Computational Geometry

Convex Hulls, Intersections, and Triangulations! Oh My!

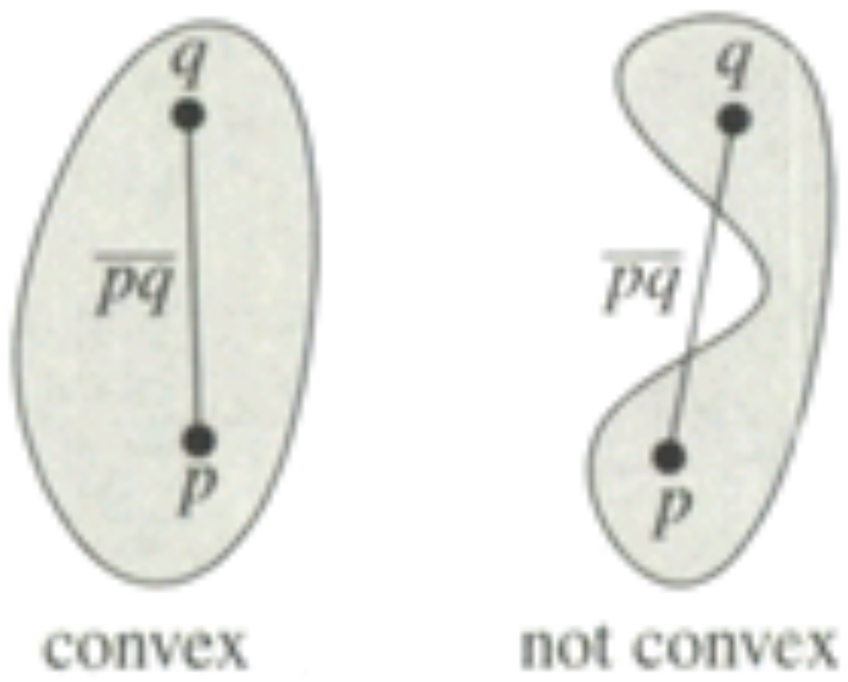
http://c.qrcard.us/pnuklo

- Michael Schade, scan or follow link to up top to view my QR Card & find me online.
    - Follow, I don’t think there are any StL people I don’t follow back
- Providing an overview
- Not proving anything, nor discussing algorithms in detail
- Showing you a few things that I’ve learned
- I am **not** an expert at this.
- Resources at end
- Will post online

Convex Hulls

http://c.qrcard.us/pnuklo

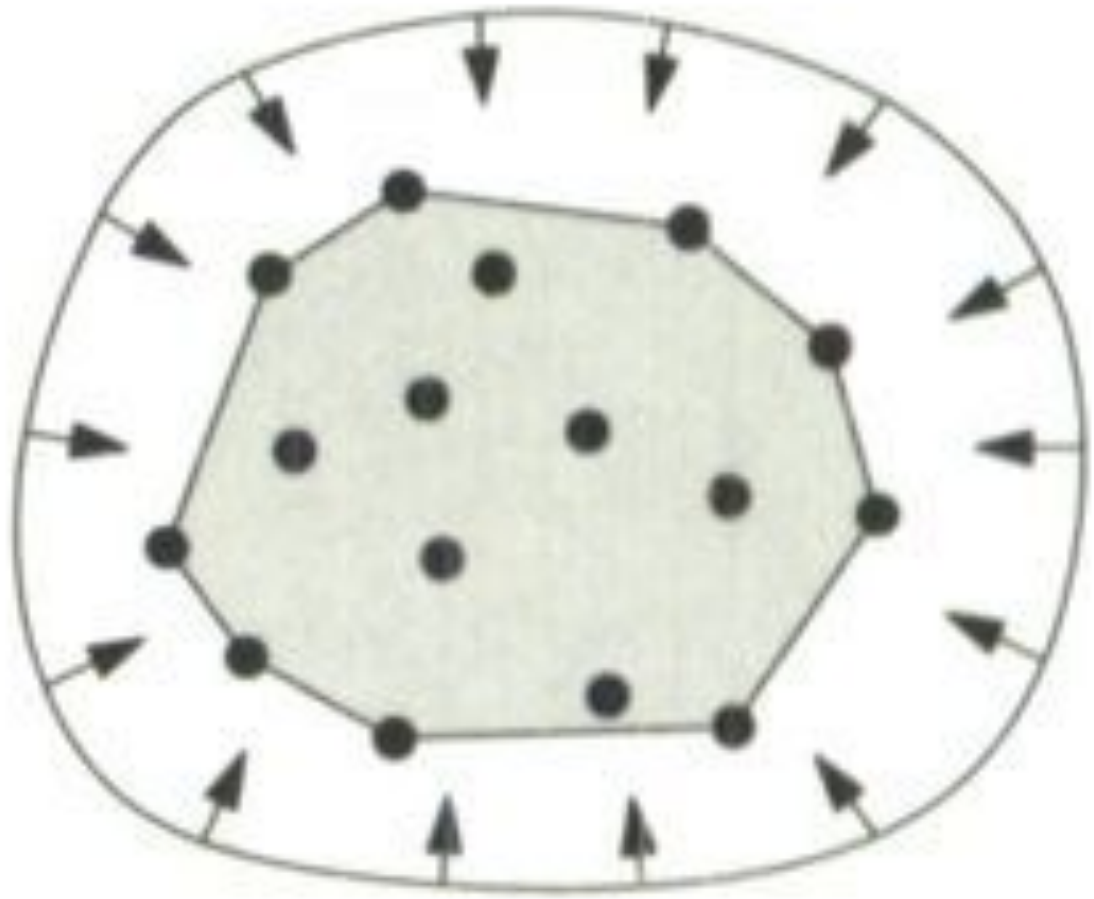
- Not going to spend much time on this one, I don't think it's as interesting



Motivation

http://c.qrcard.us/pnuklo

- Subset S of a plane is called convex iff
    - any pair of points is completely contained in it
- CH of S is smallest convex set that contains S
    - Intersection of all convex sets
- It has its uses, but for our purposes, it’s a warm-up



Goal

http://c.qrcard.us/pnuklo

- We want this.

$O(n^2)$ time

Thursday, December 1, 11

- n is the number of vertices
- Brute force: can be $O(n^2)$
- Of course, we can do better...

O(nlogn) time

Thursday, December 1, 11

- ...and we do!
- On to a demo

Demo

http://c.qrcard.us/pnuklo

Intersections

http://c.qrcard.us/pnuklo

Motivation

http://c.qrcard.us/pnuklo

- Layers of maps
- Overlay cities and roads, intersection is important
- One approach
    - Consider a set of n segments
        - Could certainly brute force this, check every pair
    - But...

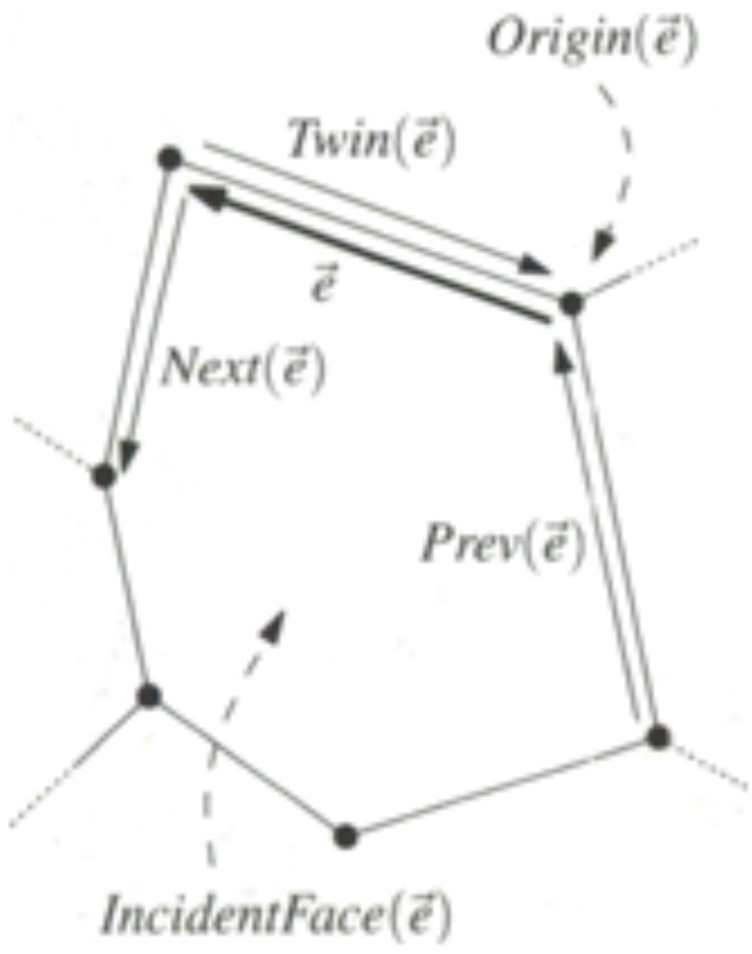
$O(n^2)$ time

Thursday, December 1, 11

- ...unnecessary quadratic running times depress countless kittens all over the world
- This isn't totally wrong, though
    - Optimal is at least n^2 if each pair of segments intersect
- Practically speaking, this isn't the case.
    - We want something better

Thursday, December 1, 11

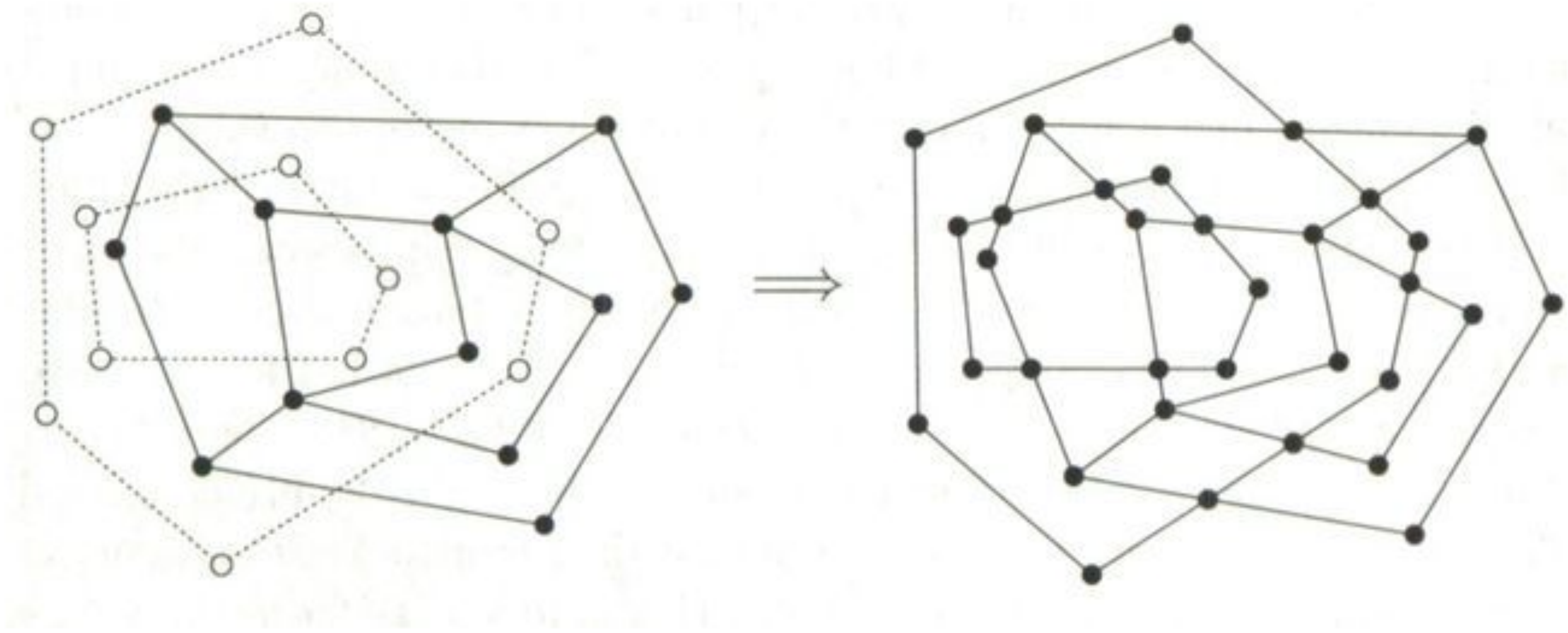
- We want an output-sensitive algorithm
    - intersection-sensitive
- “Plane sweep”
- O(nlogn + Tlogn) where T is the number of intersections
- O(n) space



DCEL

http://c.qrcard.us/pnuklo

- Contains record for every
    - face
    - edge
    - subdivision vertex



Map Overlay

http://c.qrcard.us/pnuklo

- More important than just finding intersections, we want to show regions
  - Good use case: find the bad parts of St. Louis.
  - Split StL into regions.
  - Language-based, efficiently determine if you're heading into a PHP neighborhood, turn around.

$$n = n_1 + n_2$$

$$k = \textit{overlay complexity}$$

$$O(n \log n + k \log n)$$

http://c.qrcard.us/pnuklo

- n is the sum of the segments of the two DCELs
- k is complexity of the overlay subdivision
    - this is the number of
        - vertices
        - edges
        - faces

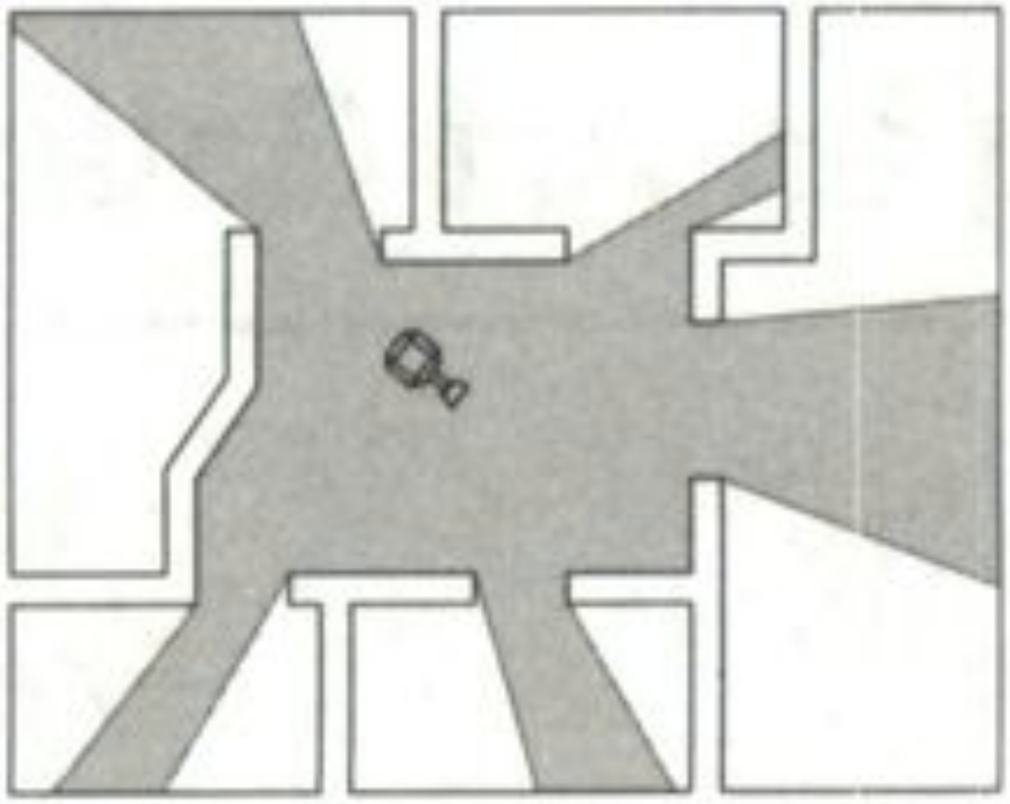
Demo

http://c.qrcard.us/pnuklo

- Best explained with a demo

Triangulations

http://c.qrcard.us/pnuklo

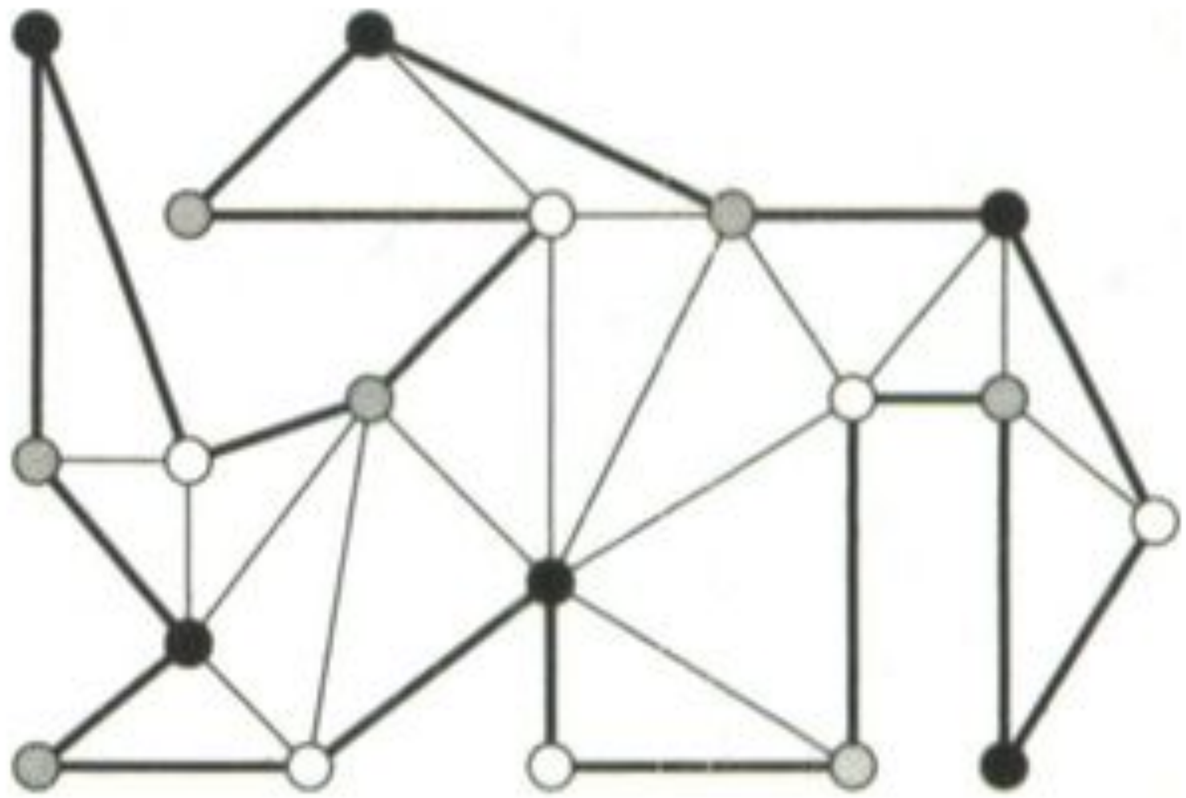


Motivation

http://c.qrcard.us/pnuklo

Thursday, December 1, 11

- Guarding an art museum
- Minimizing this number is unfortunately NP-hard
    - Still, want to protect our precious art
    - We'll find a placement of cameras
- We're concerned about **simple polygons**
    - regions enclosed by a polygon chain without intersecting itself
    - Camera sees those points that can have a diagonal within interior of polygon



Approach

http://c.qrcard.us/pnuklo

Thursday, December 1, 11

- Polygon of n vertices can be guarded by n−2 cameras
- Art costs so much money already, no room in the budget to be wasteful.
- Place cameras on diagonal for n/2
- Better yet, place on vertices.
  - We're going to do this.
  - Try for vertices that are incident to many triangles.

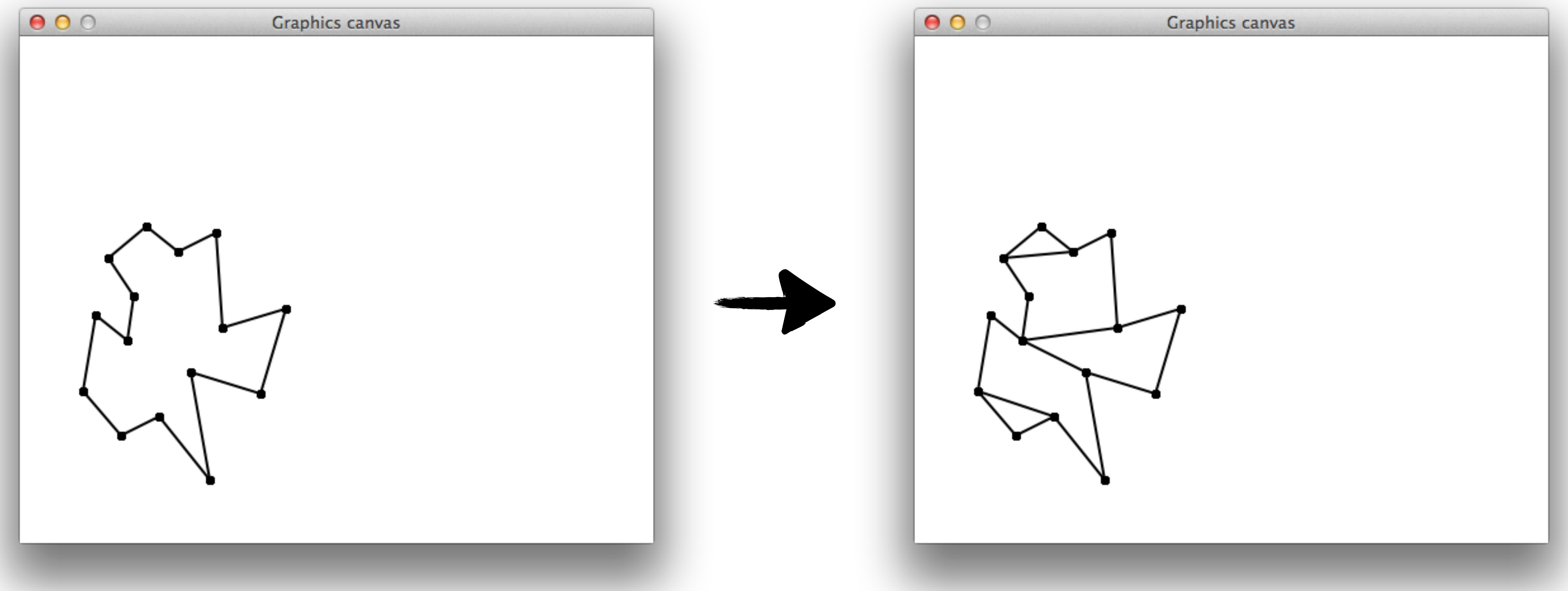
Let `P` be a simple polygon
with `n` vertices.

A set of `floor(n/3)` camera positions
in `P` such that any point inside `P` is visible
from at least one of the cameras

Time Complexity: `O(n logn)`

Theorem 3.3

http://c.qrcard.us/pnuklo

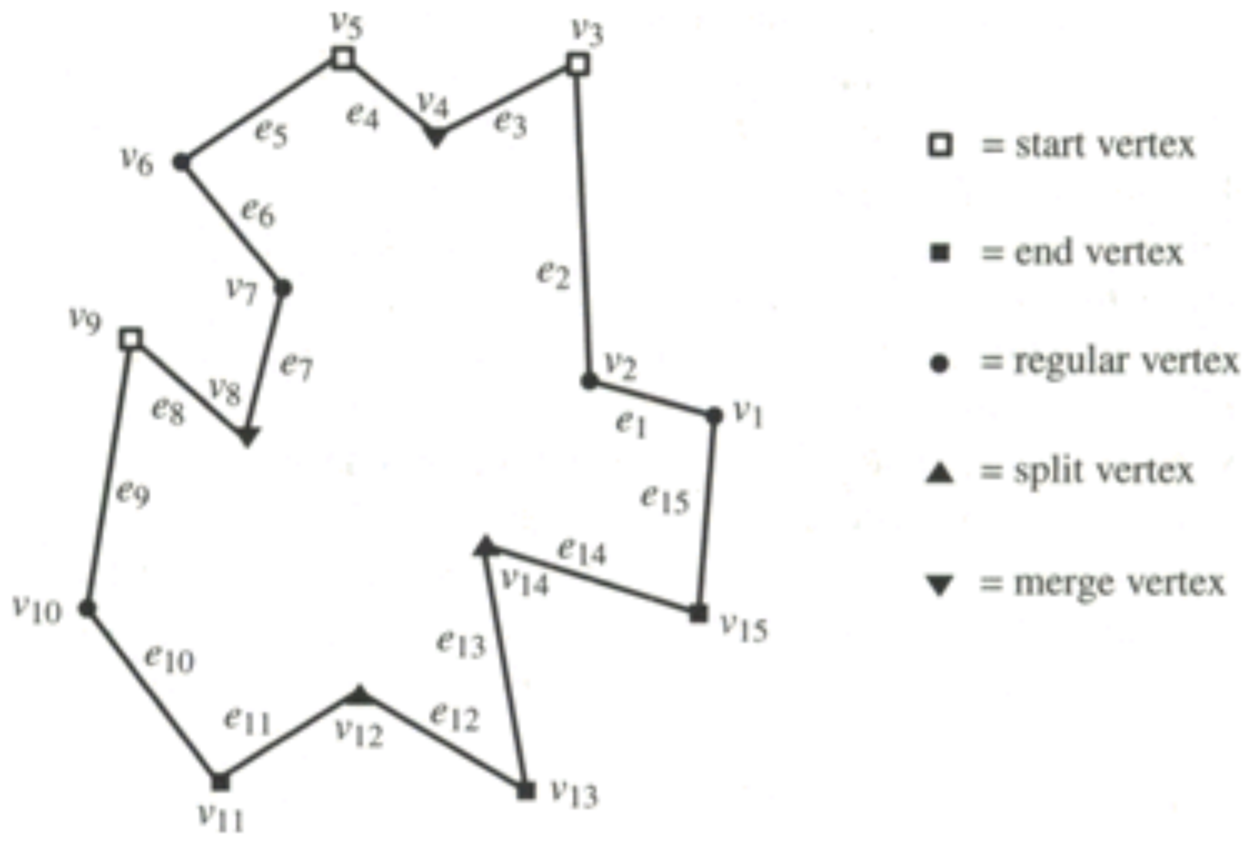


Y U MUST BE Y-Monotone?

http://c.qrcard.us/pnuklo

Thursday, December 1, 11

- Y-monotone: walk from topmost to bottommost vertex along a given chain
    - Always move downward or horizontally
- Makes it easy to draw diagonal from one vertex to all others
- Involves breaking up the original polygon into many sub-polygons
- Won't describe how to do this, but...

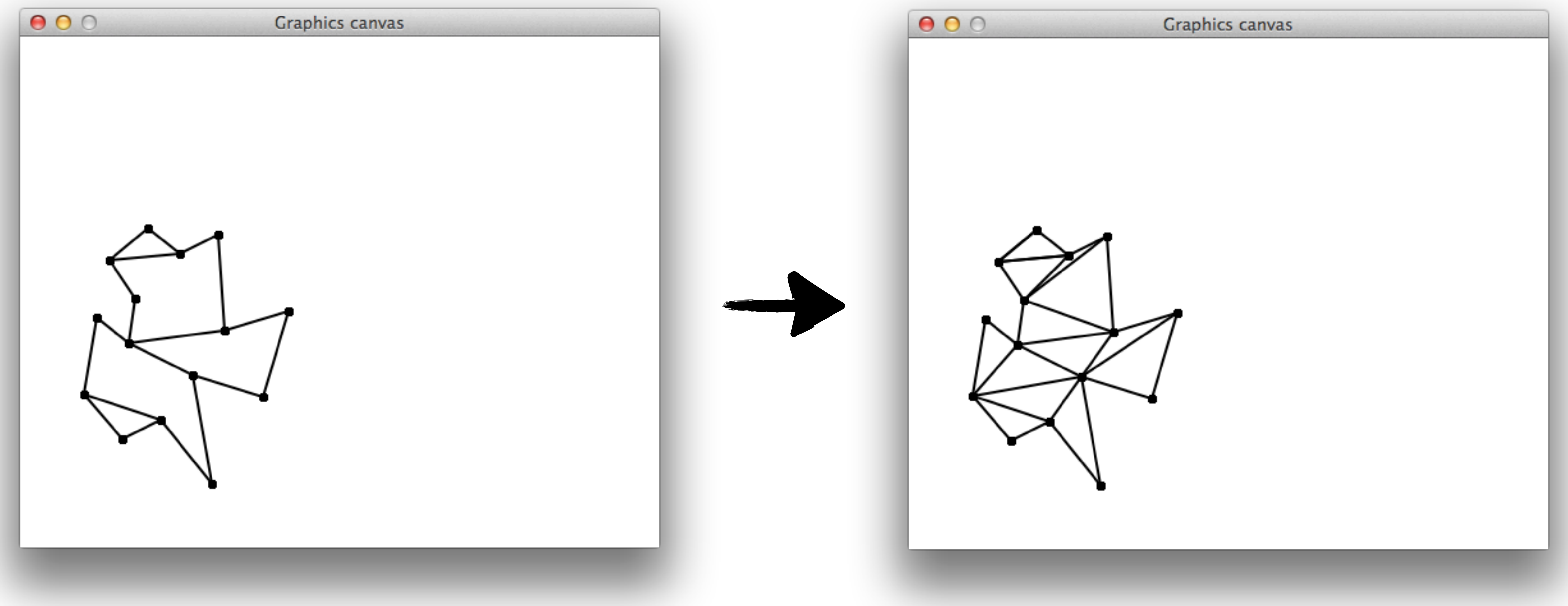


Vertex Types

http://c.qrcard.us/pnuklo

Thursday, December 1, 11

- ...I will let you know that these are the types of vertices you use to do so.



Now, Triangulate!

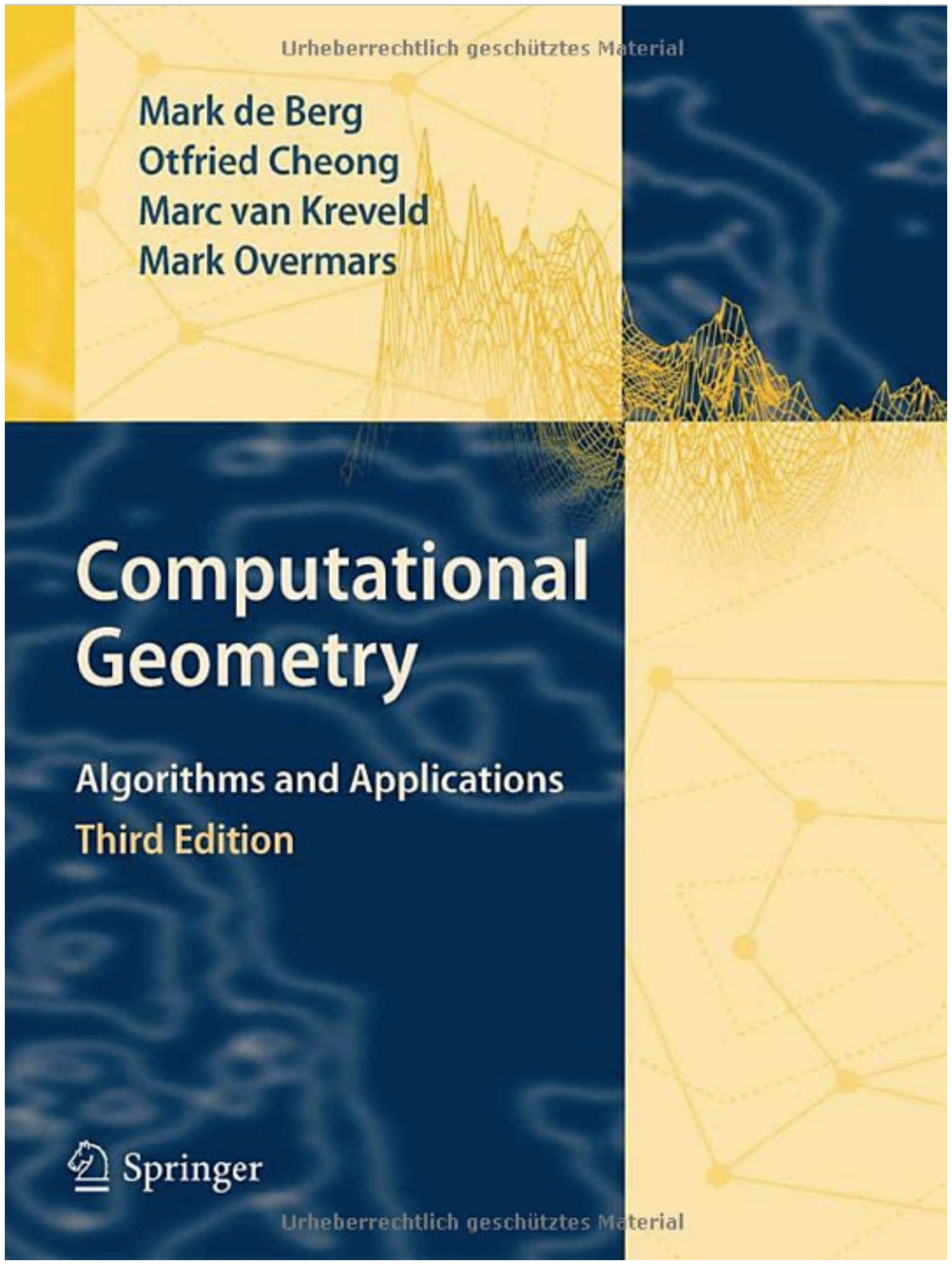
http://c.qrcard.us/pnuklo

- This part is really easy
- Triangulate each of the faces
- Algorithm is in the book

Resource: Book

Computational Geometry: Algorithms and Applications

by Mark de Berg
, Otfried Cheong
, Marc van Kreveld
, Mark Overmars



http://c.qrcard.us/pnuklo

Thursday, December 1, 11

- Written very well
- If you buy the ebook, be careful. Code not indented, at least on Kindle :-(

Resource: Person

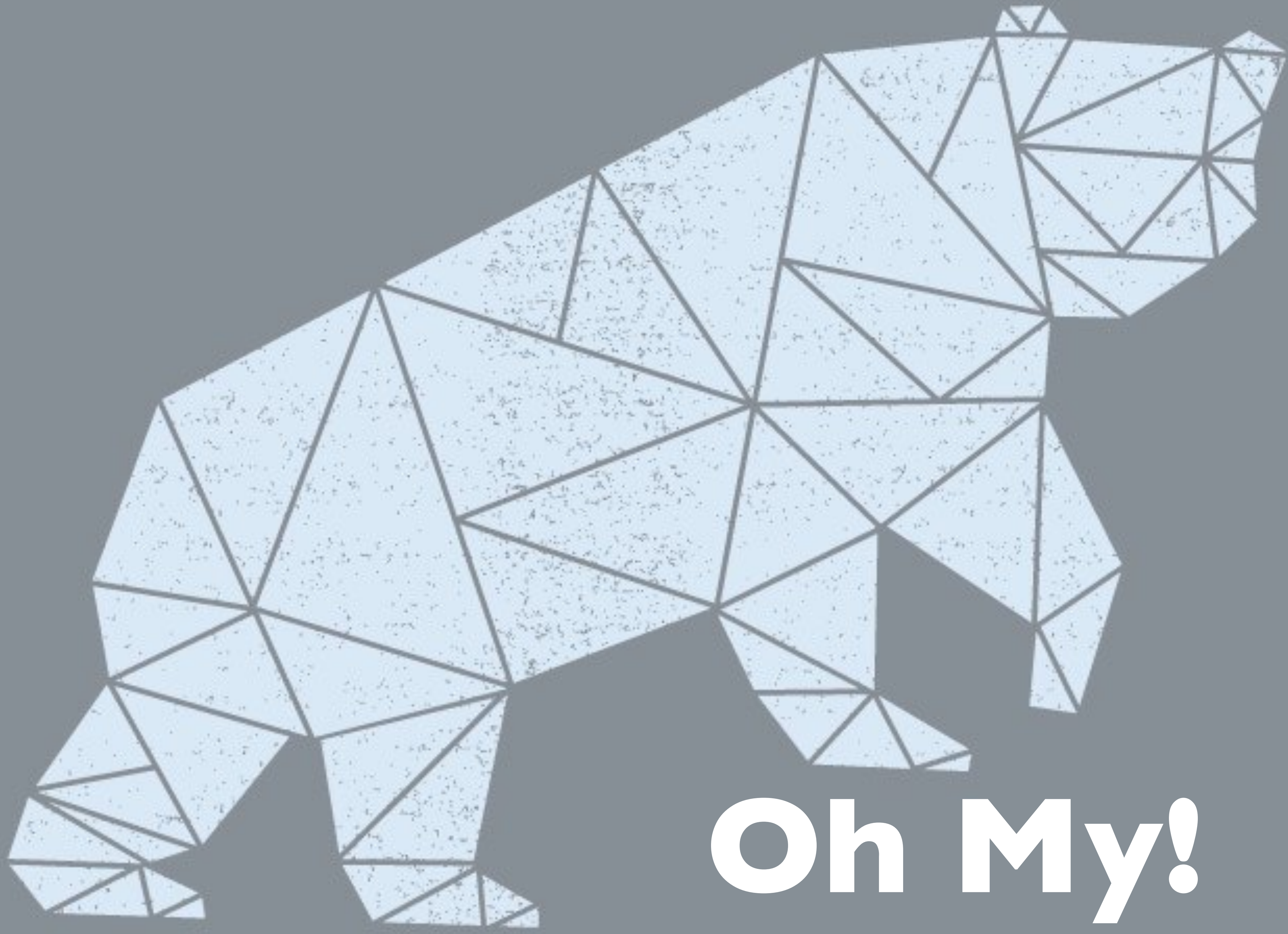
Dr. Michael Goldwasser
Director of Computer Science
Saint Louis University

http://cs.slu.edu/~goldwamh/

http://c.qrcard.us/pnuklo

Thursday, December 1, 11

- Spoken at Lambda Lounge before
- Much of the code was from him
- He spent a lot of time going over algorithms, drawing pretty pictures
- Came up with nicer, more general variations of the algorithms



Oh My!

http://c.qrcard.us/pnuklo

Thursday, December 1, 11

Lions, tigers, and bears-esque run-through of computational geometry.