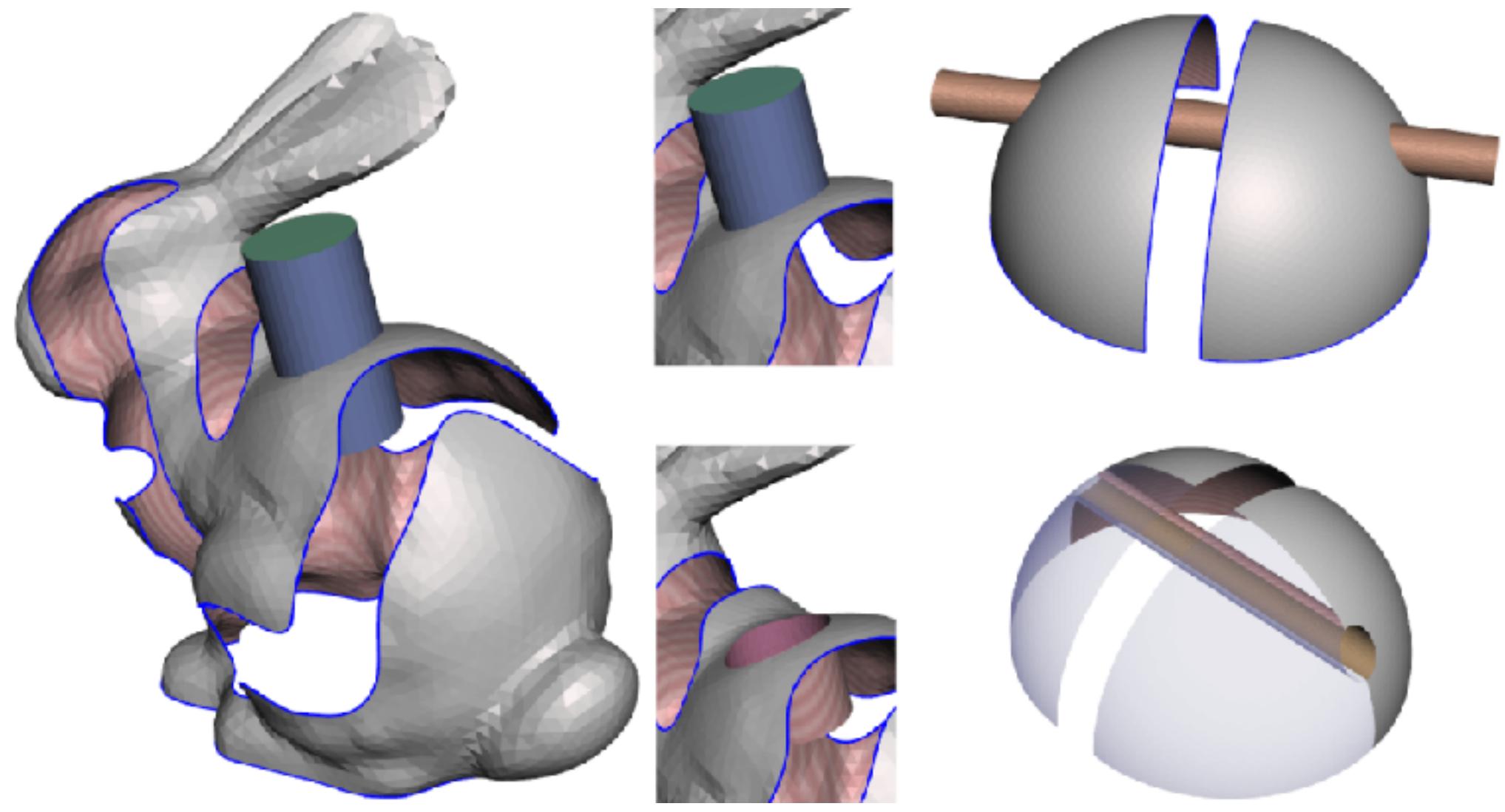
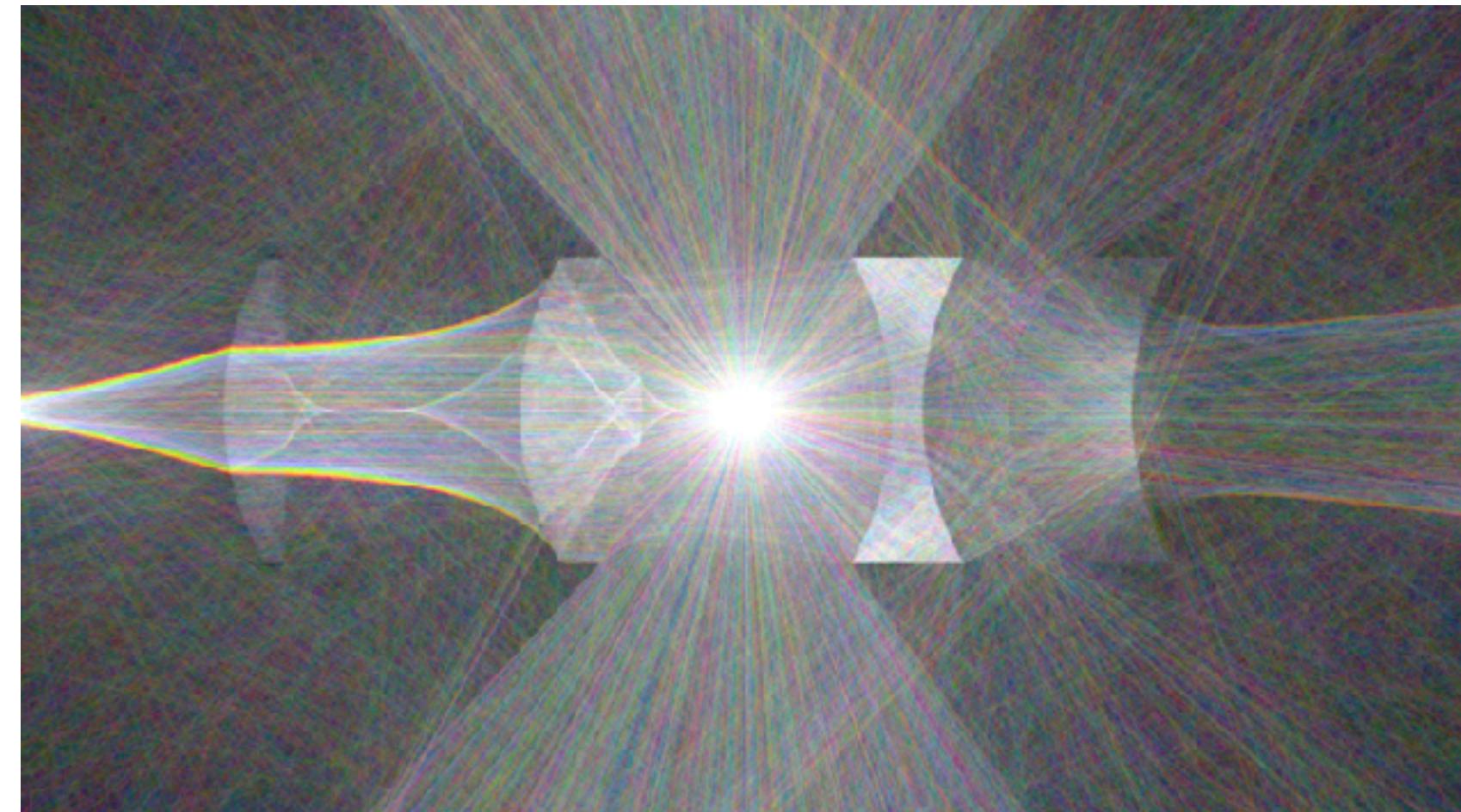
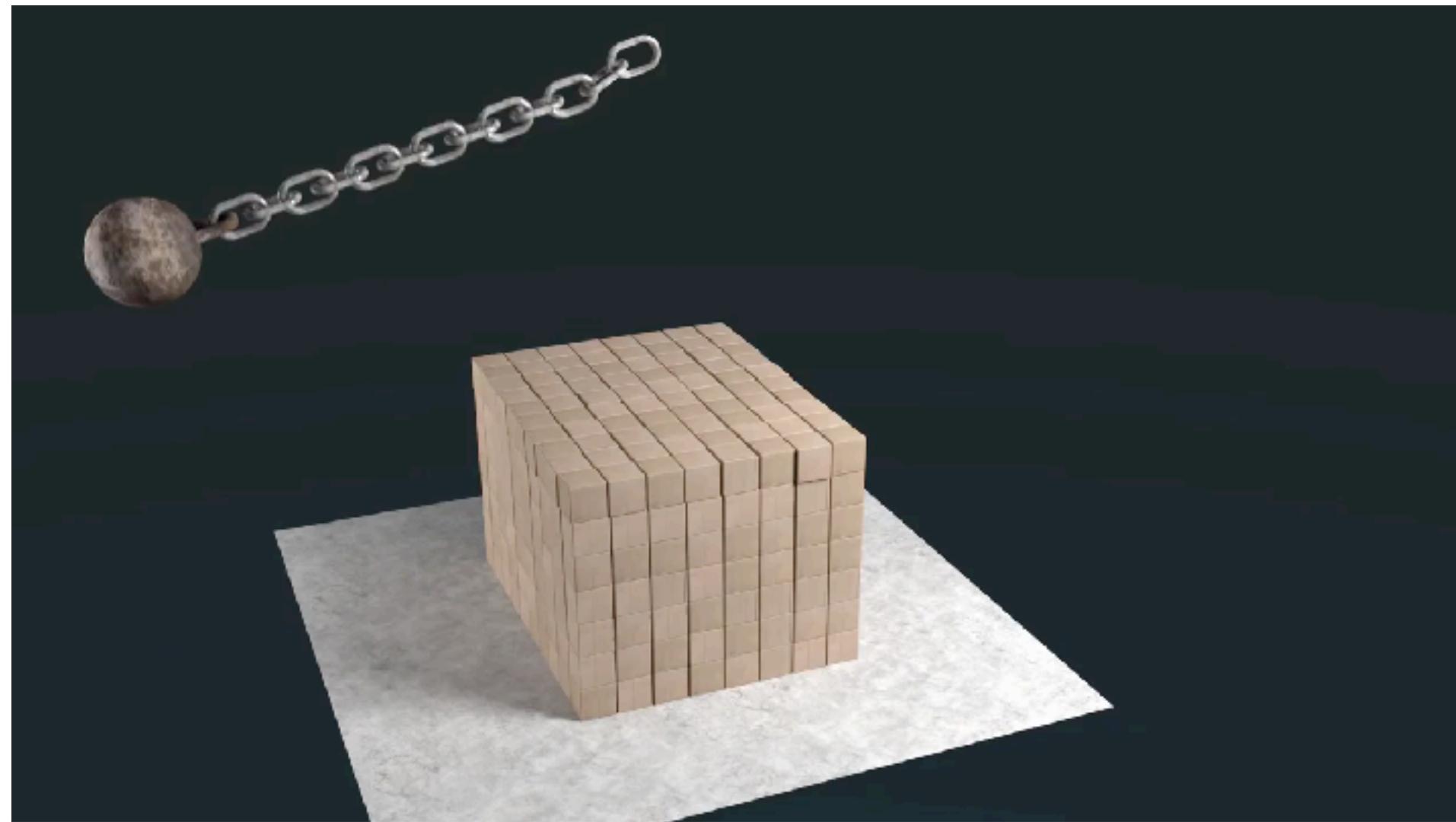


Geometric Queries

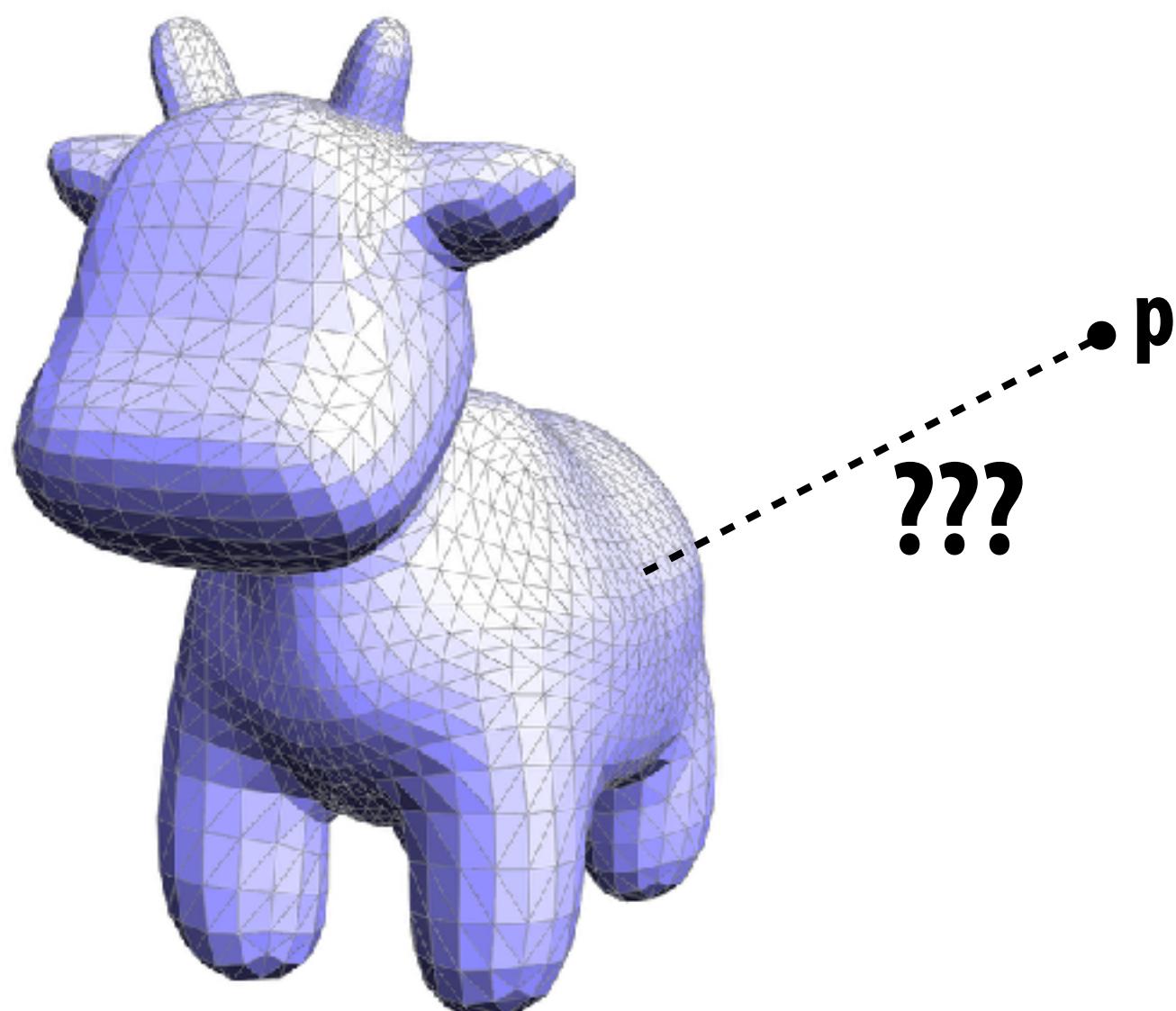
Computer Graphics
CMU 15-462/15-662

Geometric Queries—Motivation



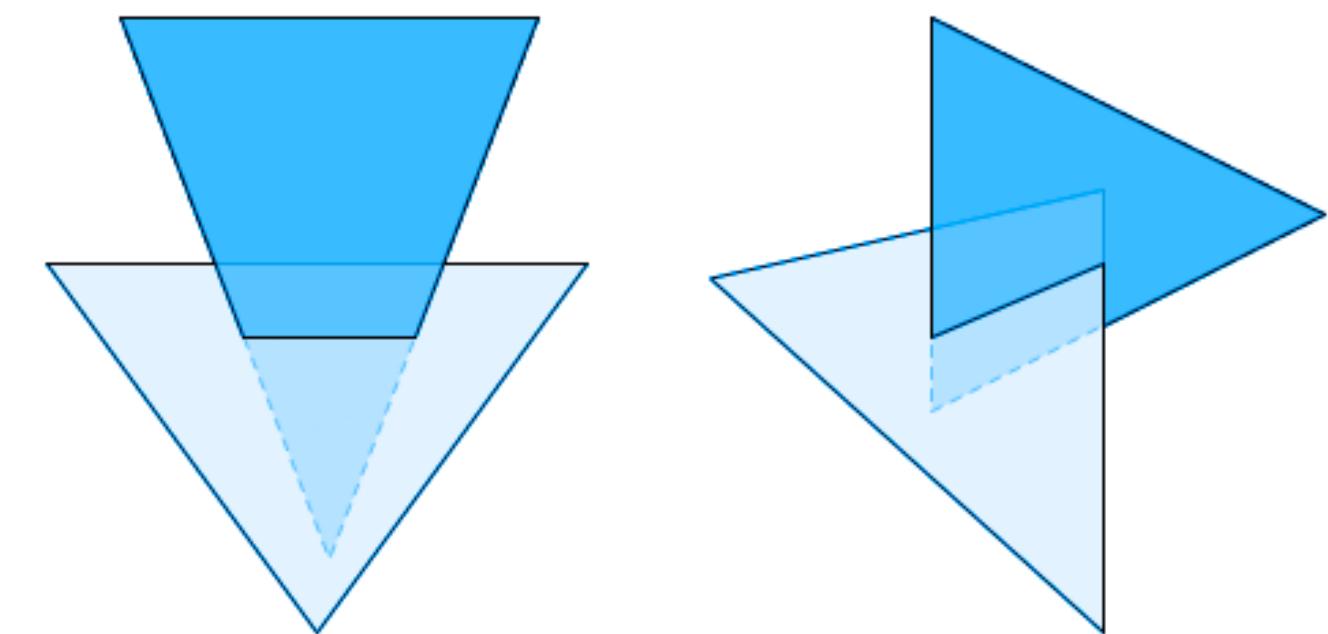
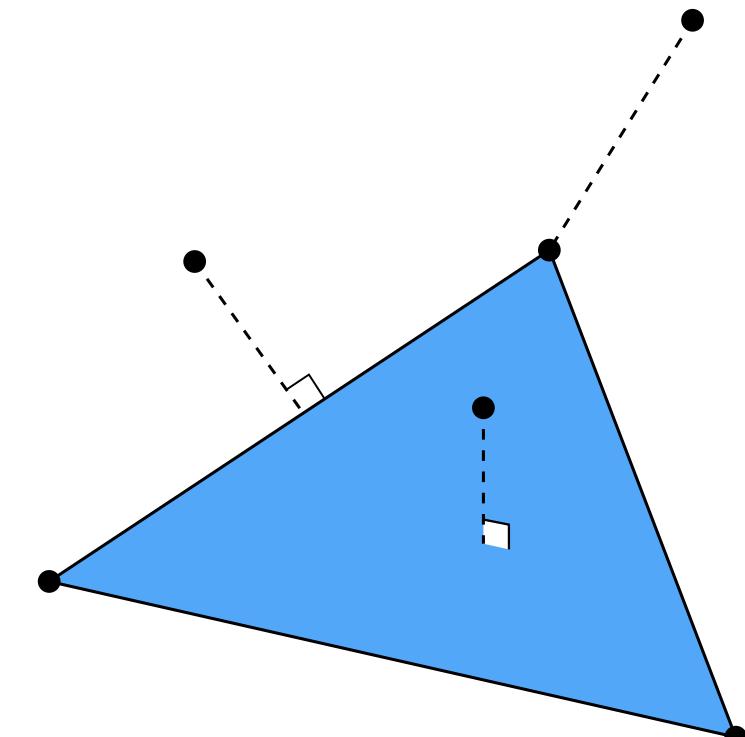
Closest Point Queries

- Q: Given a point, in space (e.g., a new sample point), how do we find the closest point on a given surface?
- Q: Does implicit/explicit representation make this easier?
- Q: Does our halfedge data structure help?
- Q: What's the cost of the naïve algorithm?
- Q: How do we find the distance to a single triangle anyway?
- So many questions!



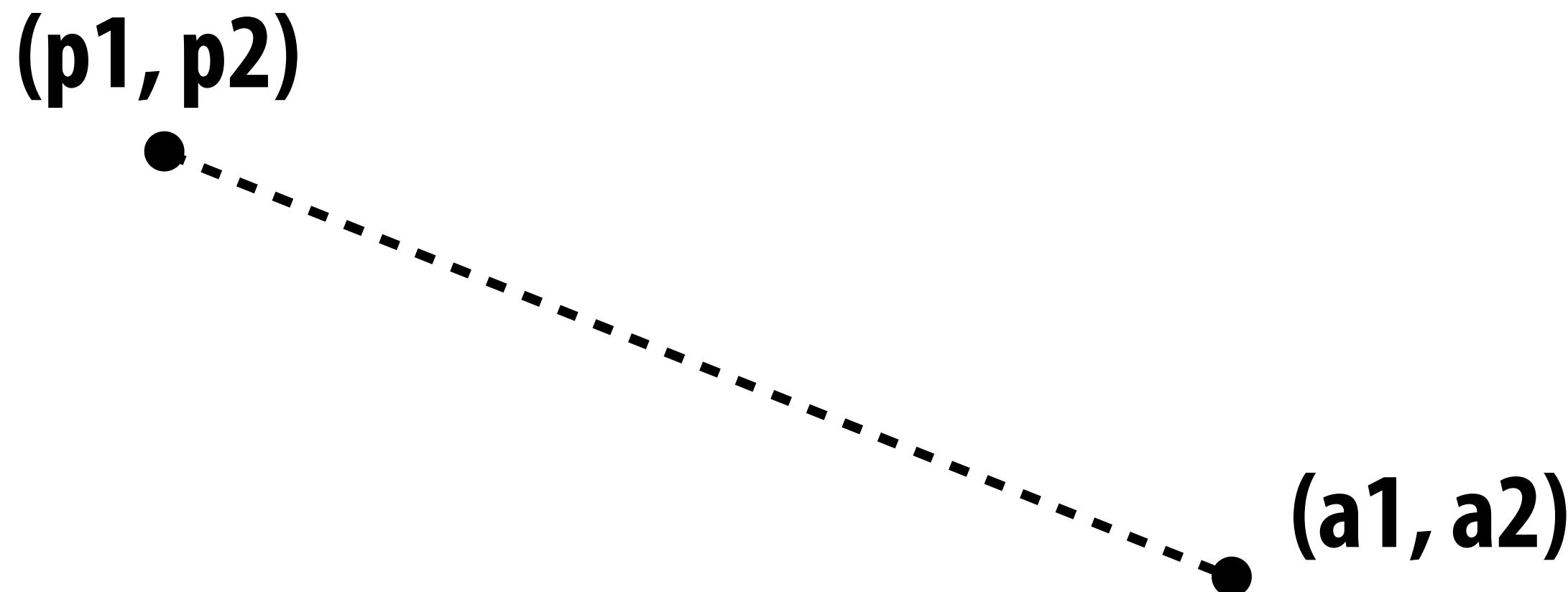
Many types of geometric queries

- Already identified need for “closest point” query
- Plenty of other things we might like to know:
 - Do two triangles intersect?
 - Are we inside or outside an object?
 - Does one object contain another?
 - ...
- Data structures we’ve seen so far not really designed for this...
- Need some new ideas!
- TODAY: come up with simple (read: slow) algorithms.
- NEXT TIME: intelligent ways to accelerate geometric queries.



Warm up: closest point on point

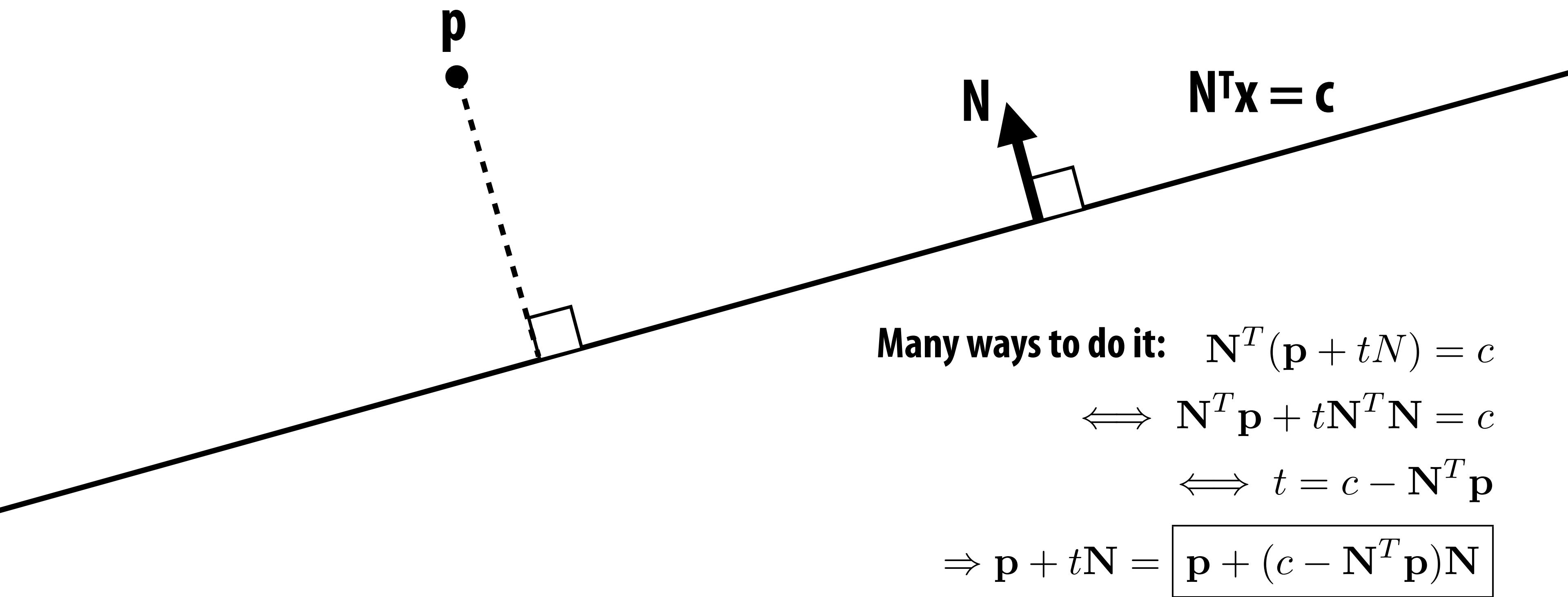
- Goal is to find the point on a mesh closest to a given point.
- *Much simpler question:* given a query point (p_1, p_2) , how do we find the closest point on the point (a_1, a_2) ?



Bonus question: what's the distance?

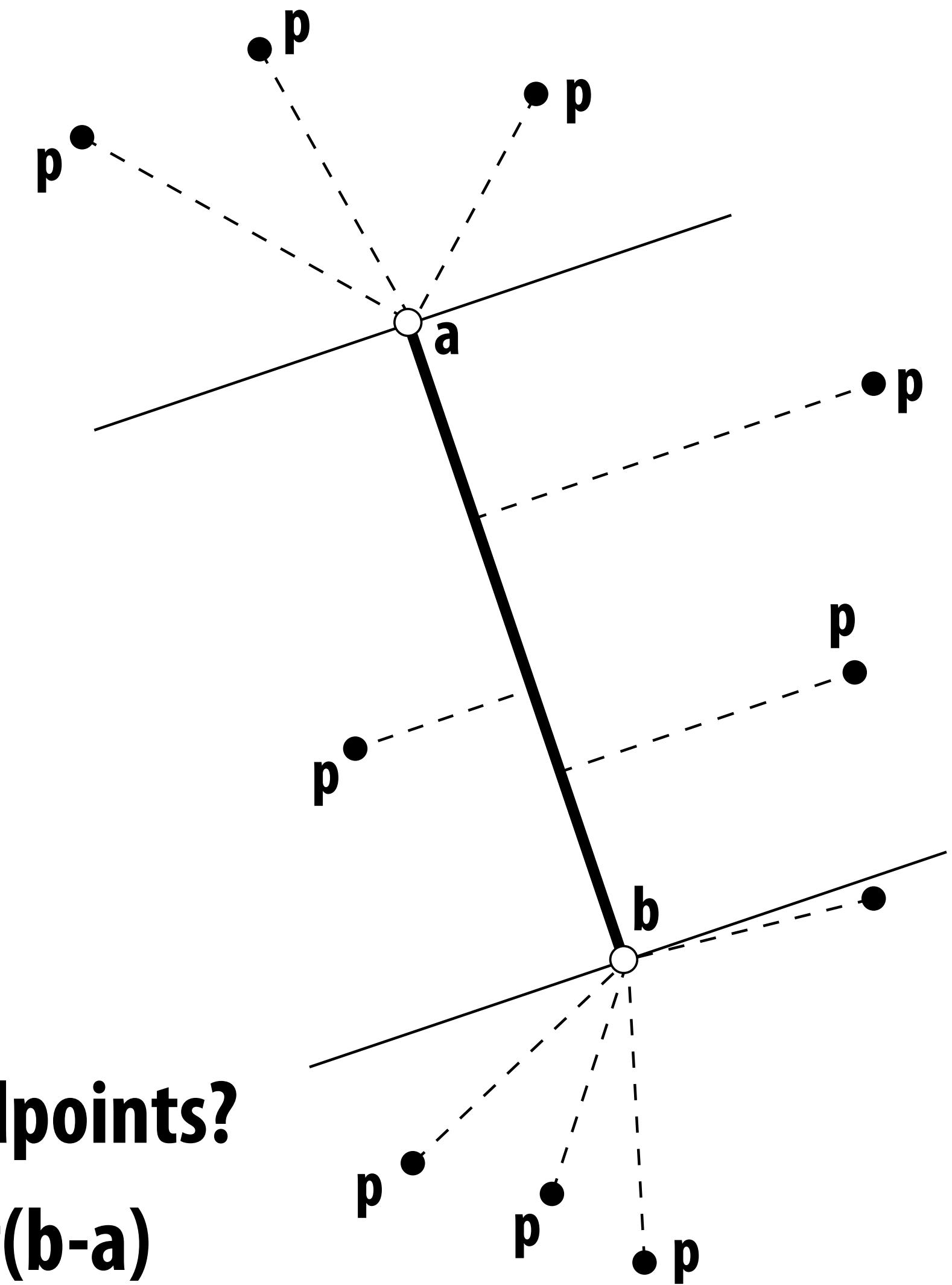
Slightly harder: closest point on line

- Now suppose I have a line $\mathbf{N}^T \mathbf{x} = c$, where \mathbf{N} is the unit normal
- How do I find the point closest to my query point \mathbf{p} ?



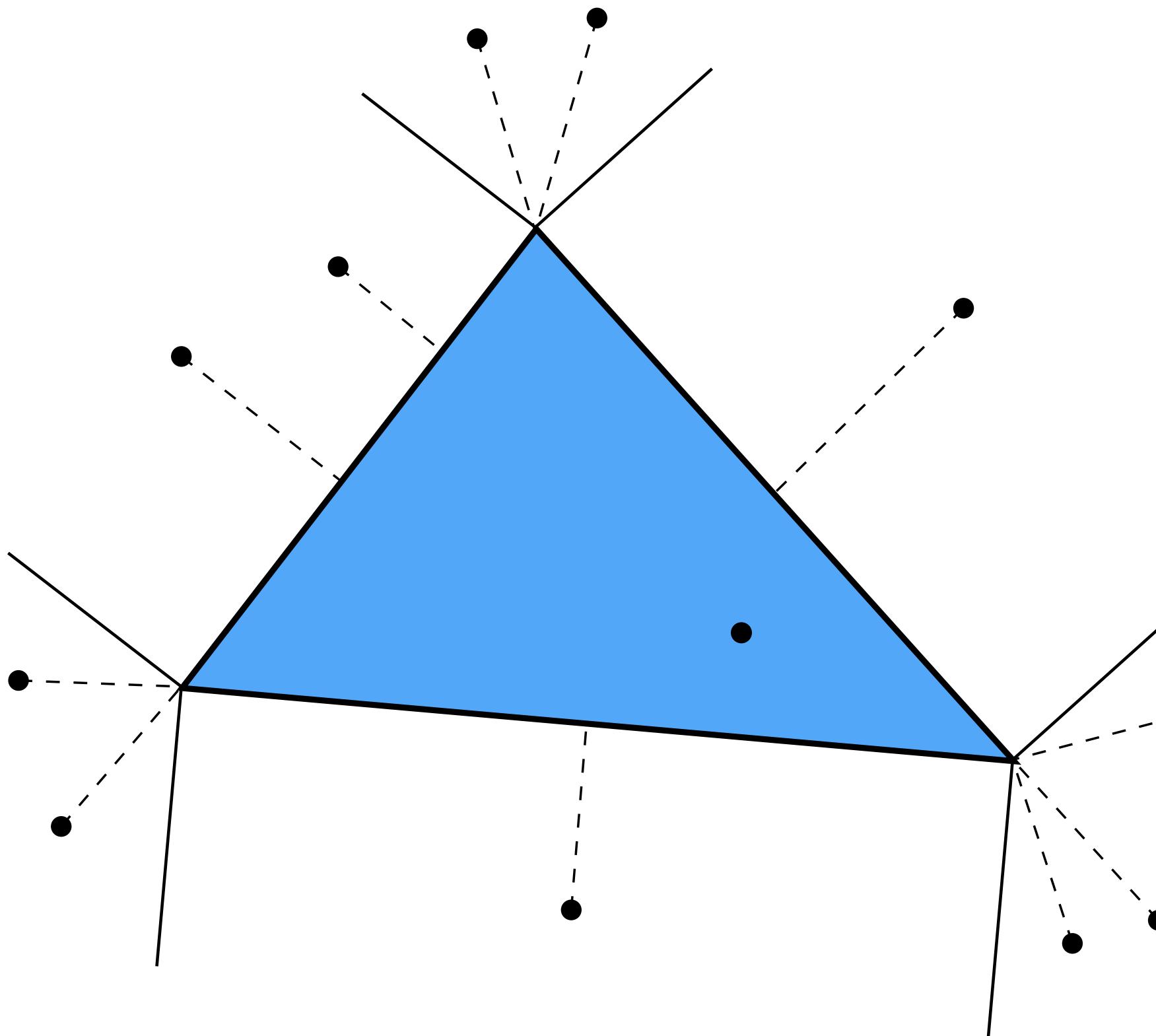
Harder: closest point on line segment

- Two cases: endpoint or interior
- Already have basic components:
 - point-to-point
 - point-to-line
- Algorithm?
 - find closest point on line
 - check if it's between endpoints
 - if not, take closest endpoint
- How do we know if it's between endpoints?
 - write closest point on line as $a+t(b-a)$
 - if t is between 0 and 1, it's inside the segment!



Even harder: closest point on triangle

- What are all the possibilities for the closest point?
- Almost just minimum distance to three segments:



Q: What about a point inside the triangle?

Closest point on triangle in 3D

- Not so different from 2D case
- Algorithm?
 - project onto plane of triangle
 - use half-space tests to classify point (vs. half plane)
 - if inside the triangle, we're done!
 - otherwise, find closest point on associated vertex or edge
- By the way, how do we find closest point on plane?
- Same expression as closest point on a line!
- E.g., $p + (c - N^T p) N$

Closest point on triangle mesh in 3D?

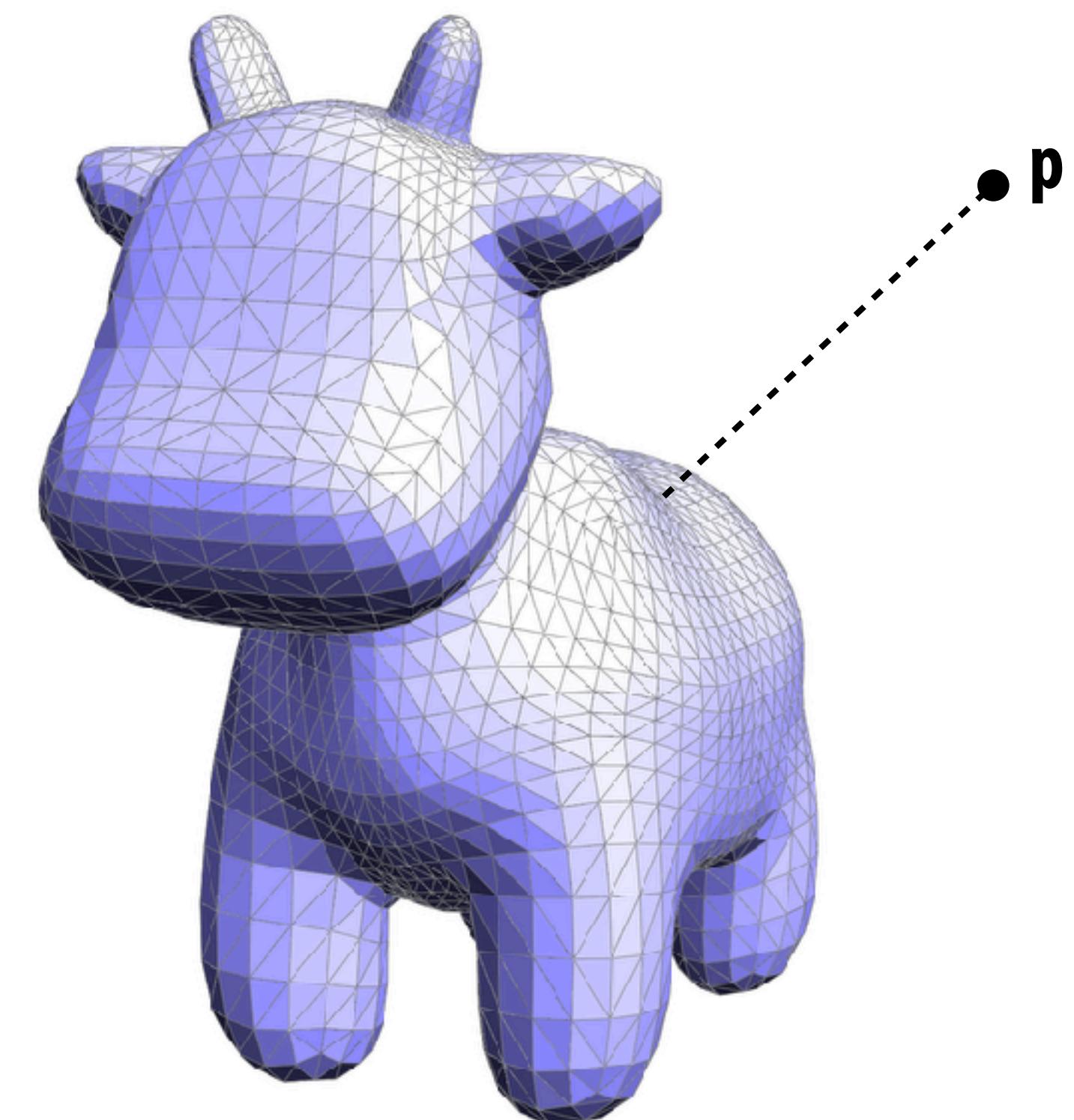
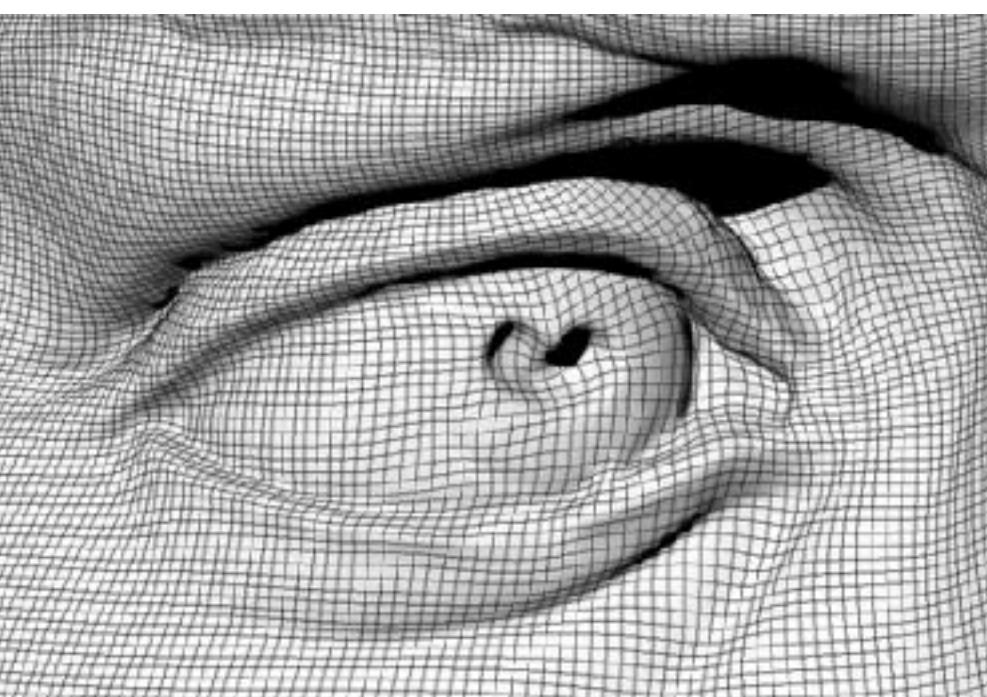
- Conceptually easy:

- loop over all triangles
 - compute closest point to current triangle
 - keep globally closest point

- Q: What's the cost?

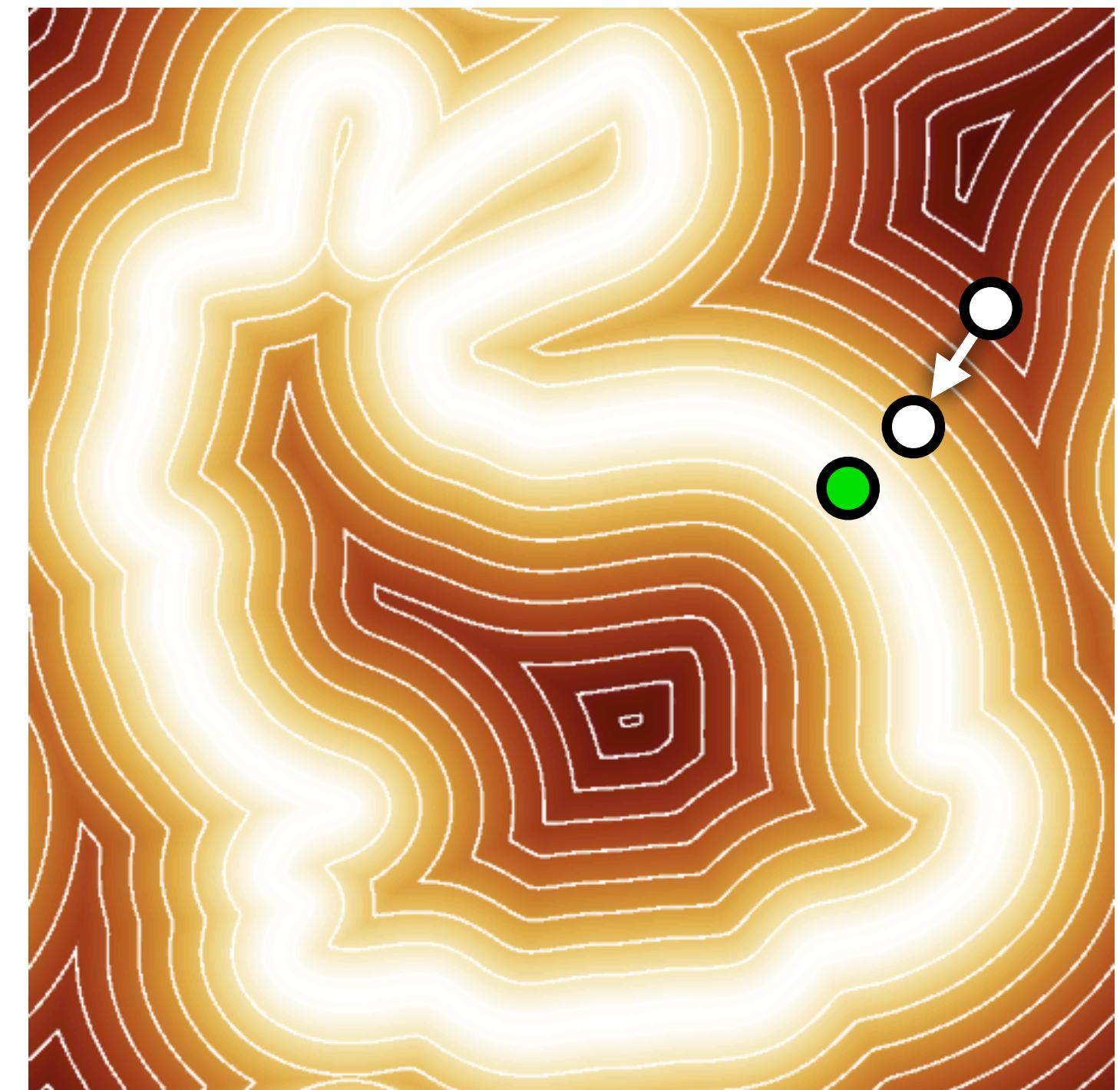
- What if we have *billions* of faces?

- NEXT TIME: Better data structures!



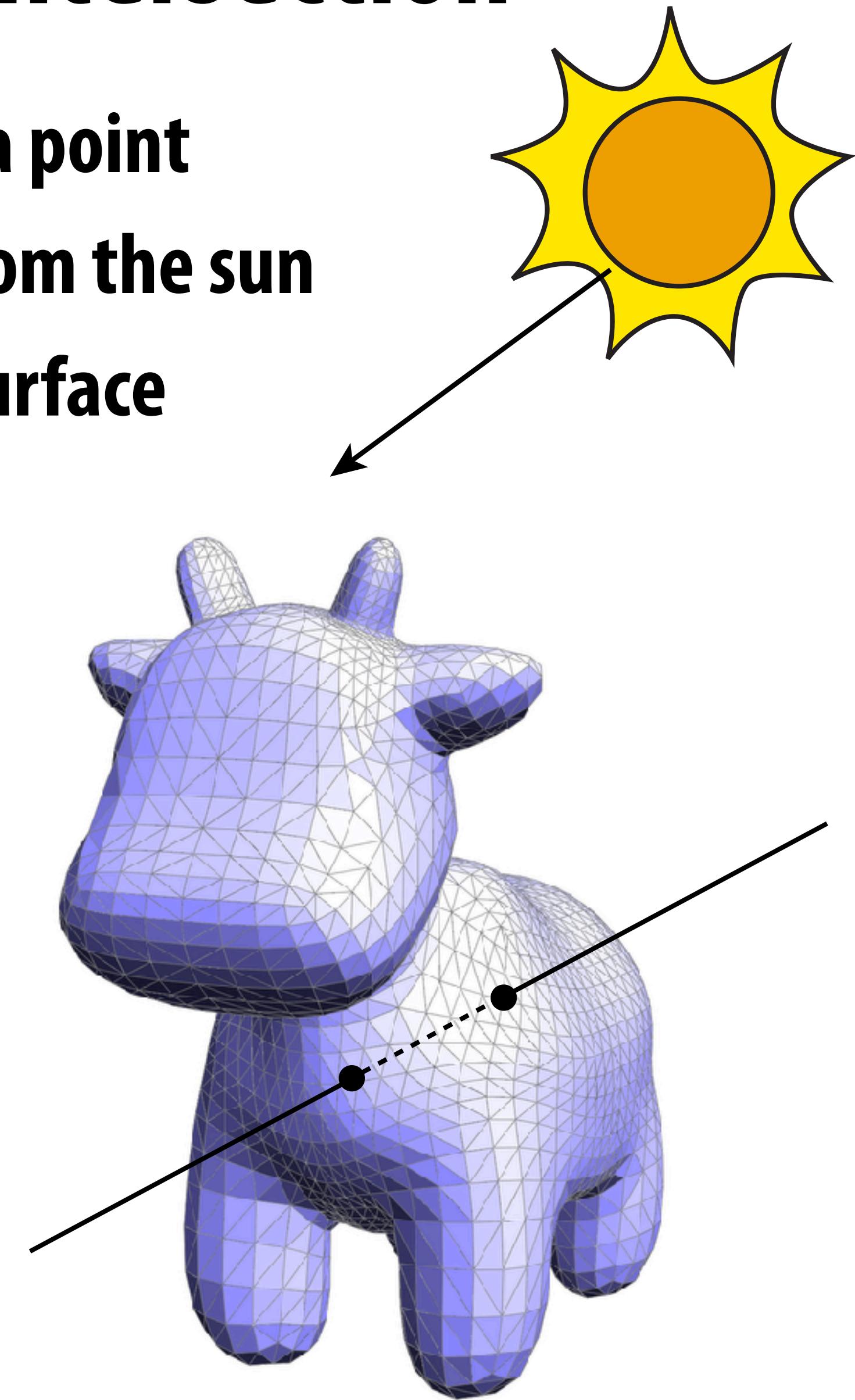
Closest point to *implicit* surface?

- If we change our representation of geometry, algorithms can change completely
- E.g., how might we compute the closest point on an implicit surface described via its distance function?
- One idea:
 - start at the query point
 - compute gradient of distance (using, e.g., finite differences)
 - take a little step (decrease distance)
 - repeat until we're at the surface (zero distance)
- Better yet: just store closest point for each grid cell! (speed/memory trade off)



Different query: ray-mesh intersection

- A “ray” is an oriented line starting at a point
- Think about a ray of light traveling from the sun
- Want to know where a ray pierces a surface
- Why?
 - GEOMETRY: inside-outside test
 - RENDERING: visibility, ray tracing
 - ANIMATION: collision detection
- Might pierce surface in many places!



Ray equation

- Can express ray as

$$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$$

Diagram illustrating the Ray equation:

- point along ray**: A red arrow pointing from the origin \mathbf{o} towards the right.
- “time”**: A red arrow pointing upwards from the origin \mathbf{o} .
- origin**: A red arrow pointing downwards from the term \mathbf{o} .
- unit direction**: A red arrow pointing towards the sun, indicating the direction of the ray.

The diagram shows a sun-like object with a center point labeled \mathbf{o} . A solid black line segment extends from \mathbf{o} through a point on the sun's surface, labeled \mathbf{d} . A dashed black line segment extends further from \mathbf{d} , labeled $\mathbf{r}(t)$, representing the ray equation.

Intersecting a ray with an implicit surface

- Recall implicit surfaces: all points x such that $f(x) = 0$
- Q: How do we find points where a ray pierces this surface?
- Well, we know all points along the ray: $r(t) = o + td$
- Idea: replace “ x ” with “ r ” in 1st equation, and solve for t
- Example: unit sphere

$$f(x) = |x|^2 - 1$$

quadratic formula:

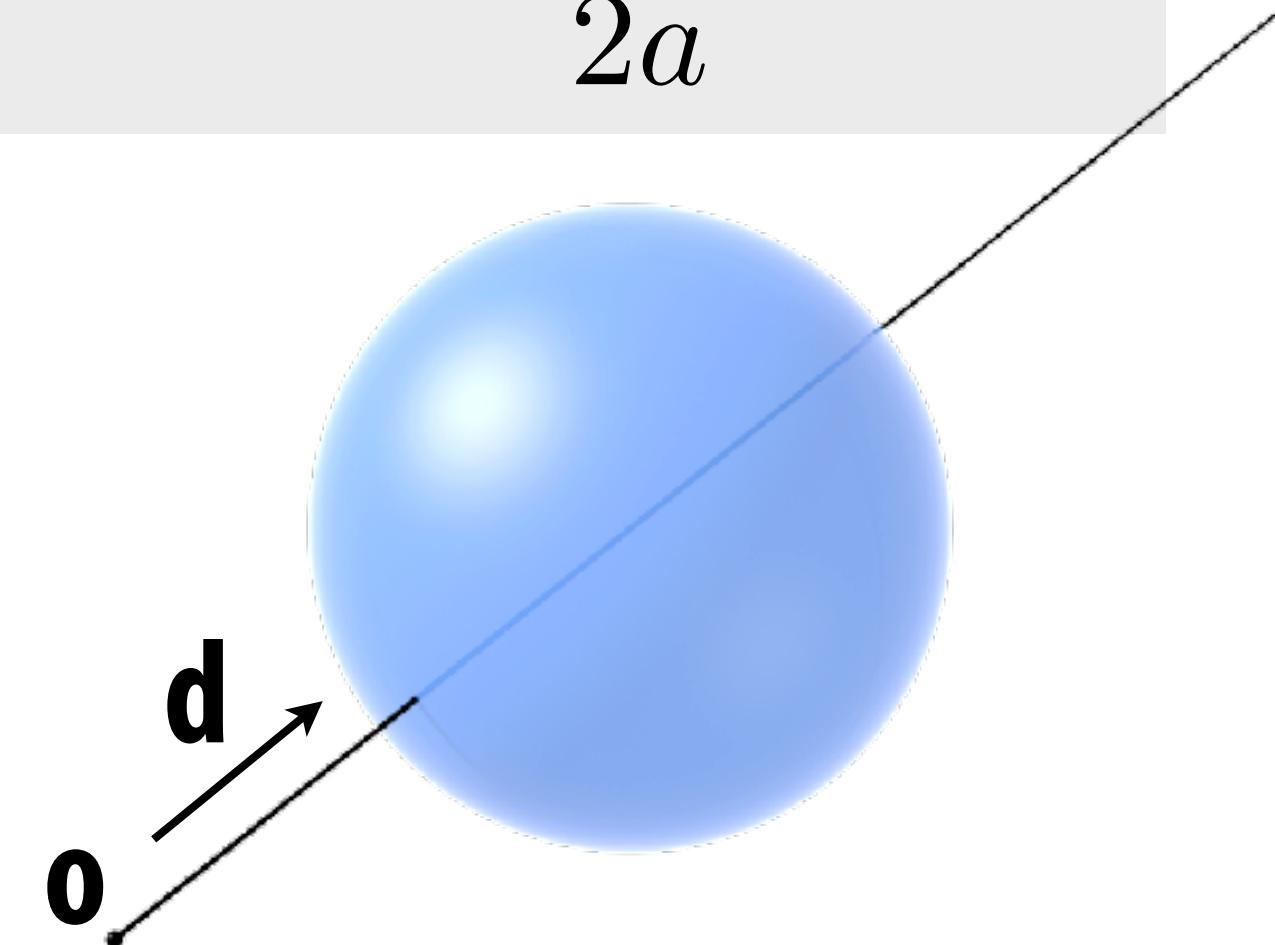
$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$\Rightarrow f(r(t)) = |o + td|^2 - 1$$

$$\underbrace{|d|^2 t^2}_a + \underbrace{2(o \cdot d)t}_b + \underbrace{|o|^2 - 1}_c = 0$$

$$t = \boxed{-o \cdot d \pm \sqrt{(o \cdot d)^2 - |o|^2 + 1}}$$

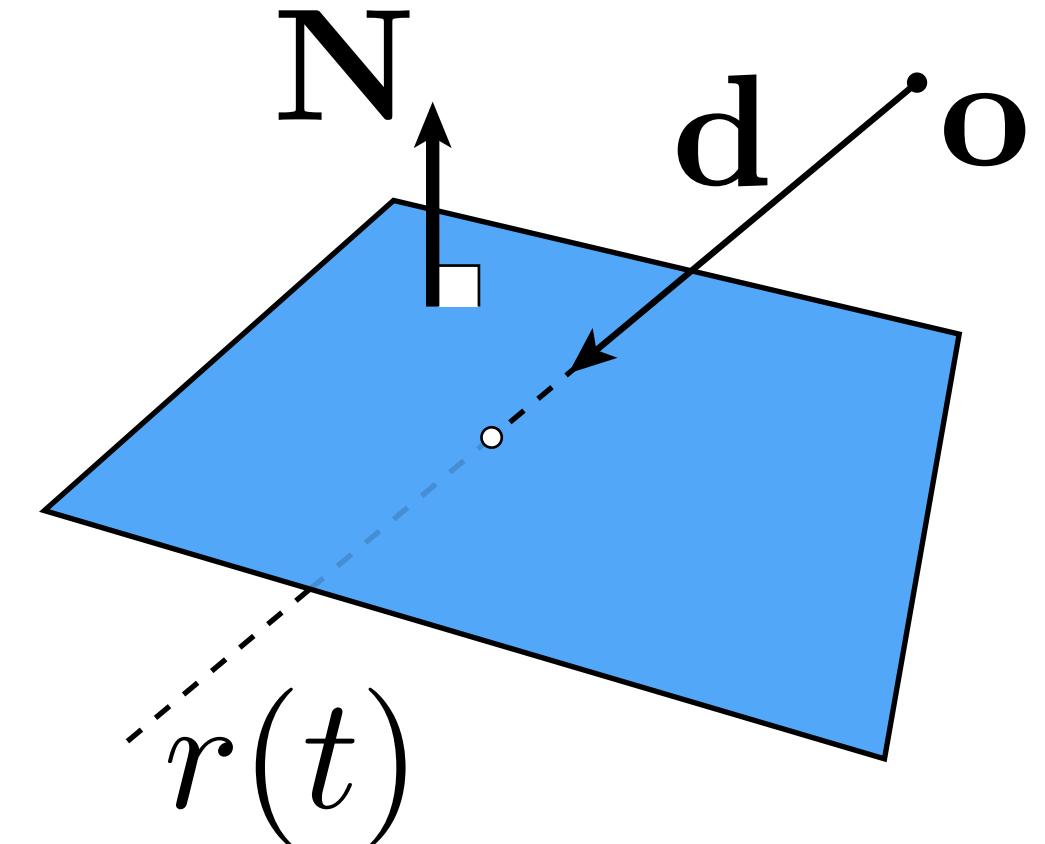
Why two solutions?



Ray-plane intersection

- Suppose we have a plane $\mathbf{N}^T \mathbf{x} = c$

- \mathbf{N} - unit normal
 - c - offset



- How do we find intersection with ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$?
- *Key idea:* again, replace the point \mathbf{x} with the ray equation t :

$$\mathbf{N}^T \mathbf{r}(t) = c$$

- Now solve for t :

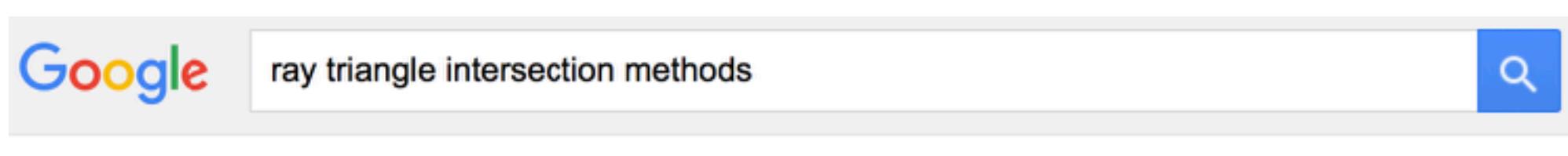
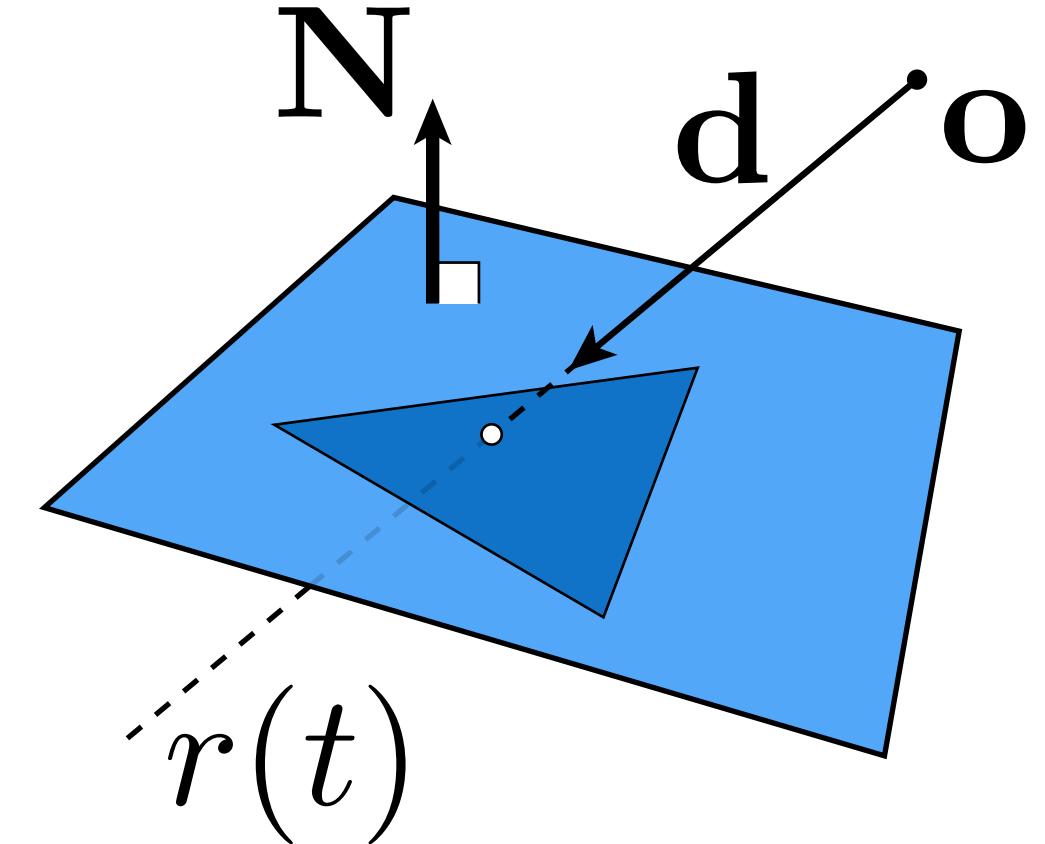
$$\mathbf{N}^T(\mathbf{o} + t\mathbf{d}) = c \quad \Rightarrow \quad t = \frac{c - \mathbf{N}^T \mathbf{o}}{\mathbf{N}^T \mathbf{d}}$$

- And plug t back into ray equation:

$$\mathbf{r}(t) = \mathbf{o} + \frac{c - \mathbf{N}^T \mathbf{o}}{\mathbf{N}^T \mathbf{d}} \mathbf{d}$$

Ray-triangle intersection

- Triangle is in a plane...
- Not much more to say!
 - Compute ray-plane intersection
 - Q: What do we do now?
 - A: Why not compute barycentric coordinates of hit point?
 - If barycentric coordinates are all positive, point in triangle
- Actually, a *lot* more to say... if you care about performance!



Web Shopping Videos News Images More Search tools

About 443,000 results (0.44 seconds)

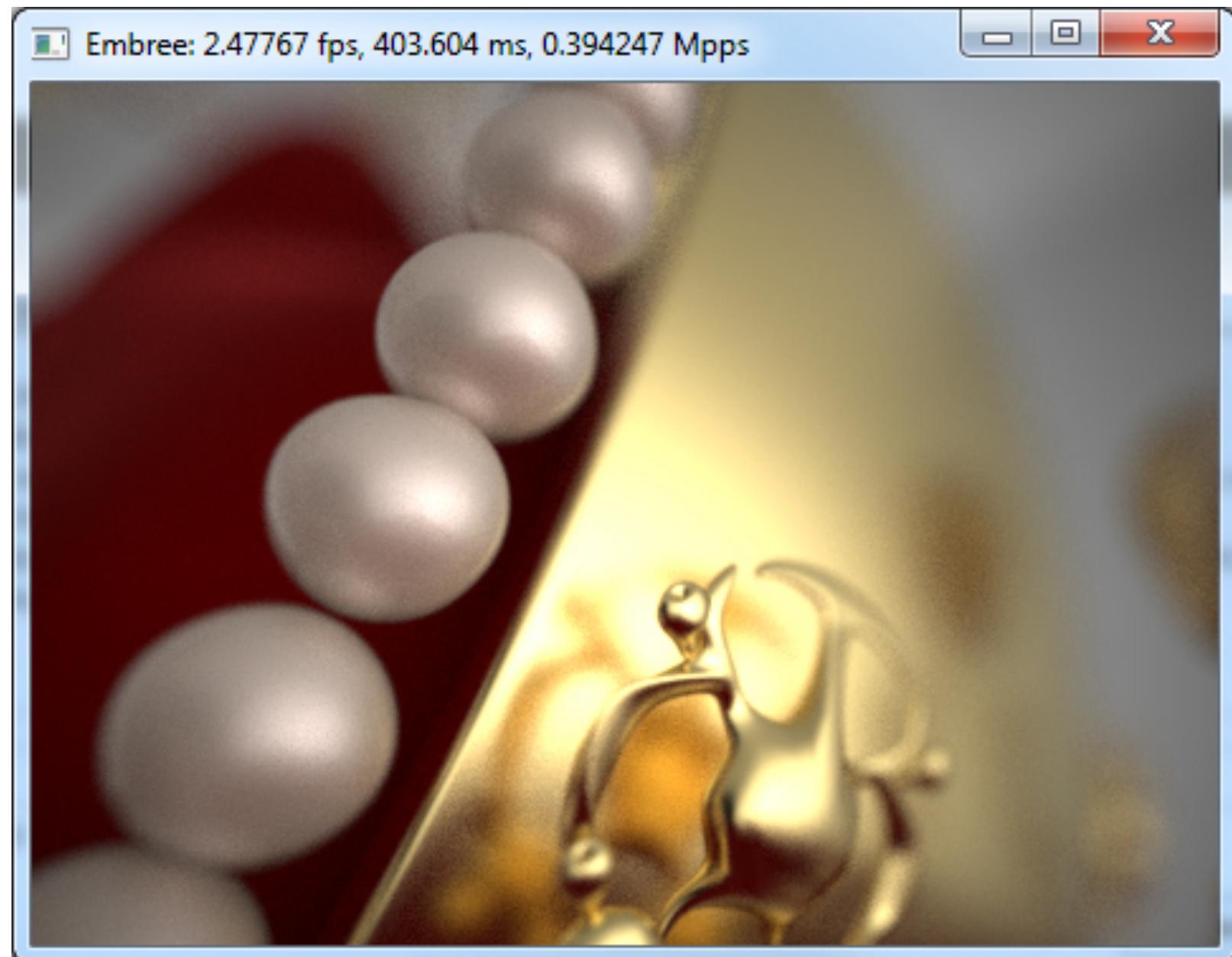
[Möller–Trumbore intersection algorithm - Wikipedia, the free ...](https://en.wikipedia.org/w/index.php?title=M%C3%B6ller–Trumbore_intersection_algorithm&oldid=5208111)
https://en.wikipedia.org/w/index.php?title=M%C3%B6ller–Trumbore_intersection_algorithm&oldid=5208111 ▾ Wikipedia ▾
The Möller–Trumbore ray-triangle intersection algorithm, named after its inventors Tomas Möller and Ben Trumbore, is a fast method for calculating the ...

[\[PDF\] Fast Minimum Storage Ray-Triangle Intersection.pdf](https://www.cs.virginia.edu/~pcab/pubs/Fast%20MinimumStorage.pdf)
<https://www.cs.virginia.edu/~pcab/pubs/Fast%20MinimumStorage.pdf> ▾ University of Virginia ▾
by PC AB - Cited by 650 - Related articles
We present a clean algorithm for determining whether a ray intersects a triangle. ... ble in speed to previous methods, we believe it is the fastest ray/triangle.

[\[PDF\] Optimizing Ray-Triangle Intersection via Automated Search](http://www.cs.utah.edu/~aek/research/triangle.pdf)
www.cs.utah.edu/~aek/research/triangle.pdf ▾ University of Utah ▾
by A Kensler - Cited by 33 - Related articles
method is used to further optimize the code produced via the fitness function. ... For these 3D methods we optimize ray-triangle intersection in two different ways.

[\[PDF\] Comparative Study of Ray-Triangle Intersection Algorithms](http://www.graphicon.ru/html/proceedings/2012/.../gc2012Shumskiy.pdf)
www.graphicon.ru/html/proceedings/2012/.../gc2012Shumskiy.pdf ▾
by V Shumskiy - Cited by 1 - Related articles
optimized SIMD ray-triangle intersection method evaluated on. GPU for path- tracing

Why care about performance?



Intel Embree



NVIDIA OptiX

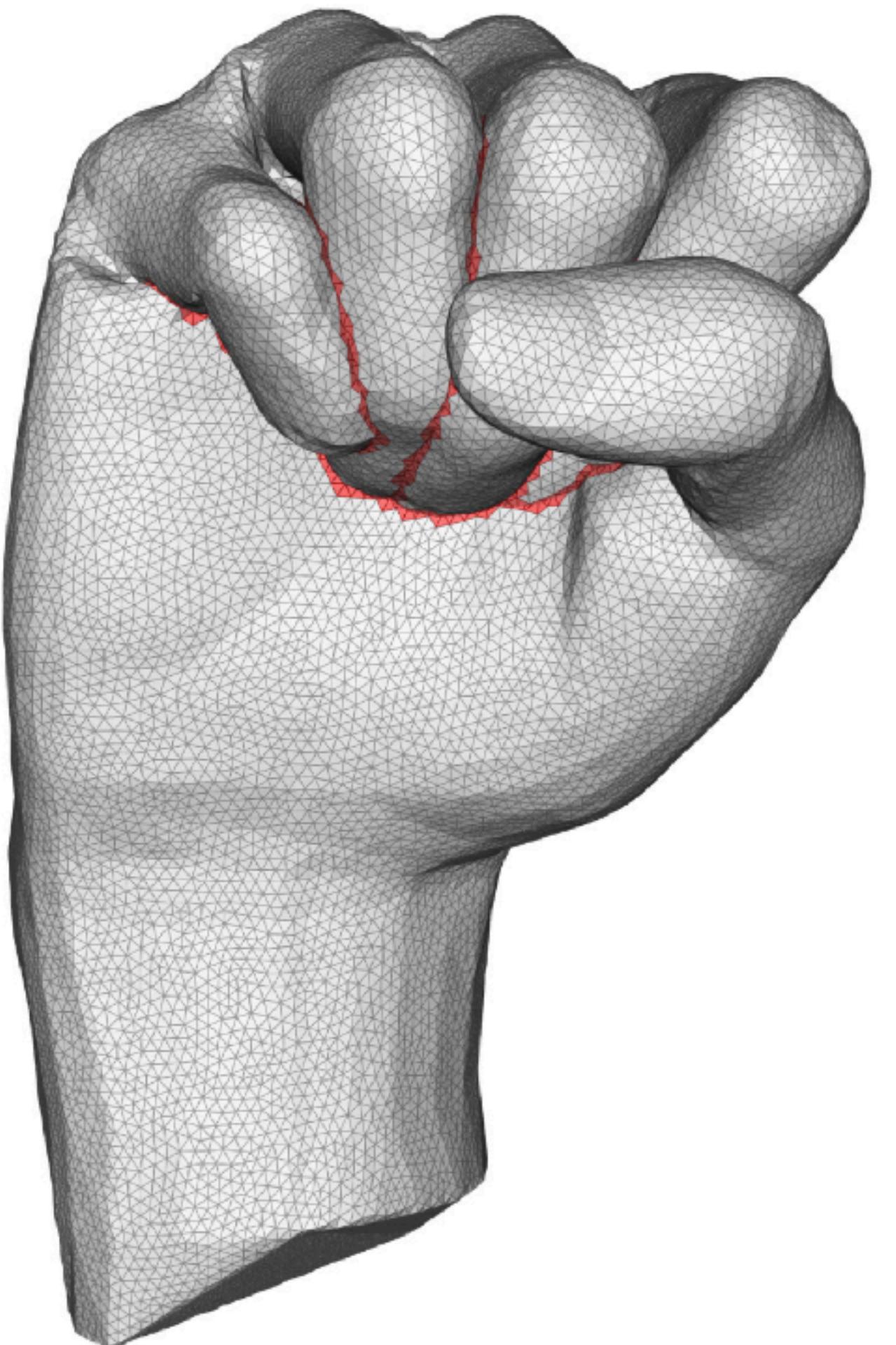
Why care about performance?



“Brigade 3” real time path tracing demo

One more query: mesh-mesh intersection

- **GEOMETRY:** How do we know if a mesh intersects itself?
- **ANIMATION:** How do we know if a collision occurred?



Warm up: point-point intersection

- Q: How do we know if p intersects a?
- A: ...check if they're the same point!

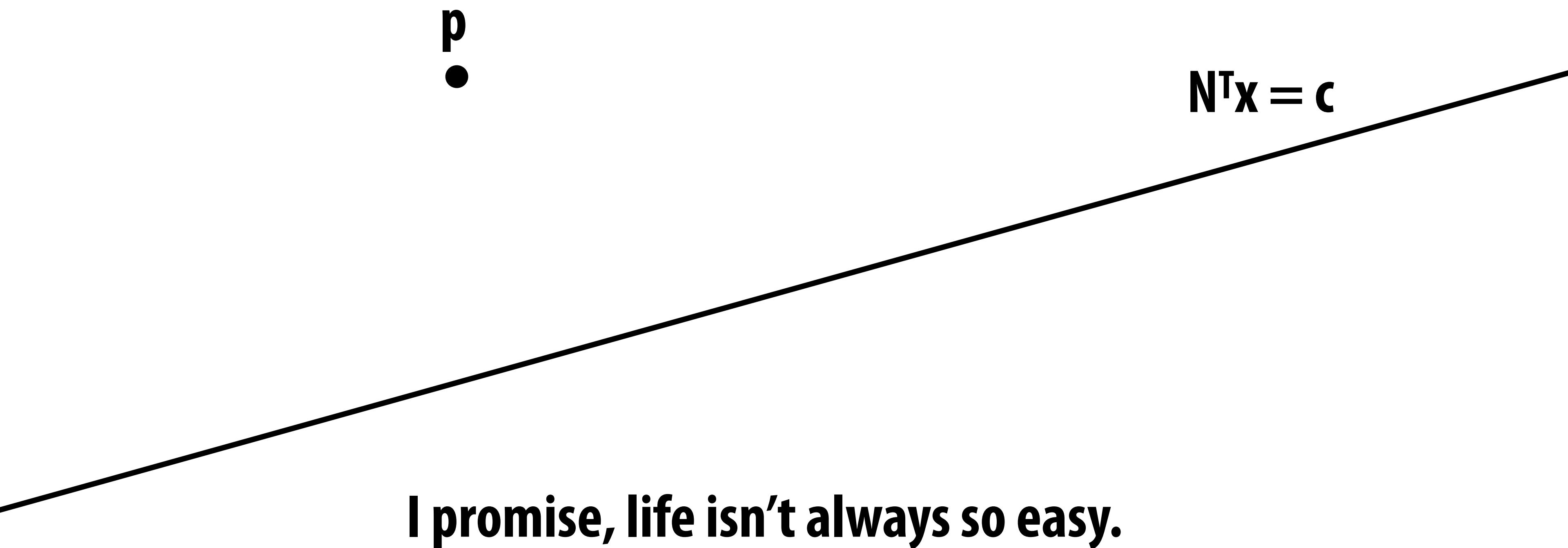
(p_1, p_2)
•

• (a_1, a_2)

Sadly, life is not always so easy.

Slightly harder: point-line intersection

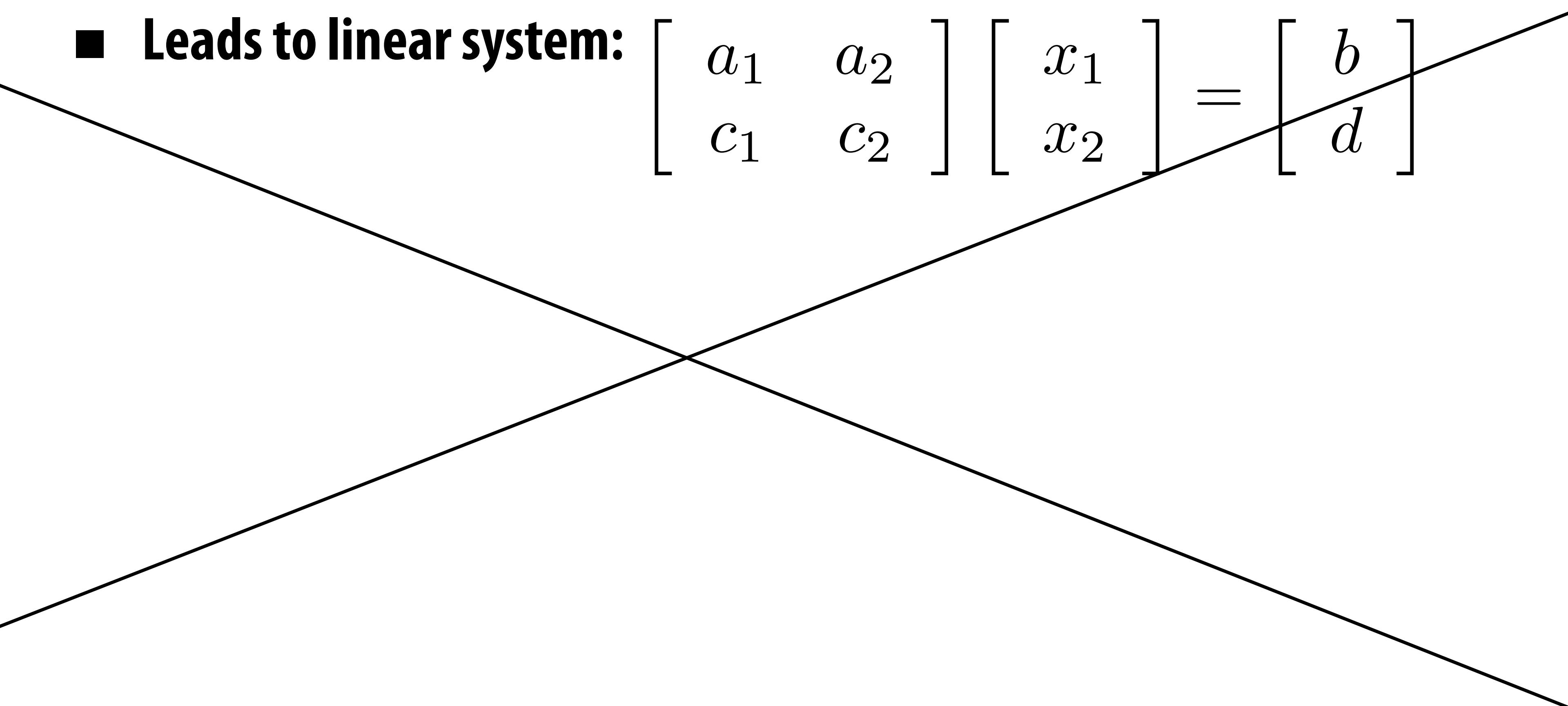
- Q: How do we know if a point intersects a given line?
- A: ...plug it into the line equation!



Finally interesting: line-line intersection

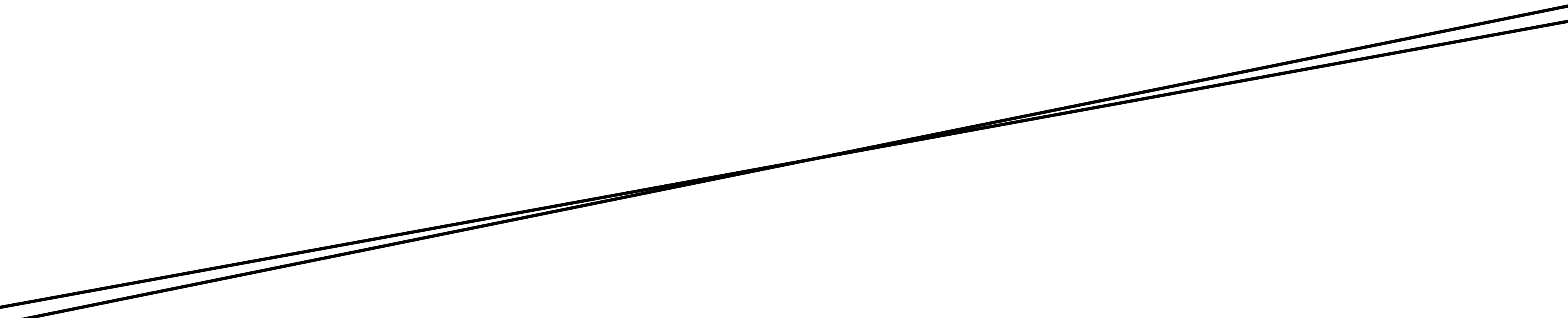
- Two lines: $ax=b$ and $cx=d$
- Q: How do we find the intersection?
- A: See if there is a simultaneous solution
- Leads to linear system:

$$\begin{bmatrix} a_1 & a_2 \\ c_1 & c_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix}$$



Degenerate line-line intersection?

- What if lines are almost parallel?
- Small change in normal can lead to big change in intersection!
- Instability very common, very important with geometric predicates. Demands special care (e.g., analysis of matrix).



Triangle-Triangle Intersection?

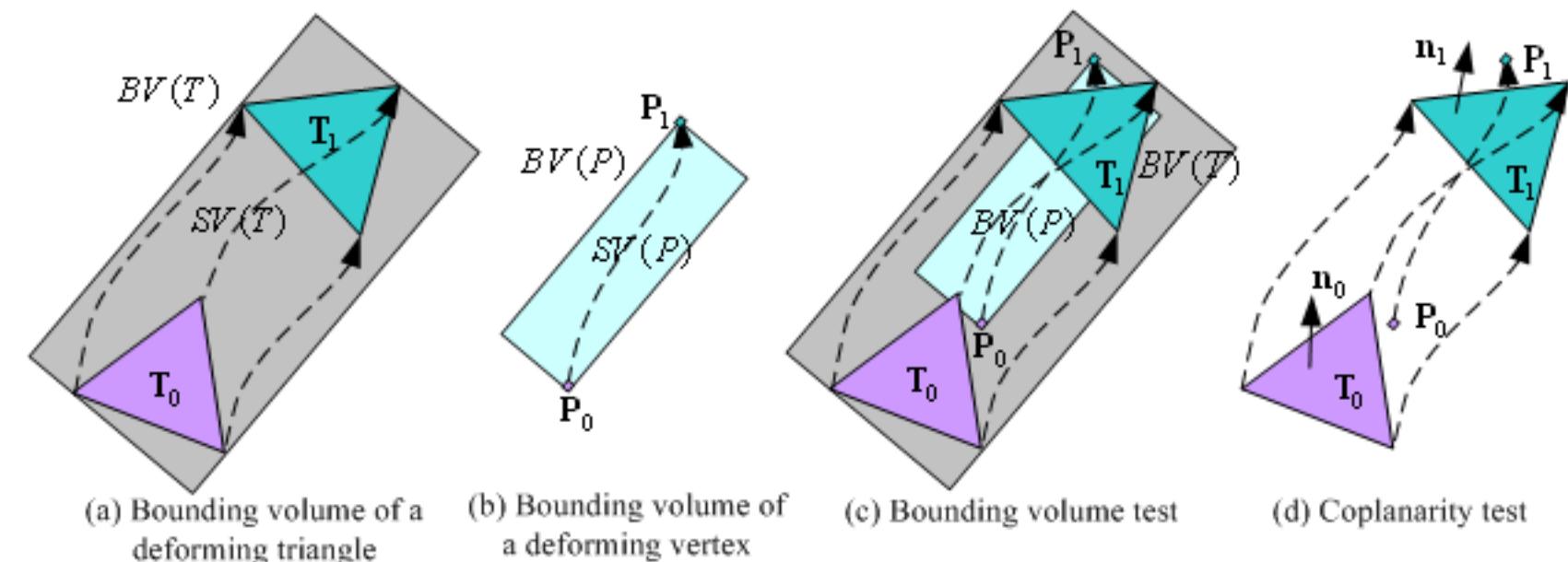
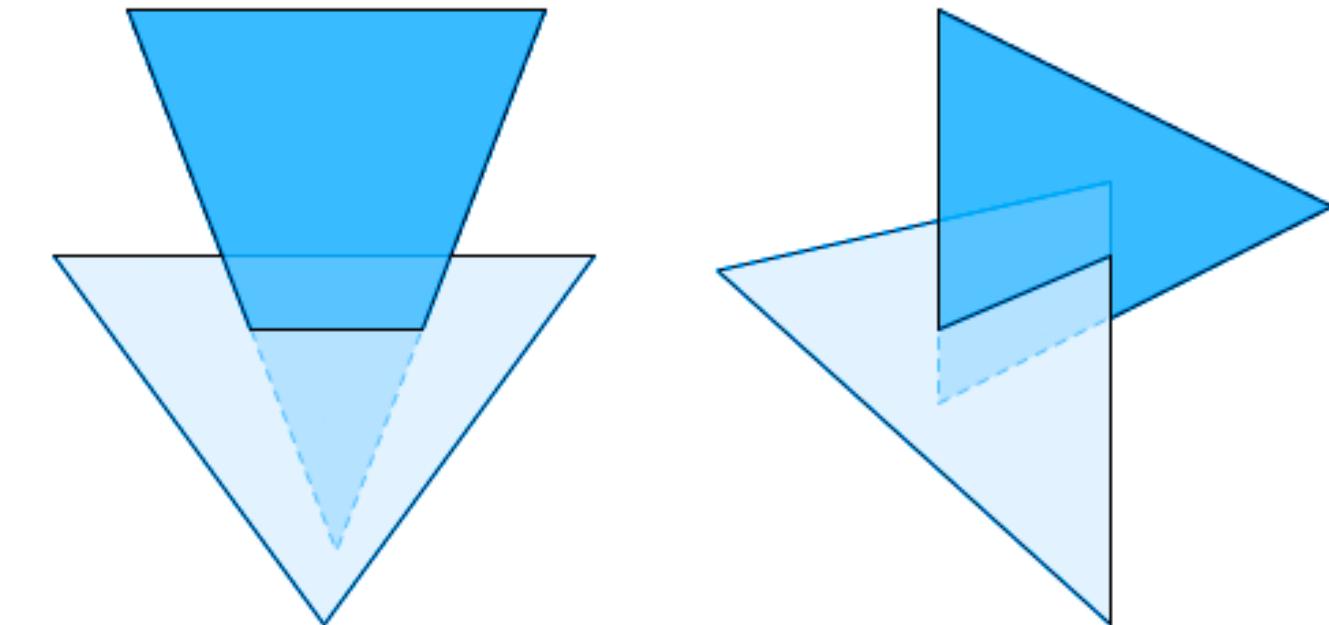
- Lots of ways to do it

- Basic idea:

- Q: Any ideas?
- One way: reduce to edge-triangle intersection
- Check if each line passes through plane
- Then do interval test

- What if triangle is *moving*?

- Important case for animation
- Can think of triangles as *prisms* in time
- Turns dynamic problem ($nD + \text{time}$) into purely geometric problem in $(n+1)$ -dimensions



Up Next: Spatial Acceleration Data Structures

- Testing every element is *slow!*
- E.g., linearly scanning through a list vs. binary search
- Can apply this same kind of thinking to geometric queries

