Computational Geometry

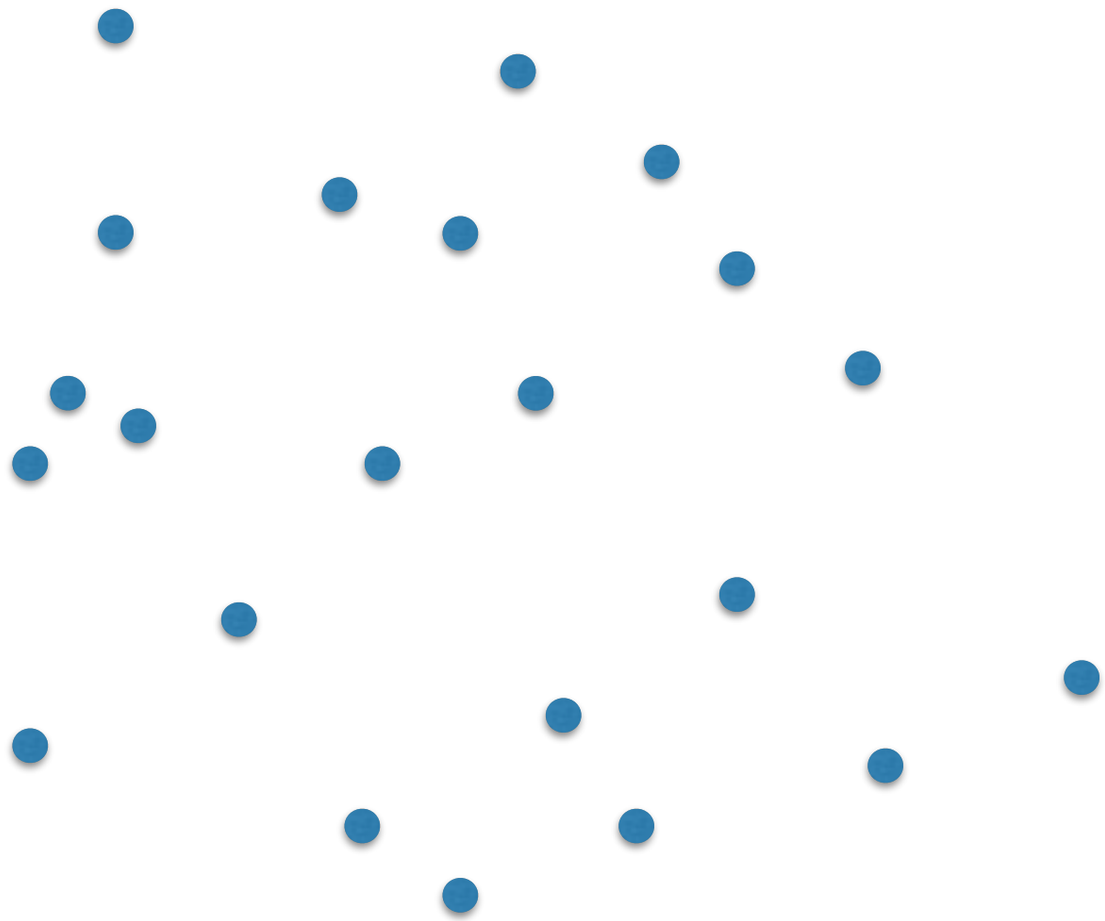
(csci3250)

Laura Toma

Bowdoin College

Introduction

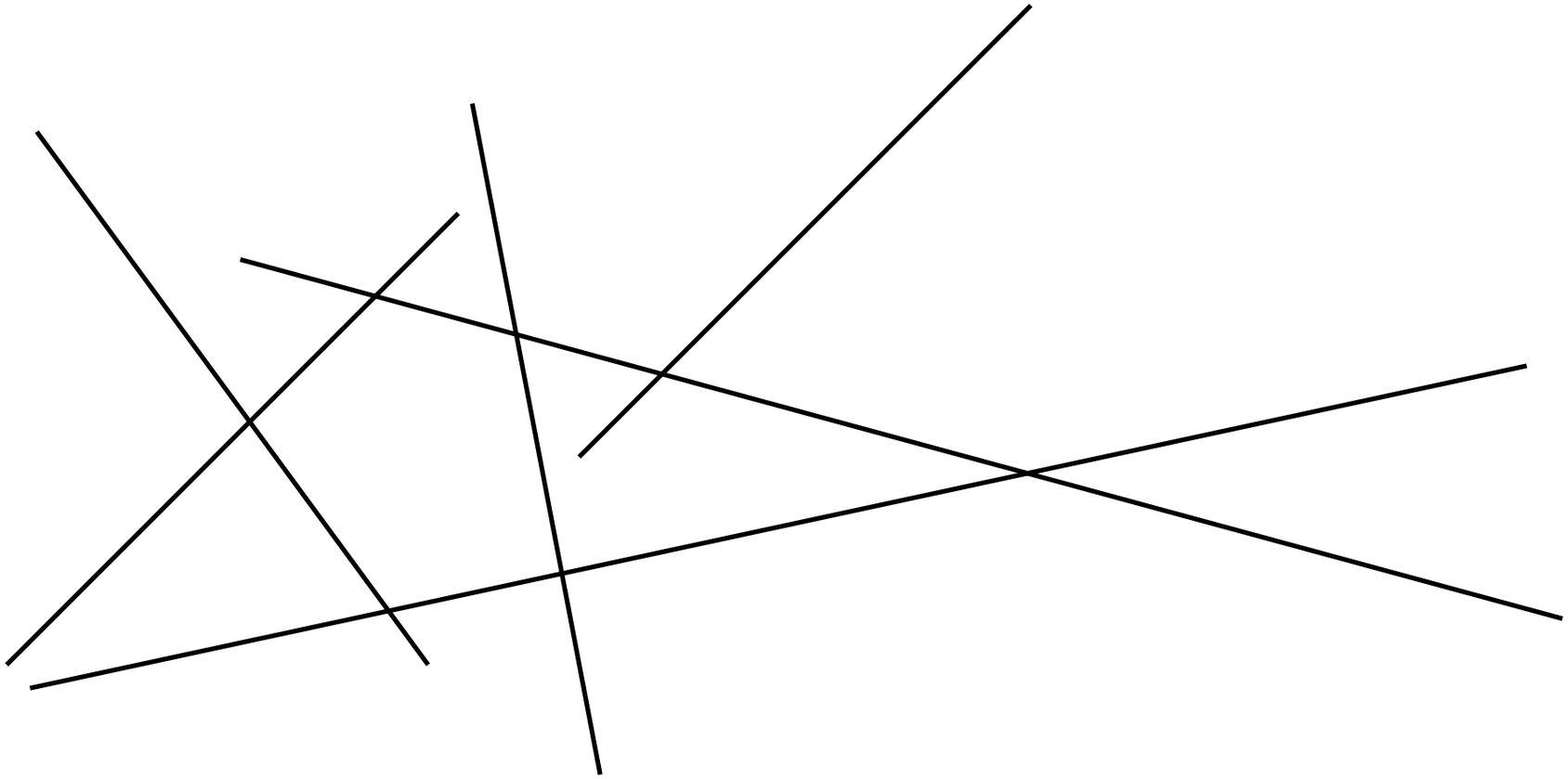
- CG deals with algorithms for geometric data



points

Introduction

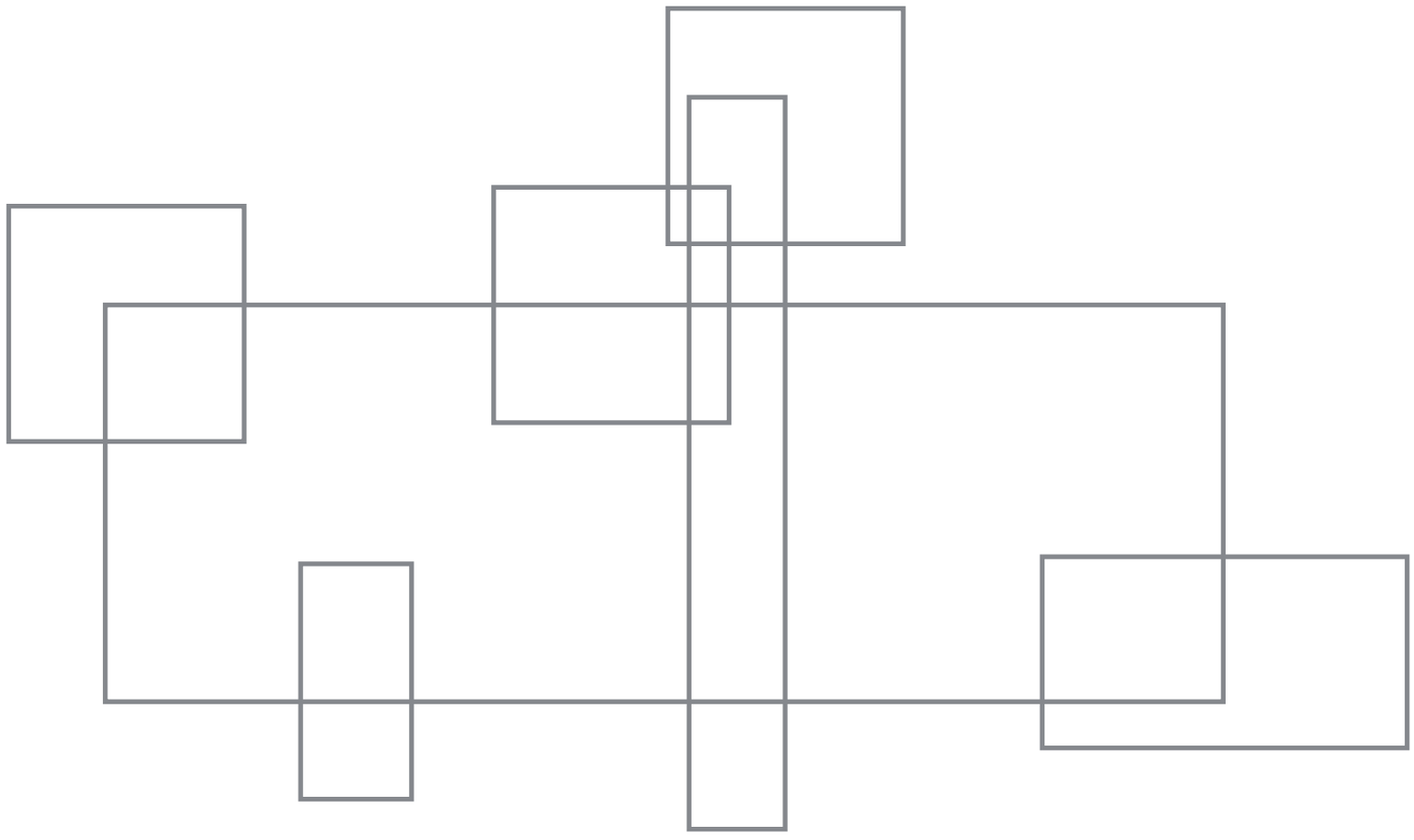
- CG deals with algorithms for geometric data



lines and line segments

Introduction

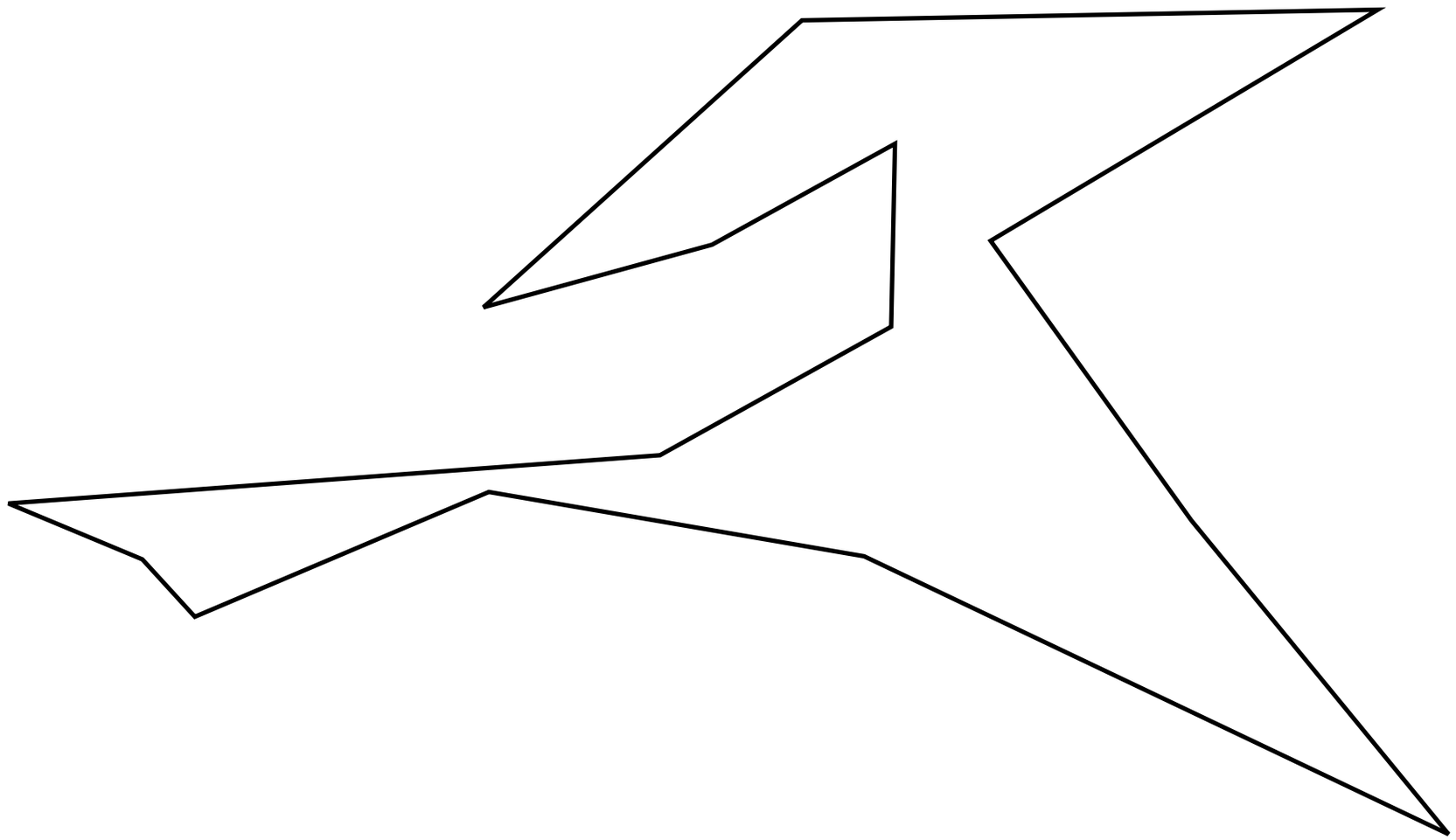
- CG deals with algorithms for geometric data



polygons

Introduction

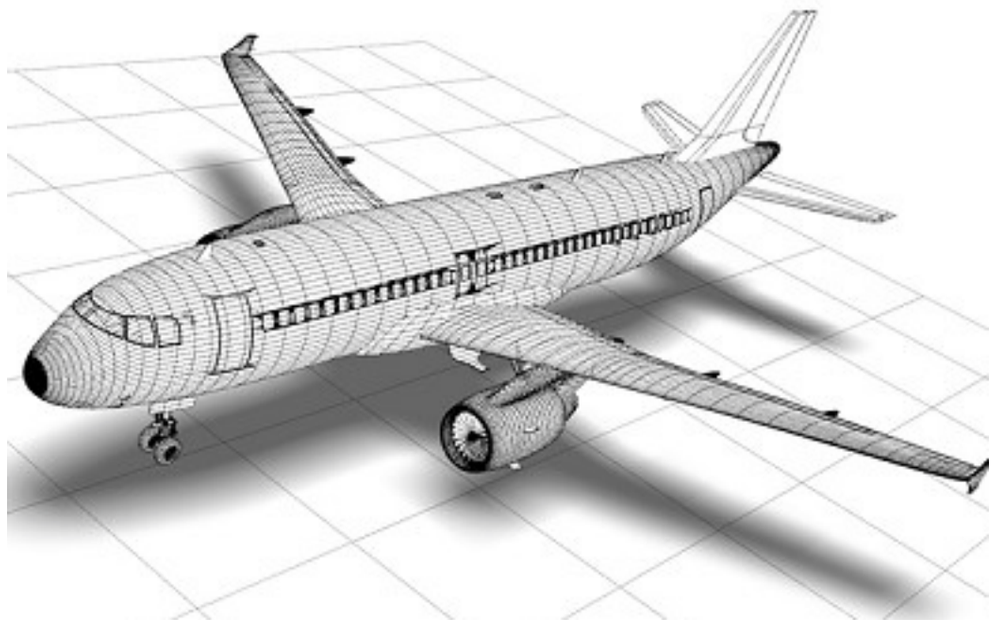
- CG deals with algorithms for geometric data



polygons

Introduction

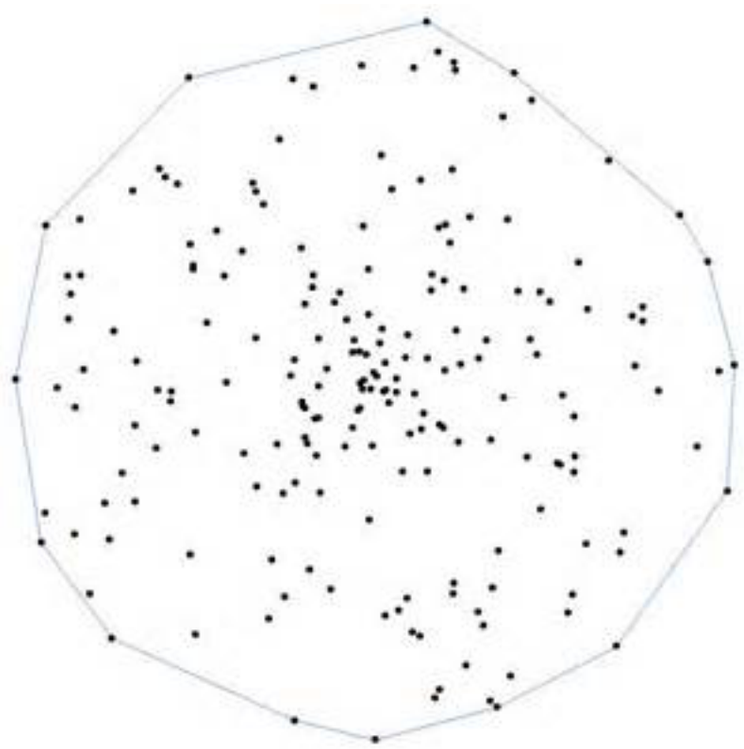
- CG deals with algorithms for geometric data



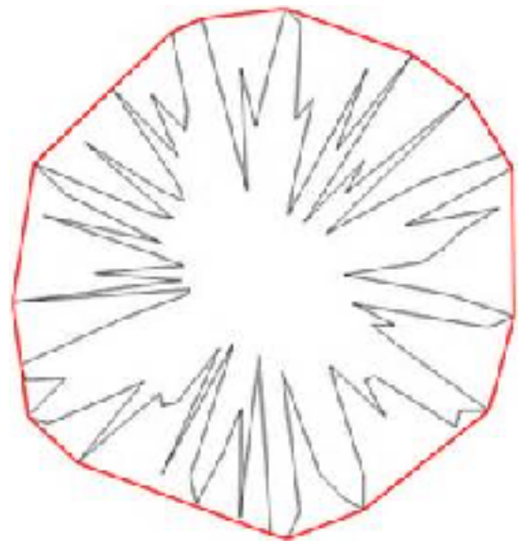
2D, 3D..

Class overview

- Convex hull

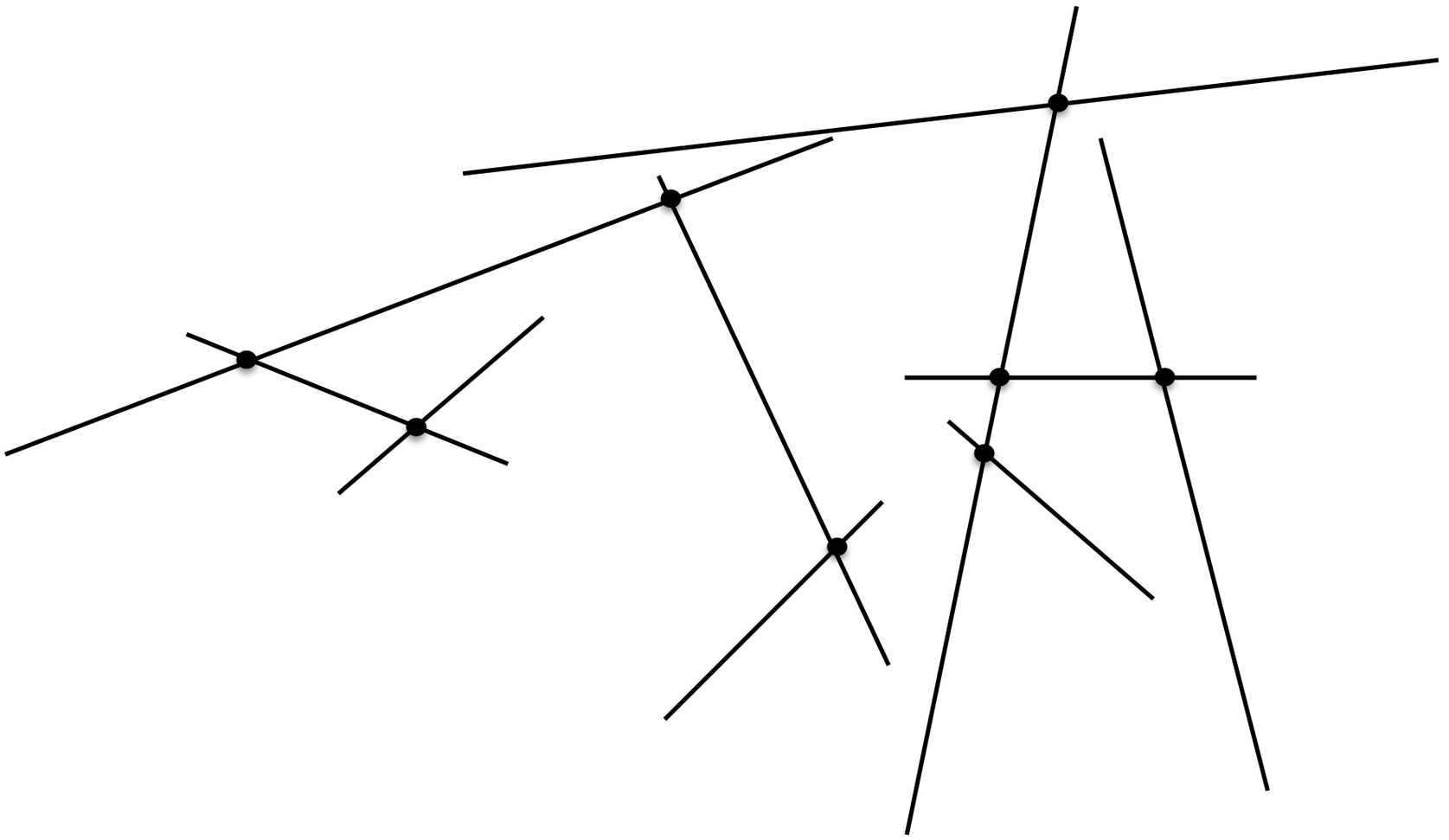


- Applications: objects are approximated by their CH shape



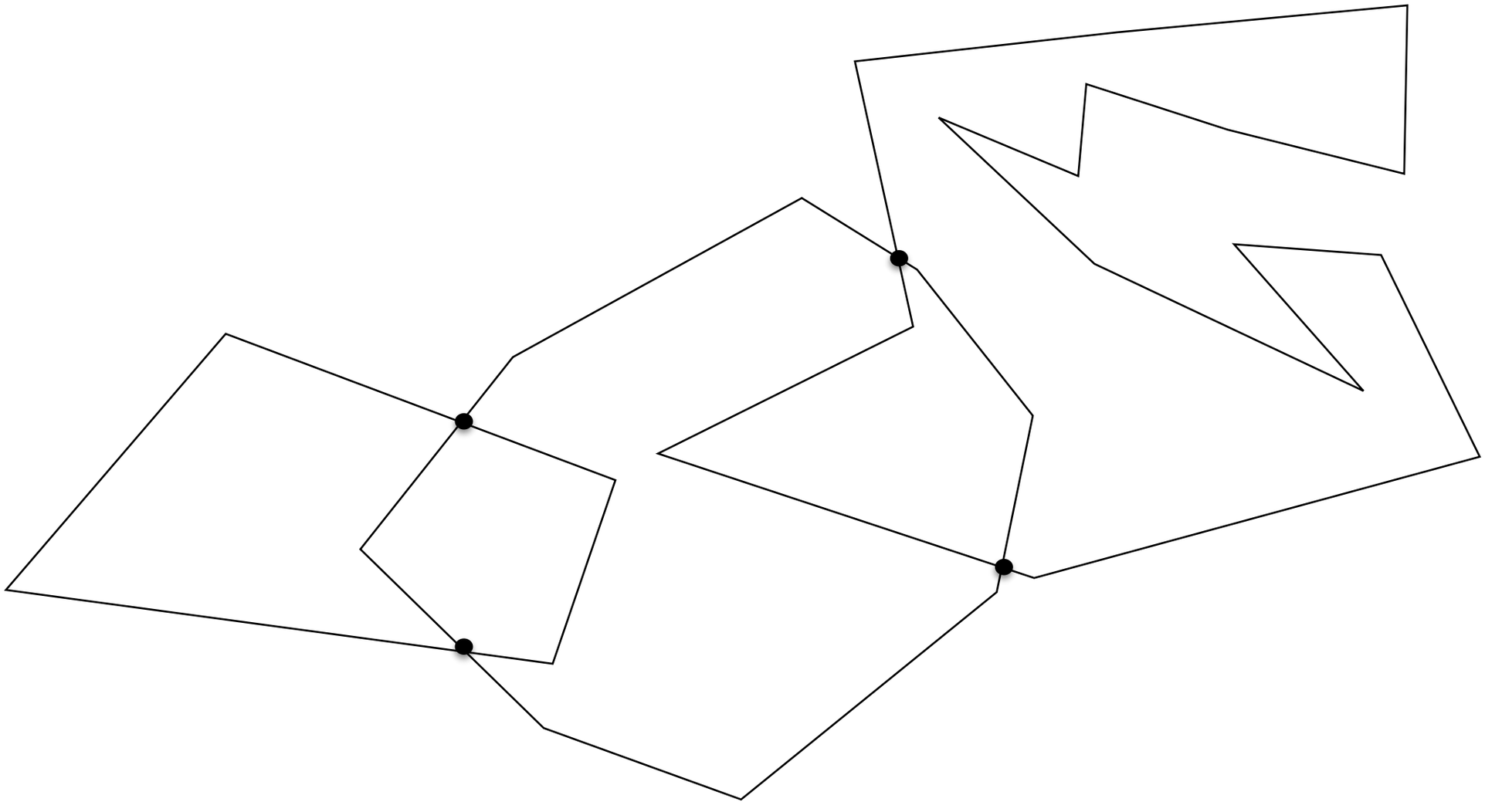
Class overview

- Line segment intersection



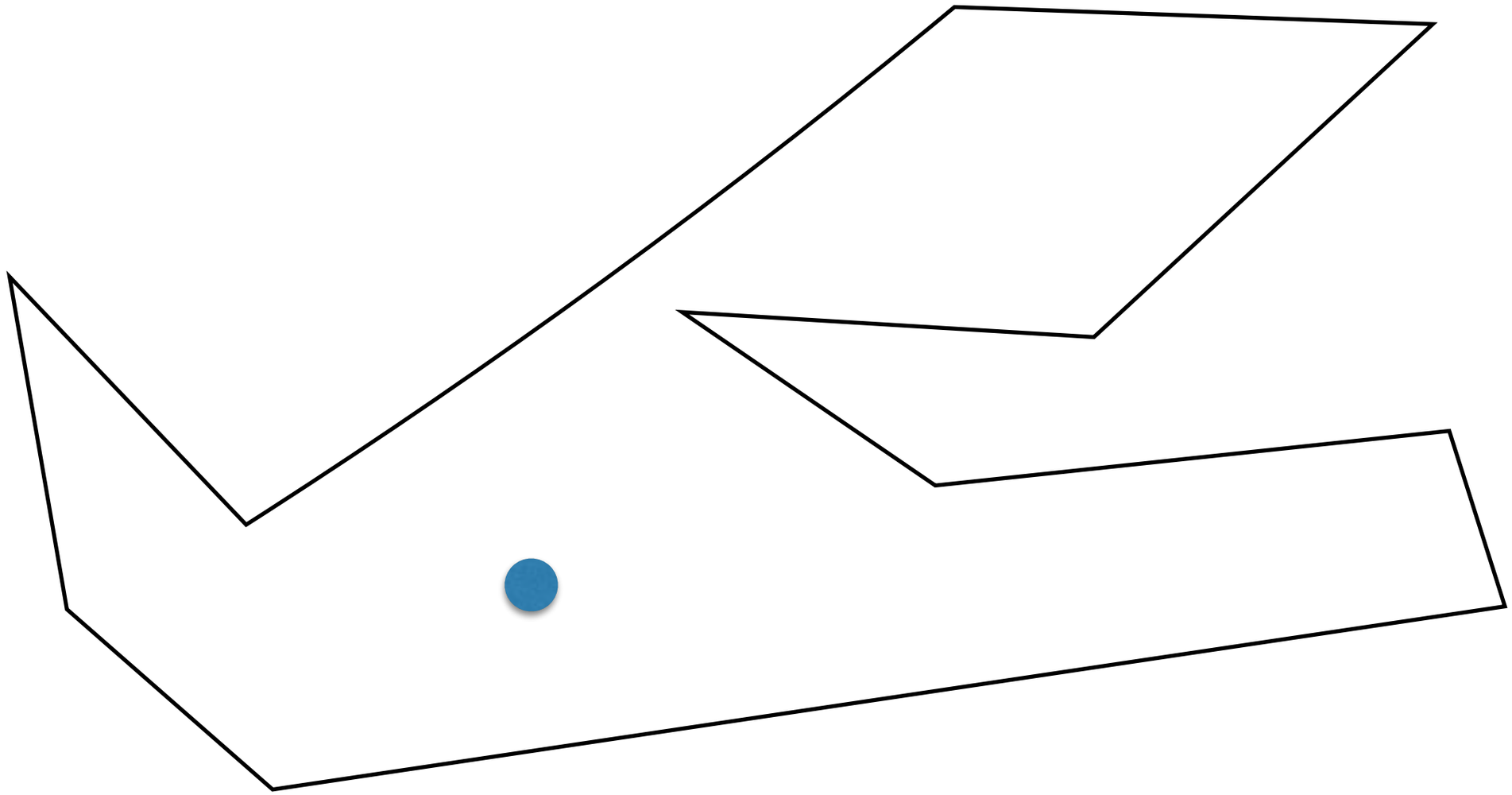
Class overview

- Line segment intersection



Class overview

- Visibility: art gallery problem

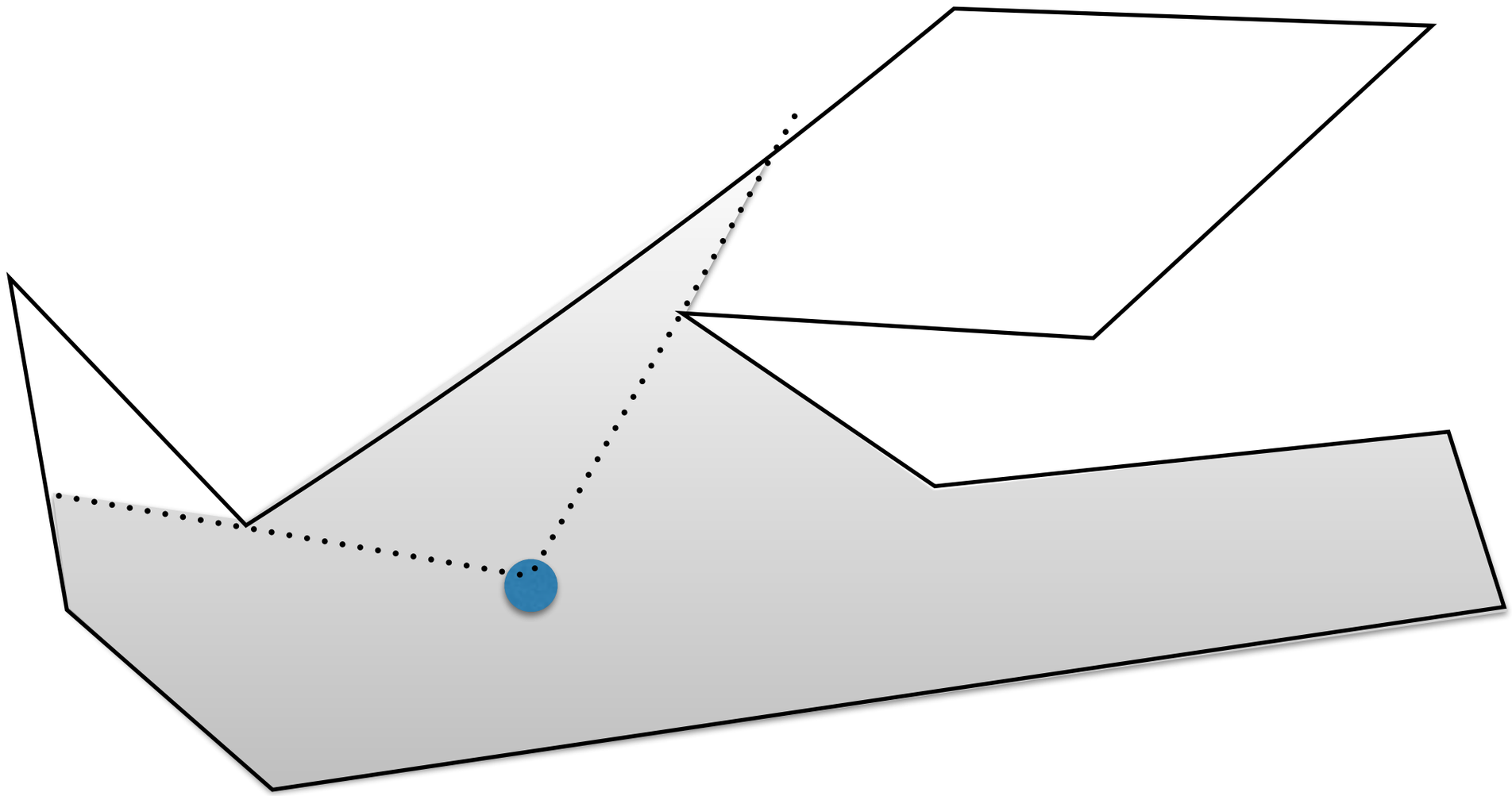


What part of the polygon can the guard see?

How many guards necessary to cover this polygon?

Class overview

- Visibility: art gallery problem



What part of the polygon can the guard see?

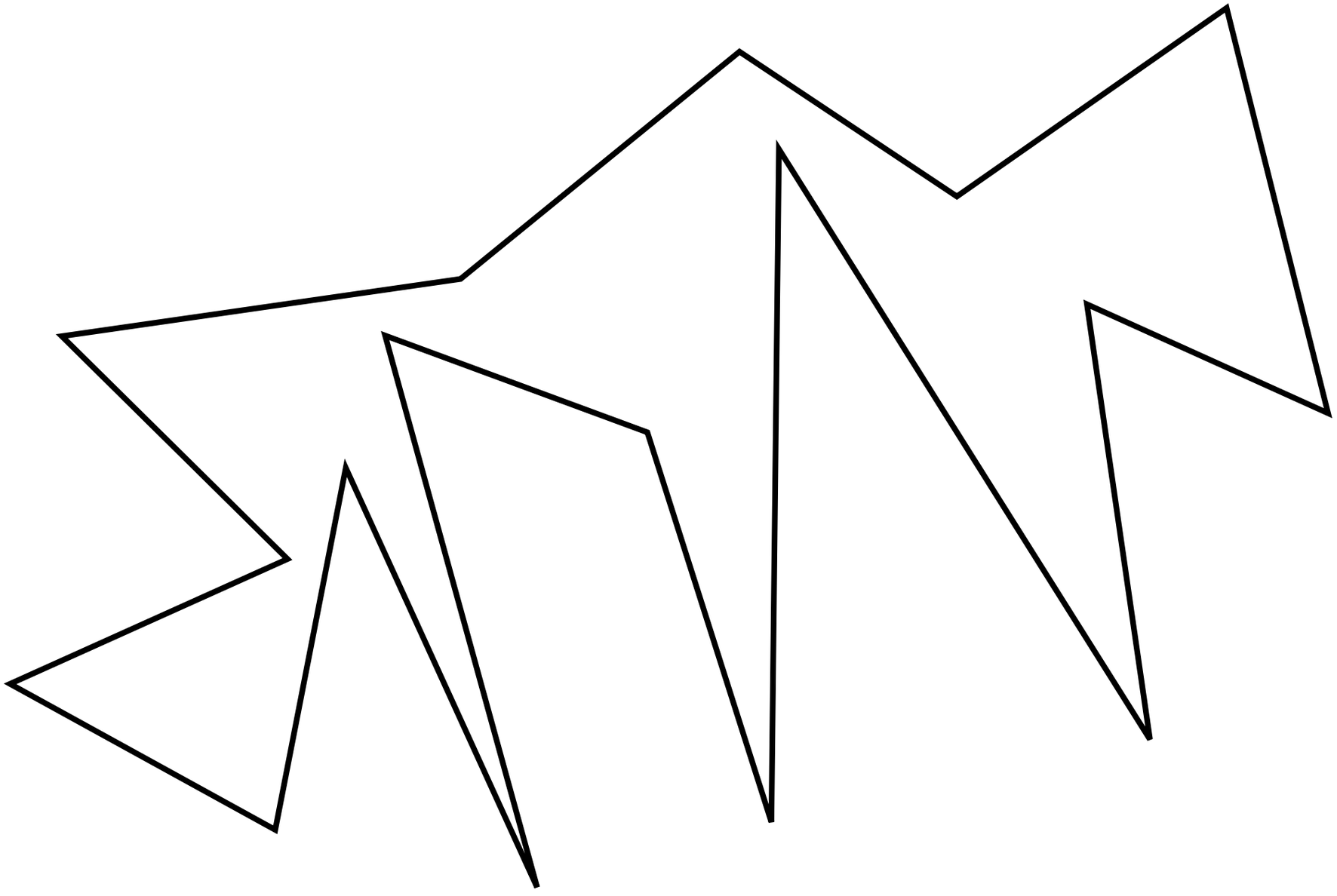
How many guards necessary to cover this polygon?

Class overview

- Partitioning
  - subdivide a complex domain into simpler objects
  - simplest object: triangle => triangulation

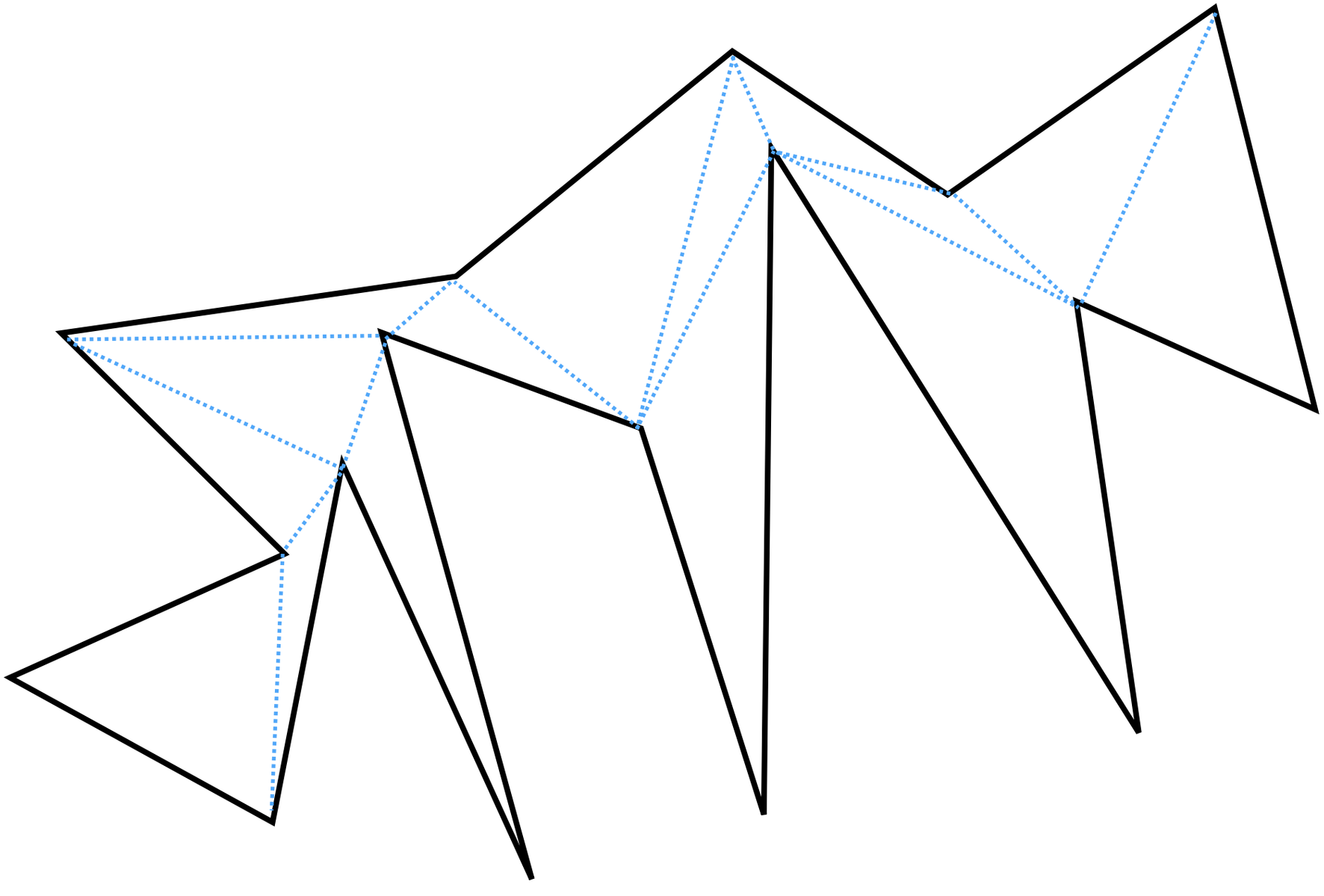
Class overview

- Polygon triangulation
  - output a set of diagonals that partition the polygon into triangles



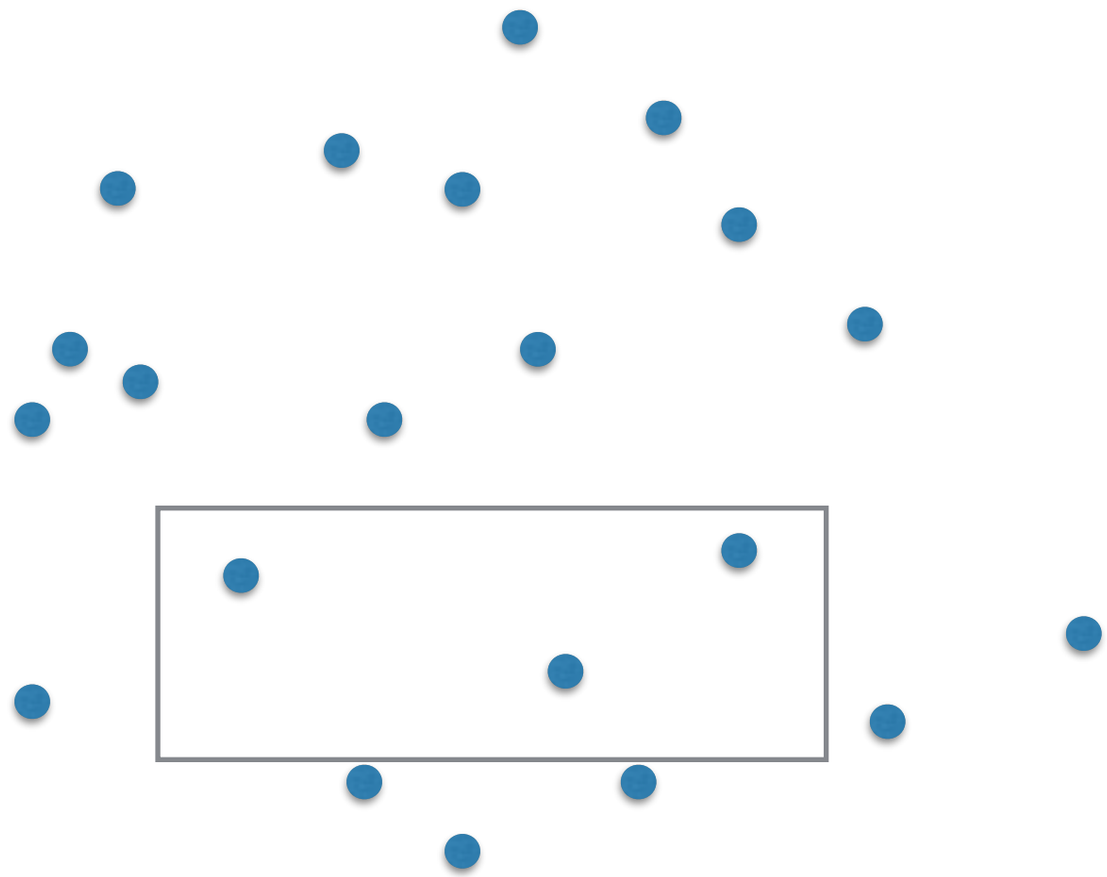
Class overview

- Polygon triangulation
  - output a set of diagonals that partition the polygon into triangles



Class overview

- Range searching

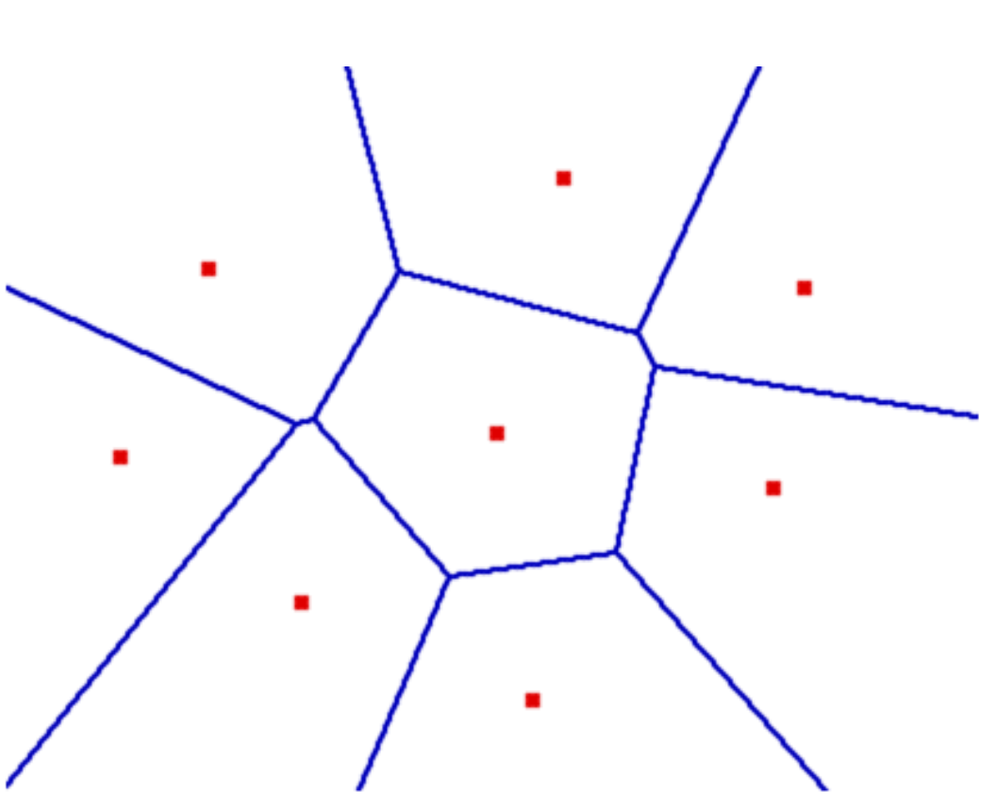


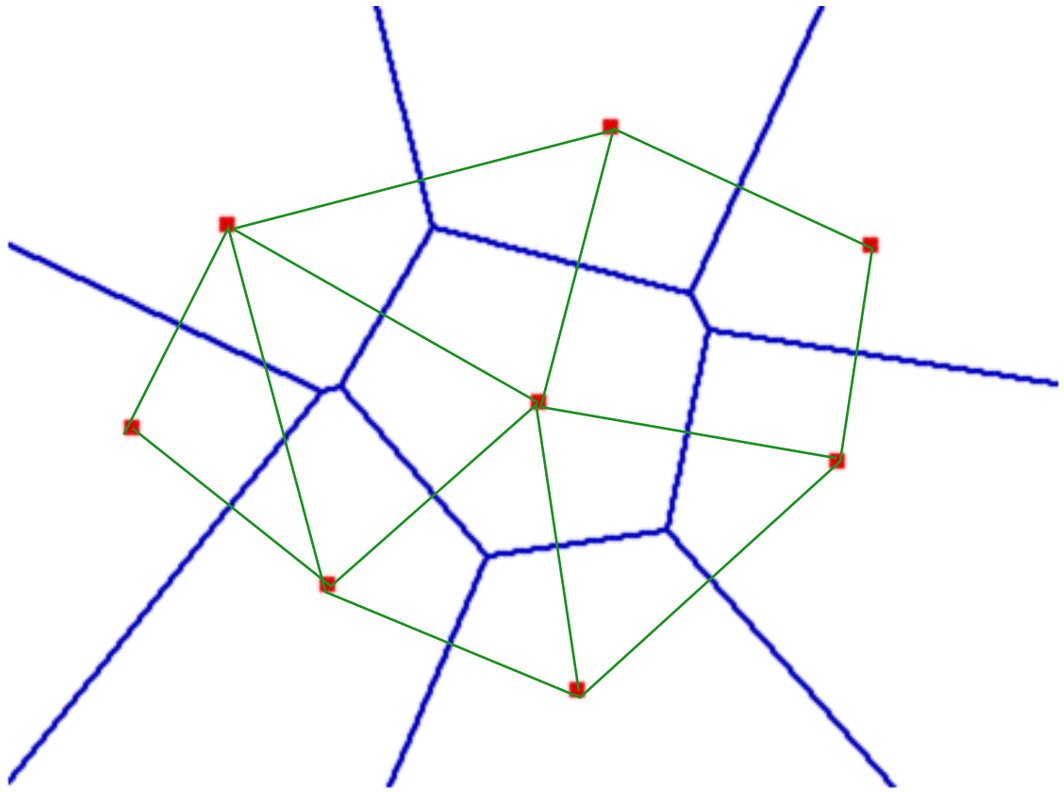
find all points in this range

- kd-tree
- range tree

Class overview

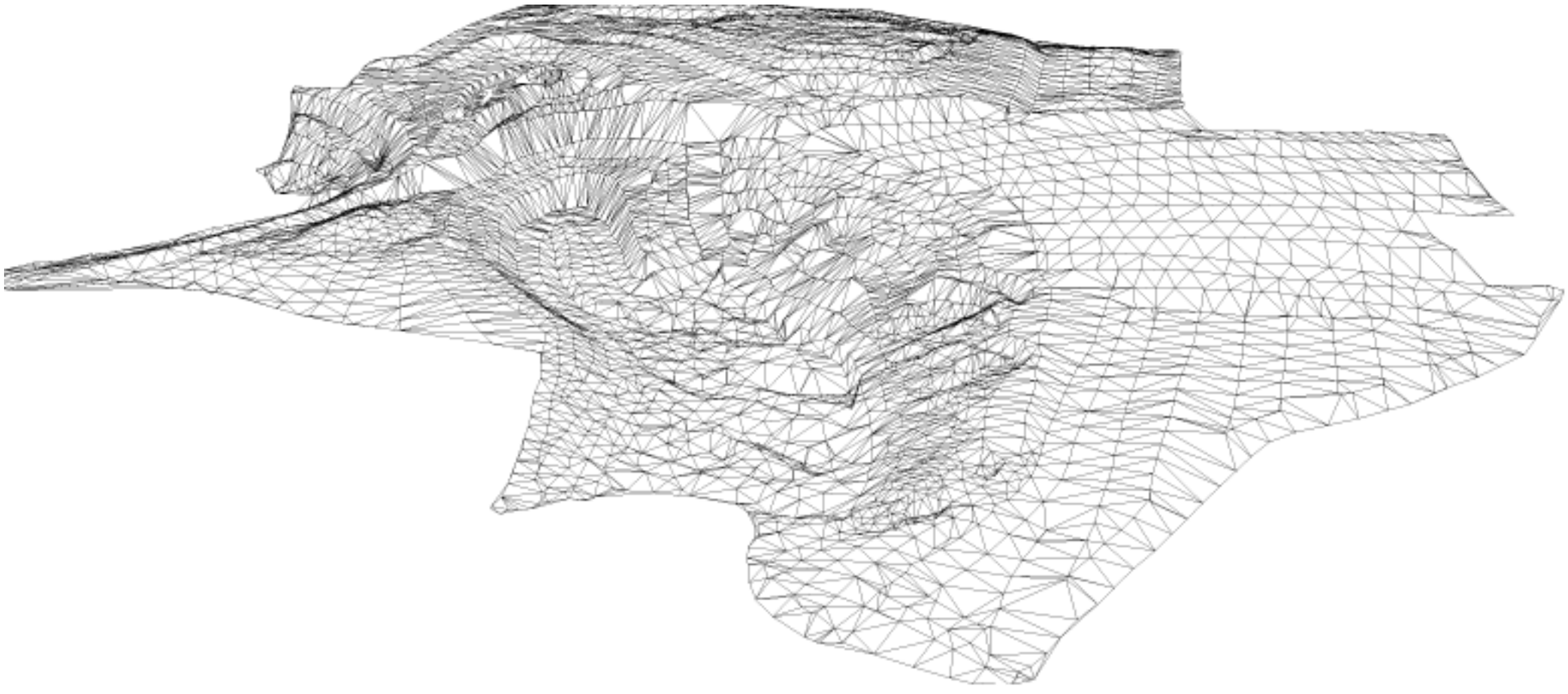
- Voronoi diagram and Delaunay triangulations





Class overview

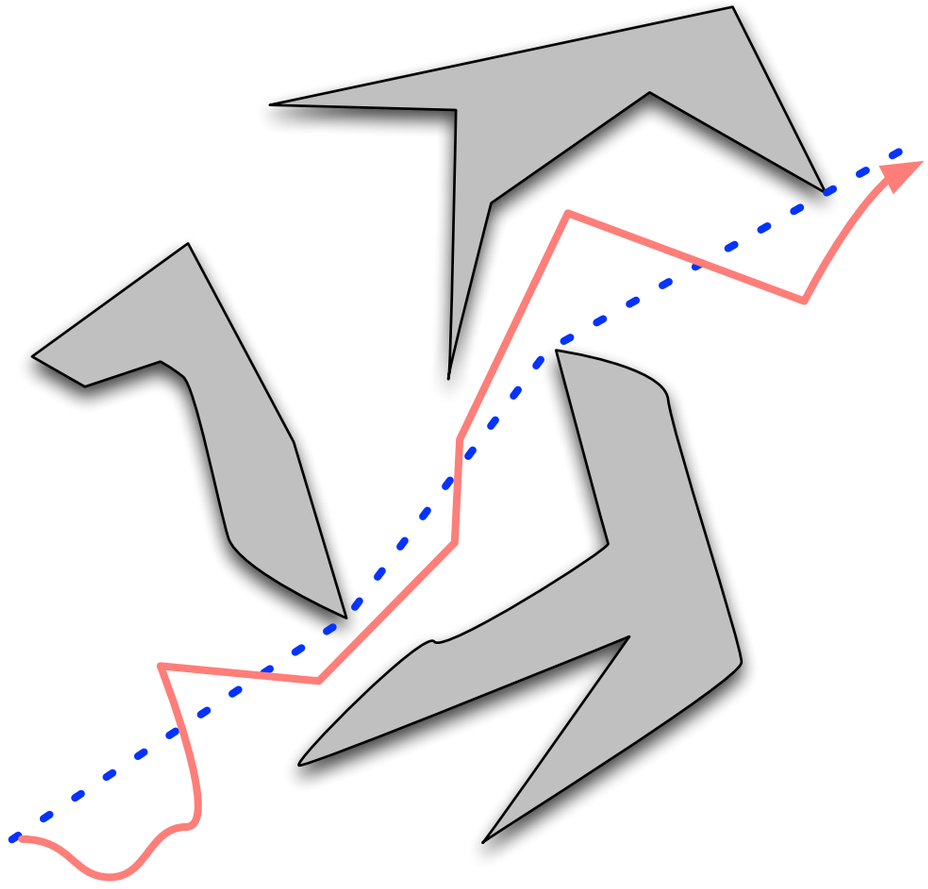
- Voronoi diagram and Delaunay triangulations



used in meshing

Class overview

- Path planning : find collision-free path from start to end



Computational geometry

- We'll talk about algorithms
- The usual questions
  - Properties
  - Complexity of the result?
  - Worst-case running time?
  - Can we do better?
  - Lower bound for the problem?
  - Is the algorithm practical?
  - Handle degeneracies in the input?
  - Can we make some practical assumptions about the data?

Applications

- Computer graphics
  - rendering, hidden surface removal, lighting, moving, collision detection,..

- Robotics and motion planning
  - path planning involves finding paths that avoid obstacles; this involves finding intersections

- Spatial database engines
  - contain specialized data structures for answering queries on geometric data, such as finding all intersections between two sets of line segments (road and rivers)

- Cell phone data
  - Data: stream of coordinates (x,y, time)
  - Problems: find congestion patterns, model real-time traffic conditions (done by cell phone apps)