CSCI 4560/6560 Computational Geometry

https://www.cs.rpi.edu/~cutler/classes/computationalgeometry/F23/

Lecture 4: Triangulation, part 1

Outline for Today

- Homework 2 Questions?
- Last Time: Line Intersection & Map Overlay
- Today's Motivation
  - Art gallery problem
  - Visibility for architectural walkthrough
- Triangulation
  - Proof of Existence & Size
  - Algorithm & Analysis
- Next Time: Improved Algorithm / Analysis
- Other Applications
  - Mesh Simplification
  - Hole filling for 3D Scanning

Homework 2

- Use CGAL's Surface Mesh (Halfedge) data structure

---

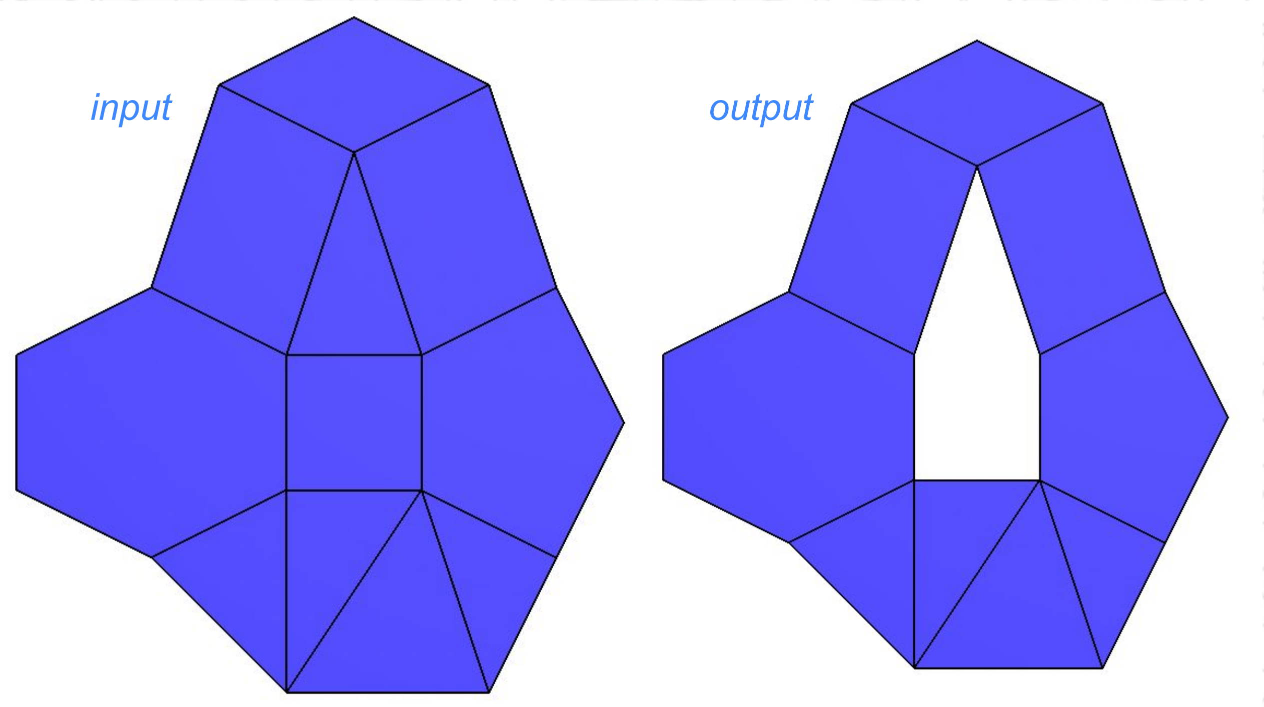
use case 1

- Input: all edges
- Output: all faces on any boundary

---

use case 2

- Input: 1 edge on a boundary
- Output: all faces on that boundary



Homework 2

- Each Halfedge stores:
  - **vertex** at end of directed edge
  - **symmetric** halfedge
  - **face** to left of edge
  - **next** points to the Halfedge counter-clockwise around face on left

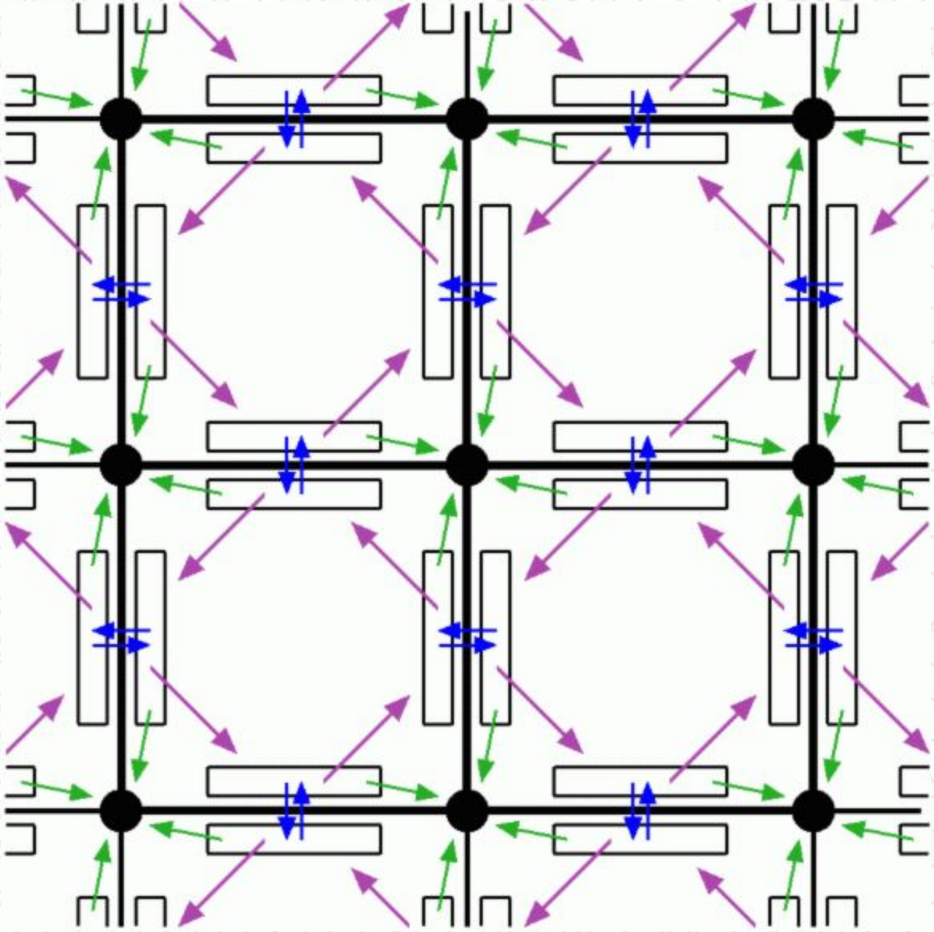


Image from Justin Legakis

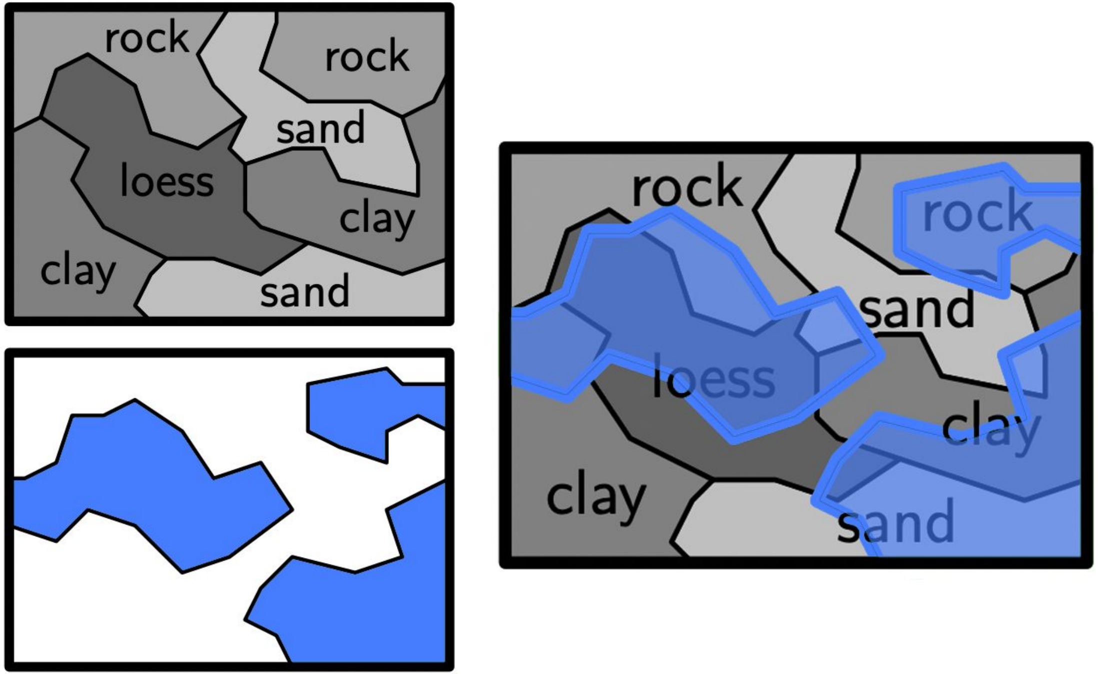
Outline for Today

- Homework 2 Questions?
- Last Time: Line Intersection & Map Overlay
- Today's Motivation
  - Art gallery problem
  - Visibility for architectural walkthrough
- Triangulation
  - Proof of Existence & Size
  - Algorithm & Analysis
- Next Time: Improved Algorithm / Analysis
- Other Applications
  - Mesh Simplification
  - Hole filling for 3D Scanning

Last Time?

- “What is the total area of all lakes that occur over the geological soil type “rock”?

→ *Need to compute intersection* of areas/regions from two or more map layers



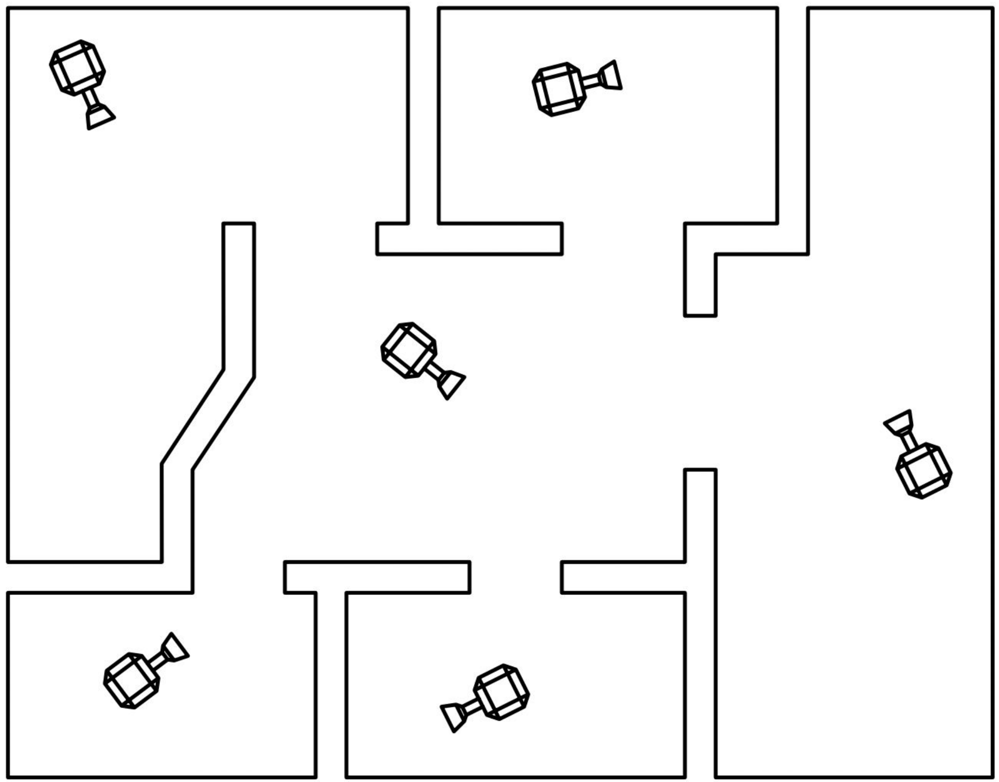
Frank Staals, http://www.cs.uu.nl/docs/vakken/ga/2021/

Outline for Today

- Homework 2 Questions?
- Last Time: Line Intersection & Map Overlay
- Today's Motivation
    - Art gallery problem
    - Visibility for architectural walkthrough
- Triangulation
    - Proof of Existence & Size
    - Algorithm & Analysis
- Next Time: Improved Algorithm / Analysis
- Other Applications
    - Mesh Simplification
    - Hole filling for 3D Scanning

Motivation: Art Gallery Problem

- What is the minimum number of cameras (with 360° rotation) we need to place to get 100% coverage of a 2D floor plan?

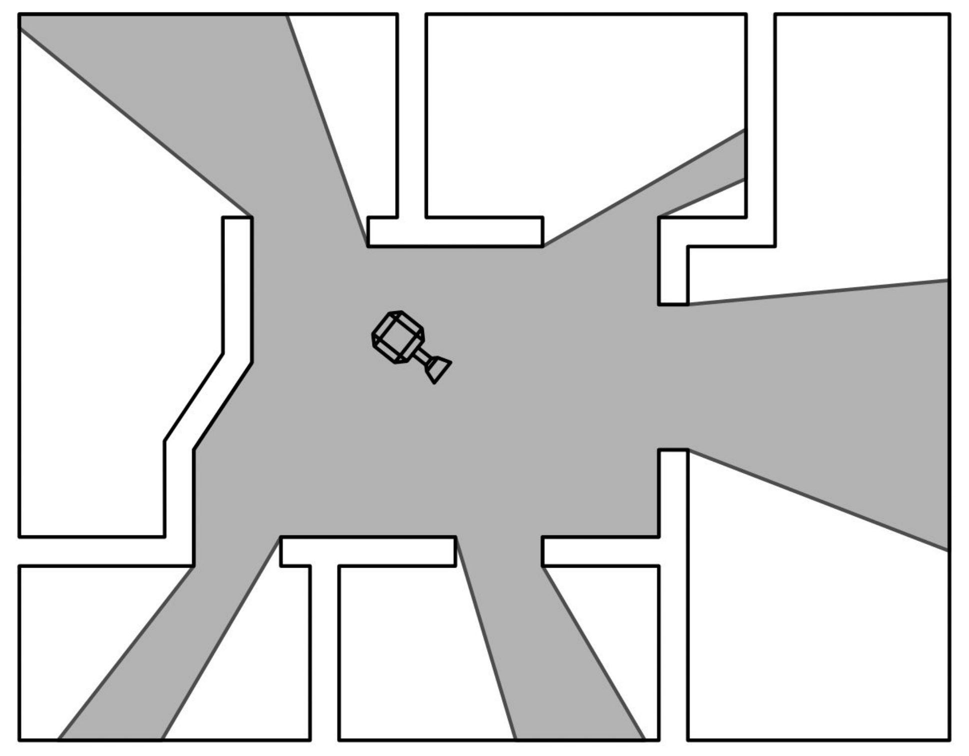


Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 3

Definition: Simple Polygon

- The gallery will be a *simple polygon*.
- What can be viewed from a single camera is also a *simple polygon*.

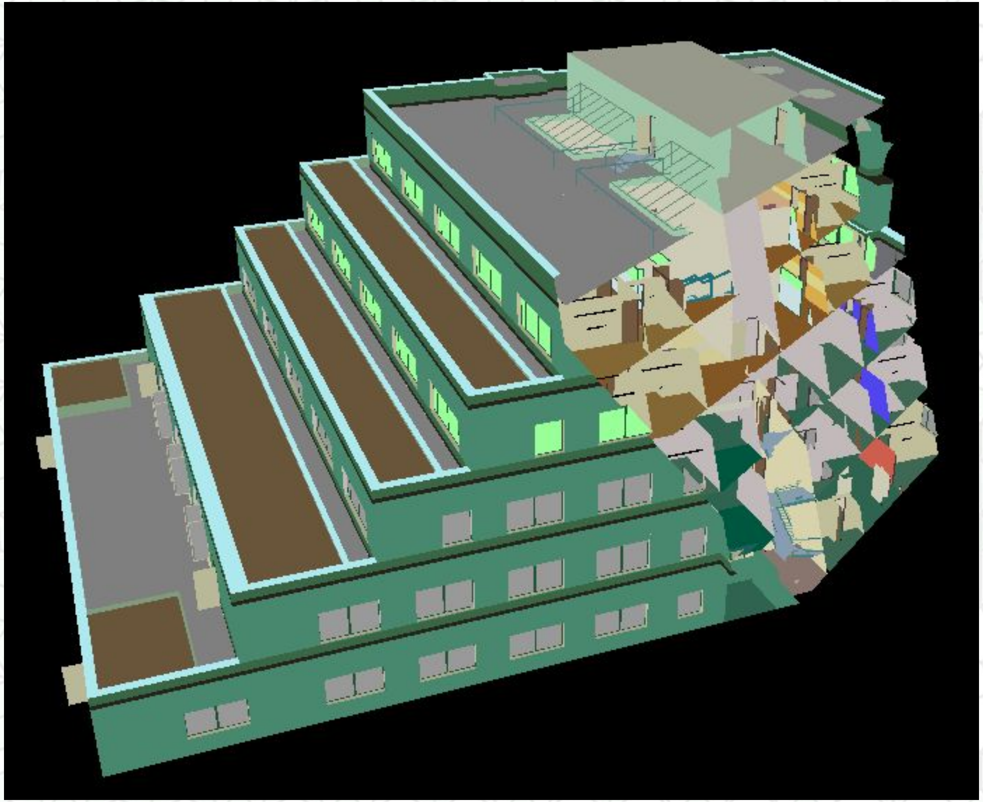
- Single closed polygonal chain boundary
- Connected
- No interior holes
- Does not self intersect



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 3

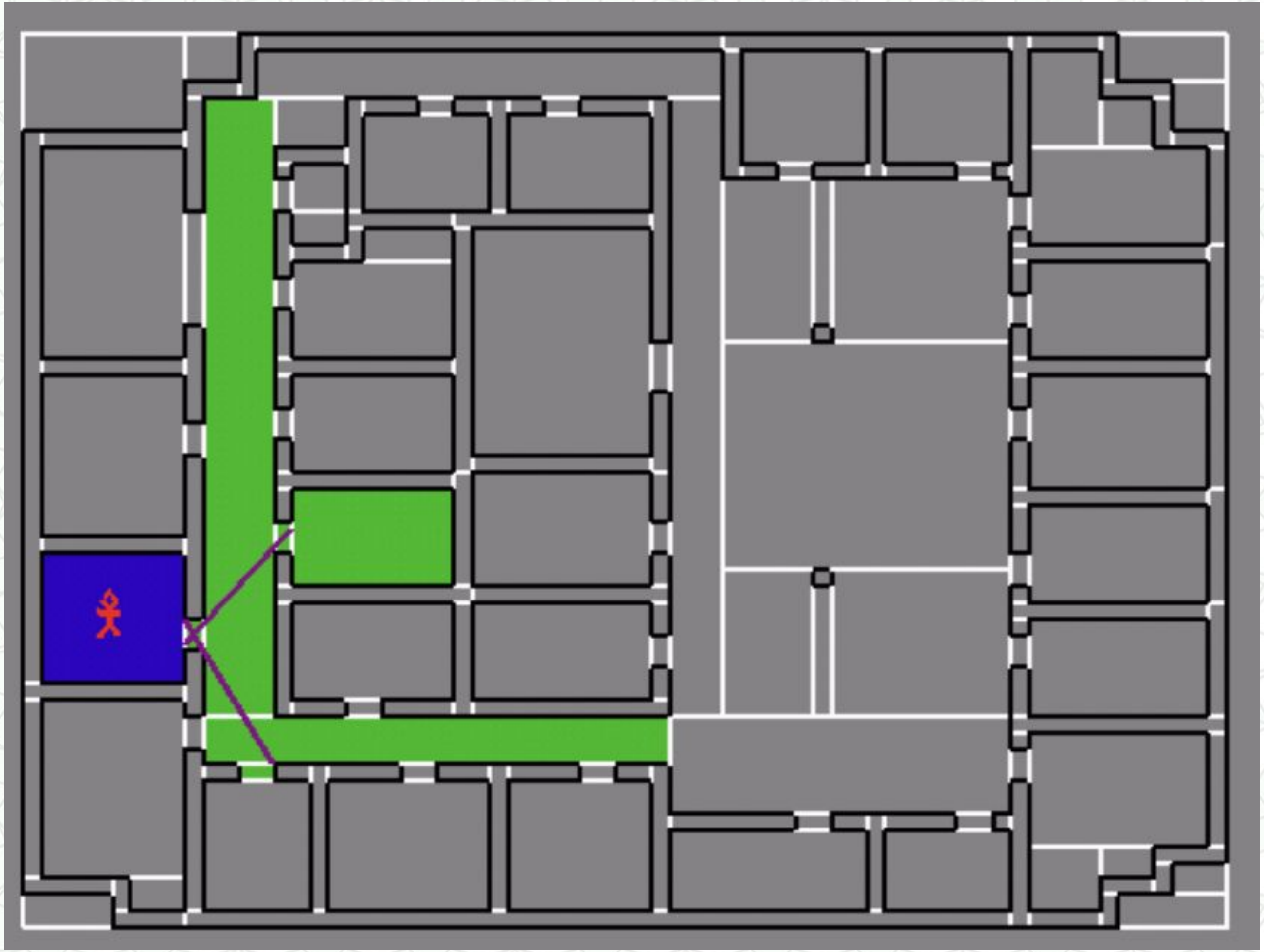
Motivation: Architectural Walkthrough

- UC Berkeley's new Computer Science Building
- Pre-construction visualization
- Very large dataset!
- Interactive/real-time camera motion!

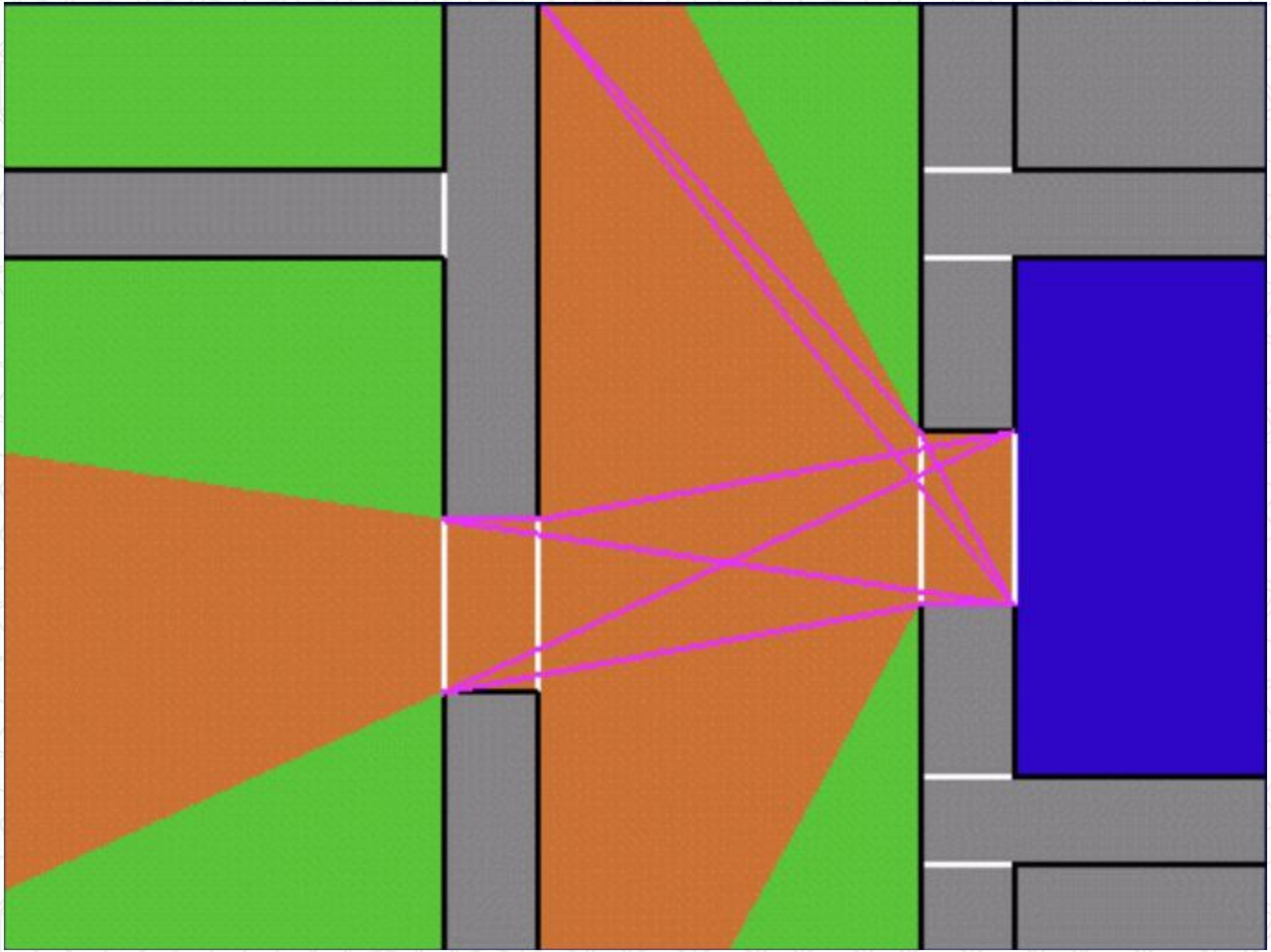


Seth Teller, PhD thesis, 1992, Berkeley Soda Hall walkthrough

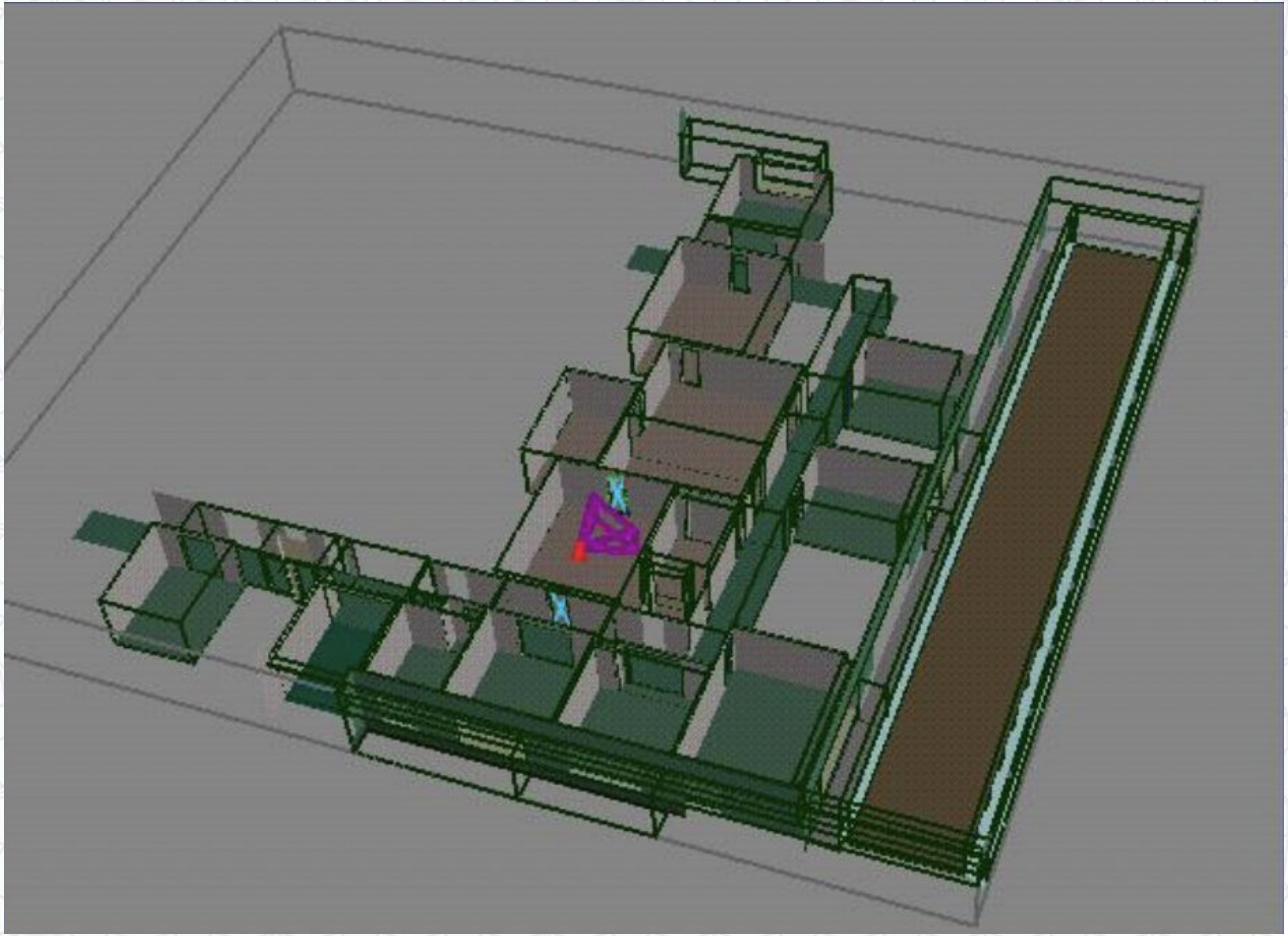
Seth Teller, PhD thesis, 1992, Berkeley Soda Hall walkthrough



Seth Teller, PhD thesis, 1992, Berkeley Soda Hall walkthrough



Seth Teller, PhD thesis, 1992, Berkeley Soda Hall walkthrough

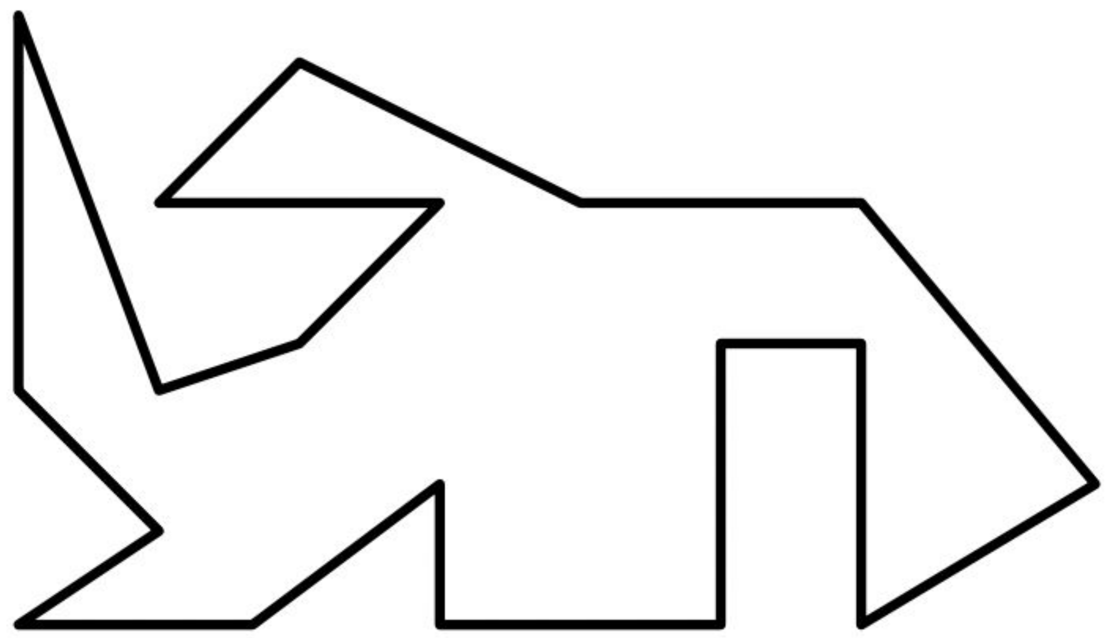


Seth Teller, PhD thesis, 1992, Berkeley Soda Hall walkthrough

Application: Art Gallery Problem

- How many cameras are necessary for 100% coverage?
- Where should we place these cameras?

- *Note: The optimal solution is NP hard!*



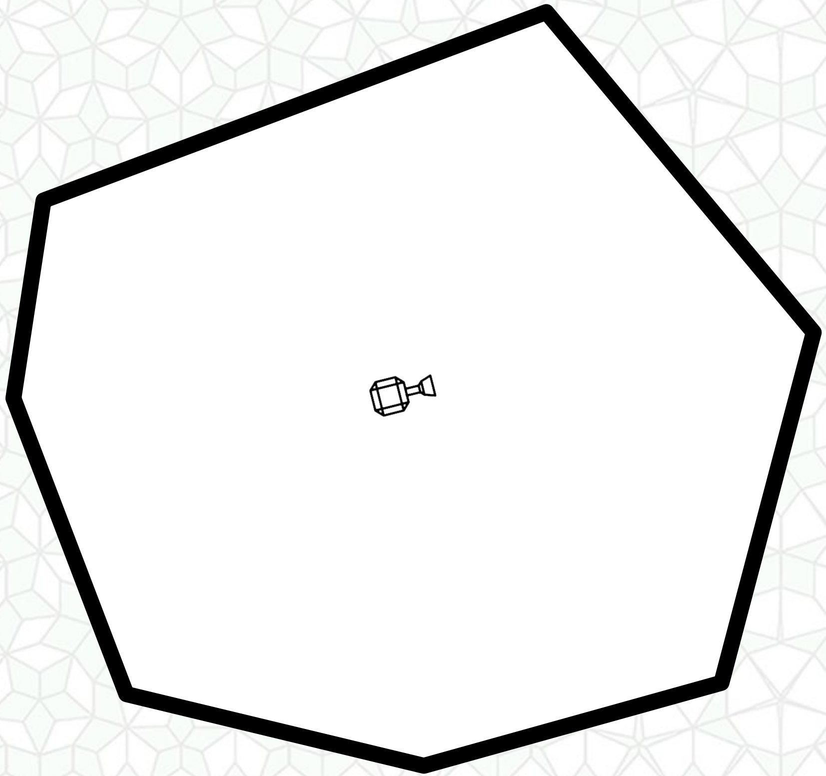
Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 3

Application: Art Gallery Problem

- If the gallery is convex, we can just use a single camera placed anywhere inside that polygon.

- If we chop up a non-convex polygon into convex polygons, we can place 1 camera per polygon and get 100% coverage.

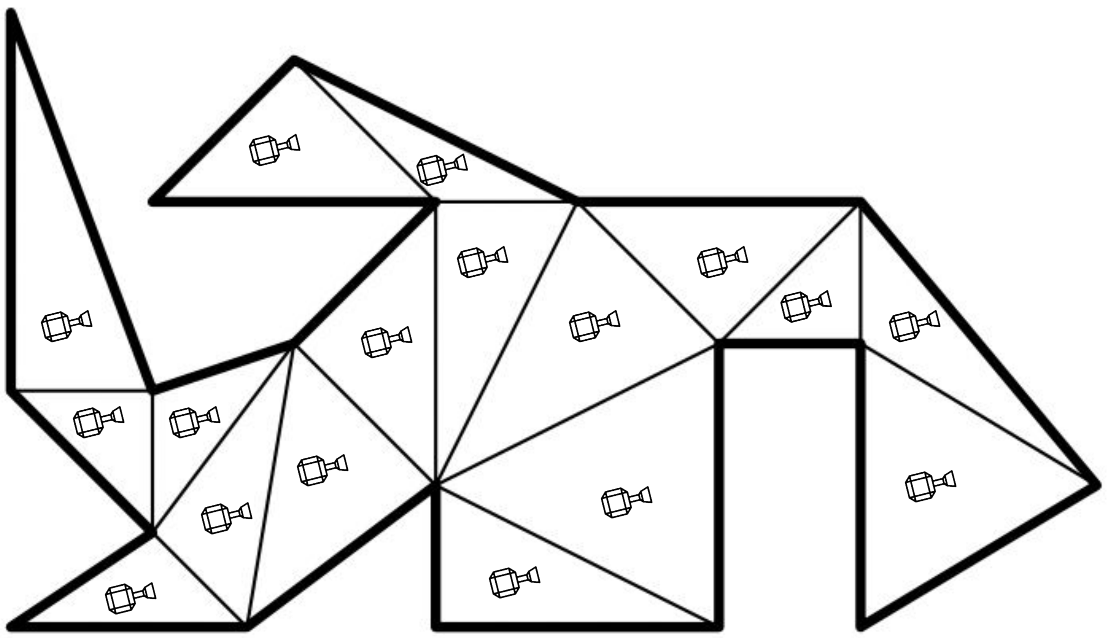
  This isn't easy to do optimally…



Application: Art Gallery Problem

- Let’s chop up a non-convex polygon into triangles (which are convex).

- Place 1 camera per triangle and get 100% coverage.

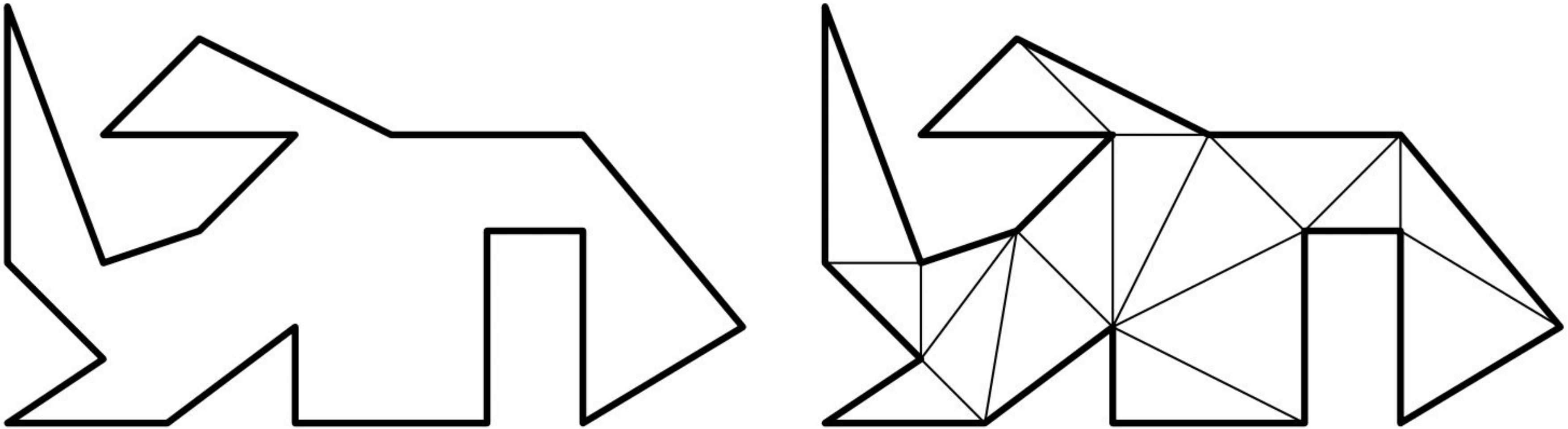


Outline for Today

- Homework 2 Questions?
- Last Time: Line Intersection & Map Overlay
- Today's Motivation
    - Art gallery problem
    - Visibility for architectural walkthrough
- Triangulation
    - Proof of Existence & Size
    - Algorithm & Analysis
- Next Time: Improved Algorithm / Analysis
- Other Applications
    - Mesh Simplification
    - Hole filling for 3D Scanning

Can every shape be Triangulated?

- Theorem 3.1 (CCAA book): Every simple polygon admits a triangulation, and any triangulation of a simple polygon with n vertices consists of exactly n−2 triangles.



Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 3

Can every shape be Triangulated?

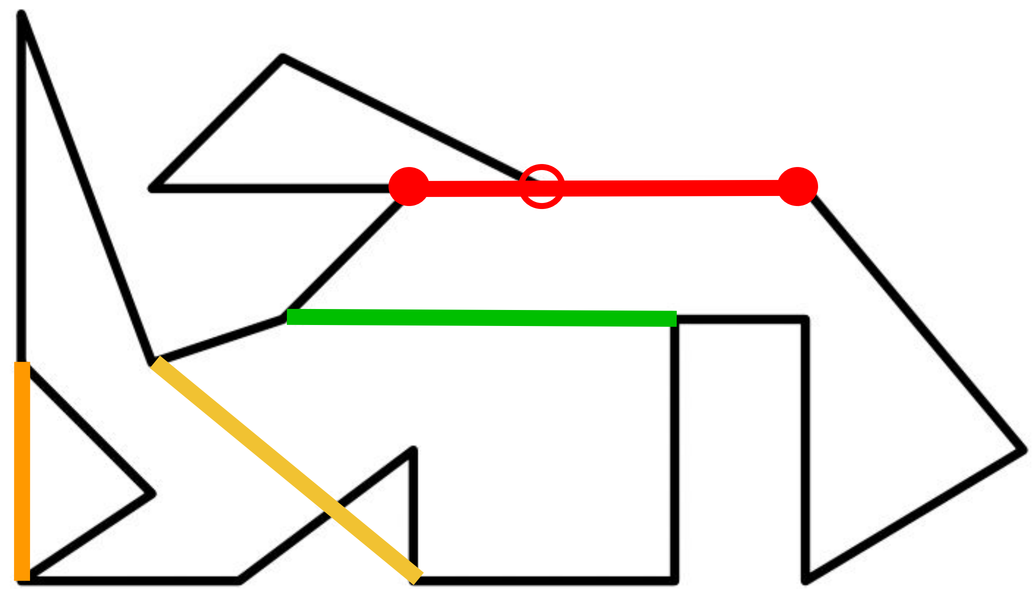
- Theorem 3.1 (CCAA book): Every simple polygon admits a triangulation, and any triangulation of a simple polygon with n vertices consists of exactly $n-2$ triangles.

- ***Proof by Induction***
- A polygon with 3 vertices is a triangle.
- Assume every polygon with $n-1$ or fewer vertices can be triangulated.
- Given a polygon with n vertices, we will draw a *diagonal line* between two vertices that cuts this shape into two smaller polygons which can be triangulated.

Which Diagonals are Allowed?

- Diagonal should connect two non-adjacent vertices on the polygonal boundary.

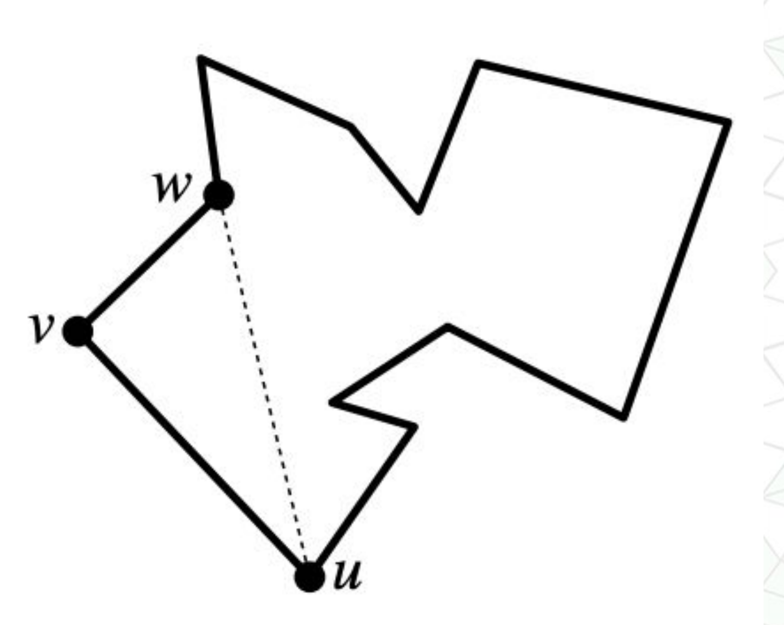
- Diagonal must not be outside the polygon.
- Diagonal may not cross any edge.
- Diagonal should not pass through any other vertex.



How do we find a Valid Diagonal?

- Start at the leftmost vertex, *v*
  - *NOTE: If two or more vertices have the same x, chose the one with smaller y.*
- Find vertices *u* and *w,* adjacent to *v*

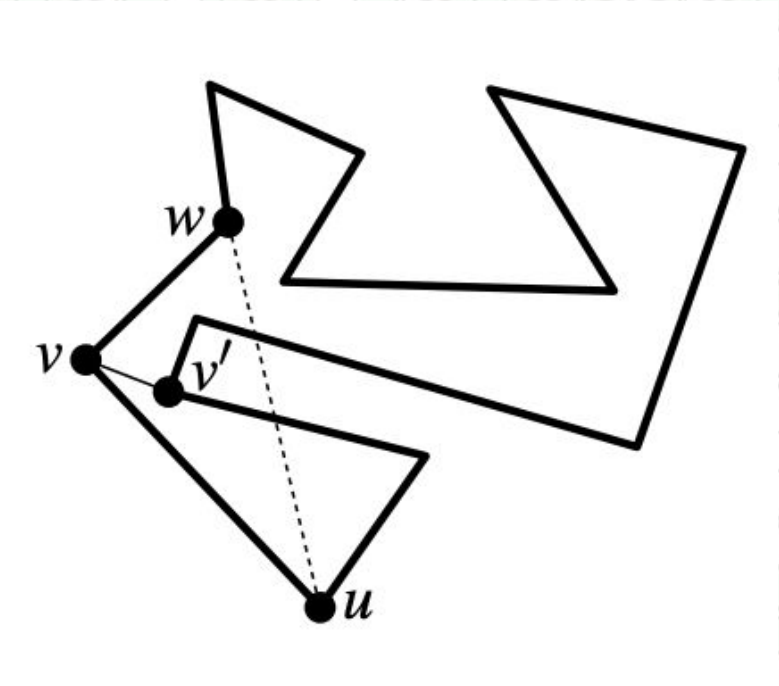
- Check if the line *uw* is a valid diagonal.
  - This line does not pass through *v.*
  - Does it intersect other line segments?
  - Does it pass through any other vertices?
  - Does it lie completely outside of the polygon? (possible if one of the vertices is the rightmost vertex)



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 3

How do we find a Valid Diagonal?

- If it does cross another line segment, there must one or more vertices inside the triangle *uvw.*
- Starting at the intersection, walk along the boundary to find those vertices.
- Choose the vertex *v'*,
  furthest from the line segment *uw*
- Draw the diagonal from *v* to *v'*



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 3

How many Triangles are Necessary?

- Theorem 3.1 (CCAA book): Every simple polygon admits a triangulation, and any triangulation of a simple polygon with n vertices consists of exactly $n-2$ triangles.

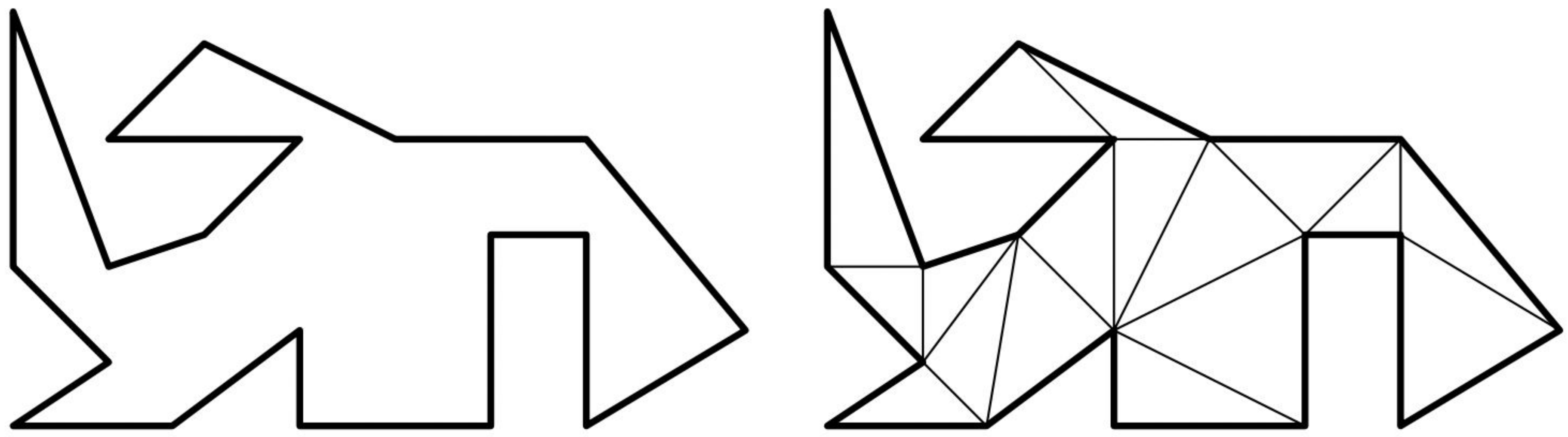
How many Triangles are Necessary?

- Theorem 3.1 (CCAA book): Every simple polygon admits a triangulation, and any triangulation of a simple polygon with n vertices consists of exactly $n-2$ triangles.

- When we draw a diagonal and split the polygon with n vertices into two smaller polygons with m_1 and m_2 vertices.
- Every vertex will be used in exactly one of the two smaller polygons, except two vertices will appear in both polygons.
- $m_1 + m_2 = n + 2$
- By induction, triangulations of these smaller polygons will have $m_1 - 2$ and $m_2 - 2$ triangles.
- Overall: $m_1 - 2 + m_2 - 2 = (m_1+m_2) - 4 = n+2-4 = n-2$ triangles

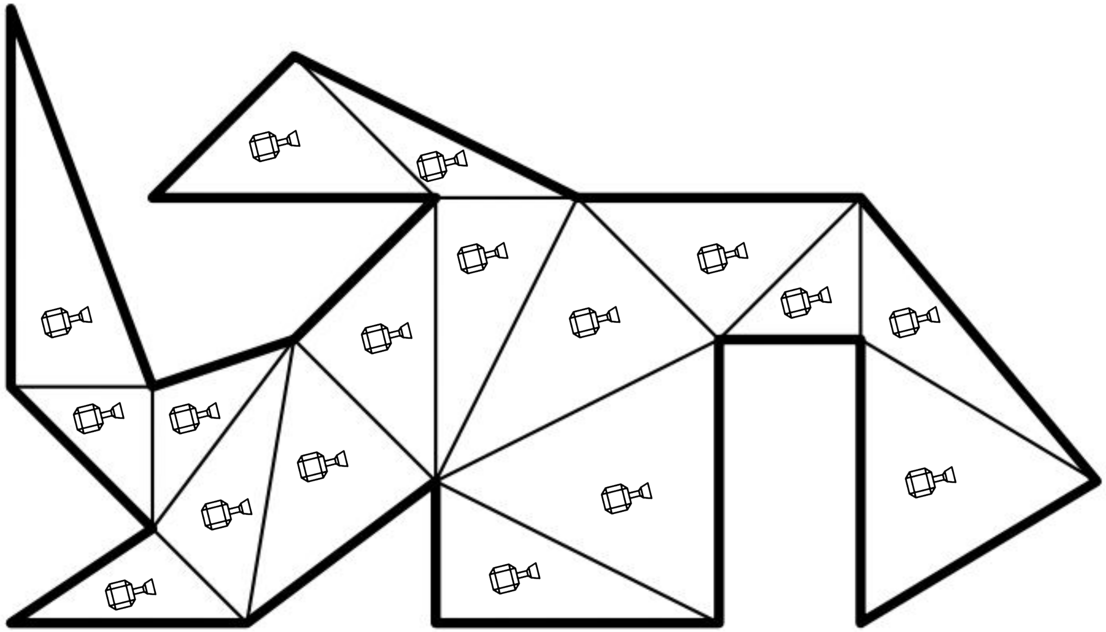
Non-Uniqueness of Triangulation

- Observation: There's more than one way to chop up this non-convex polygon into triangles *… more on this later in the term*



Application: Art Gallery Problem

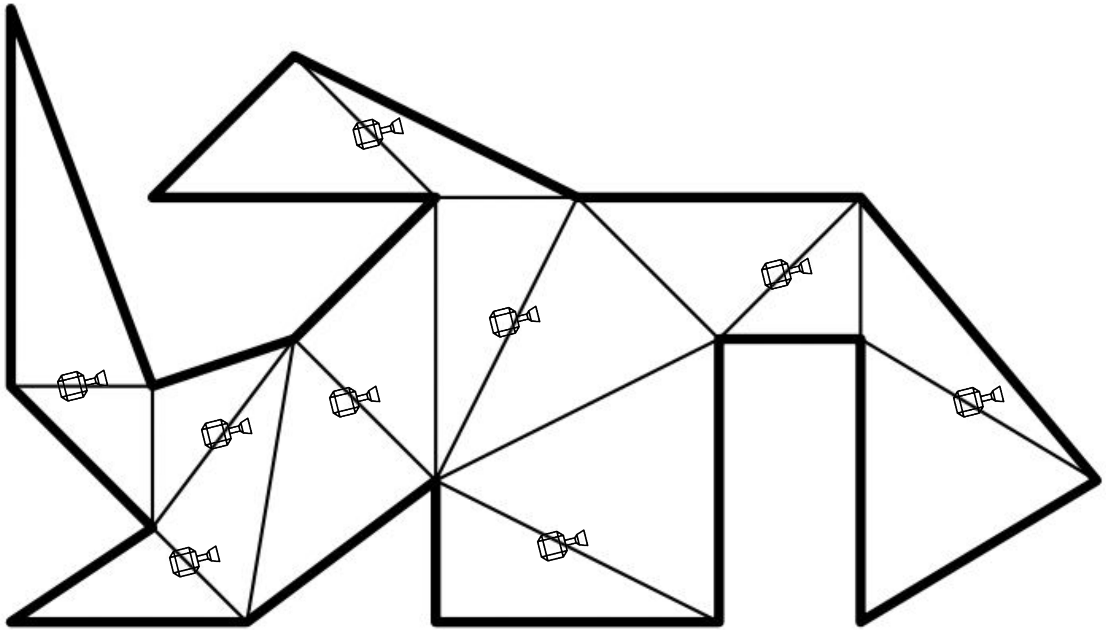
- Non convex gallery with *n* edges, *n*-2 triangles
- Place 1 camera per triangle
- Requires *n*-2 cameras
- Can we do better?



Application: Art Gallery Problem

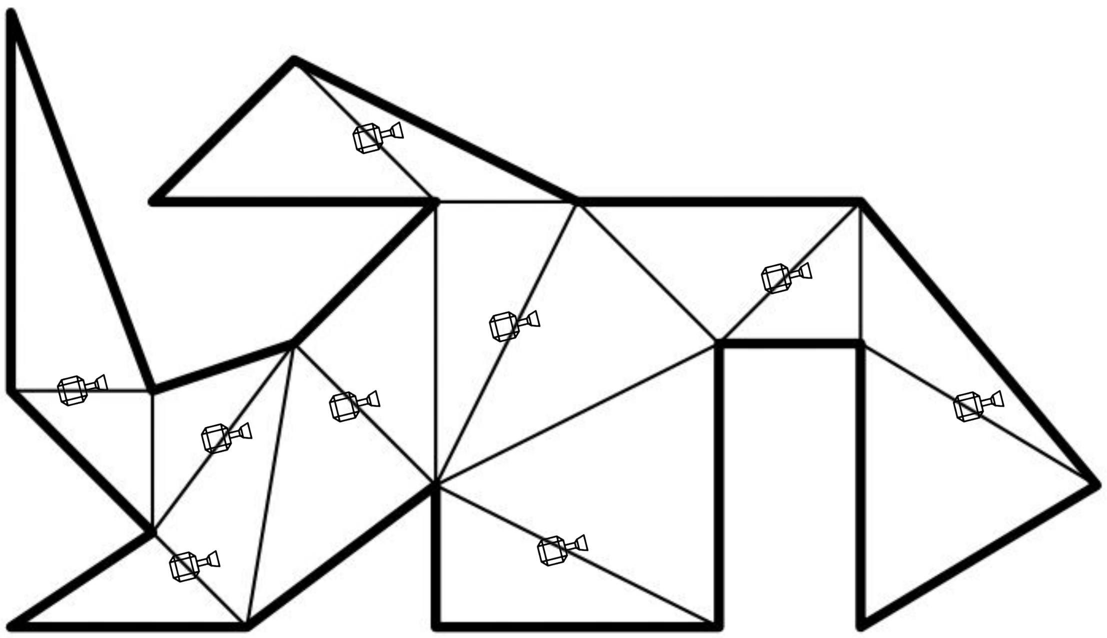
- Place cameras on edge between 2 triangles

- Covers both triangles



Application: Art Gallery Problem

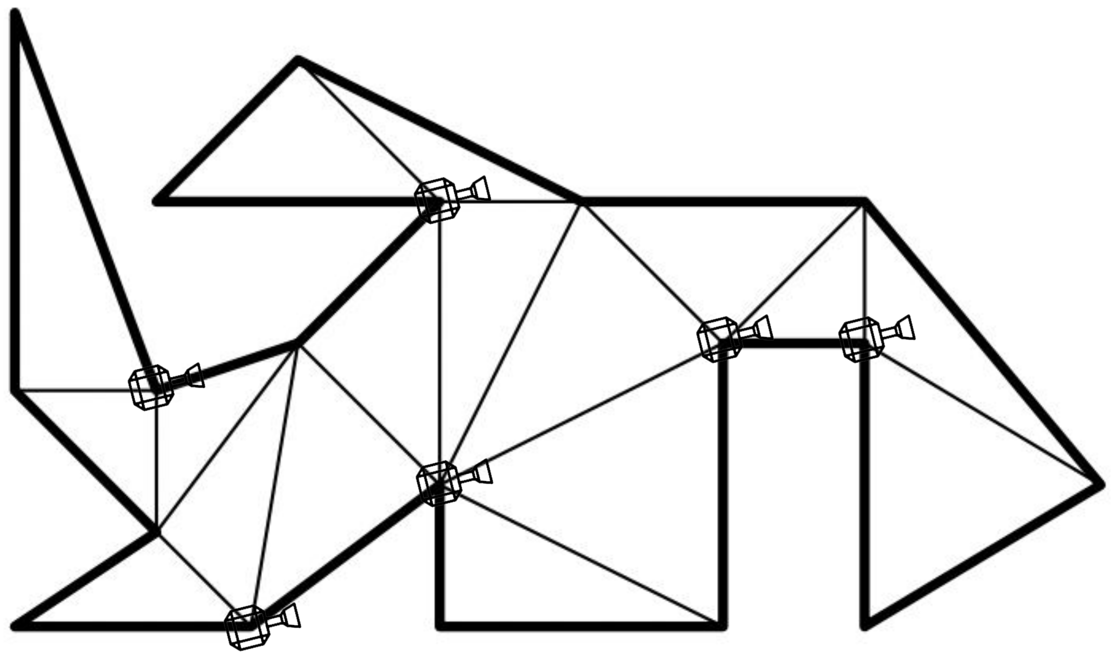
- Place cameras on edge between 2 triangles
- Covers both triangles
- Requires $\approx n/2$ cameras
- Can we do better?



Application: Art Gallery Problem

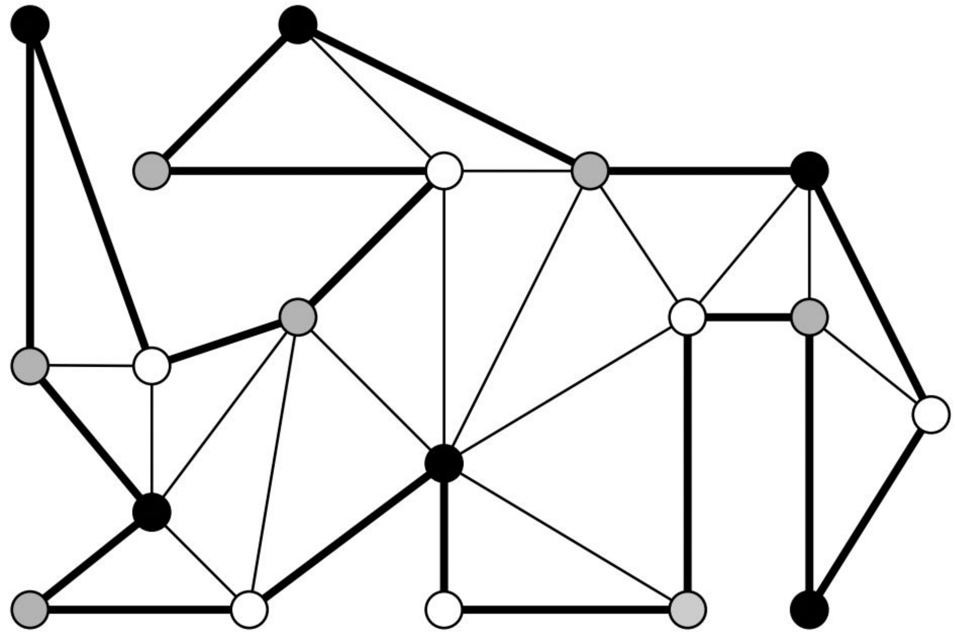
- Place cameras on vertices
- Can view all triangles that touch that vertex

- On which vertices should we place the cameras?



3 Coloring of a Triangulated Simple Polygon

- The vertices of a triangulated simple polygon can be colored with 3 colors (white, grey, black) such that each triangle has one vertex of each color (no duplicates).



- Place cameras on color is used the least
- ≤ *n/3* cameras

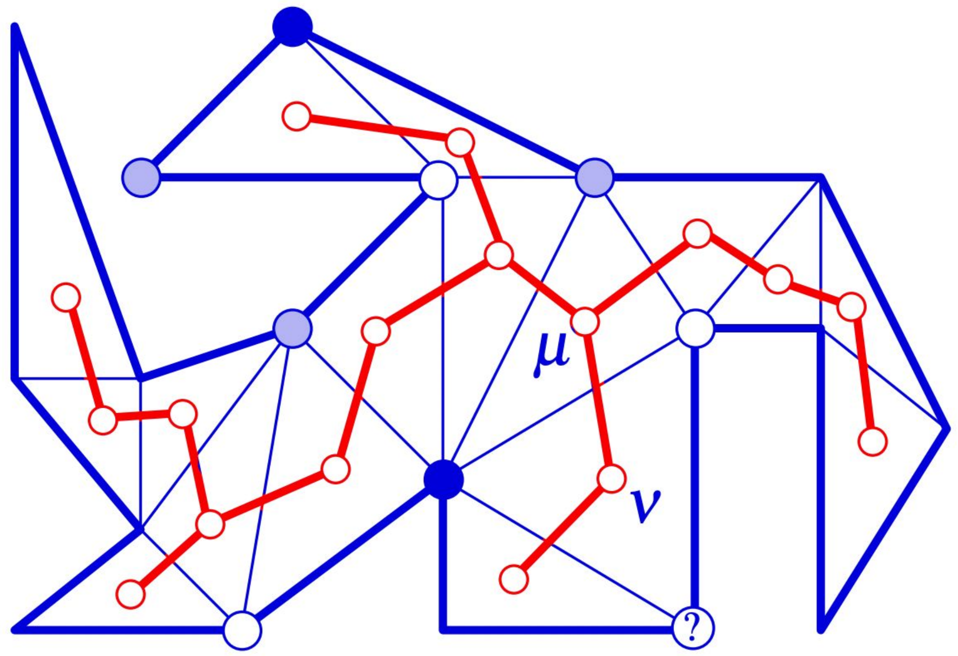
Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 3

Definition: Dual Graph

A common and very important tool in our Computational Geometry toolbox!

- We place a **vertex** in the dual graph at the center of every triangle in the primary graph

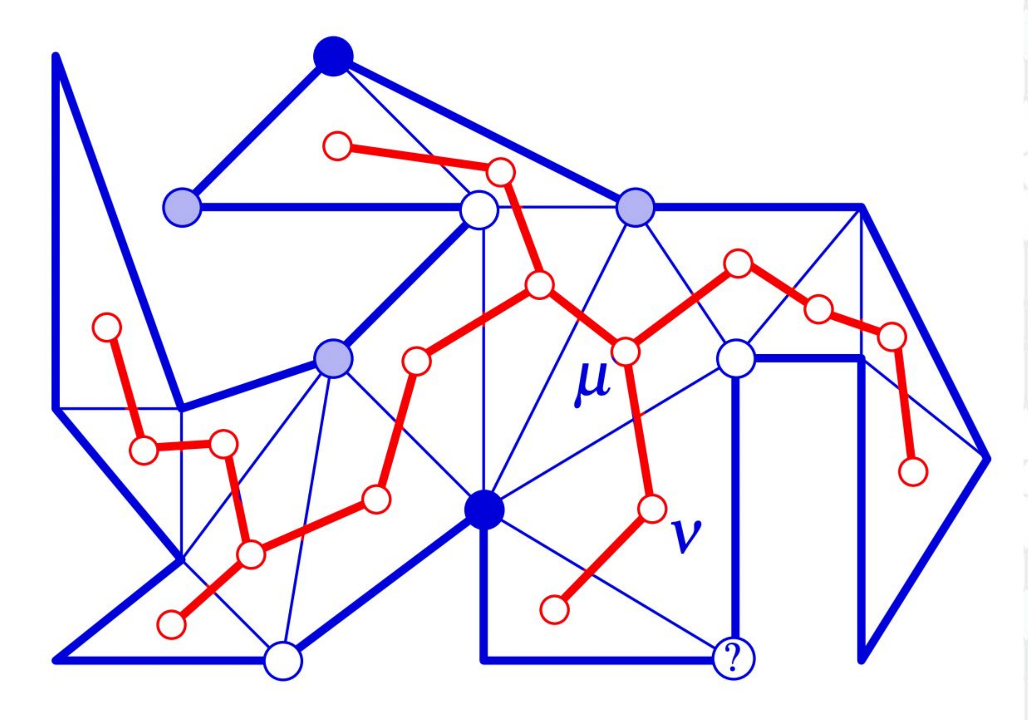
- We draw an **edge** in the dual graph connecting two vertices if the corresponding triangles in the primary graph share an edge.



Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 3

Dual Graph - Is a 3 Coloring Always Possible?

- The dual graph for our triangulated simple polygon is a tree (no cycles!)
  - Connected
  - No interior holes in the polygon
  - No interior vertices in the triangulation
- We can perform a depth-first tree walk and color the vertices without duplicates



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 3

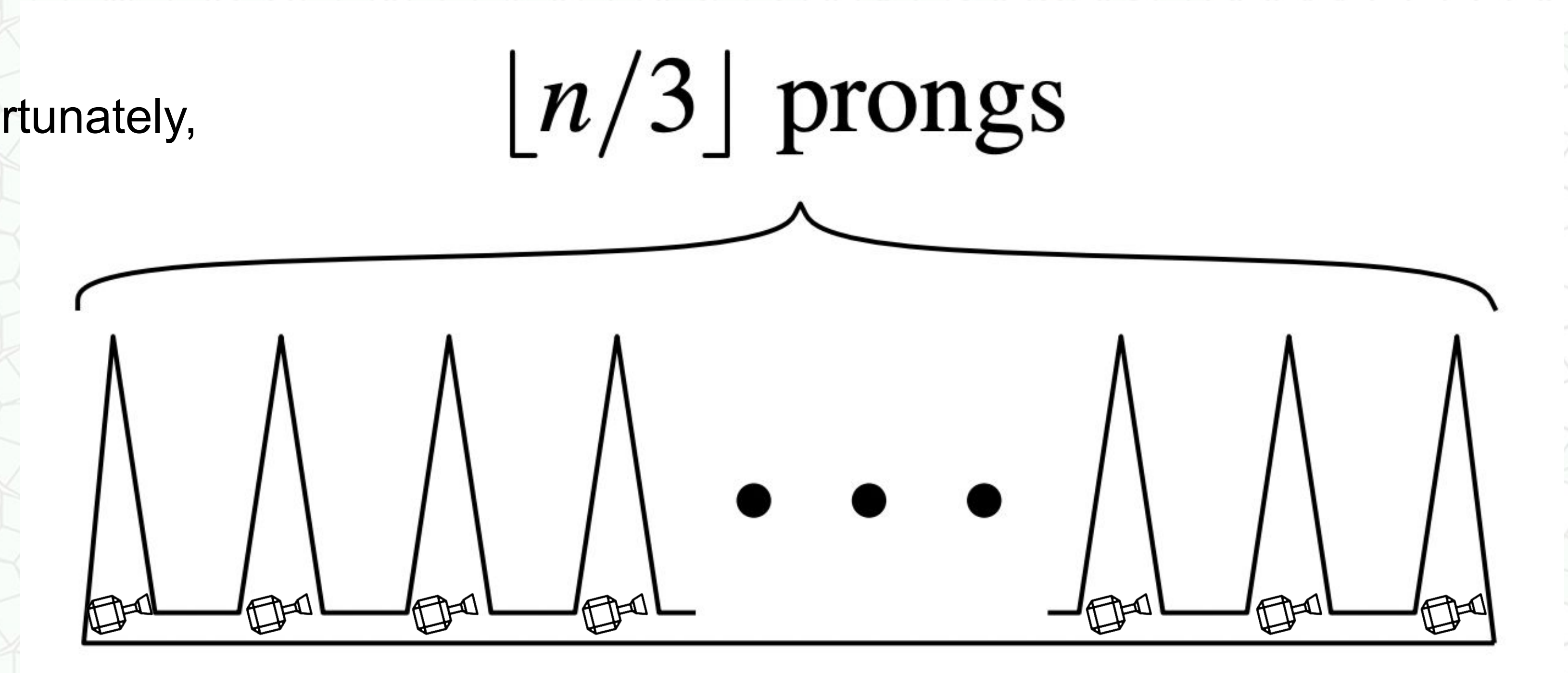
Application: Art Gallery Problem

- Can we do better than *n/3* cameras?

Application: Art Gallery Problem

- Can we do better than *n/3* cameras?

- Unfortunately, no…



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 3

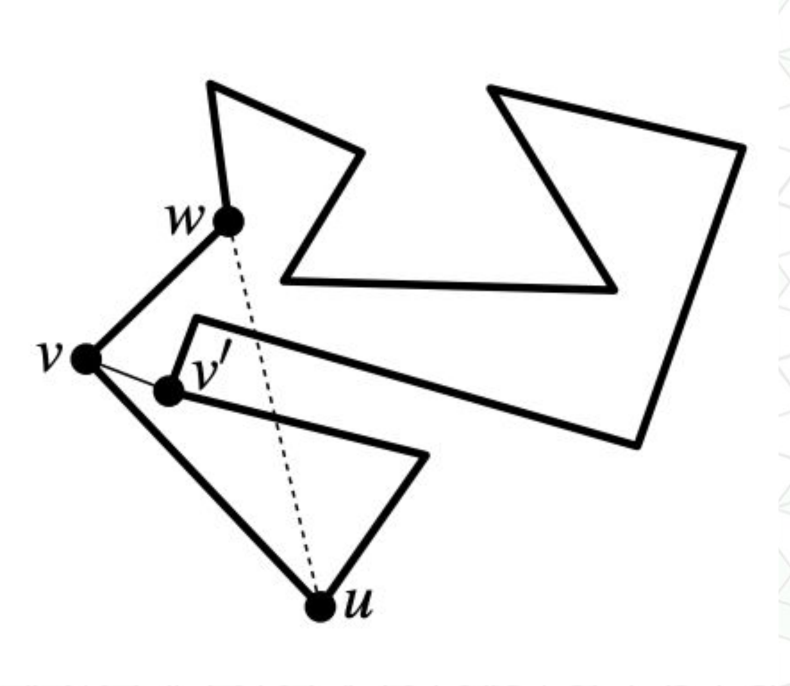
Preliminary Analysis?

- What is the worst case running time to triangulate a non-convex, simple polygon with *n* vertices?

- Identify a legal diagonal

- Split into two smaller polygons

- Overall:



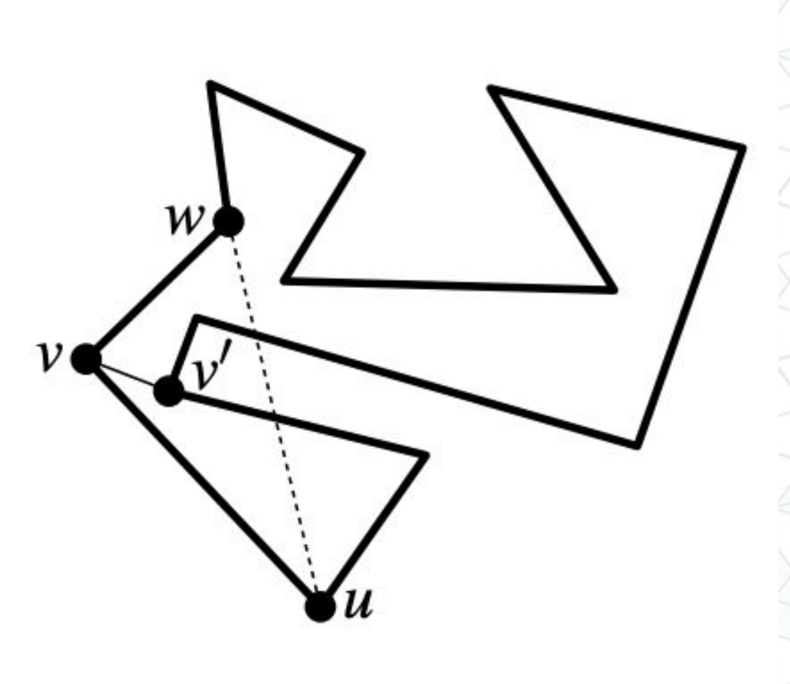
Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 3

Preliminary Analysis?

- What is the worst case running time to triangulate a non-convex, simple polygon with *n* vertices?

- Identify a legal diagonal
  - O(n) in worst case
- Split into two smaller polygons
  - Worst case:
    $m_1 = 3$ vertices and
    $m_2 = n-1$ vertices

- Overall: $O(n^2)$ running time



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 3

Outline for Today

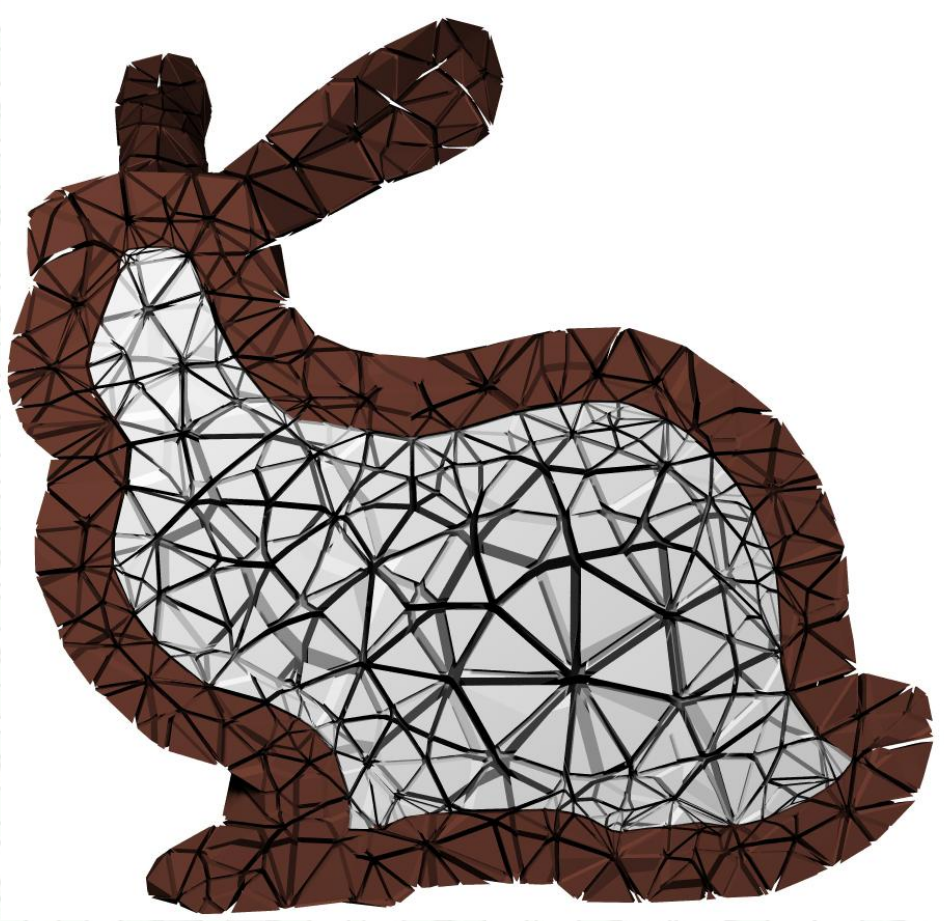
- Homework 2 Questions?
- Last Time: Line Intersection & Map Overlay
- Today's Motivation
  - Art gallery problem
  - Visibility for architectural walkthrough
- Triangulation
  - Proof of Existence & Size
  - Algorithm & Analysis
- Next Time: Improved Algorithm / Analysis
- Other Applications
  - Mesh Simplification
  - Hole filling for 3D Scanning

Next Time…

- Analysis: Can we do better than $O(n^2)$?

  YES!

- Does this work in 3D too?
- Can we triangulate or tetrahedralize the interior of a polytope?

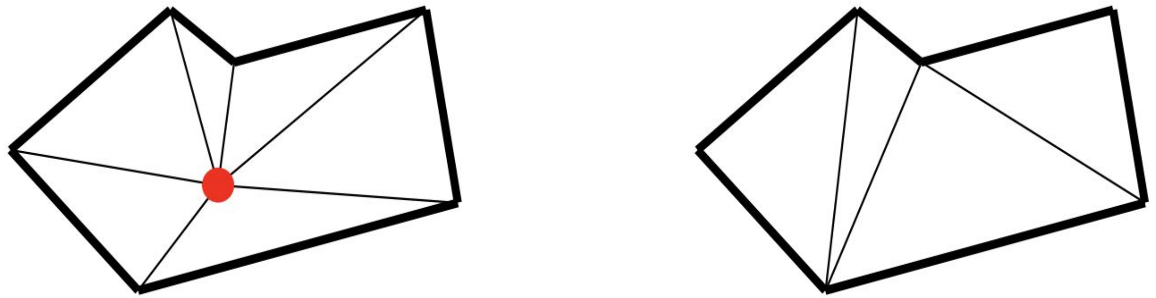


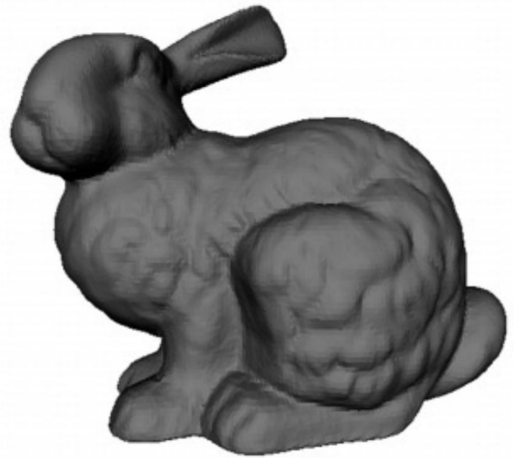
Outline for Today

- Homework 2 Questions?
- Last Time: Line Intersection & Map Overlay
- Today's Motivation
  - Art gallery problem
  - Visibility for architectural walkthrough
- Triangulation
  - Proof of Existence & Size
  - Algorithm & Analysis
- Next Time: Improved Algorithm / Analysis
- Other Applications
  - Mesh Simplification
  - Hole filling for 3D Scanning

Application: Mesh Simplification

- Identify a *relatively unimportant* vertex to remove
- Remove the connected triangles
- Re-triangulate the hole



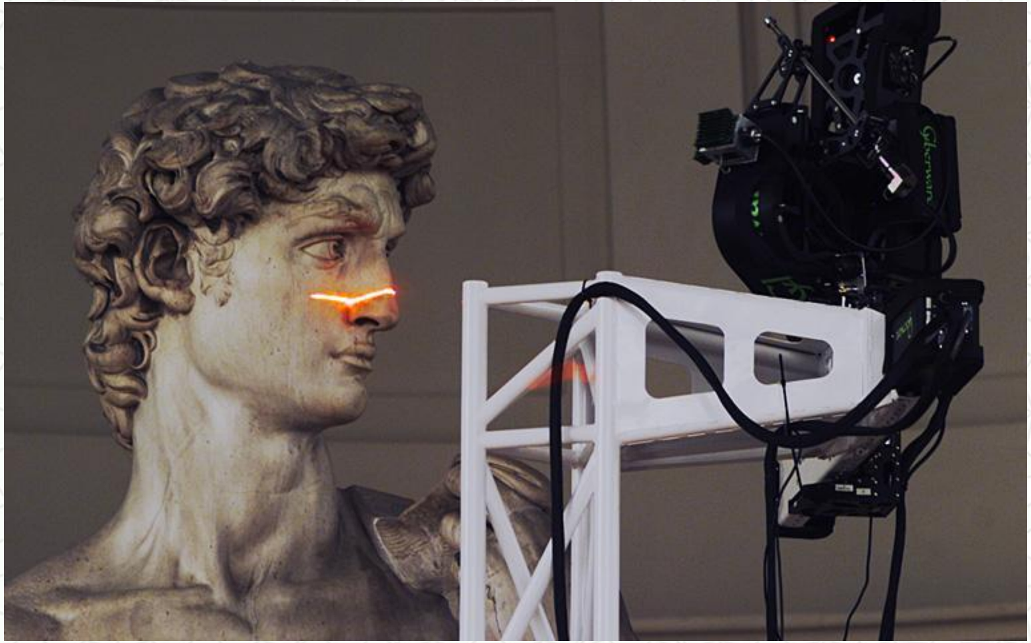


Original: 70,000 triangles

Simplified: 1,000 triangles

"Surface Simplification Using Quadric Error Metrics"
Garland & Heckbert, SIGGRAPH 1997

Application: 3D Digitizing

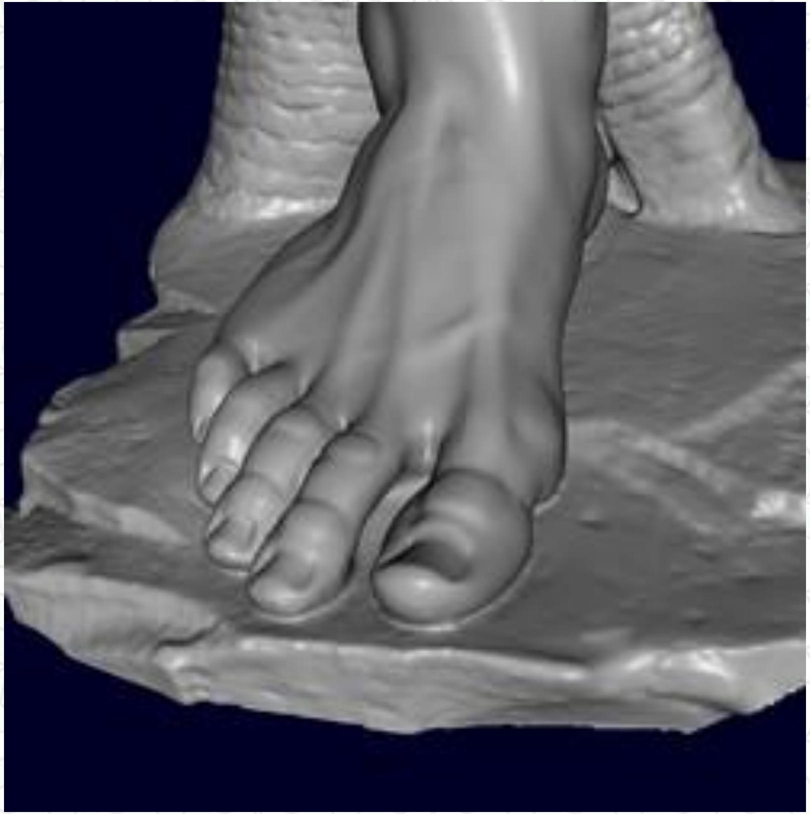


The Digital Michelangelo Project: 3D Scanning of Large Statues, Levoy et al., *SIGGRAPH 2000*

Cyberware

Application: Hole Filling

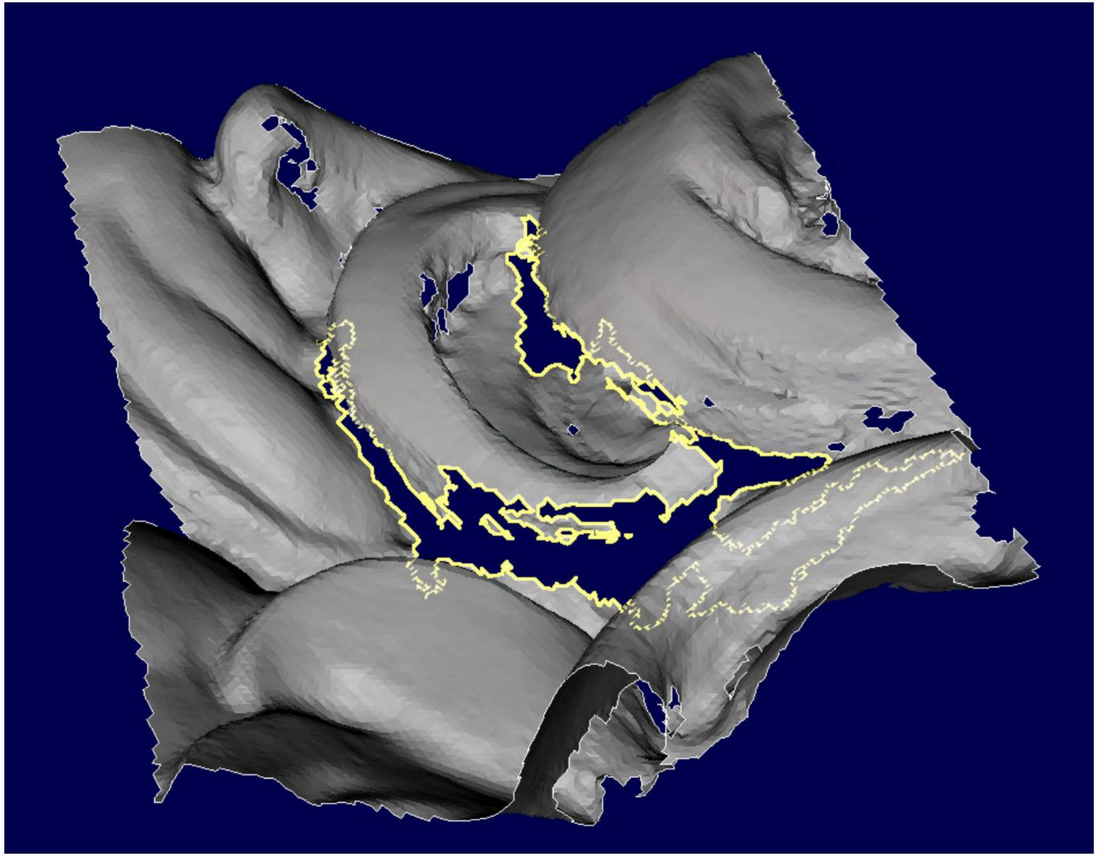
"Filling holes in complex surfaces using volumetric diffusion"
Marschner, Davis, Garr, and Levoy



https://graphics.stanford.edu/papers/holefill-tr-2001-07/

Application: Hole Filling

"Filling holes in complex surfaces using volumetric diffusion"
Marschner, Davis, Garr, and Levoy



NOTE: The edges/vertices on this hole boundary may not be flat / co-planar!

Usually have multiple ways to triangulate a hole, with different flatness/curvature/ 3D shape of the resulting surface.

*The hole-filling triangles might **(self-)intersect** with other hole-filling triangles or with the triangles of rest of the surface!*