Einführung in Visual Computing

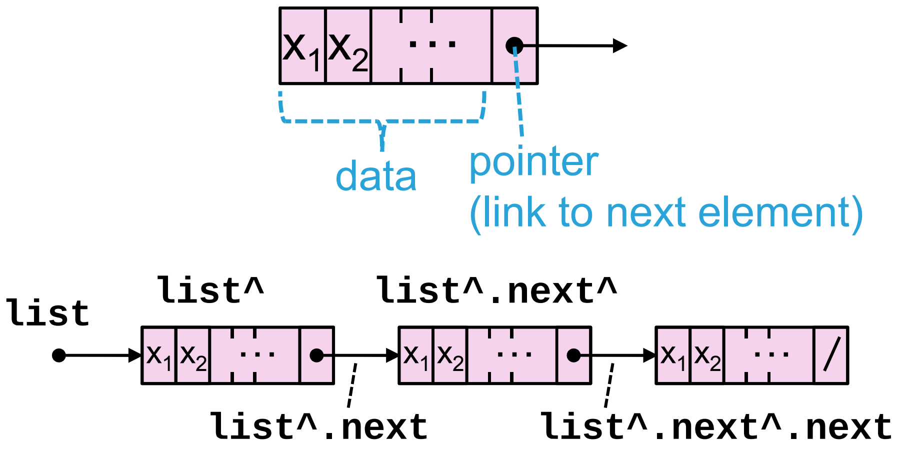
186.822

Polygon Filling

Werner Purgathofer

Linked Lists

- flexible data structure



data

pointer (link to next element)

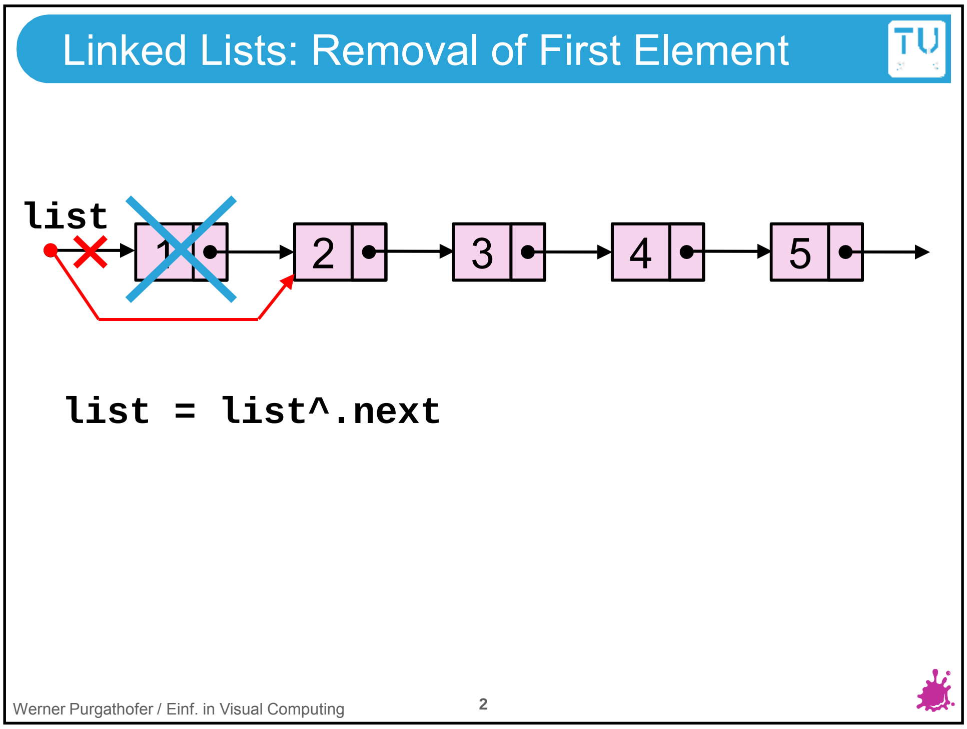
list

list^

list^.next^

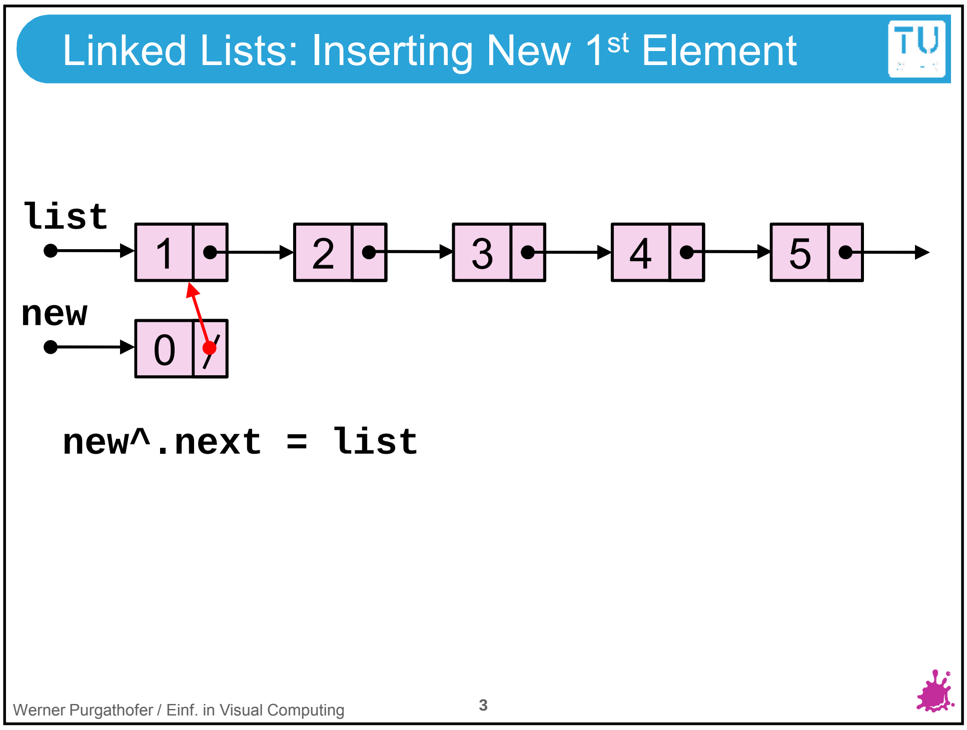
list^.next

list^.next^.next



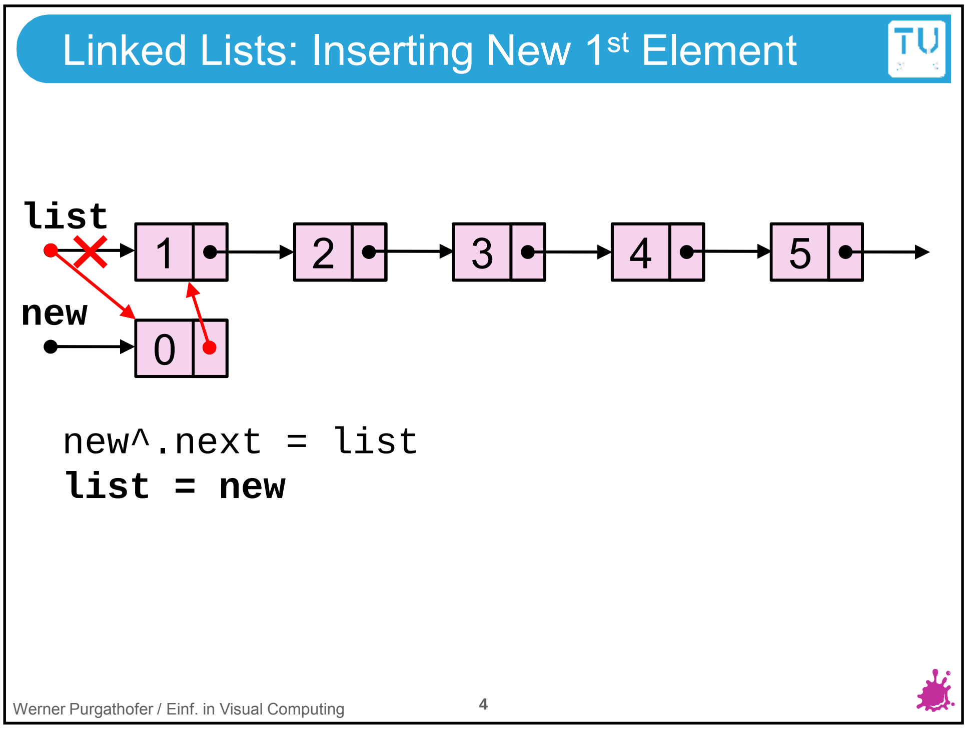
Linked Lists: Removal of First Element

```
list = list^.next
```



Linked Lists: Inserting New 1st Element

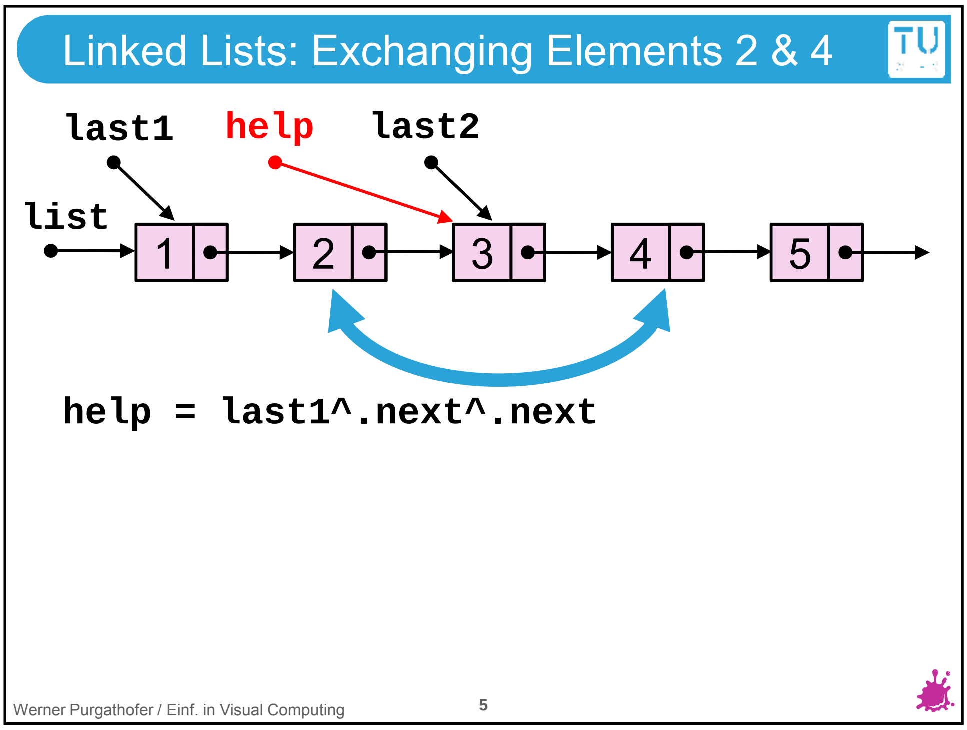
```
new^.next = list
```



Linked Lists: Inserting New 1st Element

```
new^.next = list
list = new
```

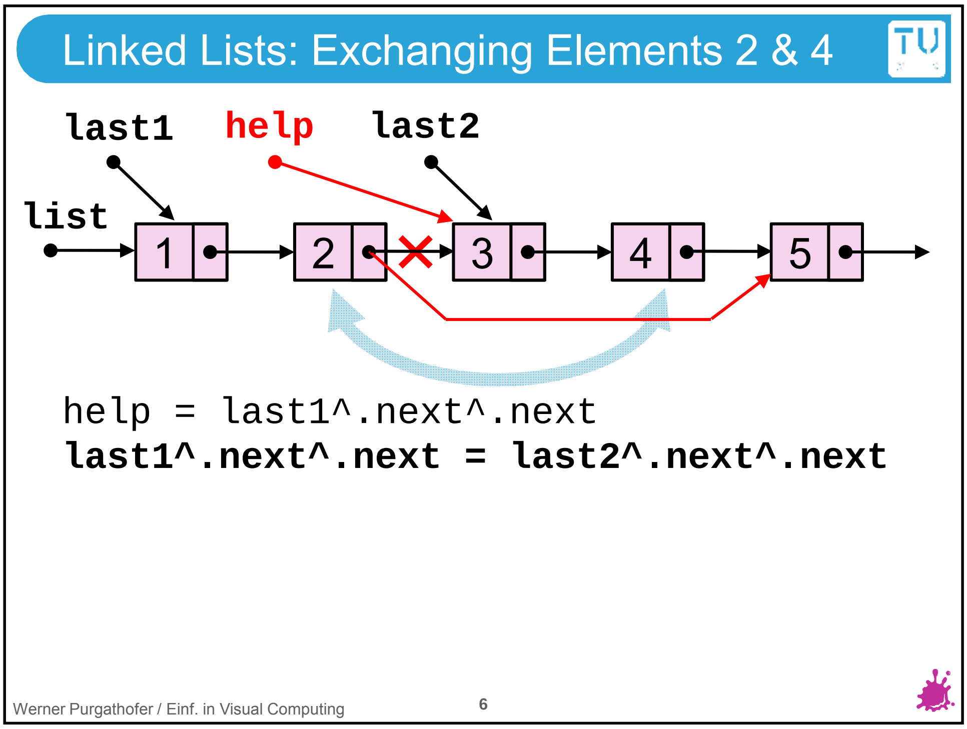
Werner Purgathofer / Einf. in Visual Computing



Linked Lists: Exchanging Elements 2 & 4

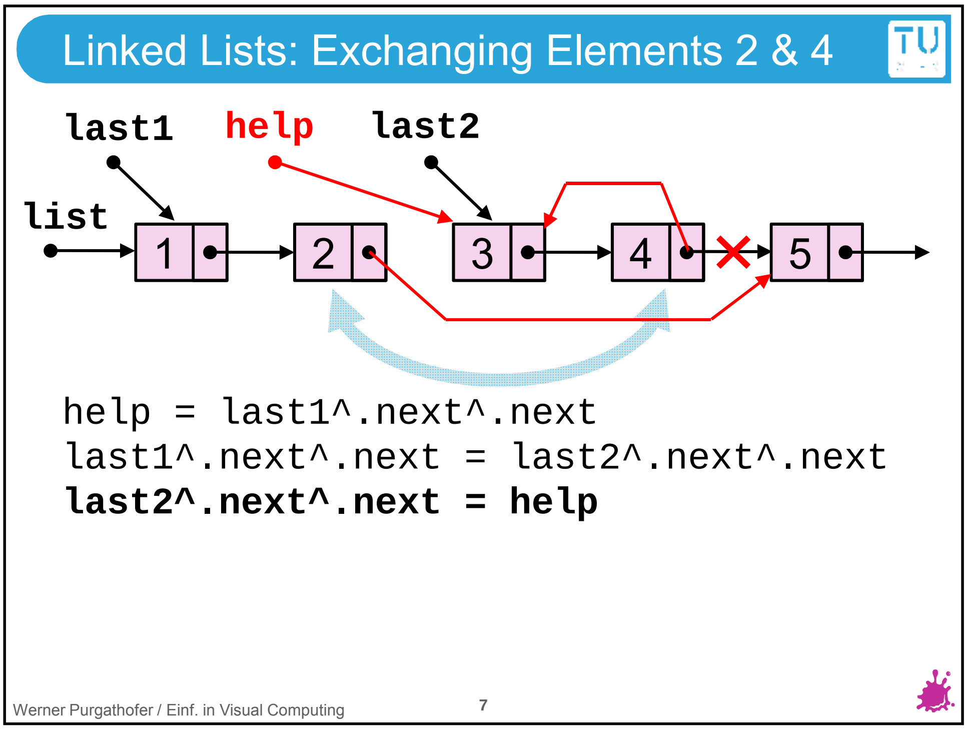
```
help = last1^.next^.next
```

Werner Purgathofer / Einf. in Visual Computing



Linked Lists: Exchanging Elements 2 & 4

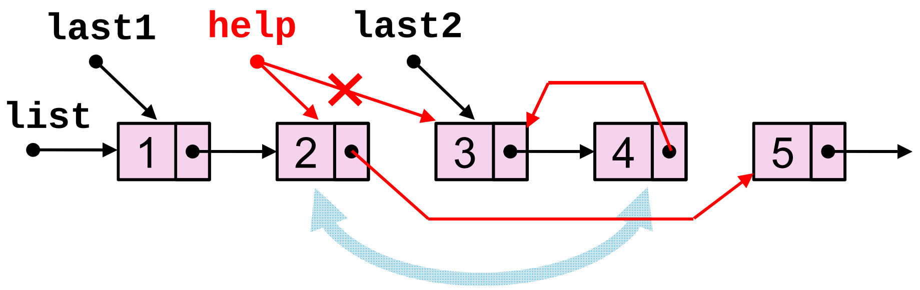
```
help = last1^.next^.next
last1^.next^.next = last2^.next^.next
```



Linked Lists: Exchanging Elements 2 & 4

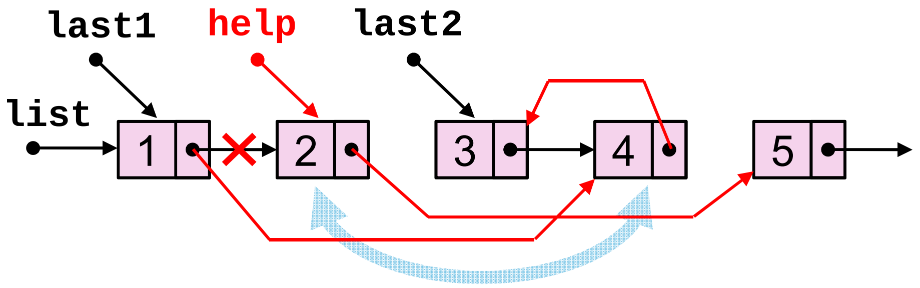
```
help = last1^.next^.next
last1^.next^.next = last2^.next^.next
last2^.next^.next = help
```

Linked Lists: Exchanging Elements 2 & 4

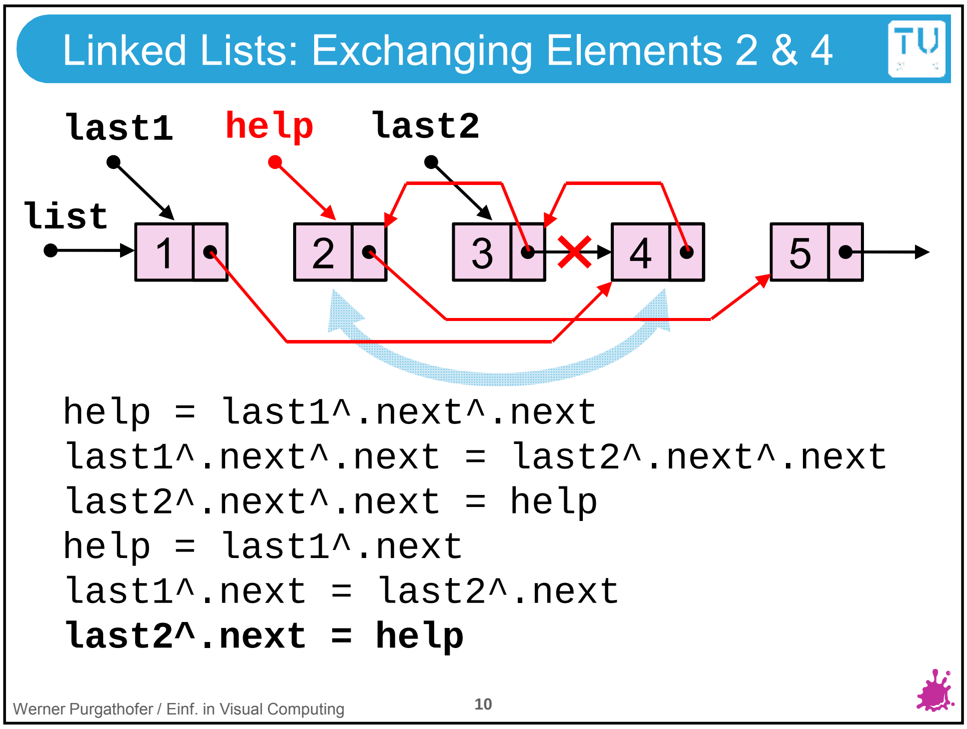


```
help = last1^.next^.next
last1^.next^.next = last2^.next^.next
last2^.next^.next = help
help = last1^.next
```

Linked Lists: Exchanging Elements 2 & 4

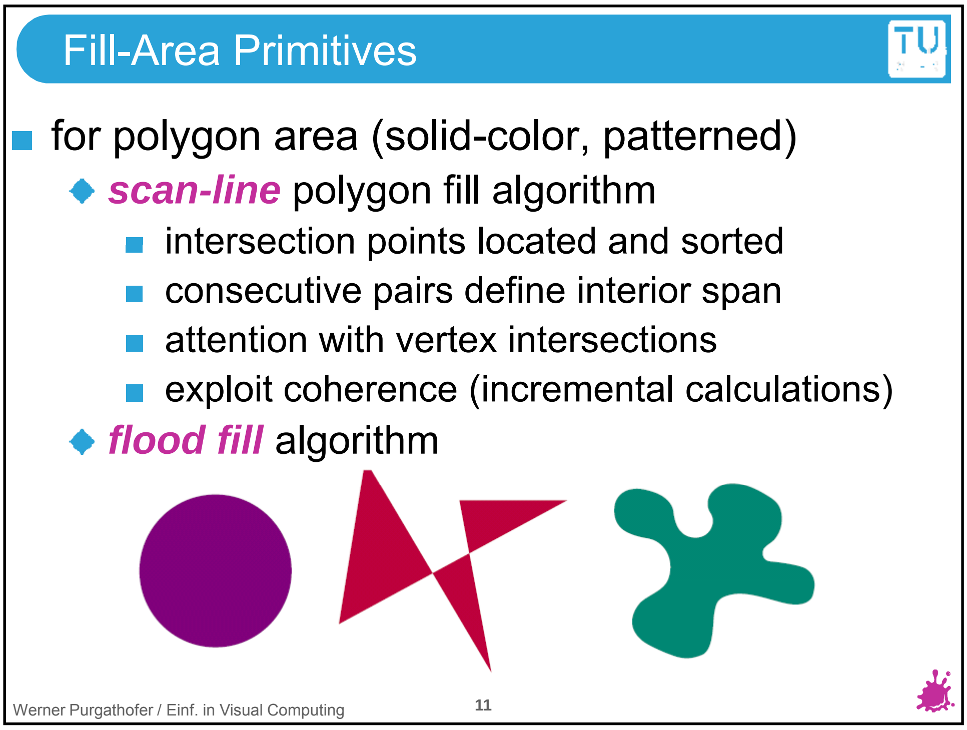


```
help = last1^.next^.next
last1^.next^.next = last2^.next^.next
last2^.next^.next = help
help = last1^.next
last1^.next = last2^.next
```



Linked Lists: Exchanging Elements 2 & 4

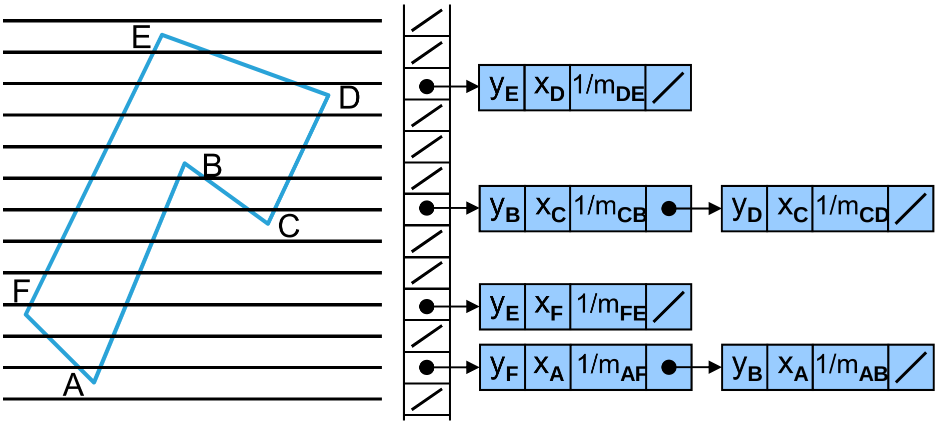
```
help = last1^.next^.next
last1^.next^.next = last2^.next^.next
last2^.next^.next = help
help = last1^.next
last1^.next = last2^.next
last2^.next = help
```



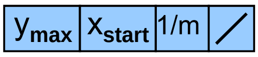
Fill-Area Primitives

- for polygon area (solid-color, patterned)
  - ***scan-line*** polygon fill algorithm
    - intersection points located and sorted
    - consecutive pairs define interior span
    - attention with vertex intersections
    - exploit coherence (incremental calculations)
  - ***flood fill*** algorithm

Scan-Line Fill: Sorted Edge Table

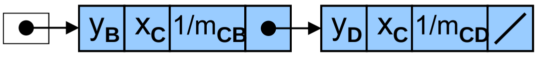


The sorted edge table contains all polygon edges sorted by lowest y-value

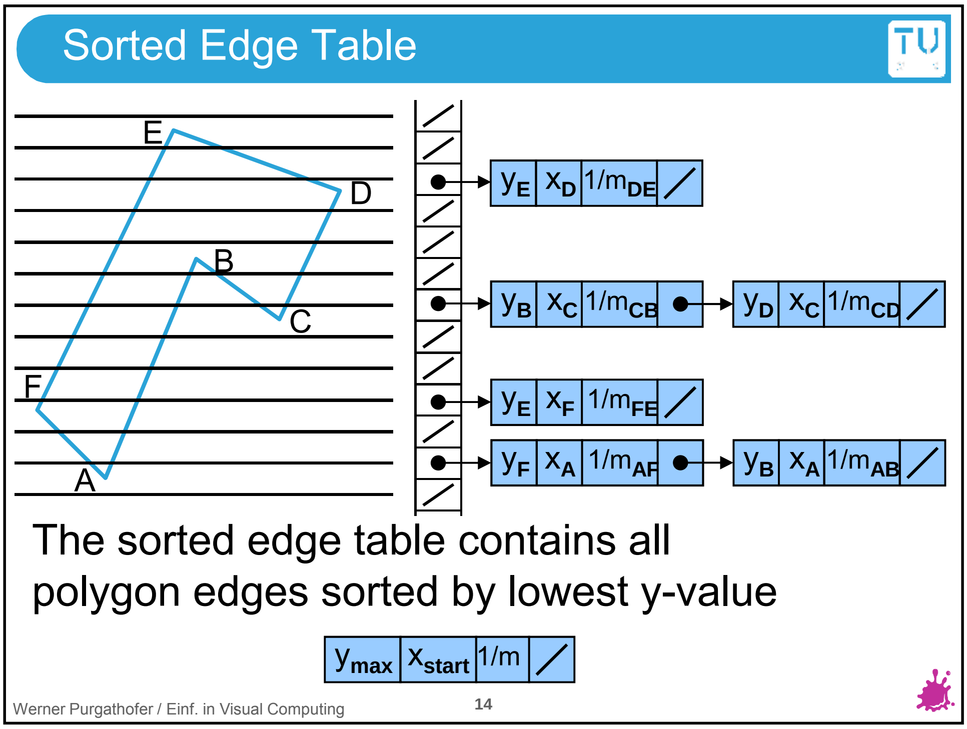


Scan-Line Fill: Sorted Edge Table

- sort all edges by 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



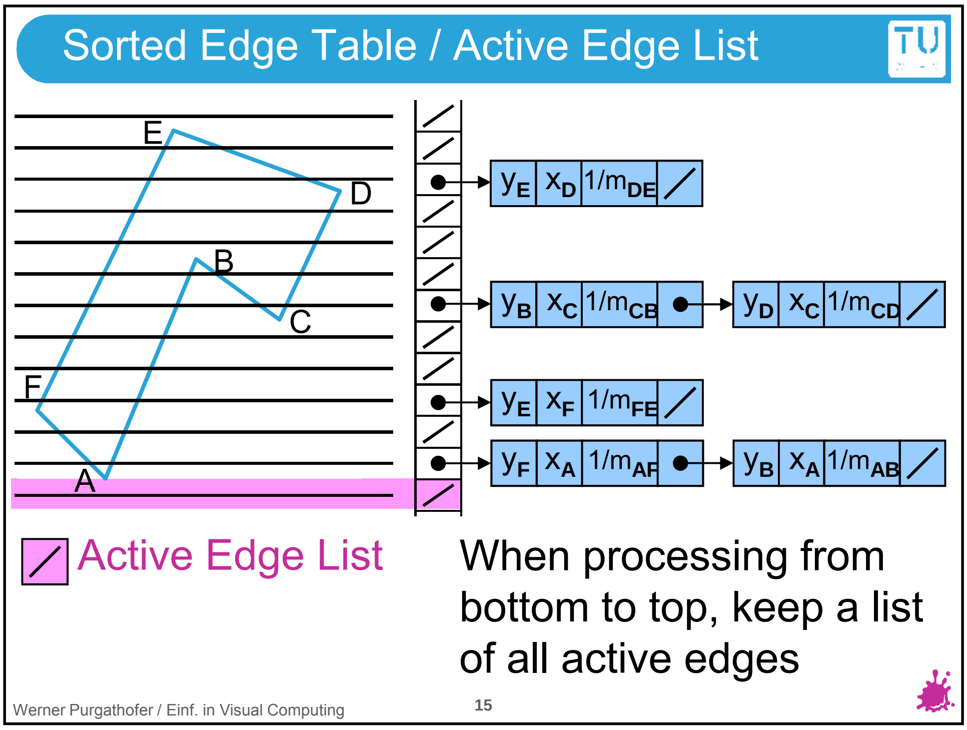
Sorted Edge Table

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 | $1/m_{AB}$

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

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

Werner Purgathofer / Einf. in Visual Computing 14



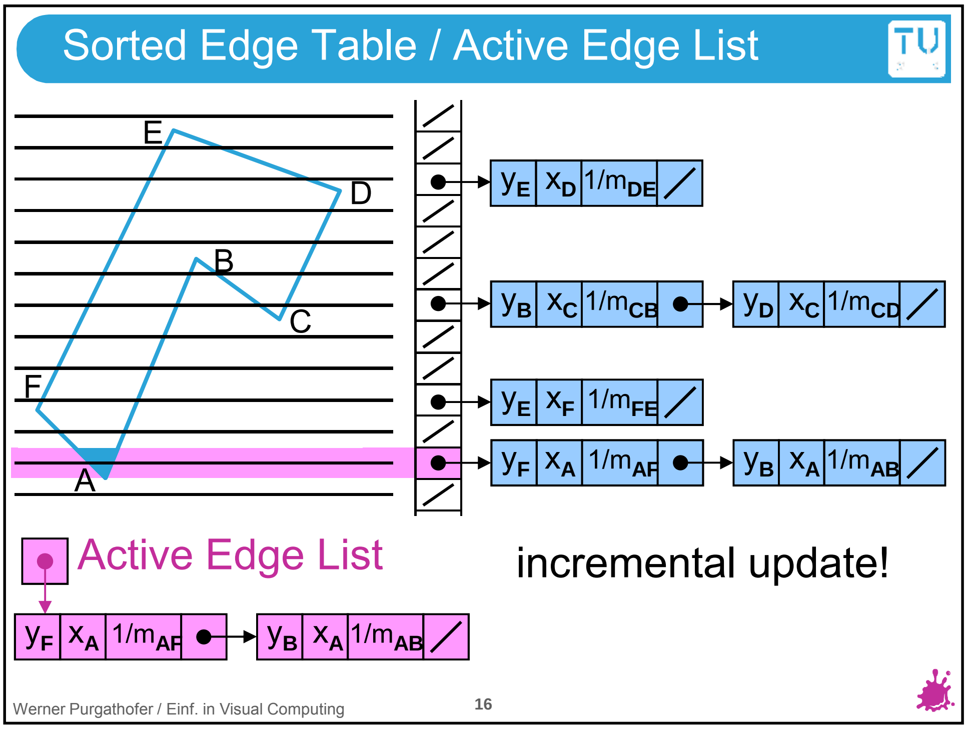
Sorted Edge Table / Active Edge List

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

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

Werner Purgathofer / Einf. in Visual Computing 15



Sorted Edge Table / Active Edge List

y_E | x_D | $1/m_{DE}$

y_B | x_C | $1/m_{CB}$ → y_D | x_C | $1/m_{CD}$

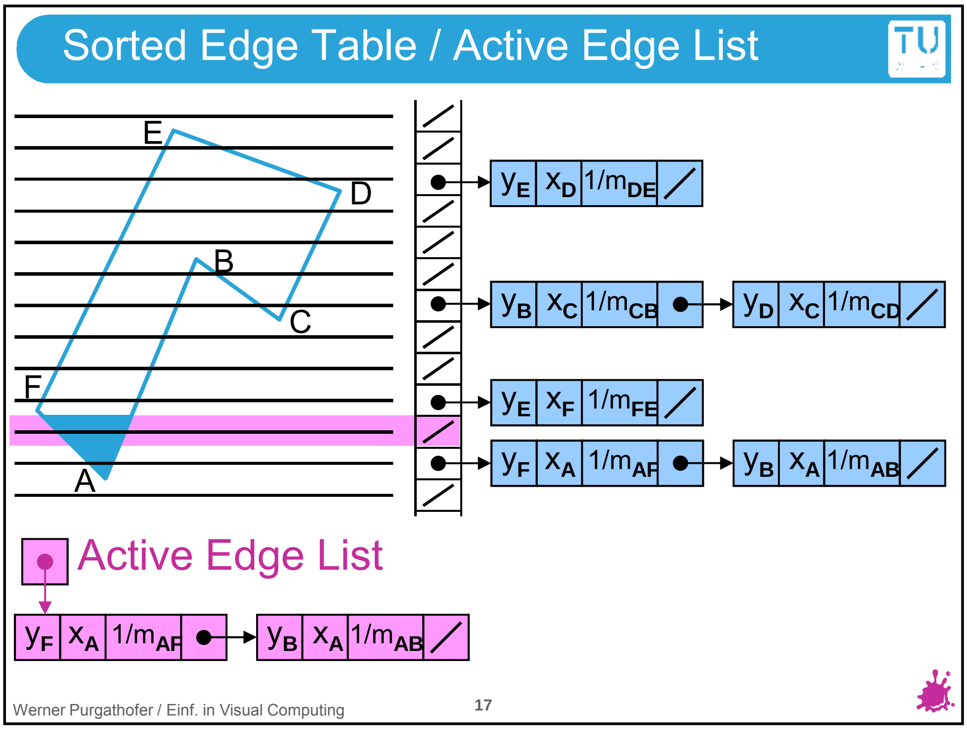
y_E | x_F | $1/m_{FE}$

y_F | x_A | $1/m_{AF}$ → y_B | x_A | $1/m_{AB}$

Active Edge List

y_F | x_A | $1/m_{AF}$ → y_B | x_A | $1/m_{AB}$

incremental update!



Sorted Edge Table / Active Edge List

y_E | x_D | $1/m_{DE}$

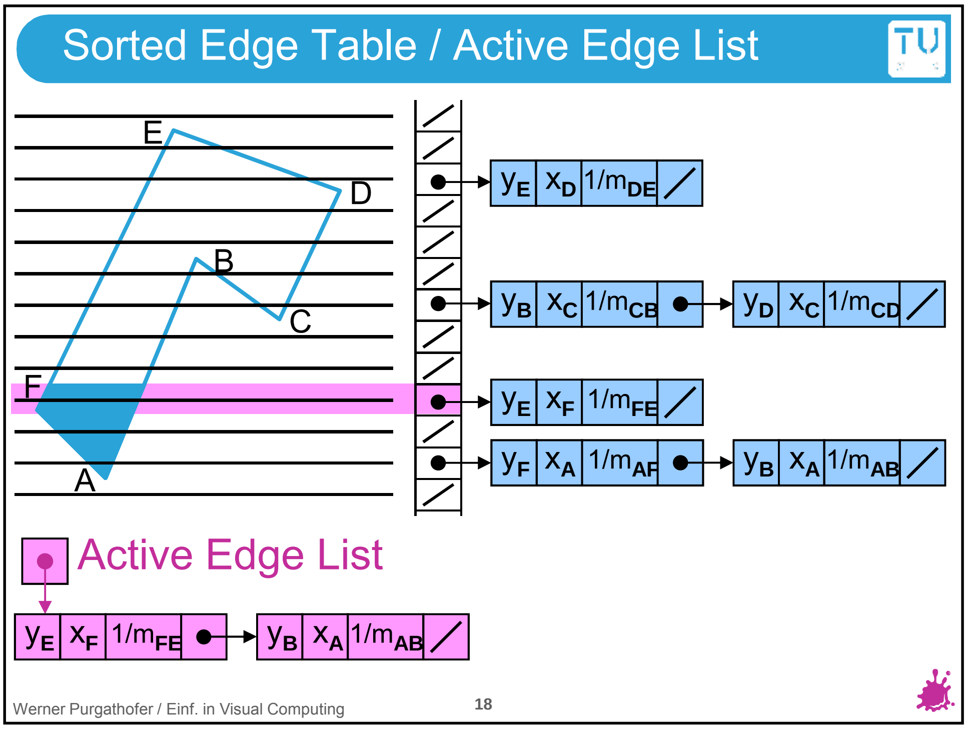
y_B | x_C | $1/m_{CB}$ → y_D | x_C | $1/m_{CD}$

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_F | x_A | $1/m_{AF}$ → y_B | x_A | $1/m_{AB}$



Sorted Edge Table / Active Edge List

y_E x_D $1/m_{DE}$

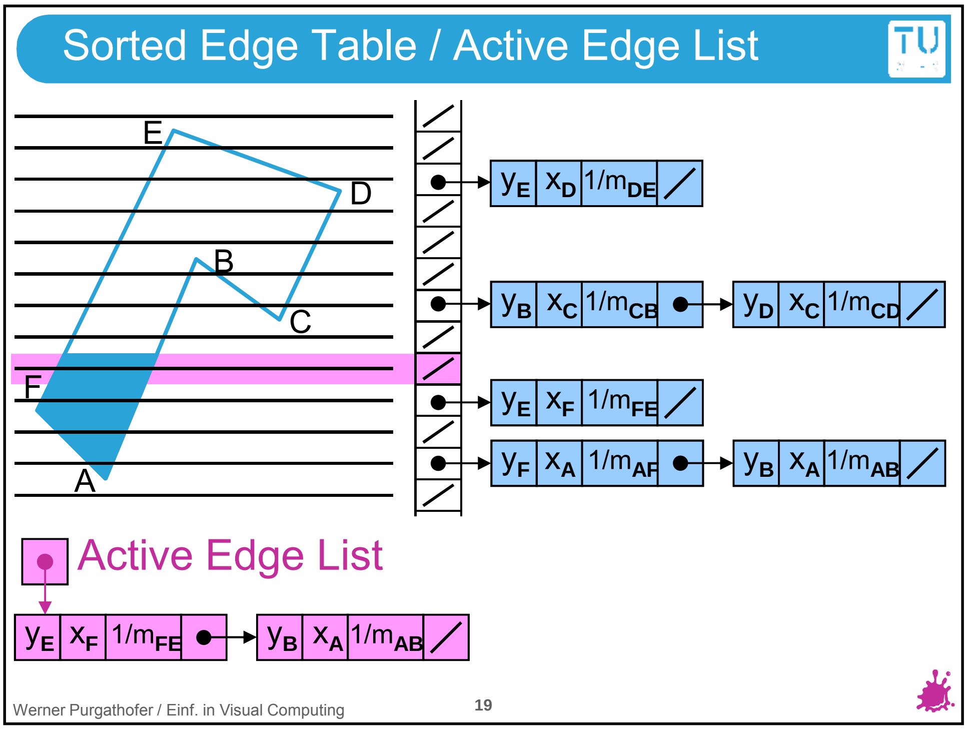
y_B x_C $1/m_{CB}$ → y_D x_C $1/m_{CD}$

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 $1/m_{AB}$



Sorted Edge Table / Active Edge List

y_E x_D $1/m_{DE}$

y_B x_C $1/m_{CB}$ → y_D x_C $1/m_{CD}$

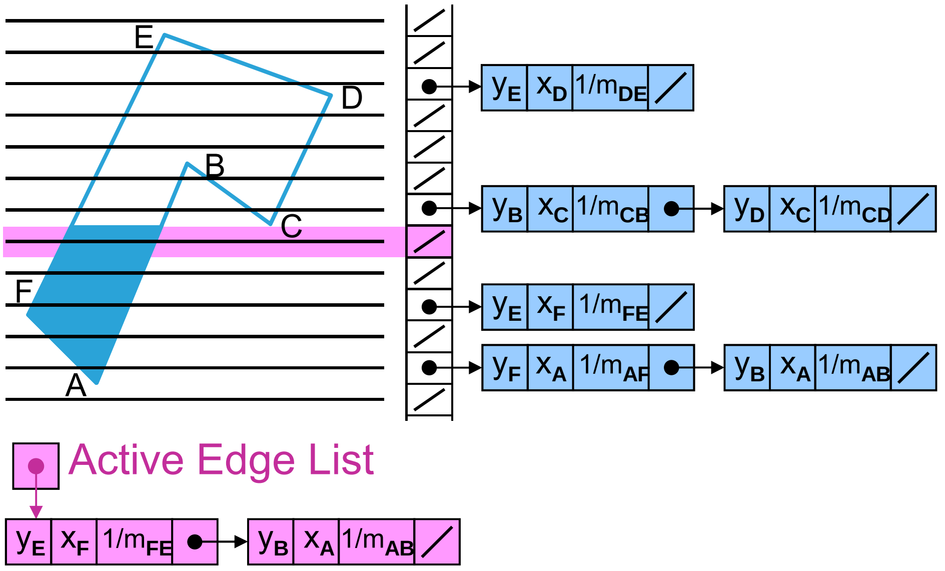
y_E x_F $1/m_{FE}$

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 $1/m_{AB}$

Sorted Edge Table / Active Edge List



y_E x_D $1/m_{DE}$

y_B x_C $1/m_{CB}$ → y_D x_C $1/m_{CD}$

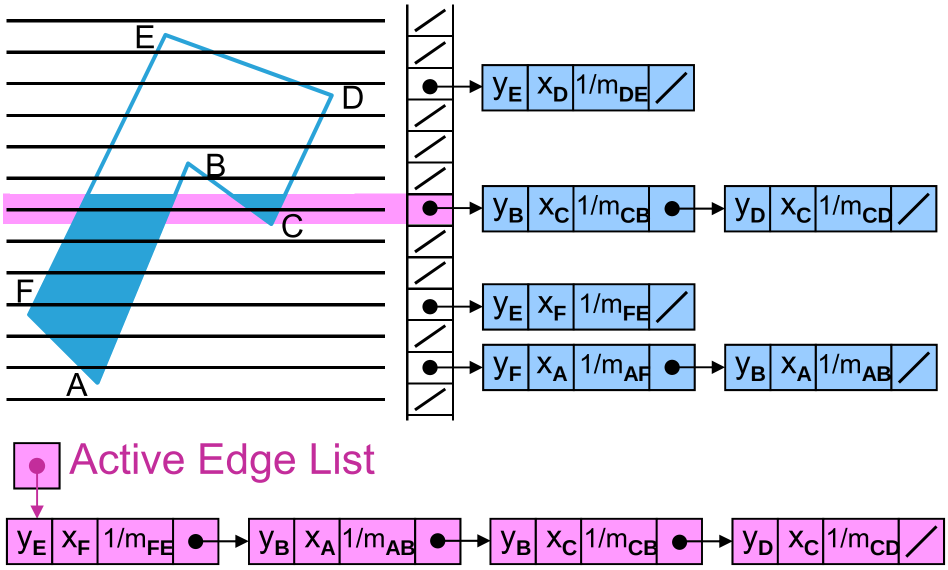
y_E x_F $1/m_{FE}$

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 $1/m_{AB}$

Sorted Edge Table / Active Edge List



y_E x_D $1/m_{DE}$

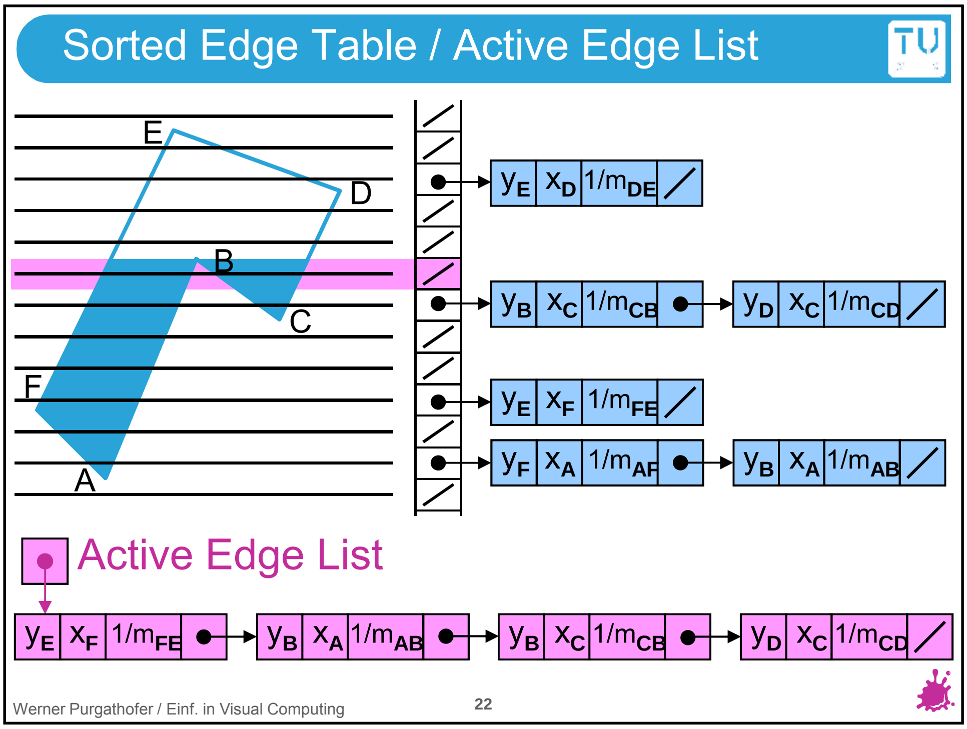
y_B x_C $1/m_{CB}$ → y_D x_C $1/m_{CD}$

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 $1/m_{CD}$

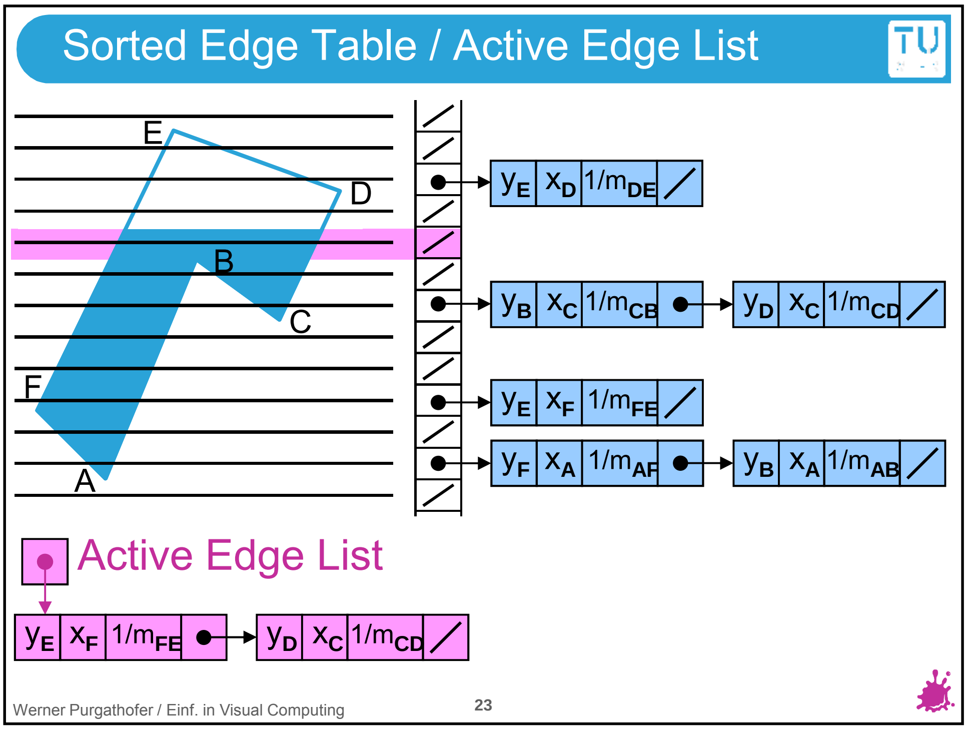


Sorted Edge Table / Active Edge List

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 | $1/m_{CD}$

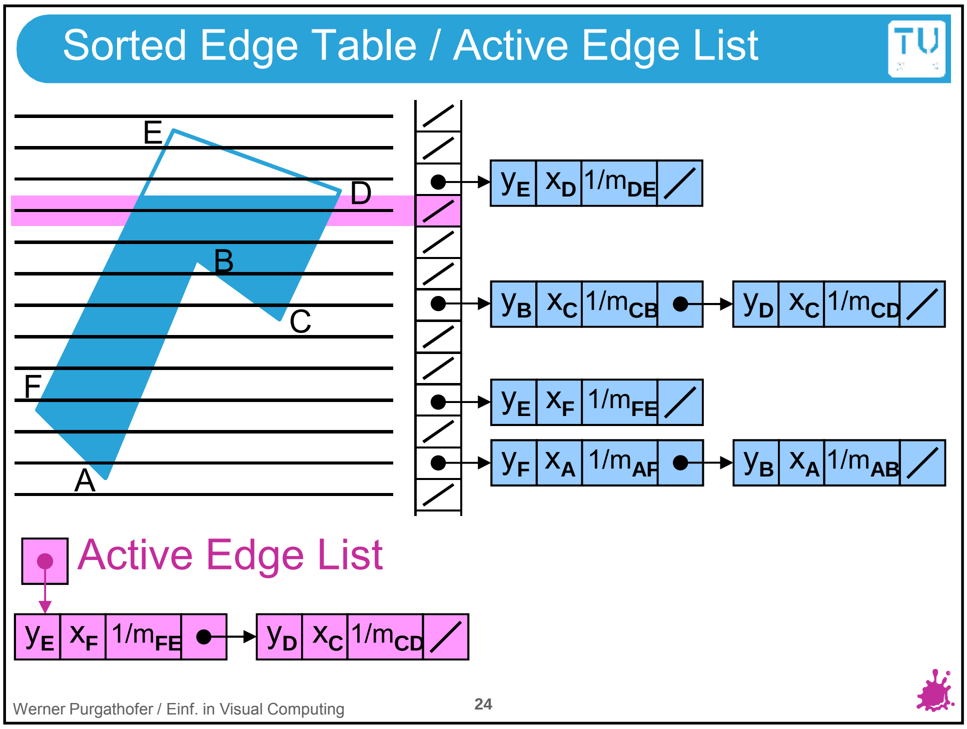


Sorted Edge Table / Active Edge List

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 | $1/m_{CD}$



Sorted Edge Table / Active Edge List

y_E x_D $1/m_{DE}$

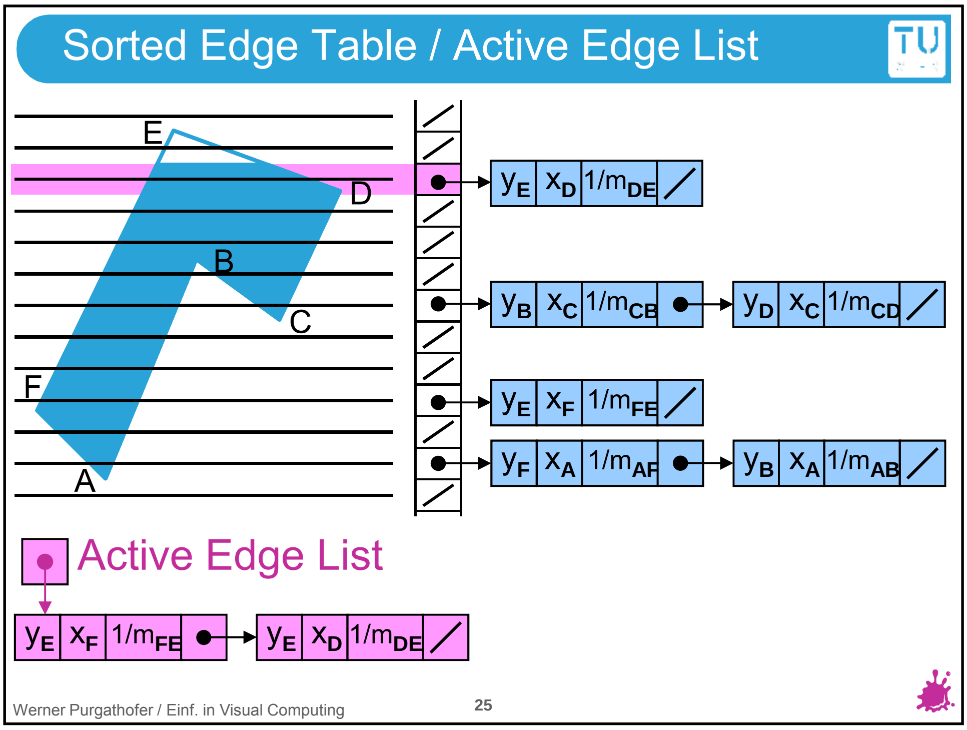
y_B x_C $1/m_{CB}$ → y_D x_C $1/m_{CD}$

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 $1/m_{CD}$



Sorted Edge Table / Active Edge List

y_E x_D $1/m_{DE}$

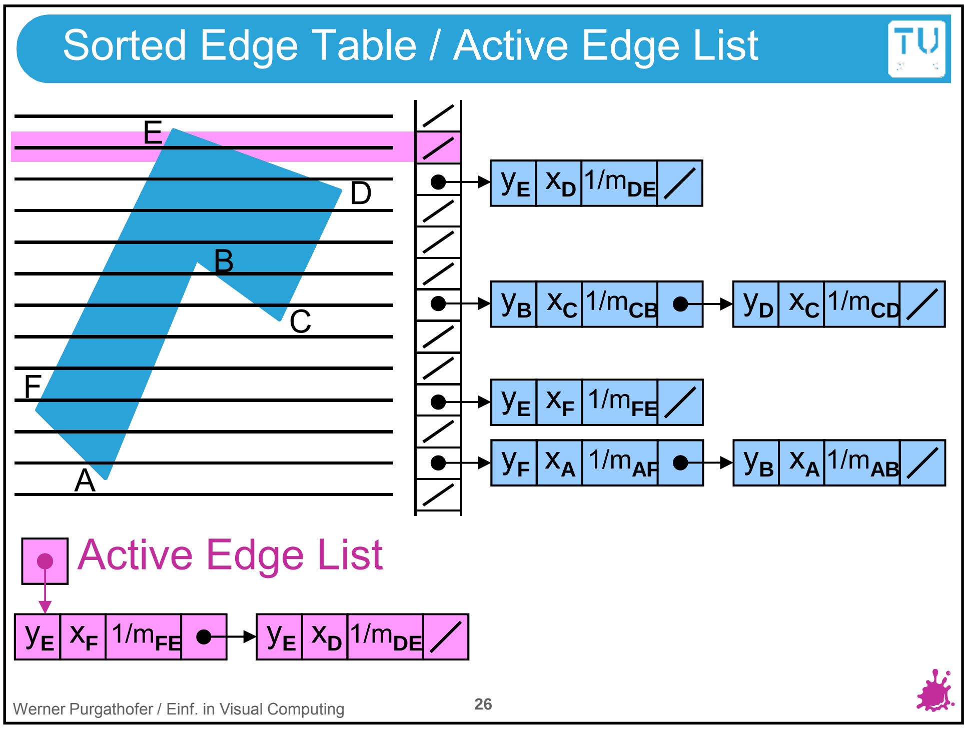
y_B x_C $1/m_{CB}$ → y_D x_C $1/m_{CD}$

y_E x_F $1/m_{FE}$

y_F x_A $1/m_{AF}$ → y_B x_A $1/m_{AB}$

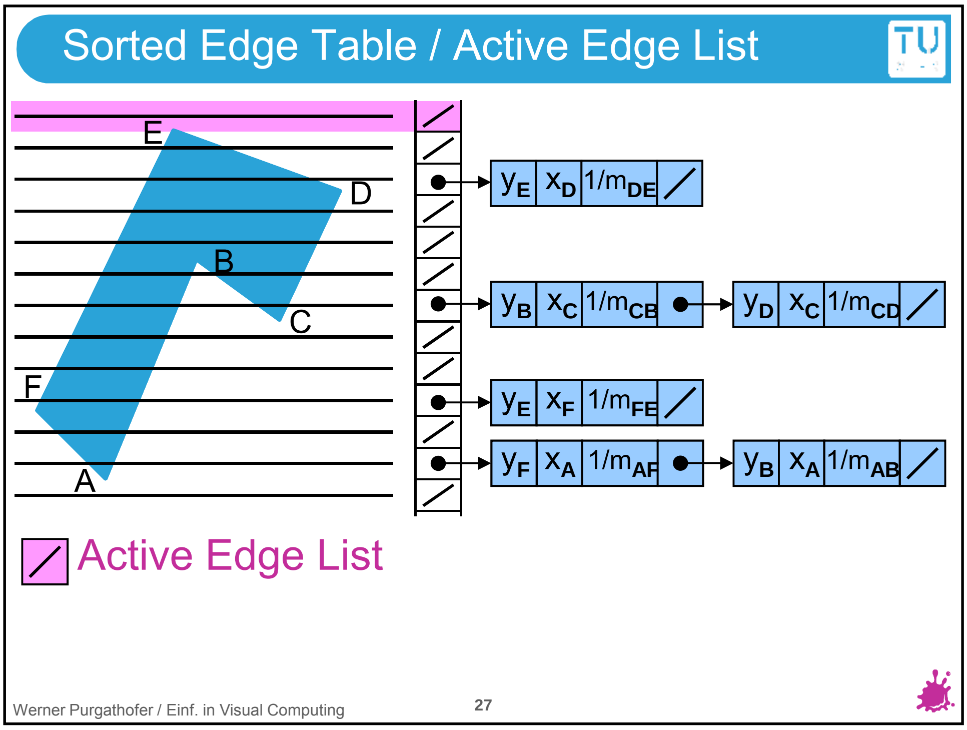
Active Edge List

y_E x_F $1/m_{FE}$ → y_E x_D $1/m_{DE}$



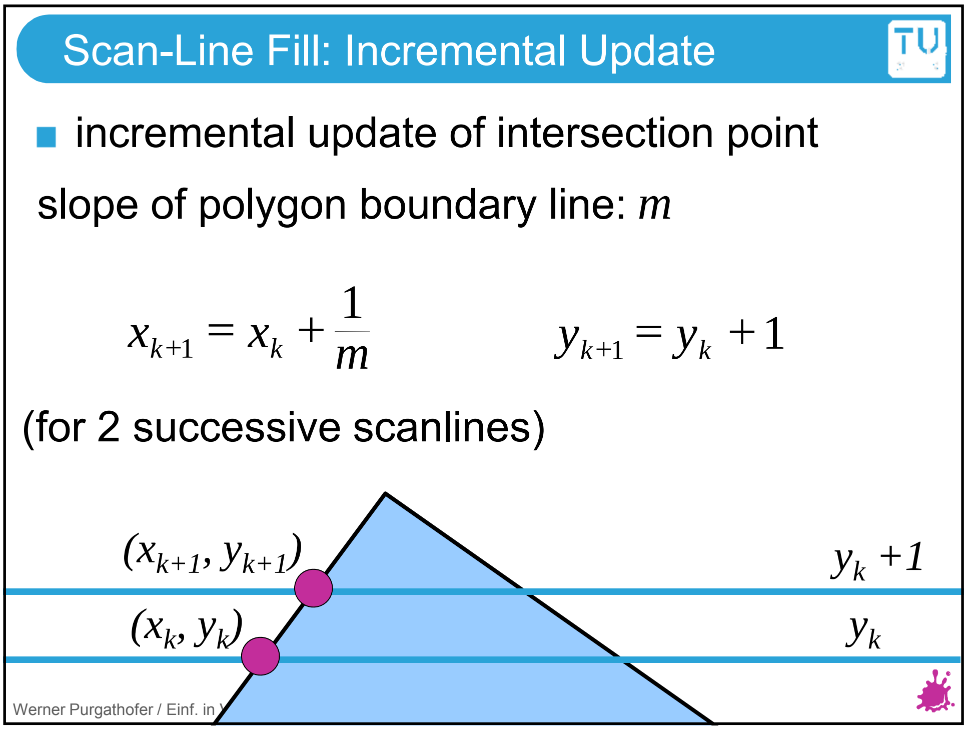
Sorted Edge Table / Active Edge List

Active Edge List



Sorted Edge Table / Active Edge List

Active Edge List



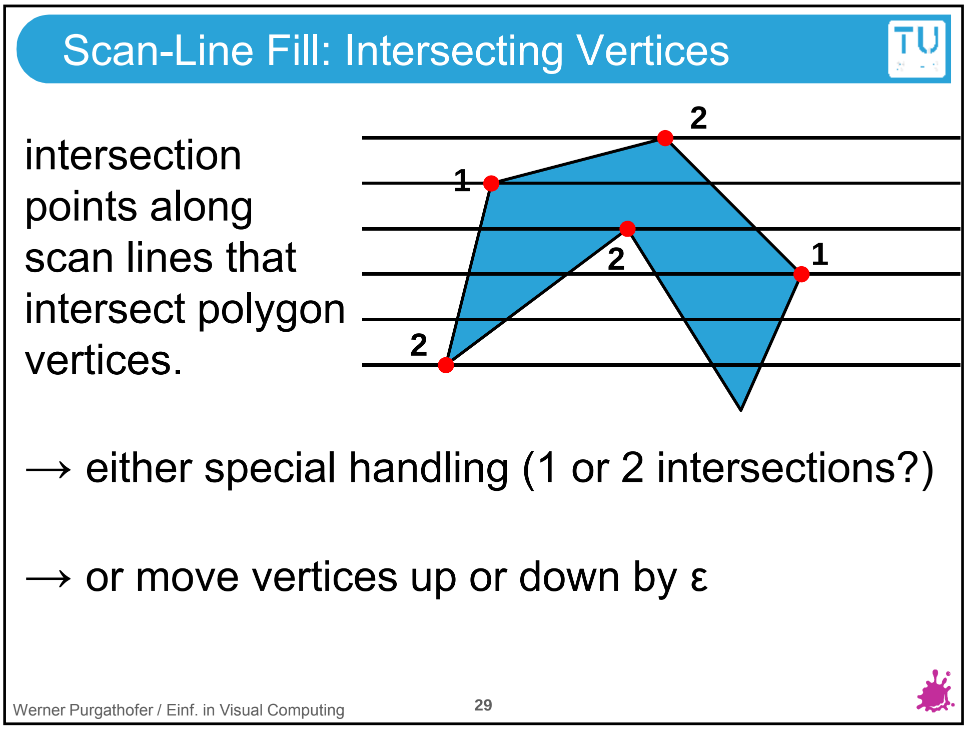
Scan-Line Fill: Incremental Update

- incremental update of intersection point

slope of polygon boundary line: m

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

(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?)

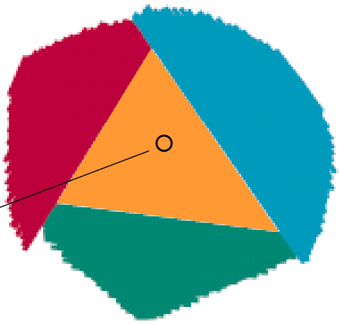
→ or move vertices up or down by ε

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

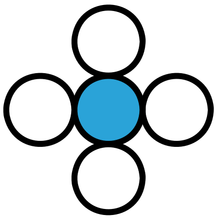
area must be distinguishable from boundaries

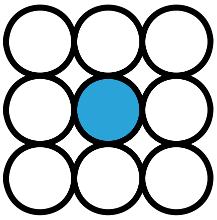


seed point

example: area defined within multiple color boundaries

Flood-Fill: Connectedness

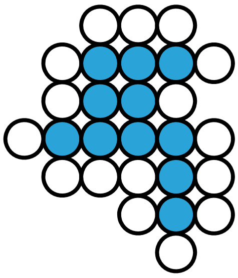


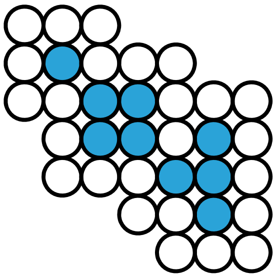


Definition:
4-connected means, that a connection is only valid in these 4 directions

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

Examples for 4- and 8-connected





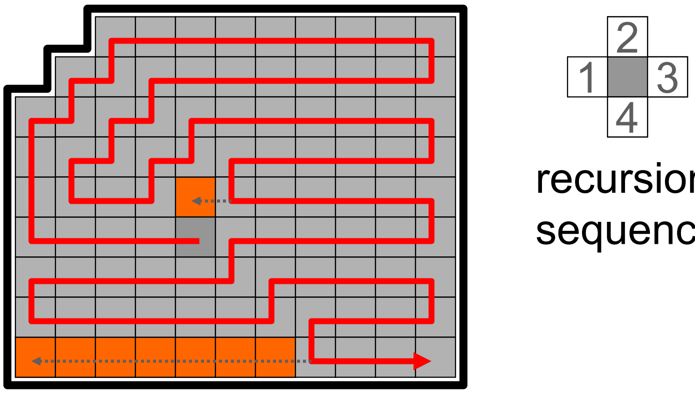
a 4-connected area has an 8-connected border

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

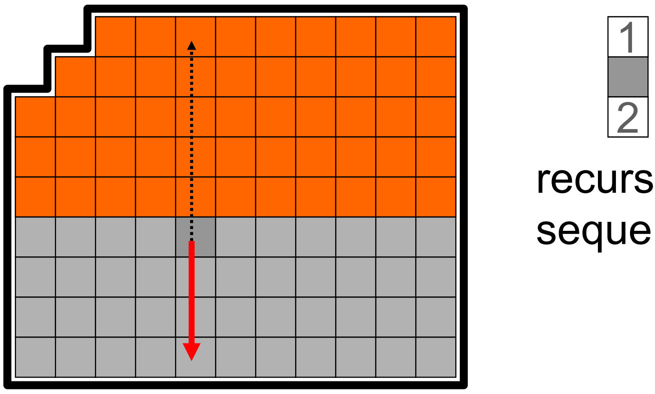


recursion sequence

Span Flood-Fill Algorithm

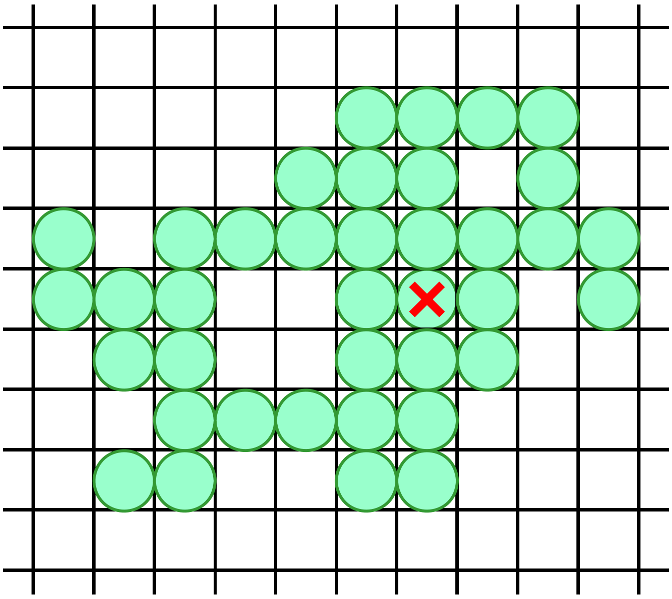
- floodFill4 produces too high stacks (recursion!)
- solution:
  - incremental horizontal fill (left to right)
  - recursive vertical fill (first up then down)

Good Behavior of Span Flood-Fill



recursion sequence

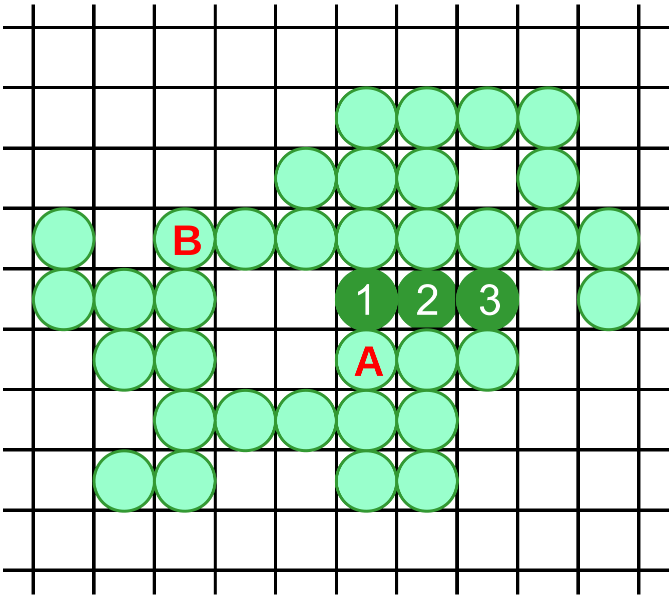
Span Flood-Fill Example



Stack:

x

Span Flood-Fill Example

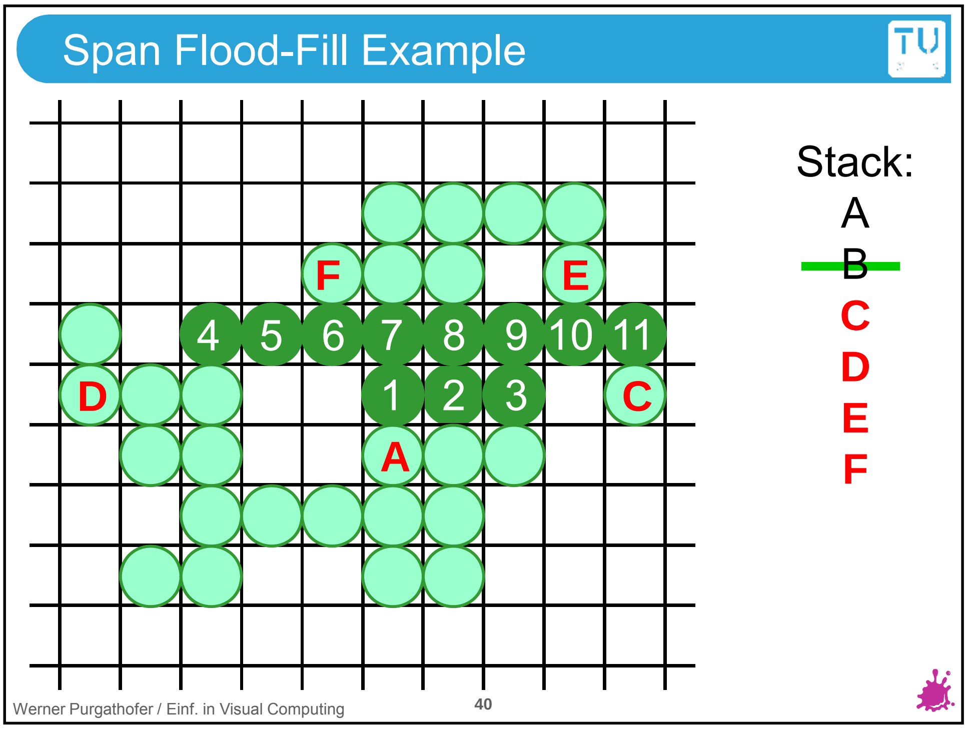


Stack:

~~x~~

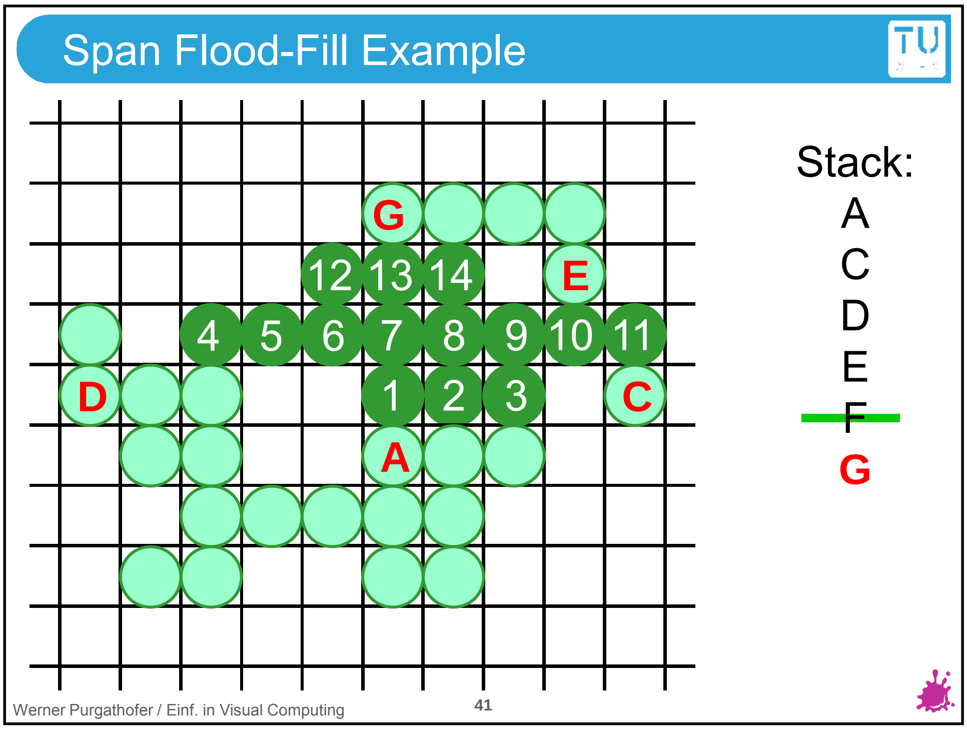
A

B



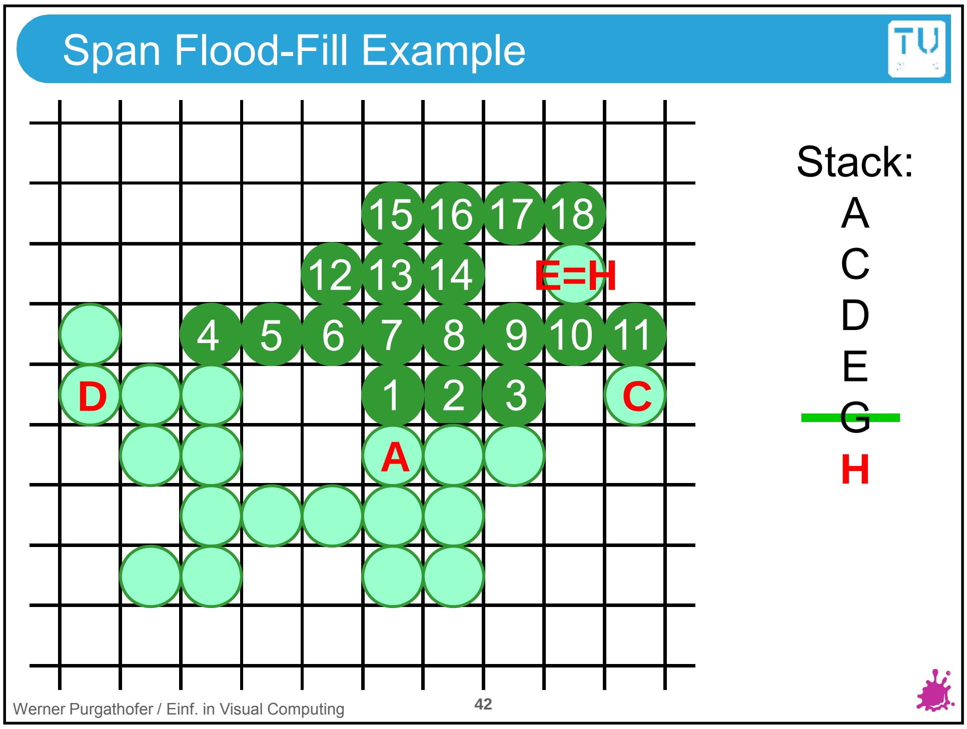
Span Flood-Fill Example

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



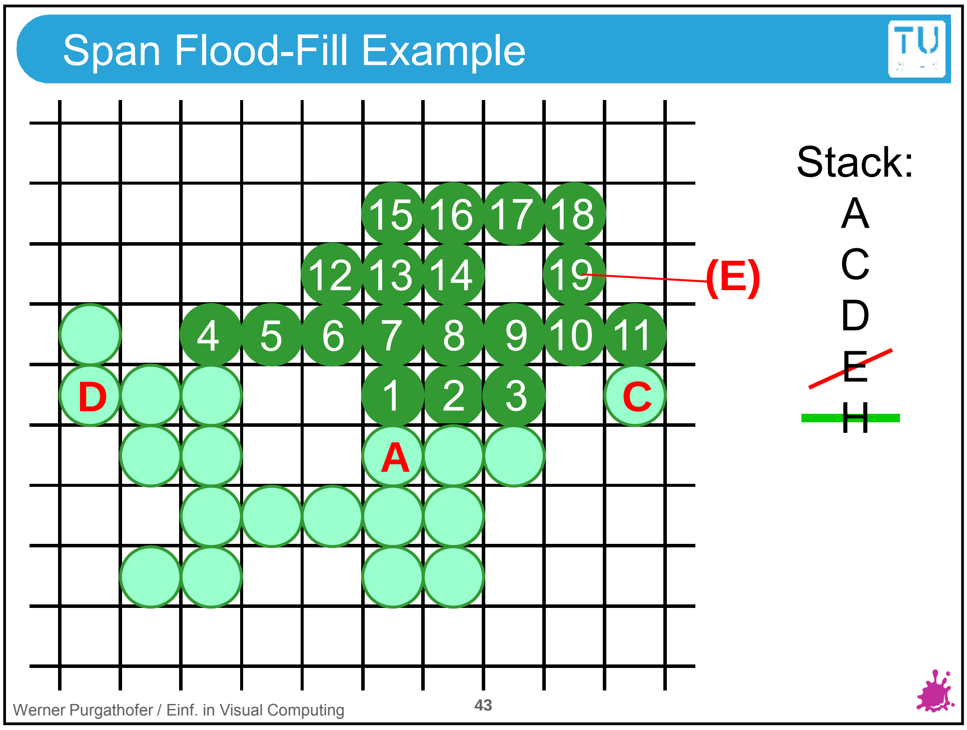
Span Flood-Fill Example

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



Span Flood-Fill Example

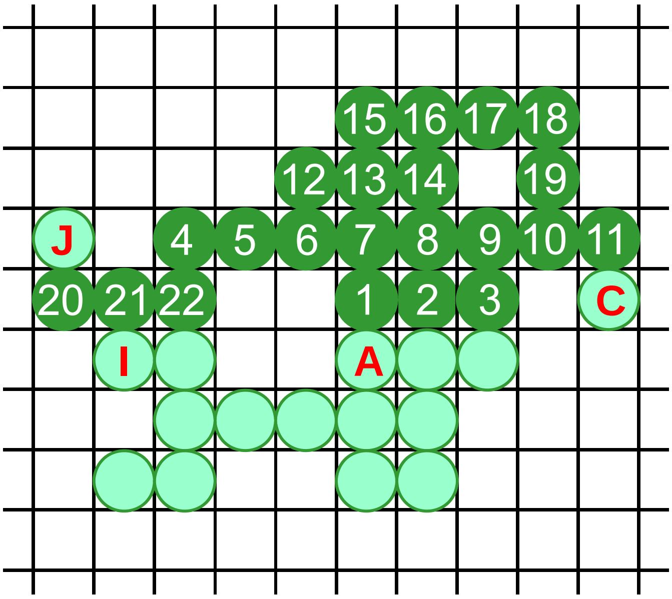
Stack:
A
C
D
E
~~G~~
H



Span Flood-Fill Example

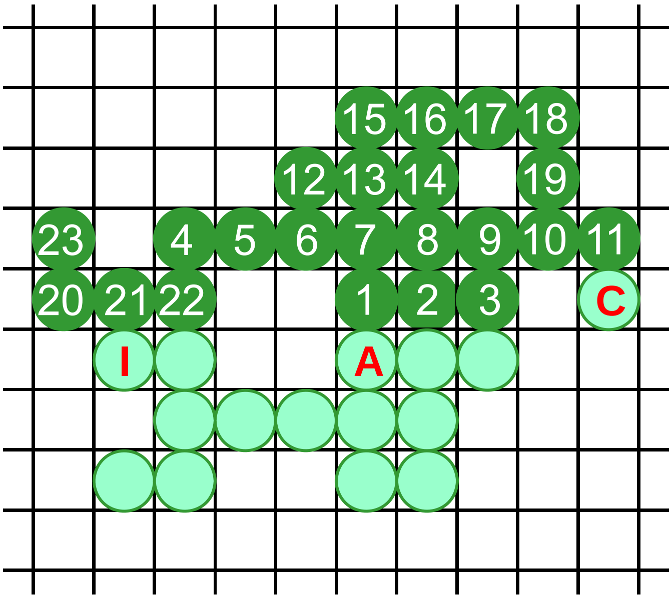
Stack:
A
C
D
~~E~~
~~H~~

Span Flood-Fill Example

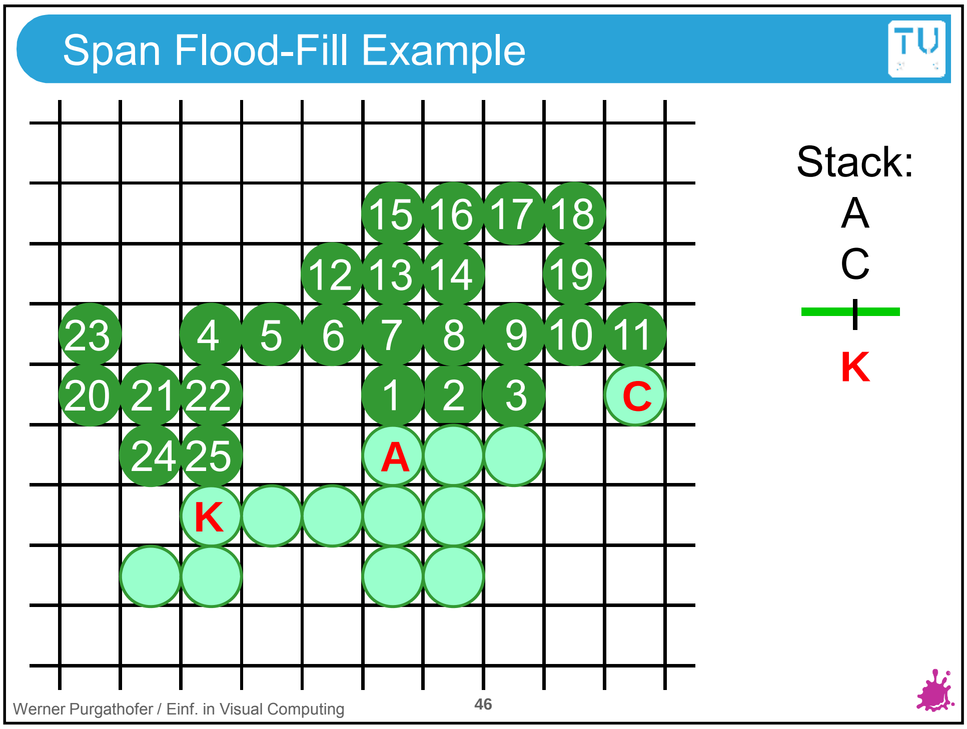


Stack:
A
C
~~D~~
I
J

Span Flood-Fill Example

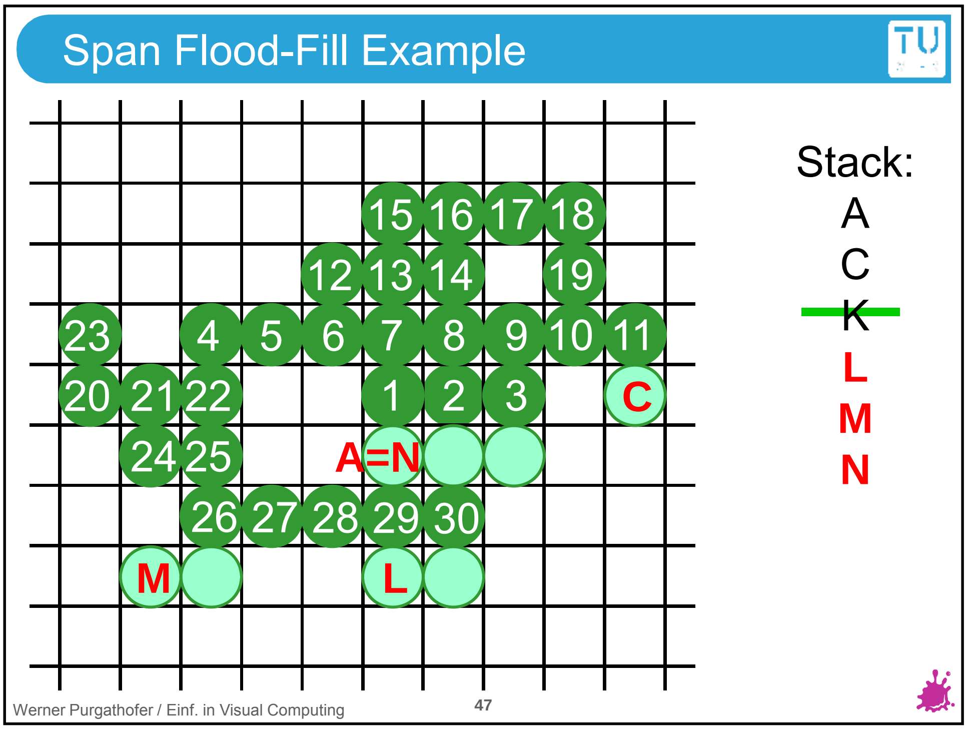


Stack:
A
C
I
~~J~~



Span Flood-Fill Example

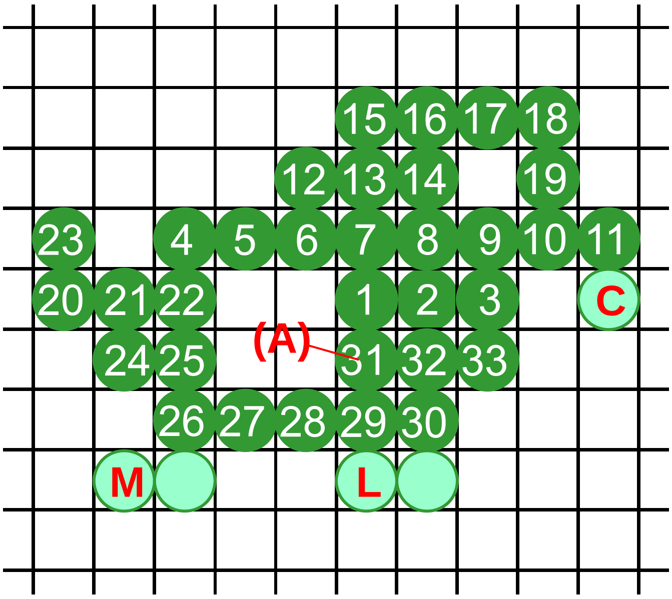
Stack:
A
C
~~I~~
K



Span Flood-Fill Example

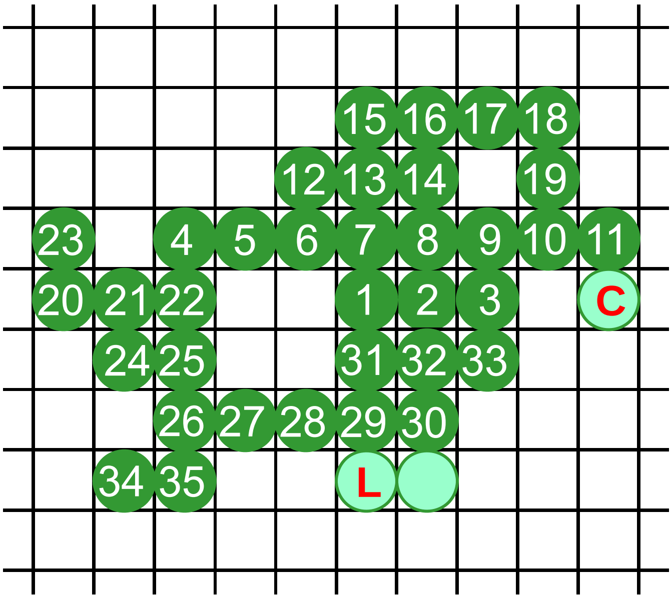
Stack:
A
C
~~K~~
L
M
N

Span Flood-Fill Example



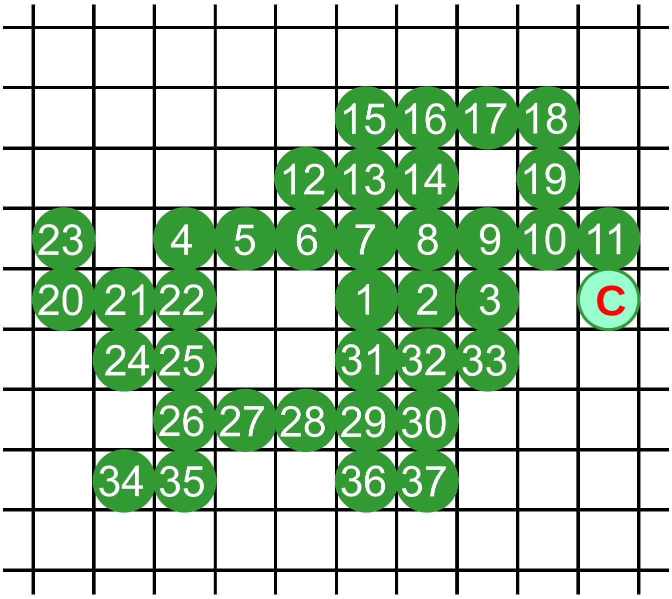
Stack:
~~A~~
C
L
M
~~N~~

Span Flood-Fill Example



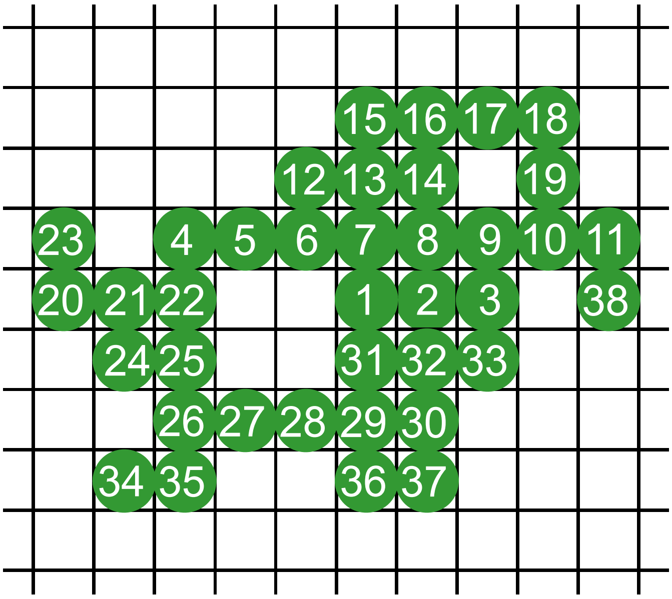
Stack:
C
L
~~M~~

Span Flood-Fill Example



Stack:
C
~~L~~

Span Flood-Fill Example



Stack:
~~C~~

finished!