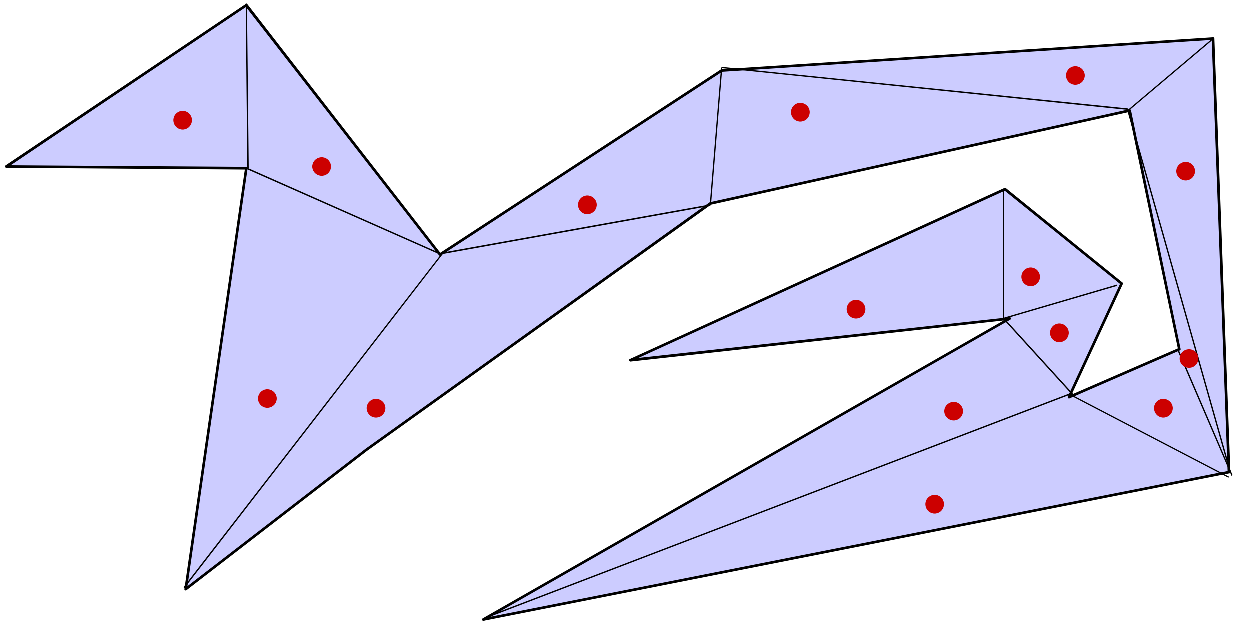
CMPS 3130/6130 Computational Geometry
Spring 2015



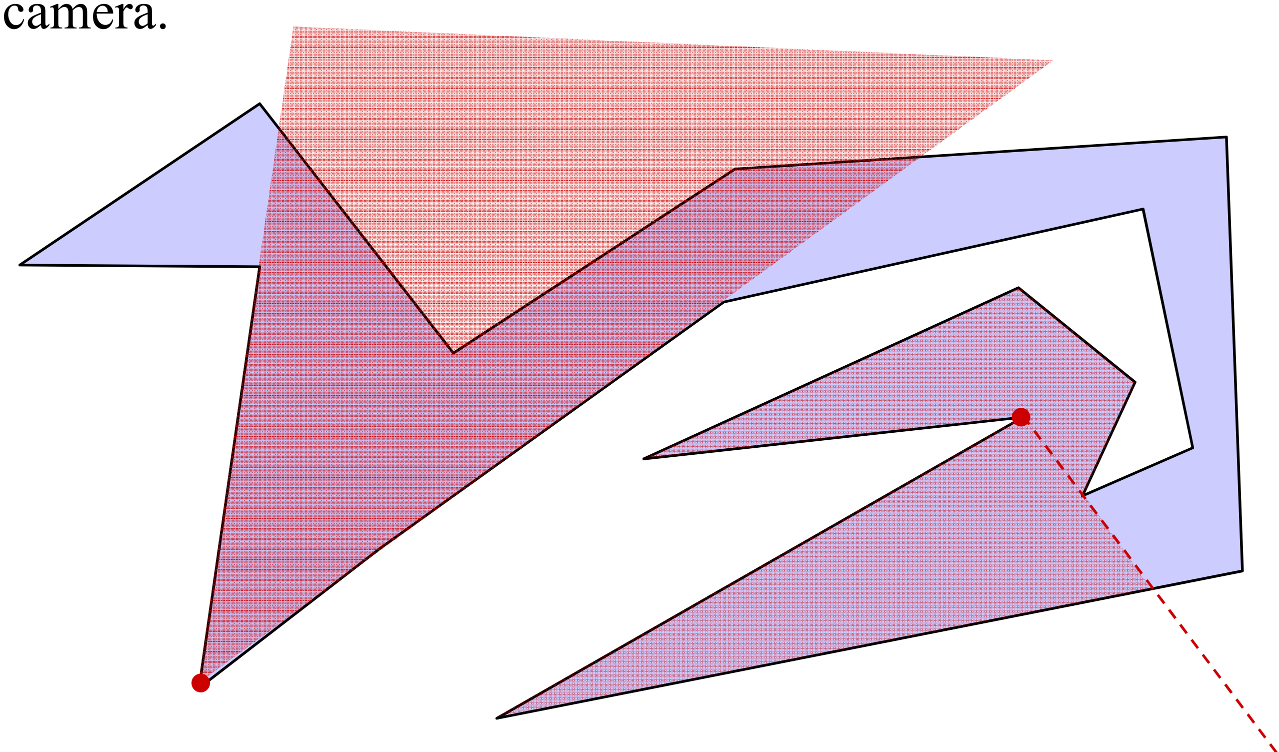
Triangulations and Guarding Art Galleries

Carola Wenk

Guarding an Art Gallery

Region enclosed by simple polygonal chain that does not self-intersect.

- **Problem:** Given the floor plan of an art gallery as a simple polygon *P* in the plane with *n* vertices. Place (a small number of) cameras/guards on vertices of *P* such that every point in *P* can be seen by some camera.

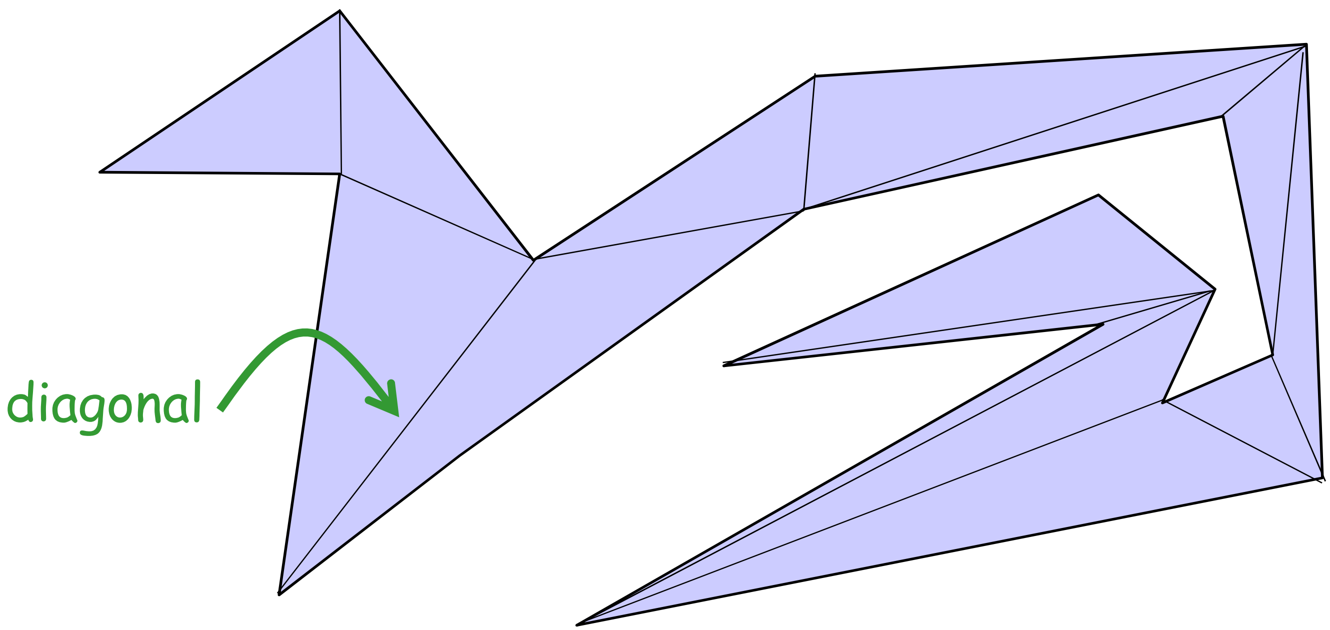


Guarding an Art Gallery

- There are many different variations:
  - Guards on vertices only, or in the interior as well
  - Guard the interior or only the walls
  - Stationary versus moving or rotating guards
- Finding the minimum number of guards is NP-hard (Aggarwal ’84)
- First subtask: Bound the number of guards that are necessary to guard a polygon in the worst case.

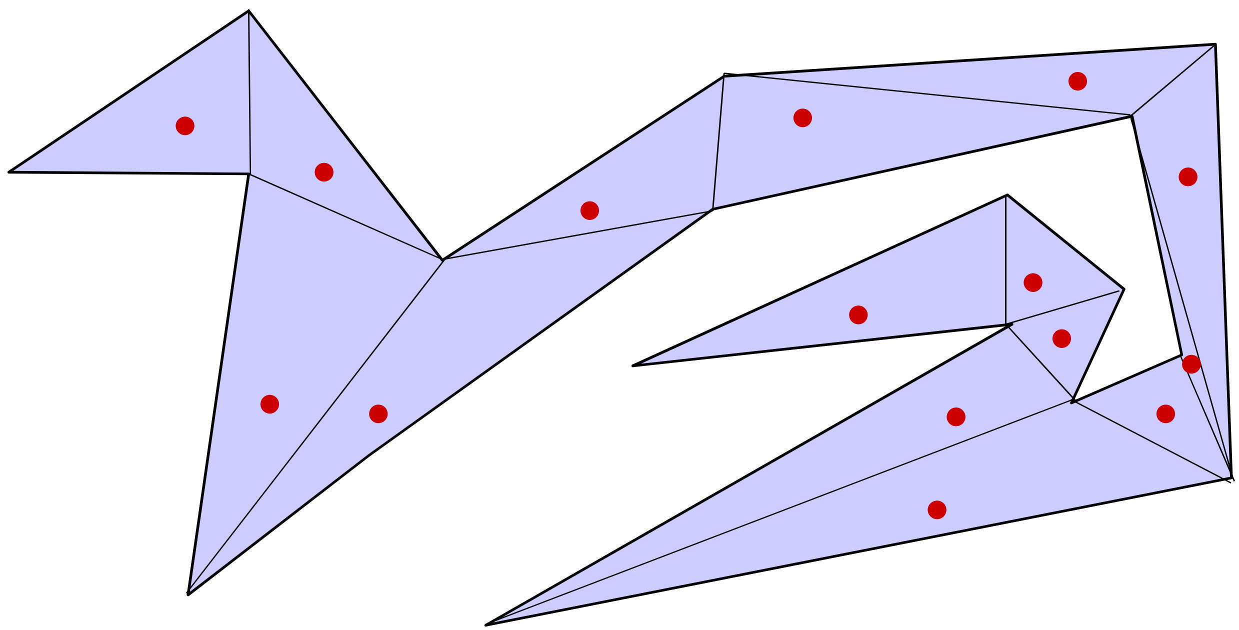
Guard Using Triangulations

- Decompose the polygon into shapes that are easier to handle: triangles
- A **triangulation** of a polygon *P* is a decomposition of *P* into triangles whose vertices are vertices of *P*. In other words, a triangulation is a maximal set of non-crossing diagonals.



Guard Using Triangulations

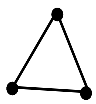
- A polygon can be triangulated in many different ways.
- Guard polygon by putting one camera in each triangle: Since the triangle is convex, its guard will guard the whole triangle.

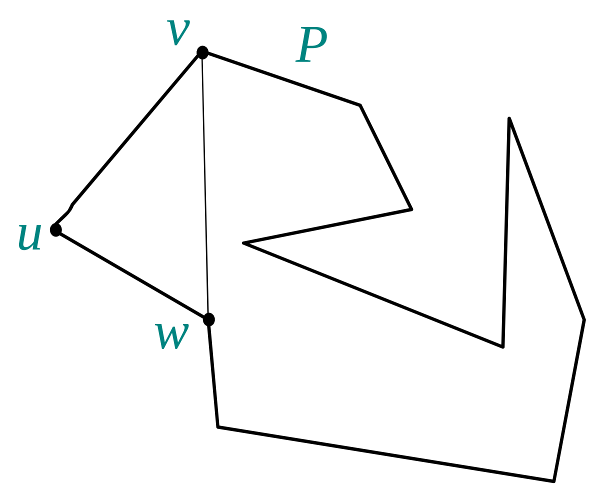


Triangulations of Simple Polygons

Theorem 1: Every simple polygon admits a triangulation, and any triangulation of a simple polygon with n vertices consists of exactly n-2 triangles.

Proof: By induction.

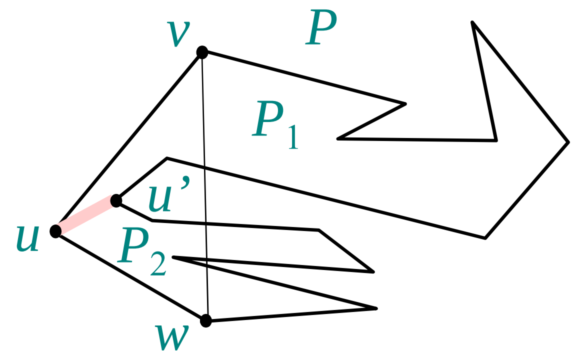
- n=3: 
- n>3: Let u be leftmost vertex, and v and w adjacent to v. If $\overline{vw}$ does not intersect boundary of P: #triangles = 1 for new triangle + (n-1)-2 for remaining polygon = n-2



Triangulations of Simple Polygons

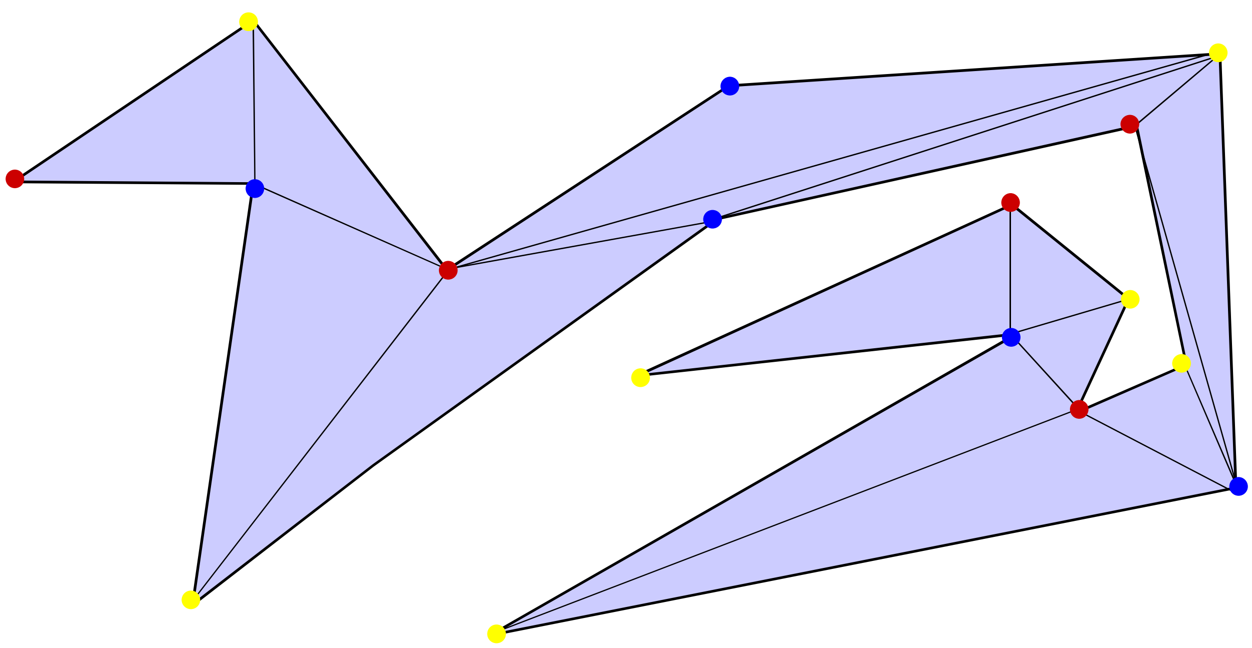
Theorem 1: Every simple polygon admits a triangulation, and any triangulation of a simple polygon with n vertices consists of exactly n-2 triangles.

If $\overleftrightarrow{vw}$ intersects boundary of P: Let $u' \neq u$ be the the vertex furthest to the left of $\overleftrightarrow{vw}$. Take $\overline{uu'}$ as diagonal, which splits P into P_1 and P_2.
#triangles in P = #triangles in P_1 + #triangles in P_2 = $|P_1|-2 + |P_2|-2 = |P_1|+|P_2|-4 = n+2-4 = n-2$



3-Coloring

- A 3-coloring of a graph is an assignment of one out of three colors to each vertex such that adjacent vertices have different colors.

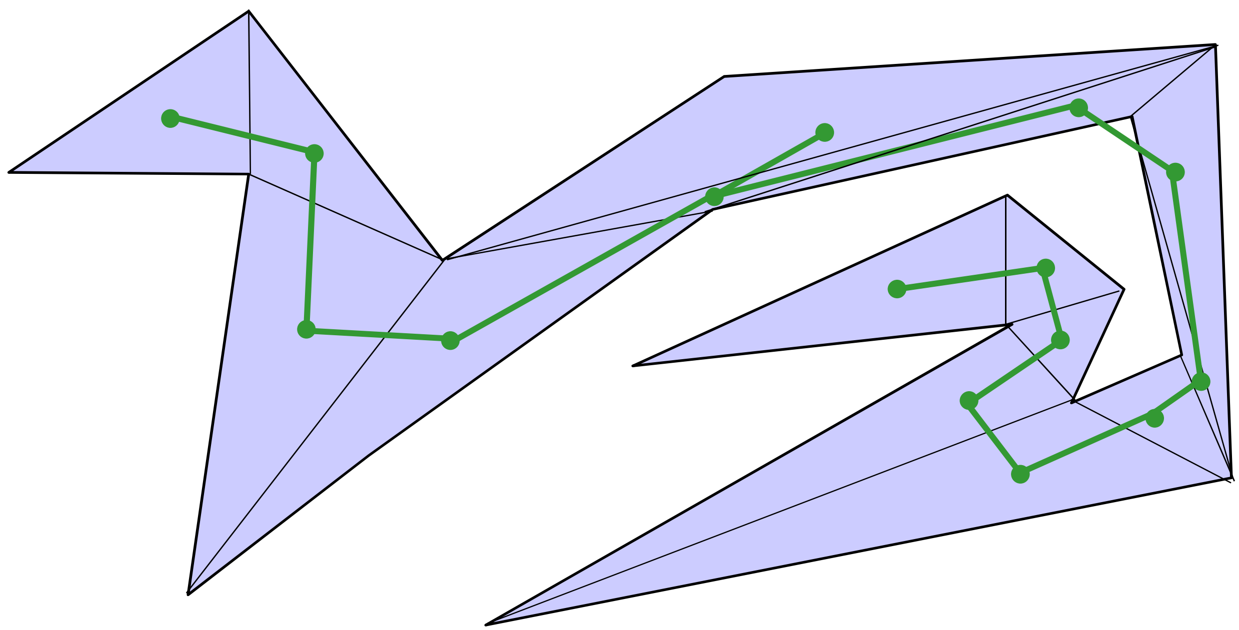


3-Coloring Lemma

Lemma: For every triangulated polgon there is a 3-coloring.

Proof: Consider the **dual graph** of the triangulation:

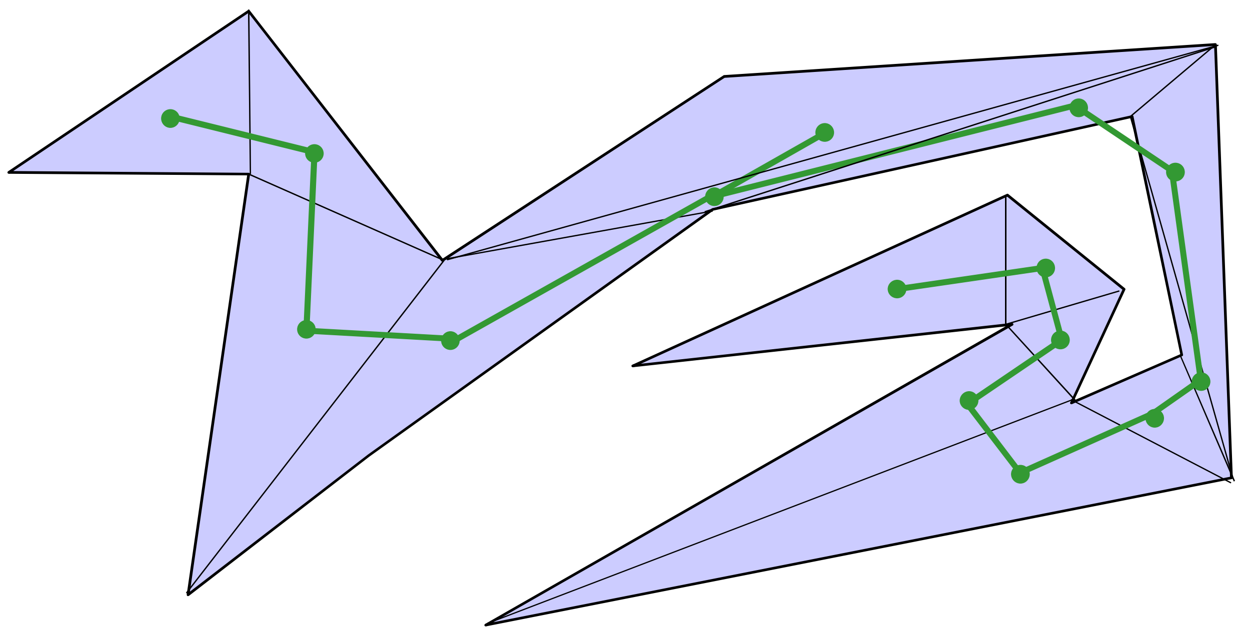
- vertex for each triangle
- edge for each edge between triangles



3-Coloring Lemma

Lemma: For every triangulated polgon there is a 3-coloring.

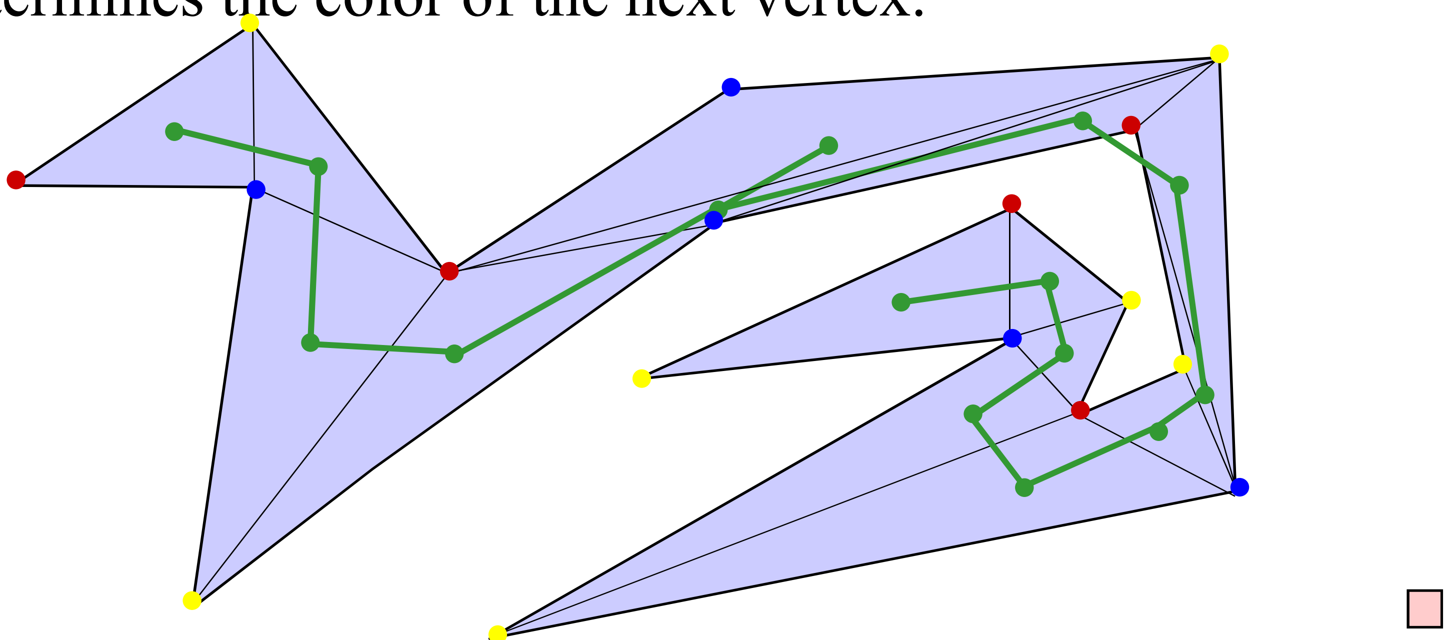
The dual graph is a tree (connected acyclic graph): Removing an edge corresponds to removing a diagonal in the polygon which disconnects the polygon and with that the graph.



3-Coloring Lemma

Lemma: For every triangulated polgon there is a 3-coloring.

Traverse the tree (DFS). Start with a triangle and give different colors to vertices. When proceeding from one triangle to the next, two vertices have known colors, which determines the color of the next vertex.

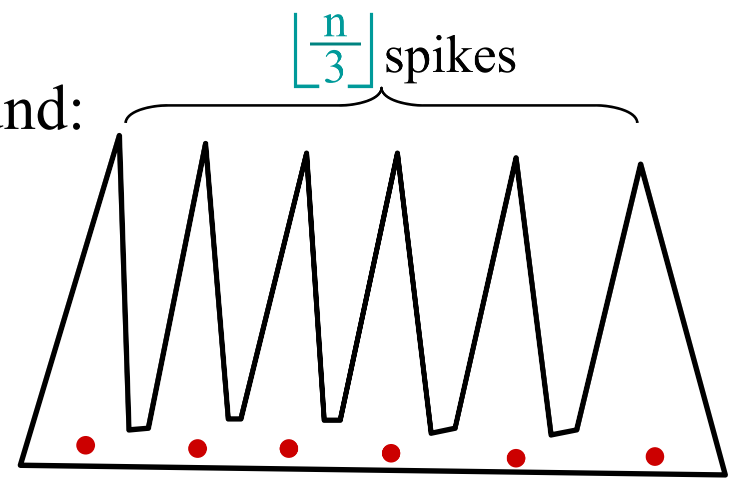


Art Gallery Theorem

Theorem 2: For any simple polygon with n vertices $\left\lfloor \frac{n}{3} \right\rfloor$ guards are sufficient to guard the whole polygon. There are polygons for which $\left\lfloor \frac{n}{3} \right\rfloor$ guards are necessary.

Proof: For the upper bound, 3-color any triangulation of the polygon and take the color with the minimum number of guards.

Lower bound:



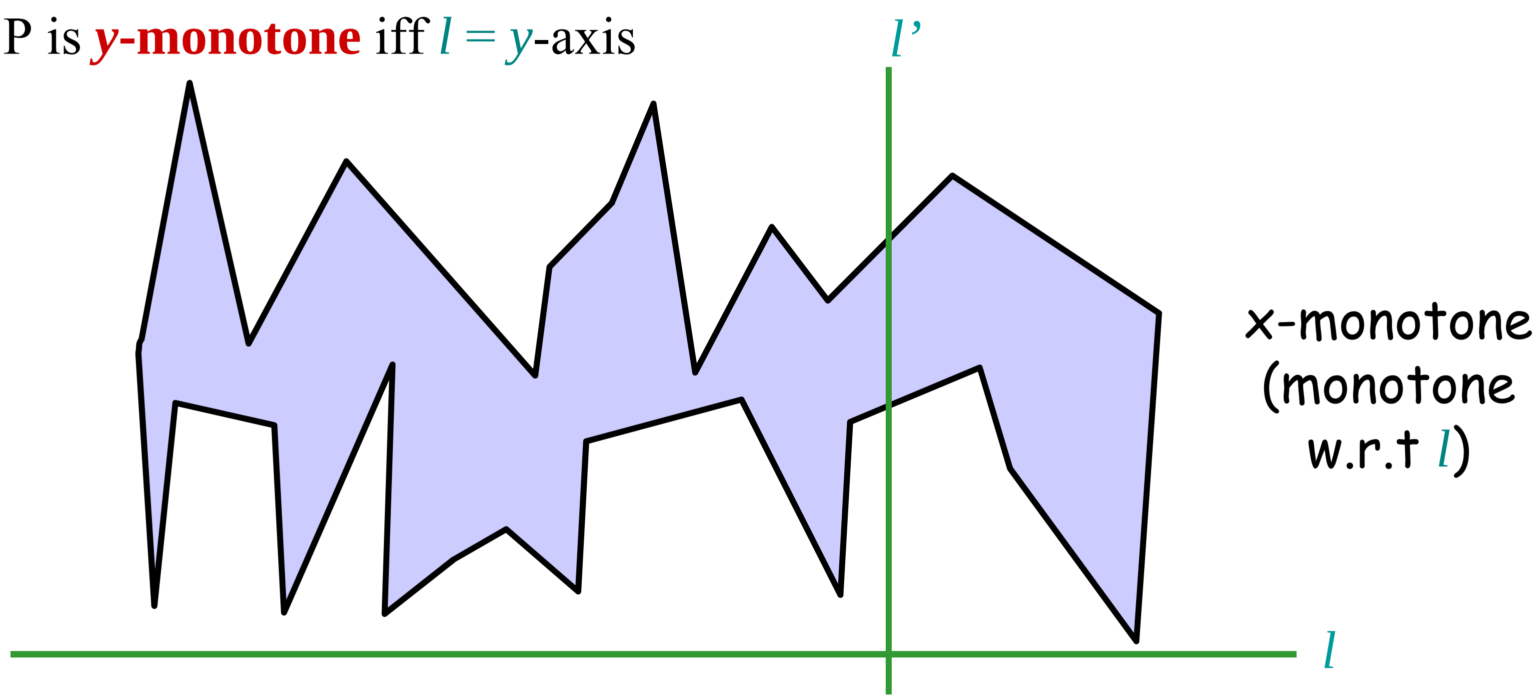
Need one guard per spike.

Triangulating a Polygon

- There is a simple $O(n^2)$ time algorithm based on the proof of Theorem 1.
- There is a very complicated $O(n)$ time algorithm (Chazelle '91) which is impractical to implement.
- We will discuss a practical $O(n \log n)$ time algorithm:
  1. Split polygon into **monotone polygons** ($O(n \log n)$ time)
  2. Triangulate each monotone polygon ($O(n)$ time)

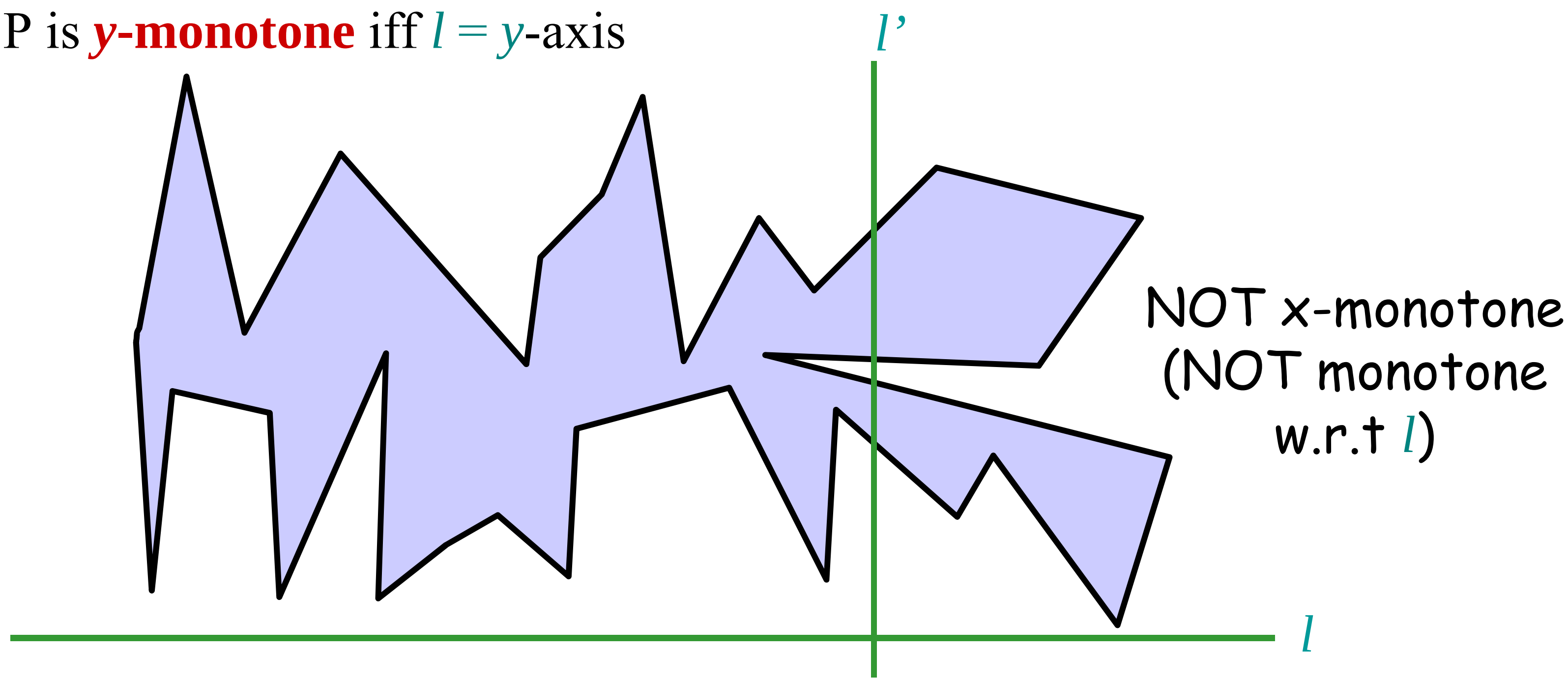
Monotone Polygons

- A simple polygon *P* is called **monotone with respect to a line** *l* iff for every line *l'* perpendicular to *l* the intersection of *P* with *l'* is connected.
  - P is ***x*-monotone** iff *l* = *x*-axis
  - P is ***y*-monotone** iff *l* = *y*-axis



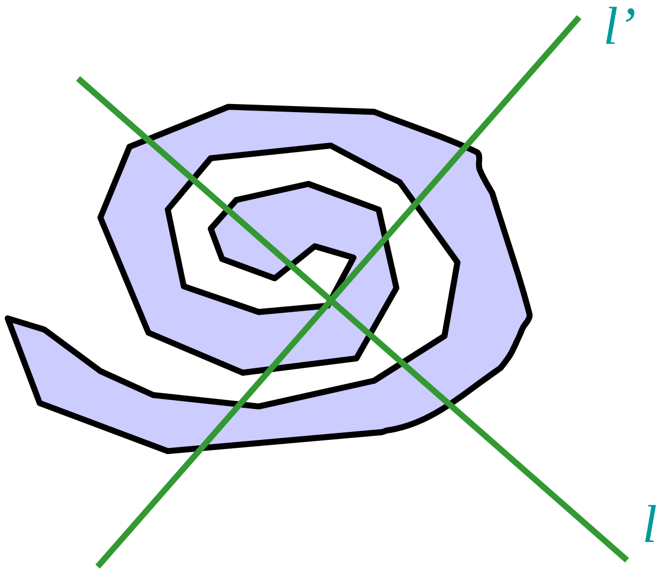
Monotone Polygons

- A simple polygon *P* is called **monotone with respect to a line** *l* iff for every line *l'* perpendicular to *l* the intersection of *P* with *l'* is connected.
  - P is ***x*-monotone** iff *l* = *x*-axis
  - P is ***y*-monotone** iff *l* = *y*-axis



Monotone Polygons

- A simple polygon *P* is called **monotone with respect to a line** *l* iff for every line *l'* perpendicular to *l* the intersection of *P* with *l'* is connected.
  - P is ***x*-monotone** iff *l* = *x*-axis
  - P is ***y*-monotone** iff *l* = *y*-axis

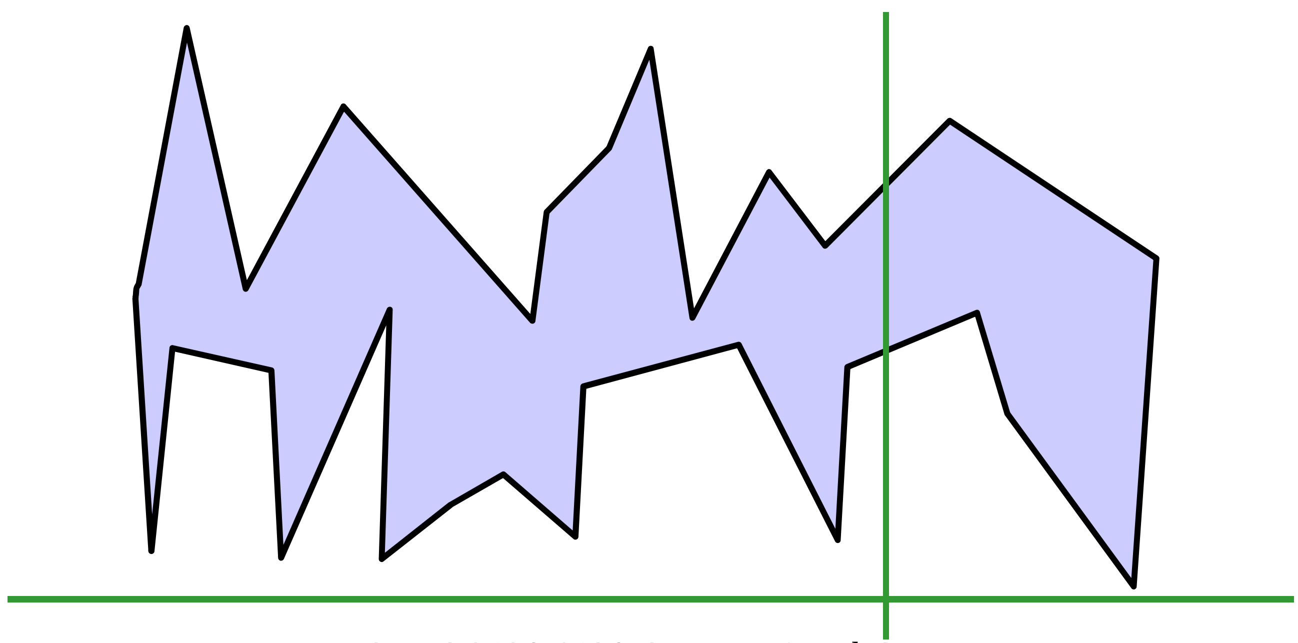


NOT monotone w.r.t any line *l*

Test Monotonicity

How to test if a polygon is *x*-monotone?

- – Find leftmost and rightmost vertices, O(*n*) time
- → Splits polygon boundary in upper chain and lower chain
- – Walk from left to right along each chain, checking that x-coordinates are non-decreasing. O(*n*) time.



Triangulating a Polygon

- There is a simple $O(n^2)$ time algorithm based on the proof of Theorem 1.
- There is a very complicated $O(n)$ time algorithm (Chazelle '91) which is impractical to implement.
- We will discuss a practical $O(n \log n)$ time algorithm:
  1. Split polygon into **monotone polygons** ($O(n \log n)$ time)
  2. Triangulate each monotone polygon ($O(n)$ time)