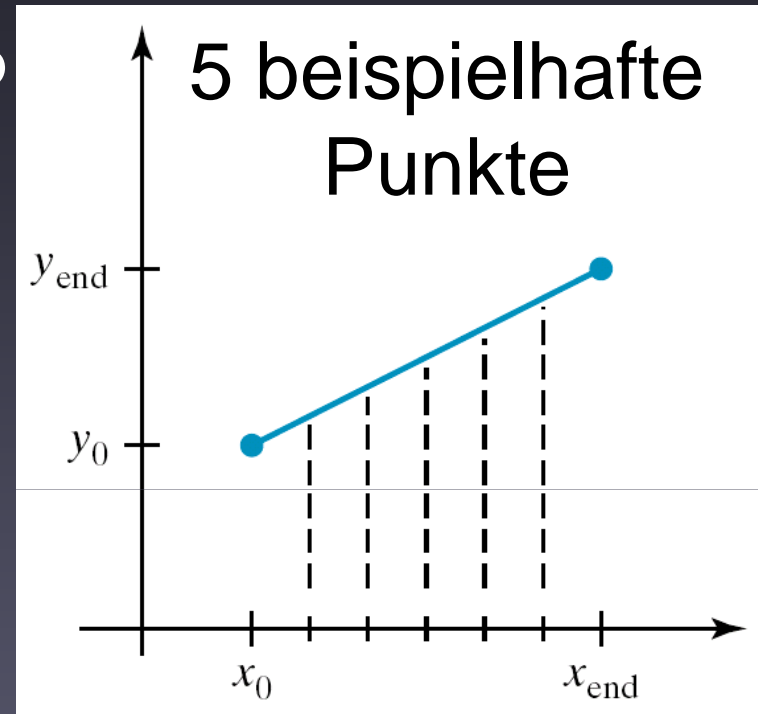


DDA Linienalgorithmus

Geradengleichung: $y = m \cdot x + b$

$$\delta y = m \cdot \delta x \quad \text{für } |m| < 1$$

$$\delta x = \frac{\delta y}{m} \quad \text{für } |m| > 1$$



■ DDA (Digital Differential Analyzer)

$$\text{für } \delta x = 1, \quad |m| < 1 : y_{k+1} = y_k + m$$

DDA – Grundprinzip des Algorithmus

```
dx = xEnd - x0; dy = yEnd - y0;
```

```
m = dy / dx;
```

```
x = x0; y = y0;
```

```
setPixel (round(x), round(y));
```

```
for (k = 0; k < dx; k++)
```

```
{ x += 1; y += m;
```

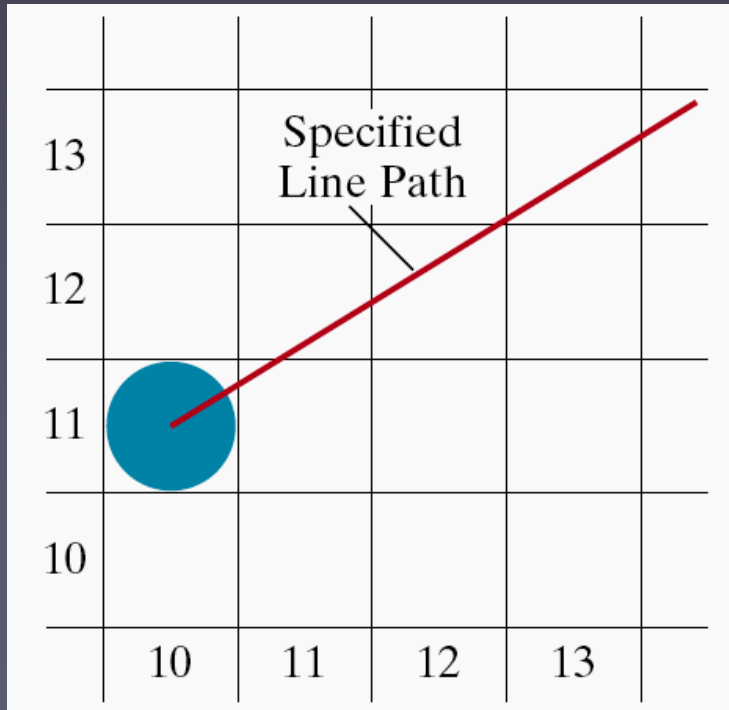
```
setPixel (round(x), round(y)}
```

Einfaches Beispiel - ausbaufähig

Bresenham's Linienalgorithmus

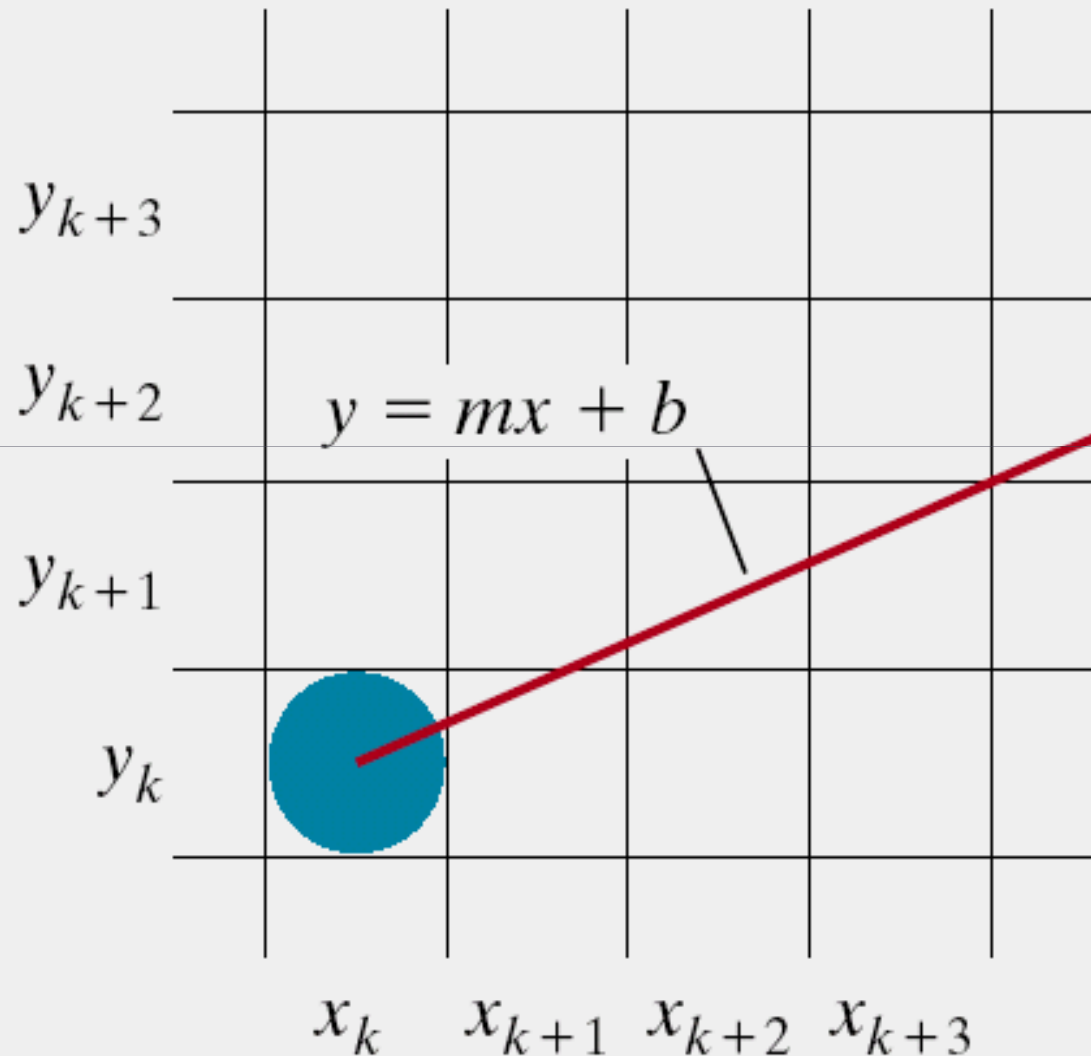
- Schneller als einfacher DDA
 - ◆ Inkrementelle Zahlenberechnung
 - ◆ Anpassungsfähig an Kreise und andere Kurven

$$y = m \cdot (x_k + 1) + b$$



Ausschnitt einer
Bildschirmanzeige, wo eine
gerade Linie gezeichnet
wird, beginnend an der
Pixelposition in Spalte 10
auf Scanlinie 11

Bresenham's Linienalgorithmus



Ausschnitt des Pixelrasters zeigt ein Pixel in Spalte x_k auf der Scanlinie y_k , welche entlang des Pfades des Liniensegmentes gezeichnet wird, mit einer Steigung $0 < m < 1$

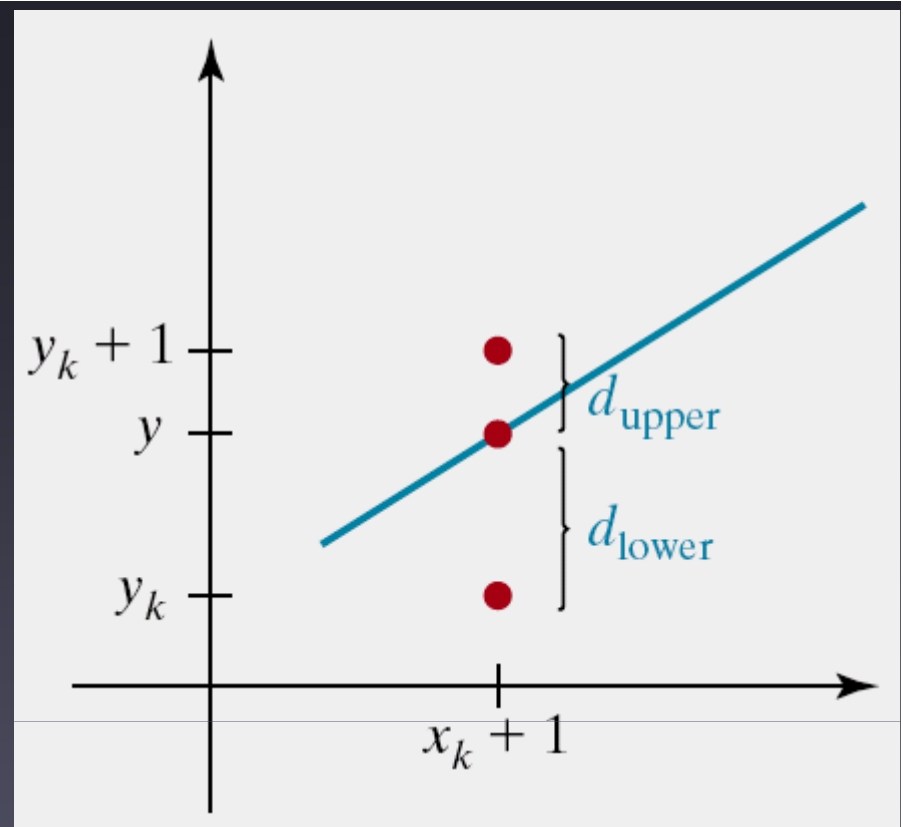
Bresenham's Linienalgorithmus (1/4)

$$y = m \cdot (x_k + 1) + b$$

$$\begin{aligned} d_{\text{lower}} &= y - y_k = \\ &= m \cdot (x_k + 1) + b - y_k \end{aligned}$$

$$\begin{aligned} d_{\text{upper}} &= (y_k + 1) - y = \\ &= y_k + 1 - m \cdot (x_k + 1) - b \end{aligned}$$

$$d_{\text{lower}} - d_{\text{upper}} = 2m \cdot (x_k + 1) - 2y_k + 2b - 1$$



Bresenham's Line Algorithm (2/4)

$$d_{\text{lower}} - d_{\text{upper}} =$$
$$= 2m \cdot (x_k + 1) - 2y_k + 2b - 1$$

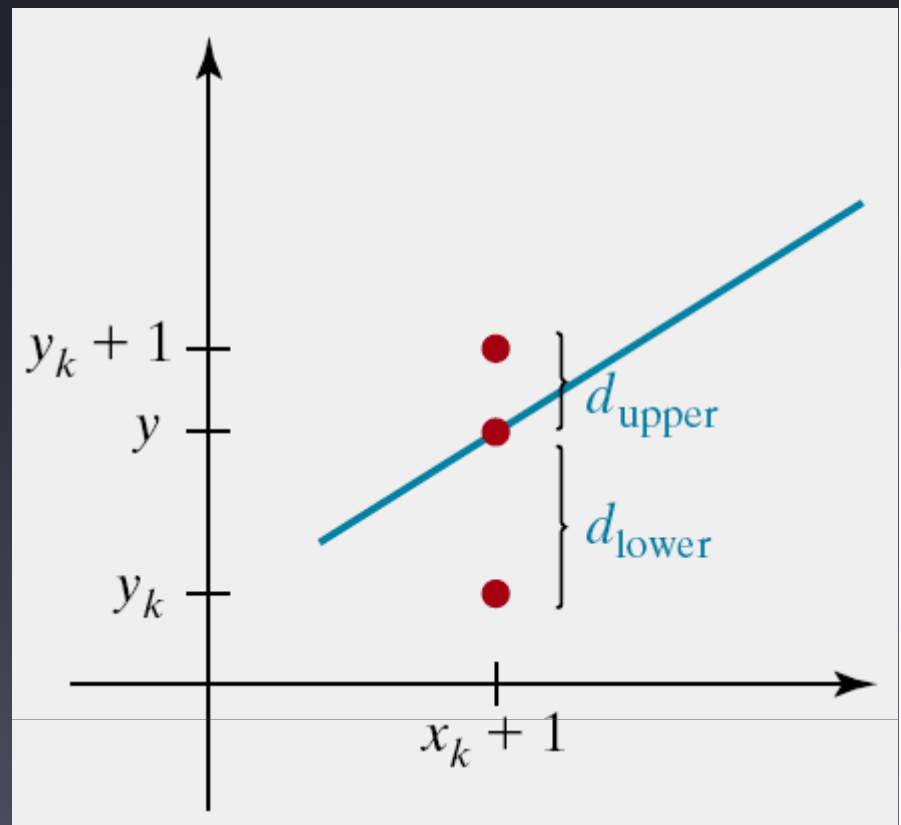
$$m = \Delta y / \Delta x$$

$\Delta x, \Delta y$: Trennung der Endpunktpositionen

**Entscheidungs-
Parameter**

$$p_k = \Delta x \cdot (d_{\text{lower}} - d_{\text{upper}}) =$$
$$= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c$$

hat gleiches Vorzeichen wie $(d_{\text{lower}} - d_{\text{upper}})$



Bresenham's Linienalgorithmus (3/4)

Aktuelle Entscheidungswerte:

$$p_k = \Delta x \cdot (d_{\text{lower}} - d_{\text{upper}}) = 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c$$

Nächste Entscheidungswerte:

$$p_{k+1} = 2\Delta y \cdot x_{k+1} - 2\Delta x \cdot y_{k+1} + c + 0$$

$$+ p_k - 2\Delta y \cdot x_k + 2\Delta x \cdot y_k - c =$$

Beginnend bei Entscheidungswert:
 $= p_k + \Delta y = \Delta x \cdot (y_{k+1} - y_k)$

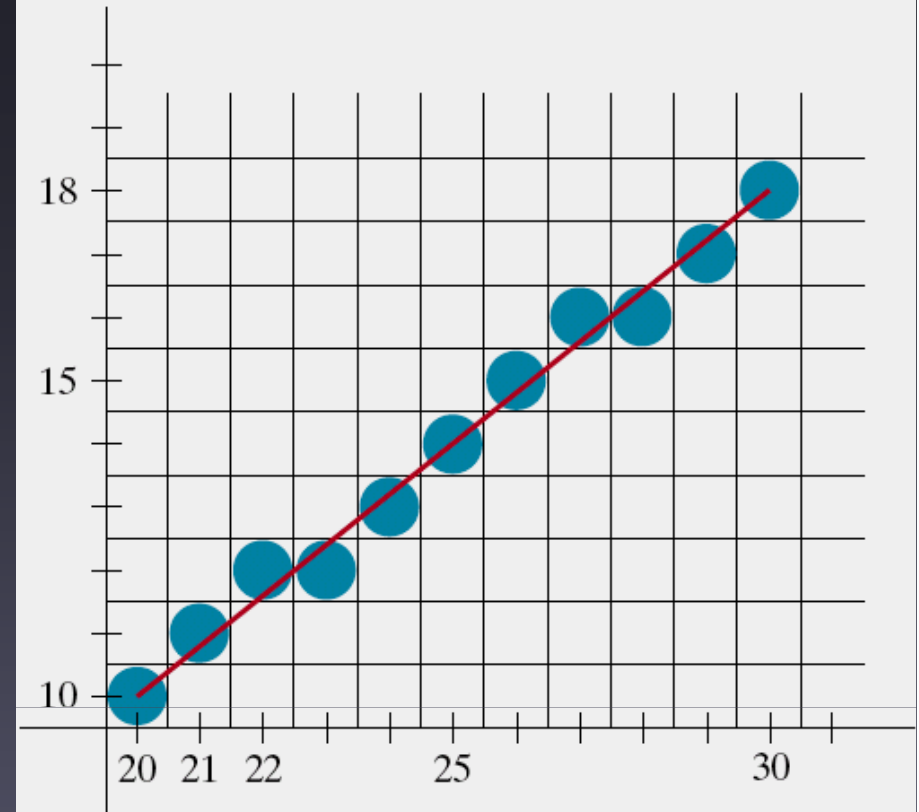
$$p_0 = 2\Delta y - \Delta x$$

Bresenham's Linienalgorithmus (4/4)

1. Speichere linken Endpunkt in (x_0, y_0)
2. Zeichne Pixel (x_0, y_0)
3. Berechne Konstanten Δx , Δy , $2\Delta y$, $2\Delta y - 2\Delta x$ und erhalte
$$p_0 = 2\Delta y - \Delta x$$
4. An jedem x_k entlang der Linie teste:
 - if $p_k < 0$
 - then zeichne pixel (x_{k+1}, y_k) ; $p_{k+1} = p_k + 2\Delta y$
 - else zeichne pixel (x_{k+1}, y_{k+1}) ; $p_{k+1} = p_k + 2\Delta y - 2\Delta x$
5. Führe diesen Schritt $(4 \Delta x - 1)$ mal aus

Bresenham: Beispiel

Linie von (20/10)
bis (30/18)



k	p_k	(x_{k+1}, y_{k+1})	k	p_k	(x_{k+1}, y_{k+1})
0	6	(21, 11)	5	6	(26, 15)
1	2	(22, 12)	6	2	(27, 16)
2	-2	(23, 12)	7	-2	(28, 16)
3	14	(24, 13)	8	14	(29, 17)
4	10	(25, 14)	9	10	(30, 18)

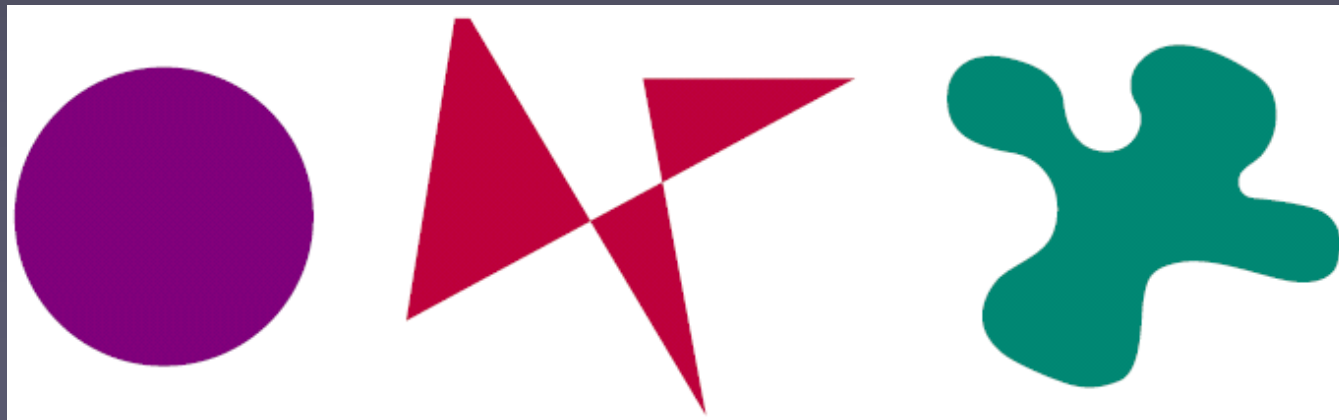
Grundlegende grafische Output-Primitive

- Punkte, Linien und parallele Linien
- Frame-Buffer Werte bestimmen
- Kreisbildung
- Ellipsenbildung
- andere Bögen
- **Füllen von Flächen**
- Pixel Arrays
- Characters
- ...



Primitive zum Füllen von Flächen

- Für Polygonflächen (solide Farbe, gemustert)
 - ◆ **Scan-Line** Polygon Füllalgorithmus
 - Schnittpunkte lokalisieren und sortieren
 - Aufeinanderfolgende Paare definieren den inneren Abstand
 - Vorsicht bei Eckschnittpunkten
 - Ausnutzen der Kohärenz (inkrementelle Berechnung)
 - ◆ **Flood Fill** Algorithmus

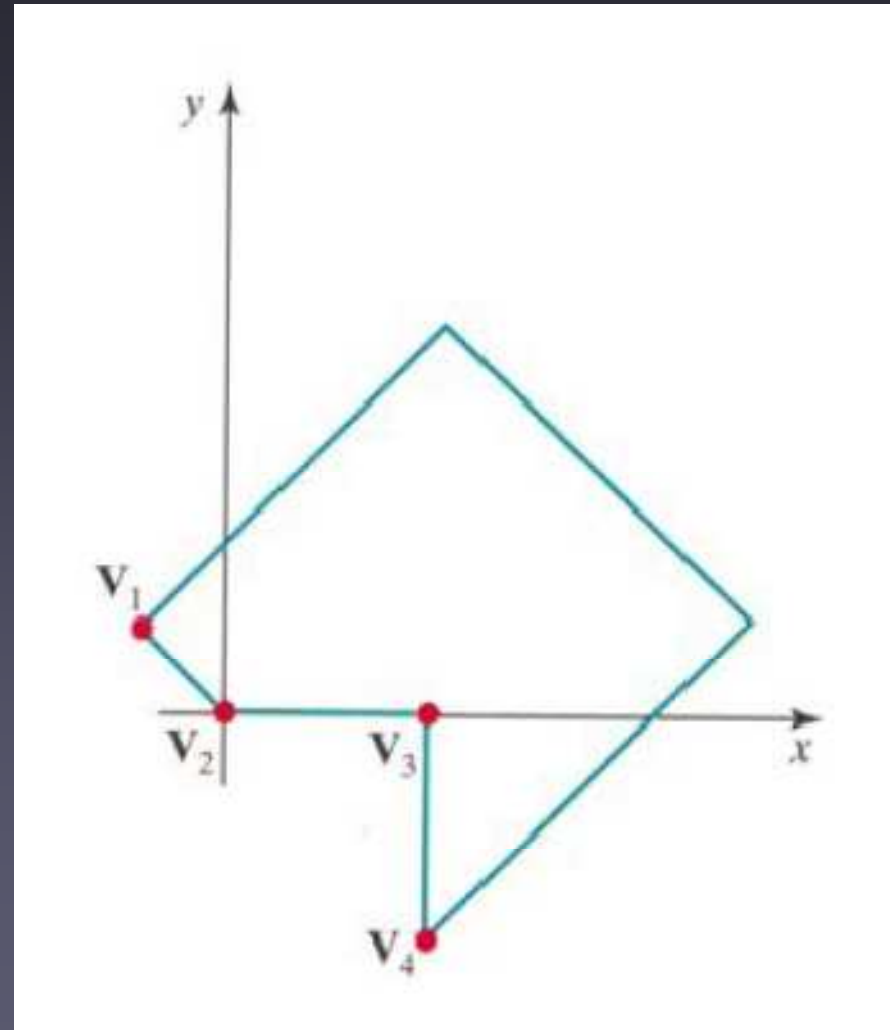


Füllflächen von Polygonen

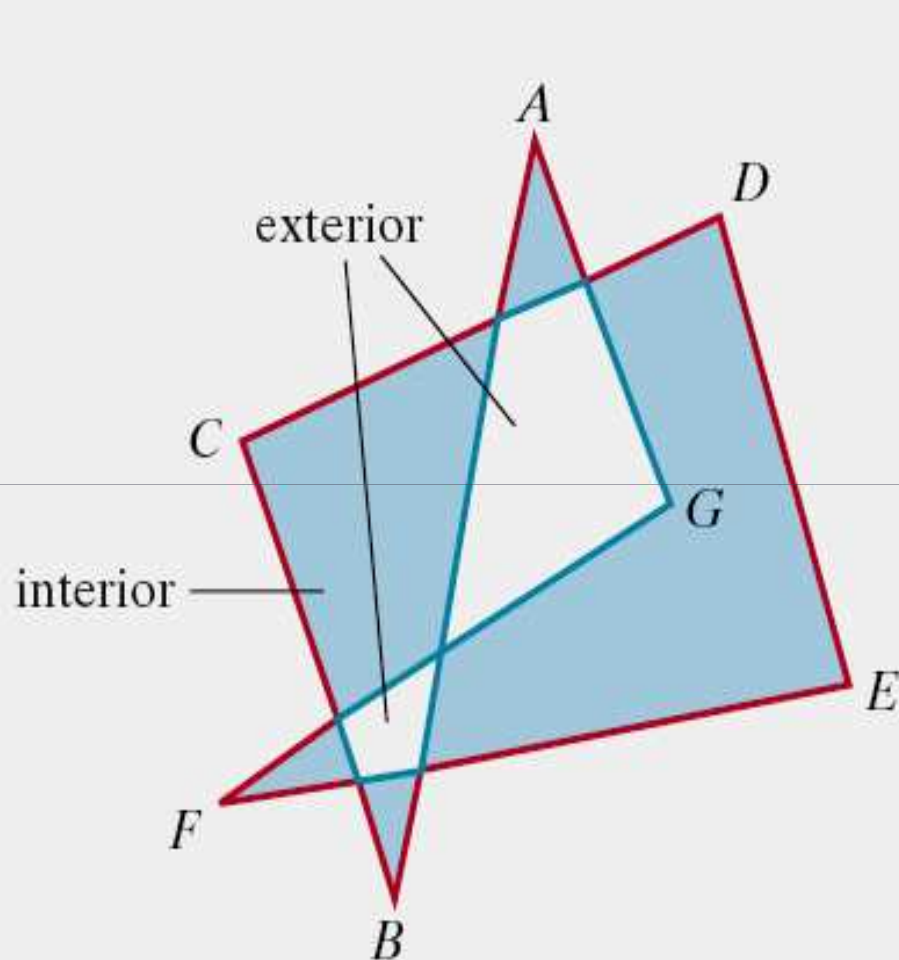
- Polygonklassifizierungen:
 - ◆ **konvex**: kein Innenwinkel $> 180^\circ$
 - ◆ **konkav**: nicht konvex
- Splitten von konkaven Polygonen
 - ◆ *Vektormethode*:
 - Alle Kreuzprodukte der Vektoren haben dasselbe Vorzeichen \rightarrow konvex
 - ◆ *Rotationsmethode*
 - Rotiere Polygonkanten um die x-Achse, immer dieselbe Richtung \rightarrow konvex

Rotationsmethode

- Für jeden Eckpunkt $v(k)$
 - ◆ Versetze $v(k)$
→ Ursprung
 - ◆ Rotiere im
Uhrzeigersinn
→ $v(k+1)$ auf x-Achse
 - ◆ Wenn $v(k+2)$ unter x-Achse dann konkav
 - ◆ Teile entlang x-Achse

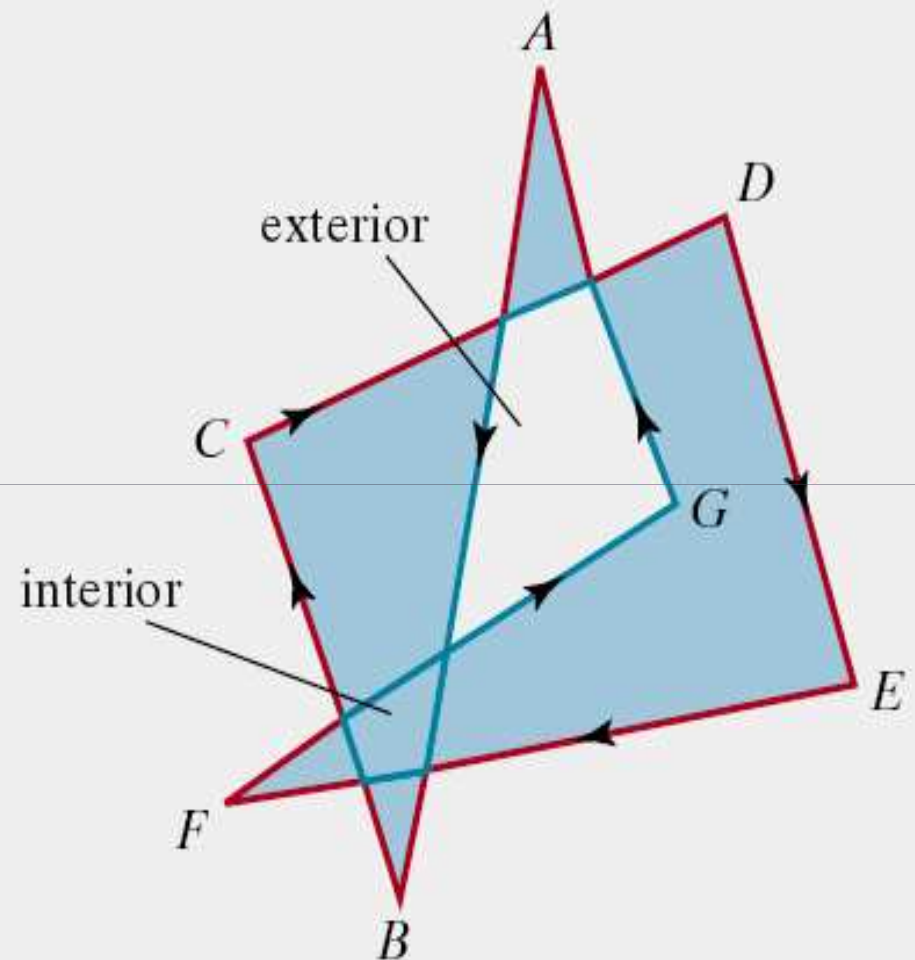


Innen-Aussen Tests: Vergleich



Imparitätsregel

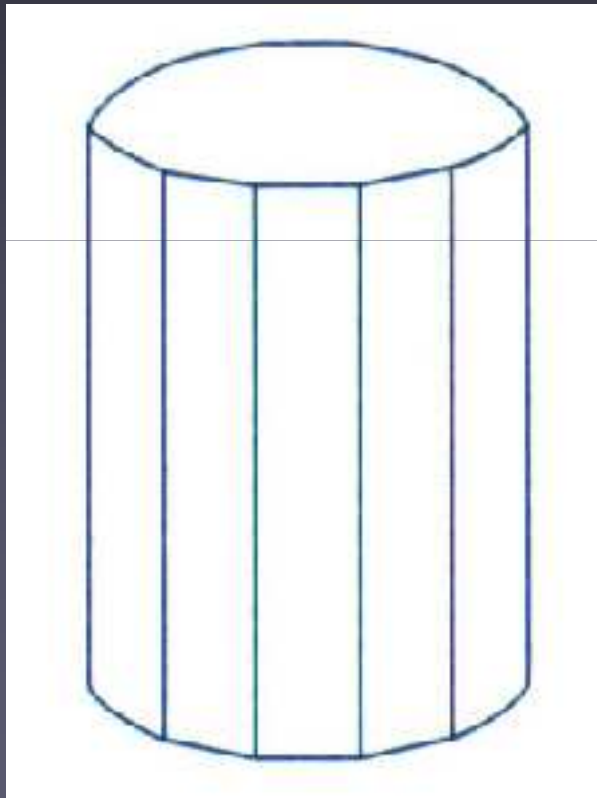
Graphics Output Primitives



“Nullwindungs”-
Regel

Polygonflächen (1)

- Bestimmung der Polygonfläche, welche das innere Objekt umhüllt



= Boundary Representation
("B-rep")

Gitterartige Darstellung
eines Zylinders, hintere
versteckte Linien entfernt

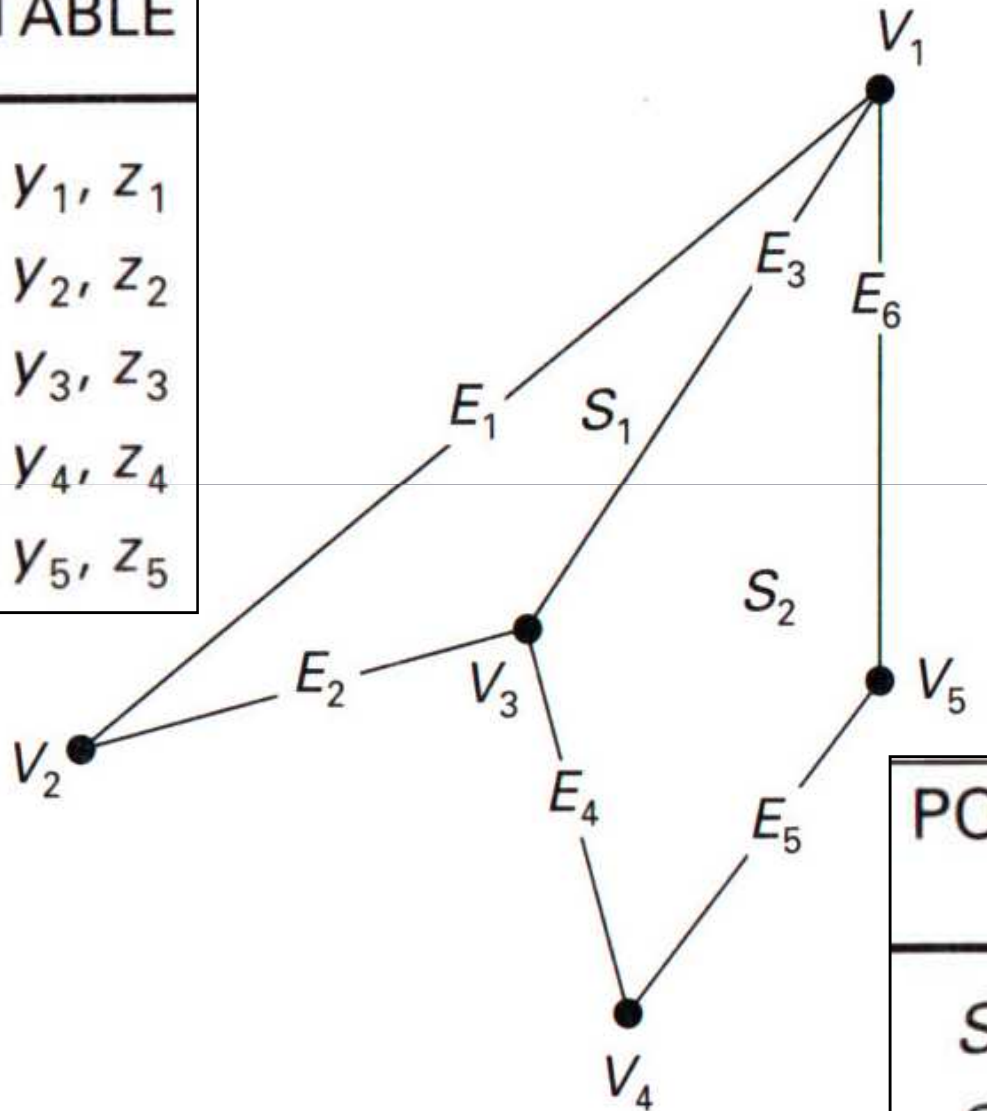
Polygonflächen (2)

- Polygon Tabellen (B-rep Liste)
 - ◆ Geometrische/Attribut-Tabelle
 - ◆ Eckpunkte, Kanten, Polygon Tabellen
 - ◆ Konsistenz, Vollständigkeitschecks

Polygonfläche: Datenstruktur

VERTEX TABLE

V_1 :	x_1, y_1, z_1
V_2 :	x_2, y_2, z_2
V_3 :	x_3, y_3, z_3
V_4 :	x_4, y_4, z_4
V_5 :	x_5, y_5, z_5



EDGE TABLE

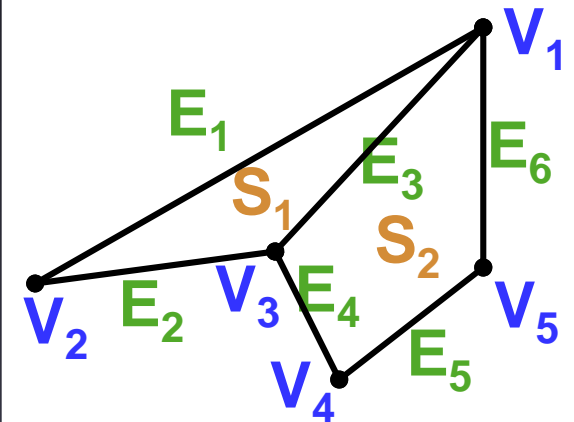
E_1 :	V_1, V_2
E_2 :	V_2, V_3
E_3 :	V_3, V_1
E_4 :	V_3, V_4
E_5 :	V_4, V_5
E_6 :	V_5, V_1

POLYGON-SURFACE TABLE

S_1 :	E_1, E_2, E_3
S_2 :	E_3, E_4, E_5, E_6

Liste für B-Reps

Flächen Liste



Kanten Liste



E_1 E_2 E_3 E_4 E_5 E_6

Eckpunkt
Liste

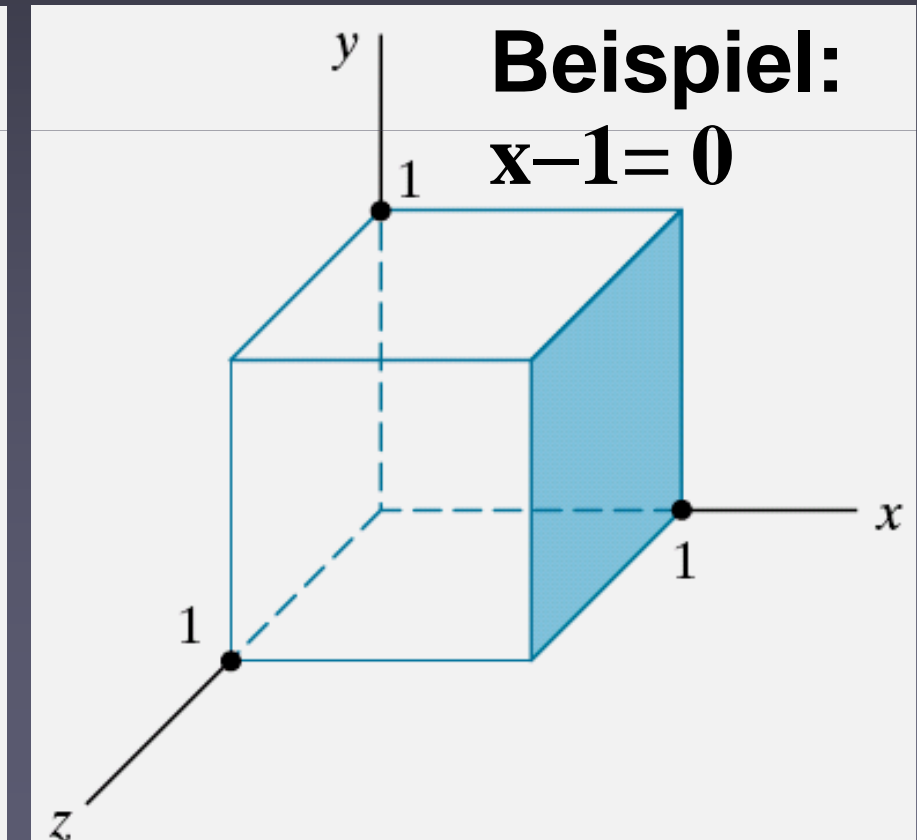
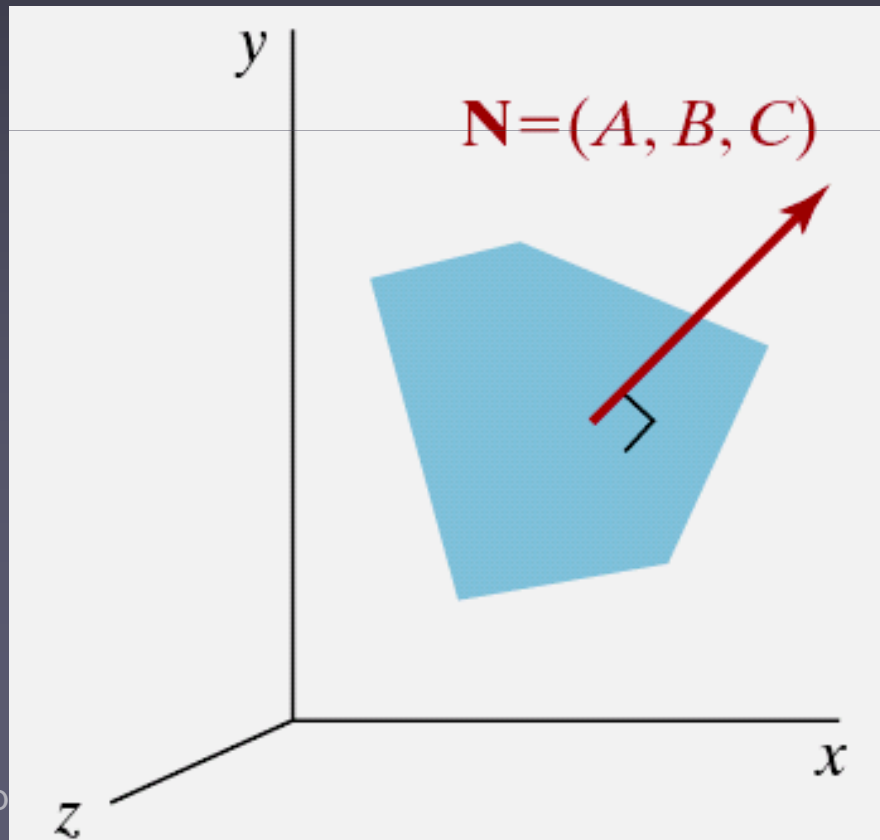


V_1 V_2 V_3 V_4 V_5

Polygonflächen: Ebenengleichung

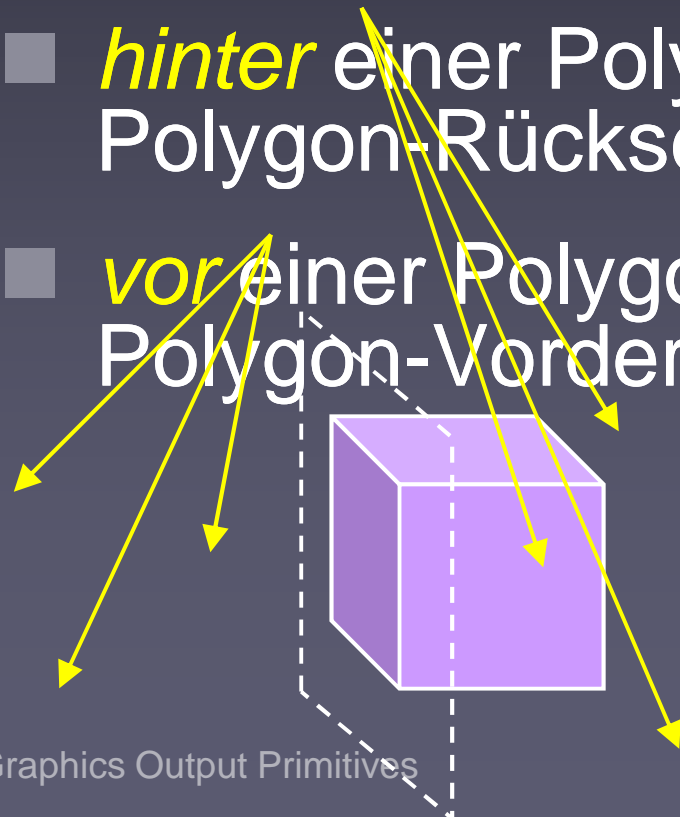
$$Ax + By + Cz + D = 0$$

- Ebenenparameter A, B, C, D
- Normale $= (A, B, C)$



Vorderansicht und Rückansicht des Polygons

- **Rückseite** = Polygonseite, welche ins innere des Objektes zeigt
- **Vorderseite** = Polygonseite, welche nach aussen zeigt
- **hintere** einer Polygon Ebene = sichtbar für die Polygon-Rückseite
- **vor** einer Polygonebene = sichtbar für die Polygon-Vorderseite



Vorder- und Rückseite eines Polygons

$Ax + By + Cz + D = 0$ für Punkte auf der Fläche
 < 0 für Punkte dahinter
 > 0 für Punkte davor

wenn (1) rechtsgewundenes Koordinatensystem
(2) Polygonpunkte gegen den
Uhrzeigersinn angeordnet sind

V_1, V_2, V_3 gegen den Uhrzeigersinn \Rightarrow
Normalvektor $\mathbf{N} = (\mathbf{V}_2 - \mathbf{V}_1) \times (\mathbf{V}_3 - \mathbf{V}_1)$

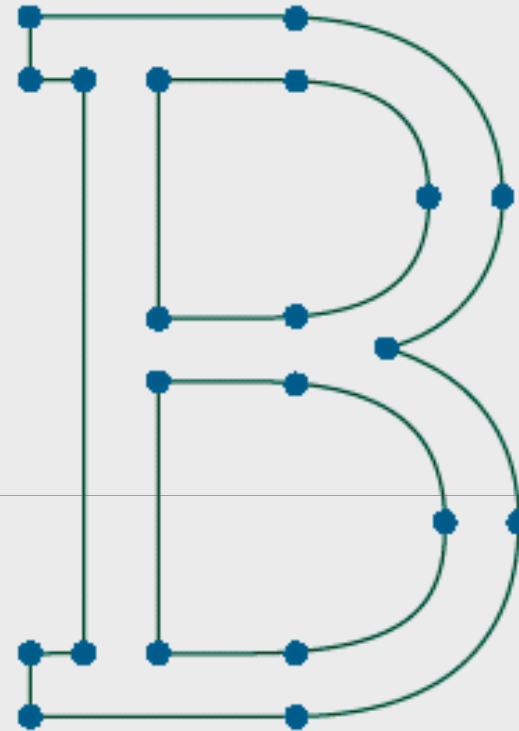
Schriftzeichen-Primitive

- Schrift (Schriftart)
 - ◆ Designs für Zeichen
- Courier, Times, Arial, ...
 - ◆ *Serif* (besser lesbar),
 - ◆ *Sans serif* (besser lesbar)
- Definitionsmodell
 - ◆ *Bitmap font* (einfach zu definieren und anzuzeigen), benötigt mehr Platz (Schriftspeicher)
 - ◆ *Outline font* (aufwendiger, weniger Platz, geometrische Transformationen)

Sfzrn
Sfzrn

Zeichengenerierung - Beispiel

1	1	1	1	1	1	0	0
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0
0	1	1	1	1	1	0	0
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0
1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0



Repräsentiert

- mit einer 8x8 Bitmap

- mit geraden Linien und Kurvensegmenten

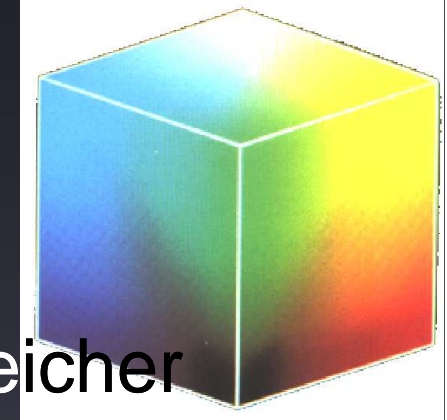
Attributes of Graphics Primitives

Hearn/Baker
4.2, 4.4-4.6, 4.9-4.13, 4.15, 4.17



One frame from a computer-generated cartoon illustrating a variety of object colors and other attributes. (Courtesy of SOFTIMAGE, Inc.)

Farb- und Grauskala



- Farbinformation: RGB
 - ◆ Farbkodierung direkt im Bildschirmspeicher
 - 1024 x 1024 x 24 bit/Farbe
⇒ 3 Mbyte Speicher

- ◆ Farb (Nachschlage) Tabellen

- weniger Speicher
- Farben leicht zu ändern (z.B. für Farbkodierung)

- Grauskalierung

- ◆ Berechnet von RGB

<u>rot</u>	<u>grün</u>	<u>blau</u>	
0	0	0	schwarz
0	0	1	blau
0	1	0	grün
0	1	1	cyan
1	0	0	rot
1	0	1	magenta
1	1	0	gelb
1	1	1	weiß

z.B. Intensität = $0.5 [\min(r,g,b) + \max(r,g,b)]$

Farb-Nachschlage Tabelle

anstatt von:

		j
i		r

		j
i		g

		j
i		b

jetzt:

		j
i		xx

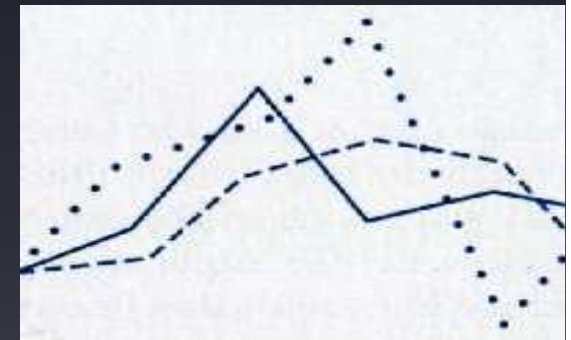
xx	r	g	b

ausgewählte
Farben mit
Tabellen-
index
adressiert

Linienattribute (1)

■ Typ

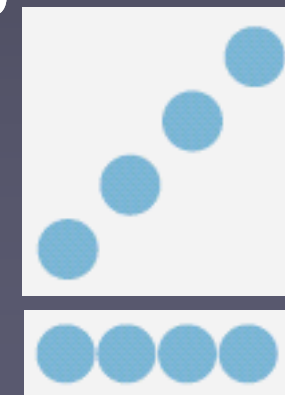
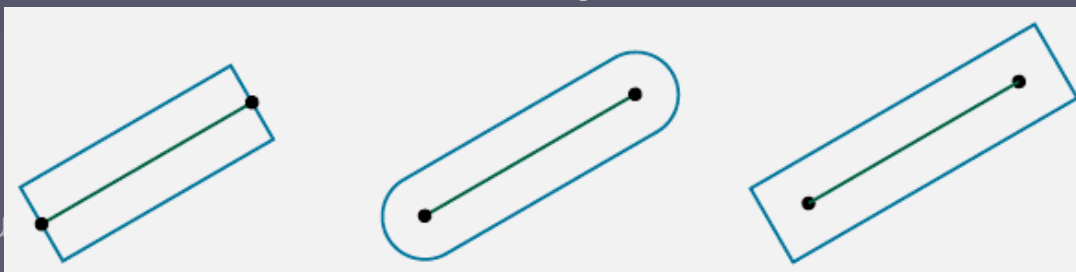
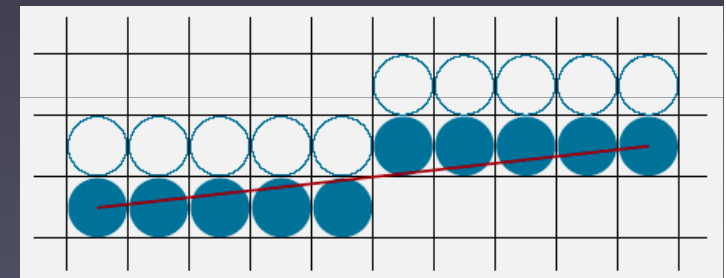
- ◆ solide, gestrichelt, gepunktet
- ◆ Pixel Maske (Algorithmen zum Rastern von Linien)



Rastern

■ Breite

- ◆ Bresenham + ergänzende vertikale (horizontale) Abgrenzungen
- ◆ Breite abhängig von Steigung
- ◆ Linienabdeckung



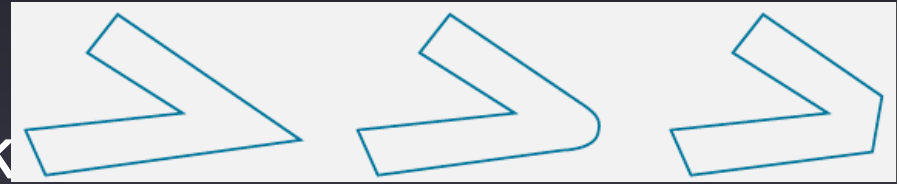
Dieter Schmalstieg

Linienattribute (2)

■ Breite

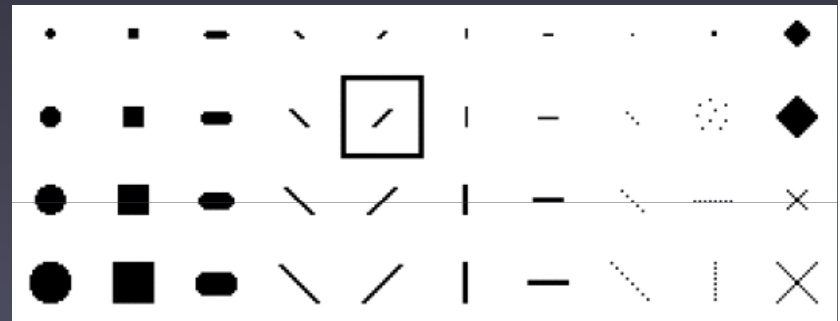
◆ starke Linien

- als gefüllte Rechtecke
- joining two segments: spitz rund schräg

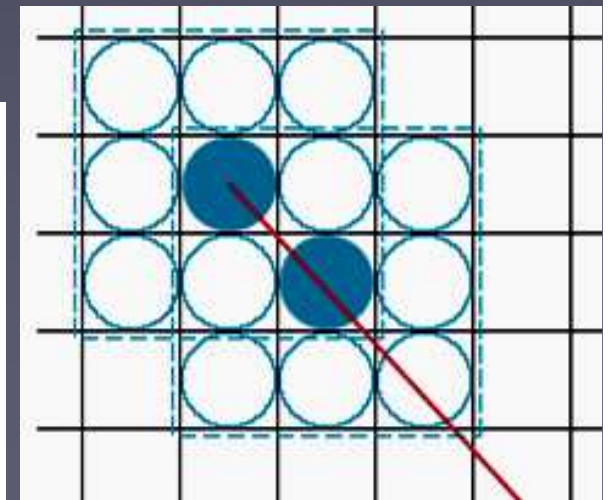


■ Stift- und Pinsoptionen

- ◆ Form, Größe, Vorlage
- ◆ Pixelmaske
- ◆ Simulation von Pinselstrichen



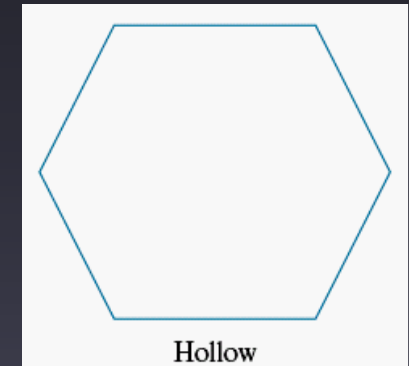
■ Farbe



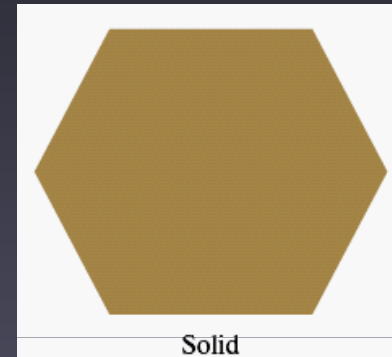
Füllflächen Attribute (1)

■ Fülltypen

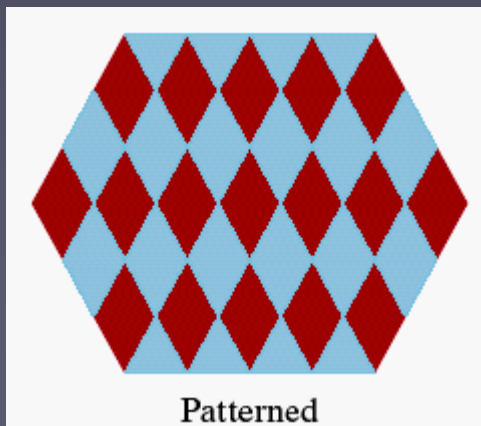
◆ ohne Füllung mit farblichen Rand



◆ gefüllt mit Einheitsfarbe

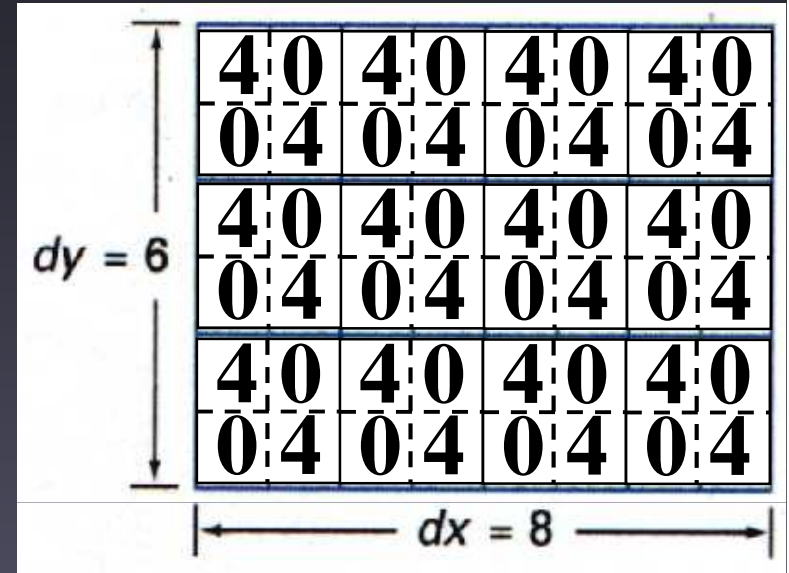


◆ Gefüllt mit speziellen Muster oder Design (z.B. schraffiert -> hatch,...)

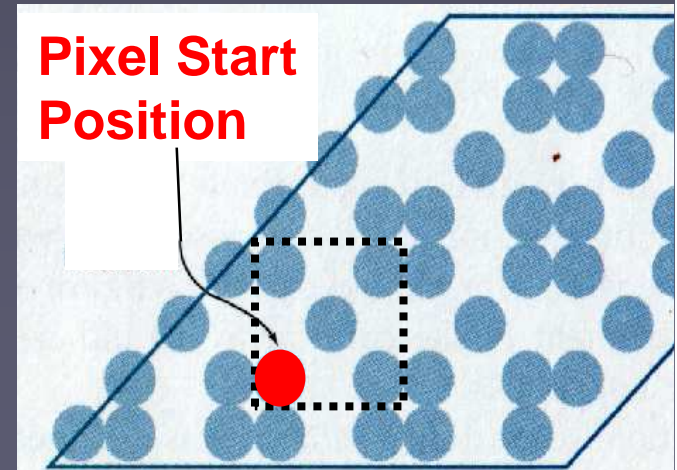
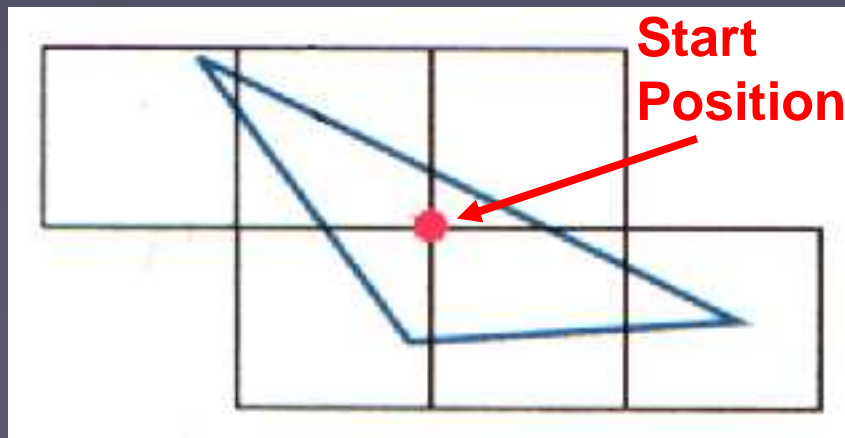


Füllflächen Attribute (2)

- Fülloptionen
 - ◆ Kantentyp, Breite, Farbe
- Modellpezifizierung
 - ◆ über Modelltabellen

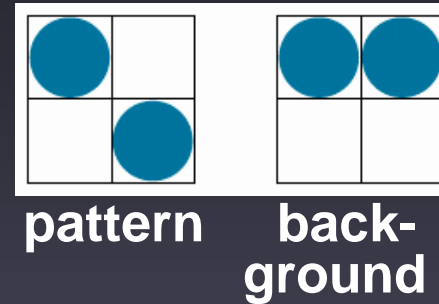


- ◆ deckend (Modell-Referenzpunkt)



Füllflächen Attribute (3)

- Kombination von Füllmodell mit Hintergrundfarben



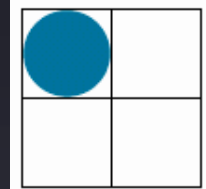
- Softfüllung
 - ◆ Farbkombinationen
 - ◆ Antialiasing bei Objekträndern
 - ◆ Simulation von halbtransparenten Pinselstrichen
 - ◆ Beispiel: Lineare Softfüllung

V...Vordergrundfarbe

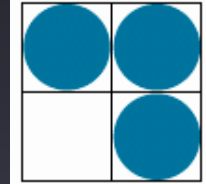
H...Hintergrundfarbe

$$P = tV + (1 - t)H$$

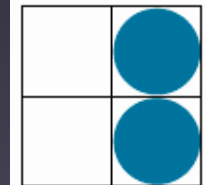
und



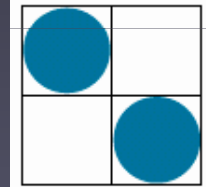
oder



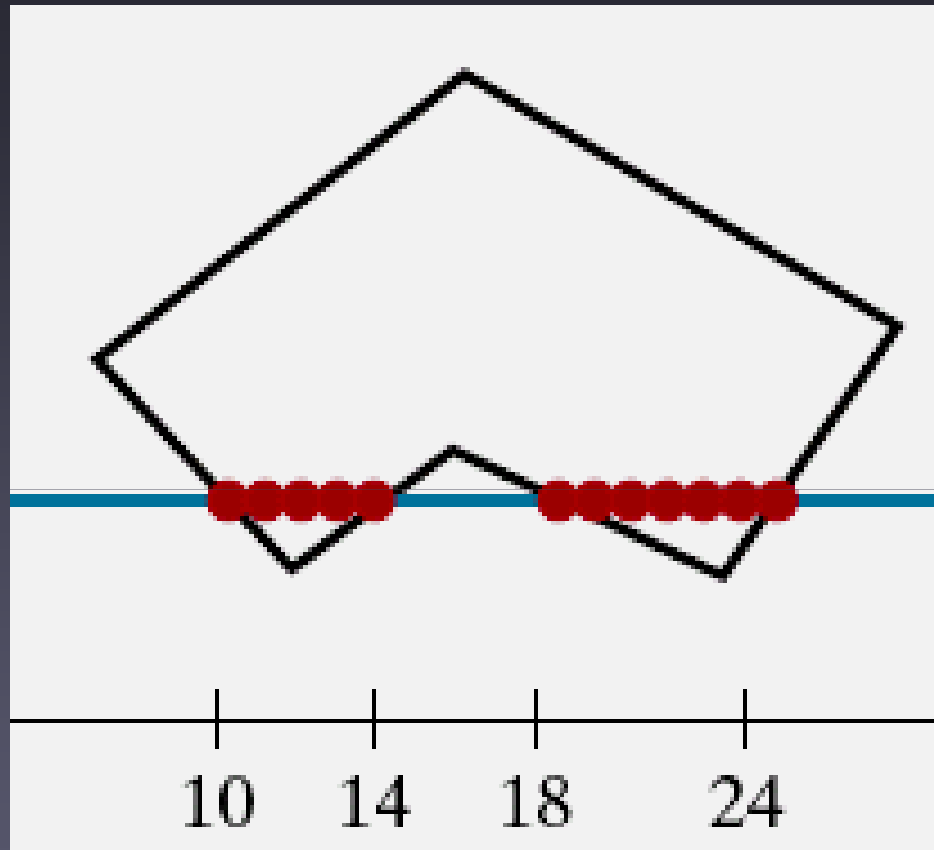
xor



austauschen



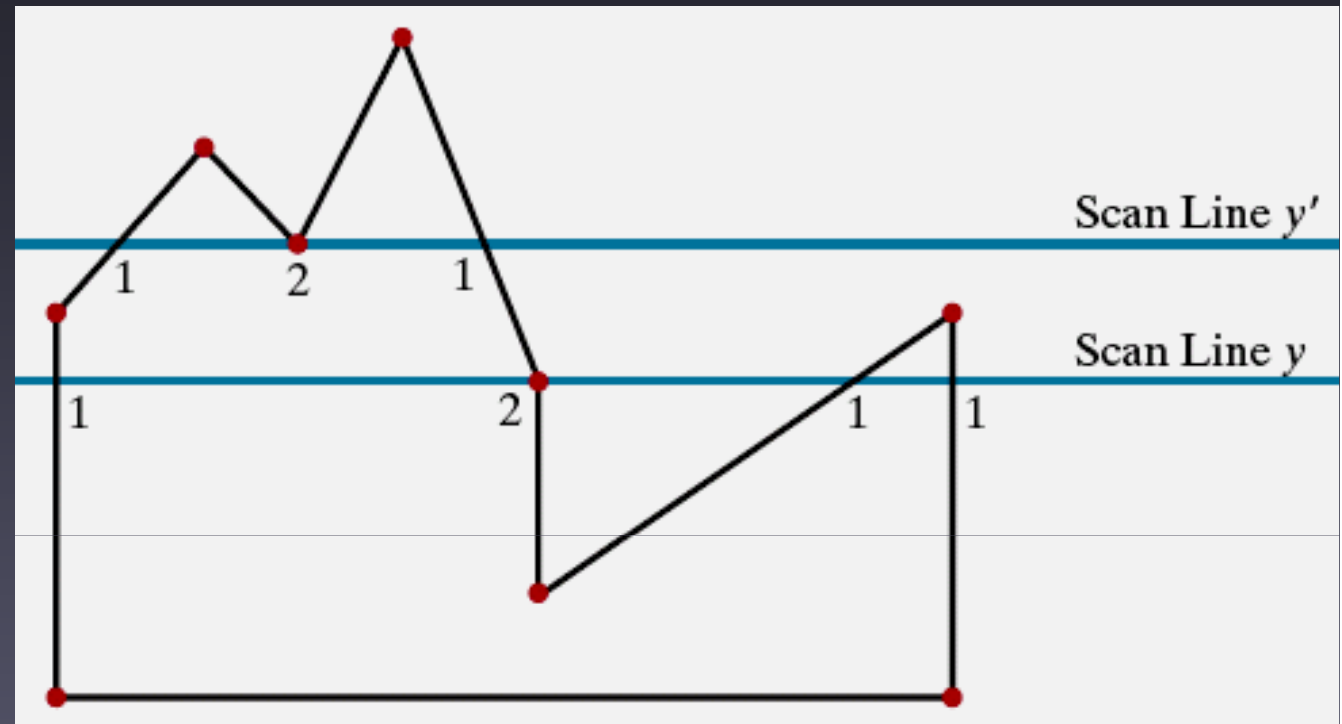
Scanline-Polygon-Füllalgorithmus



eine Scanlinie durchläuft die
Innenpixel des Polygons

Scan-L.-Füllen: Eckschnittpunkte

Schnittpunkte entlang der Scanlinien, wo Eckpunkte des Polygons sind



y: ungerade Anzahl von Schnittpunkten

y': gerade Anzahl von Schnittpunkten - können korrekt für den inneren Pixelabstand angeordnet werden

Scan-L. Füllen: Inkrementelles Update

- Inkrementelles Update der Schnittpunkte

Steigung der Polygongrenzlinie: m

$$m = \frac{y_{k+1} - y_k}{x_{k+1} - x_k} \quad \Rightarrow \quad x_{k+1} = x_k + \frac{1}{m}$$

$$y_{k+1} - y_k = 1$$

(für 2 aufeinanderfolgende Scanlinien)

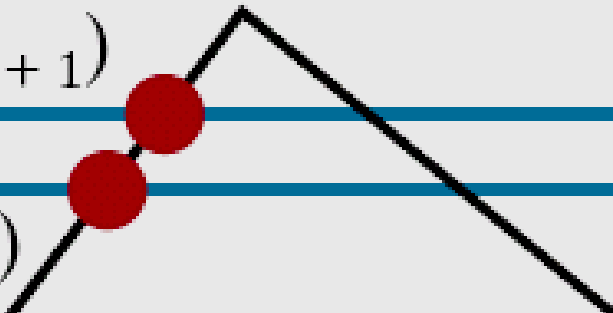
$$x_{k+1} = x_k + \frac{\Delta x}{\Delta y}$$

(x_{k+1}, y_{k+1})

Scan Line $y_k + 1$

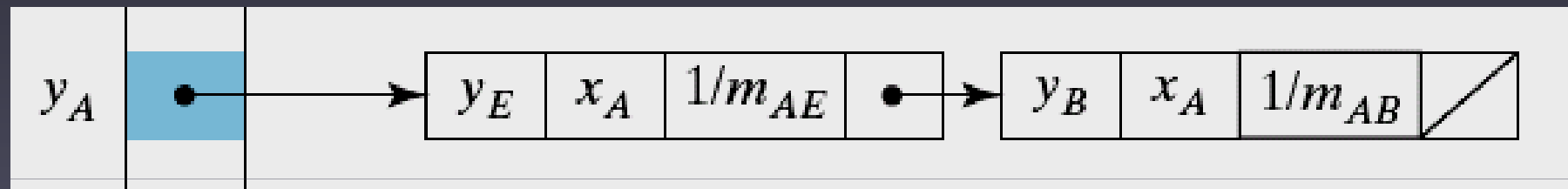
(x_k, y_k)

Scan Line y_k



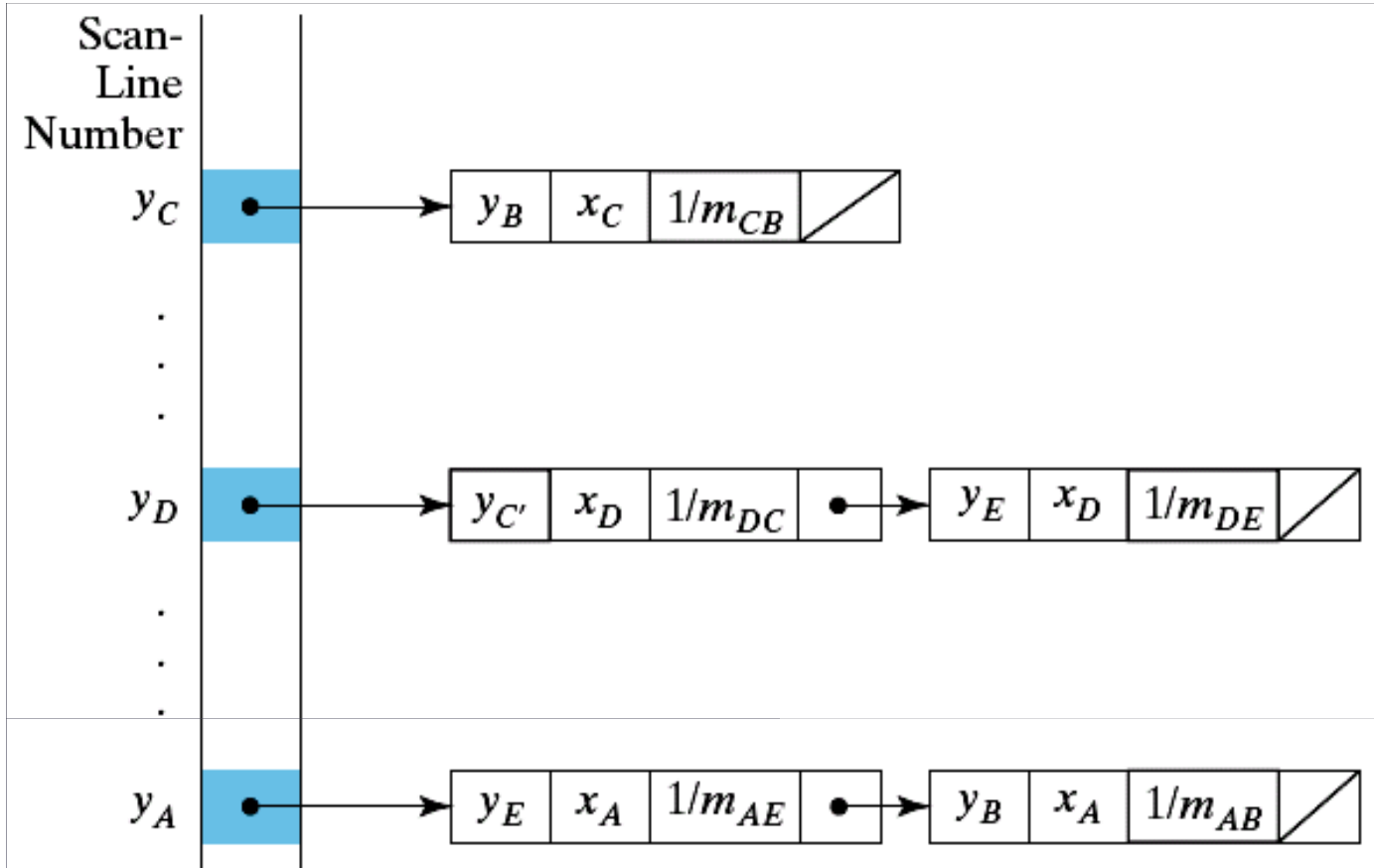
Scan-L. Füllen: Sortierte Kantenabelle

- Sortiert alle Kanten anhand des kleinsten y-Wertes
- Kanteneintrag:
[max y-Wert, x-Schnittpunkt, Inverse Steigung]



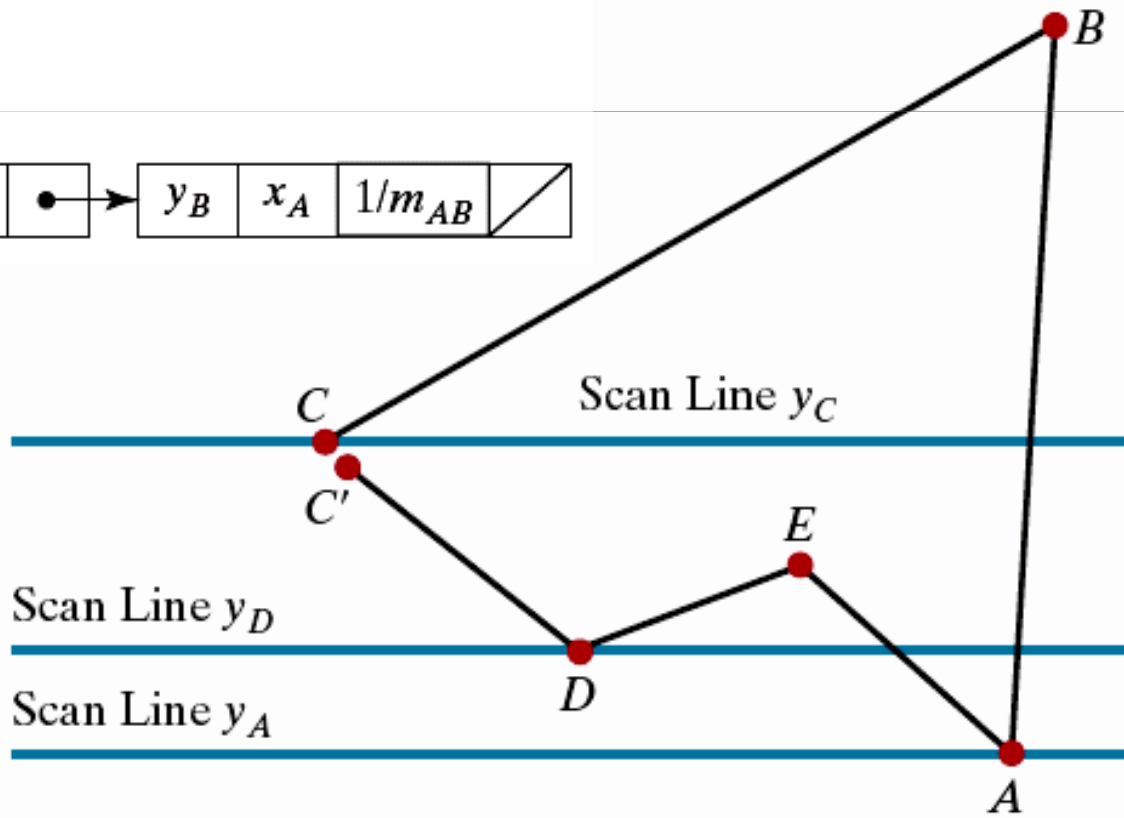
- **Aktuelle-Kanten Liste**
 - ◆ Für jede Scanlinie
 - ◆ Beinhaltet alle Kanten, welche die Scanlinie kreuzen
 - ◆ Inkrementelle Vorgangsweise
- Aufeinanderfolgende Schnittpunktpaare (Abstand) werden gefüllt

Sortierte Kanten-tabelle:



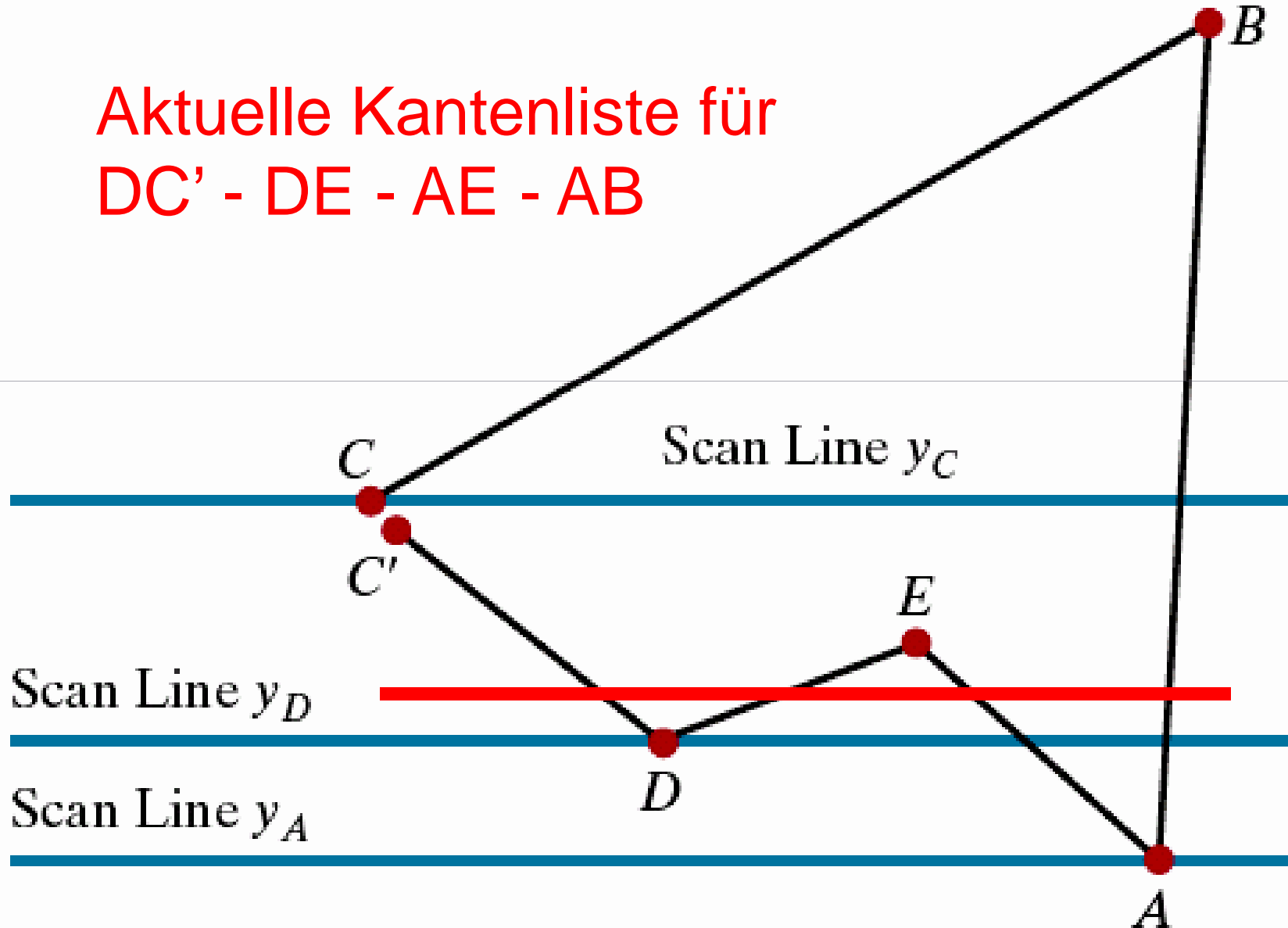
Ein Polygon und dessen sortierte Kanten-tabelle mit den Kanten DC in gekürzter Form

Graphics Output Primitives



Scan-Line Füllen: Aktuelle Kantenliste

Aktuelle Kantenliste für
DC' - DE - AE - AB

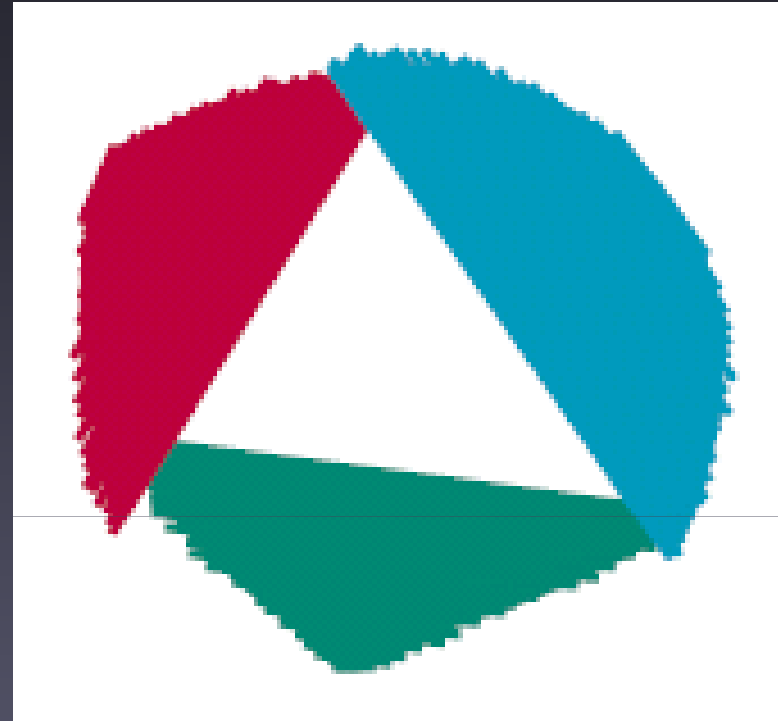


Flood-Fill Algorithmus

- Füllen von Pixelbereichen
 - ◆ Mehrfarbige Bereiche
 - ◆ Startend von Innenpunkt
 - ◆ “Überfluten” des inneren Bereichs
 - ◆ 4-verbundene, 8-verbundene Bereiche
 - ◆ Reduzieren von Stackgröße, indem man eine rekursive Aufrufe entfernt

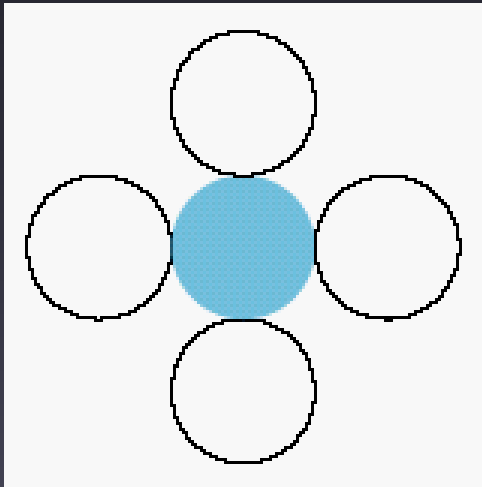
Flood-Fill: Definition der Grenzen

Bereiche müssen
anhand der Grenzen
klar erkennbar sein



Bsp: Fläche mit
Mehrfarbigen Grenzen

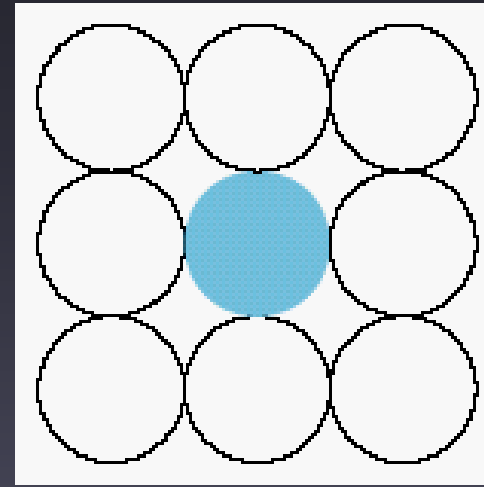
Flood-Fill: Verbundenheit



Definition:

4-verbunden

bedeutet, dass eine Verbindung nur in diesen 4 Richtungen gültig ist

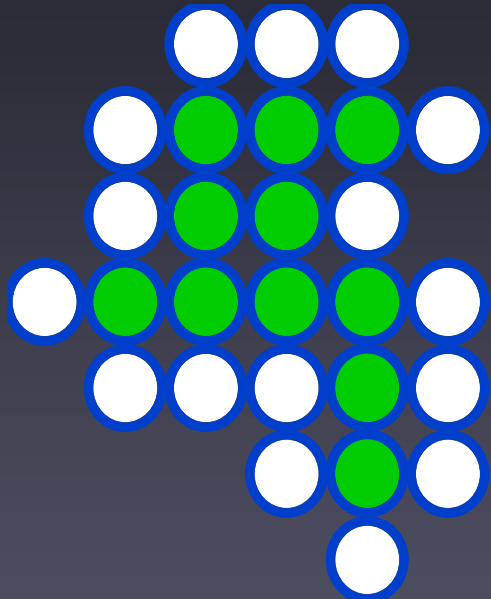


Definition:

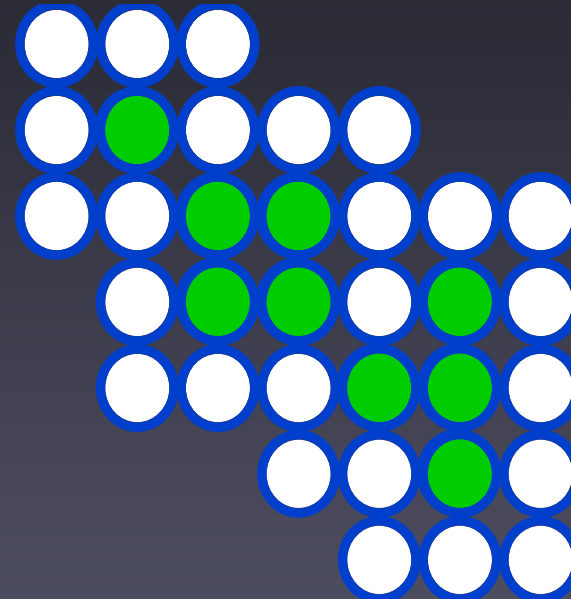
8-verbunden

bedeutet, dass eine Verbindung nur in diesen 8 Richtungen gültig ist

Beispiel für 4- und 8-verbundenen



ein 4-verbundener
Bereich hat einen 8-
verbundenen Rand

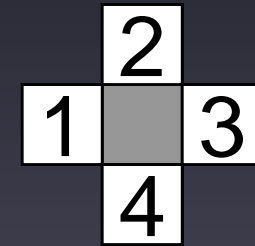
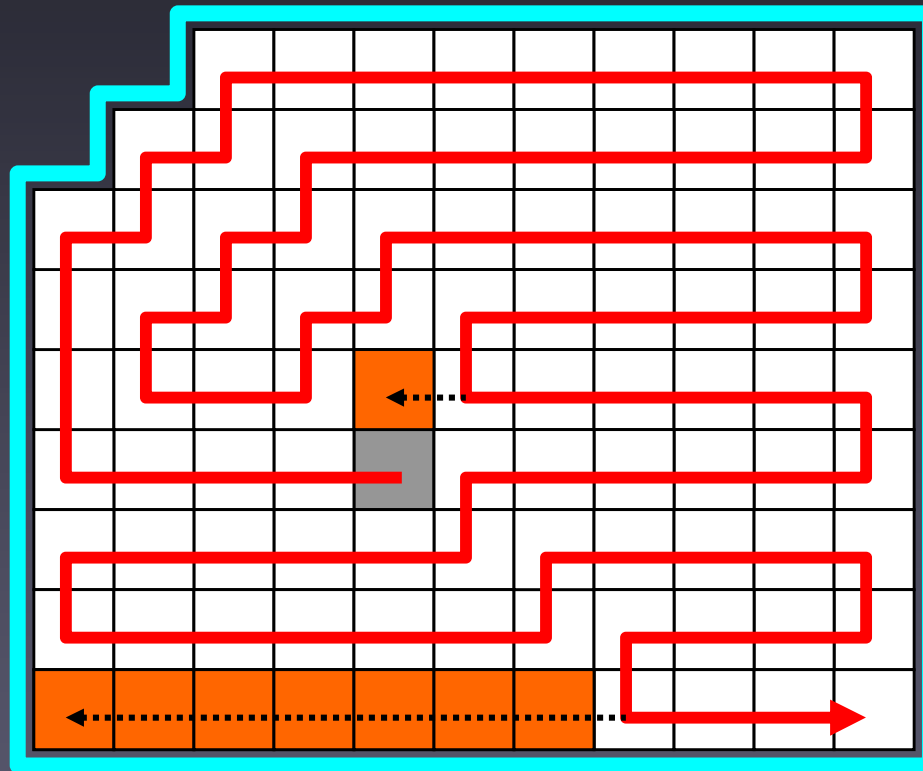


ein 8-verbundener
Bereich hat einen 4-
verbundenen Rand

Einfacher Flood-Fill Algorithmus

```
void floodFill4 (int x, int y, int fillColor, int interiorColor)
{
    int color;
    /* Set current color to fillColor, then perform following */
    getPixel (x, y, color);
    if (color = interiorColor) {
        setPixel (x, y);    // Set color of pixel to fillColor.
        floodFill4 (x + 1, y, fillColor, interiorColor);
        floodFill4 (x - 1, y, fillColor, interiorColor);
        floodFill4 (x, y + 1, fillColor, interiorColor);
        floodFill4 (x, y - 1, fillColor, interiorColor)
    }
}
```

Schlechtes Verhalten von einfachen Flood-Fill

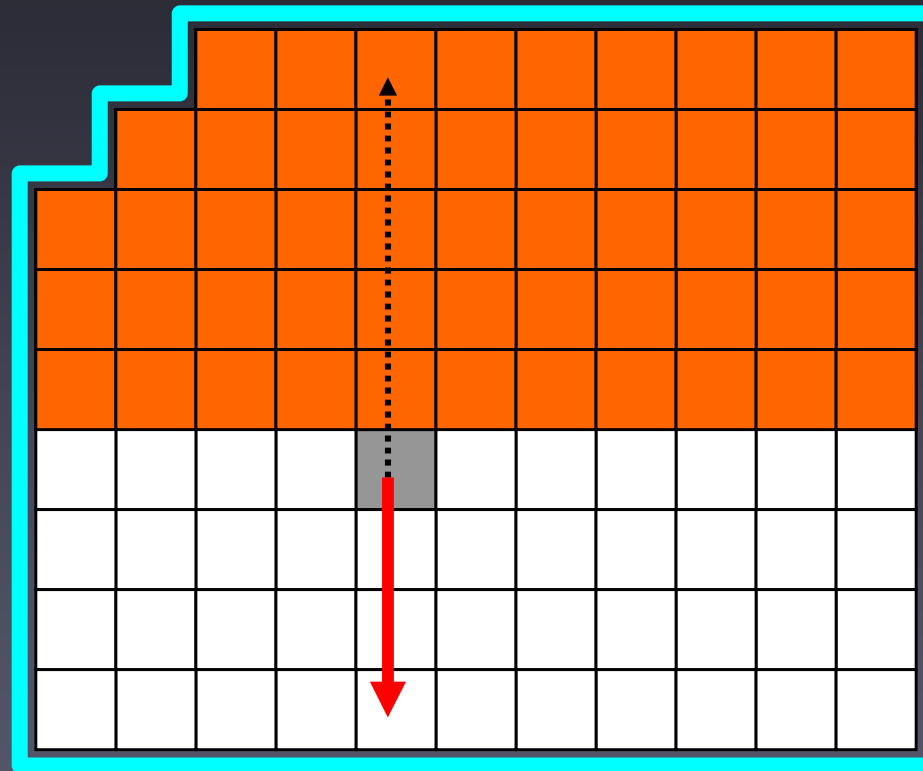


rekursiver
Ablauf

Span Flood-Fill Algorithmus

- floodFill4 erzeugt 2 große Stacks (Rekursion!)
- Lösung:
 - ◆ Inkrementelle horizontale Füllung (links nach rechts)
 - ◆ Rekursive vertikale Füllung (zuerst nach oben dann nach unten)

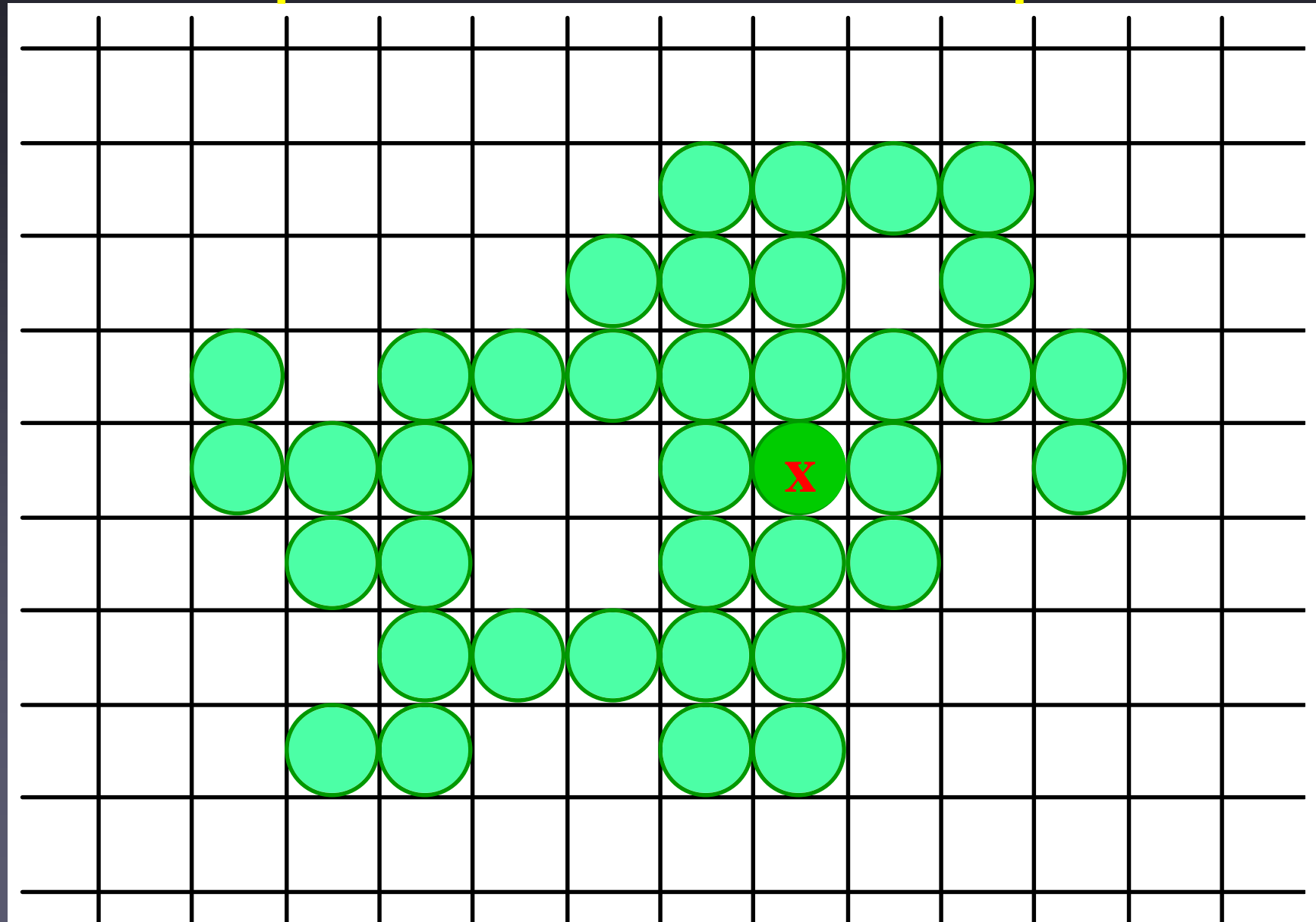
Gutes Verhalten des Span Flood-Fill



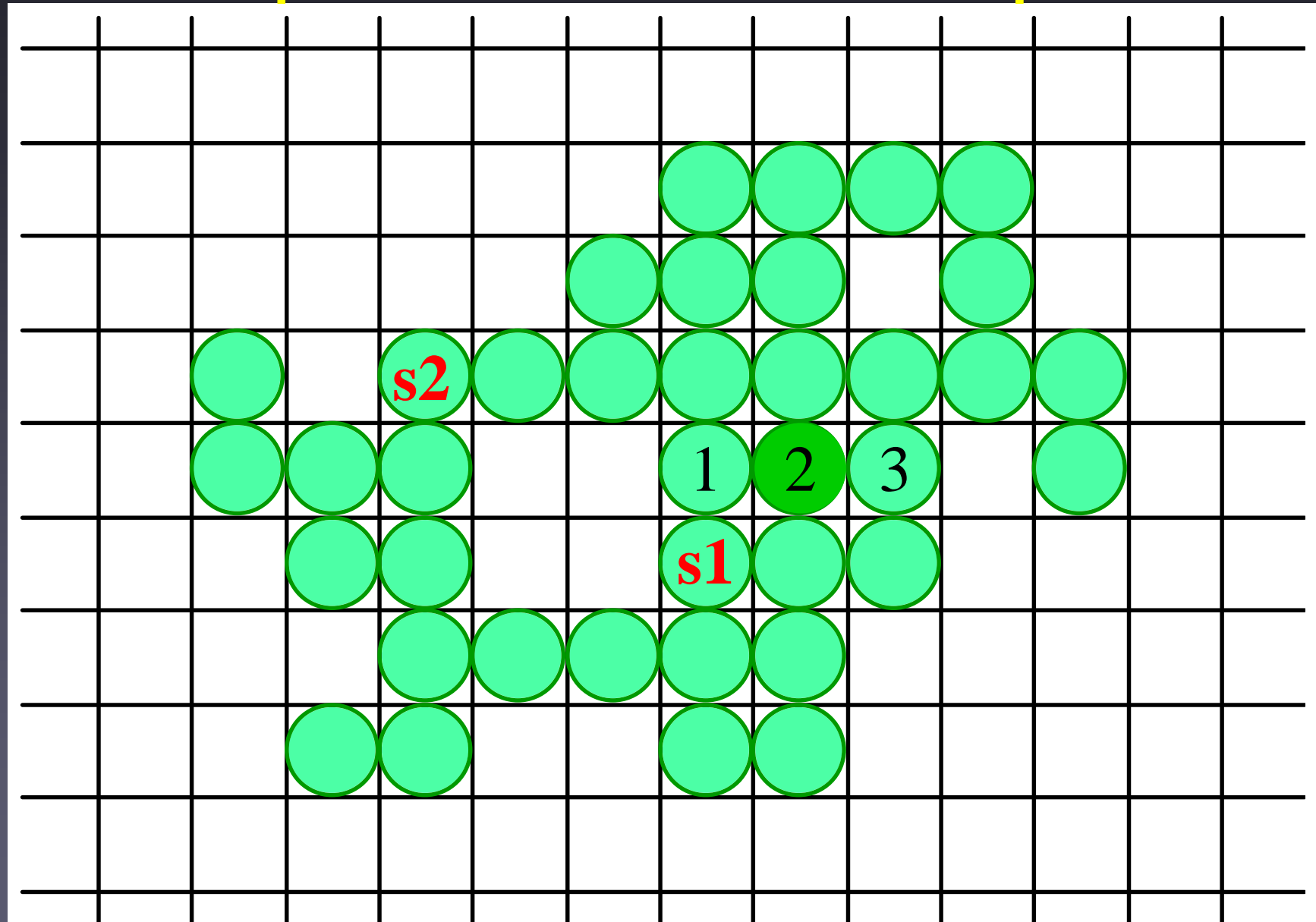
1
2

rekursiver
Ablauf

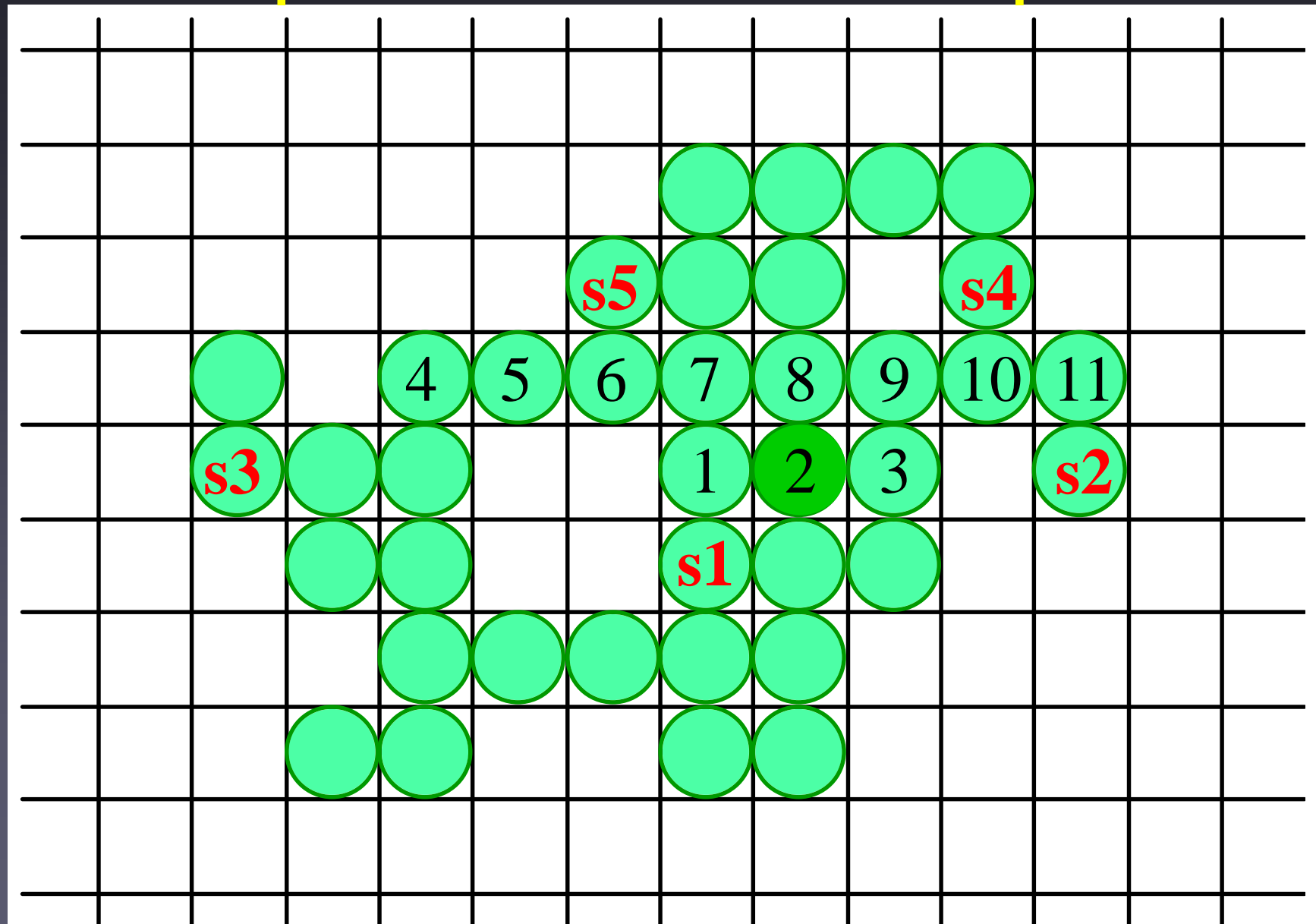
Span Flood-Fill Beispiel



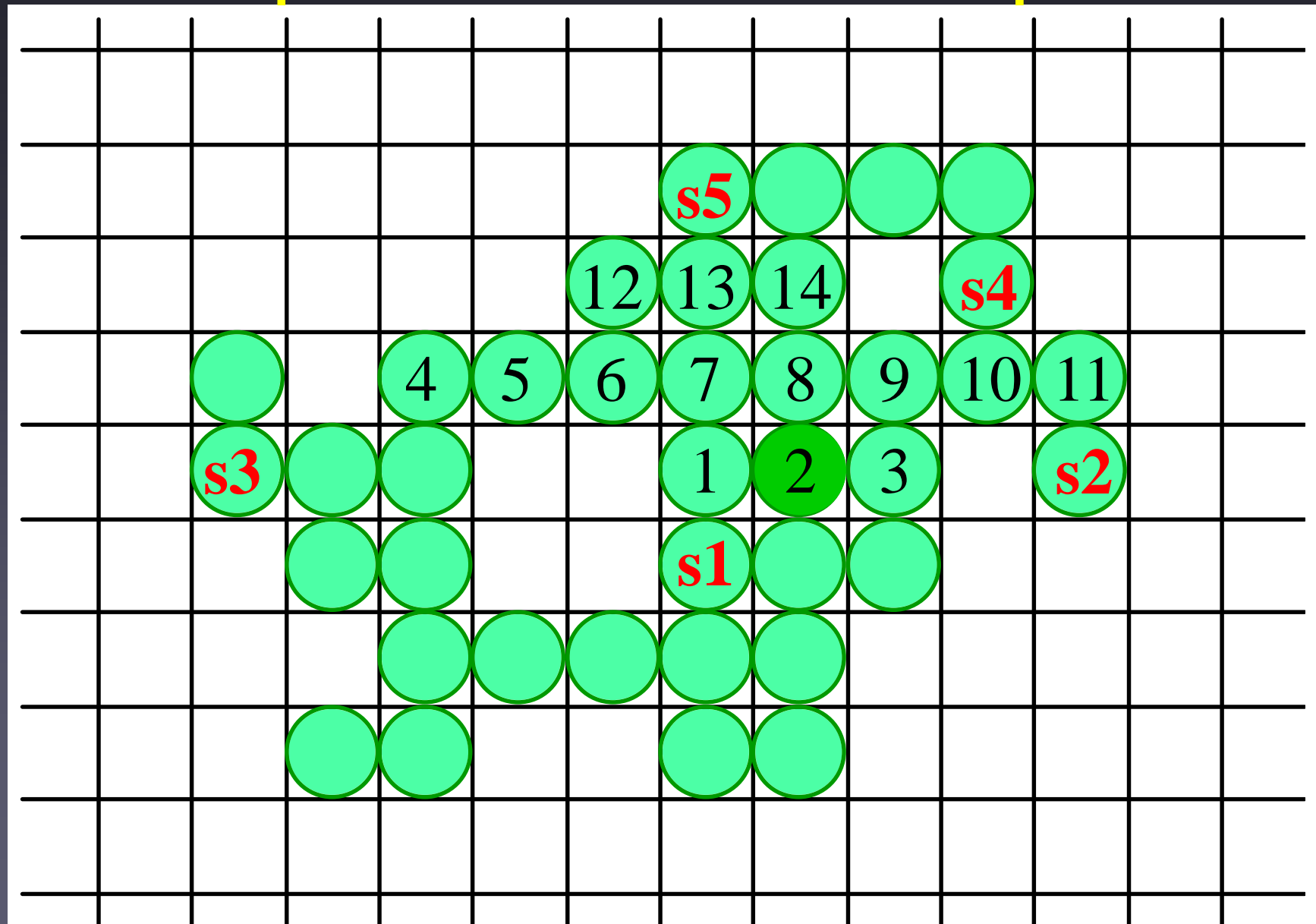
Span Flood-Fill Beispiel



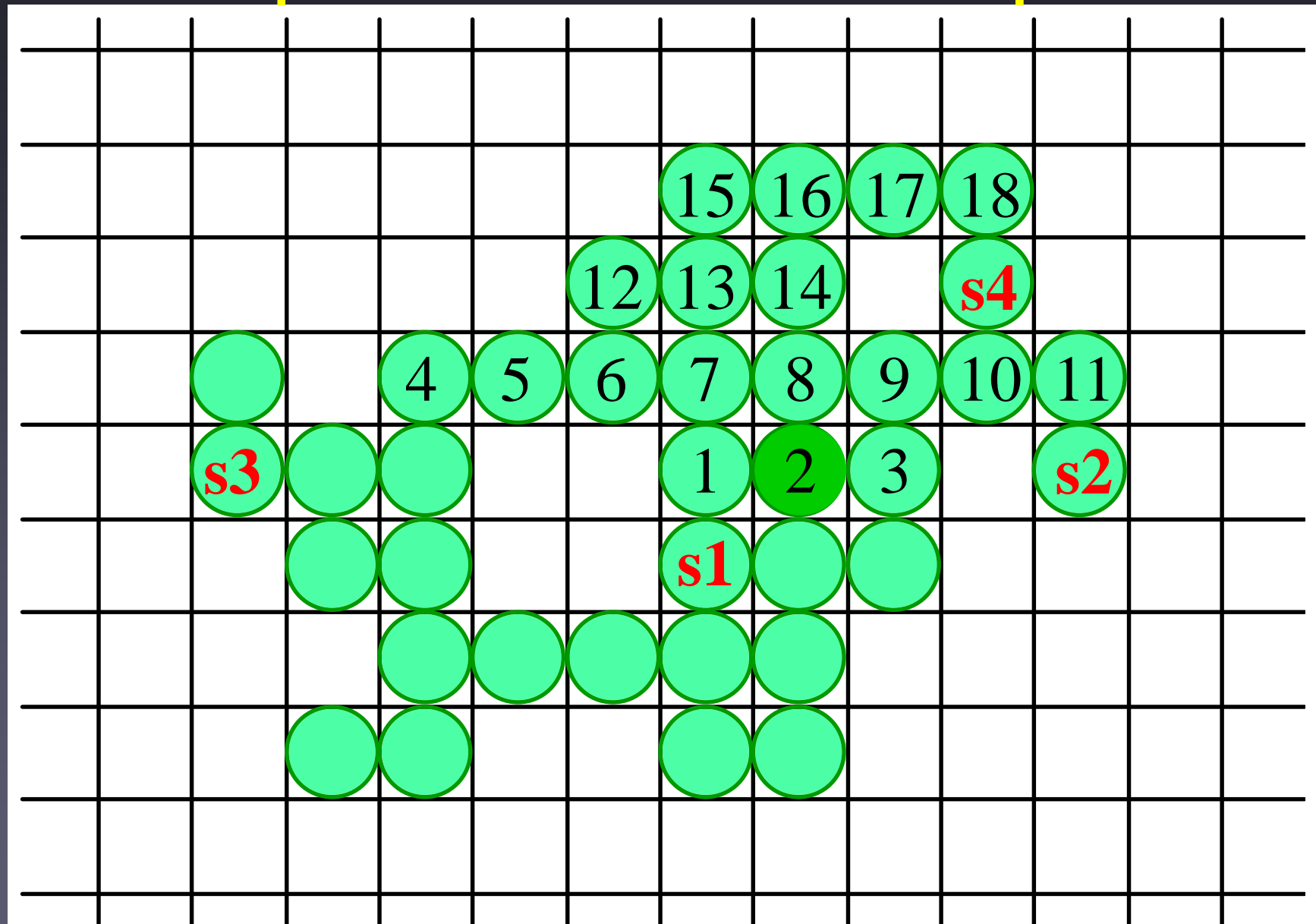
Span Flood-Fill Beispiel



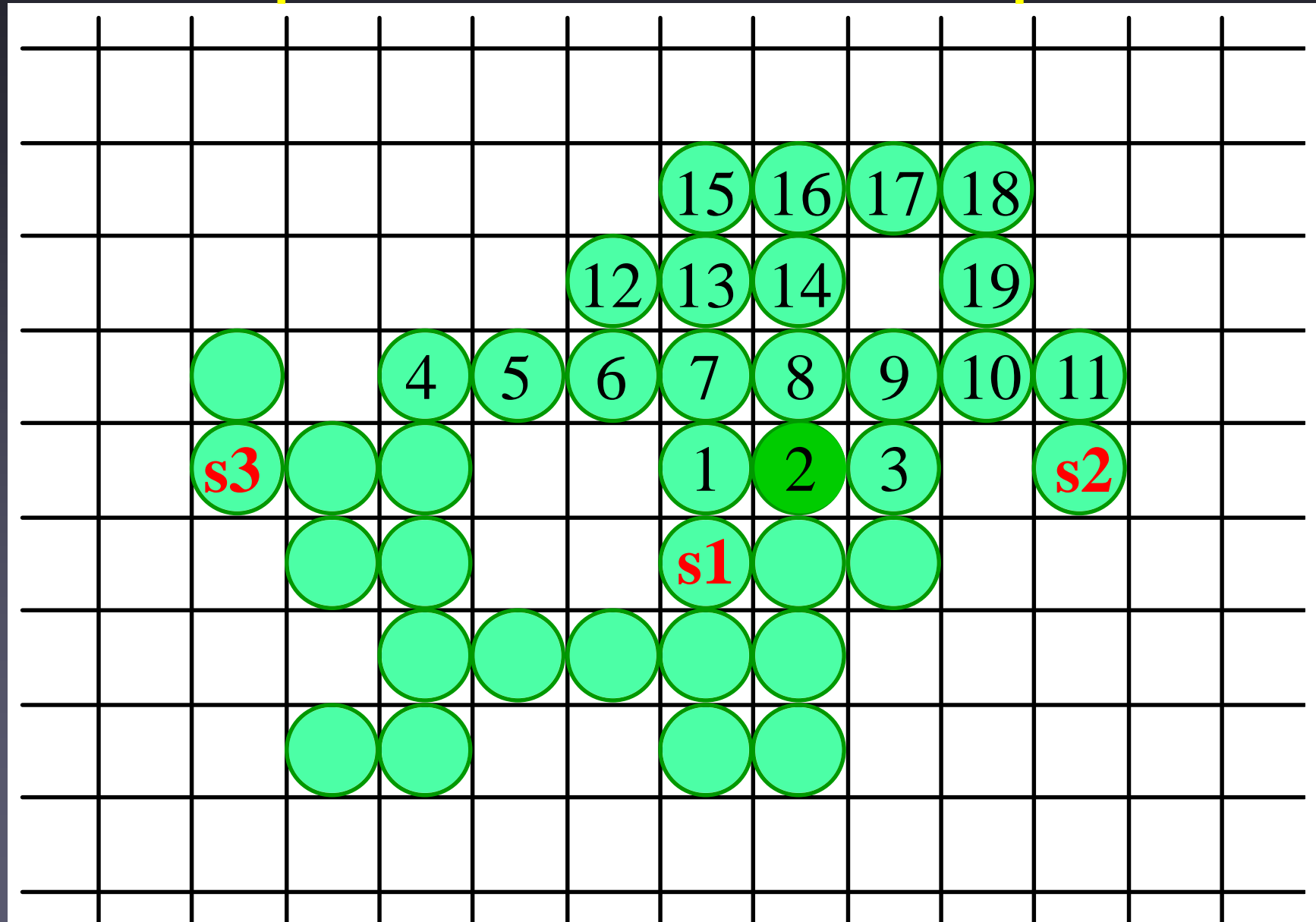
Span Flood-Fill Beispiel



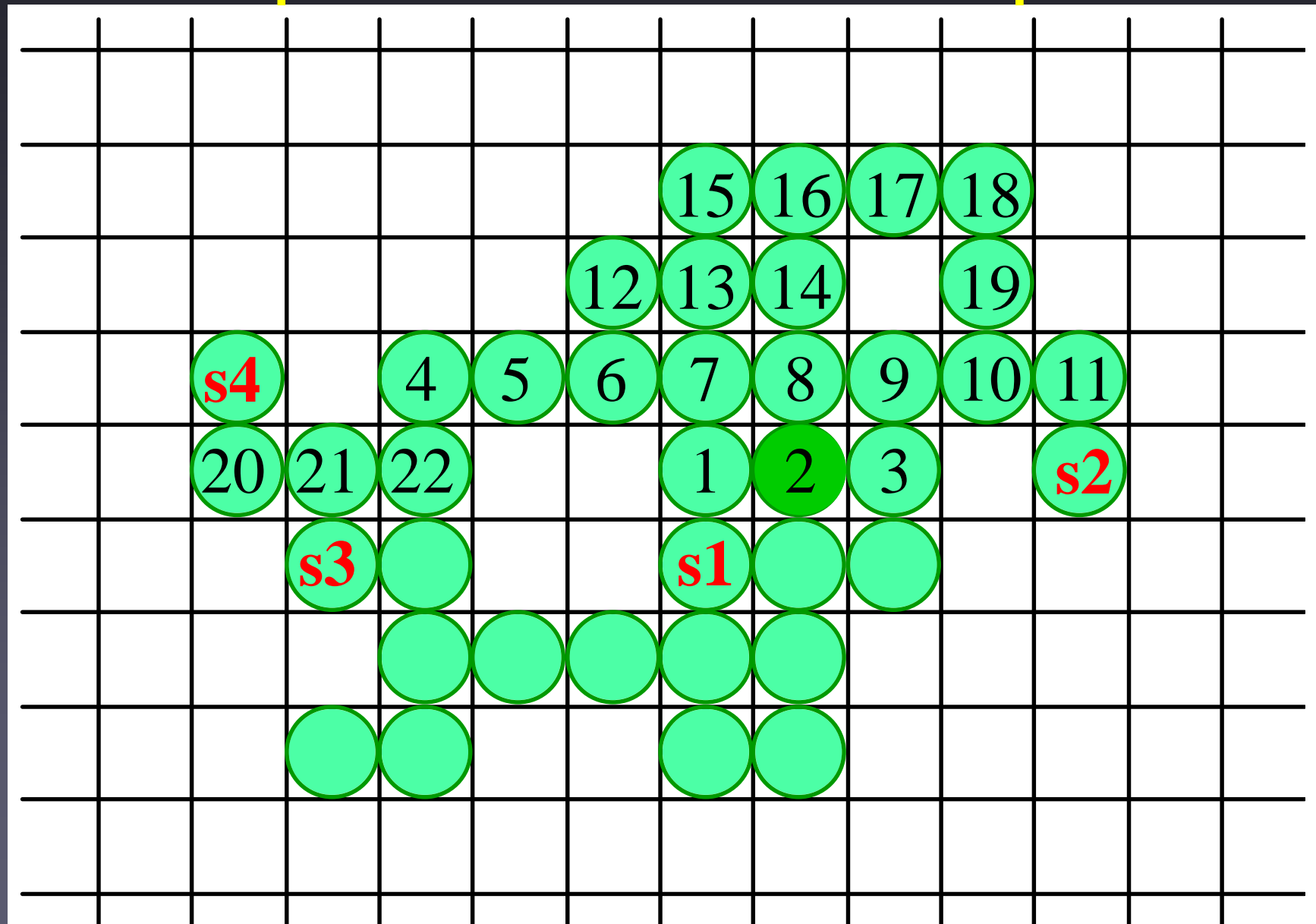
Span Flood-Fill Beispiel



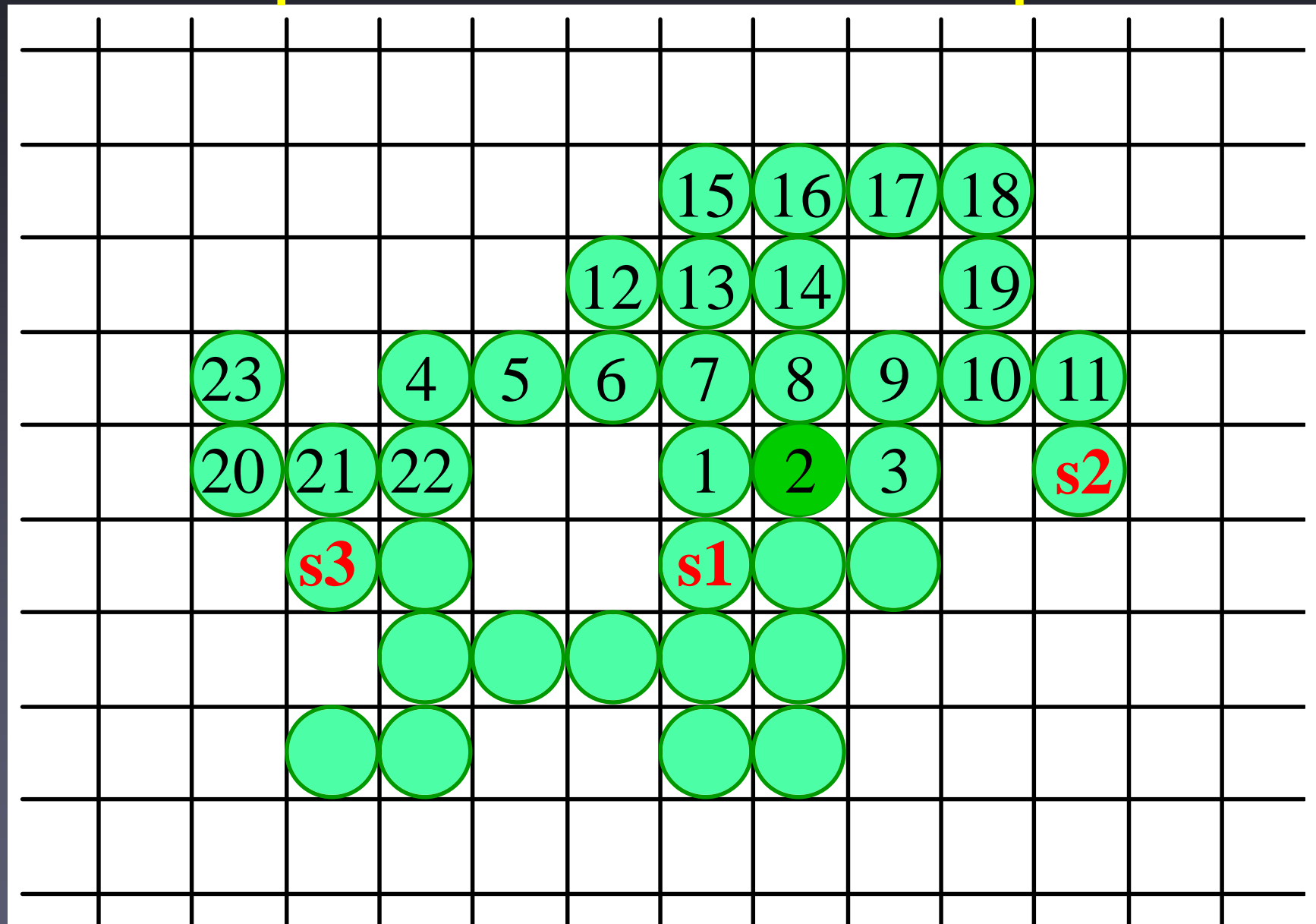
Span Flood-Fill Beispiel



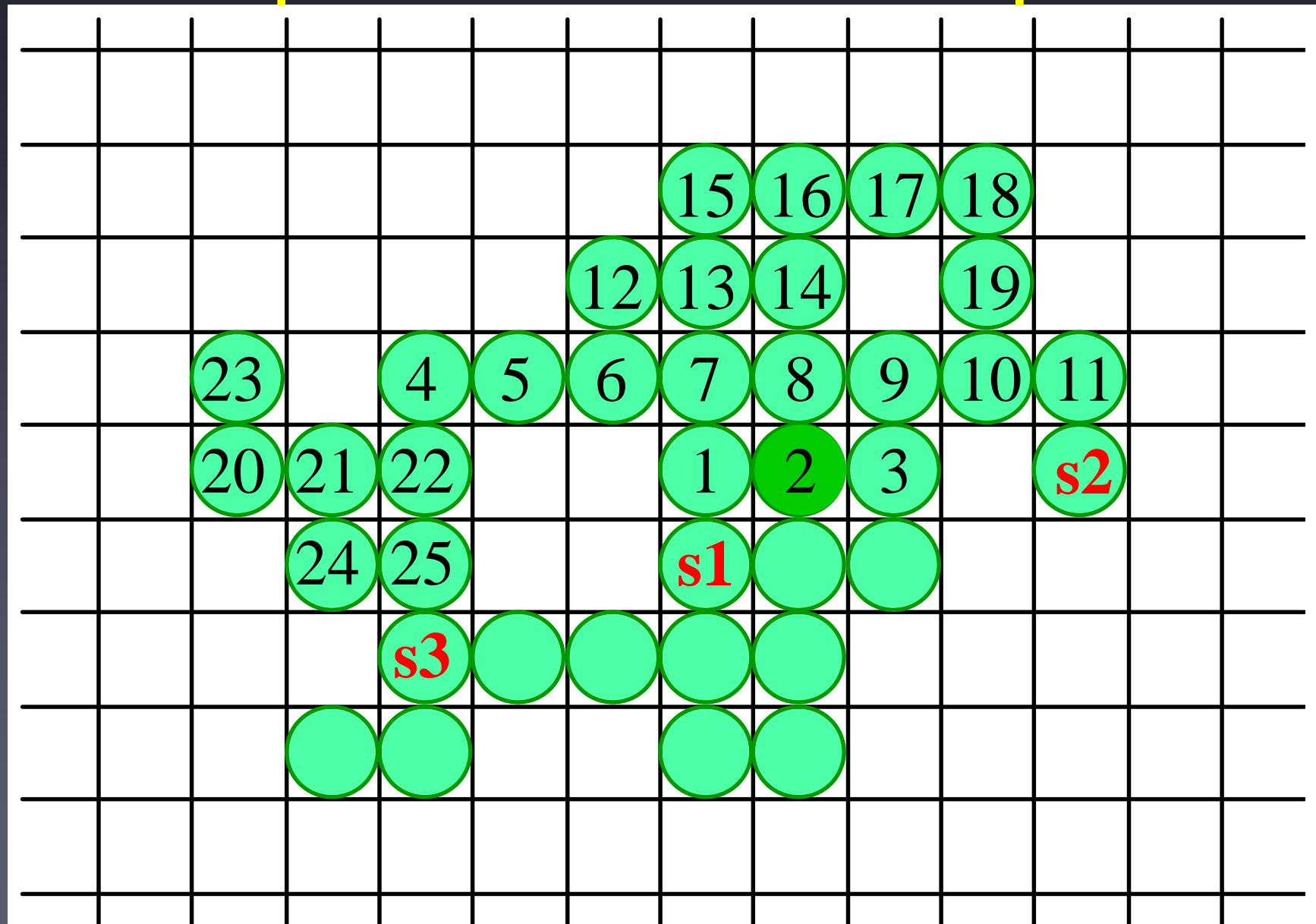
Span Flood-Fill Beispiel



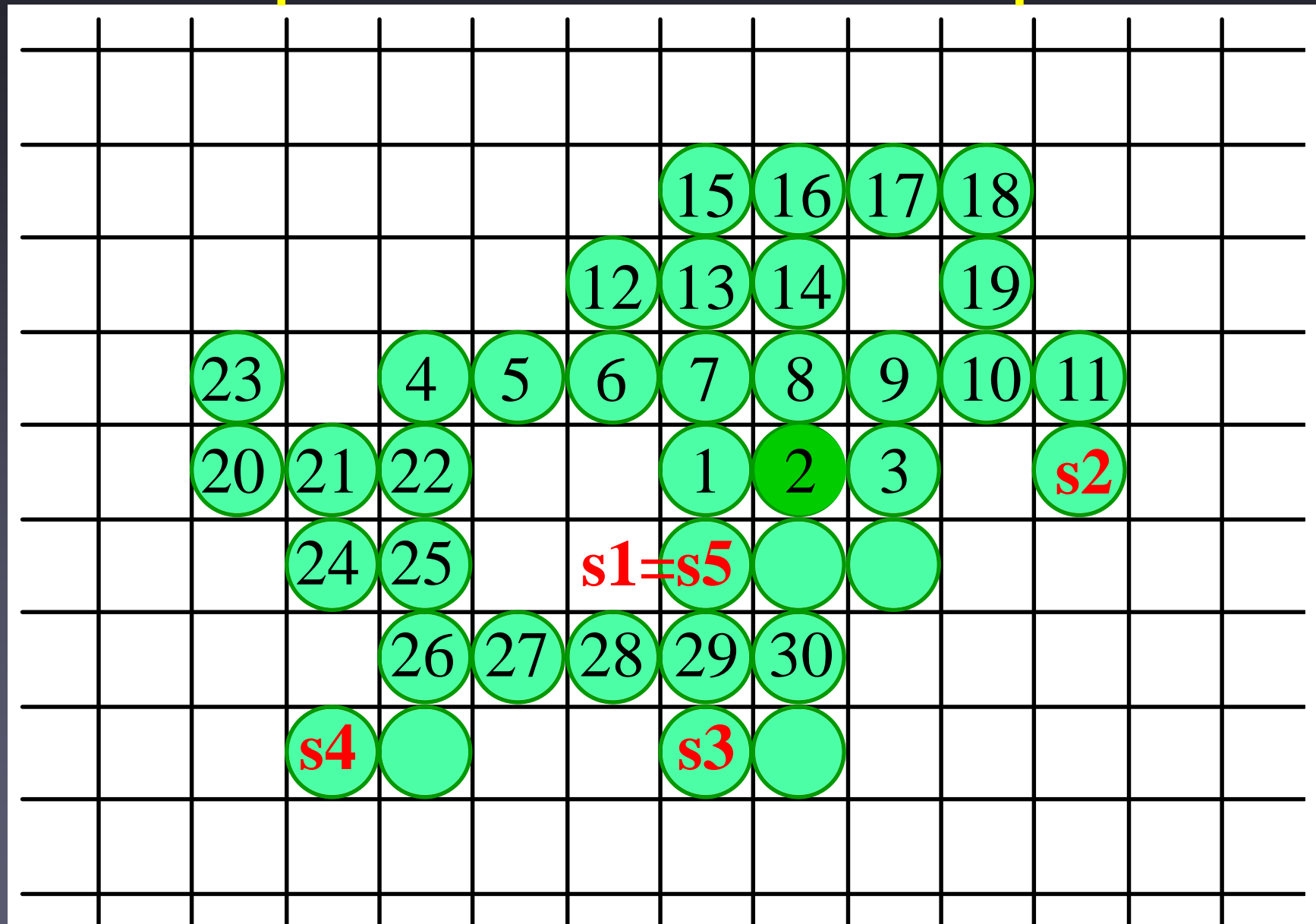
Span Flood-Fill Beispiel



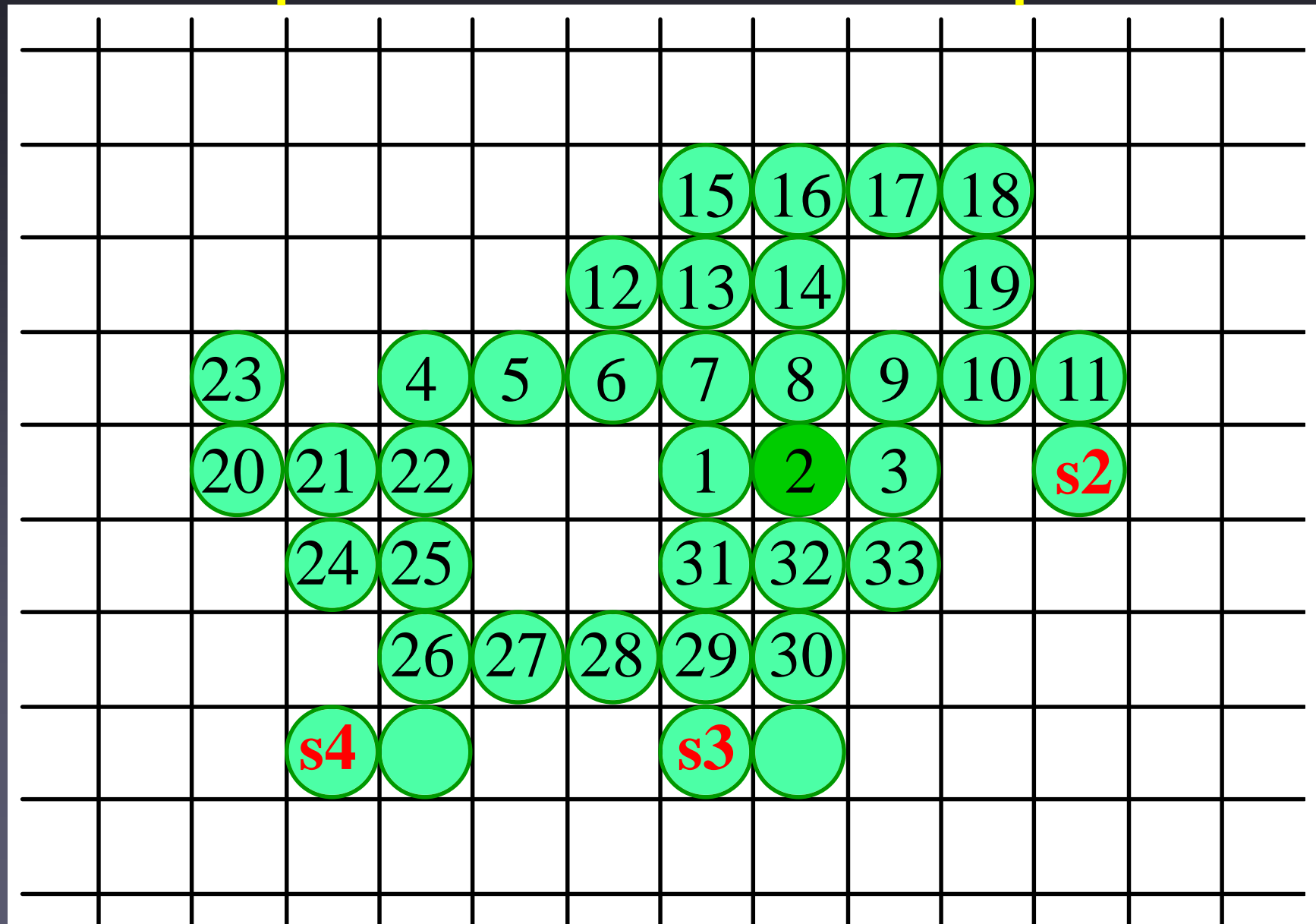
Span Flood-Fill Beispiel



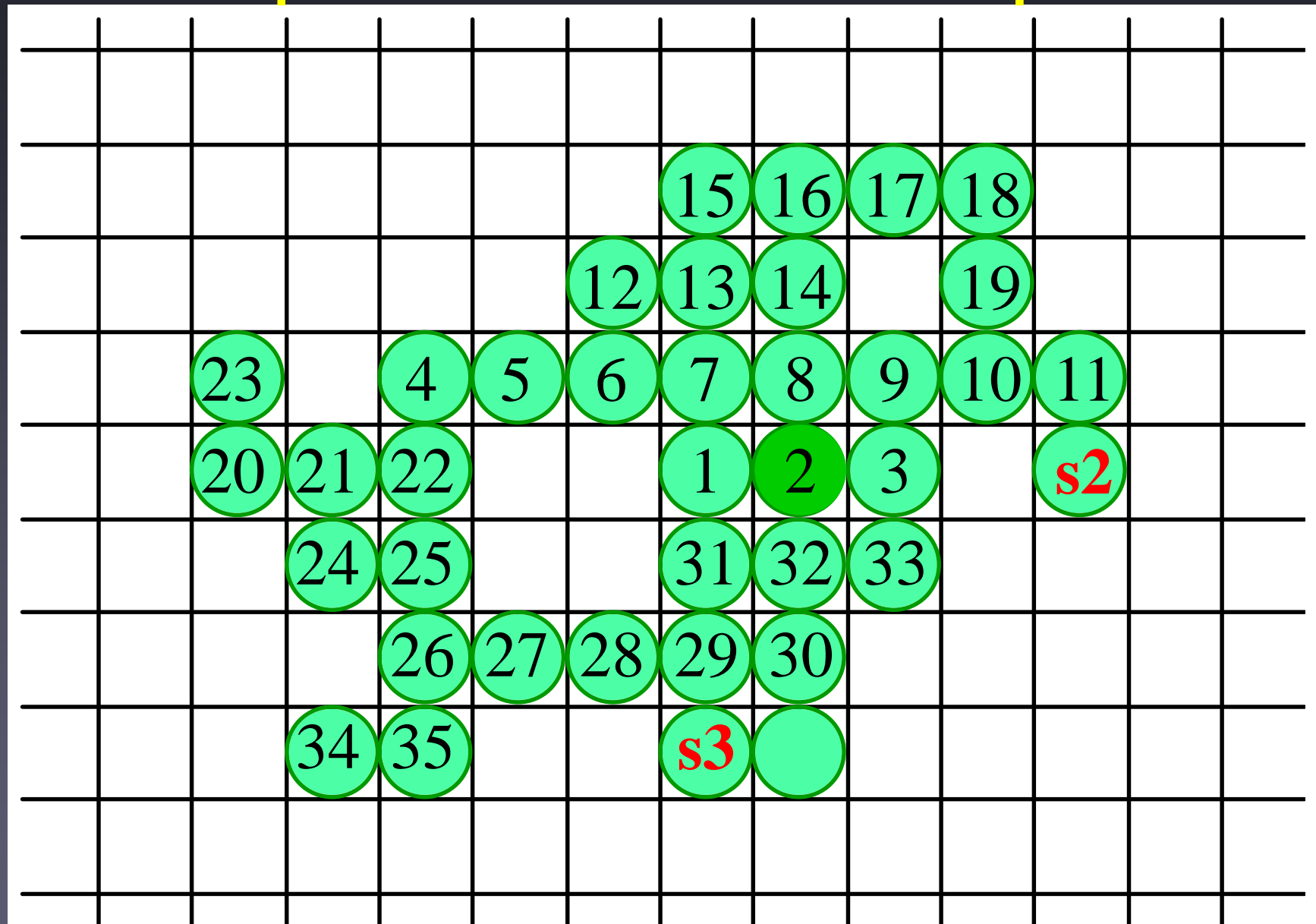
Span Flood-Fill Beispiel



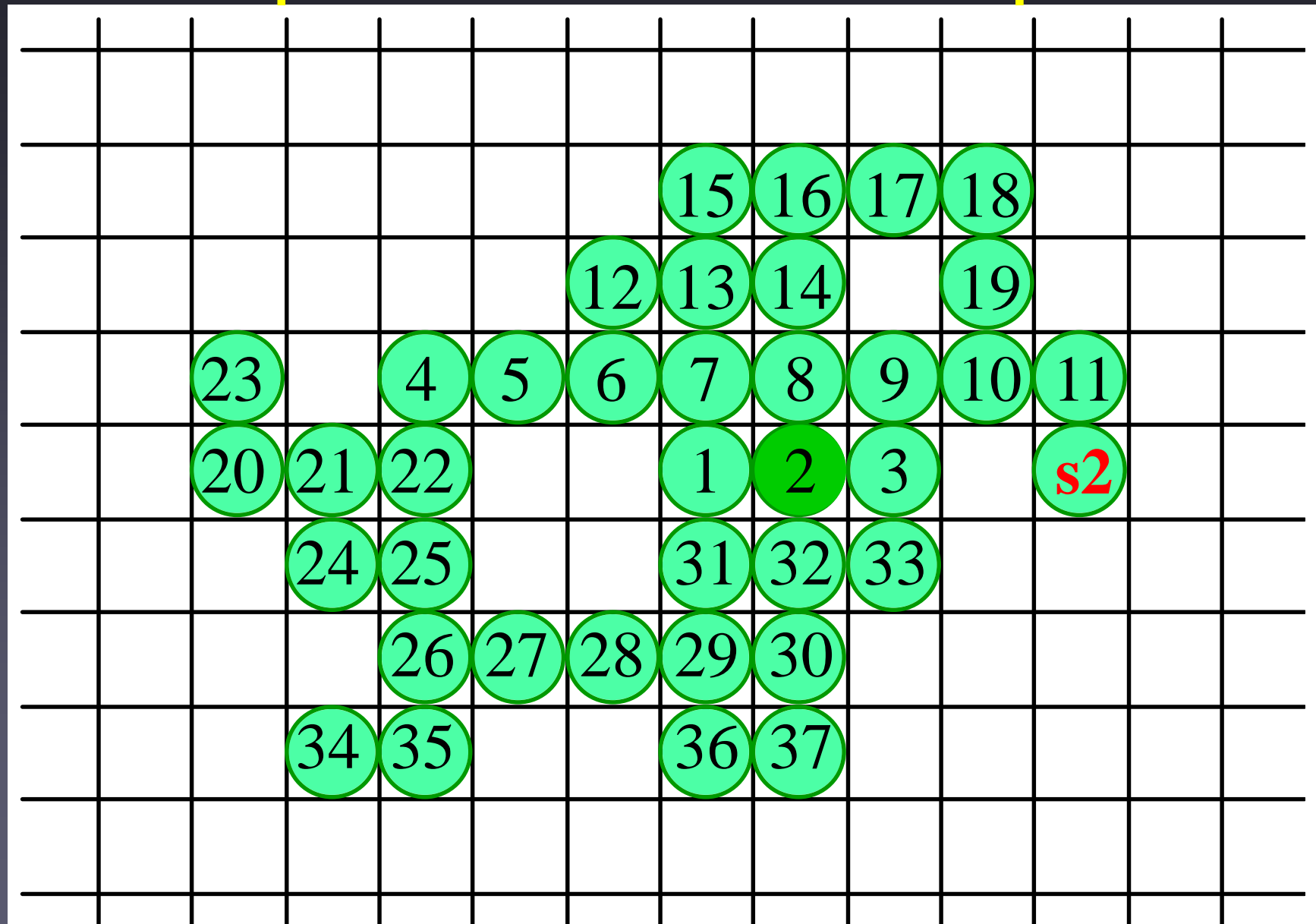
Span Flood-Fill Beispiel



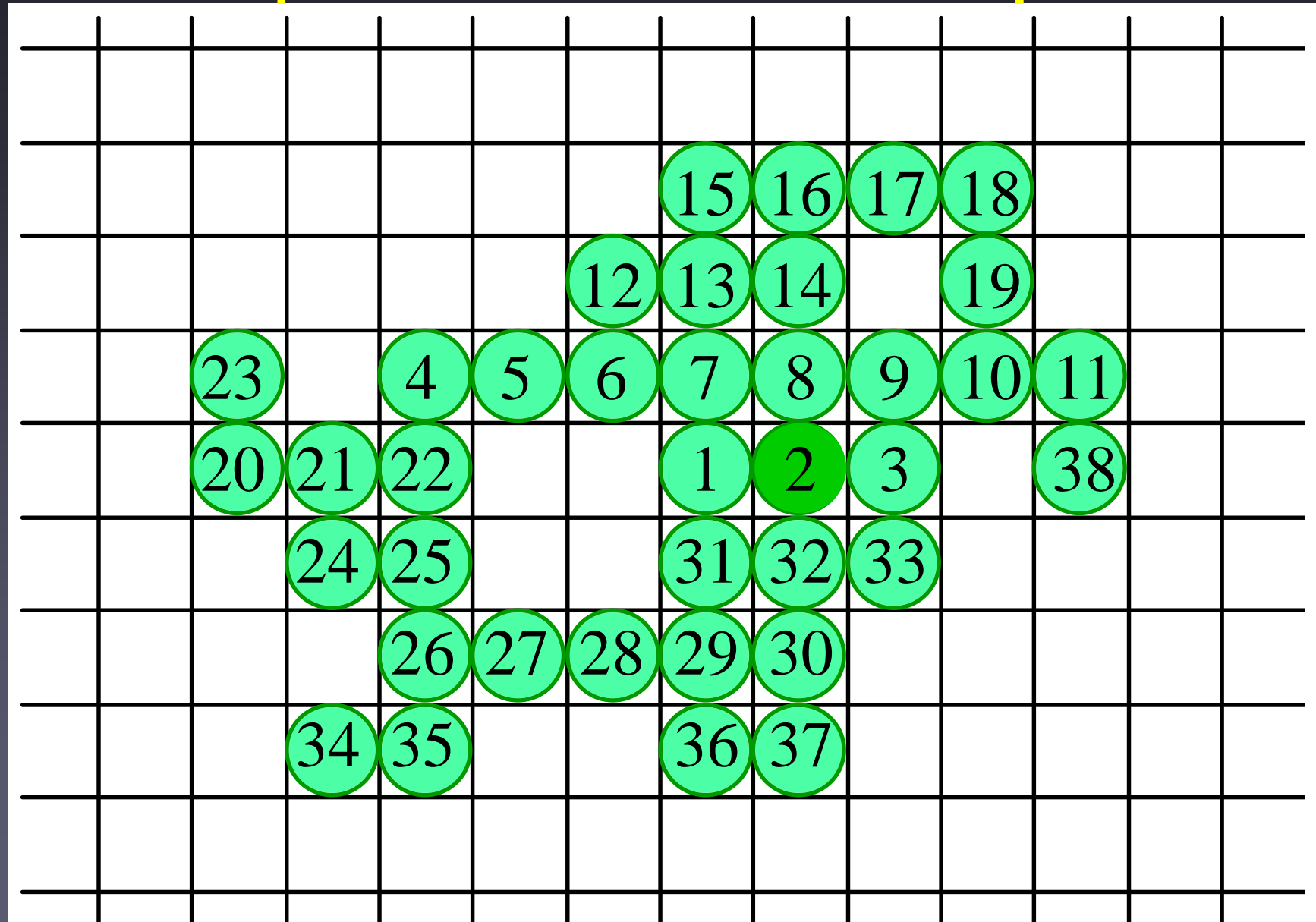
Span Flood-Fill Beispiel



Span Flood-Fill Beispiel



Span Flood-Fill Beispiel



Schriftzeichen-Attribute

■ Text-Attribute

- ◆ Schrift (e.g. Courier, Arial, Times, Roman, ...)
- ◆ Style (regular, **bold**, *italic*, underline,...)
- ◆ Größe (32 point, 1 point = 1/72 inch)
 - Proportionale Größe vs. fixe Abstände

■ Zeichenketten-Attribute

- ◆ Ausrichtung **horizontal**
- ◆ Anordnung (linksbündig, zentriert, rechtsbündig, Blocksatz)

slanted
vertical

angezeigte Primitive wurden mittels des Raster Algorithmus, welcher in Kapitel 3 durchgenommen wurde, erzeugt und haben einen zackiges oder stufenartiges Aussehen

Graphics Output Primitives

angezeigte Primitive wurden mittels des Raster Algorithmus, welcher in Kapitel 3 durchgenommen wurde, erzeugt und haben einen zackiges oder stufenartiges Aussehen.

angezeigte Primitive wurden mittels des Raster Algorithmus, welcher in Kapitel 3 durchgenommen wurde, erzeugt und haben einen zackiges oder stufenartiges Aussehen

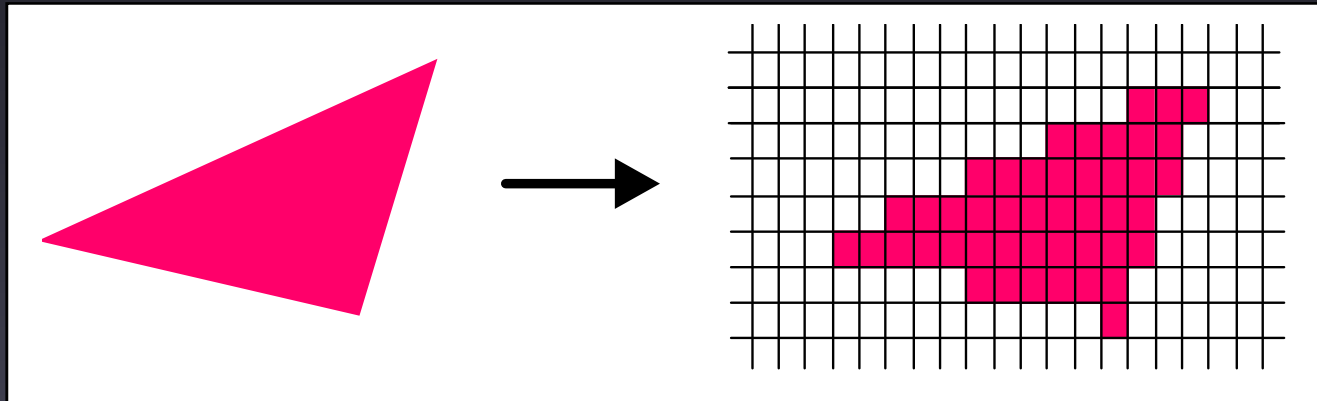
angezeigte Primitive wurden mittels des Raster Algorithmus, welcher in Kapitel 3 durchgenommen wurde, erzeugt und haben einen zackiges oder stufenartiges Aussehen

Dieter Schmalstieg

Antialiasing

- Was ist *aliasing*? ['eiliæsɪŋ]
- Was ist der Grund für *aliasing*?
- Was können wir gegen *aliasing* machen?

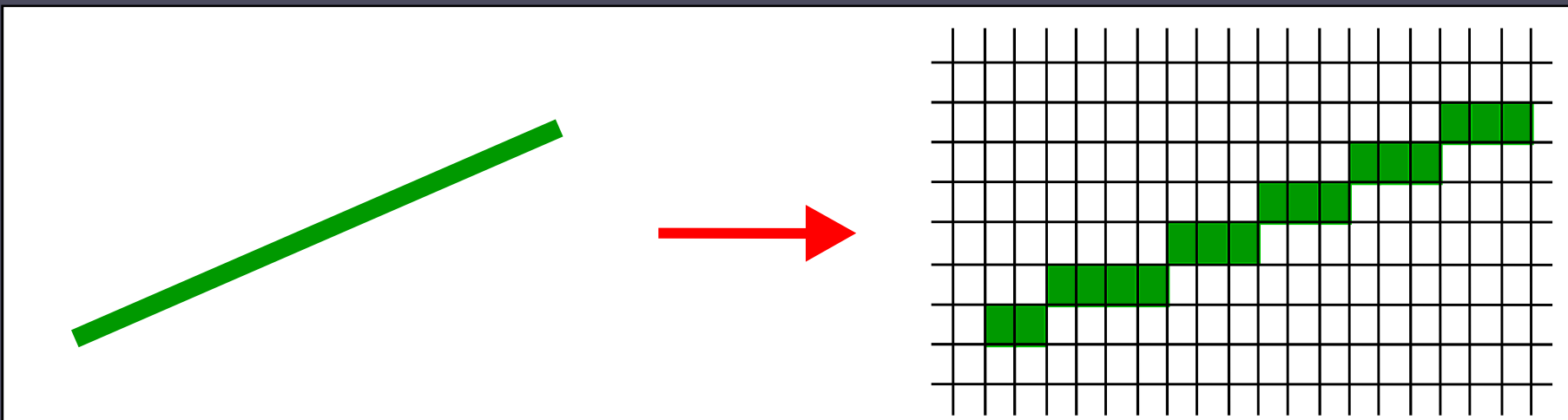
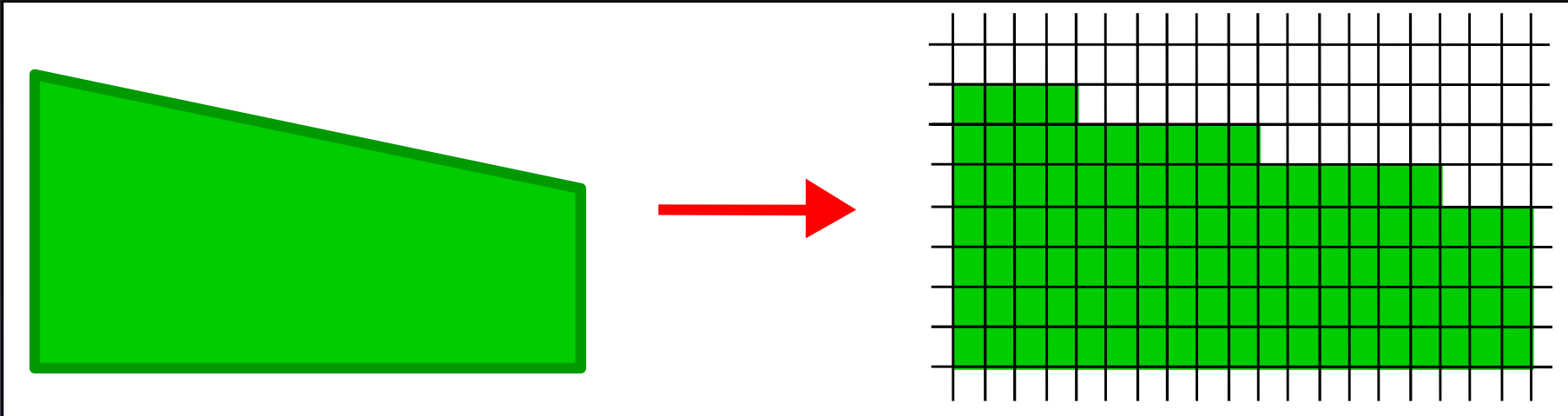
Was ist Aliasing?



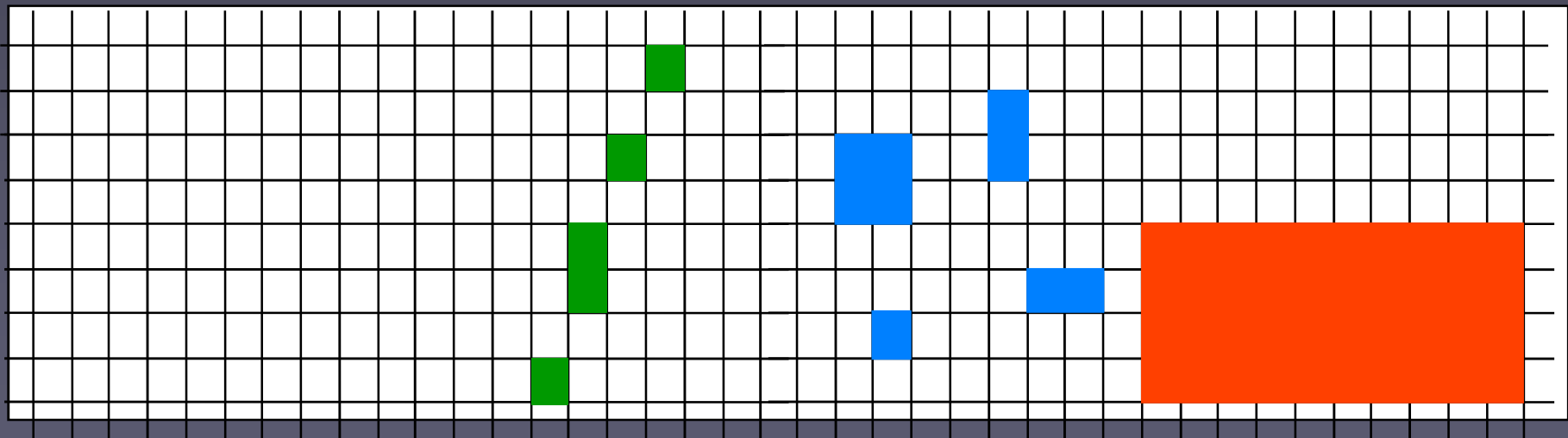
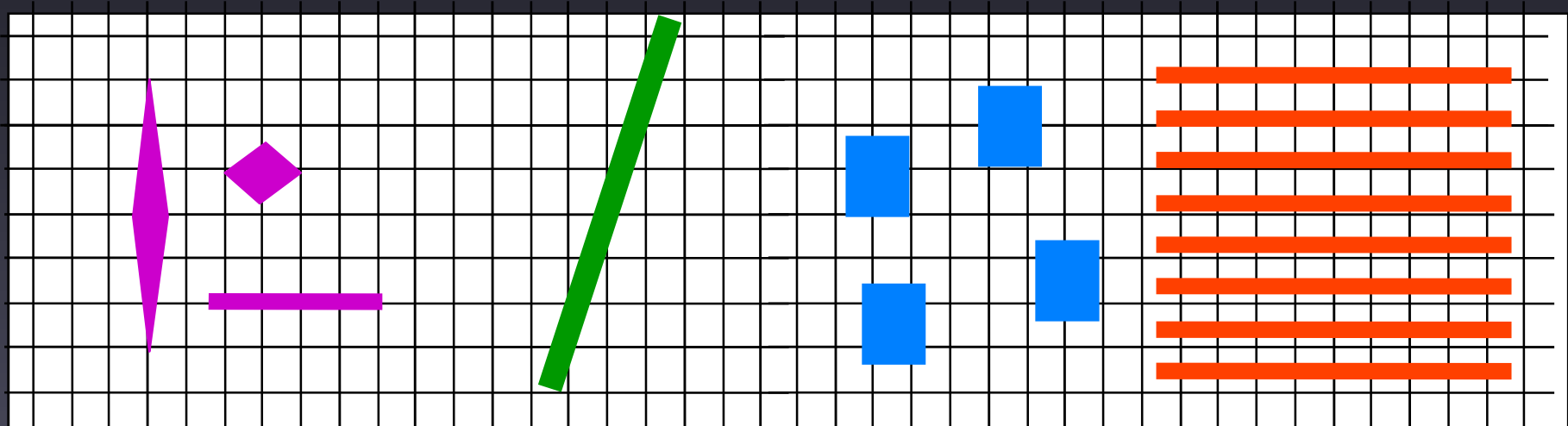
Fehler, welche durch Diskretisierung von analogen zu digitalen Daten verursacht werden

- ◆ ***Zu schlechte Auflösung***
- ◆ ***Zuviele Farben***
- ◆ ***Zuviele Bilder / sek***
- ◆ ***Geometrische Fehler***
- ◆ ***Numerische Fehler***

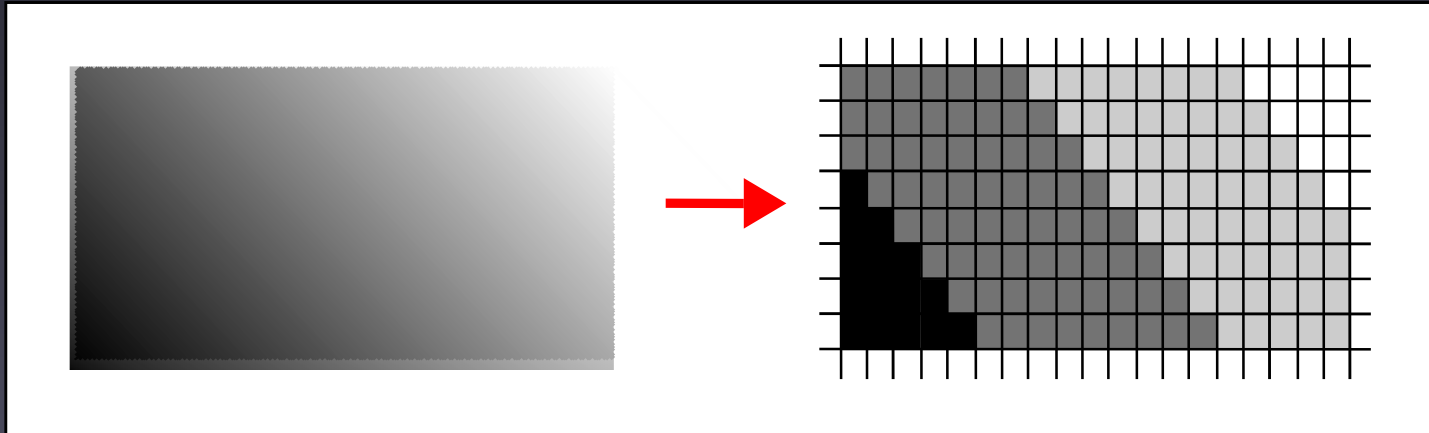
Aliasing: Treppenstufen Effekt



Verschiedene Aliasing Effekte



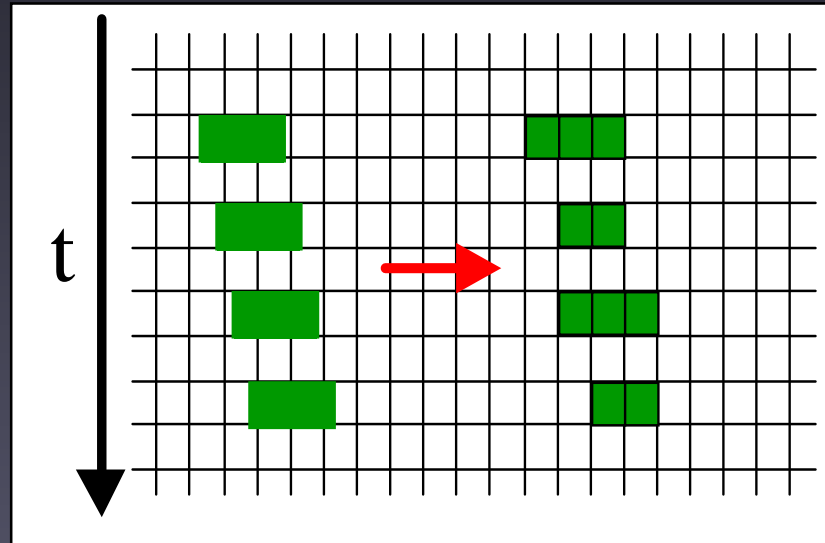
Aliasing aufgrund zuvieler Farben



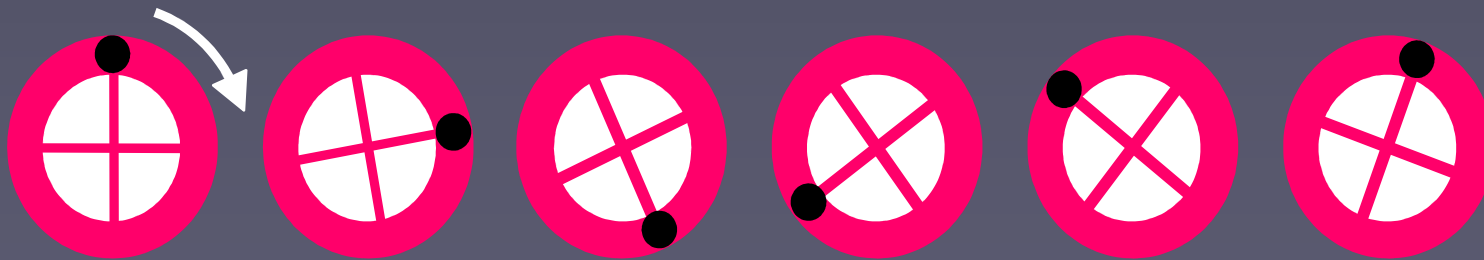
Künstliche Farbränder können auftreten

Aliasing in Animationen

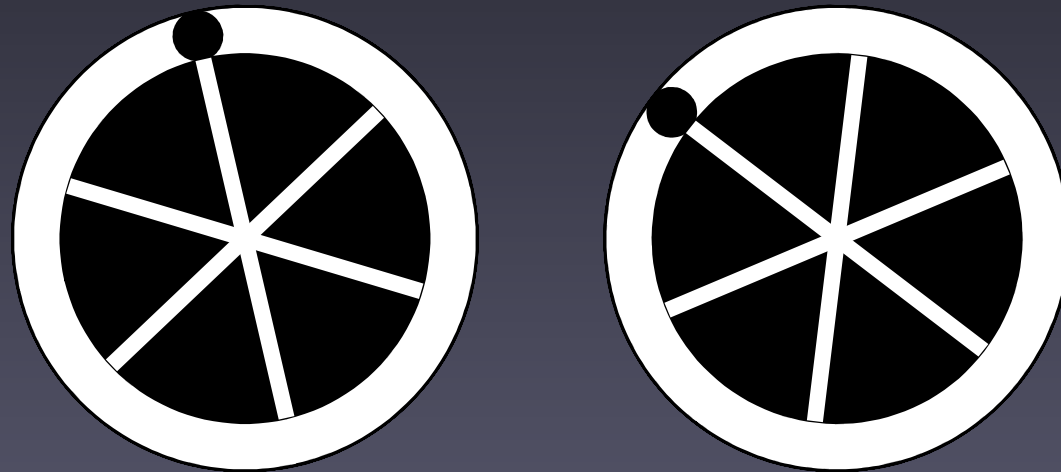
- Hüpfende Bilder
- “Worming“



- Rückwärts rotierende Räder



Rückwärts rotierende Räder



Lösungen gegen Aliasing?

■ 1. Aufrüstung der Geräte

- ◆ Höhere Auflösung
- ◆ Mehr Farblevels
- ◆ Schnellere Bildabfolge

teuer
oder
inkompatibel

■ 2. Aufbesserung der Bilder

- ◆ Nachbearbeitung
- ◆ *Antialiasing !*

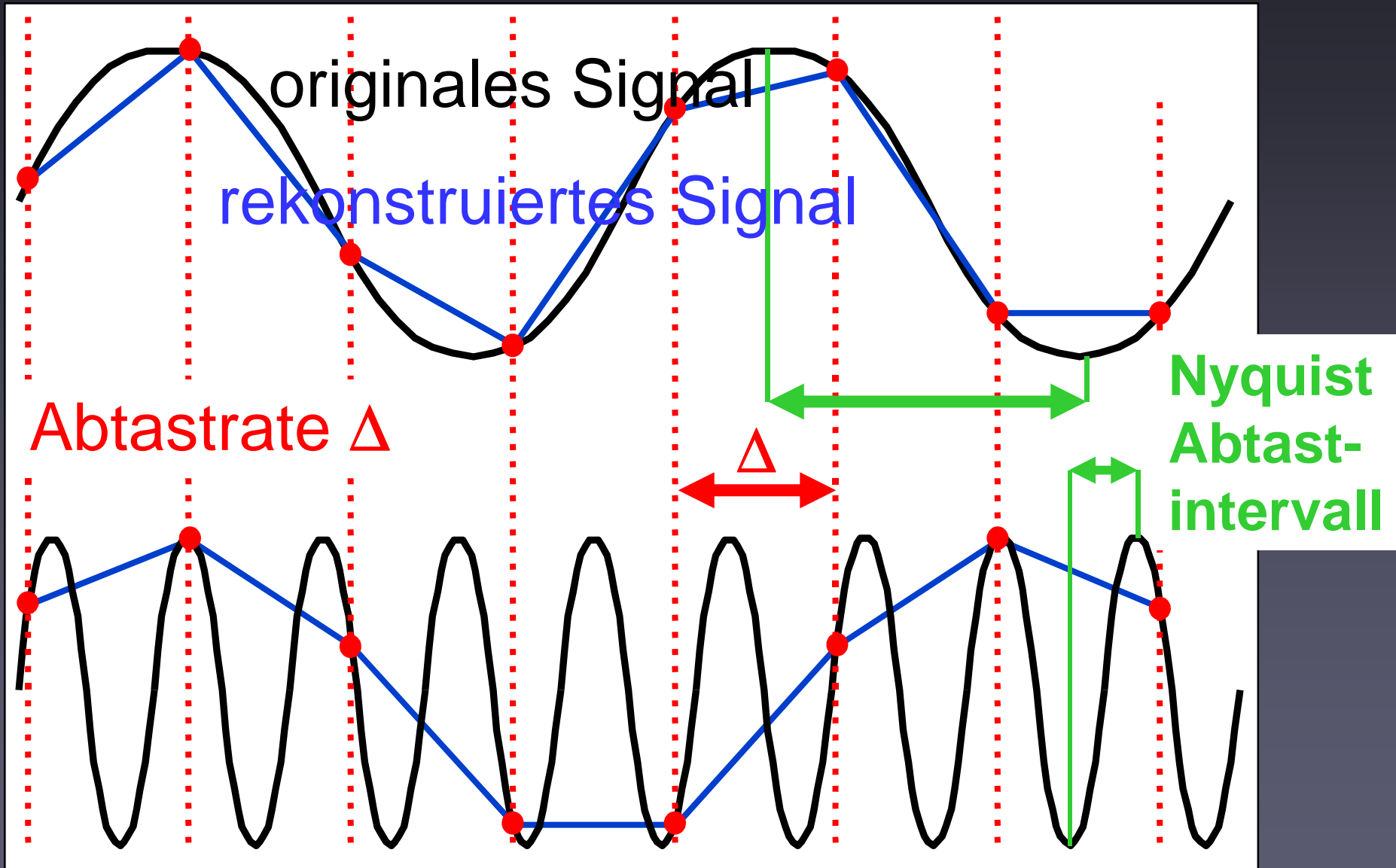
Software

Shannons Abtasttheorem

Ein Signal kann nur ohne Informationsverlust rekonstruiert werden, wenn die abgetastete Frequenz größer als die zweifach höchste Frequenz des Signals ist.

Diese Grenzfrequenz ist das "Nyquist Limit"

Shannon Abtasttheorem



Antialiasing: Nyquist Abtastfrequenz

- Ein Signal kann nur ohne Informationsverlust rekonstruiert werden, wenn die abgetastete Frequenz im Endeffekt die zweifach höchste Frequenz des Signals ist

$$\text{Nyquist Abtastfrequenz: } f_s = 2 f_{\max}$$

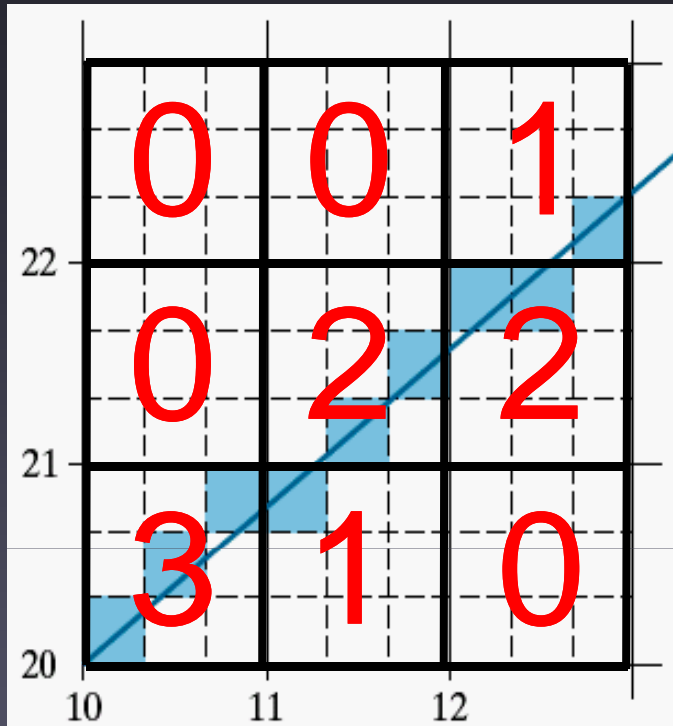
$$\Delta x_s = \frac{\Delta x_{\text{cycle}}}{2} \quad \text{with} \quad \Delta x_{\text{cycle}} = 1 / f_{\max}$$

z.B. Abtastintervall \leq Halbkreisintervall

Antialiasing Strategien

- Extra Abtasten der geraden Liniensegmente
- Subpixel-Bewertungsmaske
- Filtermethoden
- Unterschiede der Linienintensität ausgleichen

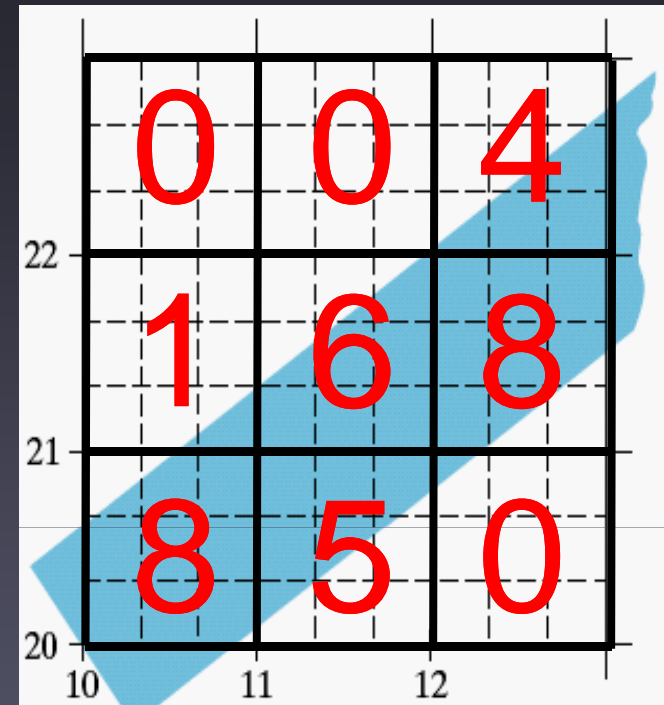
Antialiasing: Extra Abtasten der Linien



Mathematische Linie

3 = max. Intensität

...
0 = min. Intensität



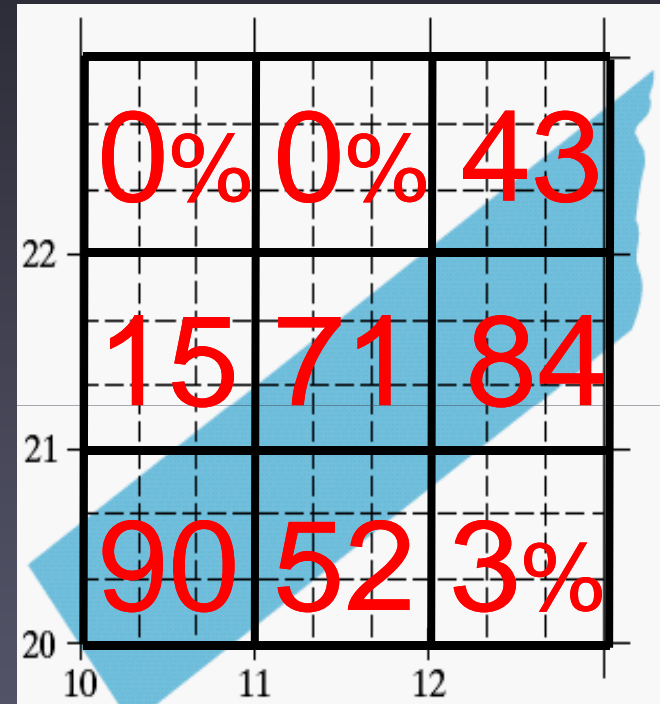
Linie mit begrenzter Breite

9 = max. Intensität

...
0 = min. Intensität

Antialiasing: Abtasten von Linienbereichen

- Berechnet exakt die erfaßten Pixel
- Könnte auch mit inkrementellen Schema behandelt werden



Antialiasing: Pixelbewertungsmaske

1	2	1
2	4	2
1	2	1

Relative Gewichte für
einen Raster von 3x3
Subpixels

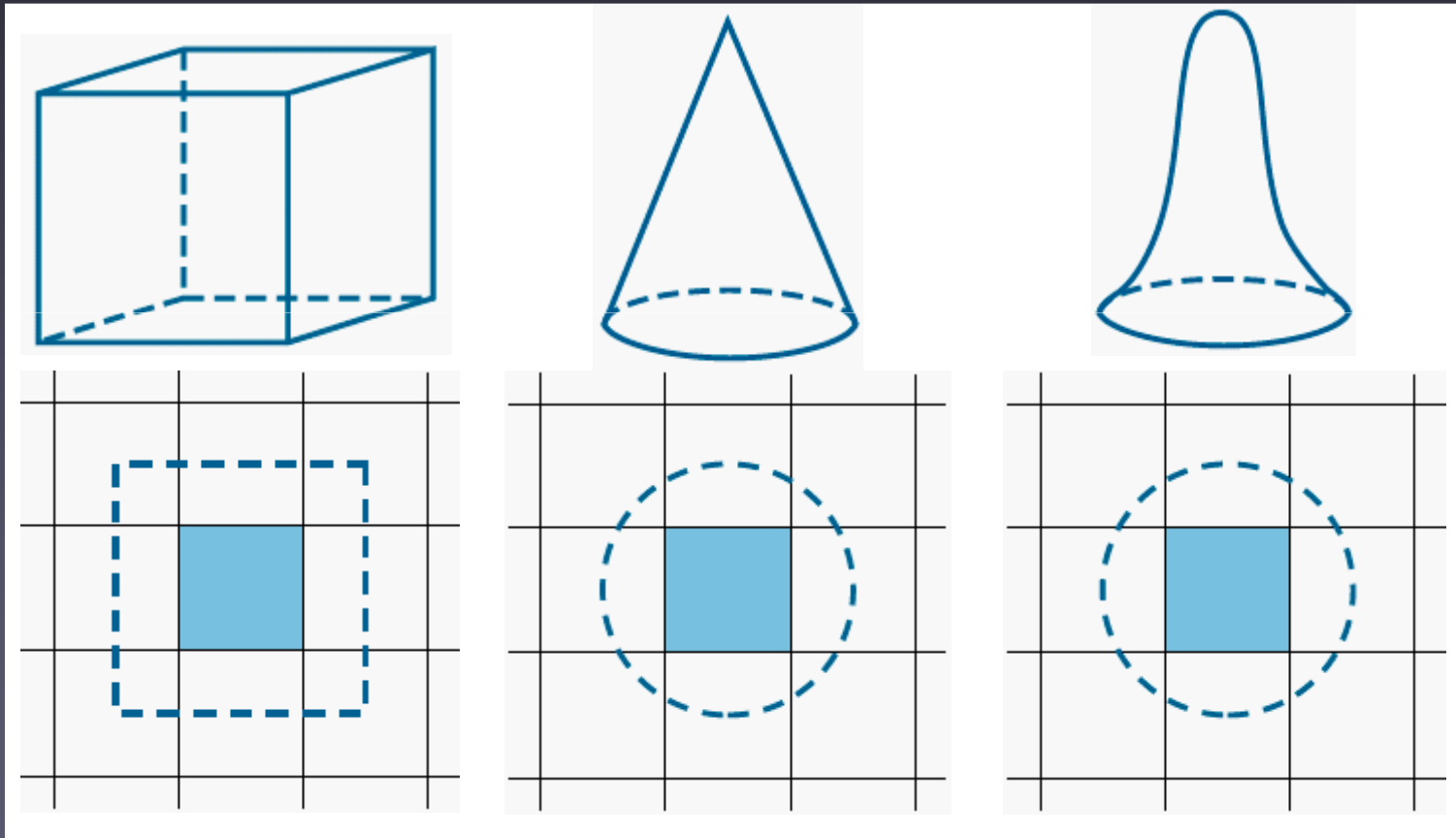
Graphics Output Primitives

- Mehr Gewicht für mittige Subpixels
- Muss durch die Summe der Gewichte dividiert werden
- Subpixel-Raster kann auch einige Nachbarpixel beinhalten

Dieter Schmalstieg

Antialiasing: Filtertechniken

Kontinuierliche überlappende Gewichtsfunktionen, um Antialiasing-Werte mittels Integral zu berechnen

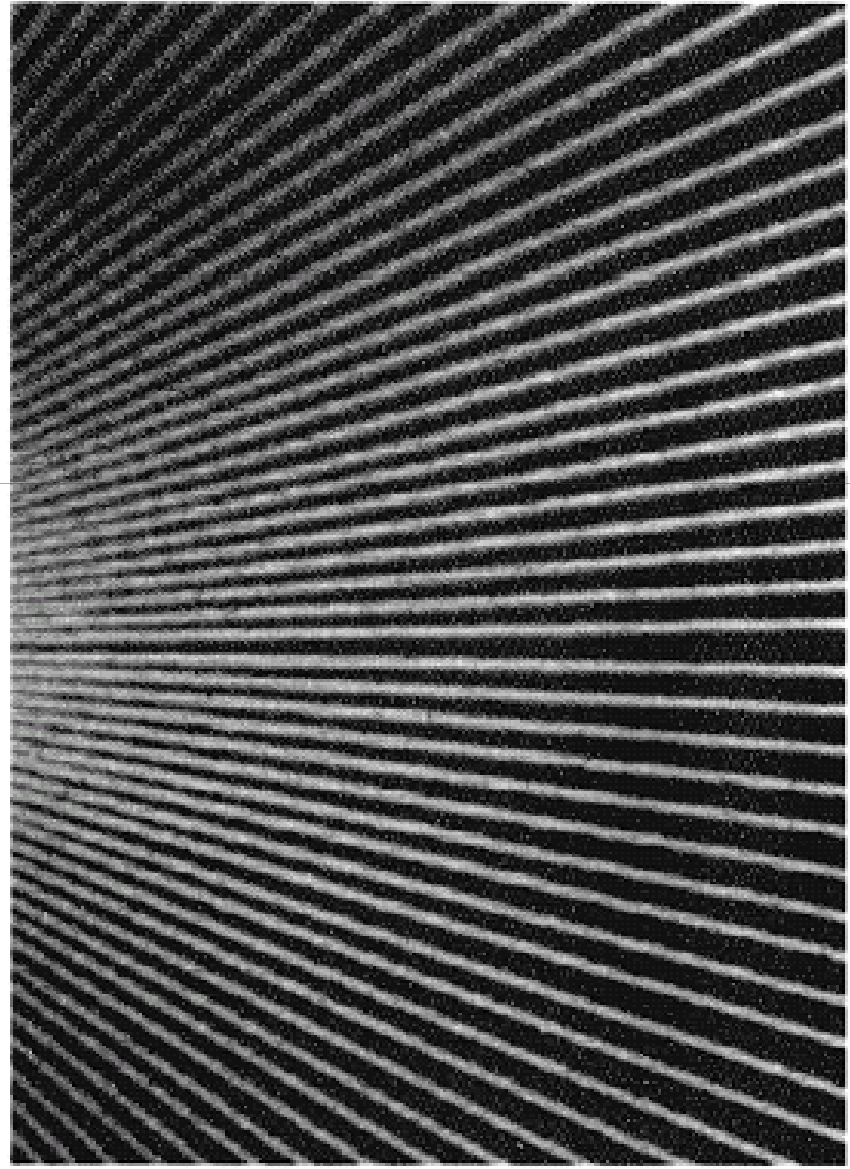
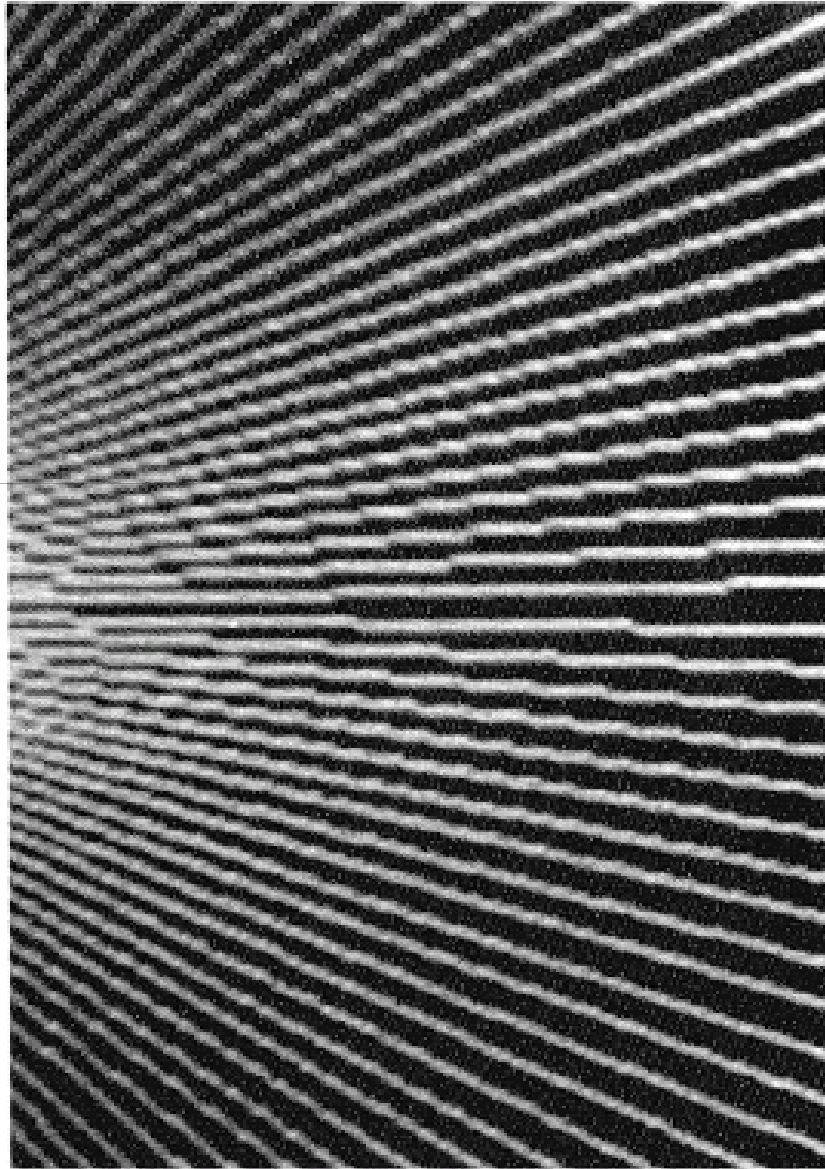


Boxfilter

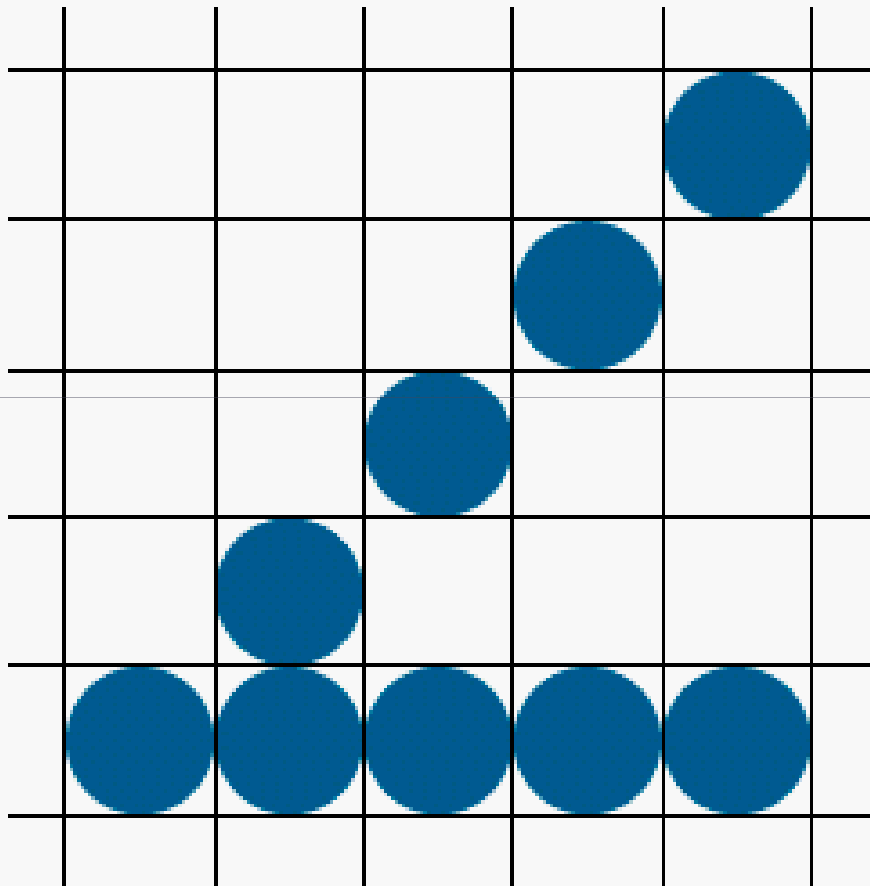
Kegelfilter

Gaußfilter

Antialiasing

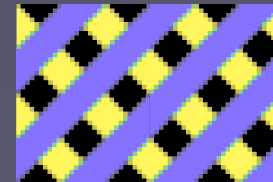
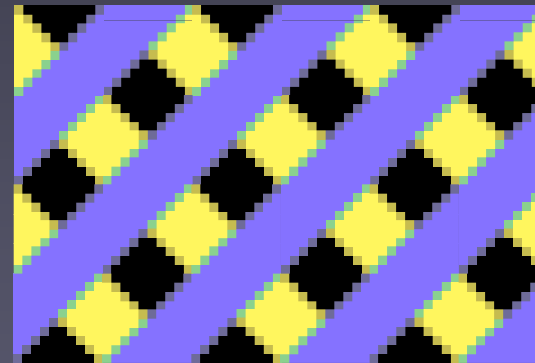
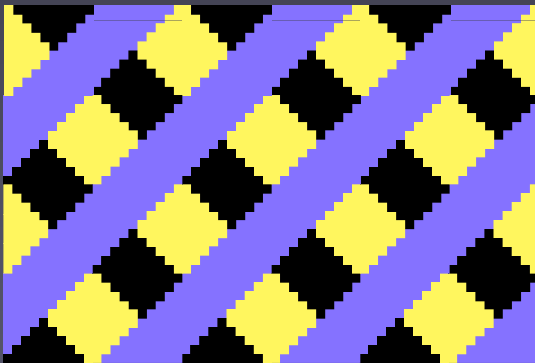
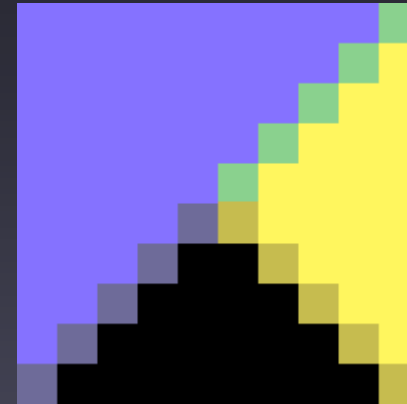
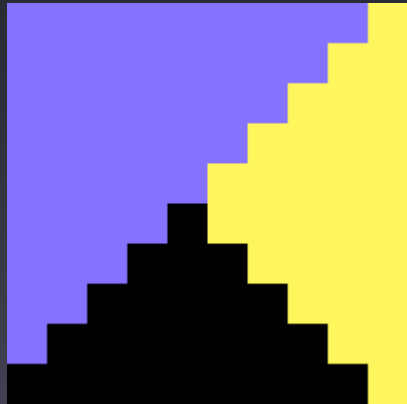


Antialiasing: Intensitätsunterschiede ausgleichen



- Ungleiche angezeigte
Linienlängen mit
gleicher Anzahl an
Pixel in jeder
Line/Reihe haben
unterschiedliche
Intensitäten
- Geeignetes
Antialiasing gleicht ab!

Antialiasing

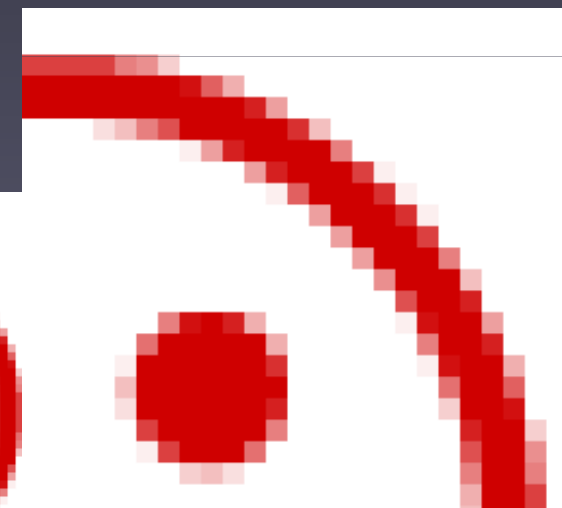
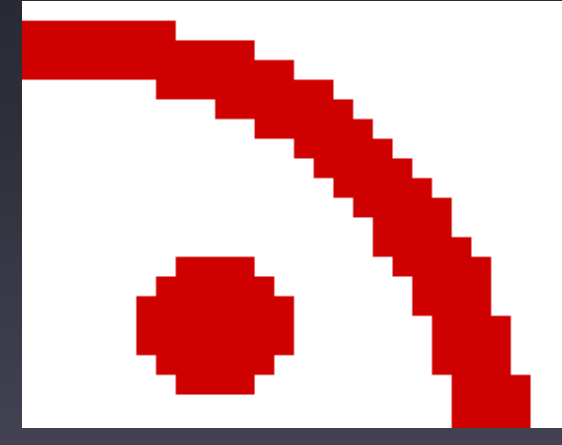


Antialiasing - Beispiele

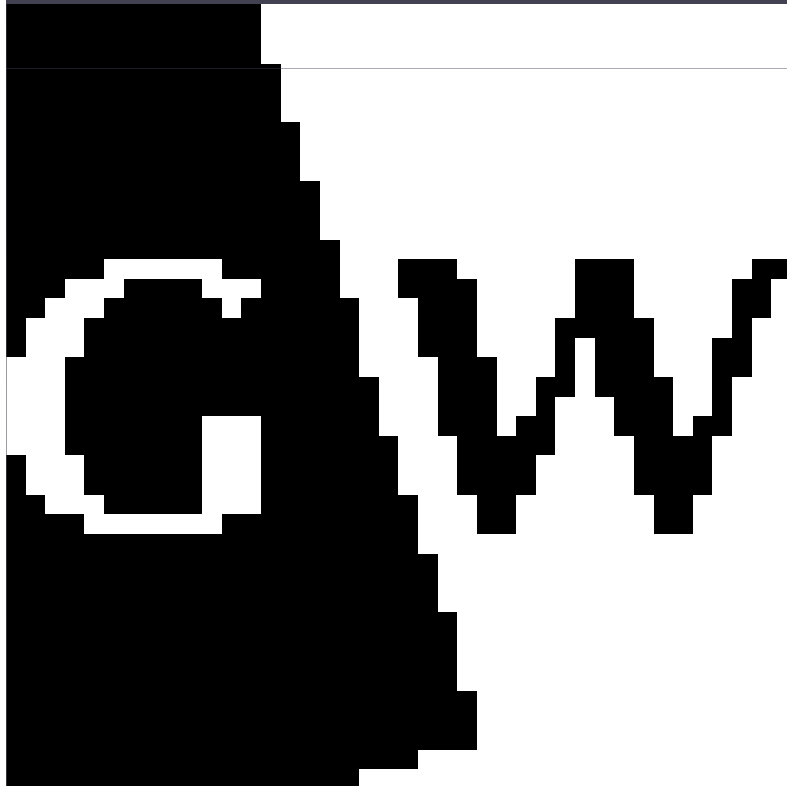
 
aliased antialiased

 
aliased antialiased

 
aliased antialiased



Antialiasing - Beispiele

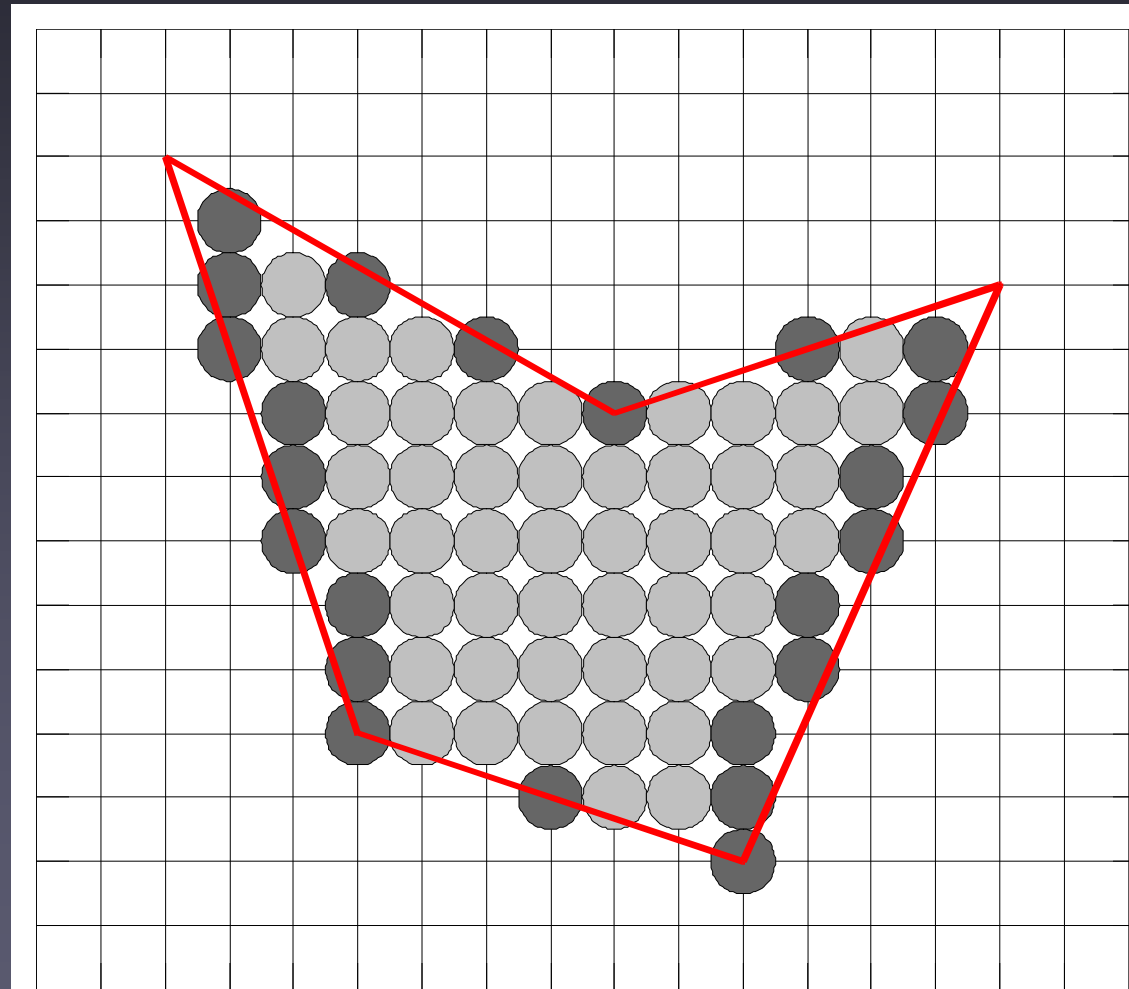


Zusammenfassung: Attribute der Primitive

- Farb- und Grauskalierung
- Linienattribute
- Füllbereich Attribute
- Zeichen Attribute
- Antialiasing
 - ◆ Aliasingeffekte
 - ◆ Gründe für Aliasing
 - ◆ Antialiasing Strategien

Füllen von Polygonen

- Midpoint line: benachbarte Flächen haben gleiche Kanten
- Nur Pixel innerhalb der Fläche zeichnen
- Pixel genau auf Kante: Füllkonvention



Modifizierter Bresenham-Alg.

- Entscheidungsvariable mit 0 initialisieren
- Faktor „2“ kann entfallen
- Kriterien für Wahl des nächsten Pixels (selbst überlegen → Übung)
- Schleife immer über y
- In x auch Schritte >1 Pixel möglich