

Computational Geometry

Winter term 2016/17

Triangulating Polygons

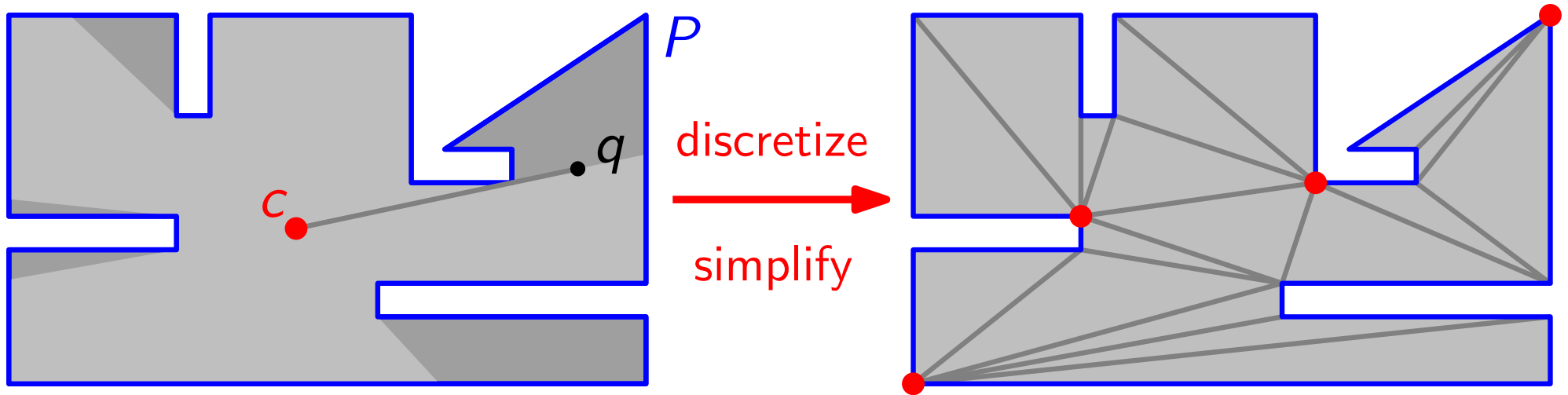
or

Guarding Art Galleries

Lecture #3

Guarding an Art Gallery

Given a *simple* polygon P (i.e., no holes, no self-intersection)...



Observation. Camera c “sees” a star-shaped region

Definition. A pt $q \in P$ is *visible* from $c \in P$ if $\overline{qc} \subseteq P$.

Aim: Use few cameras! *But minimizing them is NP-hard...*

Theorem.

1. Every simple polygon can be triangulated.
2. Any triangulation of a simple polygon with n vertices consists of $n - 2$ triangles.

How can we prove these?

The Art Gallery Theorem

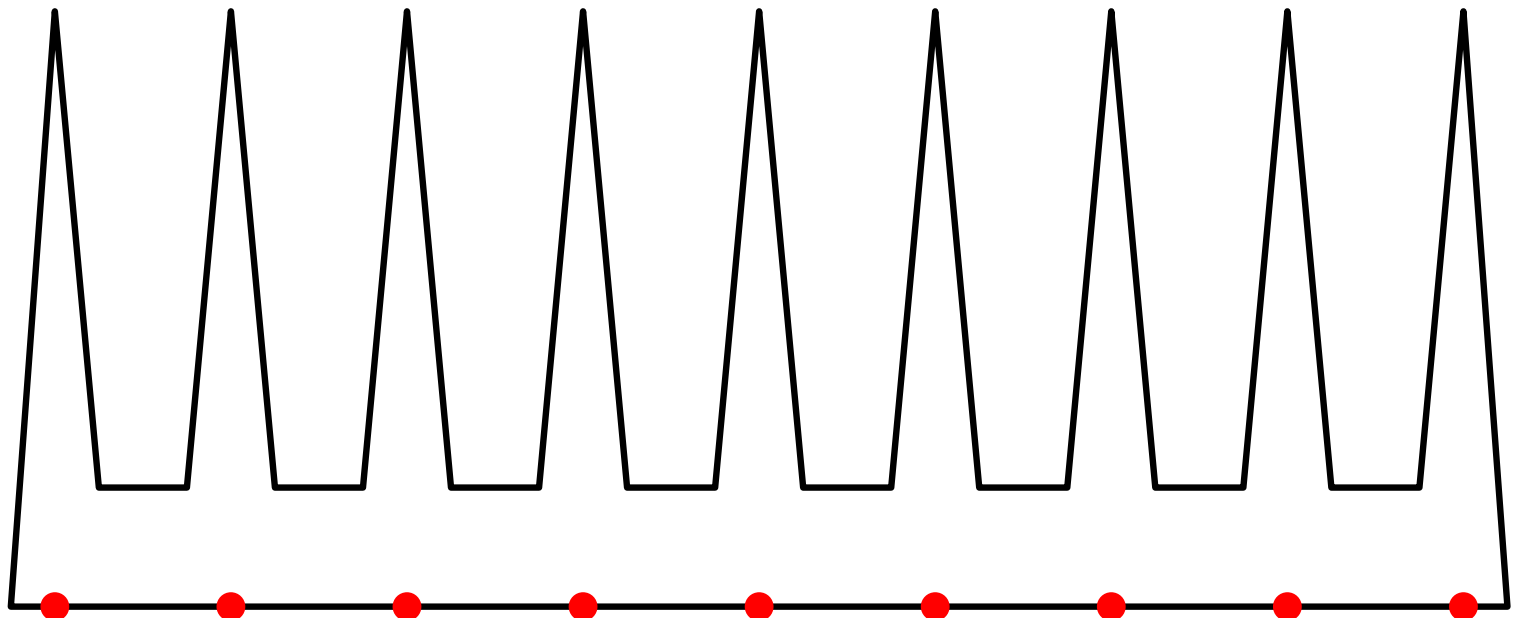
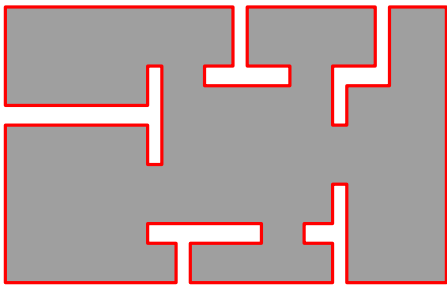
[Chvátal '75]

Theorem. For surveilling a simple polygon with n vertices, $\lfloor n/3 \rfloor$ cameras are **sometimes necessary** and always sufficient.

Exercise.

Find, for arbitrarily large n , a **rectilinear** polygon with n vertices, where $\approx n/3$ cameras are necessary.

$n/4$



The Art Gallery Theorem

[Chvátal '75]

Theorem. For surveilling a simple polygon with n vertices, $\lfloor n/3 \rfloor$ cameras are sometimes necessary and always sufficient. [black board]

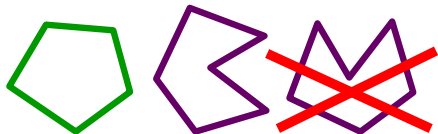
To do: Find algorithm for triangulating a simple polygon!

Brute force: follow existence proof, using recursion
running time: $O(n^2)$

Faster triangulation in two steps:

n -vtx polygon $\xrightarrow{O(n \log n)}$ "nice" pieces, n' vtc $\xrightarrow{O(n')}$ n'' triangles

Definition. A polygon P is *y-monotone* if, for any horizontal line ℓ , $\ell \cap P$ is connected.



Partitioning a Polygon into Monotone Pieces

Idea: Classify vertices of given simple polygon P

– *turn* vertices:

vertical component of walking direction changes

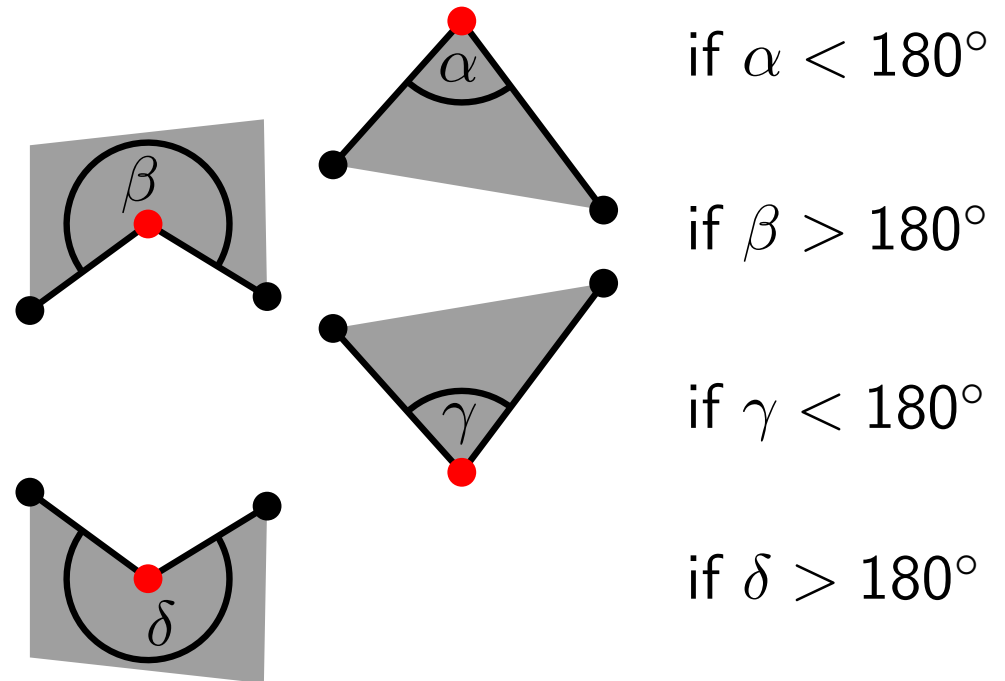
● *start* vertex

● *split* vertex

● *end* vertex

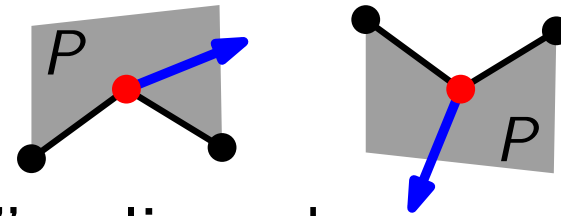
● *merge* vertex

– *regular* vertices



Lemma: Let P be a simple polygon. Then P is y -monotone $\Leftrightarrow P$ has neither split vertices nor merge vertices.

Towards an Algorithm

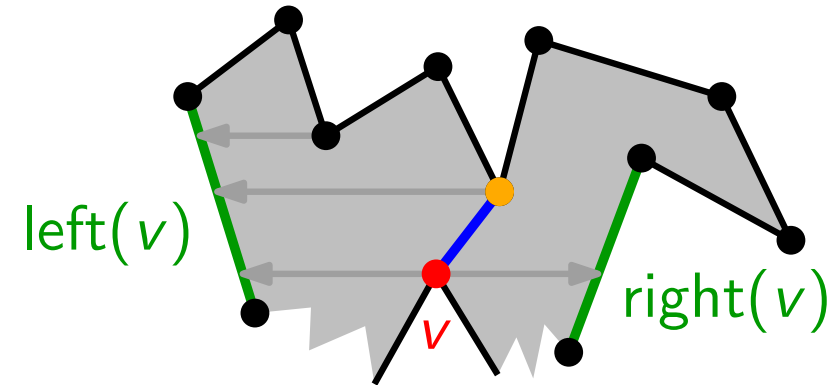


Idea: Add **diagonals** to “destroy” split and merge vertices.

Problem: Diagonals must not cross

- each other
- edges of P

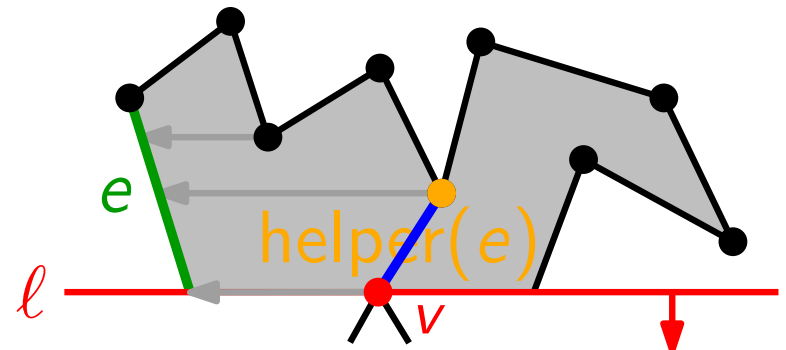
1) Treating split vertices



Connect v to vertex w^* having minimum y -coordinate among all vertices w above v and with $\text{left}(w) = \text{left}(v)$.

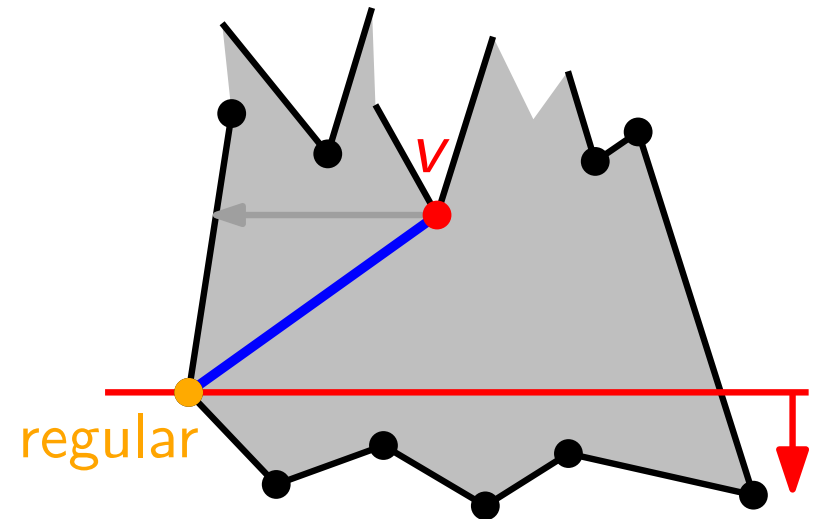
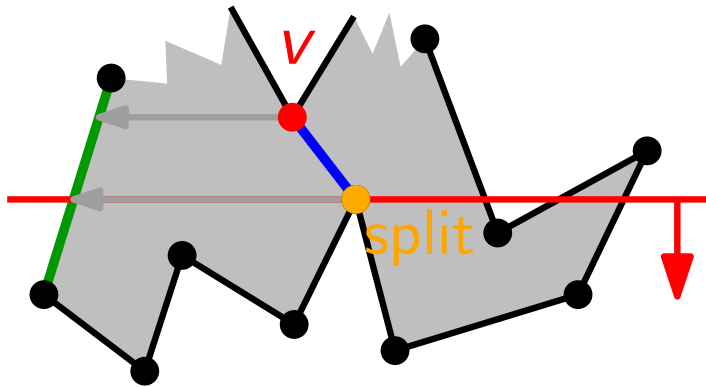
Think of a sweep-line algorithm:

Connect v to $\text{helper}(\text{left}(v))$.



An Algorithm

2) Treating merge vertices



makeMonotone(polygon P)

$\mathcal{D} \leftarrow \text{DCEL}(V(P), E(P))$

$Q \leftarrow$ priority queue on $V(P)$

$\mathcal{T} \leftarrow$ empty bin. search tree

while $Q \neq \emptyset$ **do**

$v \leftarrow Q.\text{extractMax}()$

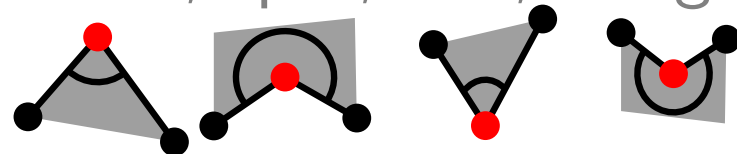
 type \leftarrow type of vertex v

 handleTypeVertex(v)

return DCEL \mathcal{D}

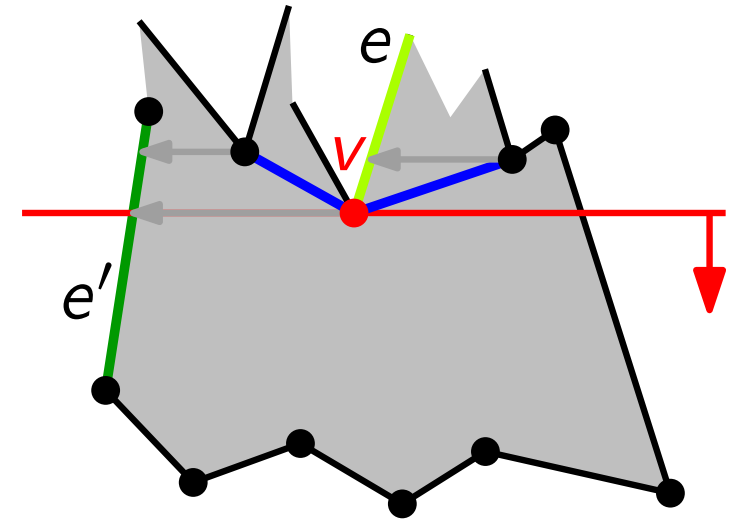
{ doubly-connected edge list:
data structure for planar subdivisions
 $(x, y) < (x', y') \Leftrightarrow$
 $y > y' \vee (y = y' \wedge x < x')$

\in start, split, end, merge, regular



An Algorithm

2) Treating merge vertices



```

makeMonotone(polygon  $P$ )
 $\mathcal{D} \leftarrow \text{DCEL}(V(P), E(P))$ 
 $Q \leftarrow$  priority queue on  $V(P)$ 
 $\mathcal{T} \leftarrow$  empty bin. search tree
while  $Q \neq \emptyset$  do
     $v \leftarrow Q.\text{extractMax}()$ 
    type  $\leftarrow$  type of vertex  $v$ 
    handleTypeVertex( $v$ )
return DCEL  $\mathcal{D}$ 
  
```

```

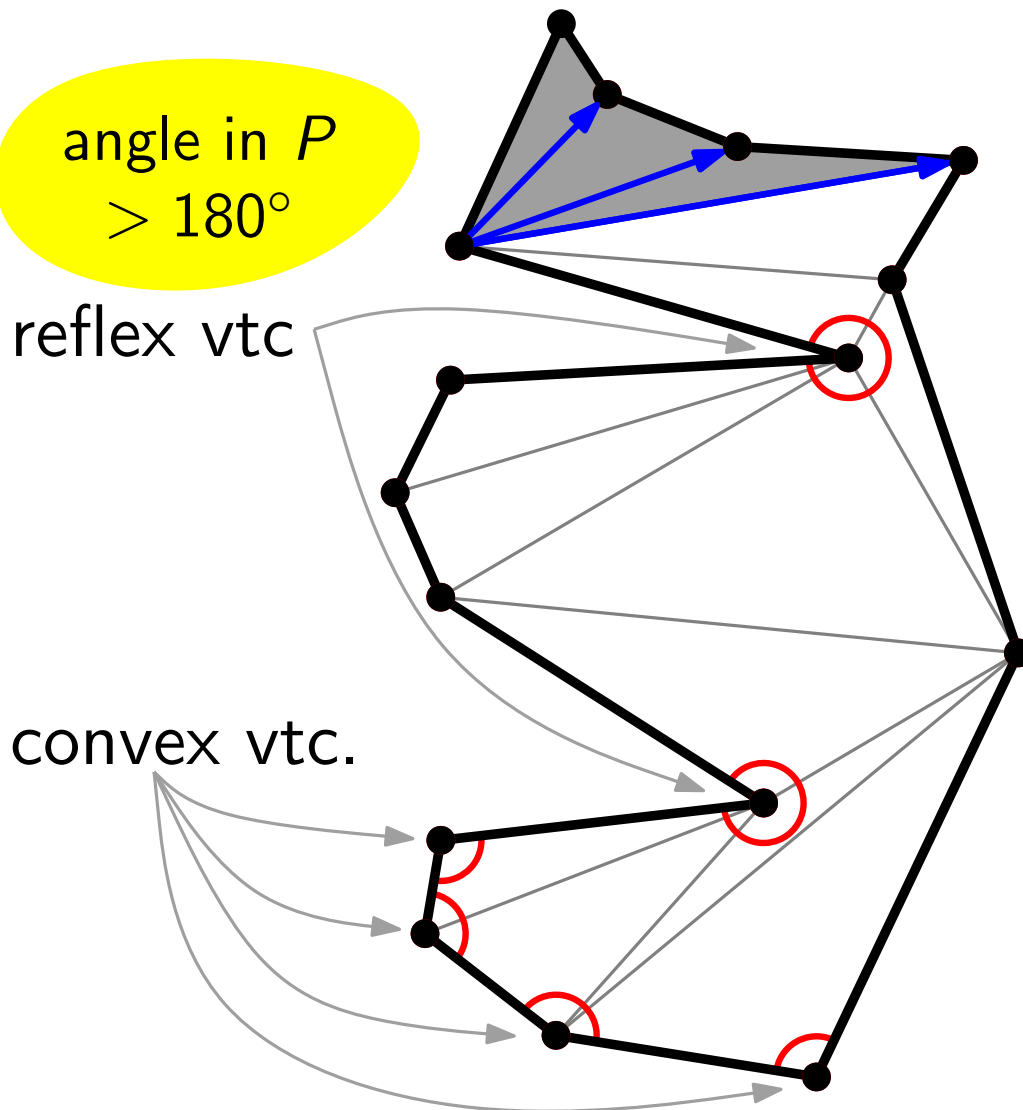
handleMergeVertex(vertex  $v$ )
 $e \leftarrow$  edge following  $v$  cw
if helper( $e$ ) merge vtx then
     $\mathcal{D}.\text{insert}(\text{diag}(v, \text{helper}(e)))$ 
 $\mathcal{T}.\text{delete}(e)$ 
 $e' \leftarrow \mathcal{T}.\text{edgeLeftOf}(v)$ 
if helper( $e'$ ) merge vtx then
     $\mathcal{D}.\text{insert}(\text{diag}(v, \text{helper}(e')))
  \text{helper}(e') \leftarrow v$ 
```


Analysis

- Lemma.** `makeMonotone()` adds a set of non-intersecting diagonals to P such that P is partitioned into y -monotone subpolygons.
- Lemma.** A simple polygon with n vertices can be subdivided into y -monotone polygons in $O(n \log n)$ time.

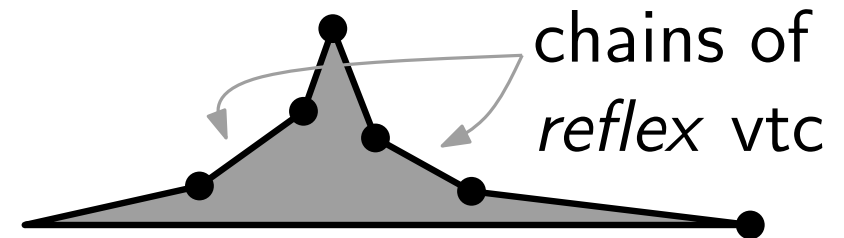
Triangulating a y -Monotone Polygon P

Approach: greedy, going from top to bottom

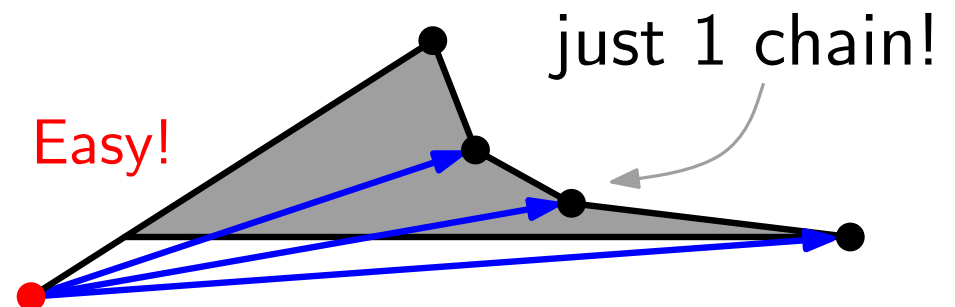


Invariant?

The part of P that we have seen but not yet triangulated is a *funnel*.



Our funnels are special:



Algorithm

Running time? $\Theta(n)$

TriangulateMonotonePolygon(Polygon P as circular vertex list)

merge left and right chain \rightarrow sequence u_1, \dots, u_n with $y_1 \geq \dots \geq y_n$

Stack S ; $S.\text{push}(u_1)$; $S.\text{push}(u_2)$

for $j \leftarrow 3$ **to** $n - 1$ **do**

if u_j and $S.\text{top}()$ lie on different chains **then**

while not $S.\text{empty}()$ **do**

$v \leftarrow S.\text{pop}()$

if not $S.\text{empty}()$ **then** draw diag. (u_j, v)

$S.\text{push}(u_{j-1})$; $S.\text{push}(u_j)$

else

$v \leftarrow S.\text{pop}()$

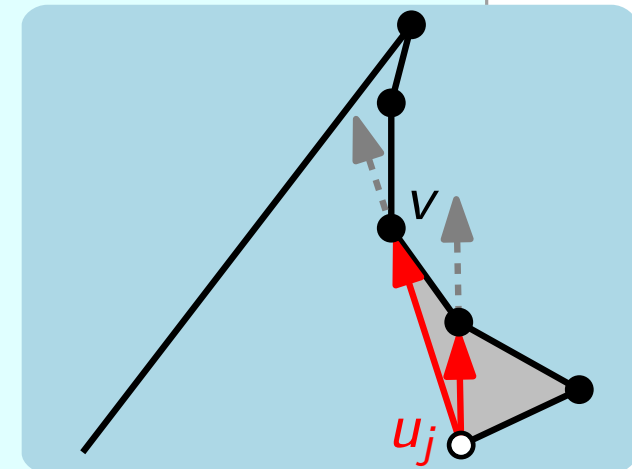
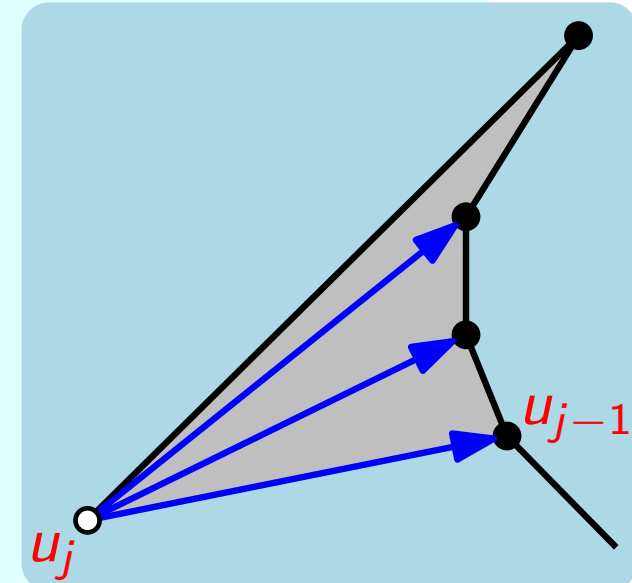
while not $S.\text{empty}()$ **and** u_j sees $S.\text{top}()$ **do**

$v \leftarrow S.\text{pop}()$

draw diagonal (u_j, v)

$S.\text{push}(v)$; $S.\text{push}(u_j)$

draw diagonals from u_n to all vtc on S except first and last one



Summary

n -vtx polygon \longrightarrow "nice" pieces
 $O(n \log n)$

n' vtc \longrightarrow n'' triangles
 $O(n')$

Lemma.



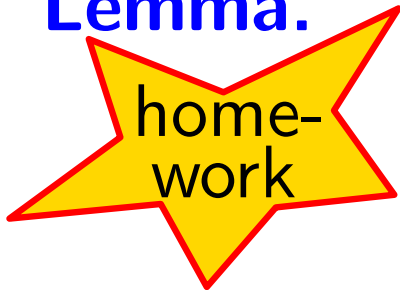
A simple polygon with n vertices can be subdivided into y -monotone polygons in $O(n \log n)$ time.

Lemma.



A y -monotone polygon with n vertices can be triangulated in $O(n)$ time.

Lemma.



Subdividing a simple polygon with n vertices by drawing d (pairwise non-crossing) diagonals yields $d + 1$ simple polygons of total complexity $O(n)$.

Theorem.

A simple polygon with n vertices can be triangulated in $O(n \log n)$ time.

Is this it?

Tarjan & van Wyk [1988]:

$O(n \log \log n)$

Kirkpatrick, Klawe, Tarjan [1992]

Clarkson, Tarjan, van Wyk [1989]:

$O(n \log^* n)$

Seidel [1991]: *randomized*

Chazelle [1991]:

$O(n)$