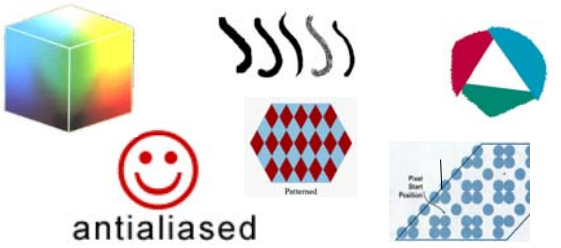
Attributes of Graphics Primitives

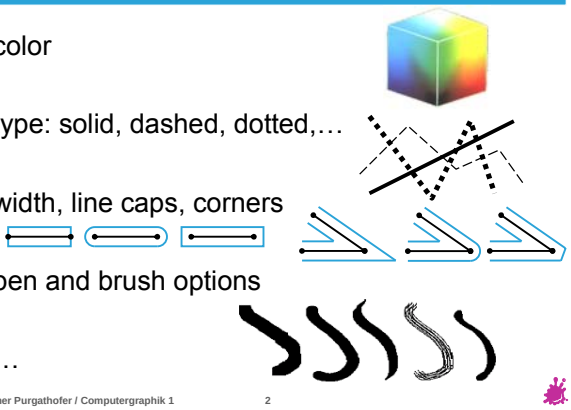


Attributes for Graphics Output Primitives

- in 2D
  - points, lines
  - polygons, circles, ellipses & other curves (also filled)
  - characters
- in 3D
  - triangles & other polygons
- anti-aliasing

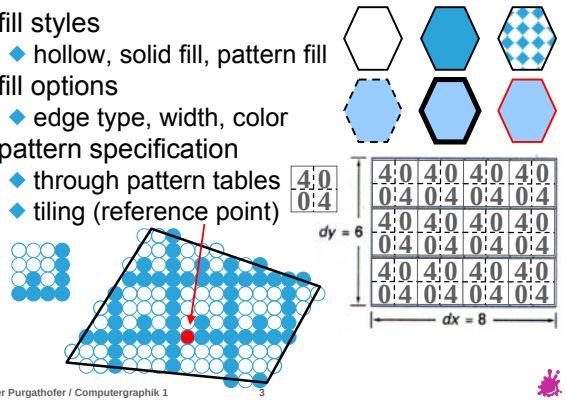
Points and Line Attributes

- color
- type: solid, dashed, dotted,…
- width, line caps, corners
- pen and brush options
- …



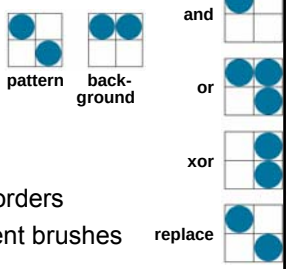
Area-Fill Attributes (1)

- fill styles
  - hollow, solid fill, pattern fill
- fill options
  - edge type, width, color
- pattern specification
  - through pattern tables
  - tiling (reference point)



Area-Fill Attributes (2)

- combination of fill pattern with background colors
- soft fill
  - combination of colors
  - antialiasing at object borders
  - simul. of semitransparent brushes
  - example: linear soft-fill

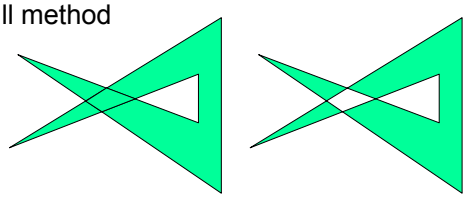


F...foreground color
B...background color

$$P = tF + (1 - t)B$$

Polygon Fill Algorithms

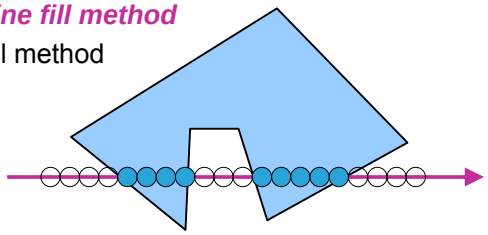
- ***what is inside?***
- scan-line fill method
- flood fill method



"interior", "exterior" for self-intersecting polygons?

Polygon Fill Algorithms

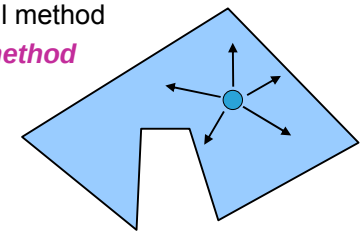
- what is inside?
- ***scan-line fill method***
- flood fill method



interior pixels along a scan line passing through a polygon area

Polygon Fill Algorithms

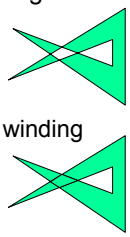
- what is inside?
- scan-line fill method
- ***flood fill method***



starting from a seed point fill until you reach a border

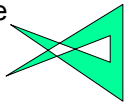
Inside-Outside Tests

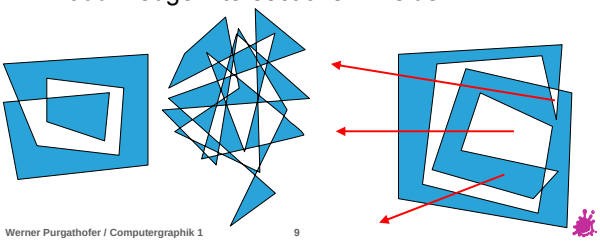
- area-filling algorithms
  - "interior", "exterior" for self-intersecting polygons?
  - odd-even rule
  - nonzero winding number rule
    - cross or dot product to determine winding number
  - same result for simple polygons



What is Inside?: Odd-Even Rule

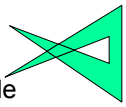
- inside/outside switches at every edge
- straight line to the outside:
  - even # edge intersections = outside
  - odd # edge intersections = inside

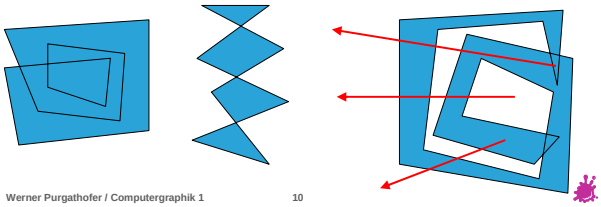




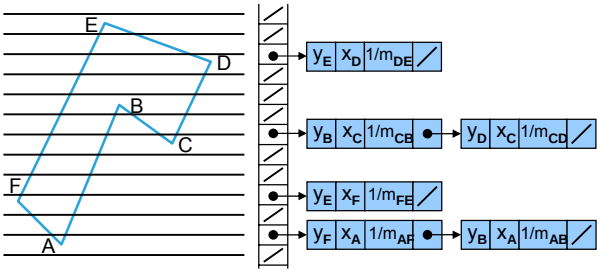
What is Inside?: Nonzero Winding Number

- point is inside if polygon surrounds it
- straight line to the outside:
  - same # edges up and down = outside
  - different # edges up and down = inside





Scan-Line Fill: Sorted Edge Table

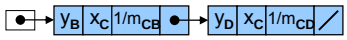


The sorted edge table contains all polygon edges sorted by lowest y-value

y_{max} x_{start} $1/m$

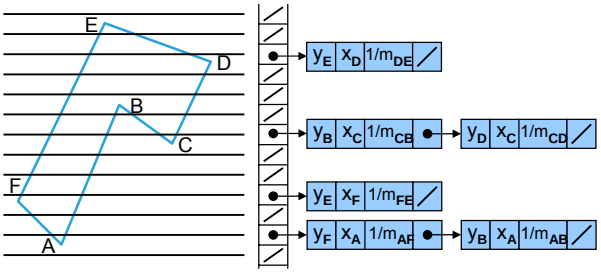
Scan-Line Fill: Sorted Edge Table

- sort all edges on smallest y-value
- edge entry:
  [max y-value, x-intercept, inverse slope]



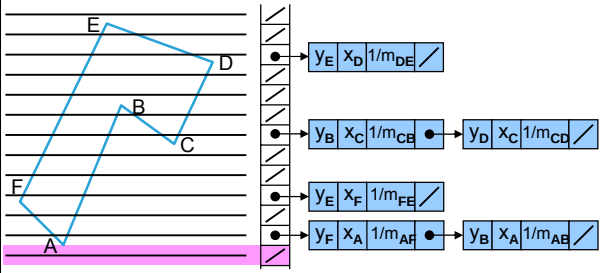
- ***active-edge list***
  - for each scan line
  - contains all edges crossed by that scan line
  - incremental update
- consecutive intersection pairs (spans) filled

Sorted Edge Table



The sorted edge table contains all polygon edges sorted by lowest y-value

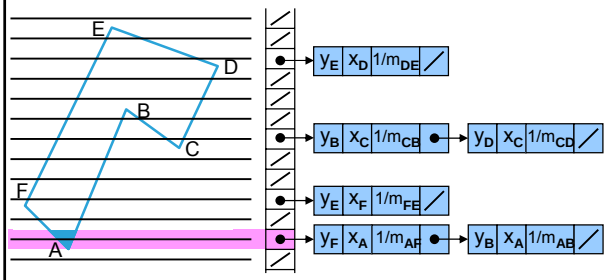
Sorted Edge Table / Active Edge List



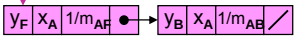
Active Edge List

When processing from bottom to top, keep a list of all active edges

Sorted Edge Table / Active Edge List

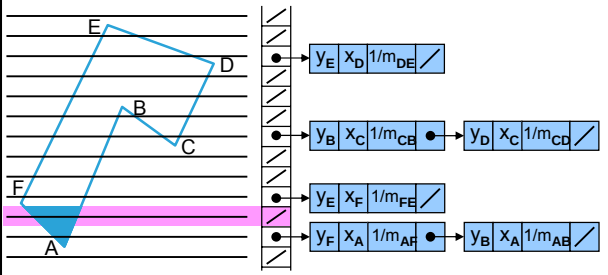


Active Edge List

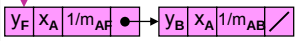


incremental update!

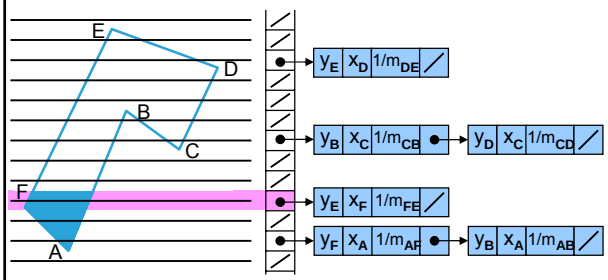
Sorted Edge Table / Active Edge List



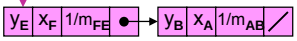
Active Edge List

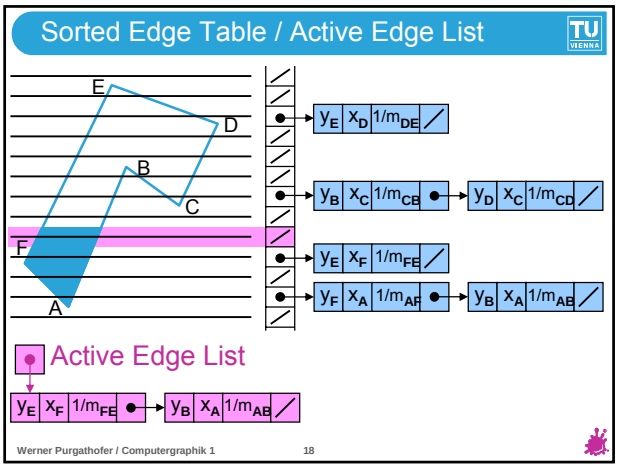


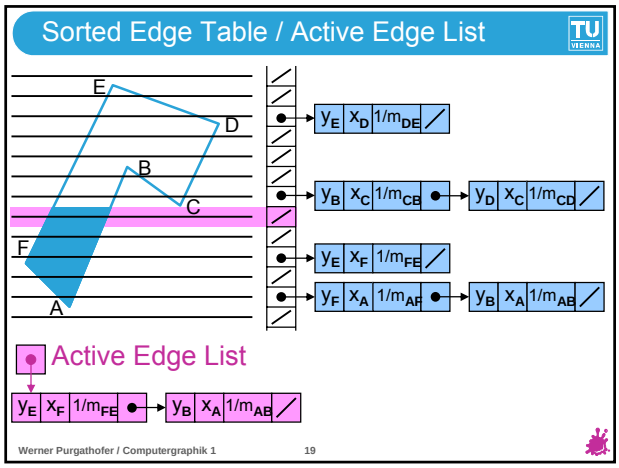
Sorted Edge Table / Active Edge List

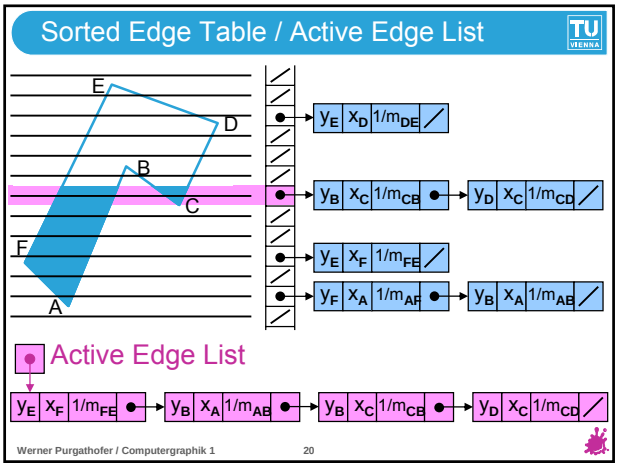


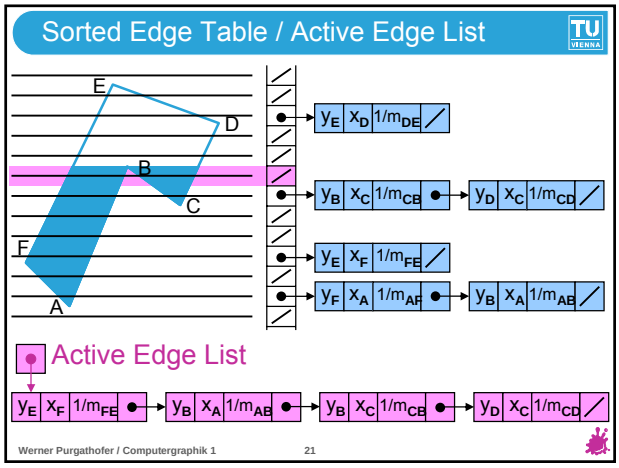
Active Edge List

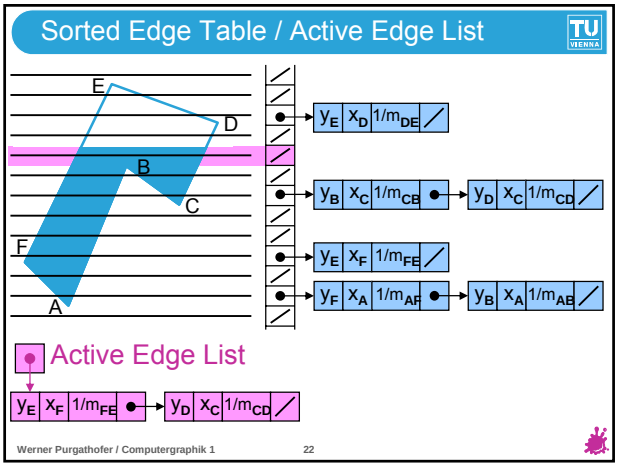


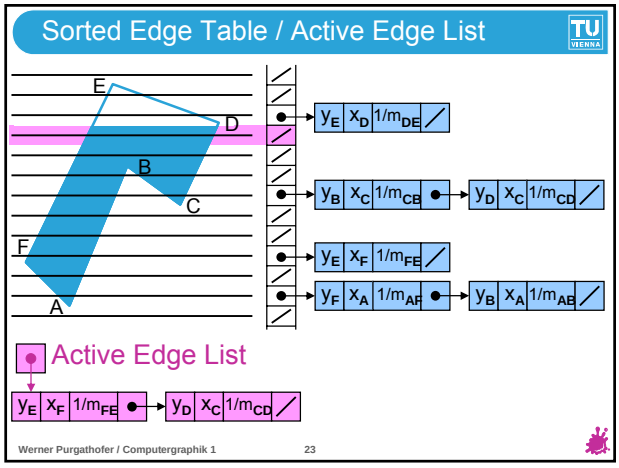

Sorted Edge Table / Active Edge List
E
D
B
C
F
A
y_E x_D $1/m_{DE}$
y_B x_C $1/m_{CB}$
y_D x_C $1/m_{CD}$
y_E x_F $1/m_{FE}$
y_F x_A $1/m_{AF}$
y_B x_A $1/m_{AB}$
Active Edge List
y_E x_F $1/m_{FE}$
y_B x_A $1/m_{AB}$
Werner Purgathofer / Computergraphik 1
18


Sorted Edge Table / Active Edge List
E
D
B
C
F
A
y_E x_D $1/m_{DE}$
y_B x_C $1/m_{CB}$
y_D x_C $1/m_{CD}$
y_E x_F $1/m_{FE}$
y_F x_A $1/m_{AF}$
y_B x_A $1/m_{AB}$
Active Edge List
y_E x_F $1/m_{FE}$
y_B x_A $1/m_{AB}$
Werner Purgathofer / Computergraphik 1
19

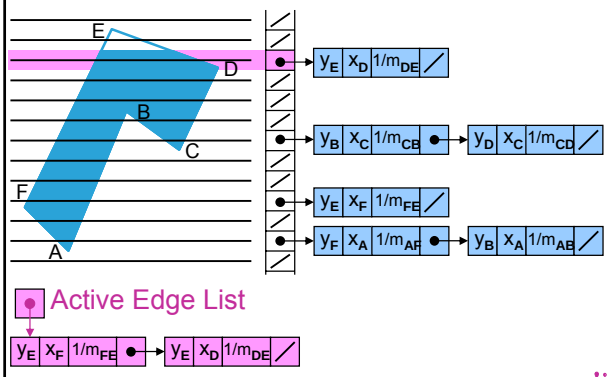

Sorted Edge Table / Active Edge List
E
D
B
C
F
A
y_E x_D $1/m_{DE}$
y_B x_C $1/m_{CB}$
y_D x_C $1/m_{CD}$
y_E x_F $1/m_{FE}$
y_F x_A $1/m_{AF}$
y_B x_A $1/m_{AB}$
Active Edge List
y_E x_F $1/m_{FE}$
y_B x_A $1/m_{AB}$
y_B x_C $1/m_{CB}$
y_D x_C $1/m_{CD}$
Werner Purgathofer / Computergraphik 1
20


Sorted Edge Table / Active Edge List
E
D
B
C
F
A
y_E x_D $1/m_{DE}$
y_B x_C $1/m_{CB}$
y_D x_C $1/m_{CD}$
y_E x_F $1/m_{FE}$
y_F x_A $1/m_{AF}$
y_B x_A $1/m_{AB}$
Active Edge List
y_E x_F $1/m_{FE}$
y_B x_A $1/m_{AB}$
y_B x_C $1/m_{CB}$
y_D x_C $1/m_{CD}$
Werner Purgathofer / Computergraphik 1
21

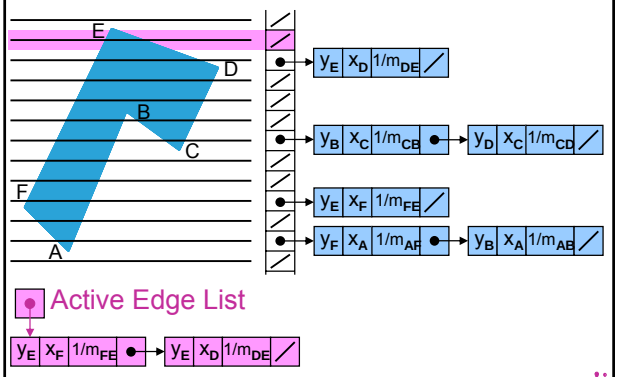

Sorted Edge Table / Active Edge List
E
D
B
C
F
A
y_E x_D $1/m_{DE}$
y_B x_C $1/m_{CB}$
y_D x_C $1/m_{CD}$
y_E x_F $1/m_{FE}$
y_F x_A $1/m_{AF}$
y_B x_A $1/m_{AB}$
Active Edge List
y_E x_F $1/m_{FE}$
y_D x_C $1/m_{CD}$
Werner Purgathofer / Computergraphik 1
22


Sorted Edge Table / Active Edge List
E
D
B
C
F
A
y_E x_D $1/m_{DE}$
y_B x_C $1/m_{CB}$
y_D x_C $1/m_{CD}$
y_E x_F $1/m_{FE}$
y_F x_A $1/m_{AF}$
y_B x_A $1/m_{AB}$
Active Edge List
y_E x_F $1/m_{FE}$
y_D x_C $1/m_{CD}$
Werner Purgathofer / Computergraphik 1
23

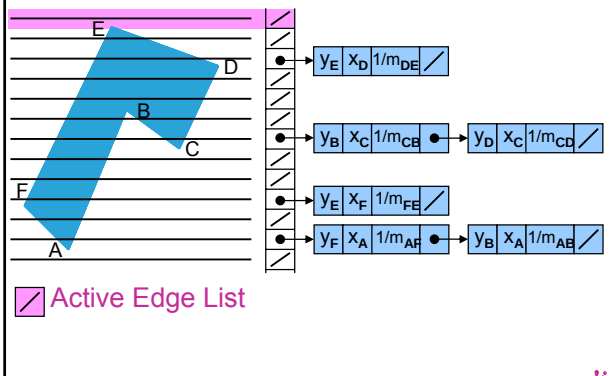
Sorted Edge Table / Active Edge List



Sorted Edge Table / Active Edge List



Sorted Edge Table / Active Edge List



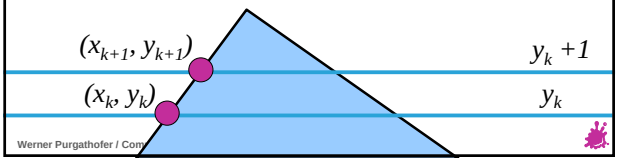
Scan-Line Fill: Incremental Update

- incremental update of intersection point

slope of polygon boundary line: m

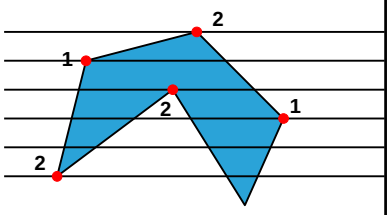
$$x_{k+1} = x_k + \frac{1}{m} \qquad y_{k+1} = y_k + 1$$

(for 2 successive scanlines)



Scan-Line Fill: Intersecting Vertices

intersection points along scan lines that intersect polygon vertices.



→ either special handling (1 or 2 intersections?)

→ or move vertices up or down by ε

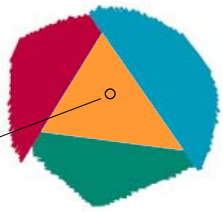
Flood-Fill Algorithm

- pixel filling of area
  - areas with no single color boundary
  - start from interior point
  - "flood" internal region
  - 4-connected, 8-connected areas
  - reduce stack size by eliminating several recursive calls

Flood-Fill: Boundary and Seed Point

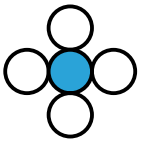
area must be distinguishable from boundaries

seed point

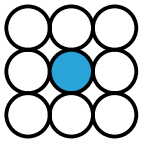


example: area defined within multiple color boundaries

Flood-Fill: Connectedness

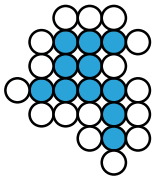


Definition:
4-connected means, that a connection is only valid in these 4 directions

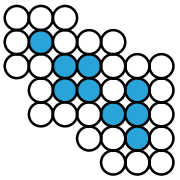


Definition:
8-connected means, that a connection is valid in these 8 directions

Examples for 4- and 8-connected



a 4-connected area has an 8-connected border

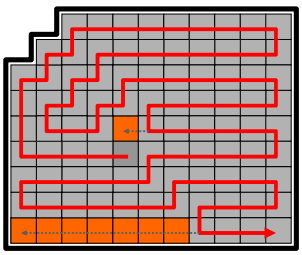


an 8-connected area has a 4-connected border

Simple Flood-Fill Algorithm

```
void floodFill4 (int x, int y, int new, int old)
{ int color;
   /* set current color to new */
   getPixel (x, y, color);
   if (color = old) {
      setPixel (x, y);
      floodFill4 (x–1, y, new, old);  /* left */
      floodFill4 (x, y+1, new, old);  /* up */
      floodFill4 (x+1, y, new, old);  /* right */
      floodFill4 (x, y–1, new, old)   /* down */
}}
```


Bad Behavior of Simple Flood-Fill

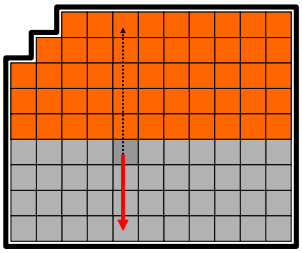


recursion sequence

Span Flood-Fill Algorithm

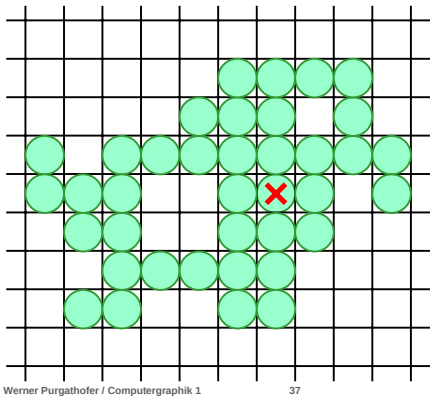
- floodFill4 produces too high stacks (recursion!)
- solution:
  - incremental horizontal fill (left to right)
  - recursive vertical fill (first up then down)

Good Behavior of Span Flood-Fill



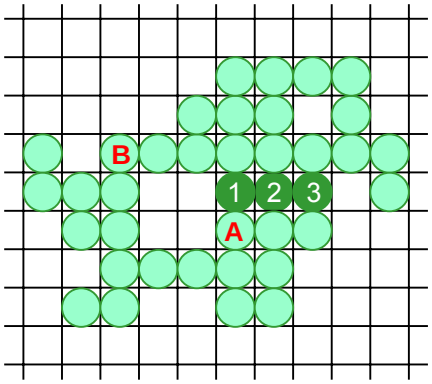
recursion sequence

Span Flood-Fill Example



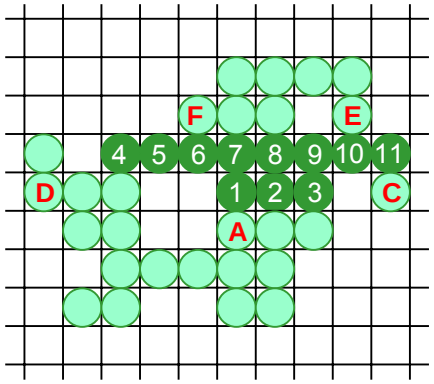
Stack:
x

Span Flood-Fill Example



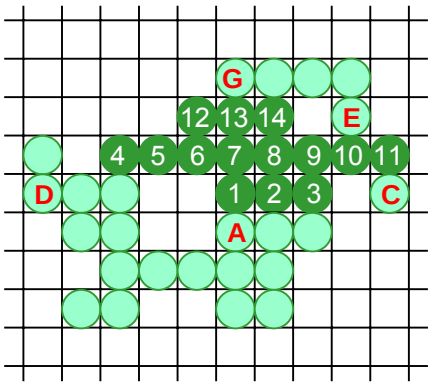
Stack:
~~x~~
A
B

Span Flood-Fill Example



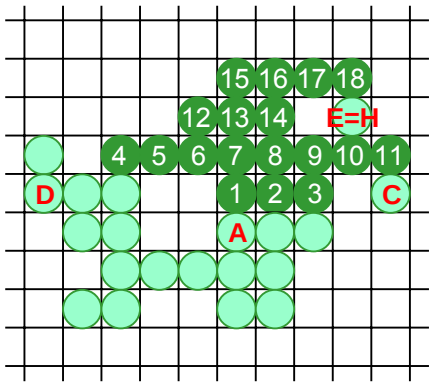
Stack:
A
~~B~~
C
D
E
F

Span Flood-Fill Example

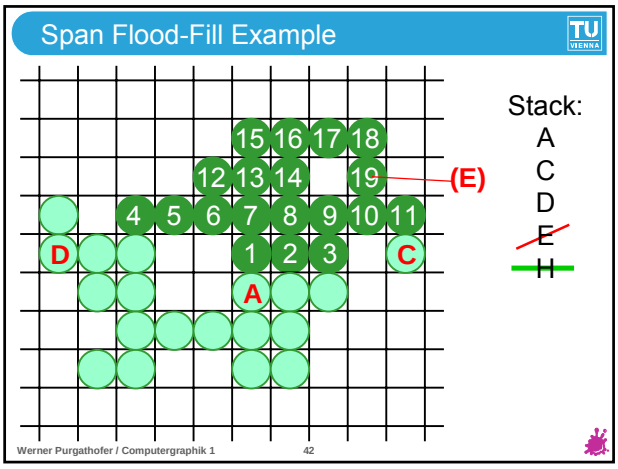


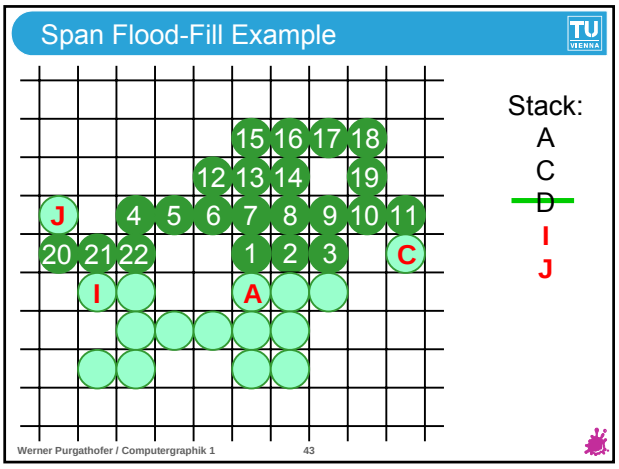
Stack:
A
C
D
E
~~F~~
G

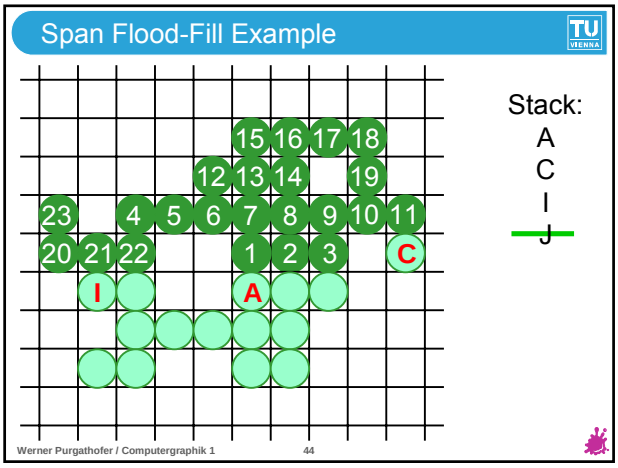
Span Flood-Fill Example

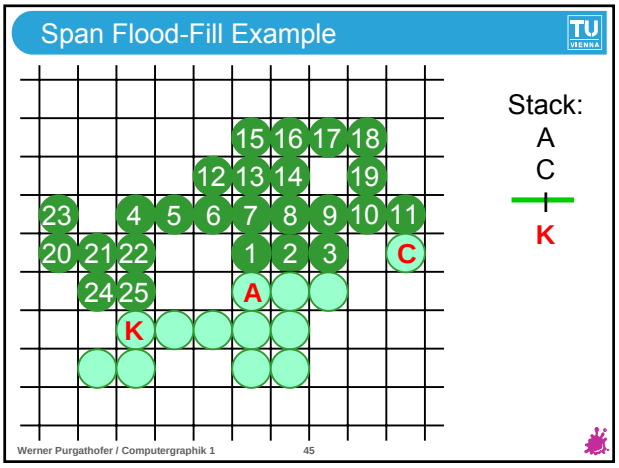


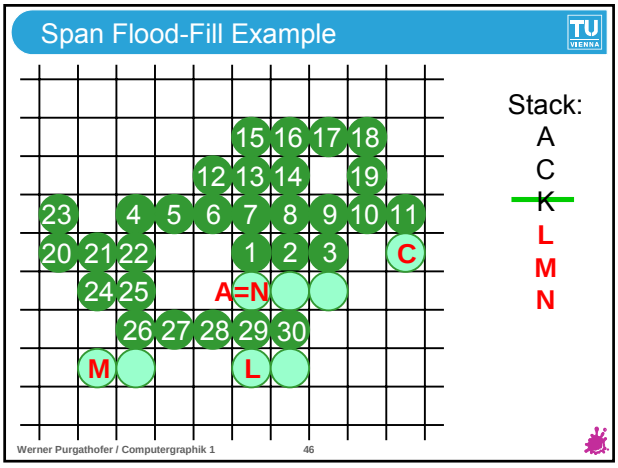
Stack:
A
C
D
E
~~G~~
H

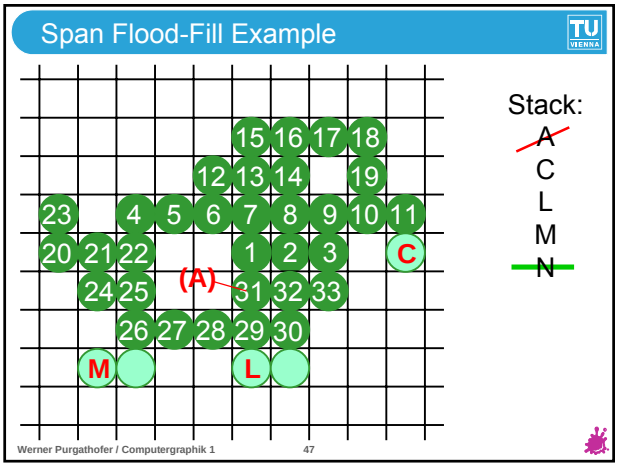

Span Flood-Fill Example
Stack:
A
C
D
E
H
(E)
Werner Purgathofer / Computergraphik 1
42


Span Flood-Fill Example
Stack:
A
C
D
I
J
Werner Purgathofer / Computergraphik 1
43

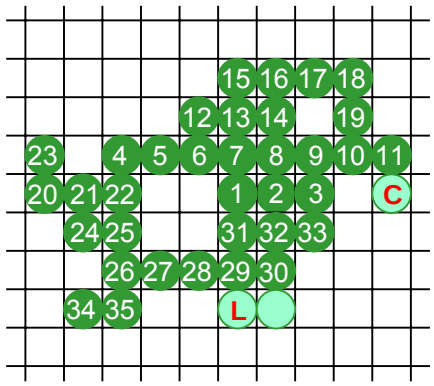

Span Flood-Fill Example
Stack:
A
C
I
J
Werner Purgathofer / Computergraphik 1
44


Span Flood-Fill Example
Stack:
A
C
I
K
Werner Purgathofer / Computergraphik 1
45


Span Flood-Fill Example
Stack:
A
C
K
L
M
N
A=N
Werner Purgathofer / Computergraphik 1
46

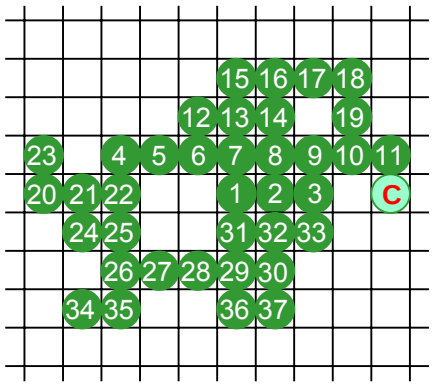

Span Flood-Fill Example
Stack:
A
C
L
M
N
(A)
Werner Purgathofer / Computergraphik 1
47

Span Flood-Fill Example



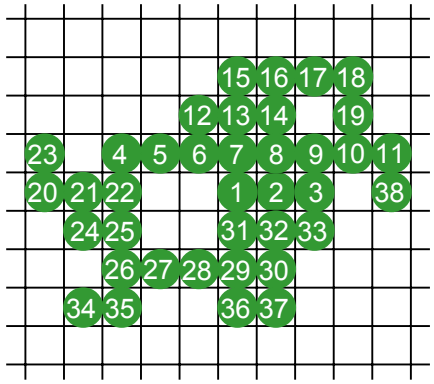
Stack:
C
L
~~M~~

Span Flood-Fill Example



Stack:
C
~~L~~

Span Flood-Fill Example



Stack:
~~C~~

finished!

Character Attributes

- text attributes
  - font (e.g. `Courier`, Arial, Times, Roman, …)
  - styles (regular, **bold**, *italic*, underline,…)
  - size (32 point, 1 point = 1/72 inch)
    - proportionally sized vs. `fixed space fonts`
- string attributes
  - orientation
  - alignment (left, center, right, justify)

`horizontal` slanted `vertical`

Displayed primitives generated by the raster algorithms discussed in Chapter 3 have a jagged, or stairstep, appearance.

Displayed primitives generated by the raster algorithms discussed in Chapter 3 have a jagged, or stairstep, appearance.

Displayed primitives generated by the raster algorithms discussed in Chapter 3 have a jagged, or stairstep, appearance.

Displayed primitives generated by the raster algorithms discussed in Chapter 3 have a jagged, or stairstep, appearance.

Triangle and Polygon Attributes

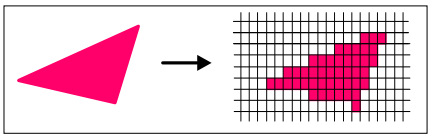
- color
- material
- transparency
- texture
- surface details
- reflexion properties, …

→ defined illumination produces effects

Aliasing and Antialiasing

- what is aliasing? ['eiliæsiŋ]
- what is the reason for aliasing?
- what can we do against it?

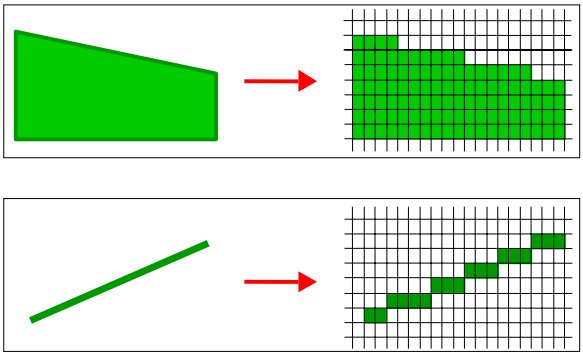
What is Aliasing?



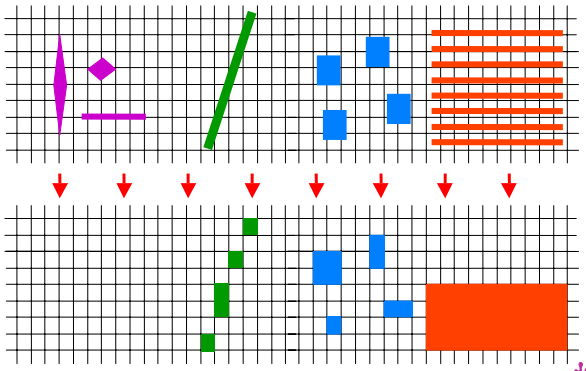
errors that are caused by the discretization of analog data to digital data

- *too bad resolution*
- *too few colors*
- *too few images / sec*
- *geometric errors*
- *numeric errors*

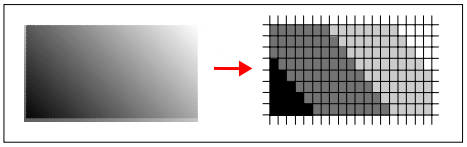
Aliasing: Staircase Effect



Various Aliasing Effects



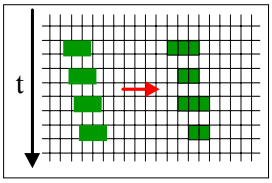
Aliasing from too few Colors



artificial color borders can appear

Aliasing in Animations

- jumping images
- "worming"



- backwards rotating wheels

Solutions against Aliasing?

- **1. improve the devices**
  - higher resolution
  - more color levels
  - faster image sequence

expensive or incompatible

- **2. improve the images = *antialiasing***
  - postprocessing
  - prefiltering !

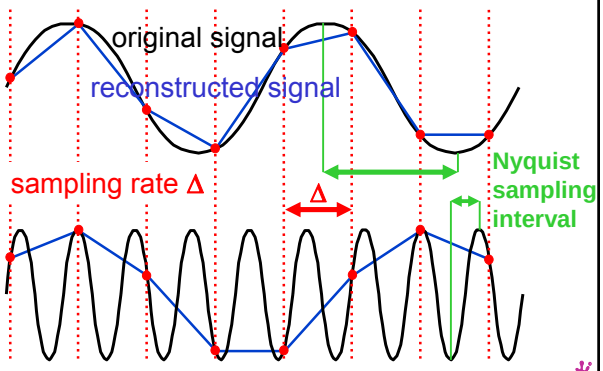
software

Shannon Sampling Theorem

a signal can only be reconstructed without information loss if the sampling frequency is at least twice the highest frequency of the signal

this border frequency is called "Nyquist Limit"

Shannon Sampling Theorem



Antialiasing: Nyquist Sampling Frequency

- **a signal can only be reconstructed without information loss if the sampling frequency is at least twice the highest frequency of the signal**

Nyquist sampling frequency: $f_s = 2 f_{max}$

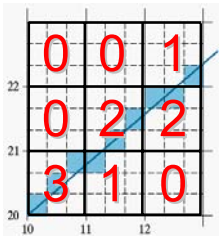
$$\Delta x_s = \frac{\Delta x_{cycle}}{2} \quad \text{with} \quad \Delta x_{cycle} = 1 / f_{max}$$

i.e. sampling interval ≤ one-half cycle interval

Antialiasing Strategies

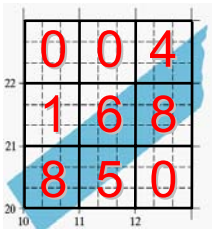
- supersampling straight-line segments
- subpixel weighting masks
- area sampling straight-line segments
- filtering techniques
- compensating for line-intensity differences
- antialiasing area boundaries
  - (adjusting boundary pixel positions)
  - adjusting boundary pixel intensity

Antialiasing: Supersampling Lines



mathematical line

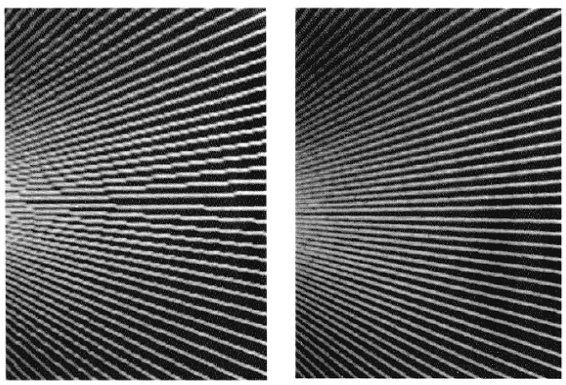
3 = max. intensity
...
0 = min. intensity



line of finite width

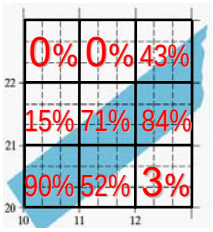
9 = max. intensity
...
0 = min. intensity

Antialiasing



Antialiasing: Area Sampling Lines

- calculate the pixel coverage exactly
- can be done with incremental schemes



Antialiasing: Pixel Weighting Masks

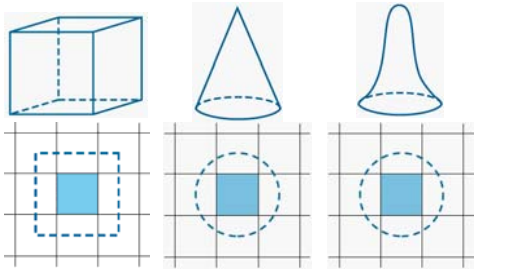
1	2	1
2	4	2
1	2	1

relative weights for a grid of 3x3 subpixels

- more weight for center subpixels
- must be divided by sum of weights
- subpixel grids can also include some neighboring pixels

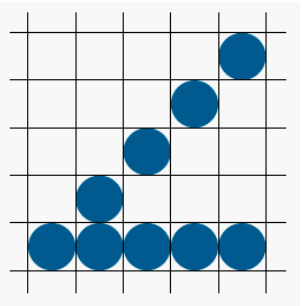
Antialiasing: Filtering Techniques

continuous overlapping weighting functions to calculate the antialiased values with integrals



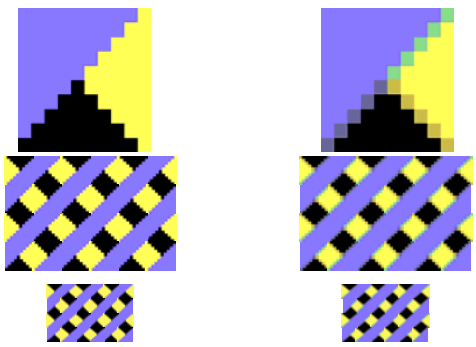
box filter cone filter Gaussian filter

Antialiasing: Intensity Differences

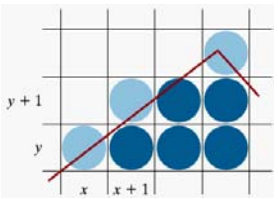


- unequal line lengths displayed with the same number of pixels in each line/row have different intensities
- proper antialiasing compensates for that!

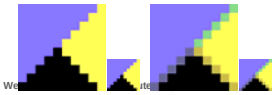
Antialiasing



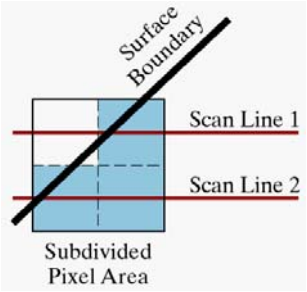
Antialiasing Area Boundaries (1)



adjusting pixel intensities along an area boundary



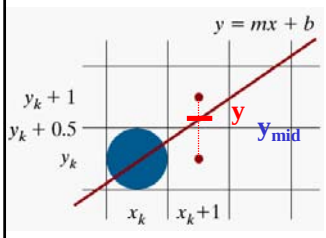
alternative 1: supersampling



Antialiasing Area Boundaries (2)

alternative 2: like midpoint line algorithm

$p' = y - y_{mid} = [m(x_k + 1) + b] - (y_k + 0.5)$



$p' < 0 \Rightarrow$ y closer to y_k

$p' > 0 \Rightarrow$ y closer to y_{k+1}

$p = p' + (1-m)$:

$p < 1-m \Rightarrow$ closer to y_k

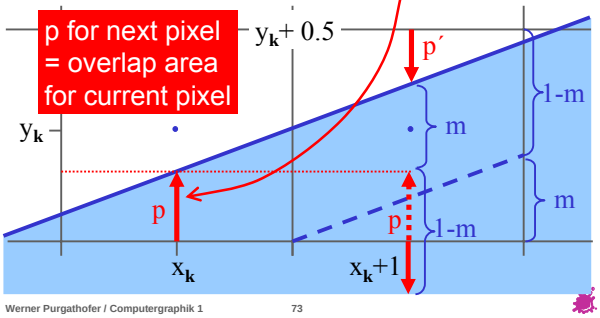
$p > 1-m \Rightarrow$ closer to y_{k+1}

(and $p \in [0,1]$)

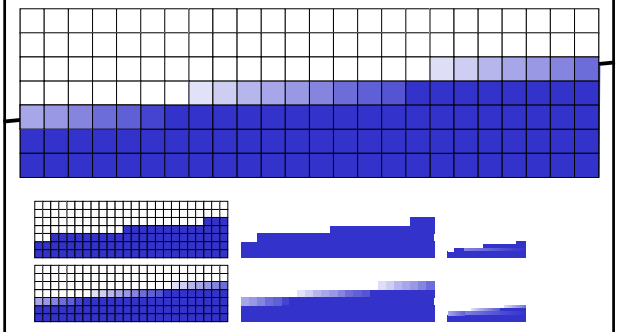
Antialiasing Area Boundaries (3)

$p = p' + (1-m) = [m(x_k + 1) + b] - (y_k + 0.5) + (1 - m) =$

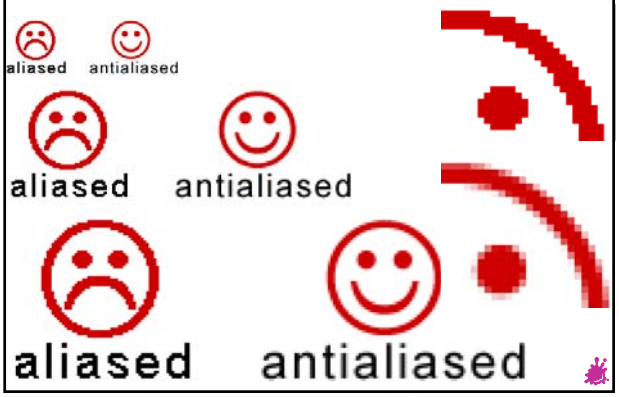
$= mx_k + b - y_k + 0.5 = mx_k + b - (y_k - 0.5)$



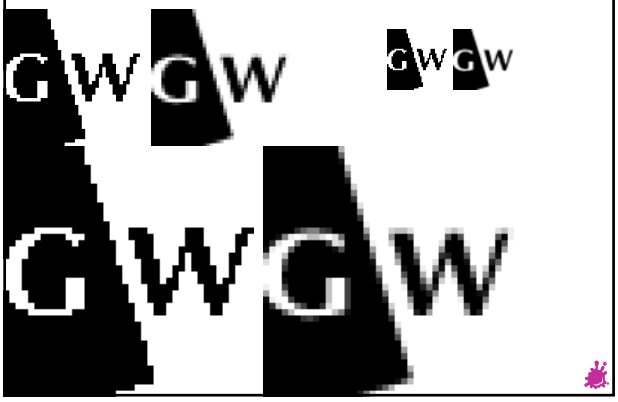
Antialiasing Area Boundaries (4)



Antialiasing Examples



Antialiasing Examples



Summary: Attributes of Primitives

- in 2D
  - points, lines
  - polygons, circles, ellipses & other curves (also filled)
  - characters
- in 3D
  - triangles & other polygons
- anti-aliasing