CENG 789 – Digital Geometry Processing

03- Point Sets

Prof. Dr. Yusuf Sahillioğlu

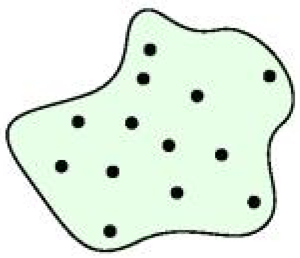
Computer Eng. Dept, MIDDLE EAST TECHNICAL UNIVERSITY, Turkey

Point Sets

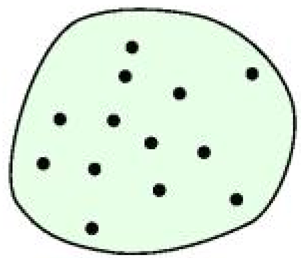
- ✓ Focusing on point sets now.
  - ✓ Convex hulls, triangulations (e.g., Delaunay), Voronoi diagrams.
- ✓ Unlike polygonal vertices (prev lectures), point sets are unordered.

Convexity

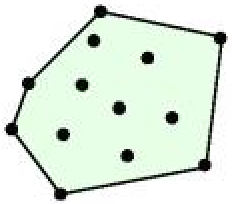
- ✓ Extend convex polygon idea to convex region.
- ✓ A region is convex if any 2 points a and b of the region are visible to each other, i.e., line segment ab lies within the region.
- ✓ S = point set.



A nonconvex region enclosing S



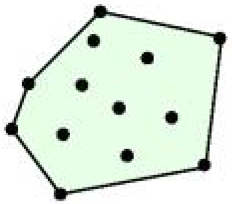
A convex region enclosing S



Convex hull of S

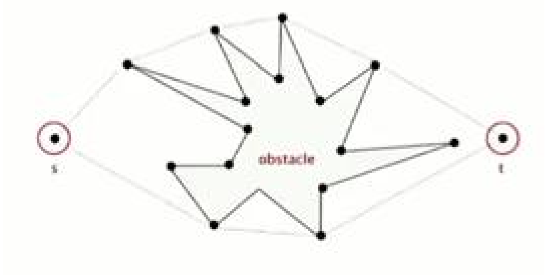
Convex Hull

- ✓ Intuition: an elastic rubber stretched around all the nails/points.
- ✓ Motivation: useful for collision detection, robotics, statistics.
- ✓ Formal: smallest convex region containing the point set S.
- ✓ Formal: intersection of all convex regions containing S.
  - ✓ Cool but impractical definition to design a construction algo.
  - ✓ Practical definition based on visibility is better for algorithmic design.

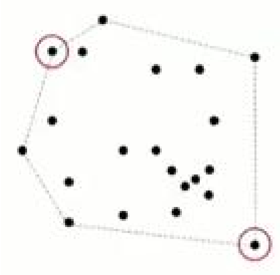


Convex Hull

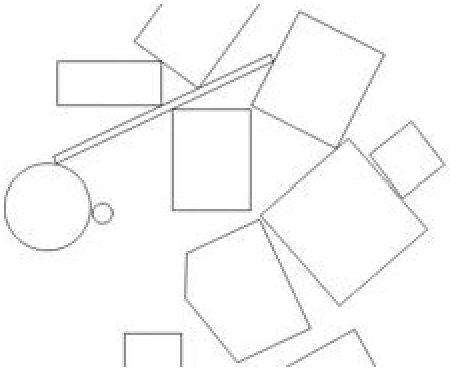
- ✓ Motivation: useful for collision detection, robotics, statistics.
- ✓ Find the shortest path from s to t that avoids the polygon obstacle.



- ✓ Find a pair of points w/ the largest Euclidean distance in between.



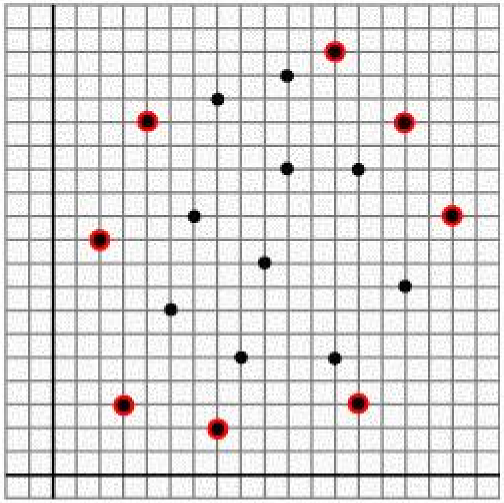
- ✓ More accurate collision detection.



Convex Hull Construction

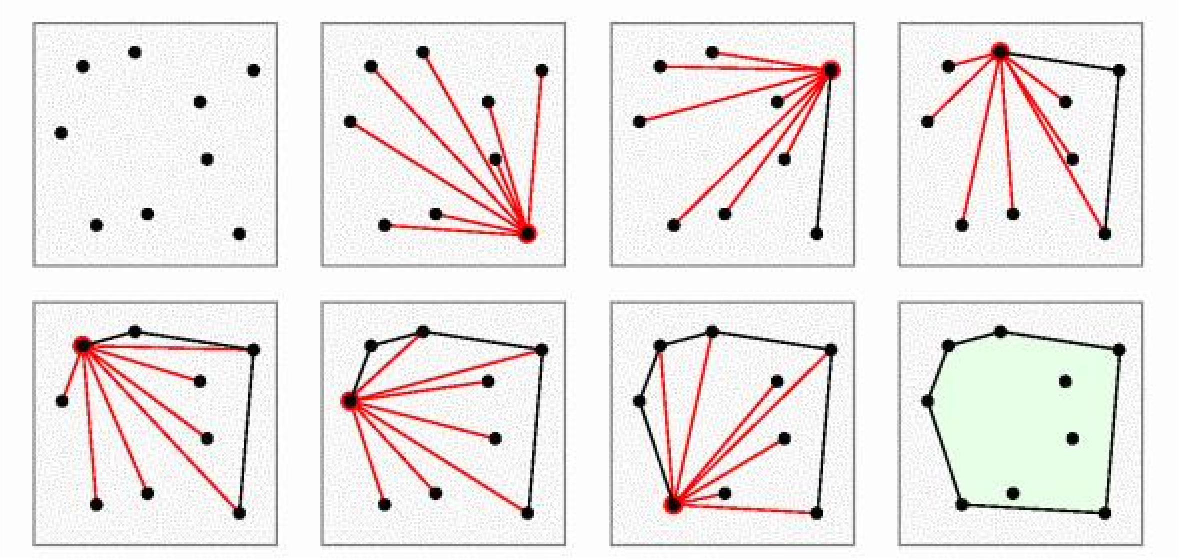
- ✓ Task: Identify hull vertices from the whole set of vertices.
- ✓ Easy for your eyes if given a paper w/ marked points.
- ✓ What if points are listed as coordinates (common case)?
  - ✓ Take the extreme points (min/max x/y) naturally.
  - ✓ What about the other hull points?
  - ✓ Identifying hull vertices w/o graphing is hard.
  - ✓ Even graphing/visualization fails for 1000+ points.

(2, 10)	(10, 17)
(3, 3)	(10, 13)
(4, 15)	(12, 5)
(5, 7)	(12, 18)
(6, 11)	(13, 3)
(7, 2)	(13, 13)
(7, 16)	(15, 8)
(8, 5)	(15, 15)
(9, 9)	(17, 11)



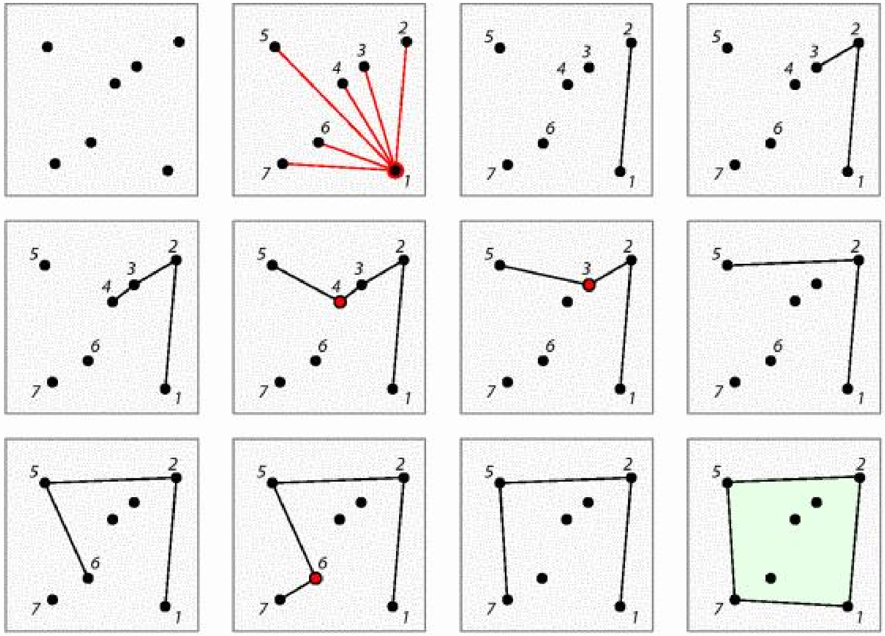
Convex Hull Construction

- ✓ Gift-wrapping algo to identify hull vertices in O(nh) time.
  - ✓ n = # of points in point set S; h = # of hull points.
  - ✓ h could be as large as n, leading to O(n^2) ☹. Such a point set? Best set?
  - ✓ Extension to 3D exists.
- ✓ Start w/ the bottommost pnt. Draw a line segmnt to all others. Choose the next turn-pnt on our wrapping of S to be the pnt that makes the largest angle w/ the last constructed hull edge (initially with the horizontal line).



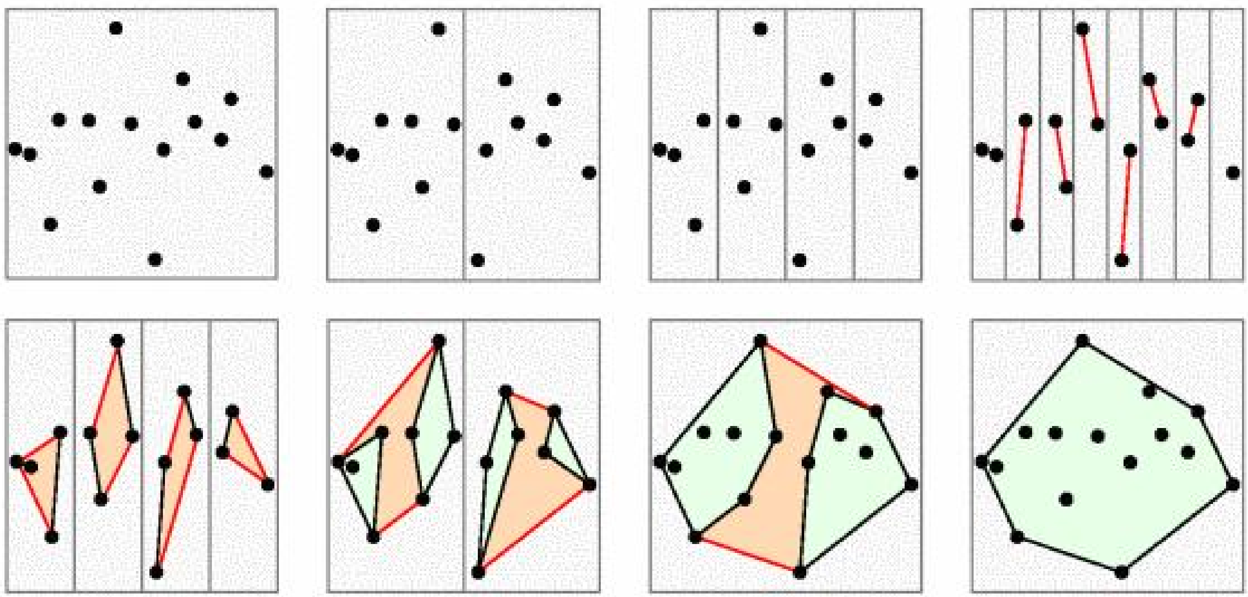
Convex Hull Construction

- ✓ Graham scan algo to identify hull vertices in O(nlogn) time.
  - ✓ No extension to 3D.
- ✓ Start w/ the bottommost pnt. Draw a line segmnt to all others. **Sort** the remaining points by the angles they make w/ the horizontal line. Construct the hull following this ordering (in triplets), adding pnts for left hull turns and deleting for right turns.



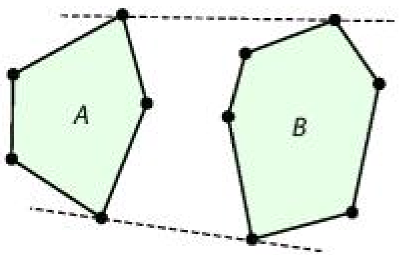
Convex Hull Construction

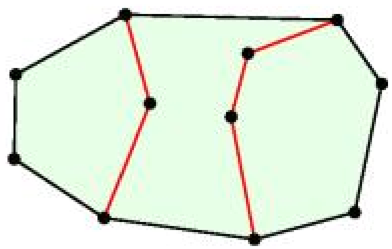
- ✓ Divide-and-conquer algo to identify hull vertices in O(nlogn) time.
  - ✓ Easy extension to 3D.
- ✓ Sort points by x-coordinate. Divide the points into 2 (nearly) equal groups. Compute the convex hull of each group recursively. Merge 2 groups with upper and lower supporting tangent lines.
- ✓ Recurse until 3 or less points in each group, whose conv hulls trivial.



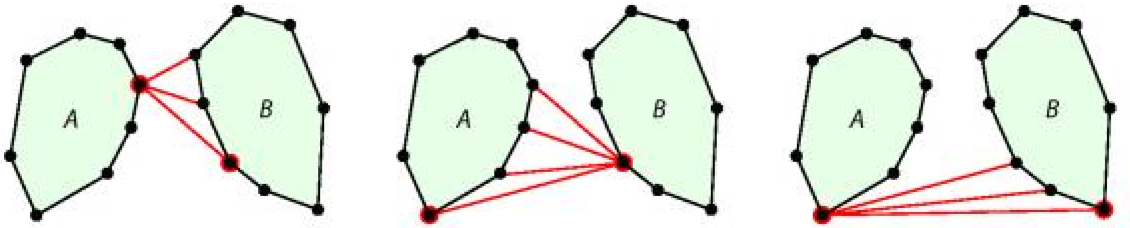
Convex Hull Construction

- ✓ Divide-and-conquer algo to identify hull vertices in O(nlogn) time.
  - ✓ Easy extension to 3D.
- ✓ Merging is tricky. Not just highest/lowest points (circle vs. tall rectangle).
- ✓ O(n^2):



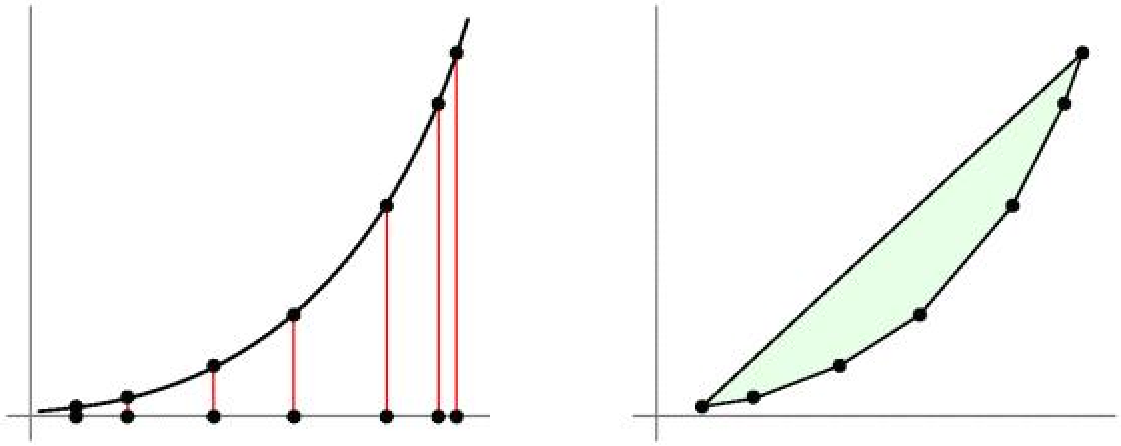


- ✓ O(n):



Convex Hull ~ Sorting

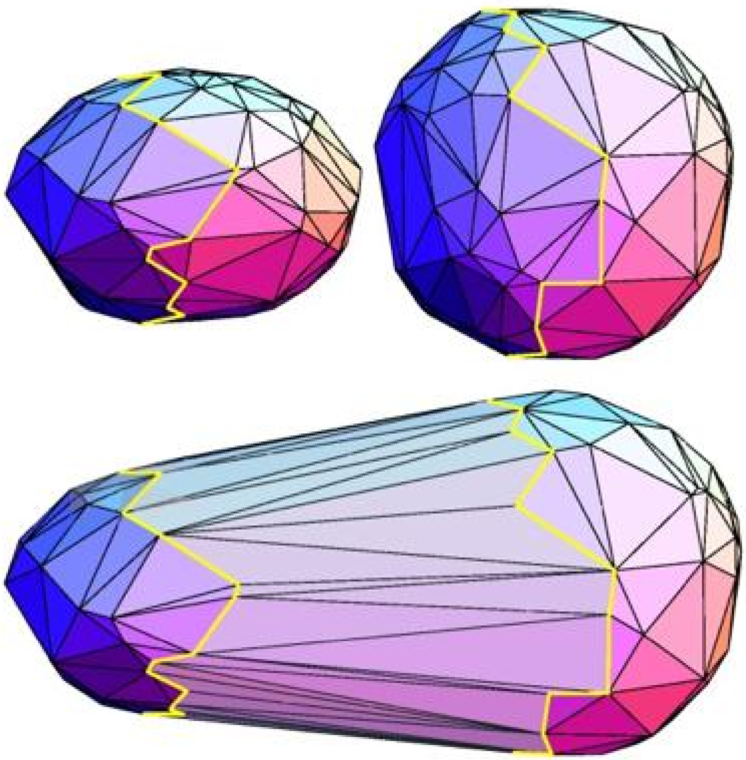
- ✓ Lower bound for any convex hull construction algo is O(nlogn) ‘cos Sorting problem reduces to Convex Hull problem: Sorting $\leq_P$ CnvxHull.
- ✓ Lower bound to sort n numbers is nlogn (decision-tree model).
- ✓ Given unsorted {x1, x2, .., xn}, construct the set of points in the plane (xi, xi^2) as shown:

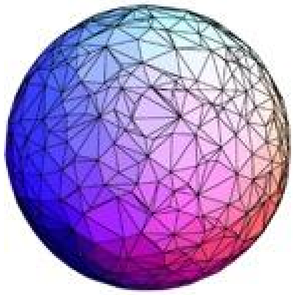


- ✓ Every point must be on the hull ‘cos parabolas are convex.
- ✓ Convex hull of this parabola = identifid ordered hull vertices = sorting.

3D Convex Hull Construction

- ✓ Divide-and-conquer algo to identify hull vertices in O(nlogn) time.
- ✓ Sort points by x-coordinate. Divide the points into 2 (nearly) equal groups. Compute the convex hull of each group recursively. Merge 2 groups with upper and lower supporting tangent **planes**.



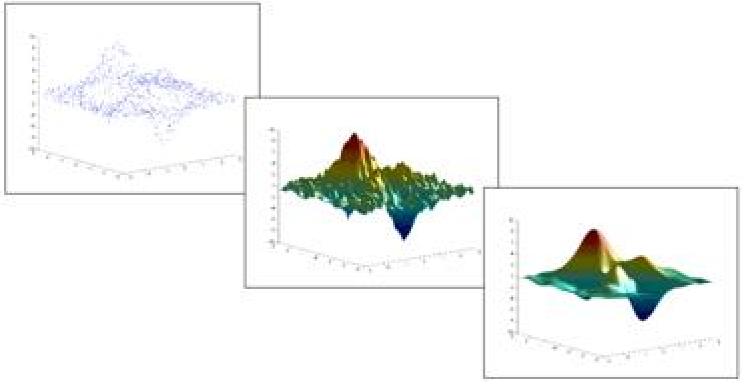


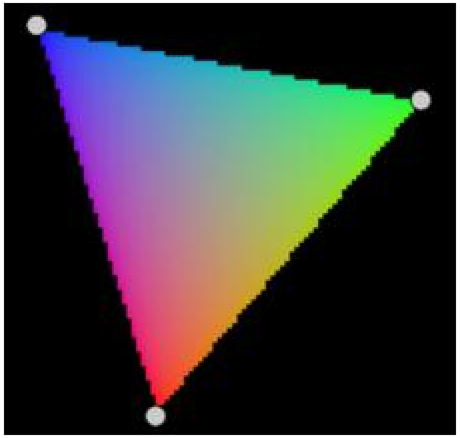
Triangulations

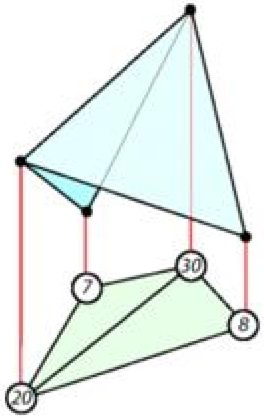
- Already did triangulation of a polygon.
- Let's do the triangulation of a structureless point set.

- Convex hull is about finding the *boundary* of a point set.
- Triangulation is about the *interior* of a point set.

- Motivation: put a structure to scattered data so that you can make interpolations at any point in space, e.g., interpolate color or height values on scattered data points.







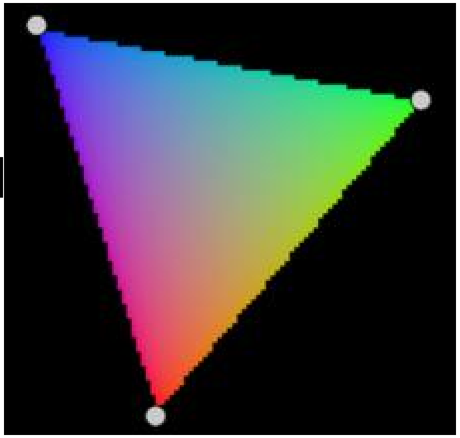
Triangulations

- ✓ Already did triangulation of a polygon.
- ✓ Let's do the triangulation of a structureless point set.

- ✓ Convex hull is about finding the *boundary* of a point set.
- ✓ Triangulation is about the *interior* of a point set.

- ✓ Motivation: put a structure to scattered data so that you can make interpolations at any point in space, e.g., interpolate color or height values on scattered data points.

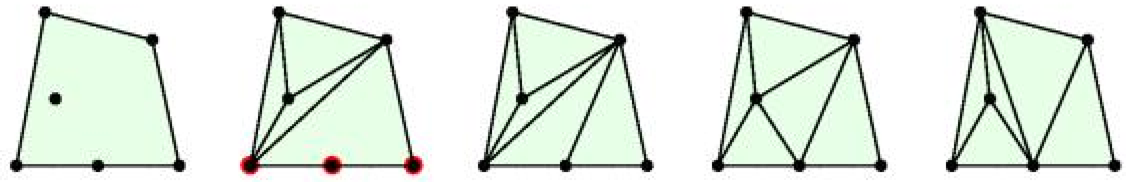
- ✓ Interpolation schemes are discussed in Deformation lecture slides 20-31.
- ✓ Focusing on triangulations here.



Triangulations

- ✓ With polygons we have boundary edges and internal diagonals.
- ✓ With point sets we just have edges to describe any line segment that includes 2 points of set as endpoints.

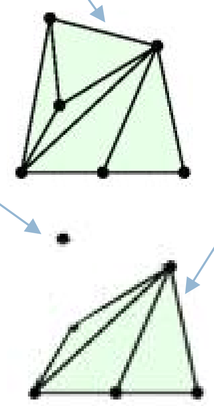
- ✓ Triangulation: of a point set S is subdivision of the plane by a maximal set of edges whose vertex set is S.



Convex hull Subdivision Triangulationsssssssss

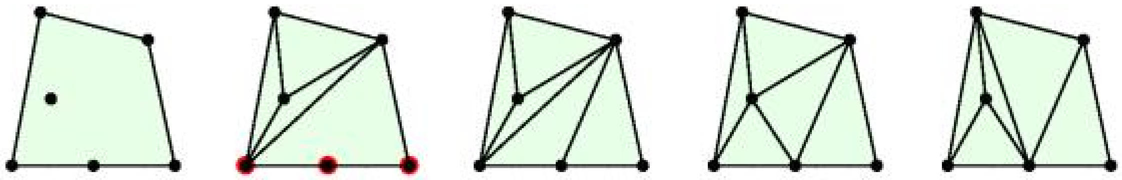
Triangulations

- ✓ Prove that the edges of the convex hull will be in all triangulations.
- ✓ Proof: suppose not; an edge of the hull not in some triangulation. Then a point of S must be outside the convex hull, a contradiction.



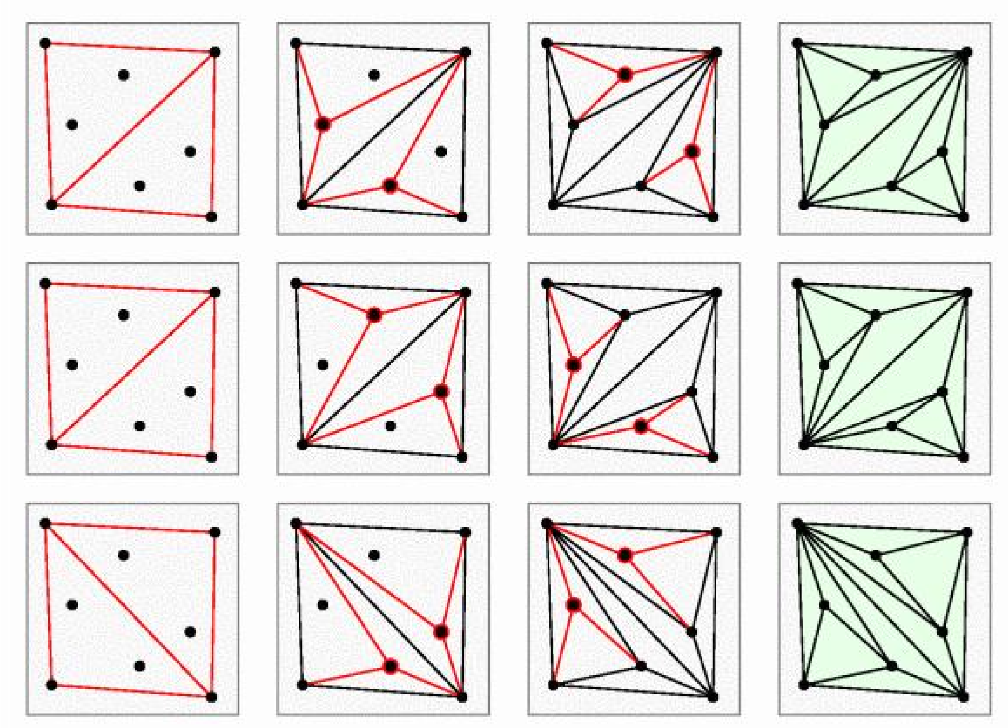
Triangulations

- ✓ Prove that all region of subdivision inside conv hull must be triangles.
- ✓ Proof: suppose not; some region has a region w/ 4+ edges. That region contradicts `maximal set of edges' claim as it can be subdivided further.



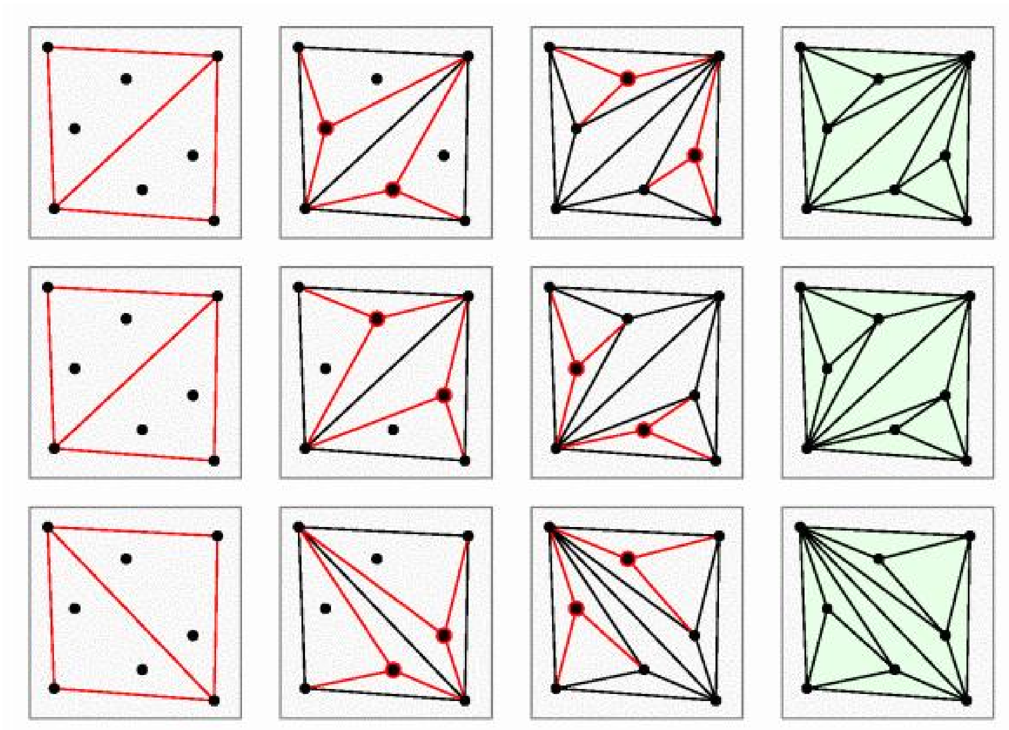
Triangulation Construction

- ✓ A simple algorithm called triangle-splitting to triangulate pnt set S.
- ✓ Find convex hull of S and triangulate it as a polygon (previous lecture). Choose an interior point and draw edges to the 3 vertices of the triangle that contains it. Continue 'till all interior points are done.



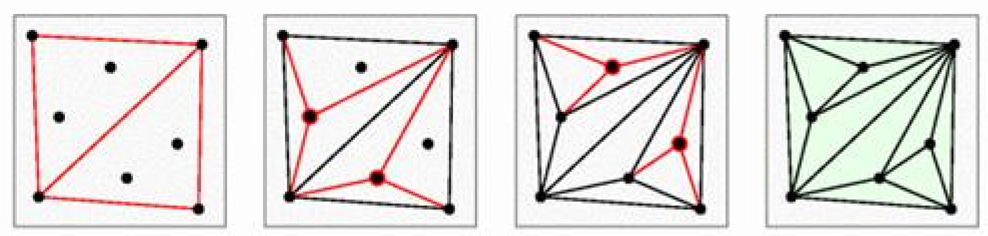
Triangulation Construction

- ✓ A simple algorithm called triangle-splitting to triangulate pnt set S.
- ✓ Notice initial hull triangulation & processing order affect the output.



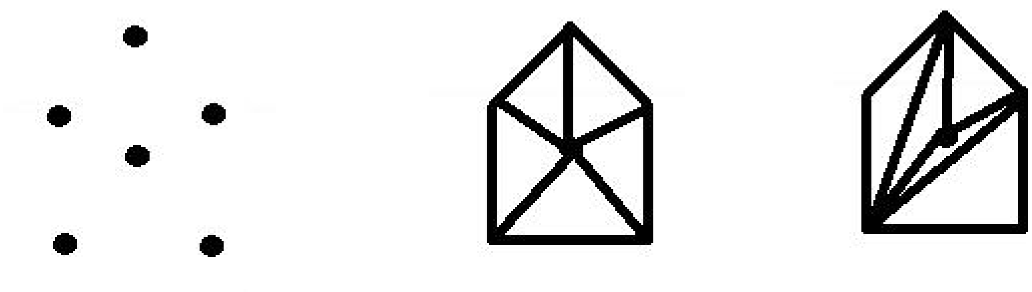
Triangulation Construction

- ✓ A simple algorithm called triangle-splitting to triangulate pnt set S.
- ✓ Any triangulation of S using this algo has 2k+h-2 triangles, where k = # of interior points, h = # of hull points.
- ✓ Proof: triangulation of the convex hull, a polygon, has h-2 triangles. An interior point replaces 1 triangle by 3, hence increasing the triangle count by 2. k interior pnts increase by 2k; hence h-2+2k.



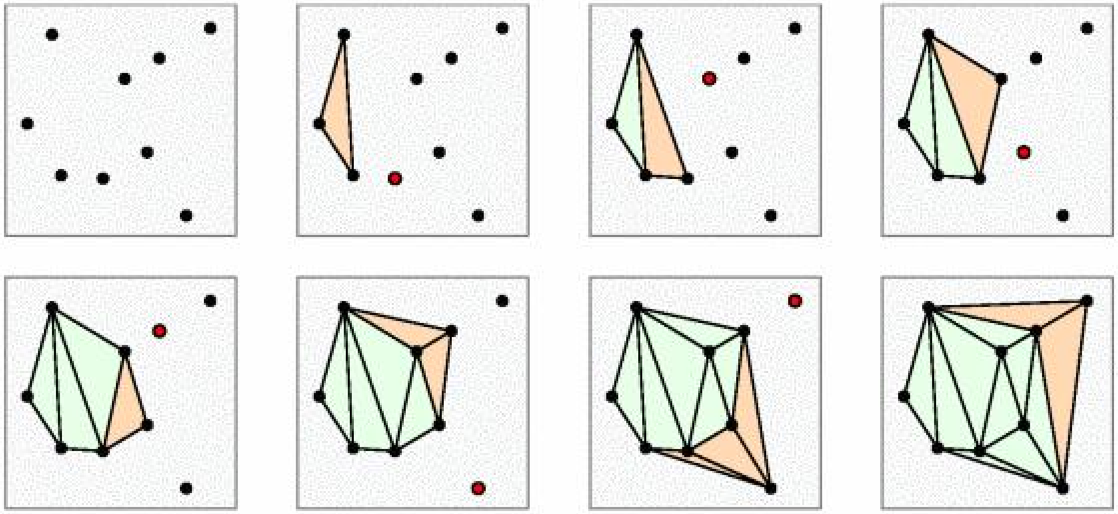
Triangulation Construction

- ✓ A simple algorithm called triangle-splitting to triangulate pnt set S.
- ✓ Disprove that this algo produces all possible triangulations of S.
- ✓ Disproof: a counterexample that can’t be produced by this algo.



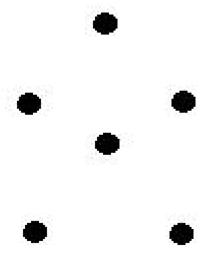
Triangulation Construction

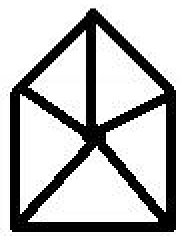
- ✓ Another algorithm called incremental to triangulate pnt set S.
- ✓ Sort the points of S by x-cords. First 3 points make a triangle. Consider the next point p in order and connect it w/ previously considered points which are visible to p. Continue incrementally.

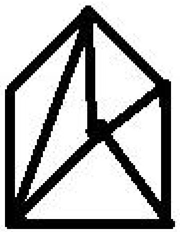


Triangulation Construction

- ✓ Another algorithm called incremental to triangulate pnt set S.
- ✓ Disprove that this algo produces all possible triangulations of S.
- ✓ Disproof: a counterexample that can't be produced by this algo.







Triangulation Construction

- ✓ Any triangulation of S using any algo has 2k+h-2 triangles, where k = # of interior points, h = # of hull points, n = k+h = # of points.
- ✓ Proof: using Euler's V – E + F = 2 formula. Triangulation subdivides the plane into t+1 faces, t triangles inside the hull, 1 big face outside the hull. Each triangle has 3 edges and outside face has h edges; hence 3t+h edges. Each edge touches 2 faces → 3t+h double counts edges → there're indeed E = (3t+h)/2 edges. Apply Euler:

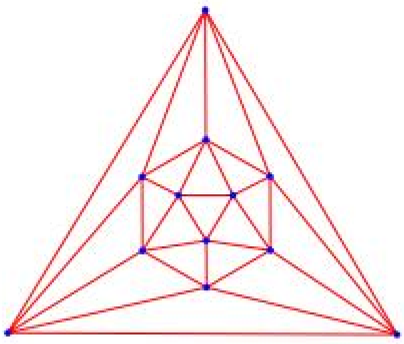
  n – (3t+h)/2 + (t+1) = 2 → t = 2n – h – 2 → 2k + h – 2

  bonus: use this t value and Euler to show # edges = 3k+2h-3

  bonus: tetrahedralizations of same point set have different tets.

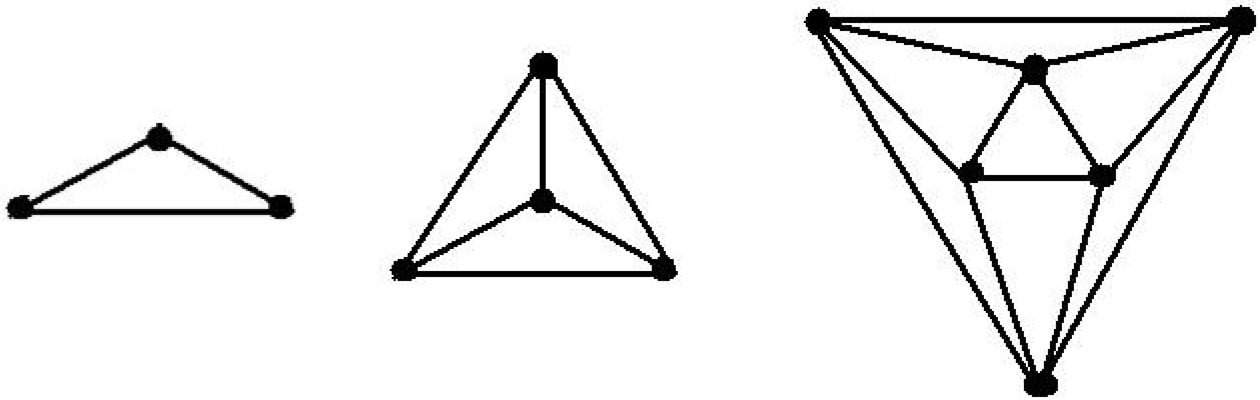
Triangulations

- ✓ What is interesting about this triangulation?



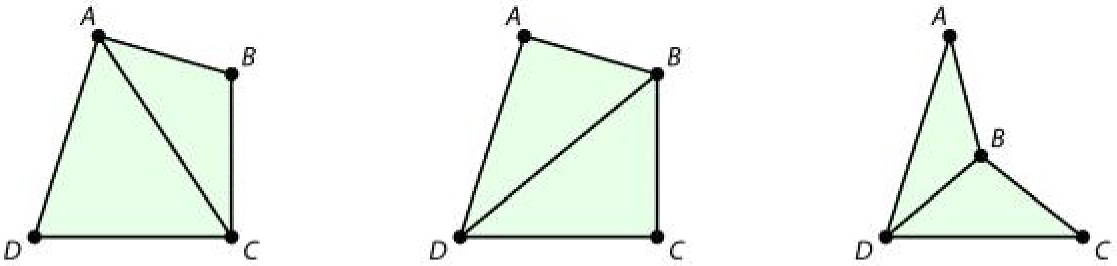
Triangulations

- ✓ What is interesting about these triangulations?



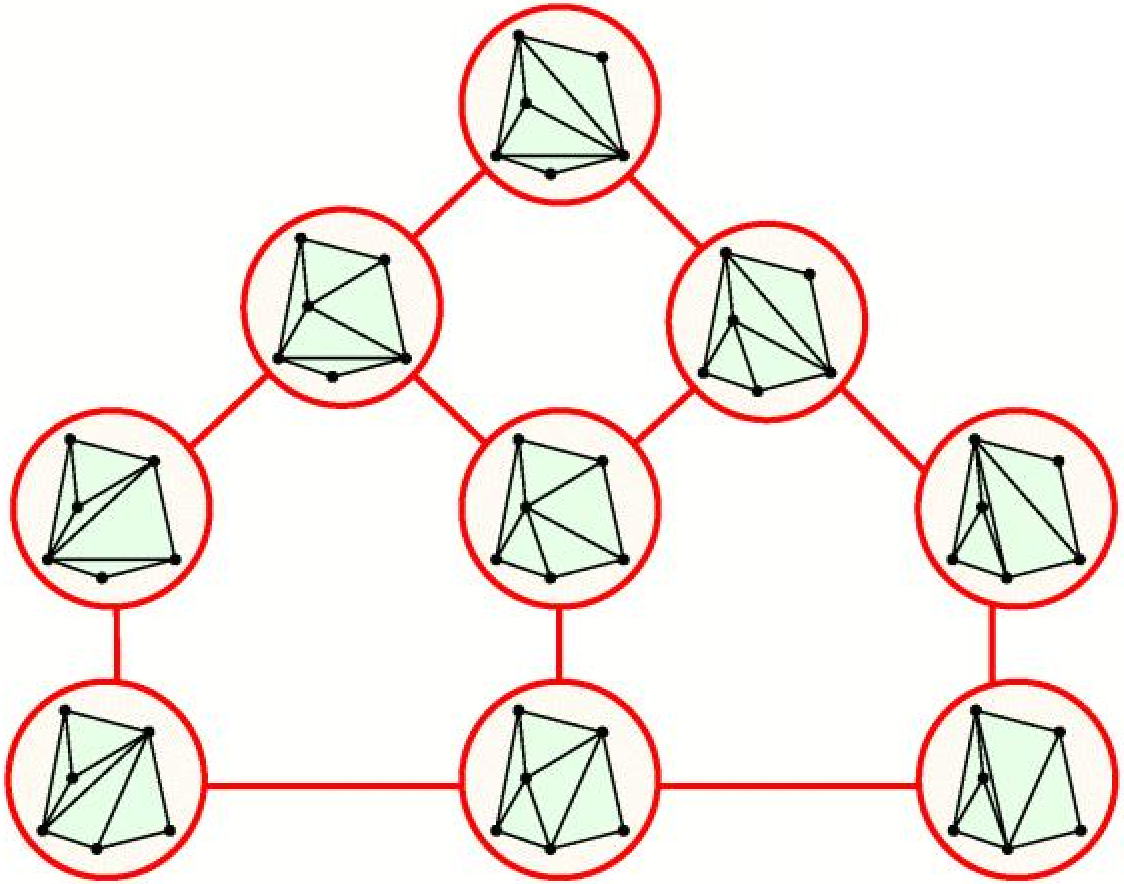
Triangulations

✓ Triangulations are related by the edge flip operation.



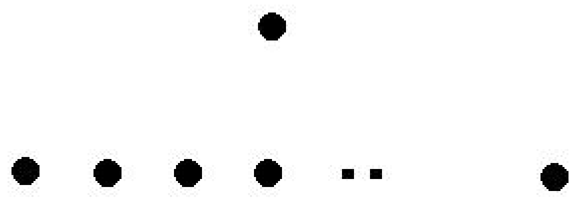
Triangulations

- ✓ Flip graph of a point set.
- ✓ Each node is a particular triangulation of the point set.



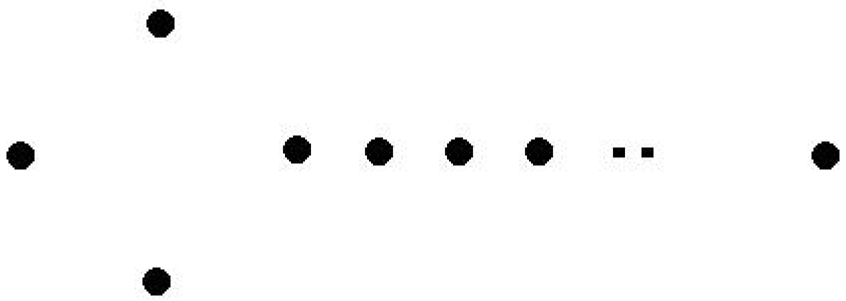
Triangulations

- ✓ For all n construct a point set with n points whose flip graph is a single node, i.e., no flip is possible.



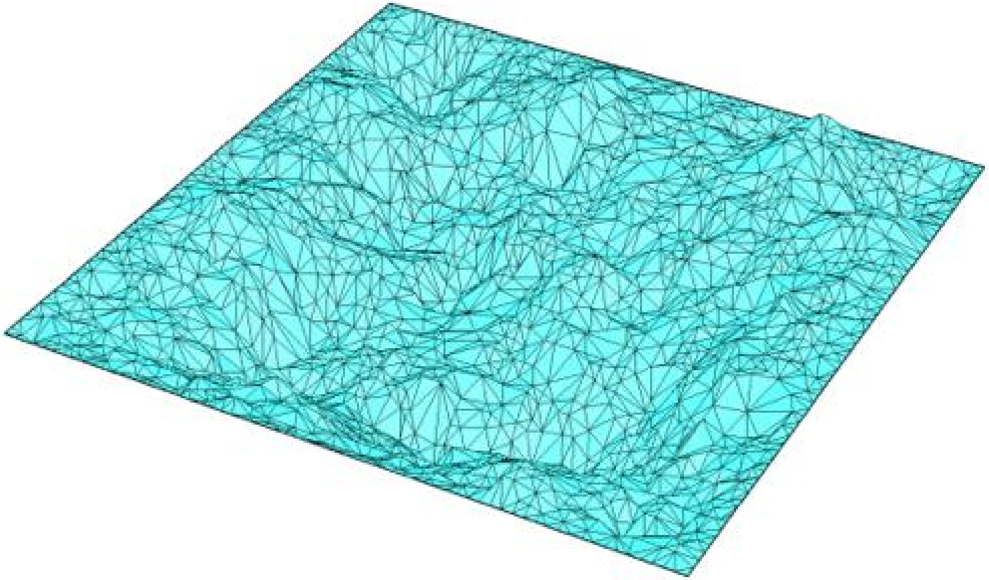
Triangulations

- ✓ For all n construct a point set with n points whose flip graph is 2 nodes, i.e., 2 flips are possible.



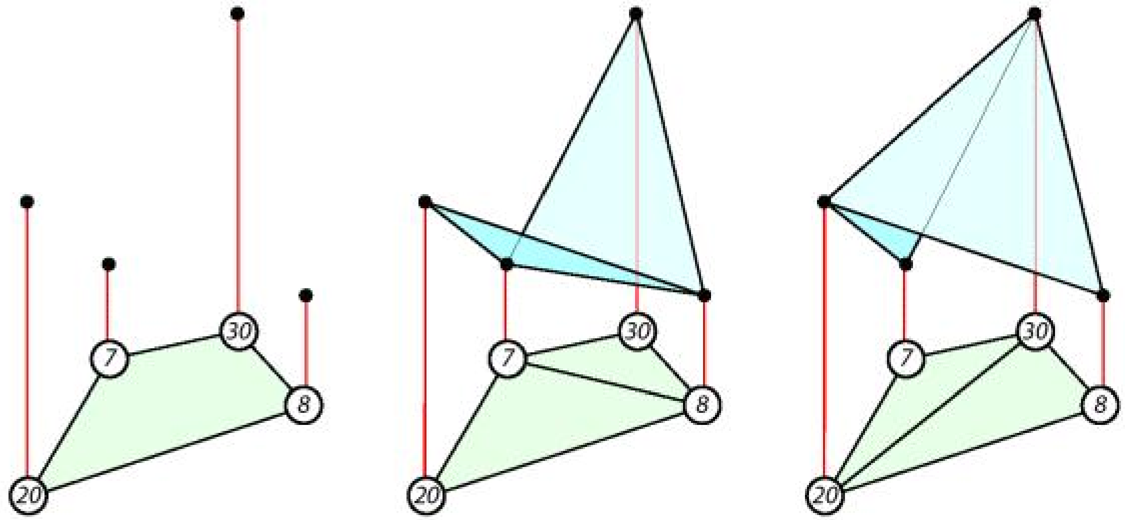
Delaunay Triangulation

- ✓ 3d maps of earth’s surface are constructed starting from a finite sample of points whose heights (altitude) are somehow measured.
- ✓ Triangulate these samples in 2D to approximate heights of nearby/unsampled points.
- ✓ Then lift each samples to its correct height, providing a piecewise-linear terrain of the earth.



Delaunay Triangulation

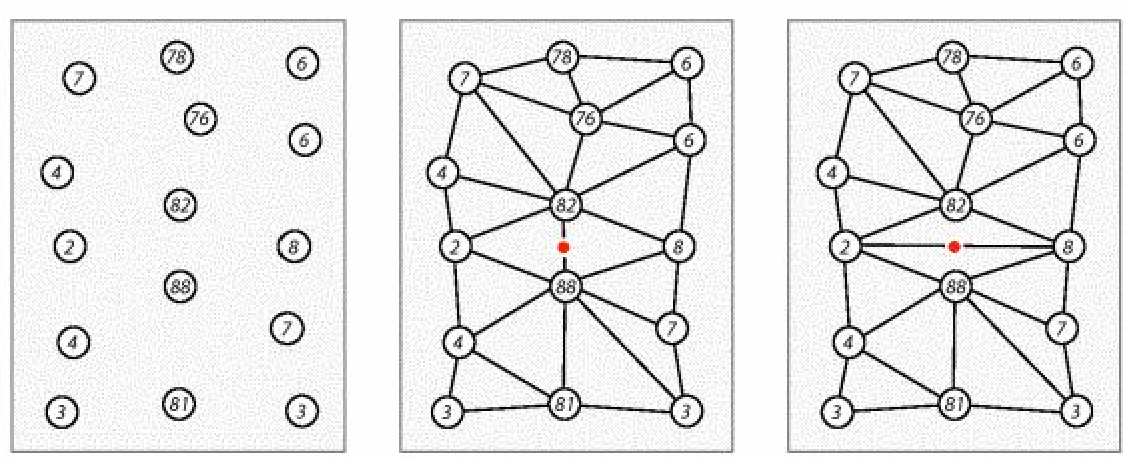
- ✓ Which triangulation is best for this reconstruction?



deep valley steep mountain

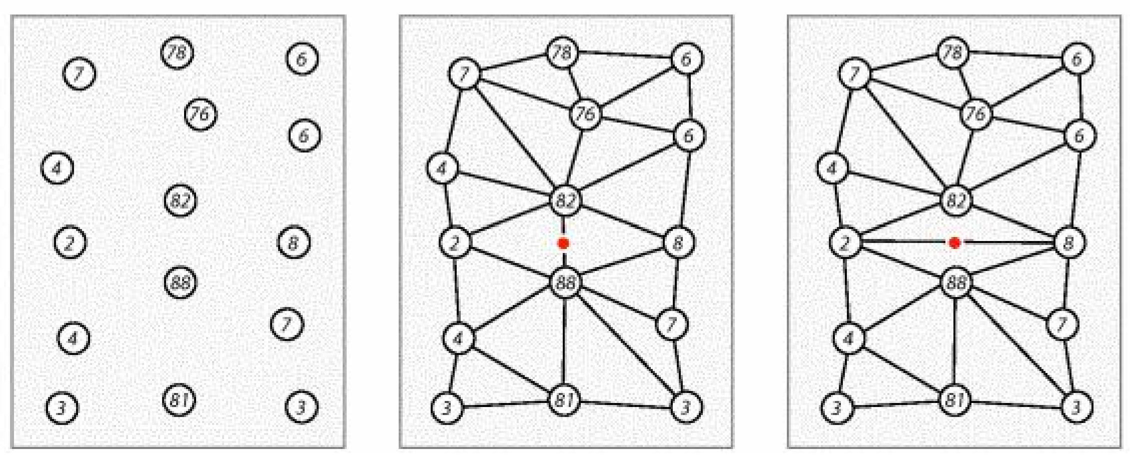
Delaunay Triangulation

- ✓ Our intuition renders some terrains more natural than others.
- ✓ Left seems to be taken from a mountain; achieved by middle; but right part creates a deep valley that cuts the ridge in two.



Delaunay Triangulation

- ✓ Our intuition renders some terrains more natural than others.
- ✓ What is cool about the middle one?
  - ✓ No skinny triangles compared to the right one.



- ✓ Avoid skinny triangles by picking a triangulation that maximizes the minimum angle: Delaunay Triangulation.

Delaunay Triangulation

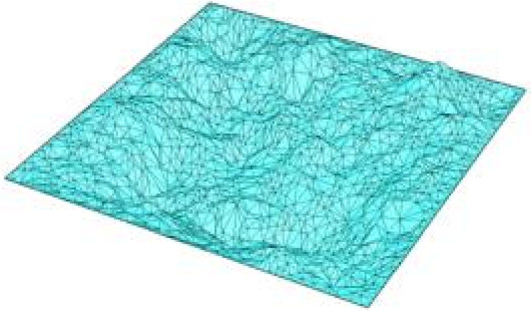
- ✓ Angle sequence (a1, a2, .., a3n) for a triangulation w/ n triangles, sorted from smallest a1 to largest a3n.
- ✓ Using the angle sequence we can compare 2 different triangulations.
- ✓ Number of triangles is constant →angle sequences have same length.

- ✓ T1: (20, 30, 45, 65, 70, 130)
- ✓ T2: (20, 30, 45, 60, 75, 130)
  - ✓ T1 > T2 (fatter) since 65 > 60.
  - ✓ The fattest triangulation is the one we desire.
  - ✓ Find it easily by edge flips.

Delaunay Triangulation

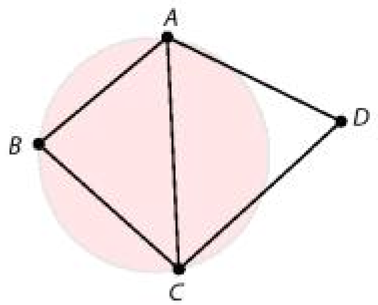
- Let e be edge of a triangulation T1 and let Q be the quad formed by 2 triangles having e as common edge. If Q is convex, let T2 be the triangulation after flipping e in T1. We say e is legal if T1 >= T2 and illegal otherwise.
- Convex hull edges of a triangulation are also legal.

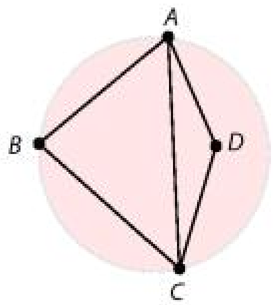
- Delaunay triangulation of a point set is a triangulation that only has legal edges.
- Delaunay triangulation is a triangulation in which interior of any circumcircle of a triangle is empty, i.e., no points inside circumcircle.

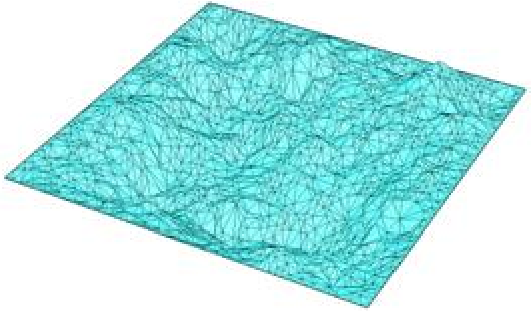


Delaunay Triangulation

- ✓ Put simply, a triangulation is Delaunay if no point lies within the circumcircle of any triangle.
  - ✓ It also happens to maximize the minimum angle of any triangle, which is why it's useful.

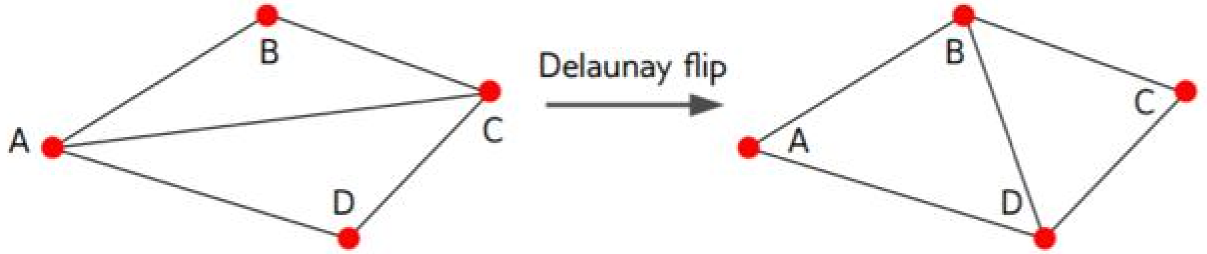




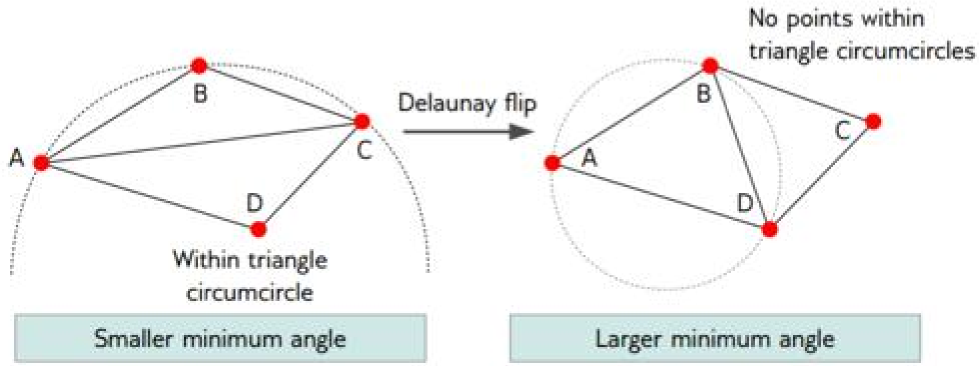


Delaunay Triangulation Construction

- ✓ Start with any triangulation T (e.g., slide 17 or 21). If T has an illegal edge, flip the edge and make it legal. Continue flipping illegal edges.



- ✓ Put simply, flip edge if the minimum of the 6 angles increases (above).
  - ✓ Guaranteed to converge (since minimum angle increases).

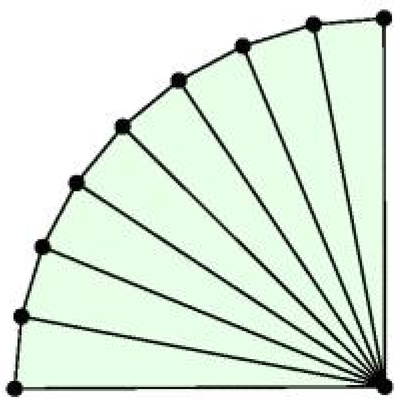


  - ✓ Only 1 of the 2 configurations satisfies the circumcircle property.

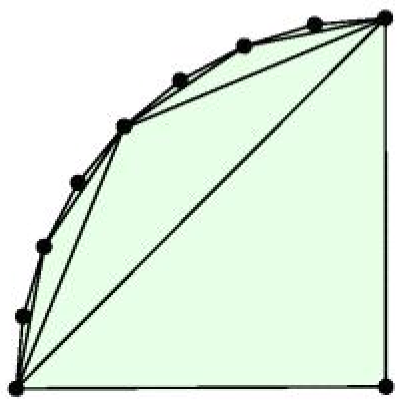
Other Special Triangulations

- ✓ Minimum weight triangulation (MWT).
- ✓ Min ink to draw it compared to all other triangulations.
- ✓ App: Minimize wiring cost of building a network.

- ✓ Skinny triangles have long edges so use Delaunay to avoid them?
- ✓ Not really.



Delaunay

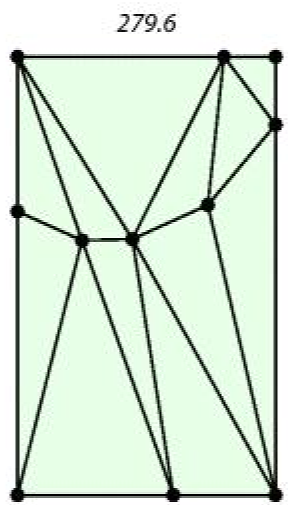


MWT

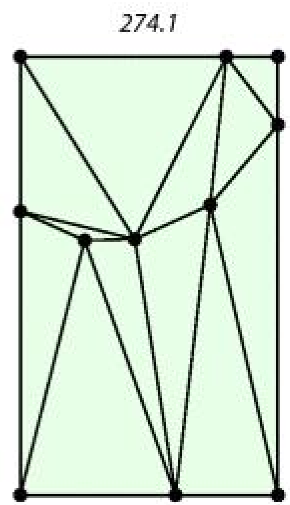
Other Special Triangulations

- ✓ Minimum weight triangulation (MWT).
- ✓ Min ink to draw it compared to all other triangulations.
- ✓ App: Minimize wiring cost of building a network.

- ✓ Skinny triangles have long edges so use Delaunay to avoid them?
- ✓ Not really.



Delaunay



MWT

Other Special Triangulations

- ✓ Minimum weight triangulation.
- ✓ Proved to be NP-hard: no polynomial-time algo known.
- ✓ Alternative: instead of complete triangulation of a point set, use a tree that spans the point set; minimum spanning tree (MST).
- ✓ Theorem: MST of a point set is a subset of its Delaunay triangulation.
  - ✓ Justifies the intuition that Delaunay triangulation is weight-minimizing in some sense.

Triangulations of 3D Points

- Find local neighborhood L_i of each point p_i in the 3D point cloud input.
  - Closest k points (using a k-d tree).
- For each L_i compute tangent plane using PCA.
  - If the input is oriented, just take the average of all normals for the plane normal.Use the mean pnt as the plane pnt (for both oriented/unoriented).
- Project all points in L_i to the tangent plane and compute their 2D Delaunay triangulation D_i.
- D_i is a set of edges: $D_i = \{e_i^1, e_i^2, .., e_i^{noe(i)}\}$ where $noe(i)$ where is the number of edges of the i^{th} Delaunay triangulation.
- Final triangulation is the composition of all N local triangulations:

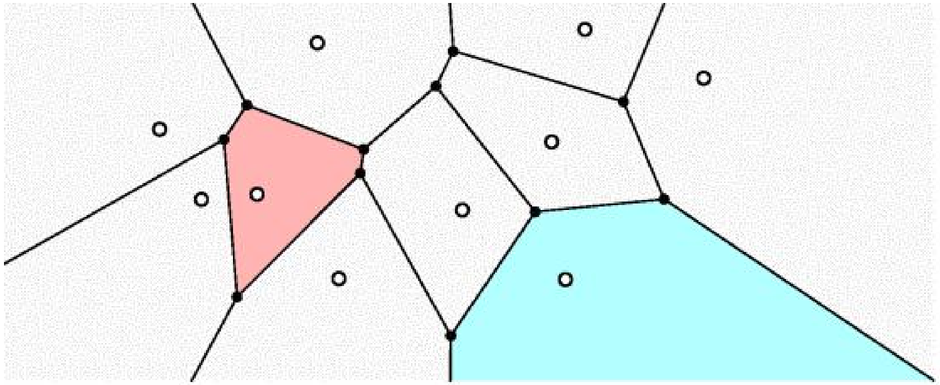
$$D = \bigcup_1^N \{e_i^1, e_i^2, \ldots, e_i^{noe(i)}\}.$$

- Note that global D not necessarily a 2-manifold. Set $k = 0.02n$ and restrict value to [8,12].
- See the Crust Algorithm in Surface Reconstruction slides as well (more principled).

Voronoi Diagrams

- ✓ Convex hull ~ boundary of S.
- ✓ Triangulation ~ interior of S.
- ✓ Voronoi ~ points not in S.

- ✓ Set of points x not in S, that are at least as close to Voronoi site p as to any other Voronoi site q in S.

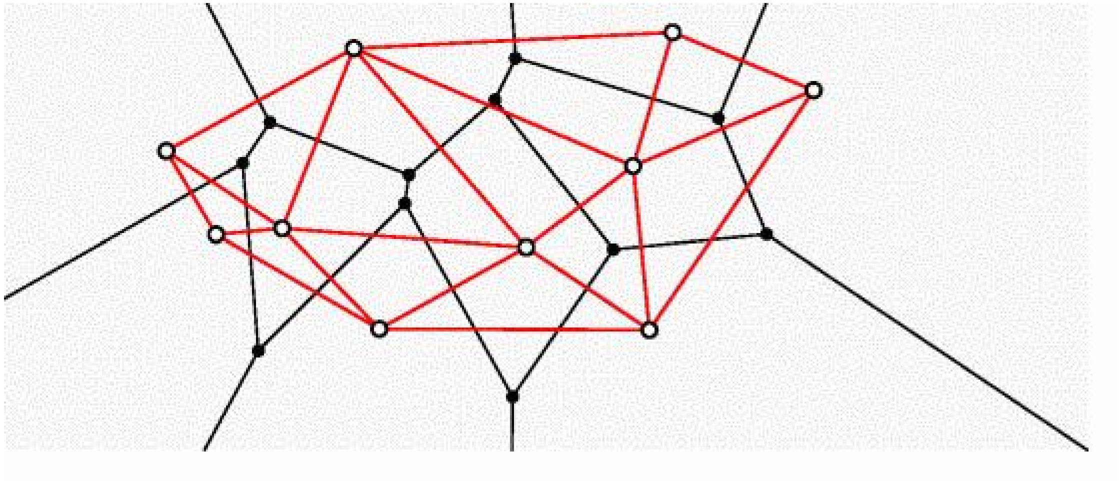


- ✓ Voronoi region for the site p_k is $R_k = \{x \mid ||x - p_k|| \leq ||x - p_j||, j \neq k\}$.
- ✓ Voronoi vertices (black dots) are equidistant from 3 neighboring sites.
- ✓ https://youtu.be/yDMtGT0b_kg

Voronoi Diagrams Application # 1

- ✓ Convex hull ~ boundary of S.
- ✓ Triangulation ~ interior of S.
- ✓ Voronoi ~ points not in S.

- ✓ Dual of Delaunay triangulation: If Voronoi regions adjacent, connect their cites w/ a Delaunay edge.

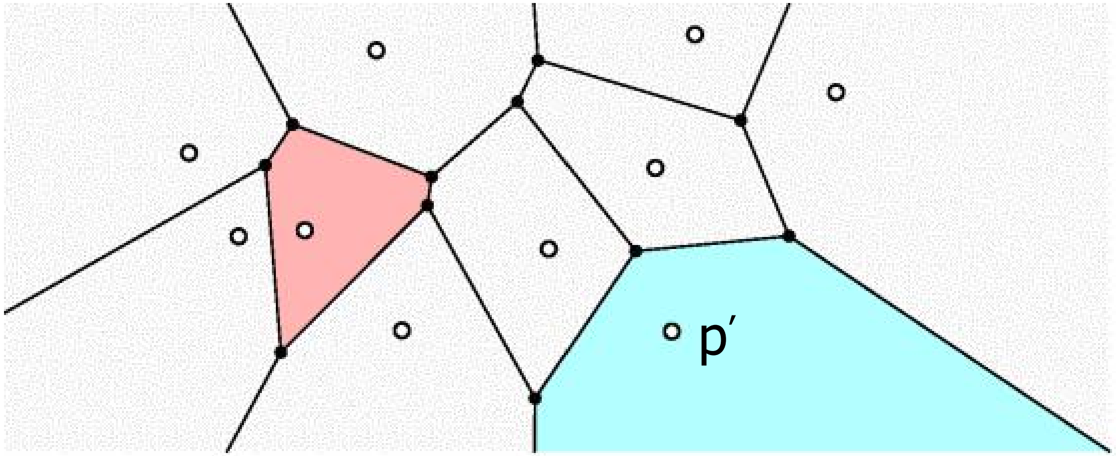


- ✓ https://youtu.be/PWlo7PvtQVw?t=2154

Voronoi Diagrams Application # 2

- ✓ Convex hull ~ boundary of S.
- ✓ Triangulation ~ interior of S.
- ✓ Voronoi ~ points not in S.

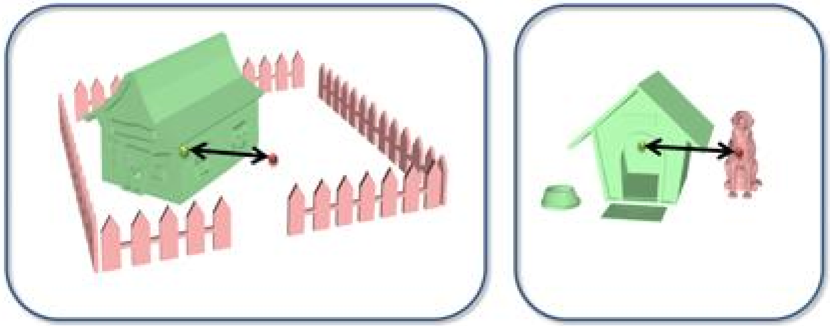
- ✓ Set of points x not in S, that are at least as close to Voronoi site p as to any other Voronoi site q in S.



- ✓ Nearest service handling: e.g., post office – apartment assignment. All the cyan apartments will be served from the site p′.

Voronoi Diagrams Application # 3

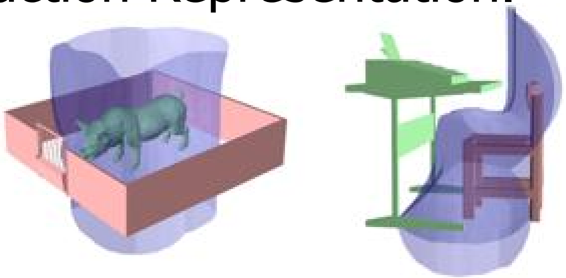
- ✓ Another application of Voronoi Diagrams: Interaction Representation.
  - ✓ Also related to finding the separating hyperplane w/ max margin in the SVM.
- ✓ Distance-based representation (not so powerful).

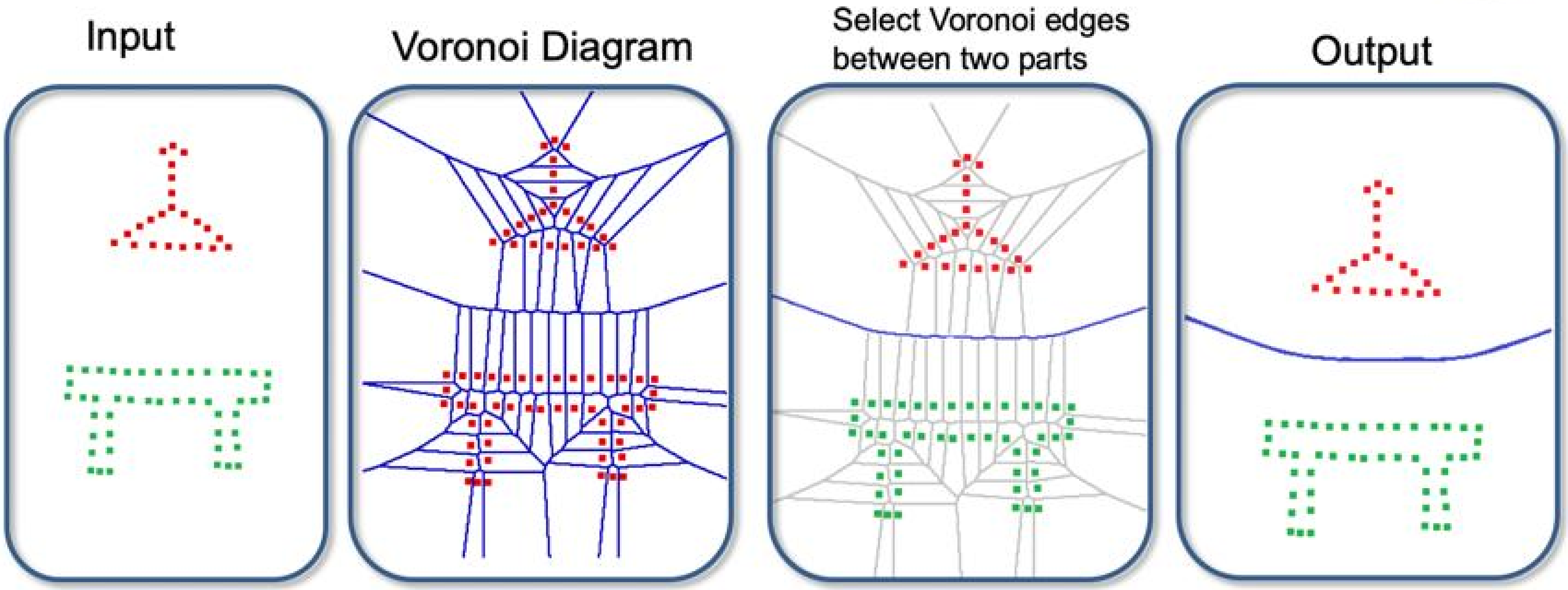


- ✓ Interaction bisector representation (powerful).

Voronoi Diagrams Application # 3

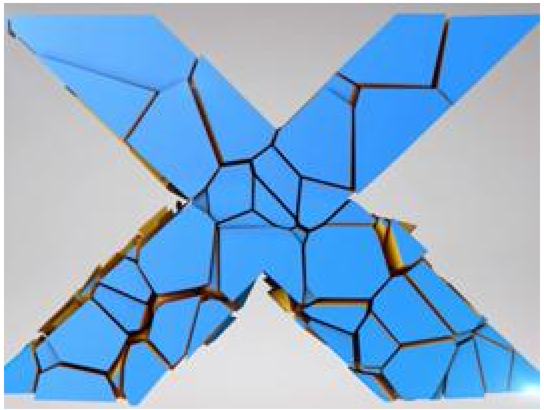
- ✓ Another application of Voronoi Diagrams: Interaction Representation.
  - ✓ Also related to the hyperplanes in SVM.
- ✓ Interaction bisector representation (powerful).



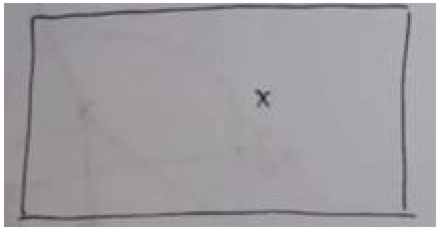
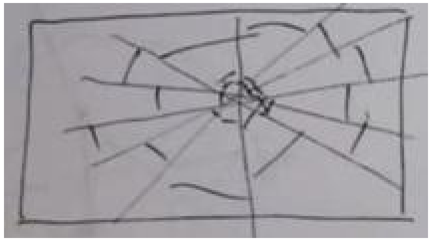
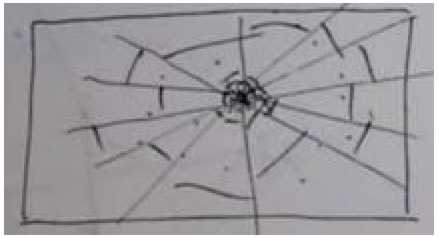


Voronoi Diagrams Application # 4

- ✓ Another application: Pre-fracture pattern creation for tearing/breaking.

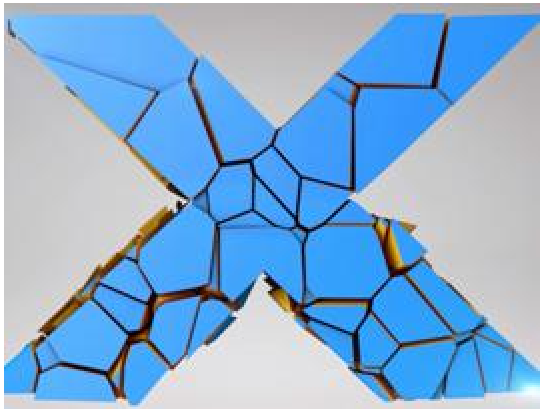
- ✓ If you know where the bullet will hit in advance, pre-fracture possible.
  - ✓ Distribute dense Voronoi sites around bullet point, sparse everywhere else.

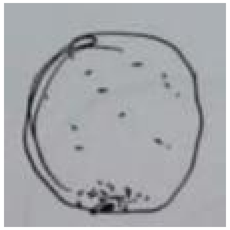
  - ✓ Fracture in action: https://youtu.be/Lg2dqFCU67Q

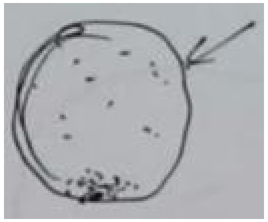
Voronoi Diagrams Application # 4

- ✓ Another application: Pre-fracture pattern creation for tearing/breaking.



- ✓ If you don't know the contact point, rotate onthefly to match the dense Voronoi part with the collision point during simulation, e.g., for sphere.

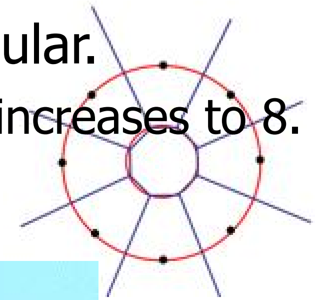




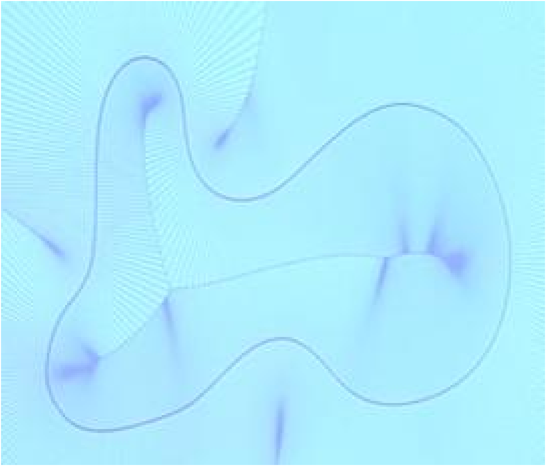
- ✓ For non-symmetric shapes, a good research problem: Real Time Dynamic Fracture with Volumetric Approximate Convex Decompositions.

Voronoi Diagrams Application # 5

- ✓ Another application: Medial axis extraction.
- ✓ Medial axis is the set of interior points that have more than 1 closest boundary points.
  - ✓ Superset of curve skeleton.
  - ✓ Useful in many apps, e.g., deform skeleton and boundary follows (LBS).
- ✓ Each Voronoi vertex has degree 3 if no 4 points are co-circular.
  - ✓ Smaller circle shrinks to nothing (no site inside) and degree increases to 8.

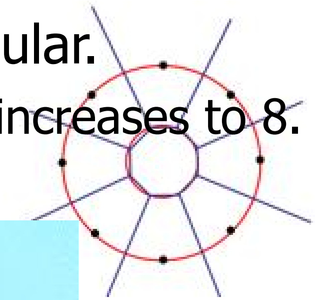


- ✓ Compute Voronoi diagram of the (boundary) points.
- ✓ Voronoi vertices give you the med axis:



Voronoi Diagrams Application # 5

- ✓ Another application: Medial axis extraction.
- ✓ Medial axis is the set of interior points that have more than 1 closest boundary points.
  - ✓ Superset of curve skeleton.
  - ✓ Useful in many apps, e.g., deform skeleton and boundary follows (LBS).
- ✓ Each Voronoi vertex has degree 3 if no 4 points are co-circular.
  - ✓ Smaller circle shrinks to nothing (no site inside) and degree increases to 8.



- ✓ Compute Voronoi diagram of the (boundary) points.
- ✓ Sensitive to noise, 1 input pnt inside boundary dooms the medial axis: