Computational Geometry

Lecture 3:
Guarding Art Galleries
or
Triangulating Polygons

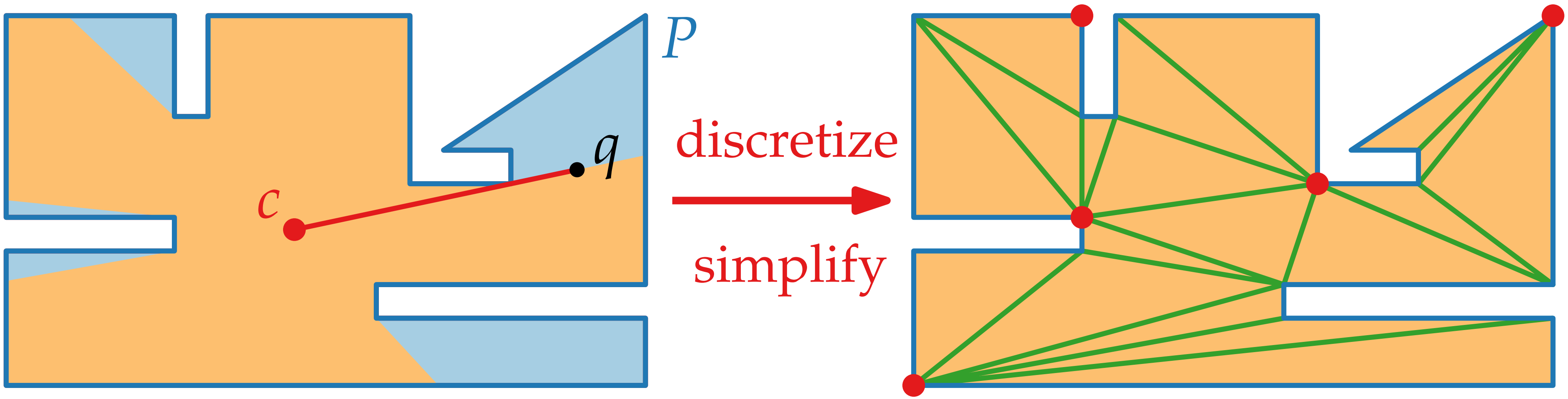
Part I:
The Art Gallery Problem

Philipp Kindermann

Winter Semester 2020

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 $\overline{qc} \subseteq P$.

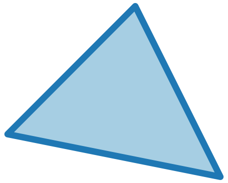
Aim: Use few cameras! *But minimizing them is NP-hard…*

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

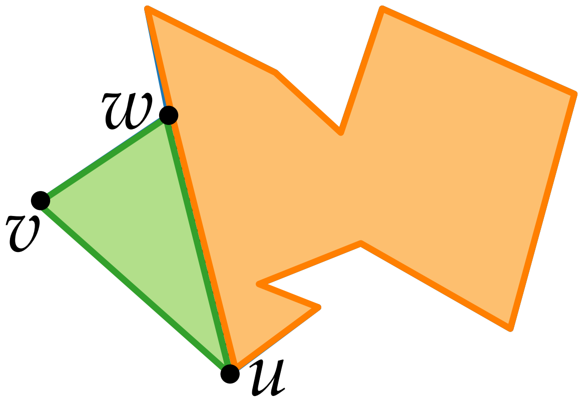
How can we prove these?

Existence of Triangulation

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

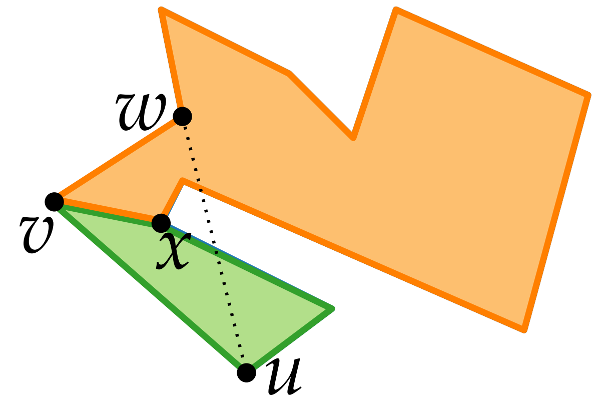
$n = 3$:  1 triangle ✓

$3, \ldots, n-1 \to n$:



3 vtcs $\Rightarrow$ 1 triangle
$n{-}1$ vtcs $\Rightarrow$ $n{-}3$ triangles
$\Rightarrow$ $n{-}2$ triangles
✓

x furthest from uw



m vtcs $\Rightarrow$ $m{-}2$ triangles
$n{-}m{+}2$ vtcs $\Rightarrow$ $n{-}m$ triangles
$\Rightarrow$ $n{-}2$ triangles
✓

Computational Geometry

Lecture 3:
Guarding Art Galleries
or
Triangulating Polygons

Part II:
The Art Gallery Theorem

Philipp Kindermann

Winter Semester 2020

The Art Gallery Theorem

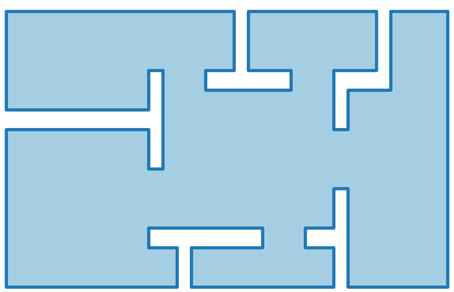
[Chvátal '75]

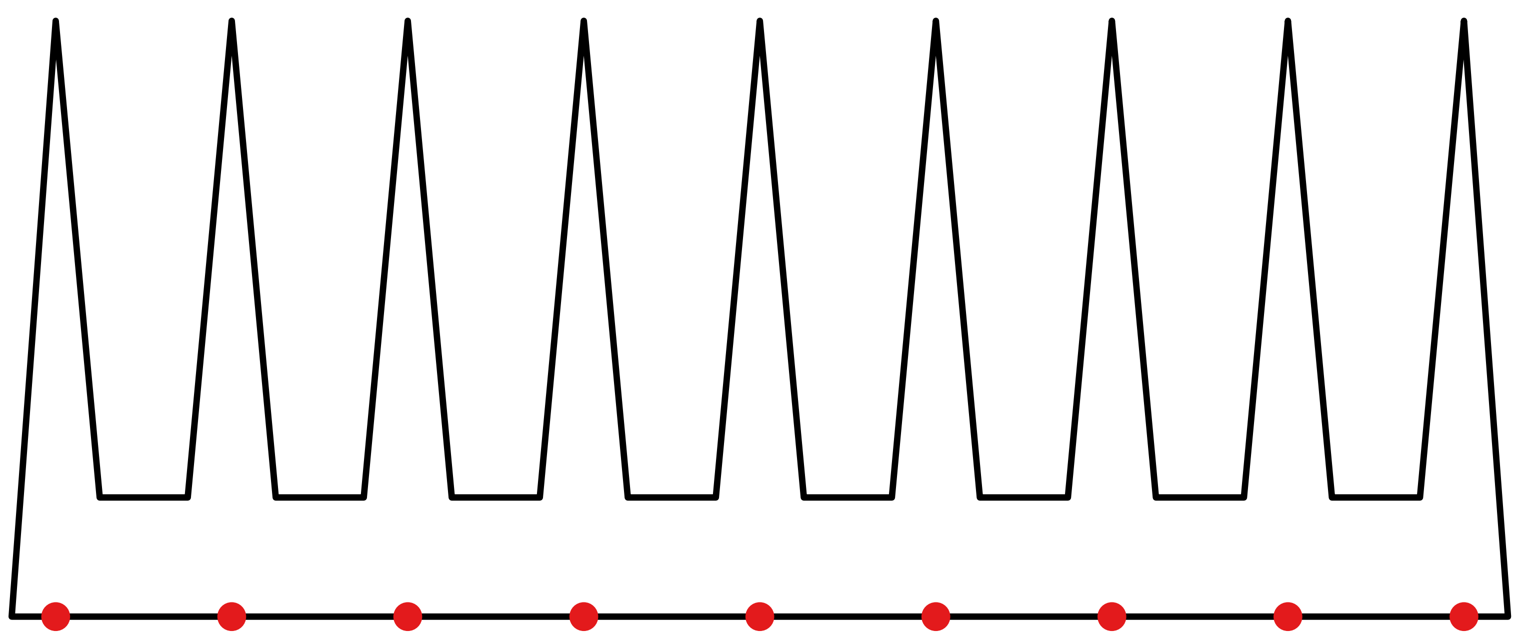
Theorem. For surveilling a simple polygon with n vertices, $\lfloor n/3 \rfloor$ cameras are sometimes necessary and always sufficient.

rectilinear

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

$n/4$





[dBCvKO'08]

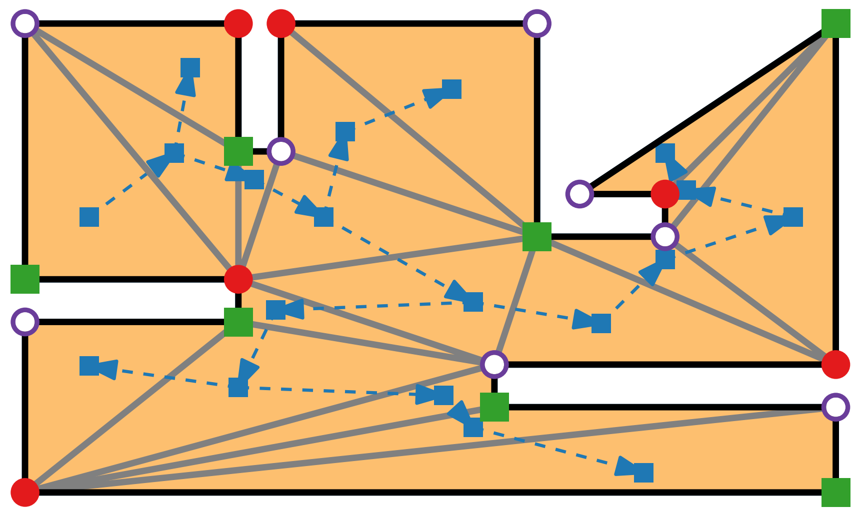
The Art Gallery Theorem

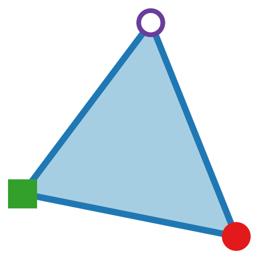
[Chvátal '75]

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

3-color the vtcs

Traverse the dual tree





Pick "smallest" color

Computational Geometry

Lecture 3:
Guarding Art Galleries
or
Triangulating Polygons

Part III:
Partitioning a Polygon into y-monotone Pieces

Philipp Kindermann

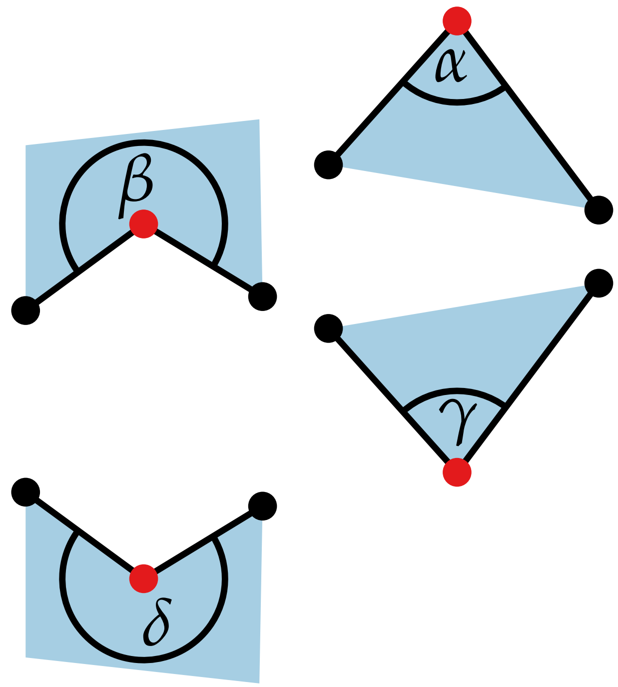
Winter Semester 2020

Part. a Polygon into Monotone Pieces

Idea: Classify vertices of given simple polygon P

– *turn* vertices:
vertical component of walking direction changes

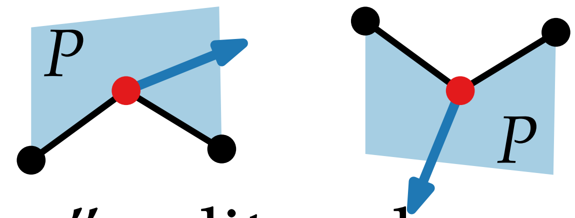
- *start* vertex — if $\alpha < 180^\circ$
- *split* vertex — if $\beta > 180^\circ$
- *end* vertex — if $\gamma < 180^\circ$
- *merge* vertex — if $\delta > 180^\circ$



– *regular* vertices

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

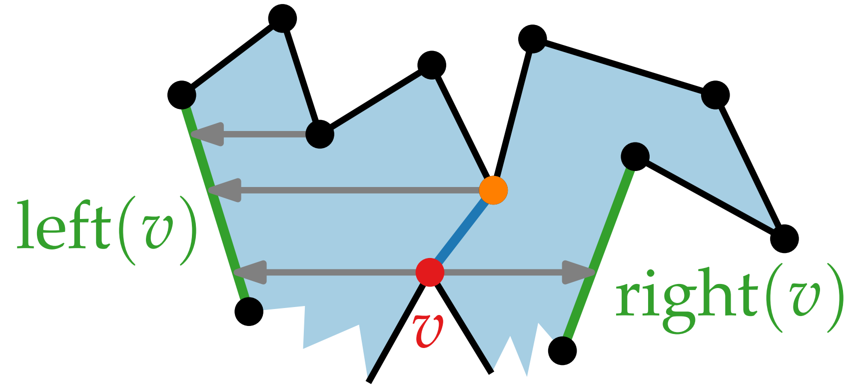
Towards an Algorithm



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

Problem: Diagonals must not cross – each other
– edges of P

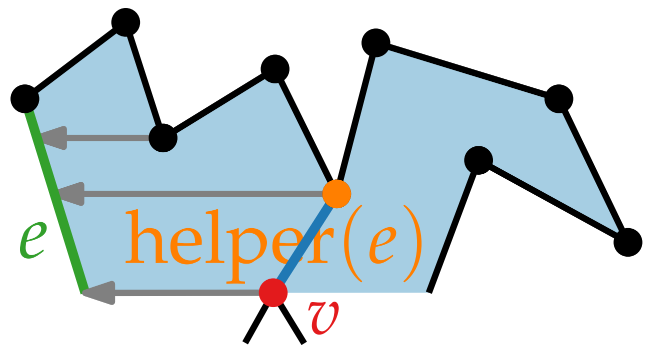
1) Treating split vertices



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

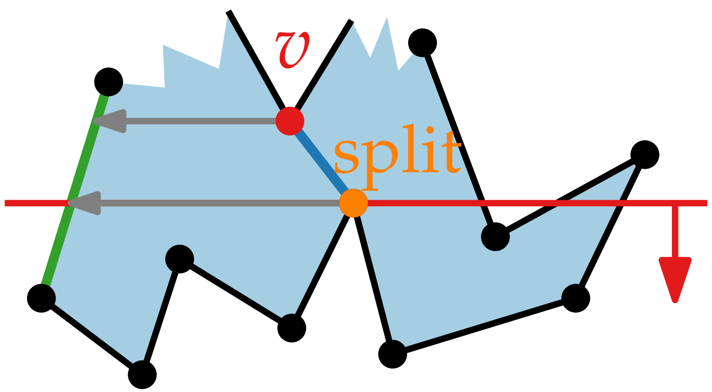
Think of a sweep-line algorithm:

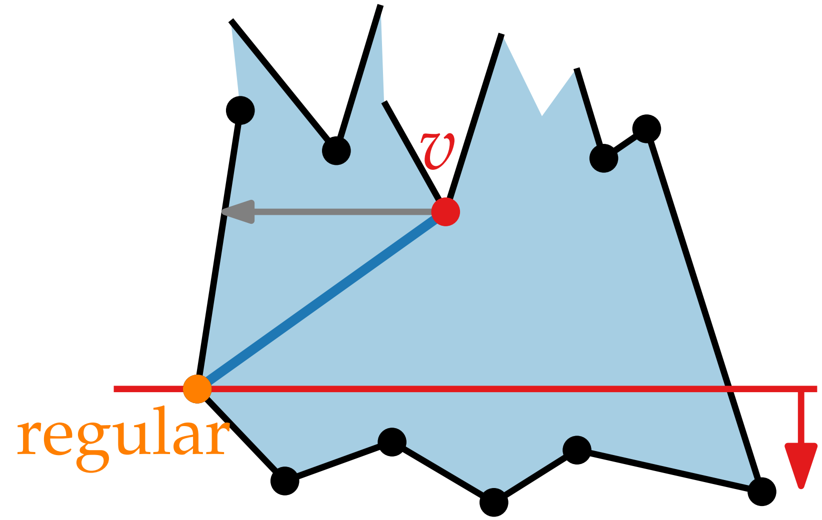
Connect v to helper(left(v)).



An Algorithm

2) Treating merge vertices

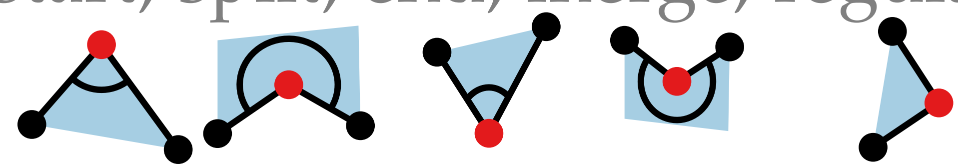




makeMonotone(polygon P**)**
$\mathcal{D} \leftarrow \mathrm{DCEL}(V(P), E(P))$
$\mathcal{Q} \leftarrow$ priority queue on $V(P)$
$\mathcal{T} \leftarrow$ empty bin. search tree
while $\mathcal{Q} \neq \emptyset$ **do**
  $v \leftarrow \mathcal{Q}$.extractMax()
  type $\leftarrow$ type of vertex $v \in$ start, split, end, merge, regular
  handleTypeVertex(v)
return DCEL $\mathcal{D}$

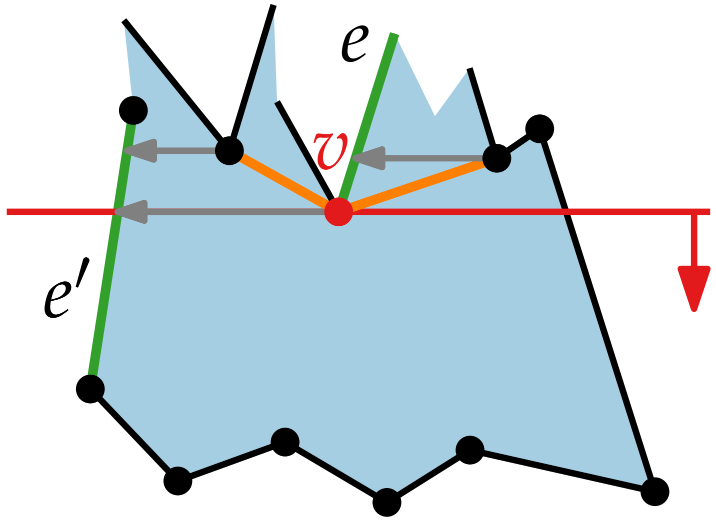
doubly-connected edge list:
data structure for planar subdivisions

$(x, y) \prec (x', y') \;:\Leftrightarrow$
$y > y' \vee (y = y' \wedge x < x')$



An Algorithm

2) Treating merge vertices



makeMonotone(polygon P**)**
$\mathcal{D} \leftarrow$ DCEL$(V(P), E(P))$
$\mathcal{Q} \leftarrow$ priority queue on $V(P)$
$\mathcal{T} \leftarrow$ empty bin. search tree
while $\mathcal{Q} \neq \emptyset$ **do**
  $v \leftarrow \mathcal{Q}$.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**
  $\mathcal{D}$.insert(diag(v, helper(e)))
$\mathcal{T}$.delete(e)
$e' \leftarrow \mathcal{T}$.edgeLeftOf$(v)$
if helper(e') merge vtx **then**
  $\mathcal{D}$.insert(diag(v, helper(e')))

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.

Computational Geometry

Lecture 3:
Guarding Art Galleries
or
Triangulating Polygons

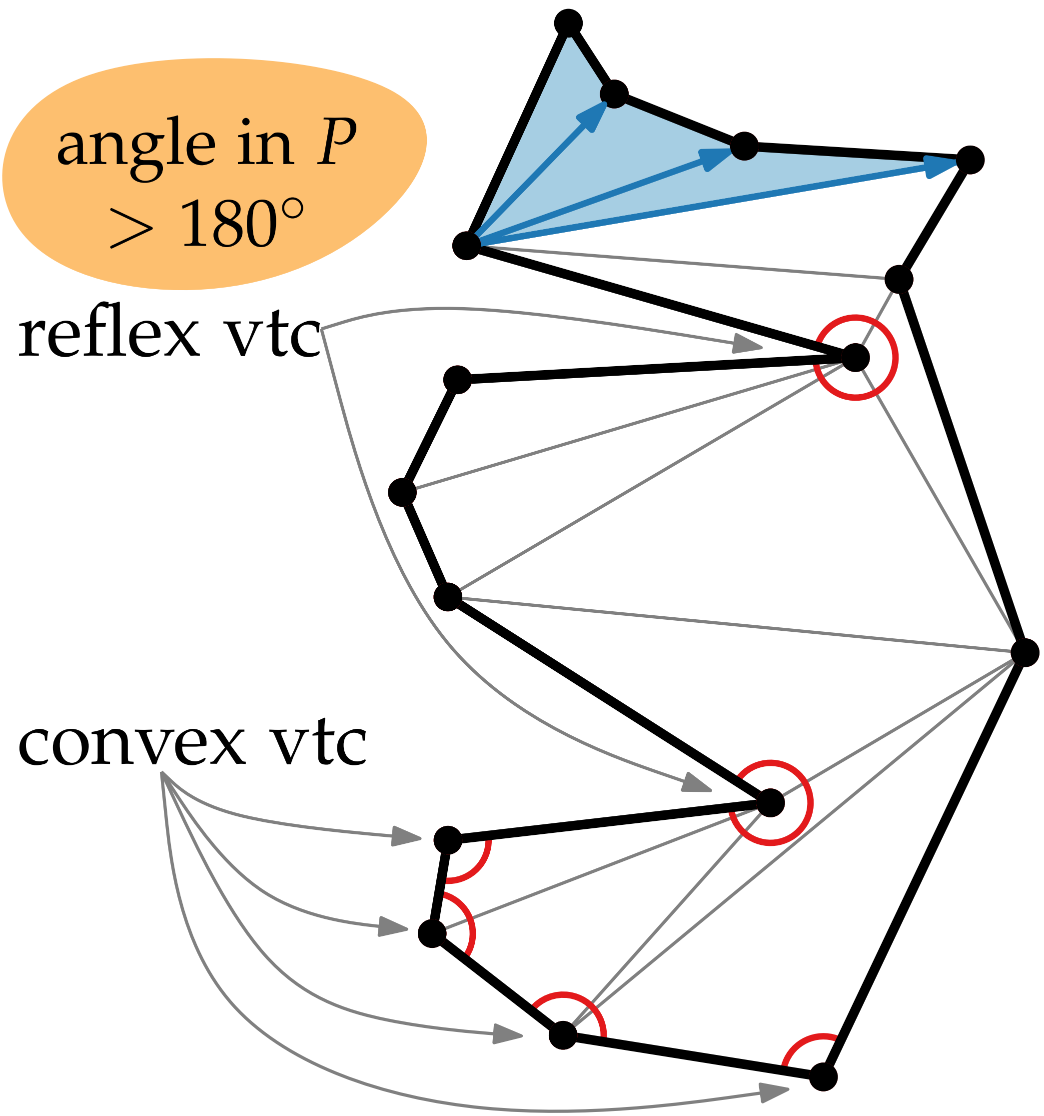
Part IV:
Triangulating a y-monotone Polygon

Philipp Kindermann

Winter Semester 2020

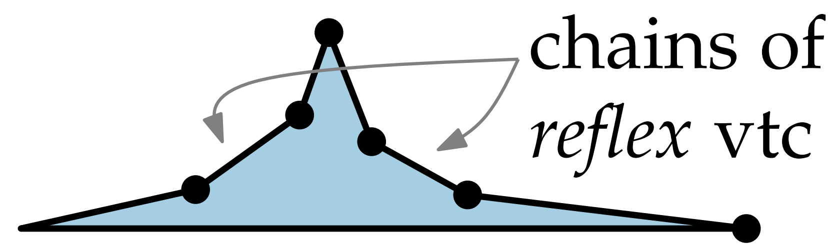
Triangulating a y-Monotone Polygon P

Approach: greedy, going from top to bottom

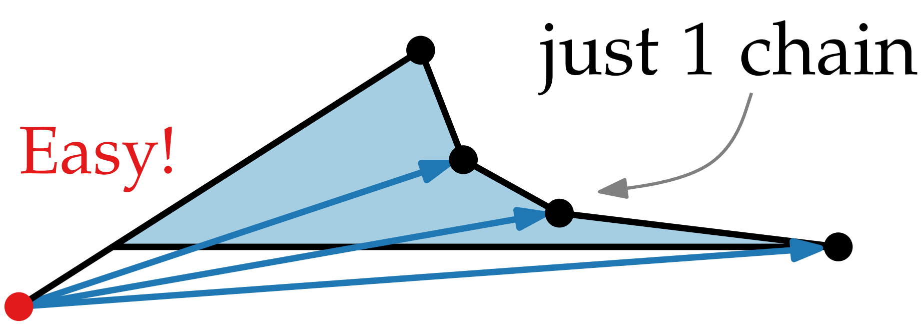


Invariant?

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



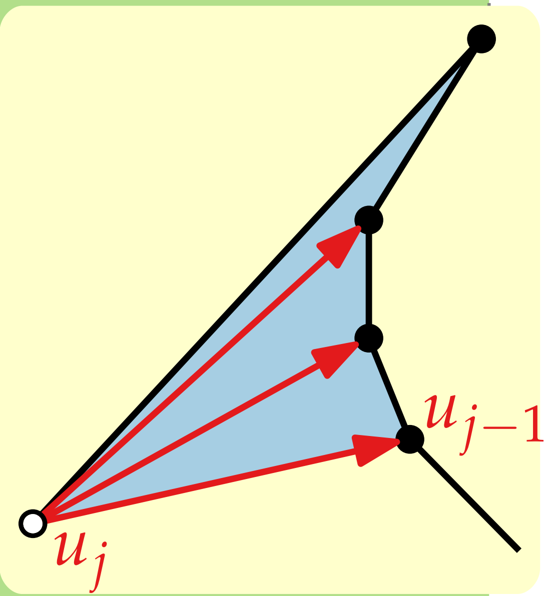
Our funnels are special:

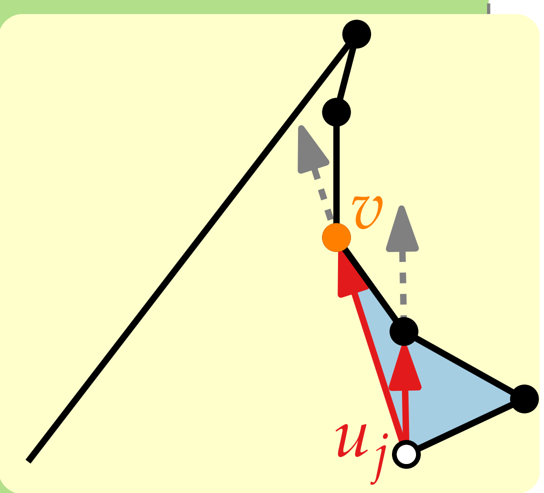


Algorithm

Running time? $\Theta(n)$

TriangulateMonotonePolygon(Polygon P as circular vertex list)
merge left and right chain $\rightarrow$ seq. $u_1, \ldots, u_n$ with $y_1 \geq \ldots \geq y_n$
Stack S; S.push(u_1); S.push(u_2)
for $j \leftarrow 3$ **to** $n-1$ **do**
  if u_j and S.top() lie on different chains **then**
    while not S.empty() **do**
      $v \leftarrow S$.pop()
      if not S.empty() **then** draw diag. (u_j, v)
    S.push(u_{j-1}); S.push(u_j)
  else
    $v \leftarrow S$.pop()
    while not S.empty() **and** u_j sees S.top() **do**
      $v \leftarrow S$.pop()
      draw diagonal (u_j, v)
    S.push(v); S.push(u_j)
draw diagonals from u_n to all vtc on S except first and last one





Summary

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

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

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

Lemma. home-work
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]: $O(n \log \log n)$
Clarkson, Tarjan, van Wyk [1989]: $O(n \log^* n)$
Chazelle [1991]: $O(n)$

Kirkpatrick, Klawe, Tarjan [1992]
Seidel [1991]: *randomized*