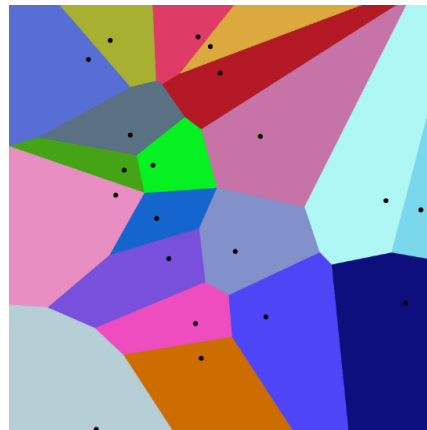
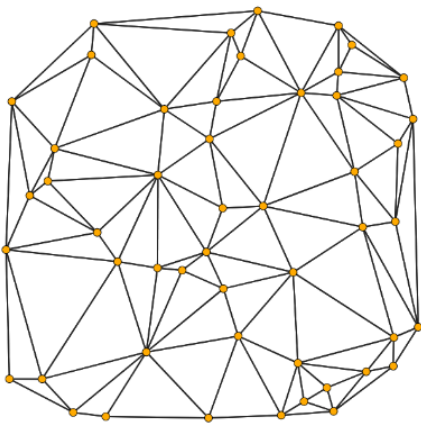


# Computational geometry

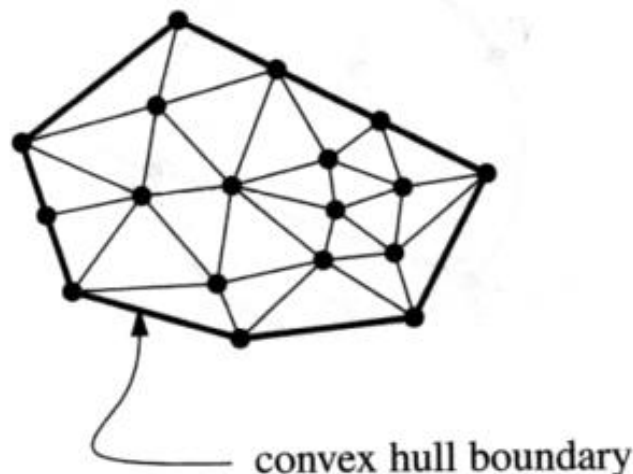
## II - Delaunay and Voronoi



Stefan BORNHOFEN

# Triangulation

- Let  $P = \{p_1, \dots, p_n\} \subseteq \mathbb{R}^2$  be a finite set of points in the plane.
- A **triangulation of  $P$**  is a simple, planar, connected graph  $T=(P,E)$  such that
  - every edge in  $E$  is a line segment between points of  $P$ ,
  - all inner faces are triangles.
  - the outer face is bounded by the convex hull of  $P$



# Theorem

Let  $P$  be a set of  $n$  points in the plane.

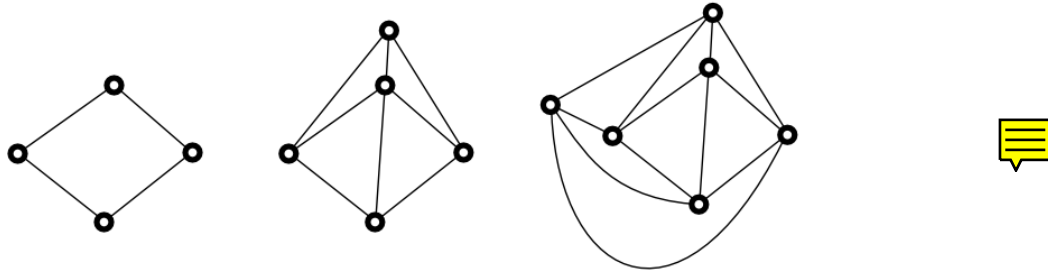
How many triangles and how many edges does a triangulation produce?

# Theorem

Let  $P$  be a set of  $n$  points in the plane.

How many triangles and how many edges does a triangulation produce?

(1) Let  $T$  be the number of triangles and  $E$  the number of edges in the triangulation. Euler's formula for connected planar graphs: " $v+f-e = 2$ "

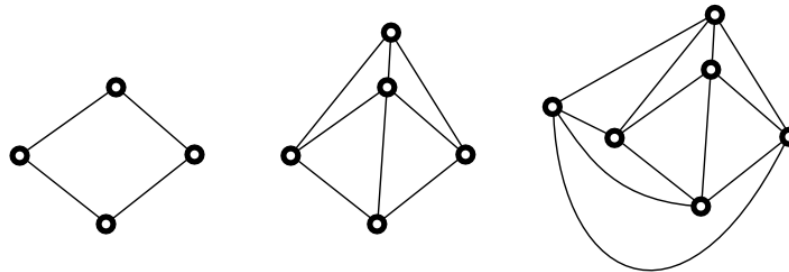


# Theorem

Let  $P$  be a set of  $n$  points in the plane.

How many triangles and how many edges does a triangulation produce?

(1) Let  $T$  be the number of triangles and  $E$  the number of edges in the triangulation. Euler's formula for connected planar graphs: " $v+f-e = 2$ "



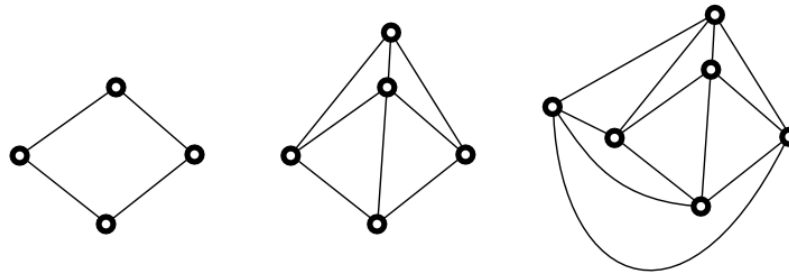
$$\Rightarrow n + (T+1) - E = 2 \quad (+1 \text{ for the outer region})$$

# Theorem

Let  $P$  be a set of  $n$  points in the plane.

How many triangles and how many edges does a triangulation produce?

(1) Let  $T$  be the number of triangles and  $E$  the number of edges in the triangulation. Euler's formula for connected planar graphs: " $v+f-e = 2$ "



$$\Rightarrow n + (T+1) - E = 2 \quad (+1 \text{ for the outer region})$$

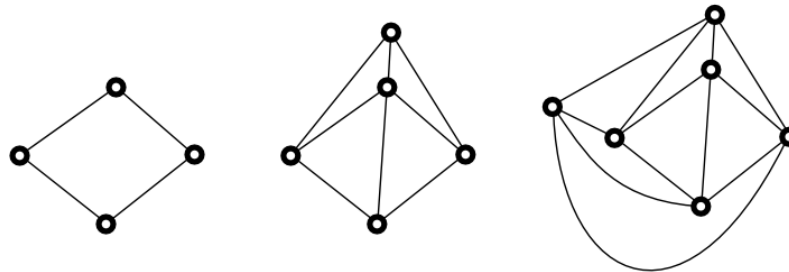
(2) Moreover, each triangle has 3 edges and each edge is incident to 2 triangles except for the convex hull. Suppose  $h$  points of  $P$  lie on the convex-hull boundary.

# Theorem

Let  $P$  be a set of  $n$  points in the plane.

How many triangles and how many edges does a triangulation produce?

(1) Let  $T$  be the number of triangles and  $E$  the number of edges in the triangulation. Euler's formula for connected planar graphs: " $v+f-e = 2$ "



$$\Rightarrow n + (T+1) - E = 2 \quad (+1 \text{ for the outer region})$$

(2) Moreover, each triangle has 3 edges and each edge is incident to 2 triangles except for the convex hull. Suppose  $h$  points of  $P$  lie on the convex-hull boundary.

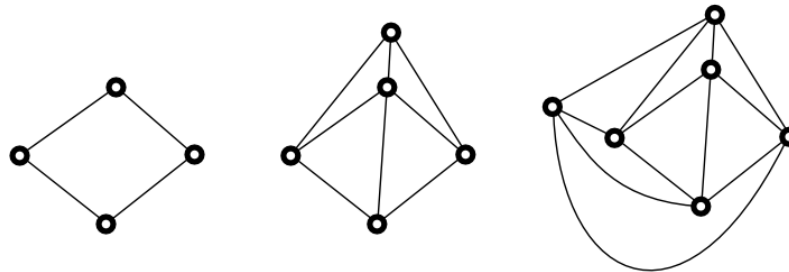
$$\Rightarrow 3T = 2E - h$$

# Theorem

Let  $P$  be a set of  $n$  points in the plane.

How many triangles and how many edges does a triangulation produce?

(1) Let  $T$  be the number of triangles and  $E$  the number of edges in the triangulation. Euler's formula for connected planar graphs: " $v+f-e = 2$ "



$$\Rightarrow n + (T+1) - E = 2 \quad (+1 \text{ for the outer region})$$

(2) Moreover, each triangle has 3 edges and each edge is incident to 2 triangles except for the convex hull. Suppose  $h$  points of  $P$  lie on the convex-hull boundary.

$$\Rightarrow 3T = 2E - h$$

Combining (1) and (2):

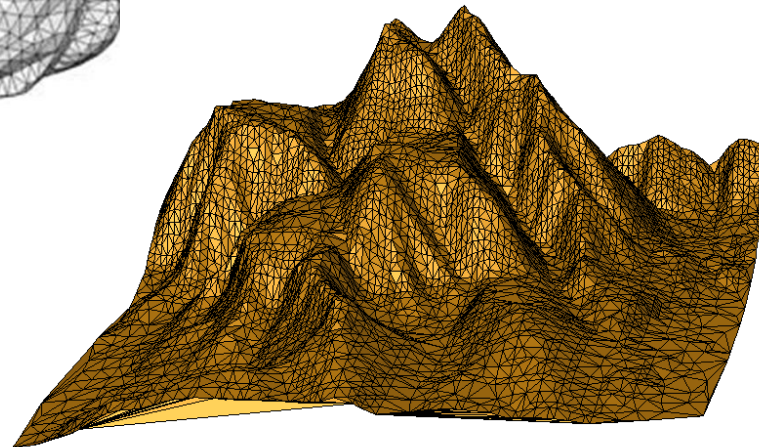
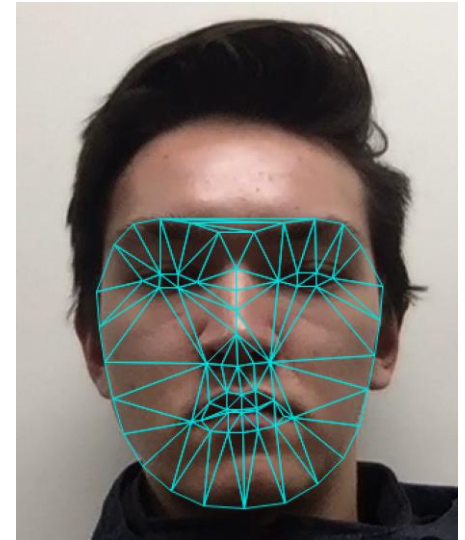
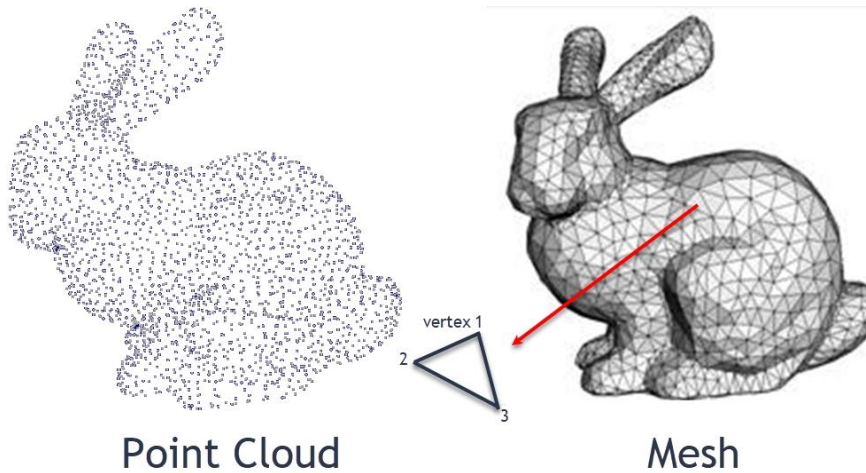
$$T = 2n - h - 2$$

$$E = 3n - h - 3.$$



# Why triangulation?

- Meshing a point cloud
- Terrain generation from elevation data
- Face recognition systems



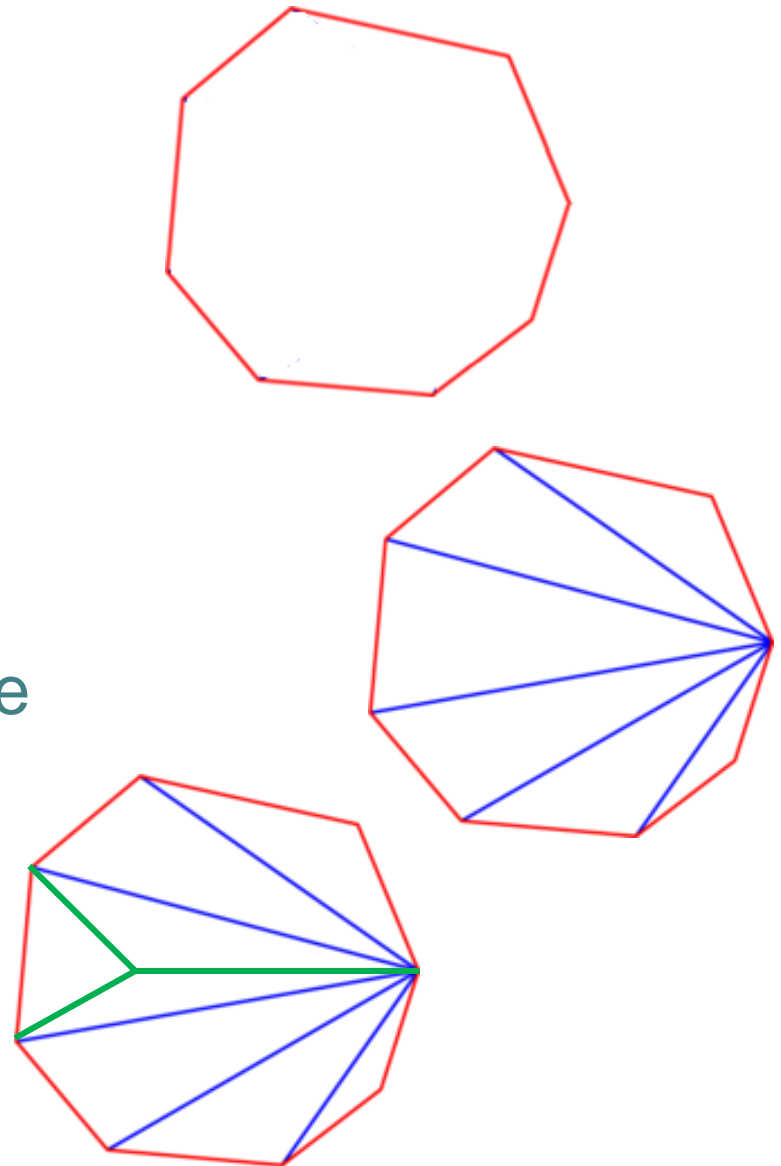
# Ok but...

Does every point set have a triangulation?

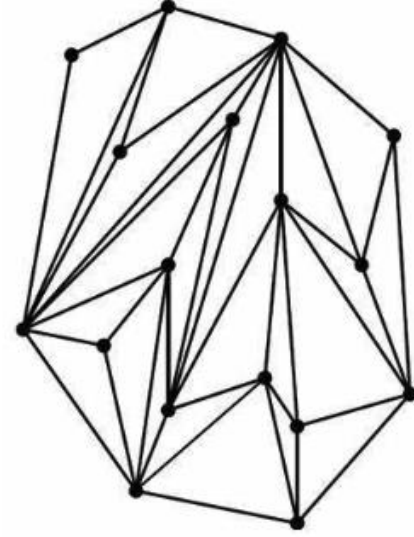
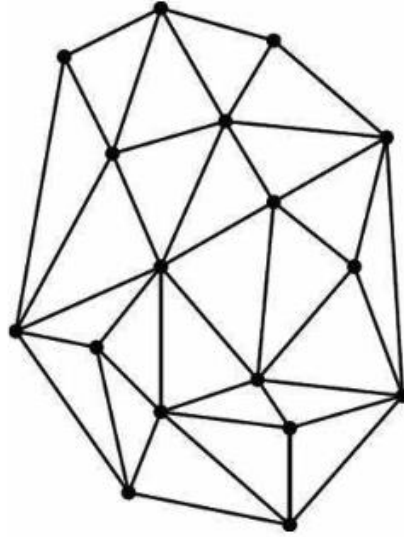
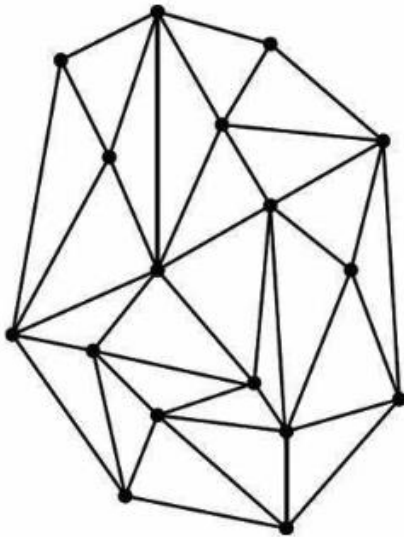


# Naive algorithm

- Take the point set and compute its convex hull
- Triangulate this polygon, e.g. fan triangulation
- Insert all inner points
  - Find the containing triangle
  - Add new edges

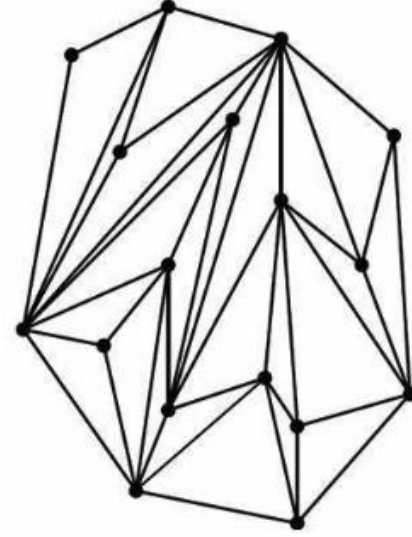
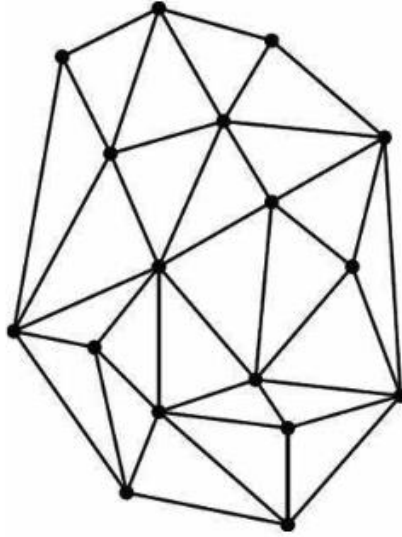
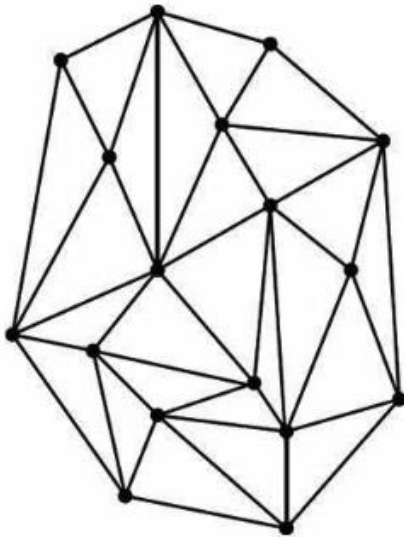


# What is a « good » triangulation?



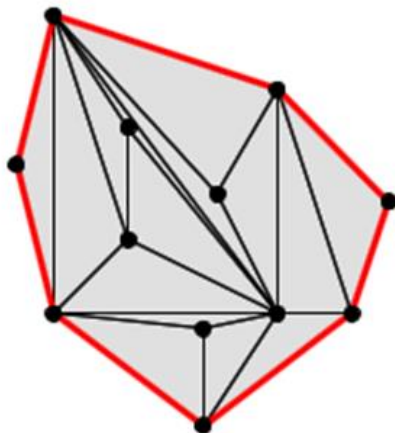
# What is a « good » triangulation?

- In most use cases (meshing, interpolation) the triangulation should provide a representation of the surface as uniform as possible
- We want to avoid slim triangles with small angles and unbalanced edges.

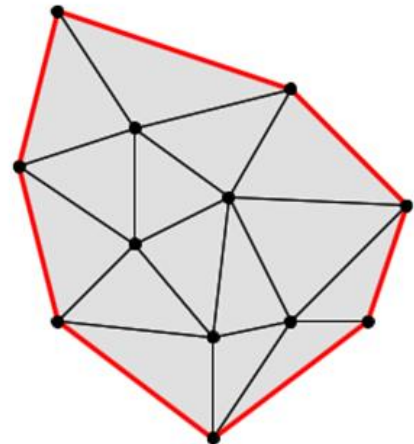


# Delaunay Triangulation

- Boris Delaunay (1934)
- The Delaunay triangulation maximizes the minimum of all the angles of the triangles, producing more equilateral triangles
- In addition to its aesthetic appeal, the Delaunay triangulation has a number of favorable properties.
- The Delaunay triangulation always exists and is unique (in the general case).
- The Delaunay triangulation corresponds to the dual graph of the Voronoi diagram.



*not Delaunay*



*Delaunay*

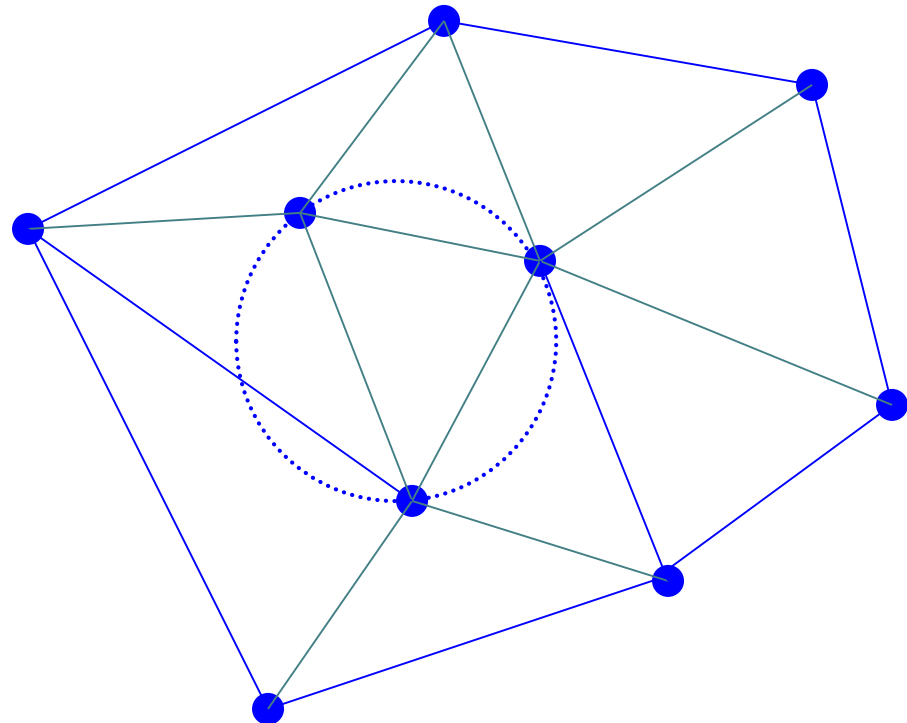
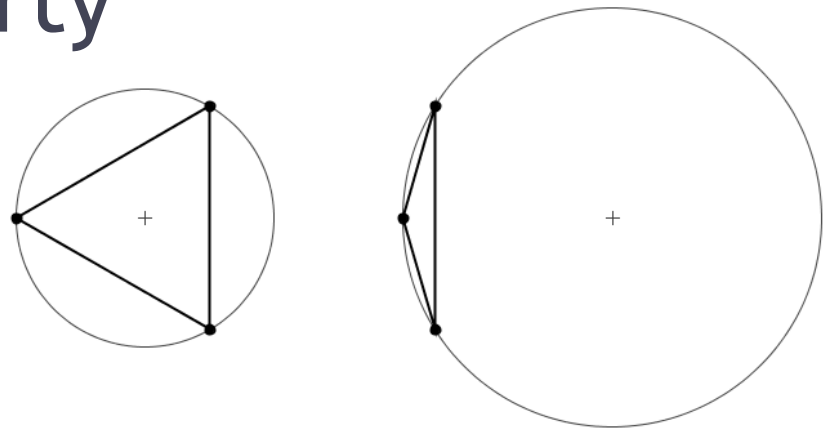
# Empty-circle property

The circumcircle of a triangle is the unique circle passing through the three vertices.

*Skinny triangles produce larger circumcircles than robust triangles of a similar size.*

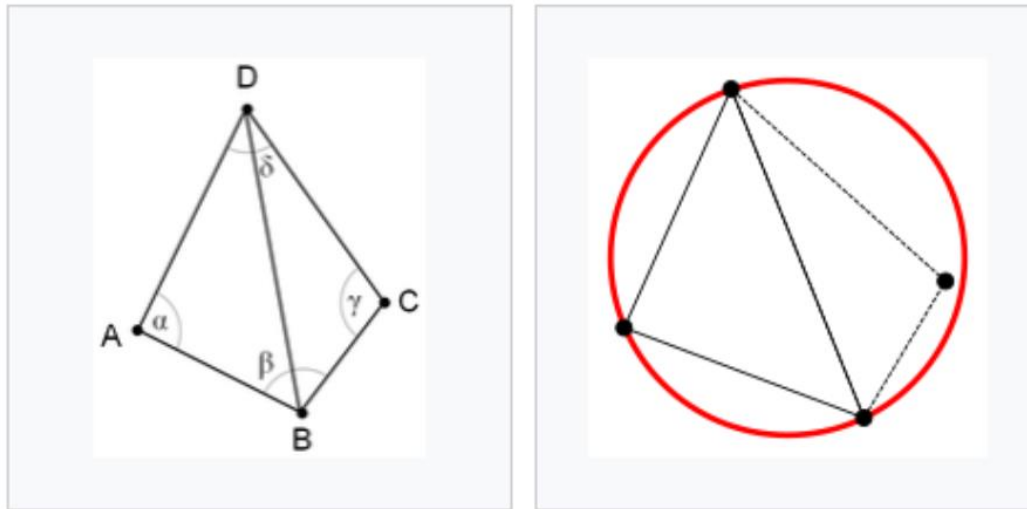
Delaunay triangulations can be equivalently characterized by the empty-circle property:

A triangulation of a set of sites is Delaunay iff the circumcircle of none of its triangles contains other sites in its interior.



# Empty-circle property

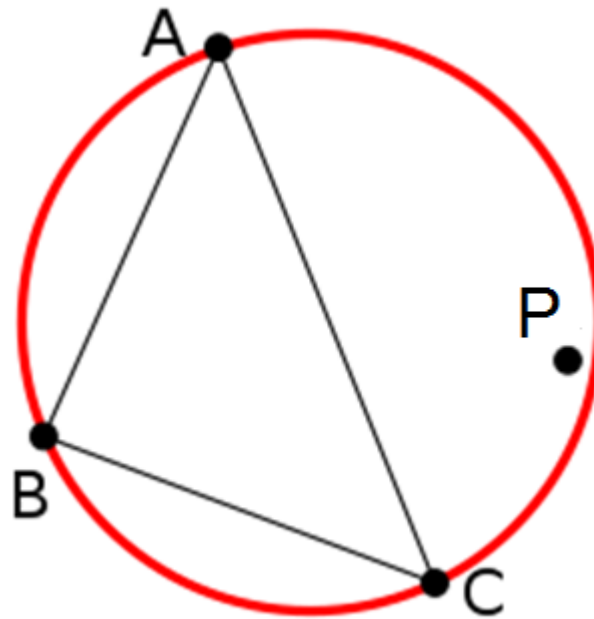
- Let us consider two triangles ABD and BCD with the common edge BD
- If the sum of the angles  $\alpha$  and  $\gamma$  is greater than  $180^\circ$ , the triangles violates the circumcircle property





# Delaunay Test

How can we detect if a point  $P$  lies in the circumcircle of three other points  $A$ ,  $B$ ,  $C$ ?

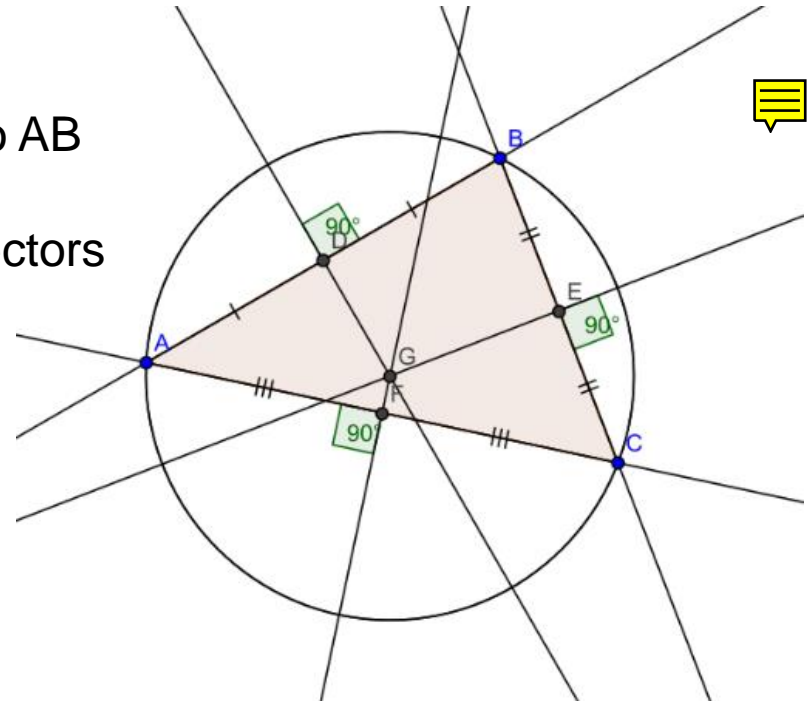


# Delaunay Test

*Compute the circumcenter  $G$  of the triangle  $ABC$ , then compare the distance  $|PG|$  to the radius of the circle.*

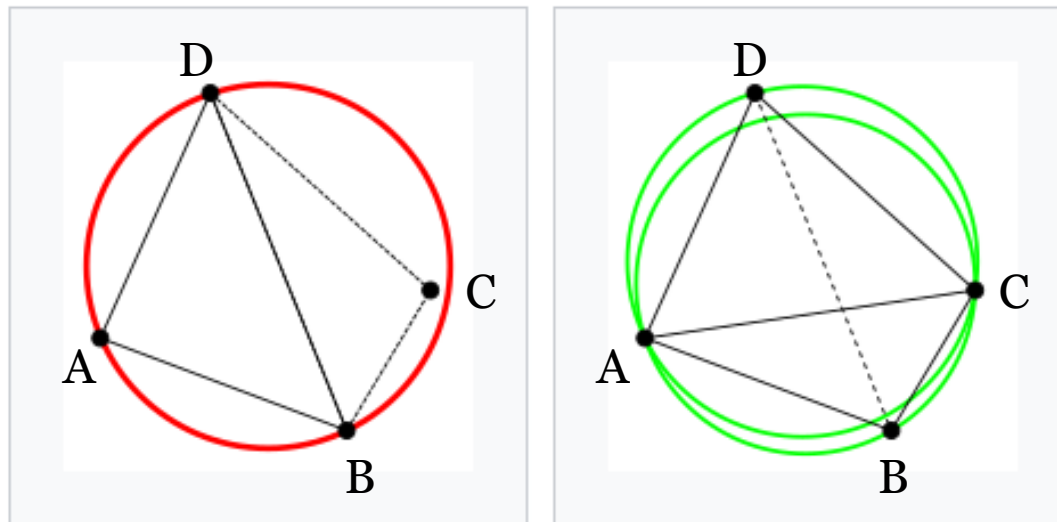
The circumcenter of a triangle is the point where all the perpendicular bisectors of the sides intersect. Note that there is no need to work with all three sides of the triangle, two sides are sufficient to find  $G$ .

- Find the midpoints  $D$  for  $AB$  and  $E$  for  $BC$
- Find two vectors  $v$  and  $w$  perpendicular to  $AB$  and  $BC$  respectively
- Considering the equations of the two bisectors  
 $b_1(t) = D + t \cdot v$   
 $b_2(t) = E + t \cdot w$   
Find  $t_1, t_2$  such that  $b_1(t_1) = b_2(t_2) = G$
- Compare  $\langle PG, PG \rangle$  to  $\langle AG, AG \rangle$



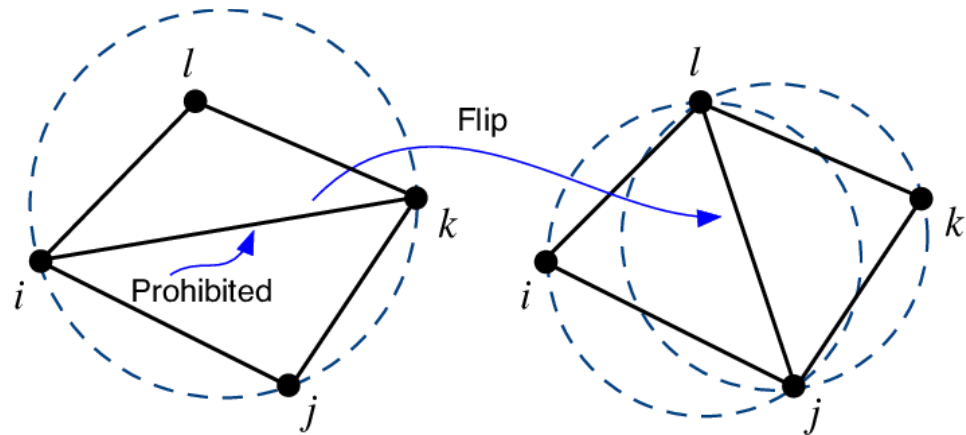
# Lawson Flip

- If two triangles do not meet the Delaunay condition (illegal edge), switching the common edge BD for the common edge AC produces two new triangles that do meet the Delaunay condition.



# Delaunay: Slow algorithm

- Apply any triangulation
- While the triangulation contains an illegal edge
  - Flip the edge




- It can be shown that the algorithm terminates after  $O(n^2)$  Lawson flips.

# Can we do better?

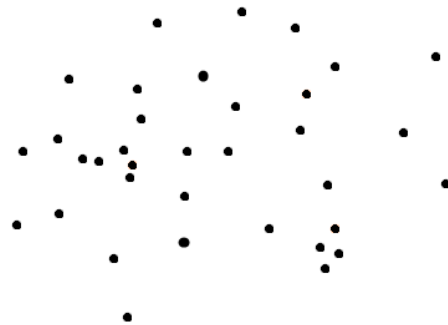
- The previous Delaunay algorithm is slow because it reinspects the **entire list** of triangles after each flip.
- We can improve on this algorithm by **incrementally** constructing a Delaunay triangulation. After each new point we immediately restore the Delaunay property, which allows us to limit our inspection to the triangles which are directly concerned with the last insertion.

# Delaunay: Incremental algorithm

- Start with a meta-triangle large enough to contain all points.
- Add the points into the triangulation one by one, maintaining the Delaunay property:  
For each point  $p$ 
  - a) Find the triangle containing  $p$ .
  - b) Create new edges to connect  $p$  to the vertices of the containing triangle.   
Now inspect the old edges of the triangle to verify that they still satisfy the empty-circle property:
    - c) Legal edges remain unchanged
    - d) Illegal edges are flipped
    - e) After each flip, two more edges  $a$  and  $b$  become candidates for inspection
    - f) The flip process continues recursively until no more candidates remain, resulting in the new Delaunay triangulation
- After having inserted all points, remove the meta-triangle.

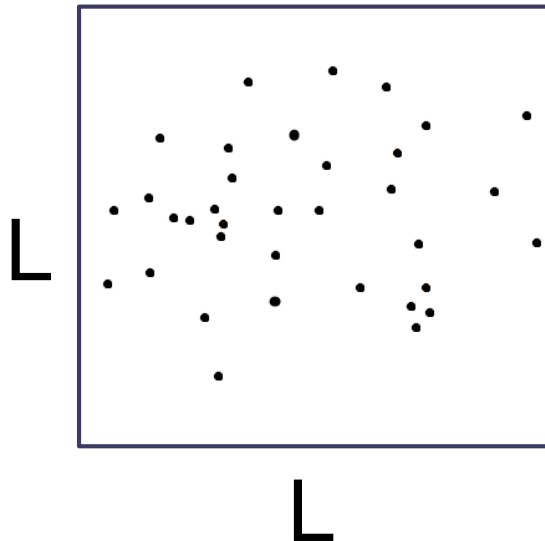
# Construct the Meta-triangle

- Take the point set  $P$



# Construct the Meta-triangle

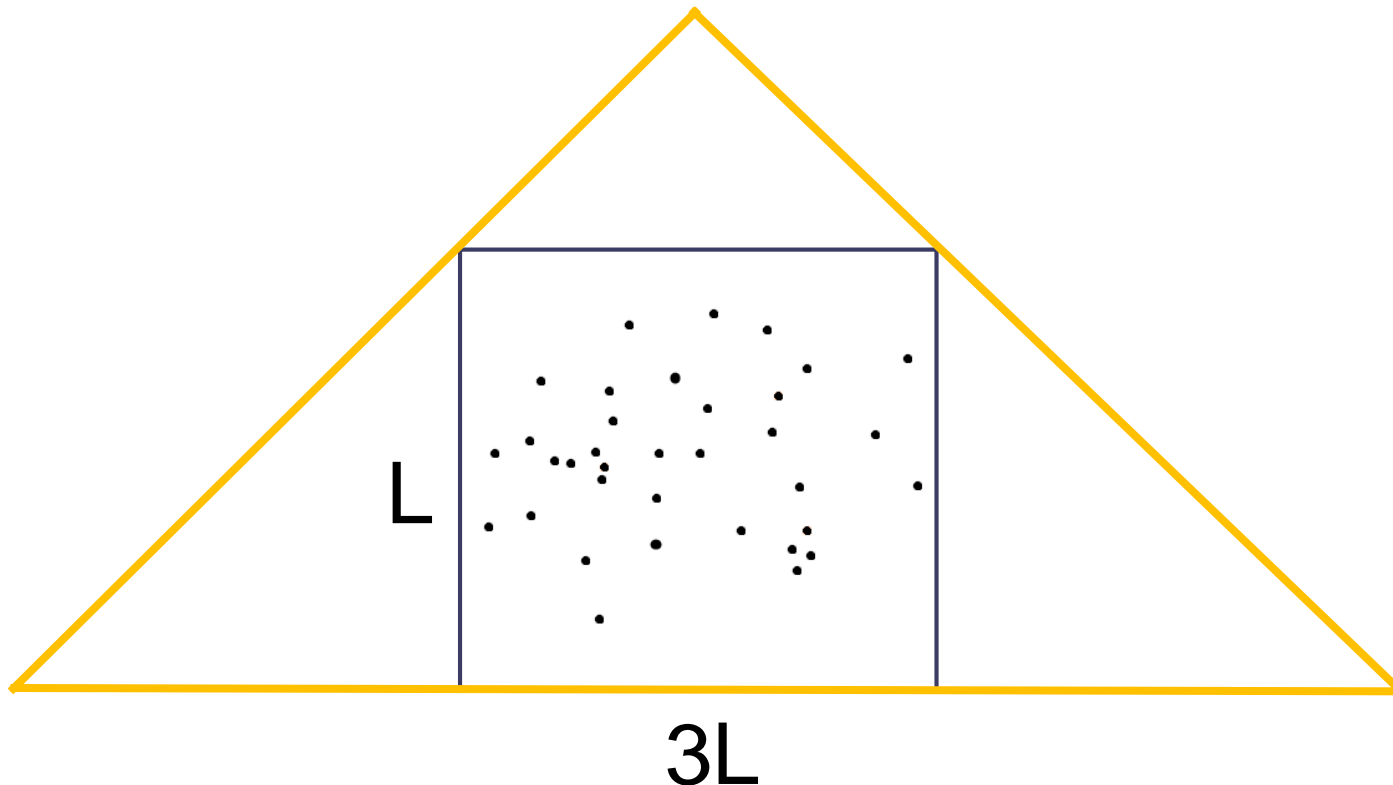
- Take the point set  $P$
- Compute a square containing  $P$





# Construct the Meta-triangle

- Take the point set  $P$
- Compute a square containing  $P$
- Compute the meta-triangle

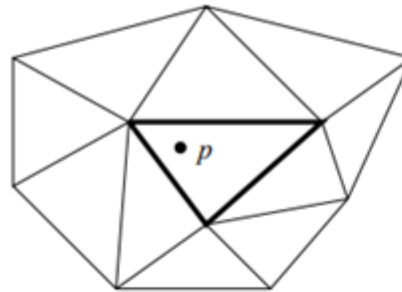


# Add Point

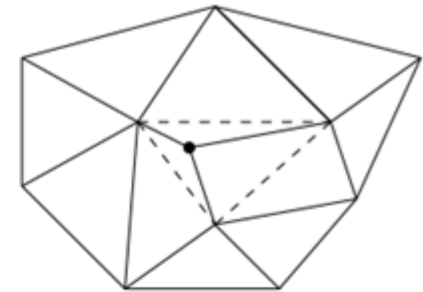
Inserting a new point into the triangle.

Dashed lines indicate edges that need to be inspected by the algorithm.

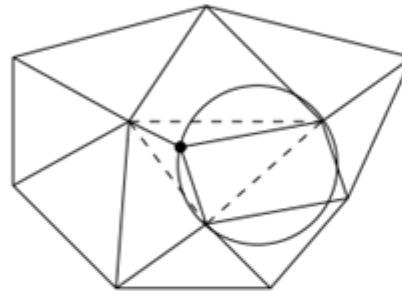
After a flip, the algorithm continues to inspect two more edges.



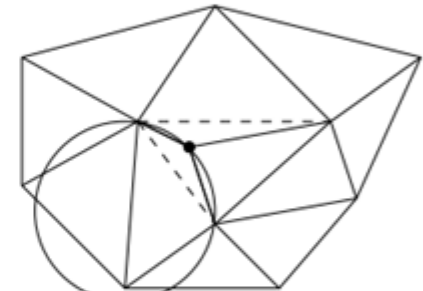
(a)



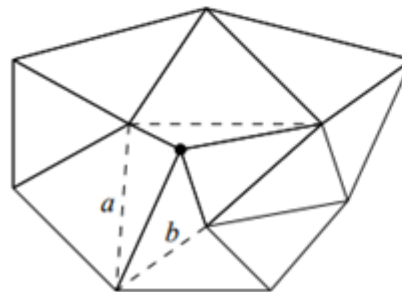
(b)



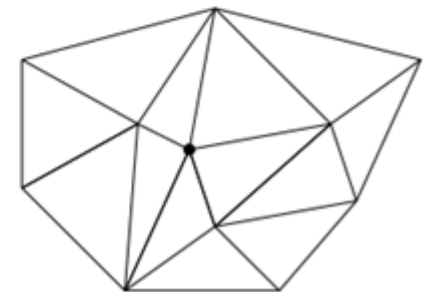
(c)



(d)



(e)



(f)

# Add Point

```
procedure addPoint(p)
```

```
  Find the triangle  $(p_1, p_2, p_3)$  containing  $p$ 
```

```
  Add edges from  $p$  to the three vertices  $p_1, p_2, p_3$ 
```

```
  // thereby splitting  $T$  into three triangles
```

```
  legalizeEdge( $p, p_1p_2$ )
```

```
  legalizeEdge( $p, p_2p_3$ )
```

```
  legalizeEdge( $p, p_1p_3$ )
```

```
end procedure
```

```
procedure legalizeEdge( $pr, pipj$ )
```

```
//  $pr$  the point being inserted,  $pipj$  the edge that may be flipped
```

```
  if ( $pipj$  is illegal) then
```

```
    let  $(p_i, p_j, p_k)$  be the triangle adjacent to  $(pr, p_i, p_j)$  along the edge  $pipj$ 
```

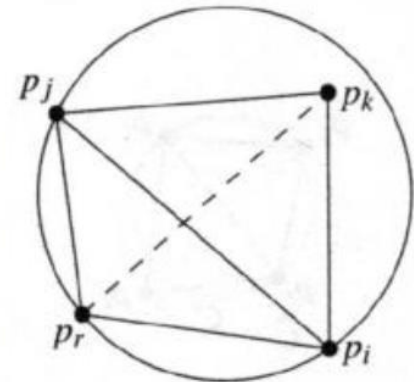
```
    flip, i.e. replace  $pipj$  with  $prp_k$ 
```

```
    legalizeEdge( $pr, pip_k$ )
```

```
    legalizeEdge( $pr, p_kp_j$ )
```

```
  endif
```

```
end procedure
```



# Find the triangle containing $p$

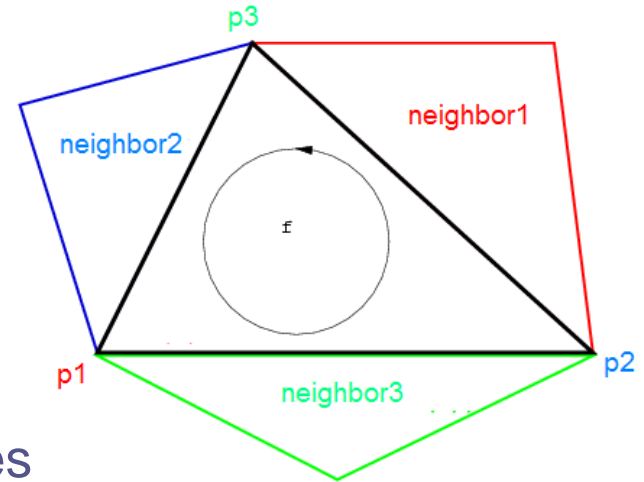
Locating the containing triangle can be done in an optimal  $O(\log n)$  time, but this requires maintaining additional data structures.

We store the history of the splits and flips performed.

- Each triangle stores a pointer to the triangles that replaced it.
- To find the triangle that contains  $p$ , we start at the root triangle and follow the pointer that points to a triangle that contains  $p$ , until we find a triangle that has not yet been replaced.

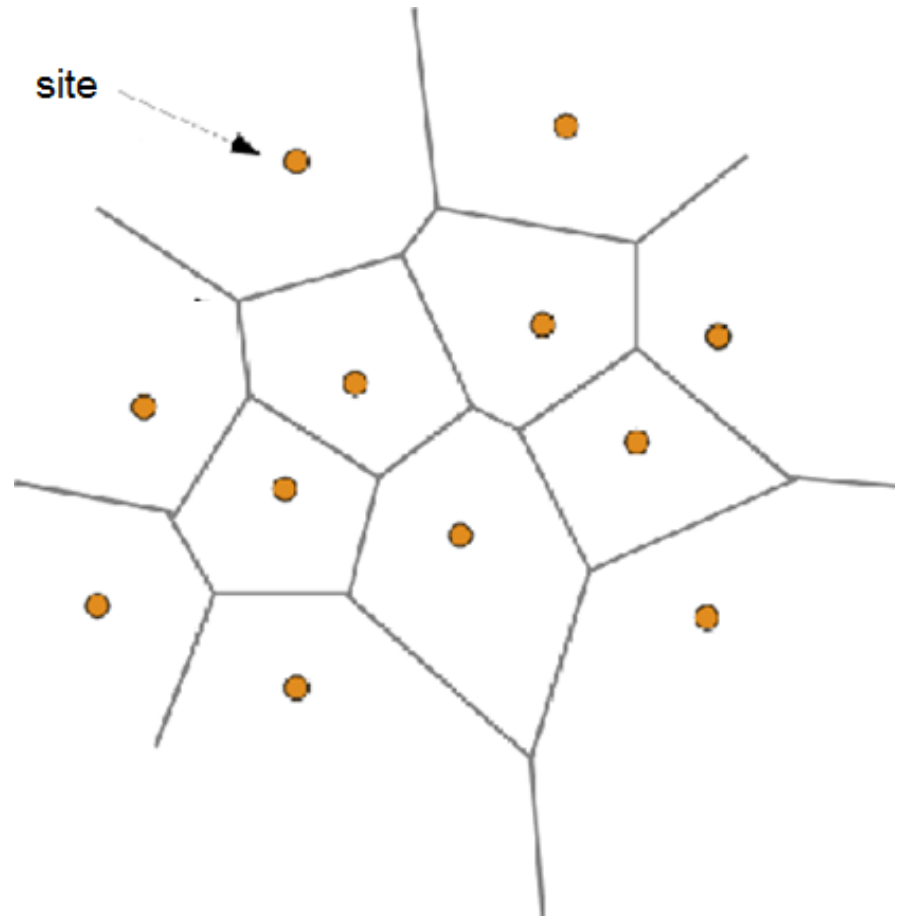
# Exercise

- Implement a data structure for
  - Triangulation = a list of Triangles
  - Triangle
    - 3 Points
    - 3 references to the neighboring triangles
- Implement the following methods
  - Find the triangle which contains a point  $p$
  - Insert a point into the triangulation
  - Delaunay test
  - Lawson flip
- Implement
  - the naive triangulation
  - the slow Delaunay  $O(n^2)$
  - the incremental Delaunay  $O(n \log n)$Show the intermediate steps of each algorithm.



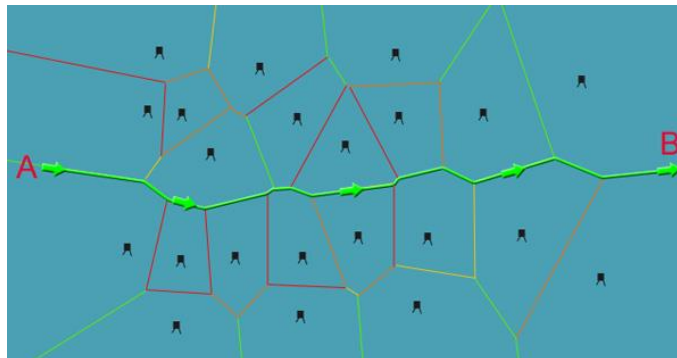
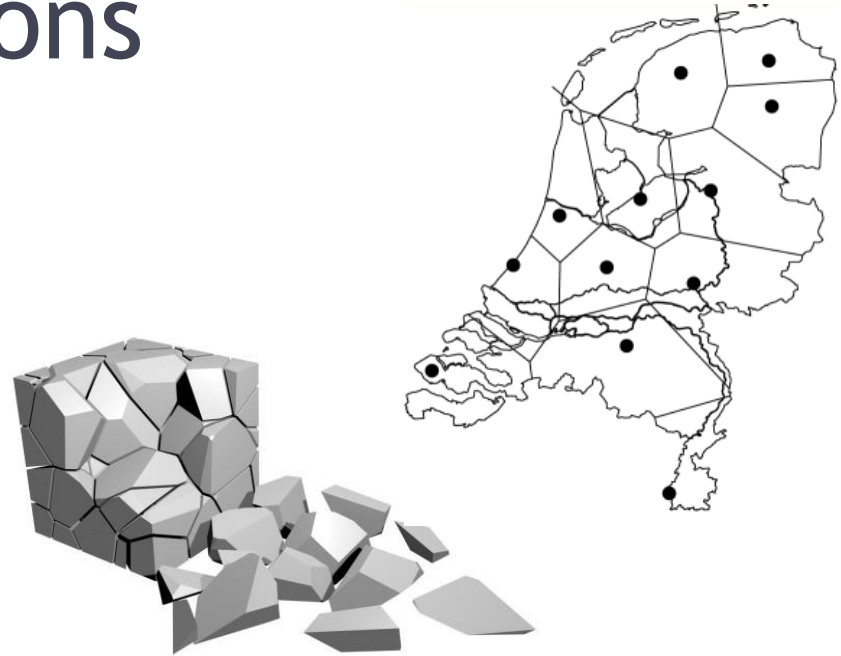
# Voronoi diagram

- For a set of points (sites), the Voronoi diagram is the subdivision of the plane into regions where
  - Inside the region exactly one site is closest among all sites
  - On the edges 2 sites are closest
  - On the vertices at least 3 sites are closest
- Voronoi vertices typically have degree 3, but it can be larger
- Voronoi regions are convex and can be bounded or unbounded



# Voronoi: applications

- Post office problem: given a set of  $n$  sites (post offices) assign every point to its nearest site, seek optimal location for a new site
- Computer graphics: 3D shattering geometry patterns
- Autonomous robot navigation: the edges of the Voronoi graph are the routes furthest from obstacles.
- And much more

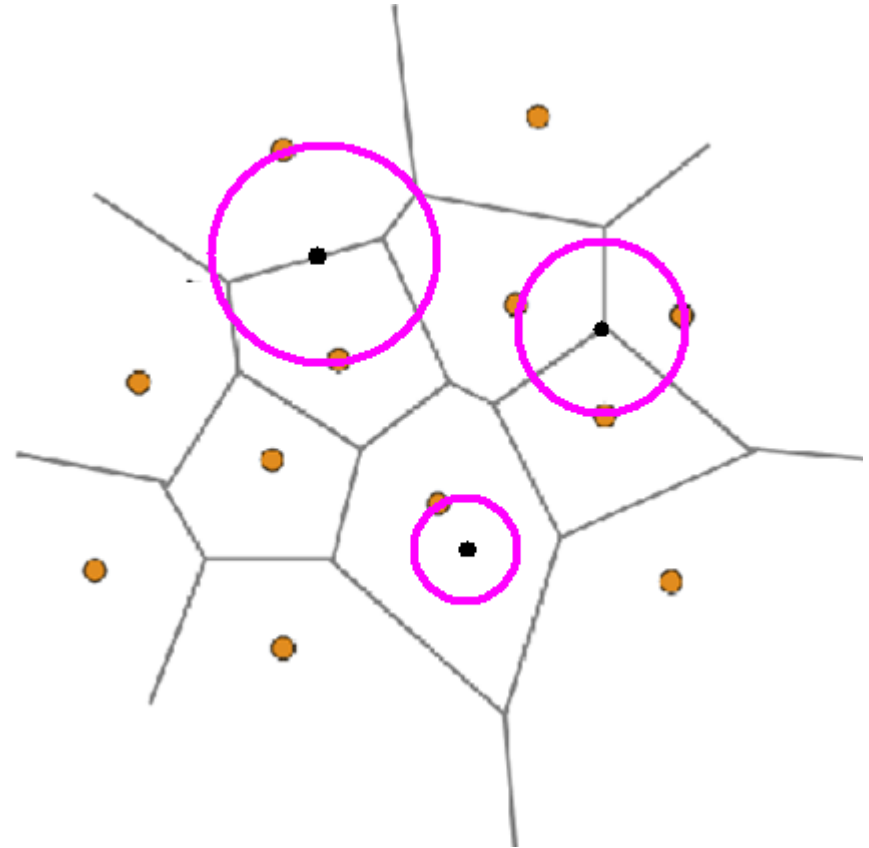


# Delaunay circles

Given a set of sites  $S$  and its Voronoi diagram, we define for any point  $a$  on the plane the Delaunay circle  $C_a$  to be the circle with center  $a$  and radius  $\min_{s \in S} \text{dist}(s, a)$

$C_a$  has the following properties:

- If  $a$  lies in a Voronoi region:  
 $C_a$  contains exactly 1 site
- If  $a$  lies on a voronoi edge:  
 $C_a$  contains 2 sites
- If  $a$  is a voronoi vertex:  
 $C_a$  contains at least 3 sites



Delaunay circles allow building a bridge between Voronoi diagrams and Delaunay triangulations.



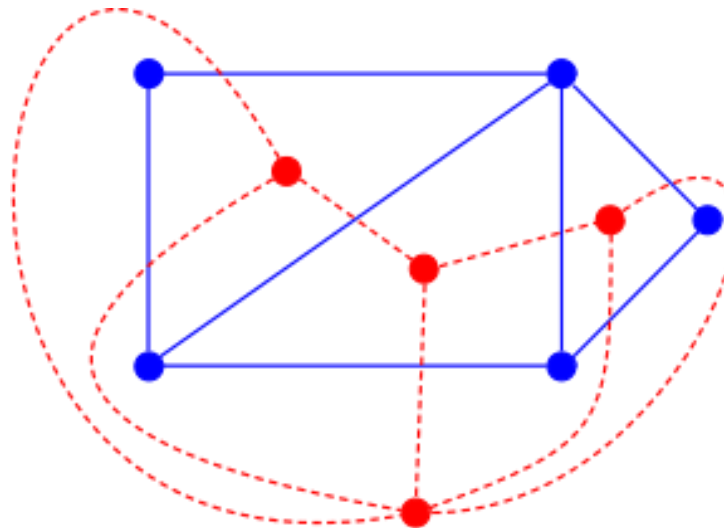
# Voronoi algorithms

- Direct
  - Green et Sibson (1978):  $O(n^2)$
  - Fortune's Algorithm (1987):  $O(n \cdot \log n)$
  - Jump Flooding (2006): independent from  $n$ ,  $O(p^2 \cdot \log p)$  for an image of  $p \times p$  pixels
- Indirect
  - Start with a Delaunay triangulation and compute its “dual graph”

# Dual graph

Let  $G=(V,E)$  be a plane graph. The dual graph  $G^*$  has

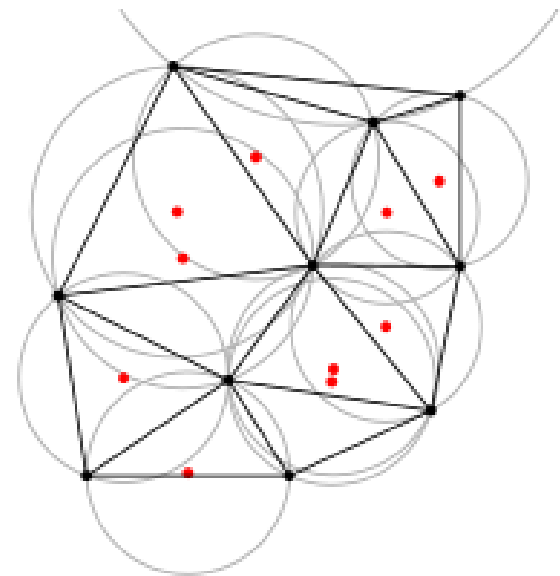
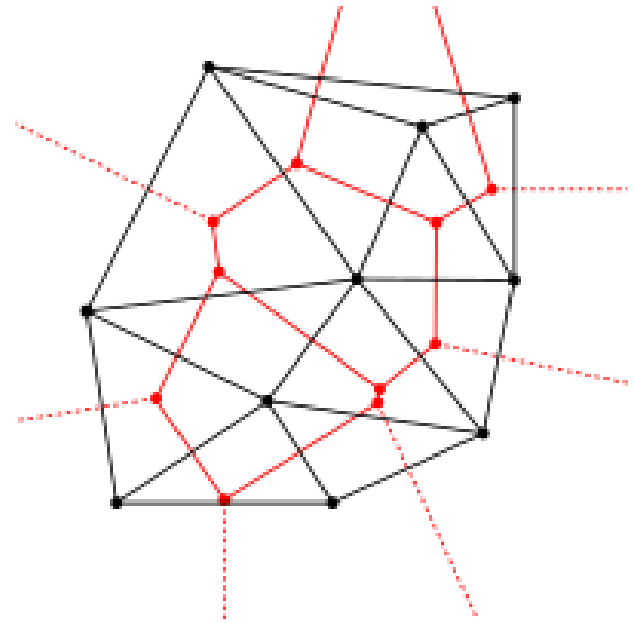
- a vertex for every face of  $G$ ,
- an edge for every edge of  $G$ , between the two faces incident to the original edge



The red graph is the dual graph of the blue graph, and vice versa.

# Voronoi vs. Delaunay

- The Delaunay triangulation of a discrete point set  $P$  corresponds to **the dual graph** of the Voronoi diagram for  $P$ .
- The **circumcenter of a Delaunay triangle** has maximum distance to all three sites, therefore it is **a vertex of the Voronoi diagram**.
- If two triangles share an edge in the Delaunay triangulation, their circumcenters are connected with a Voronoi edge.



# Exercise

- Implement a data structure for
  - Voronoi

Compute the Voronoi diagram for a point set  $P$  (indirect approach):

- Apply a Delaunay triangulation for  $P$
- Compute the **Voronoi vertices**: they are the circumcenters of the Delaunay triangles.
- Compute the **Voronoi edges**: for every Delaunay edge with two neighboring triangles, connect their circumcenters. Find a special treatment for Delaunay edges on the convex hull, having only one triangle.