Writing a Tetrahedralizer for Blender

Matthias Müller, Ten Minute Physics

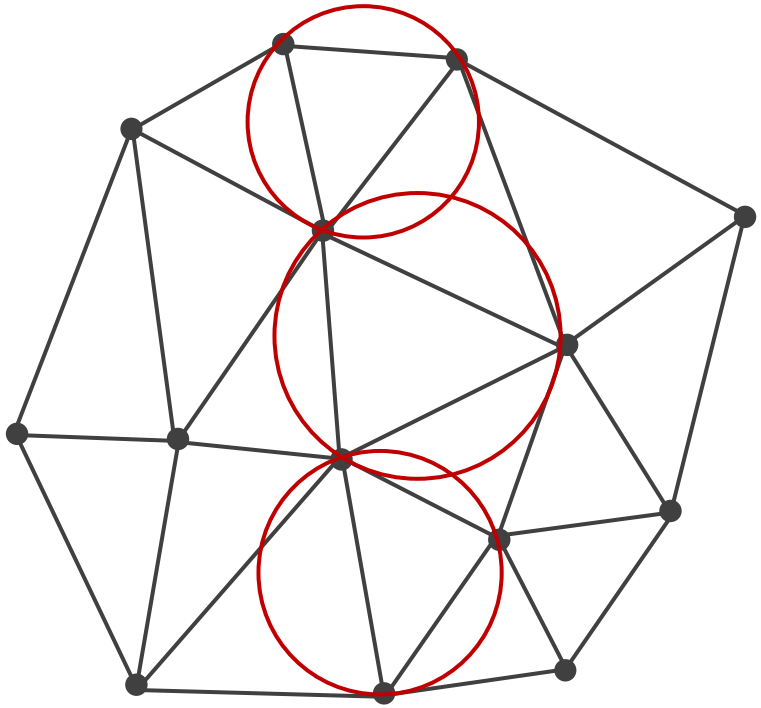
www.matthiasmueller.info/tenMinutePhysics

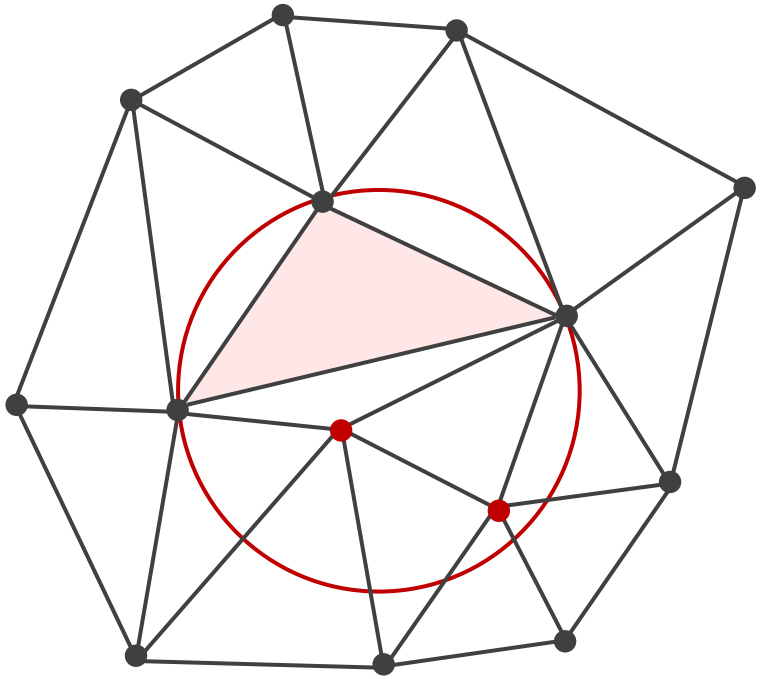
Using the Incremental Delaunay Method

Triangulation: Easier to visualize
Tetrahedralization: Analog

Delaunay Mesh

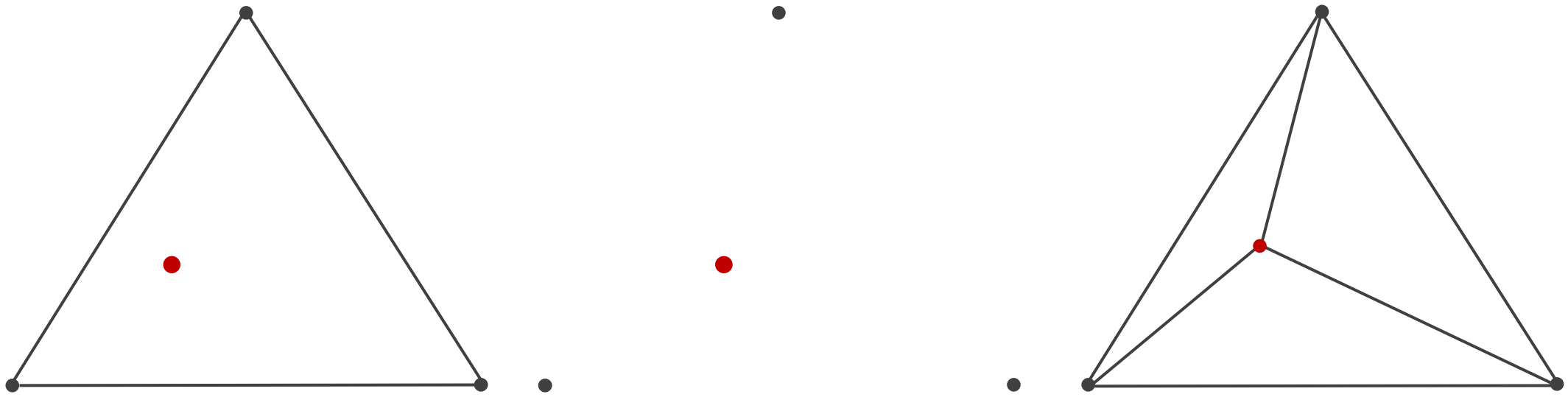
- The circumsphere of any tetrahedron only contains the four adjacent points





Incremental Tetraheralization Algorithm

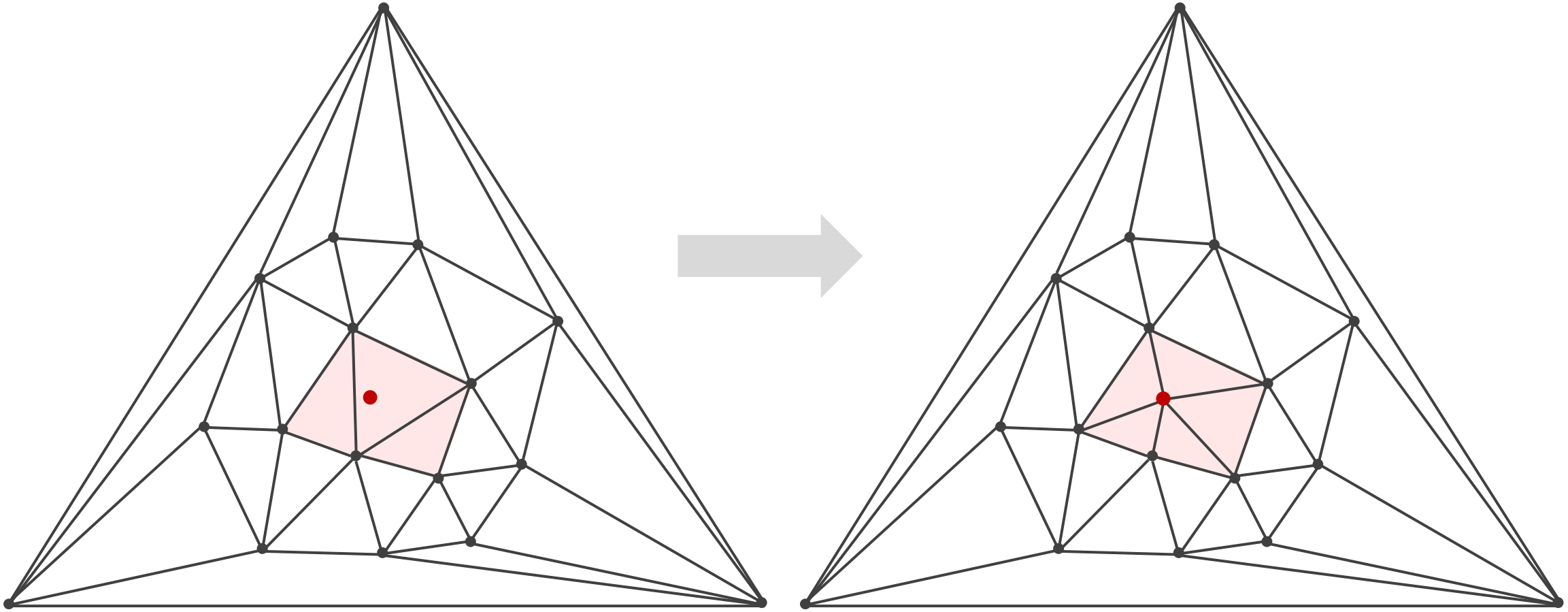
- Input: vertices of the surface mesh
- Start with 4 temporary points forming a big tetrahedron containing all points
- Add the new point



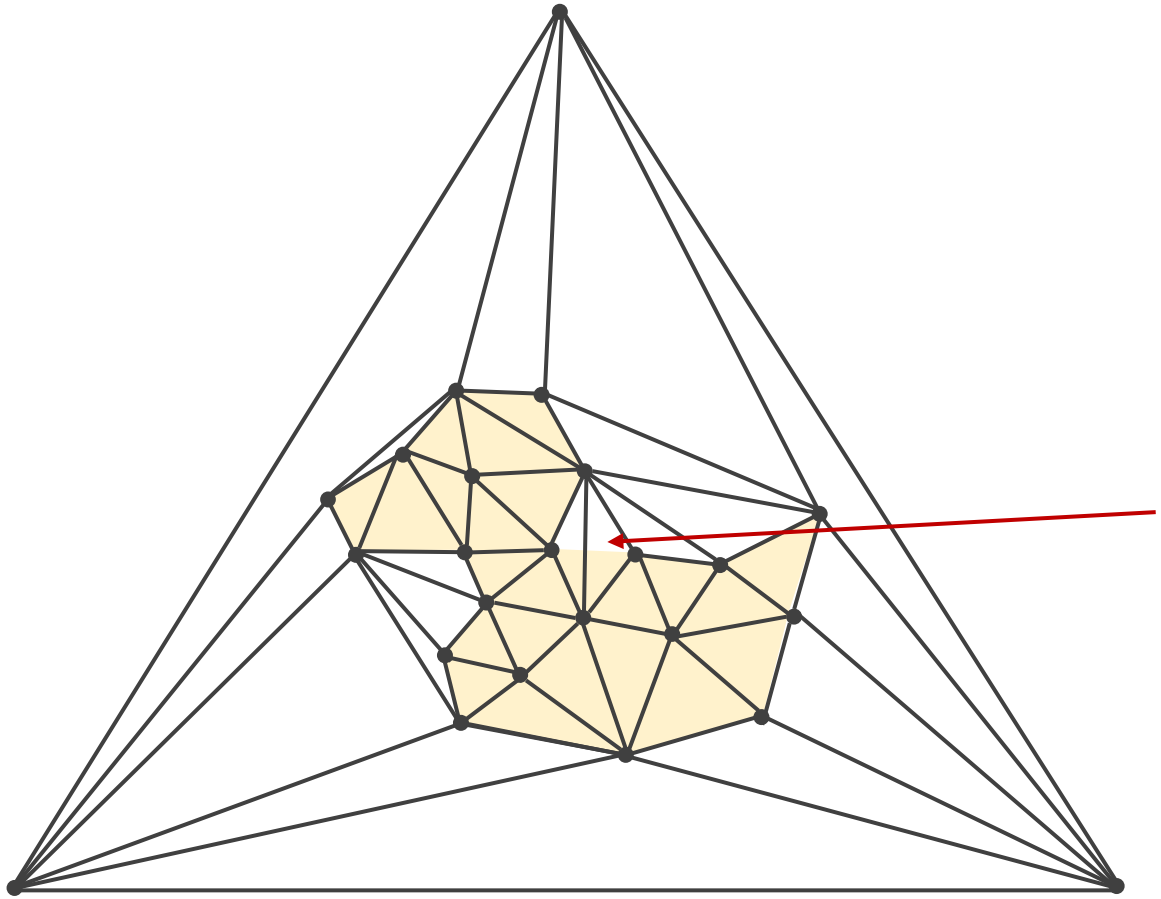
- Remove all tetrahedra whose circumsphere contains the new point
- Fill the void with a tetrahedral fan centered at the new point

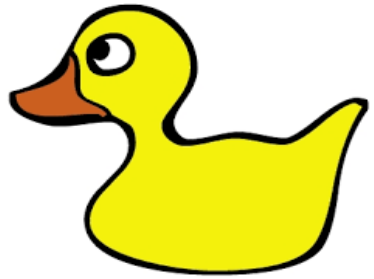
Incremental Point Insertion

- Later in the process:



Result:





Two problems:

- Too many tetrahedra

  → remove tetrahedra whose center lies outside the mesh

- Not matching the surface (non-conforming)
- Very difficult problem! Large body of work

  → Keep input triangle indices

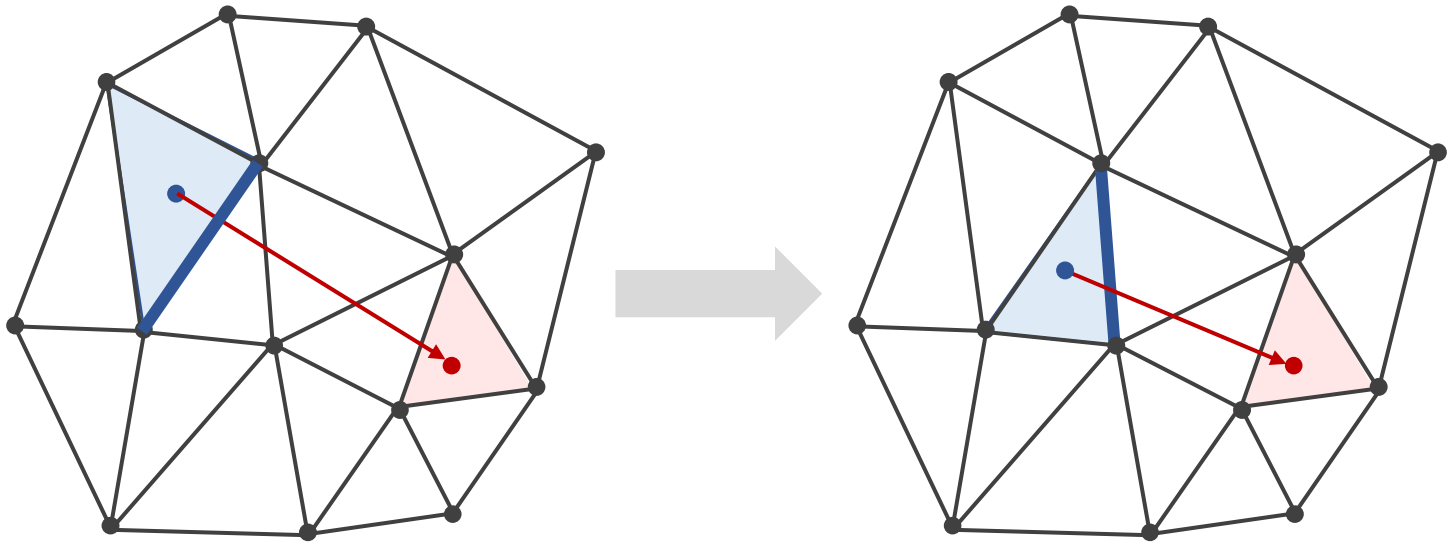
  → Use them for collision handling

  → Visual mesh embedding still works

Fast Implementation

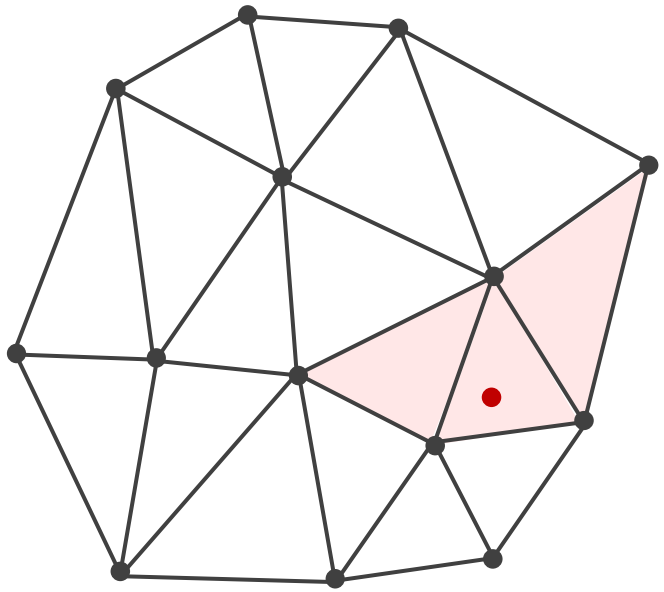
Find Violating Tetrahedra

- Find tetrahedron containing the new point



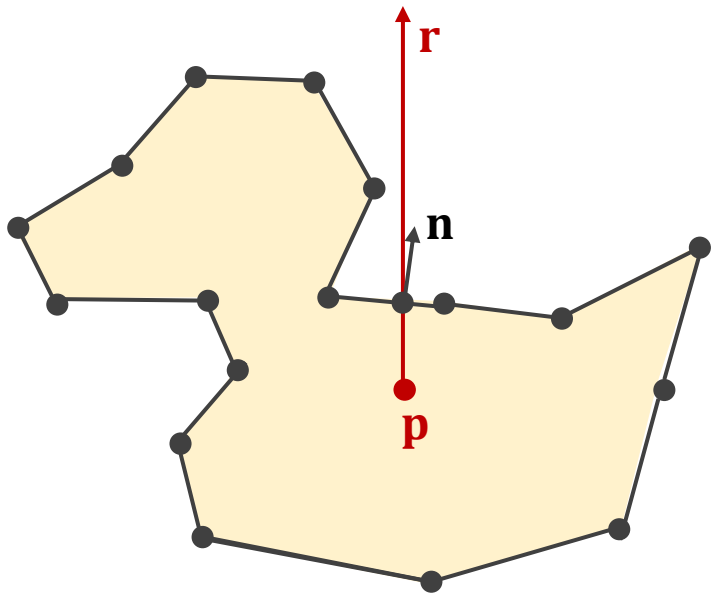
- Start with any tetrahedron / solution of last point
- Create ray from tetrahedral center to the new point
- Find the intersected face
- Move across the face to the adjacent tetrahedron

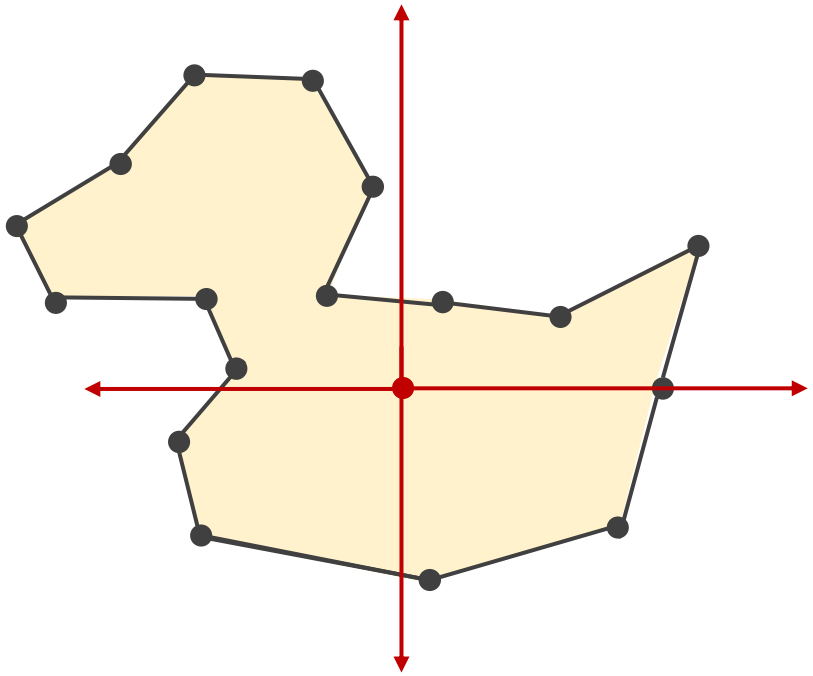
Find Violating Tetrahedra



- Start from containing tetrahedron
- Flood neighborhood checking the Delaunay condition

Inside Test for Tetrahedra Removal





- Create ray $(\mathbf{p}, \mathbf{r})$ in any direction
- Find closest intersection point and normal using triange BVH
- Inside if $\mathbf{r} \cdot \mathbf{n} > 0$

- More robust:
- Test in all canonical directions
- Majority vote