Computational Geometry

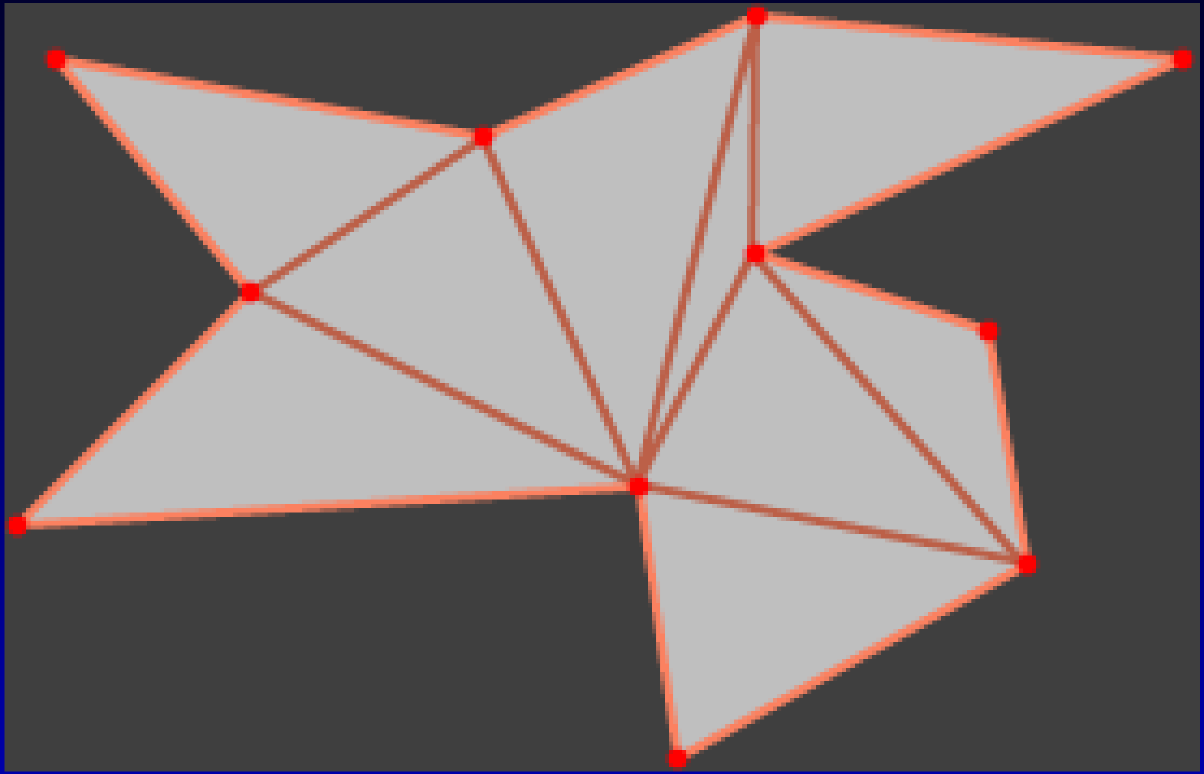
- What is computational geometry?
- How is it relevant to graphics?
- Where is the research potential?

Computational Geometry

Algorithmic study of combinatorial geometry.

- many simple elements (points, lines, triangles).
- queries and constructions.
- optimal algorithms and lower bounds.

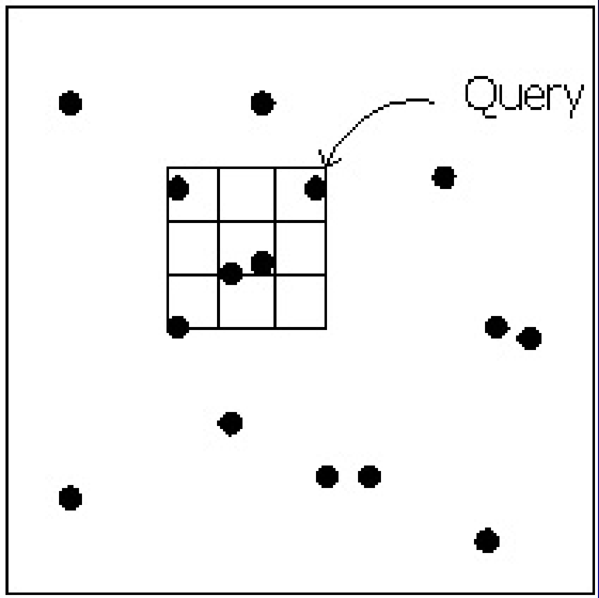
Polygon Triangulation



Decompose polygonal region into triangles.

- 2D: $n \log n$ for n vertices.
- 3D: $nr + r^2 \log r$ for $r = \mathrm{O}(n^2)$ reflex vertices.

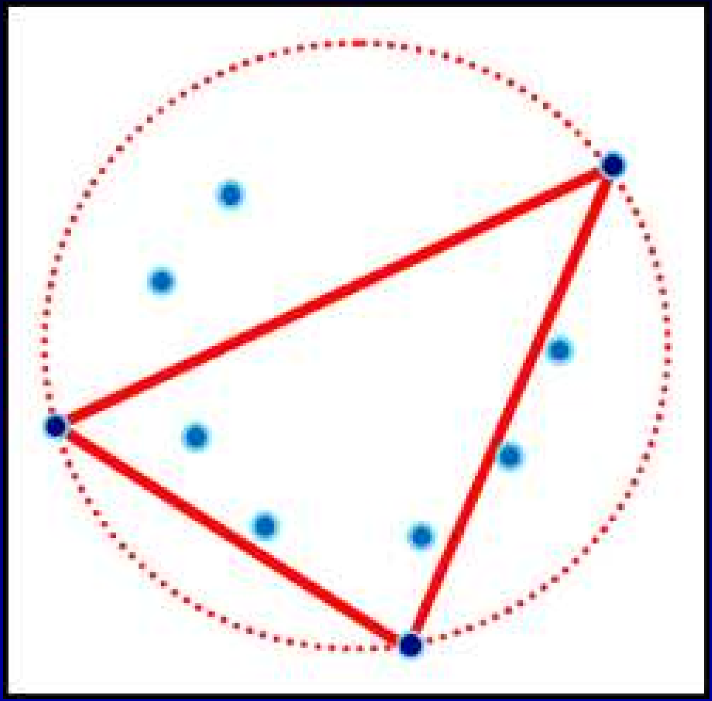
Range Search



Find points in axis-aligned box.

- 2D: $k + \log n$ query; $n \log n$ preprocessing for n points and k outputs.
- 3D: $k + \log^2 n$ query; $n \log^2 n$ preprocessing.
- Octrees and bsp trees: $k + n^2$ and $k + n^3$.

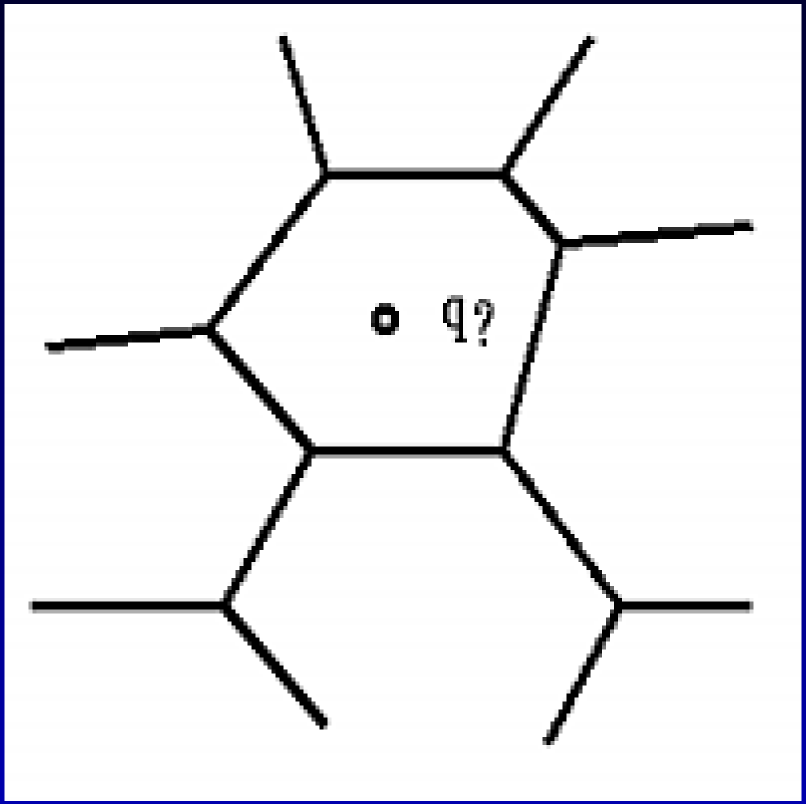
Simplex Search



Find points in triangle.

- 2D: $\log n$ query time; n preprocessing time.
- 3D: $\log n$ query time; n^2 preprocessing.
- complicated algorithms.

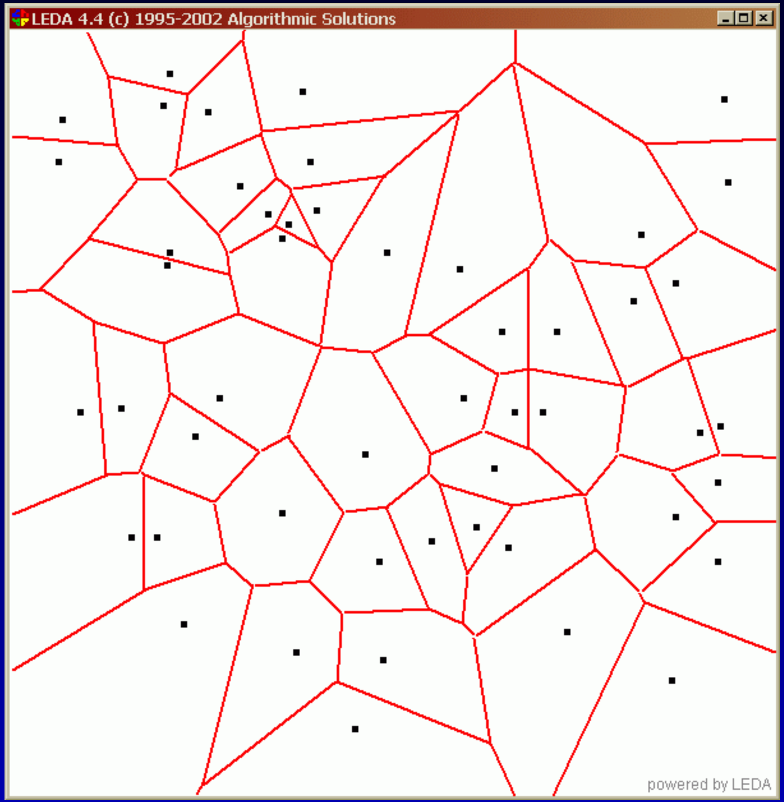
Point Location



Locate point in triangle mesh.

- 2D: $\log n$ query; $n \log n$ preprocessing for n triangles.
- 3D: open problem!

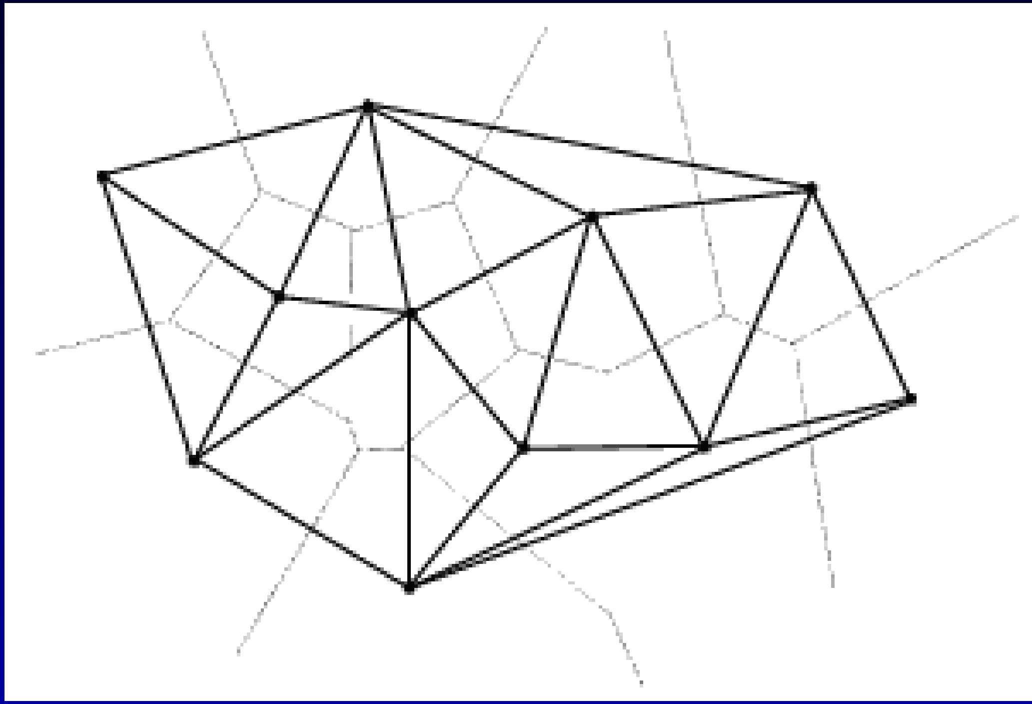
Voronoi Diagram



Compute the region that is closest to each site.

- 2D: $n \log n$ for n sites.
- 3D: $k + n \log n$ for output size $k = O(n^2)$.

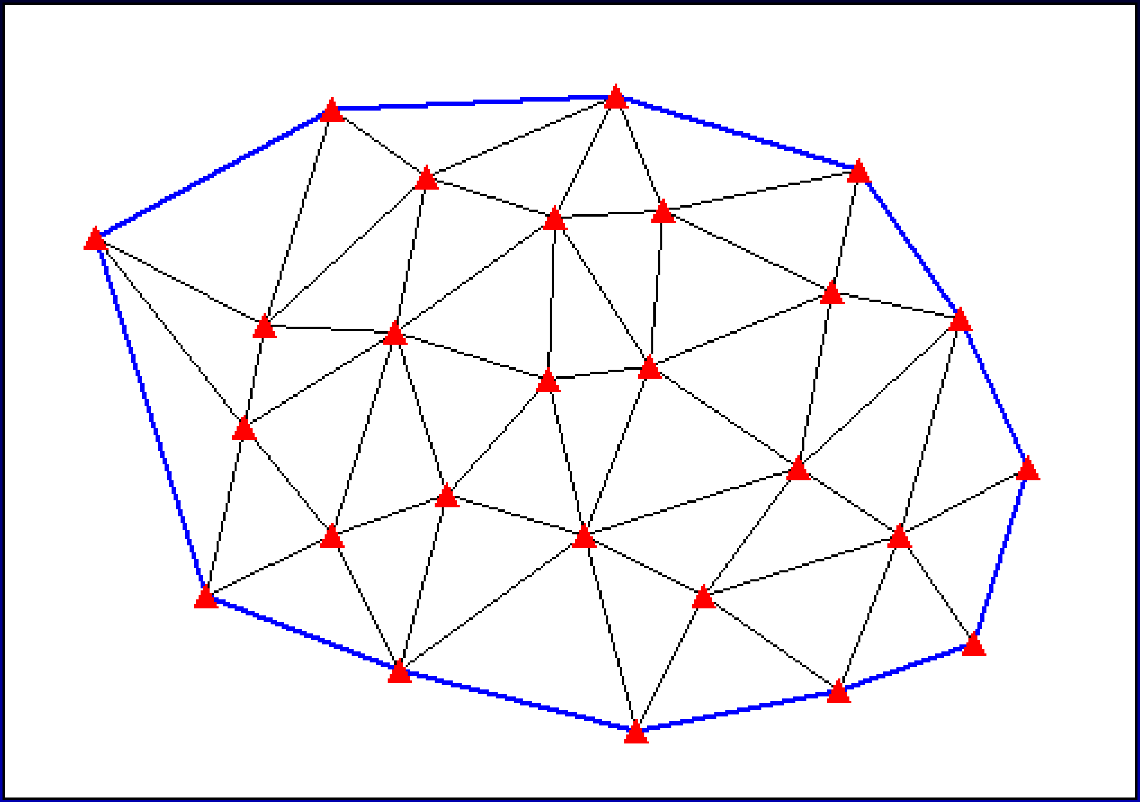
Delaunay Triangulation



Triangulation with maximal minimum angle.

- Equivalent to Voronoi diagram.
- convex hull in dimension d gives Delaunay triangulation in dimension $d - 1$.

Convex Hull



Smallest convex region containing points.

- 2D: $n \log n$ for n points.
- 3D: $n \log n$.

Graphics Relevance

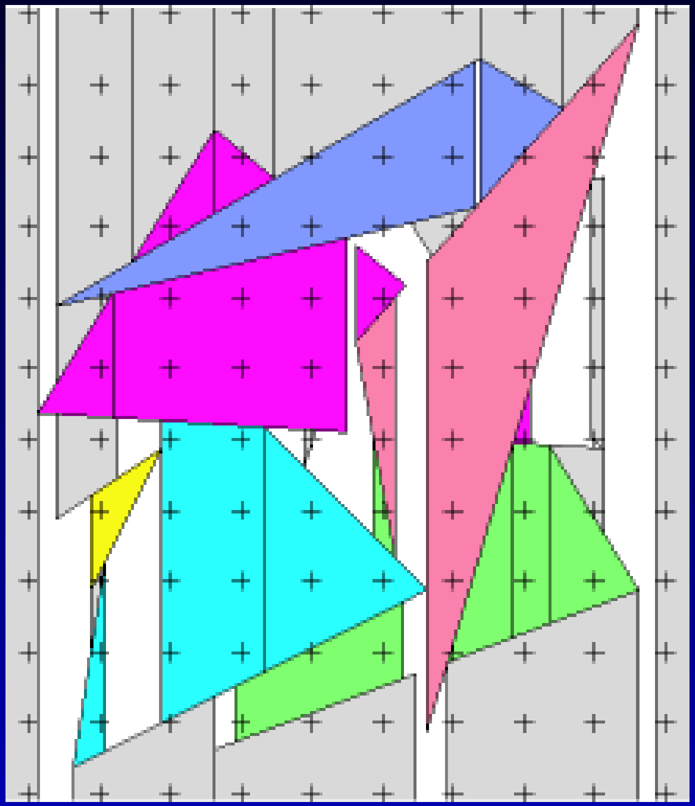
How is computational geometry relevant to graphics?

- Shared goal: fast algorithms for large geometry problems.
- Divergent strategies:
    - theory *versus* practice.
    - asymptotic *versus* real-world optimality.
    - continuous *versus* discrete.

Consequences

- no CG in graphics courses.
- no CG in graphics pipeline.
- no CG in graphics programming (except triangulation).
- little CG in graphics research.

Hidden Surface Detection



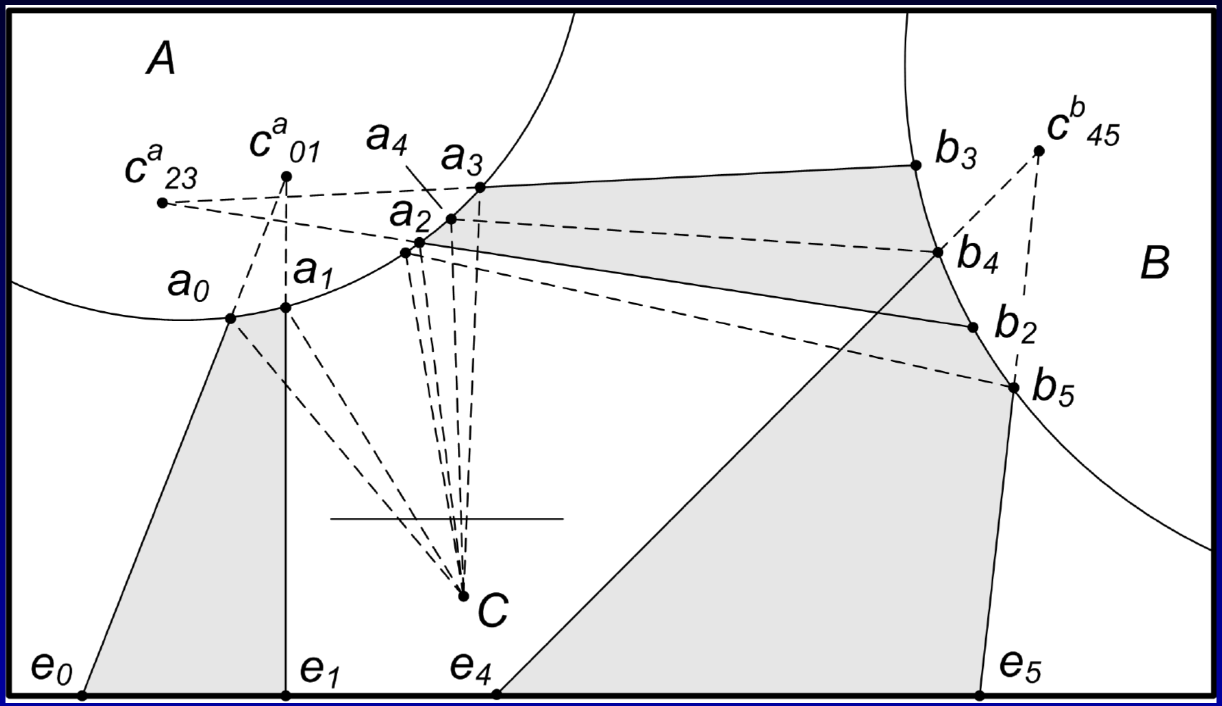
- z buffering: 2D, fixed resolution, coherence.
- spatial partitions: heuristic, $\mathrm{O}(n^3)$ for n triangles.
- bottleneck for large scenes.

How about this?

1. project triangles onto image plane.
2. construct planar arrangement.
3. rasterize closest triangle per cell (given by arrangement).

For n triangles, $n \log n$ with small constant factor.

Reflections



- Challenge: fast projection of reflected points.
- Ray tracing is slow:
  high cost per pixel, poor coherence, aliasing.
- Computational geometry approach:
  interpolate reflected rays near scene point.