



Lecture 13: 几何查询

SSE315: 计算机图形学
Computer Graphics

陈壮彬

软件工程学院

chenzhb36@mail.sysu.edu.cn

Course roadmap

光栅化 Rasterization

计算机图形学介绍
基于采样的光栅化
空间变换
纹理映射、深度和透明度

几何 Geometry

几何介绍
曲线与曲面
几何处理

材质与光线 Materials and Lighting

动画 Animation



Course roadmap

光栅化 Rasterization

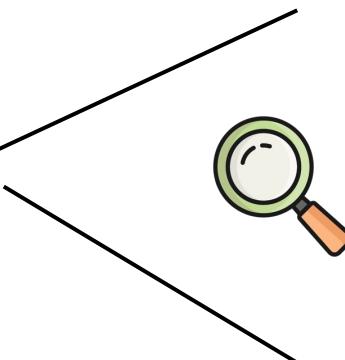
计算机图形学介绍
基于采样的光栅化
空间变换
纹理映射、深度和透明度

几何 Geometry

几何介绍
曲线与曲面
几何处理

材质与光线 Materials and Lighting

光线几何查询



光线几何查询 <

动画 Animation

Today's topics

□ 光线追踪介绍

□ 最近点查询

□ 几何交叉检测

光栅化 vs. 光线追踪

口二者都是计算机图形学中重要的渲染 (rendering) 技术

口光栅化 Rasterization

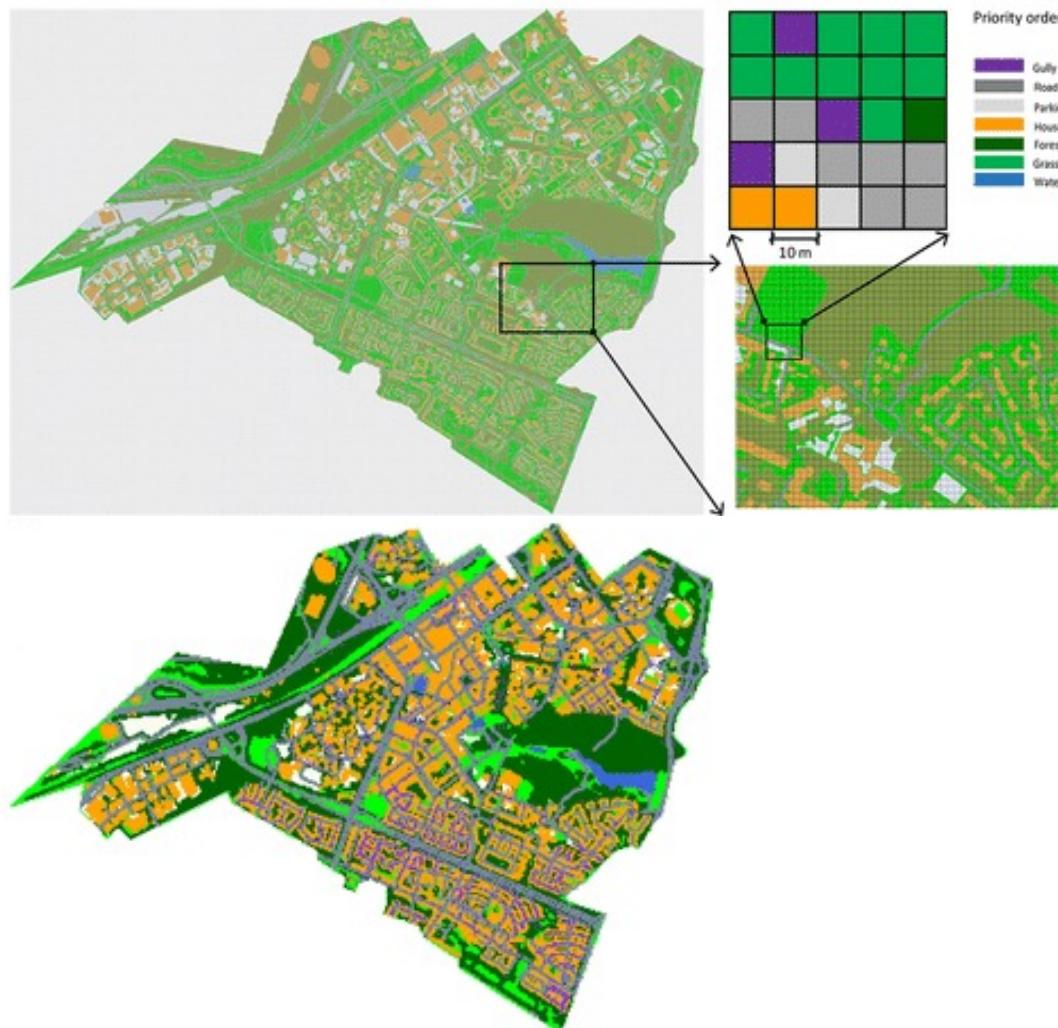
- 对于每个图形基元 (primitive)，如三角形，点亮哪些像素
- 非常快 (在 GPU 中数十亿个三角形每秒)
- 很难表现出逼真的照明和阴影效果，没有考虑到真实世界中光线在曲面反射的方式
- 非常适合 2D vector art, fonts, quick 3D preview

口光线追踪 Ray tracing

- 通过追踪光线穿过图像平面中像素的路径来创建图像
- 模拟光的物理行为，包括反射、折射和散射等，从而产生更逼真的图像，尤其是在涉及照明和阴影的情况下
- 一般来讲很慢

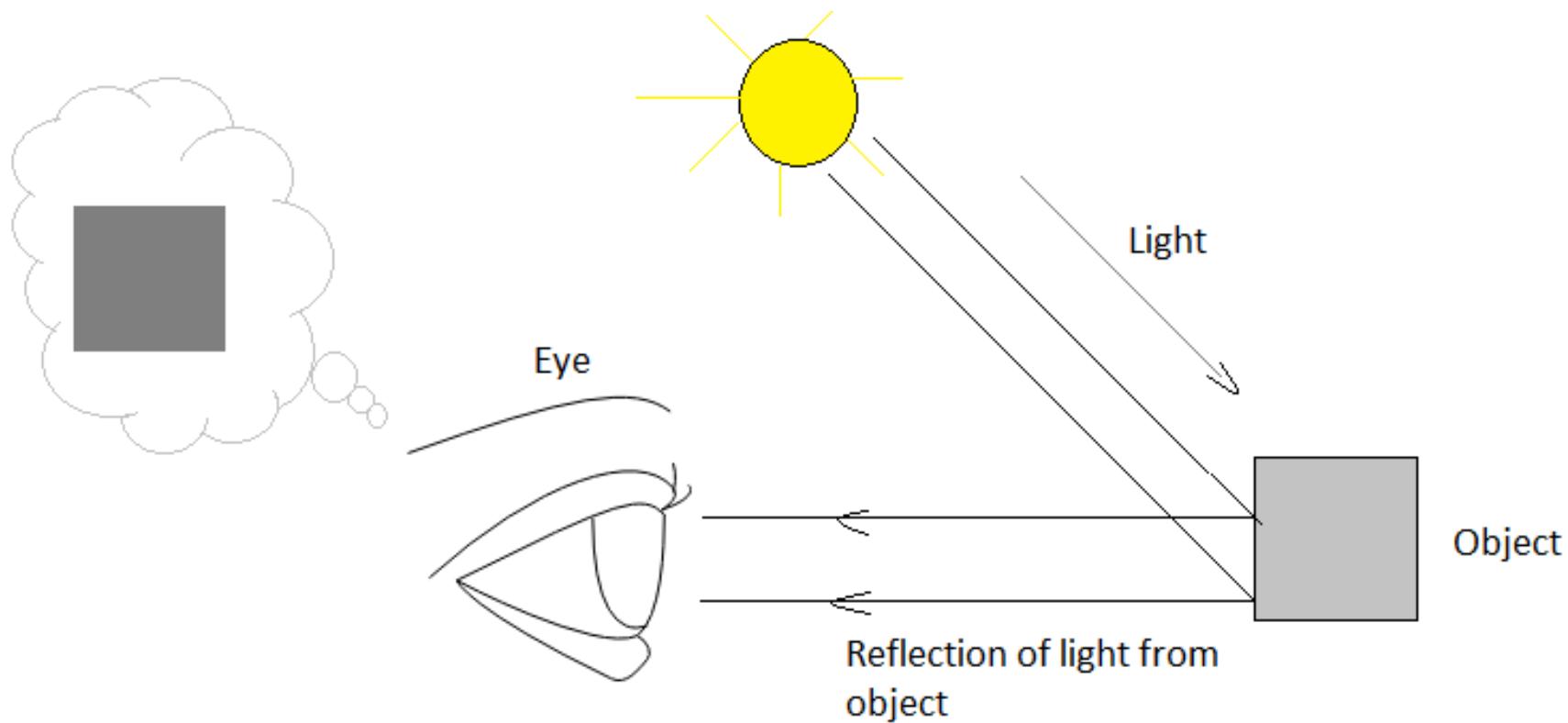
光栅化渲染质量低

□ 光栅化虽然快，但是渲染质量相对较低



我们为什么能看到物体？

□ 物体发出或者反射光线进入我们的眼睛



为什么需要光线追踪？

□ 光栅化无法很好地处理全局效果

- 很难模拟阴影效果
- 特别是当光线弹射超过一次的时候



软阴影
Soft shadow

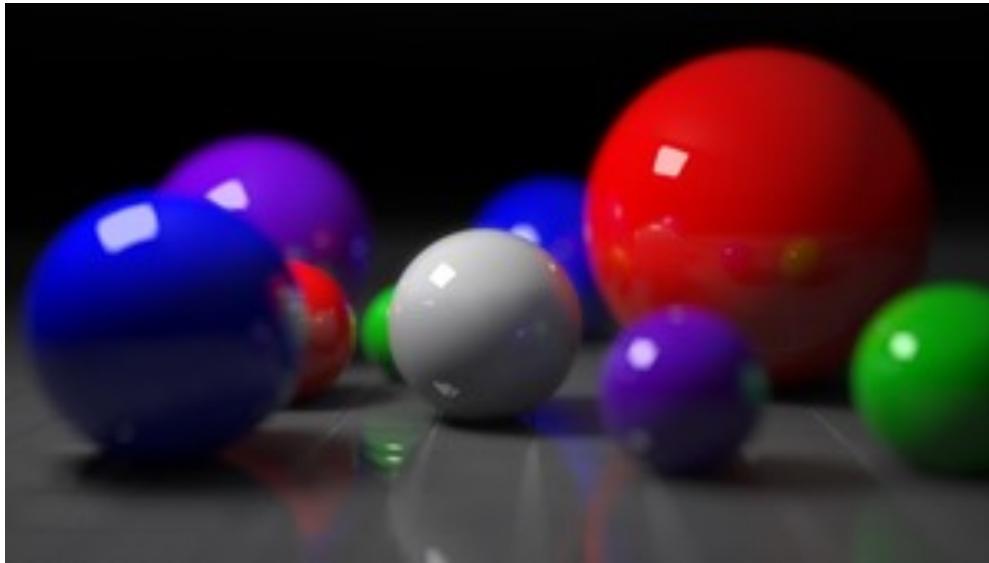


光泽反射
Glossy reflection



间接照明
Indirect illumination

光线追踪生成更真实的物体



光线追踪捕捉环境光



光线追踪在动画中的应用



Walt Disney Animation Studios: "Maestro"

Character and Prop Modeling/Look Dev (The Detective and Flying Chair), Set Dressing, Lighting and Compositing

Maya, Zbrush, Paint 3D, Hyperion, Nuke

为什么需要光线追踪？

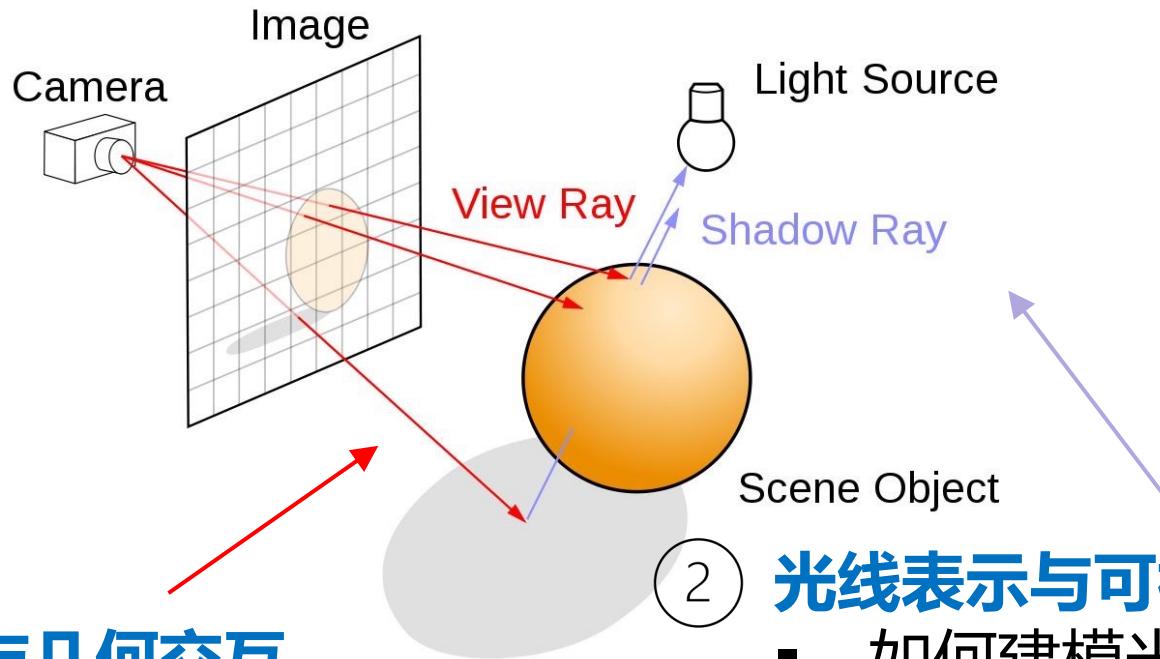
光线追踪虽然准确，但是非常慢

- 光栅化能做到**实时**，光线追踪多是**线下处理**的
- 现实生产中需要 ~10K CPU core hours 来渲染**一帧图片**



光线追踪 Ray tracing

分成 **光线与几何交互** 和 **光线表示与可视化** 两部分



① 光线与几何交互

- 查找光线照射在几何中的位置
- 如何加速上述过程

看到哪里

② 光线表示与可视化

- 如何建模光线
- 光线与物体的相互作用，包括反射、折射和散射等
- 全局光照

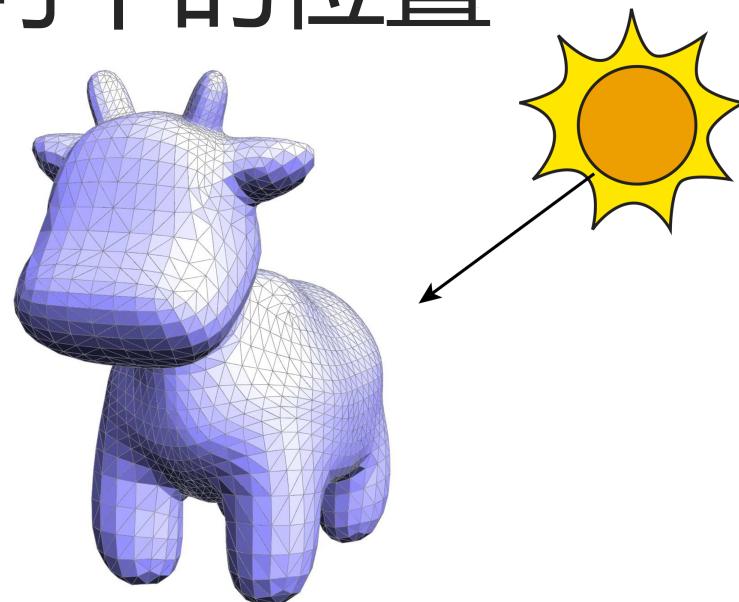
看到什么

查找光线照射在几何中的位置

□(简单) 光线模型

- 从一个光源发出的射线
- 光线沿着既定方向与几何物体发生交叉

□关注光线打在物体哪里

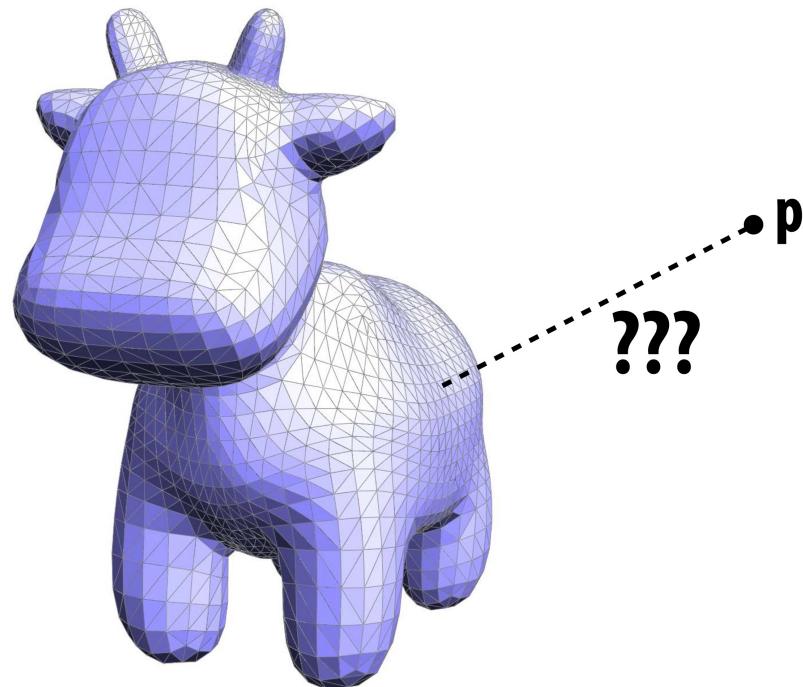


□此类问题在 CG 中称为**几何查询 (geometry queries)**

- 最近点查询 (Nearest point queries)
- 距离测量 (Distance measures)
- 交叉测试 (Intersection tests)
- 包含性测试 (Containment tests)
- 碰撞检测 (Collision detection)
- ...

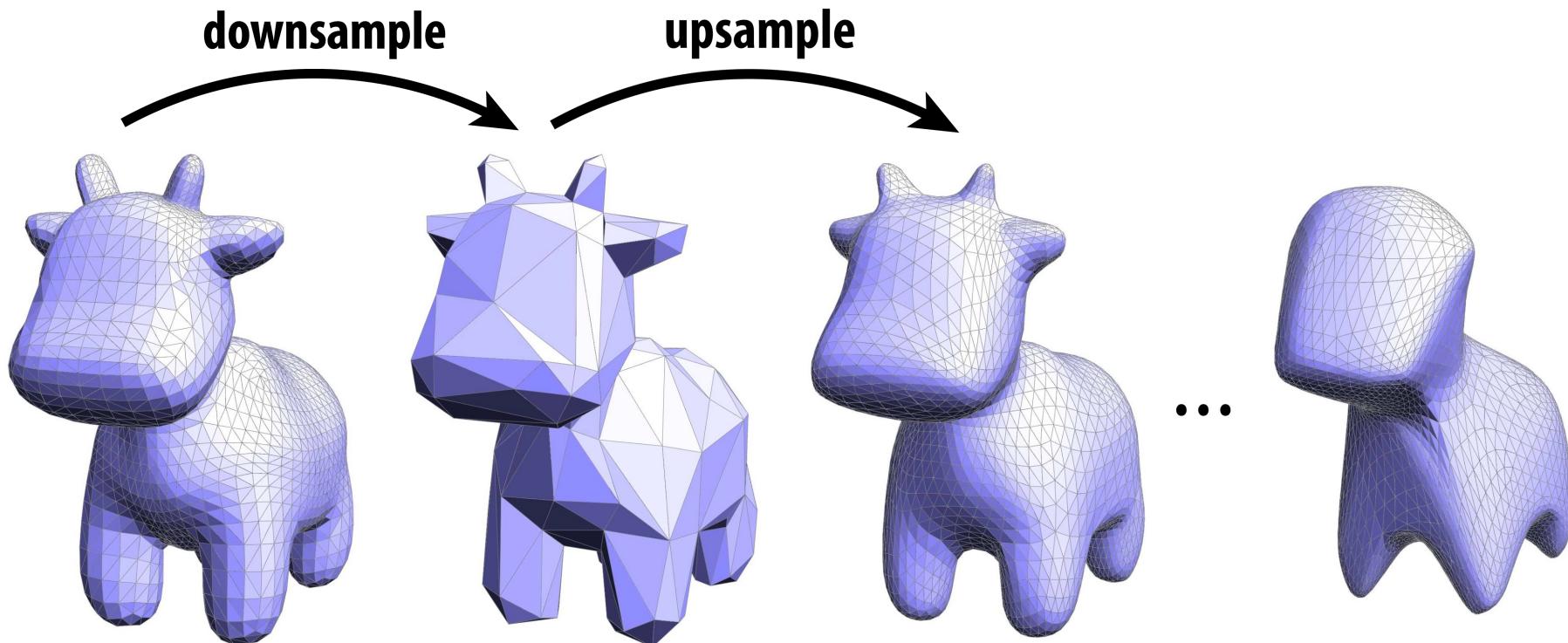
最近点查询

- Q：给定空间中一个点（比如光源），如何找到给定曲面上最近的点？
- Q：如何找到距离给定三角形最近的点（及距离的值）？
- Q：简单算法的代价是什么？有没有方法加速？



最近点查询在 CG 中应用广泛

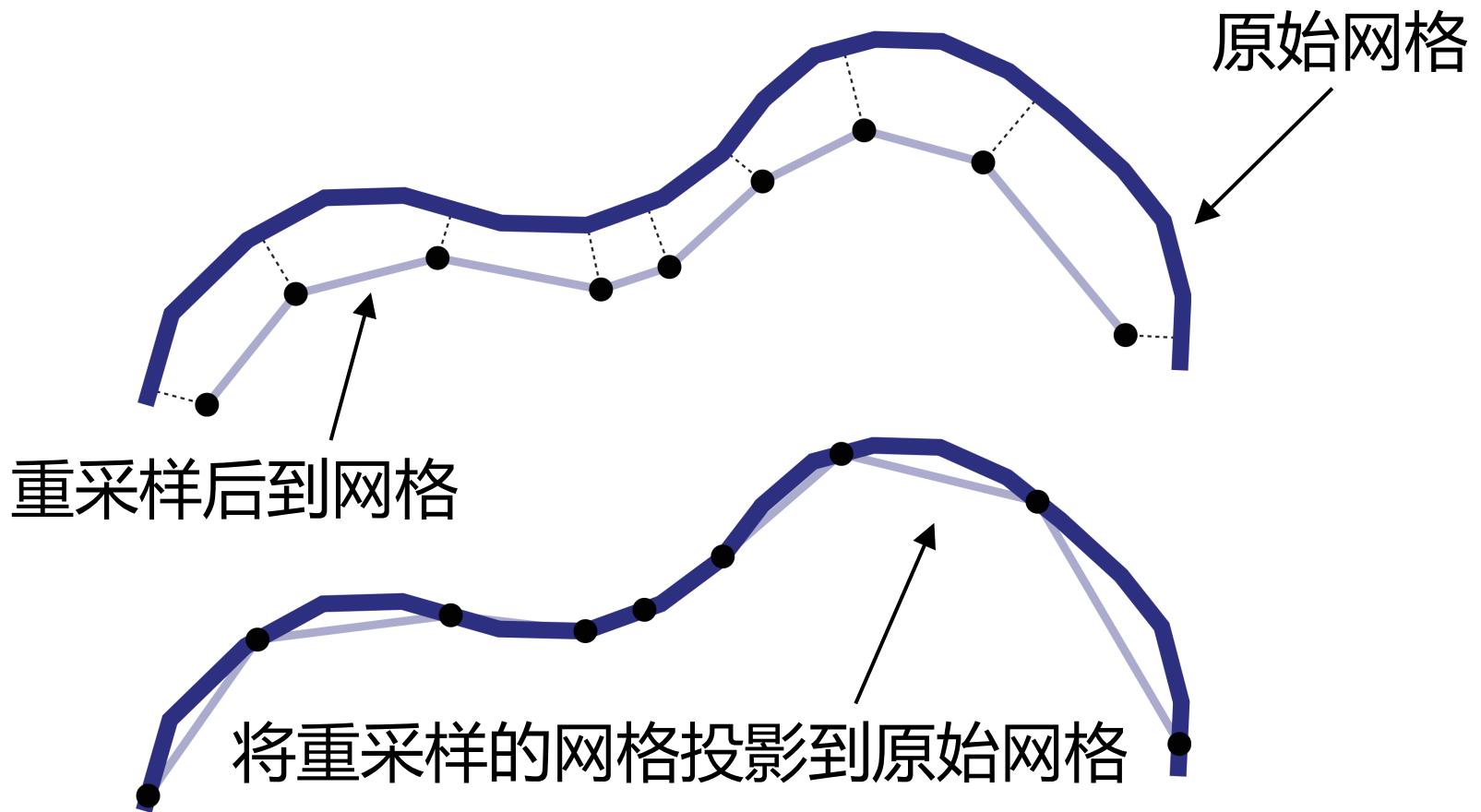
□ 几何处理中的信号退化



□ 如何才能更好地保存原始信号？

通过最近点投影恢复原始几何信息

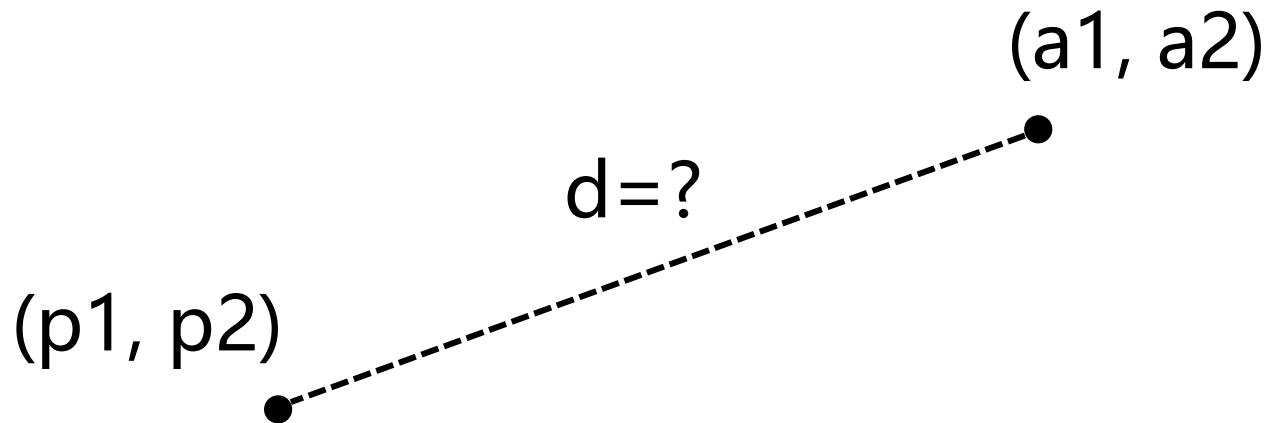
重采样后，将每个顶点投影到原始网格上



如何计算最近点？

热身：点中最近的点

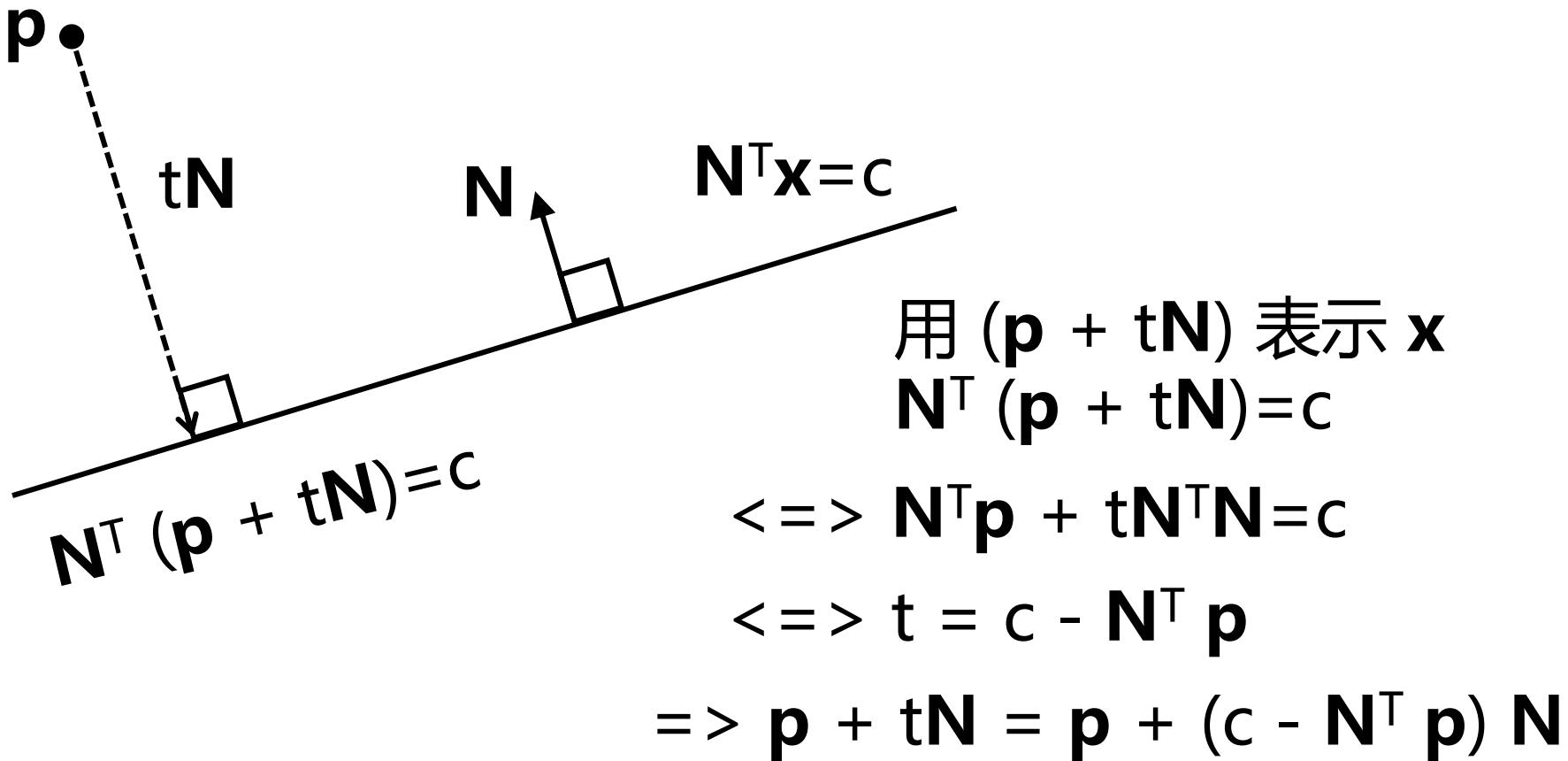
□一个简单的问题：给定一个查询点 (p_1, p_2) ，我们如何找到点 (a_1, a_2) 上最接近的点？



□最近的距离是多少？

稍微难一点：线上最近的点

- 现在假设我有一条线 $\mathbf{N}^T \mathbf{x} = c$, 其中 \mathbf{N} 为单位法线
- 如何找到线上最接近查询点 \mathbf{p} 的点?



更难一点：线段中最近的点

□ 线段上有两种点

- 端点 (**a** 和 **b**)
- 线段上的点

□ 分解成已有的算法

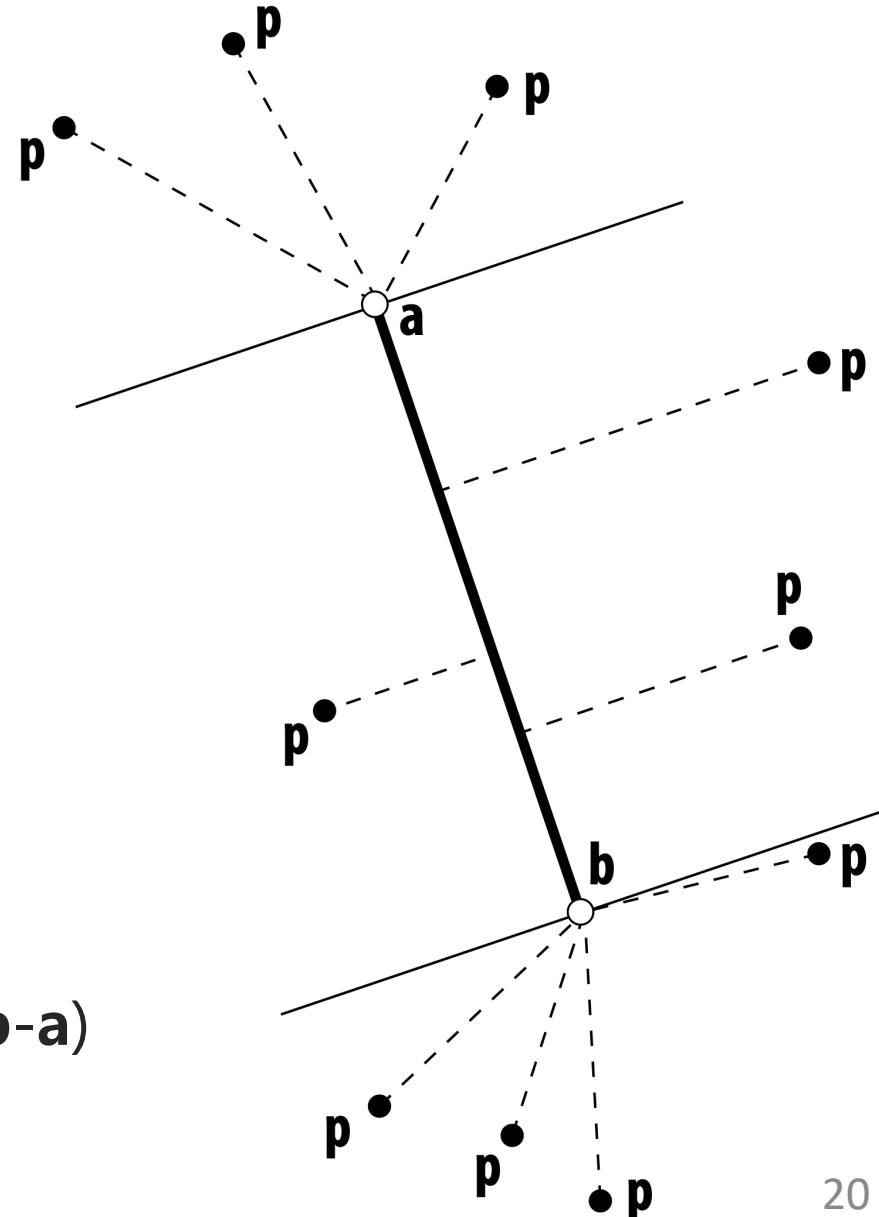
- 点上最近的点
- 线上最近的点

□ 算法：

- 找到直线上最近的点
- 检查最近的点是否在线段上
- 如果不是，返回最近的端点

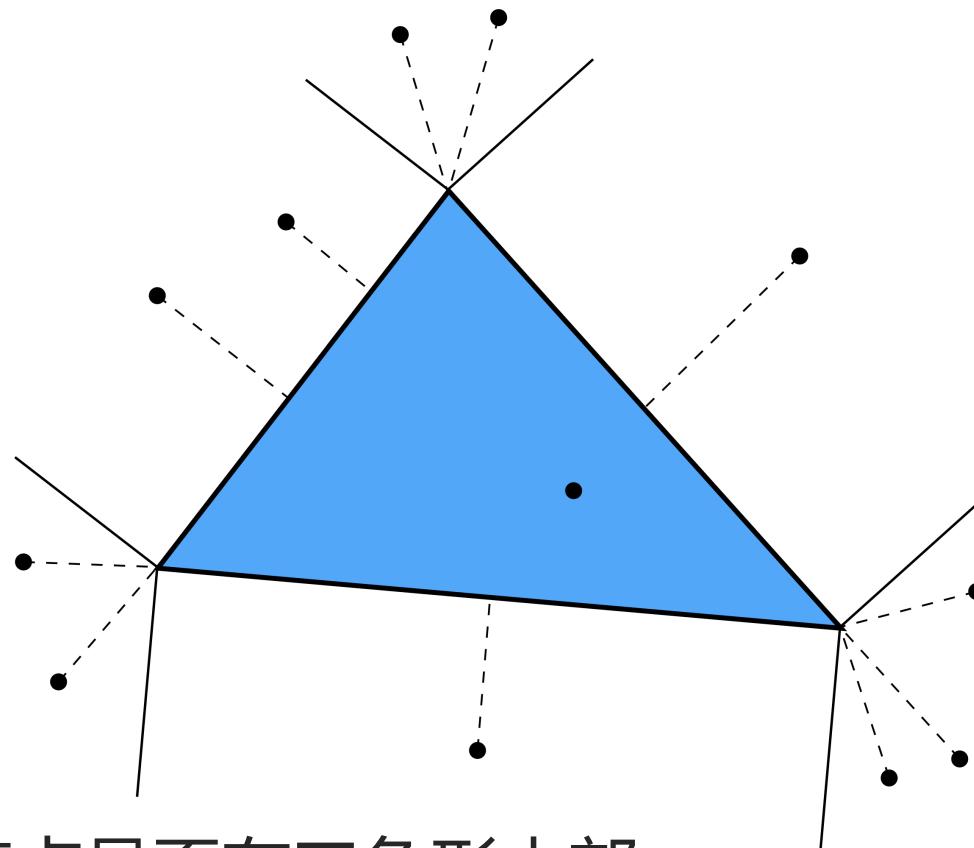
□ 怎么知道最近的点在线段内部？

- 把线上最近的点表示为 $a + t(b-a)$
- 如果 $0 < t < 1$ ，则在线段内部



再难一点：三角形上最近的点

- 三角形上最近的点有哪些情况？
- 只需要计算到三条线段上的最近点？



- 需要先检查点是否在三角形内部

3D 平面中最近的点

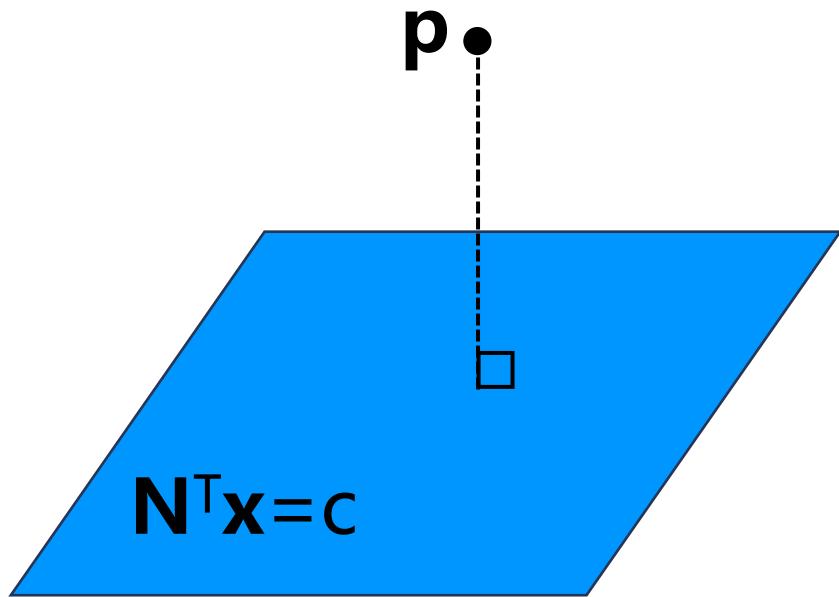
□与线上最近的点完全一样！

平面的表达式： $\mathbf{N}^T \mathbf{x} = c$

用 $(\mathbf{p} + t\mathbf{N})$ 表示 \mathbf{x}

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

$$\text{最近点为 } \mathbf{p} + (c - \mathbf{N}^T \mathbf{p}) \mathbf{N}$$



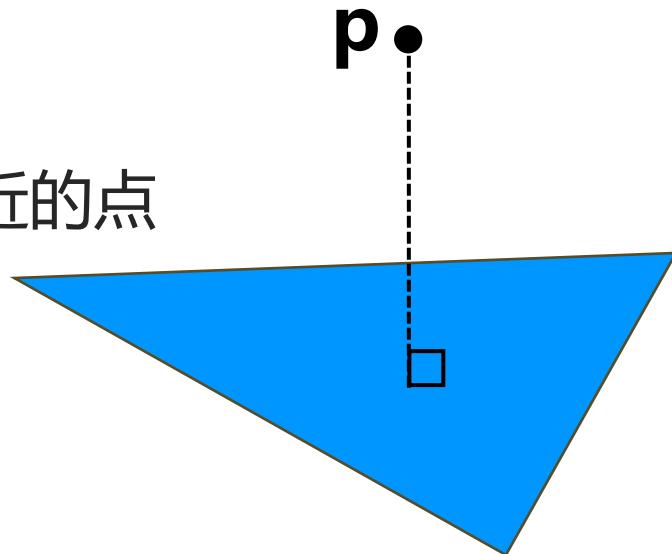
3D 中三角形最近的点

□ 跟 2D 的情况一样

□ 算法

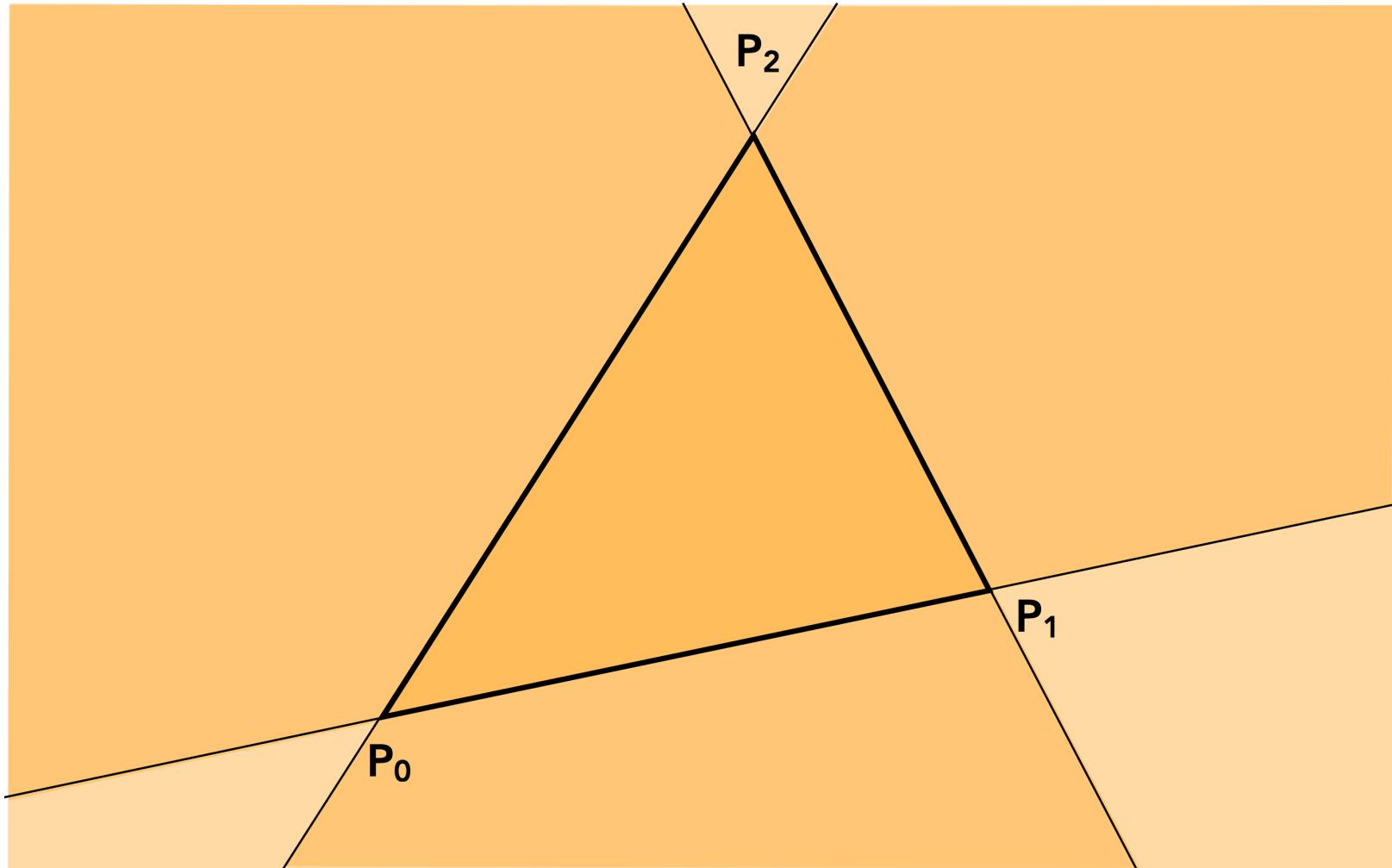
- 计算点 p 到三角形所在平面的最近点
- 判断最近点是否在三角形内部
- 如果最近点在三角形内部，结束
- 否则返回三角形边上或顶点上最近的点

□ 如何判断点是否在三角形内部？



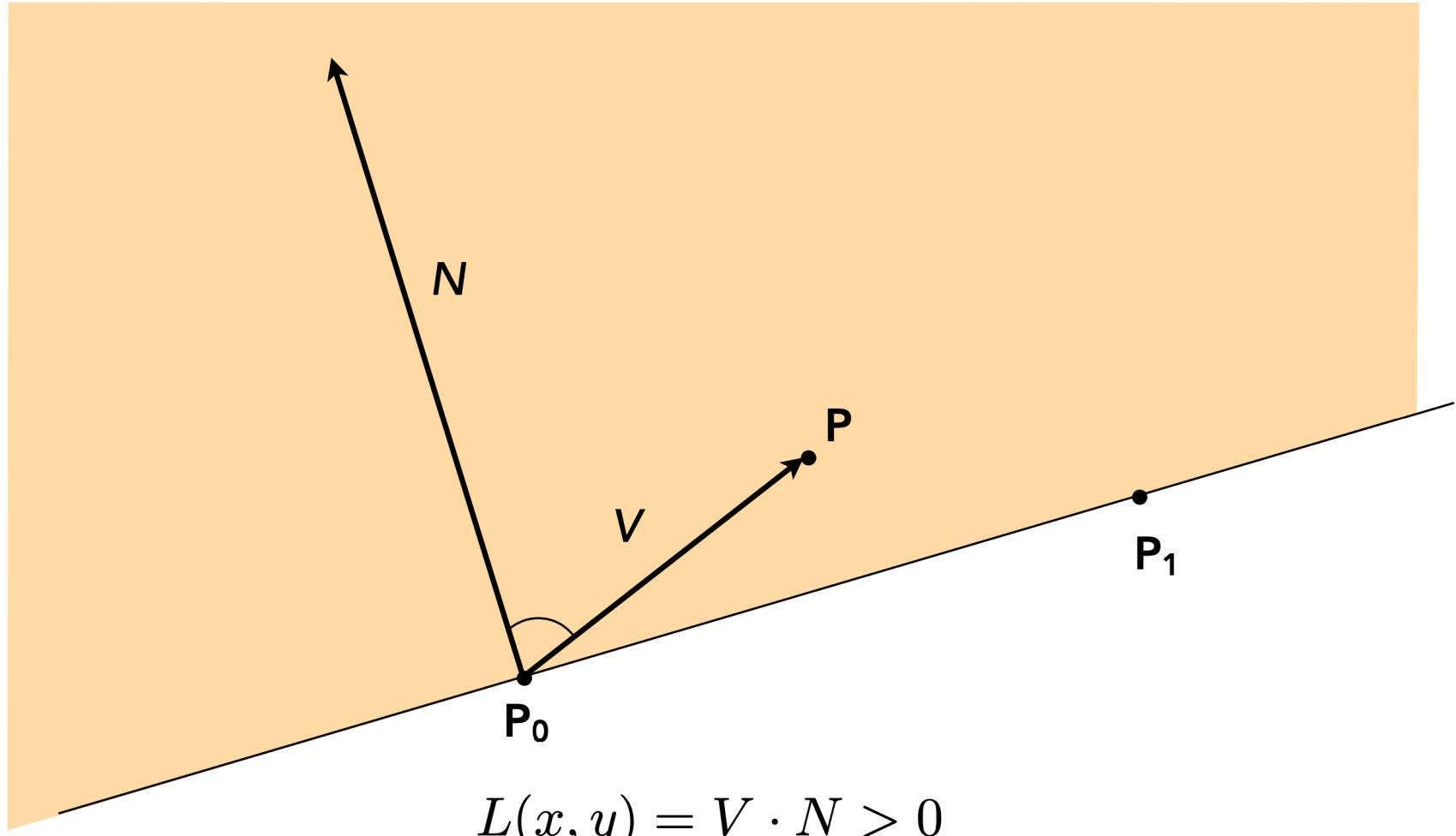
回顾：判断三角形内部点

口半平面测试



回顾：判断三角形内部点

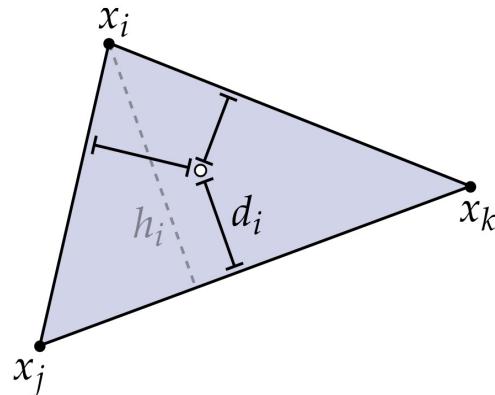
口半平面测试



$$L(x, y) = V \cdot N > 0$$

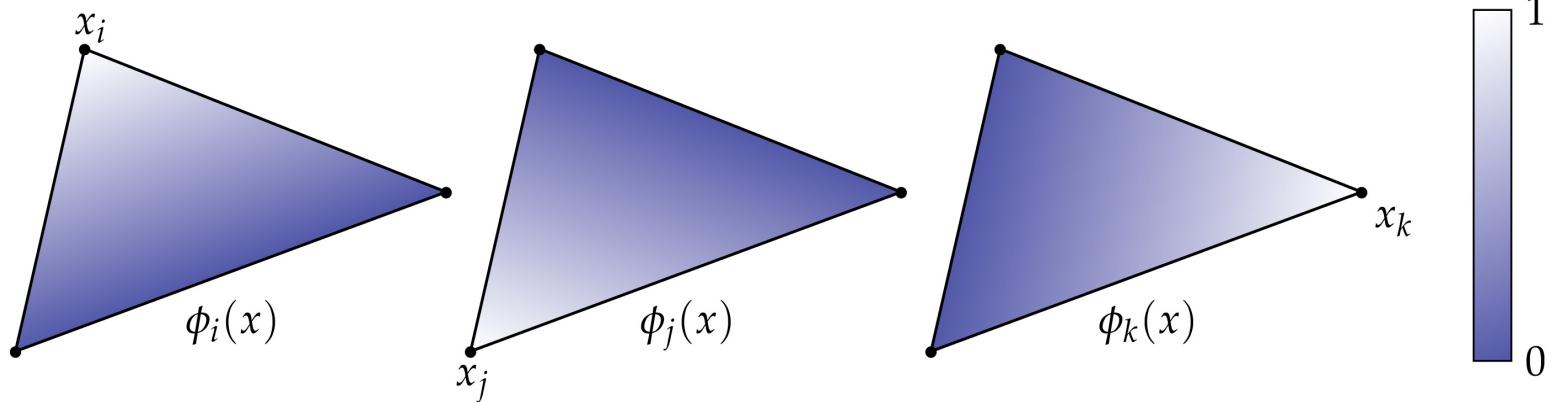
回顾：判断三角形内部点

重心坐标 (高的比值)



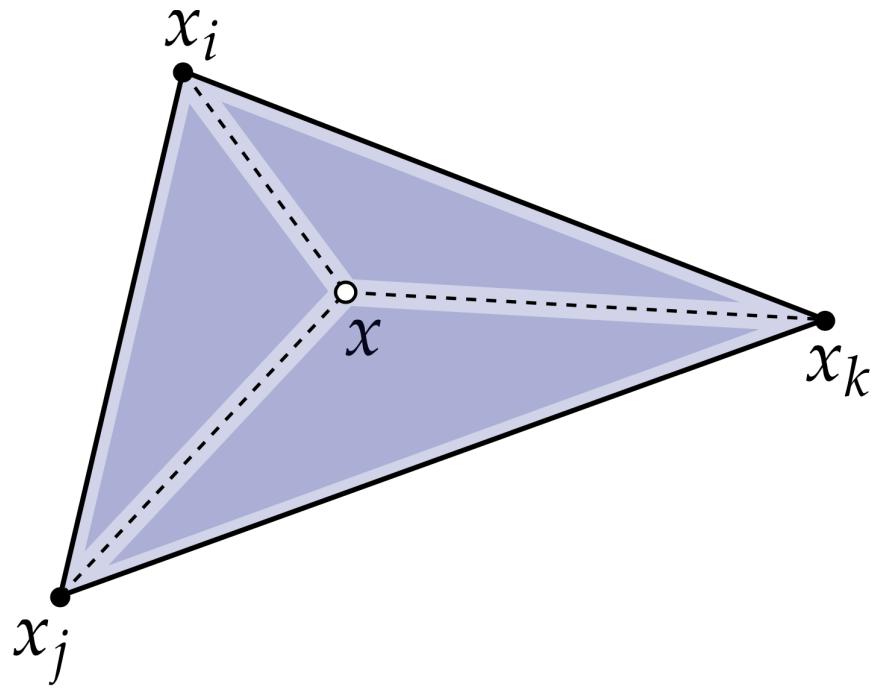
$$\phi_i(x) = d_i(x)/h_i$$

ϕ_i 为的 x_i 基函数



回顾：判断三角形内部点

口重心坐标 (面积比值)



$$\phi_i(x) = \frac{\text{area}(x, x_j, x_k)}{\text{area}(x_i, x_j, x_k)}$$

重心坐标的值均为正即在三角形内部

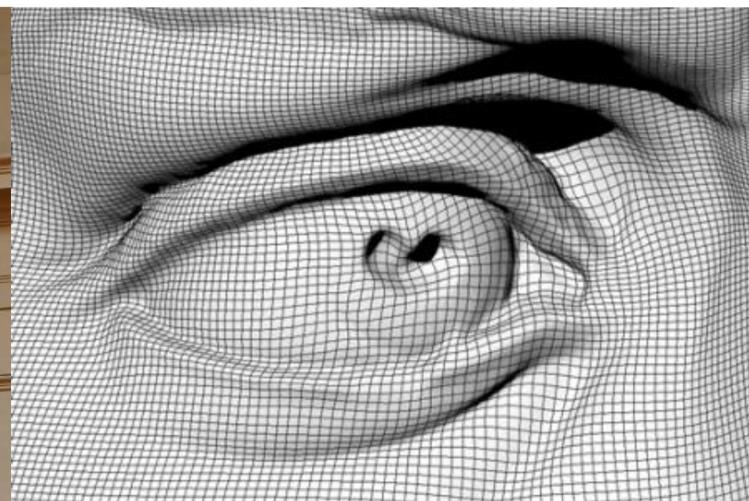
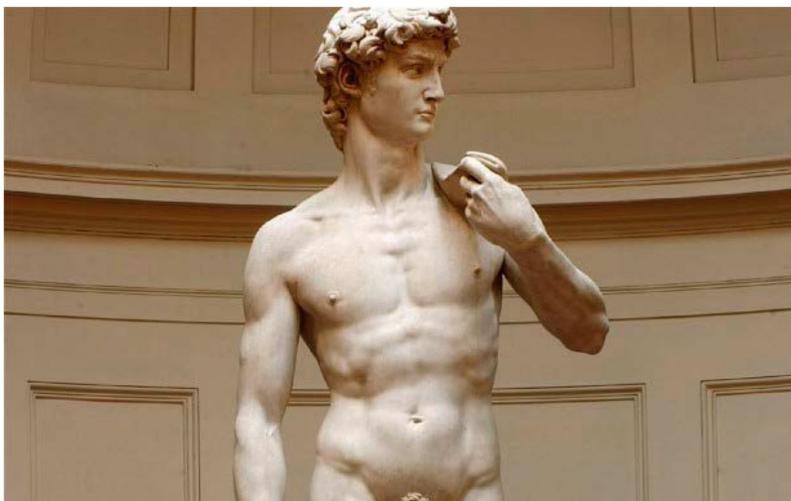
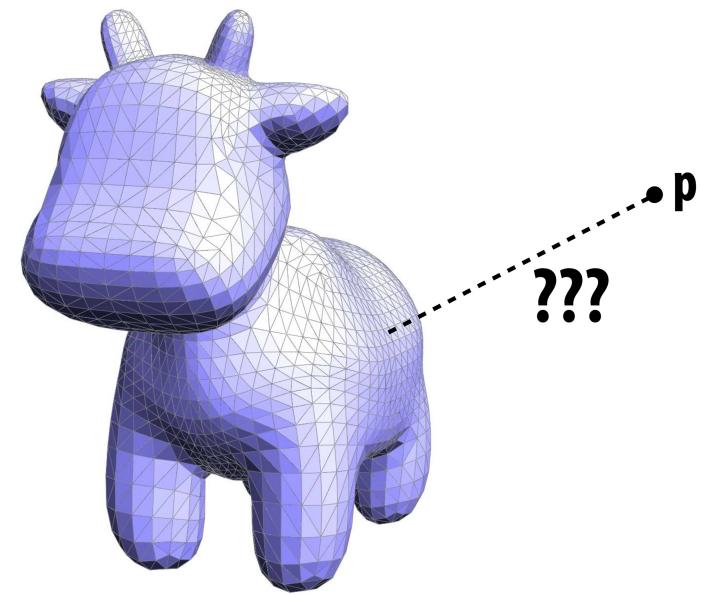
3D 三角形网格中的最近点

□ 算法很简单

- 循环网格中的所有三角形
- 计算到当前三角形最近的点
- 更新当前全局最近的点

□ 如果有几十亿的三角形呢？

□ 下一次课介绍更好的数据结构



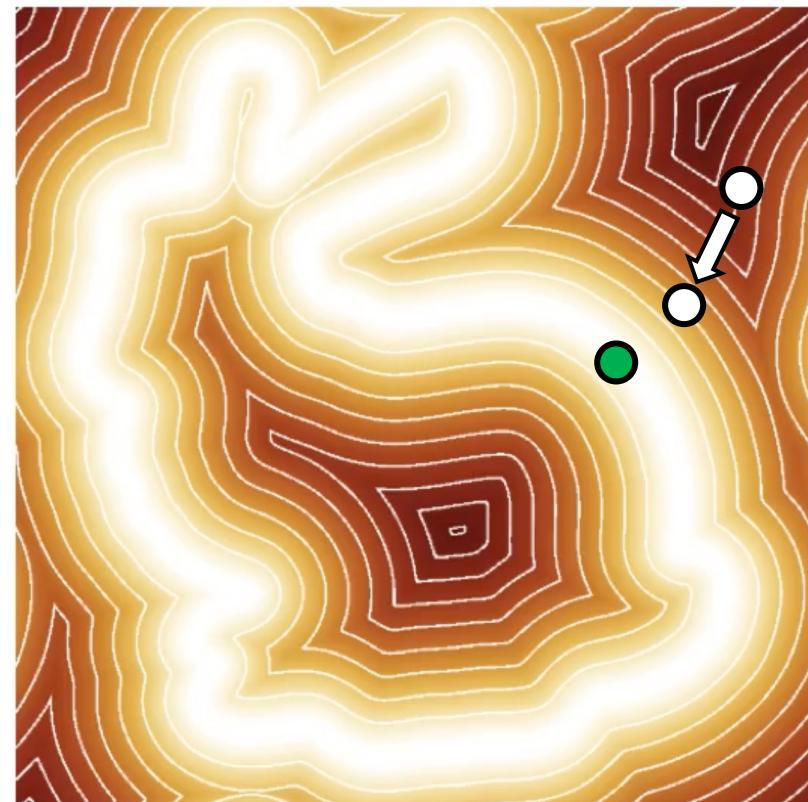
隐式 (implicit) 曲面上最近的点

- 口 使用不同的几何表示方法，算法可能会完全不一样
- 口 例如，我们如何计算通过距离函数 (distance function) 描述的隐式曲面上的最近点？

- 口 一种方法

- 从查询点开始
- 计算距离梯度
- 迈出一小步 (缩短距离)
- 重复，直到我们到达零距离曲线

- 口 加速方法：存储每个网格单元的最近点 (速度/内存的权衡)



光线与网格交叉

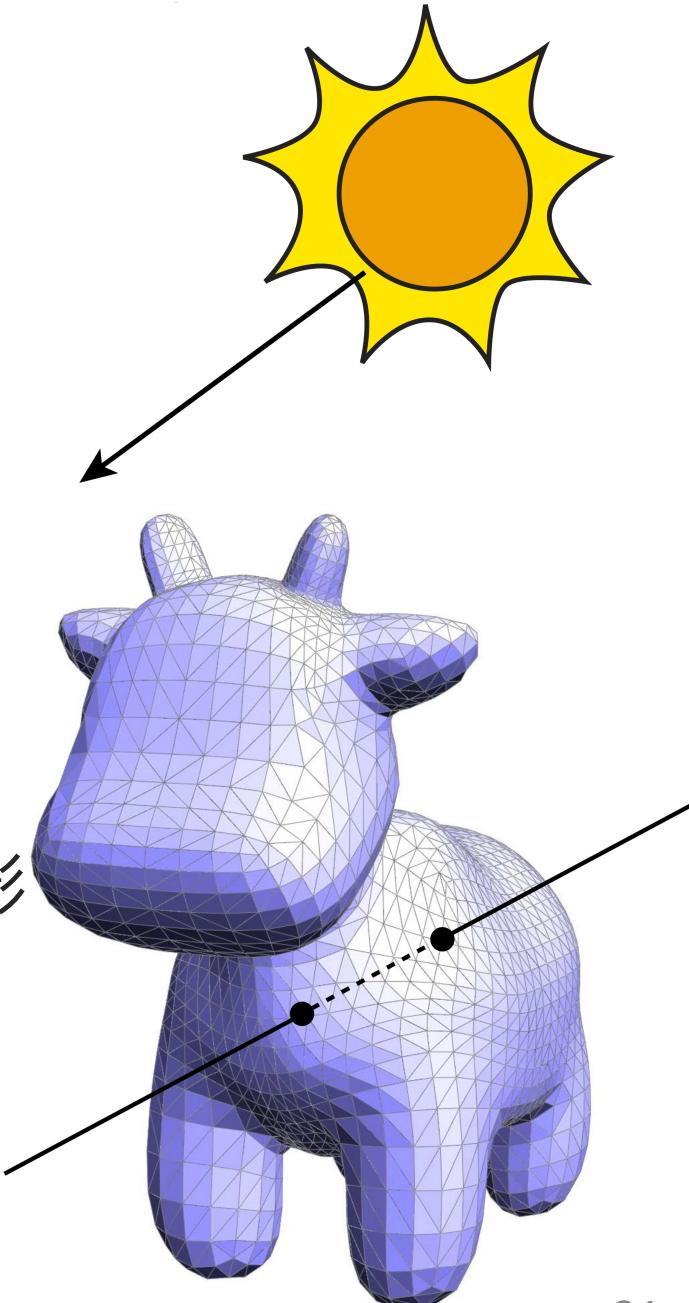
光线与网格交叉

- “光线” 是从一点开始的定向直线
- 想象一束来自太阳的光
- 我们想知道光线穿透表面的位置

□ 有什么作用？

- 几何 Geometry：内外测试
- 渲染 Rendering：可见性，生成阴影
- 动画 Animation：碰撞检测

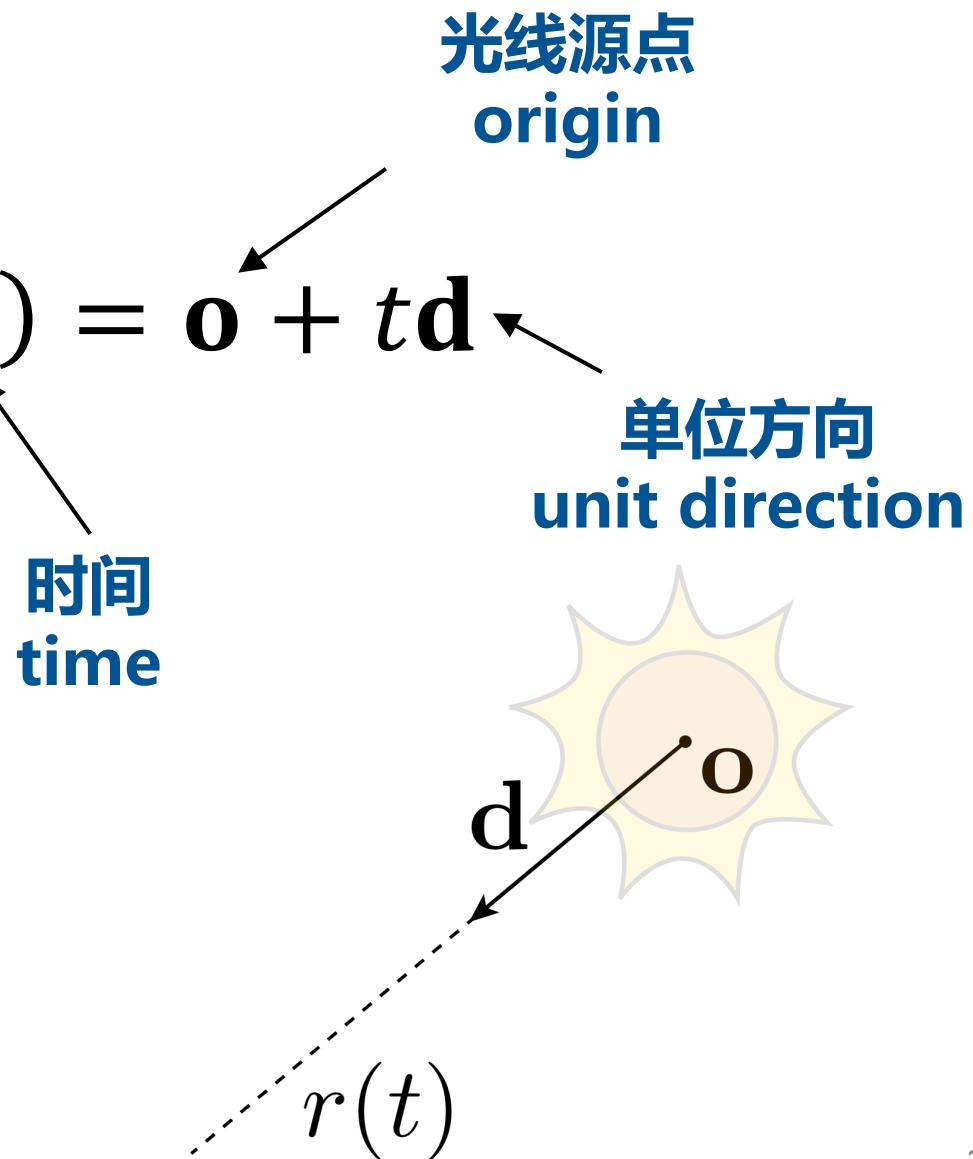
□ 可能会在许多地方穿透表面！



光线方程 Ray equation

光线可用如下方程表示

沿着光线在 t 时间上的点
point along ray at t



光线与隐式曲面相交

口回忆曲面的隐式表示法：所有使得 $f(\mathbf{x}) = 0$ 的点 \mathbf{x}

口Q：我们如何找到光线穿透这个表面的点？

口沿着光线在时间 t 的点为： $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$

口算法：将方程中的 \mathbf{x} 替换为 \mathbf{r} , 并求解 t

口示例：单位球体

$$f(x) = |\mathbf{x}|^2 - 1$$

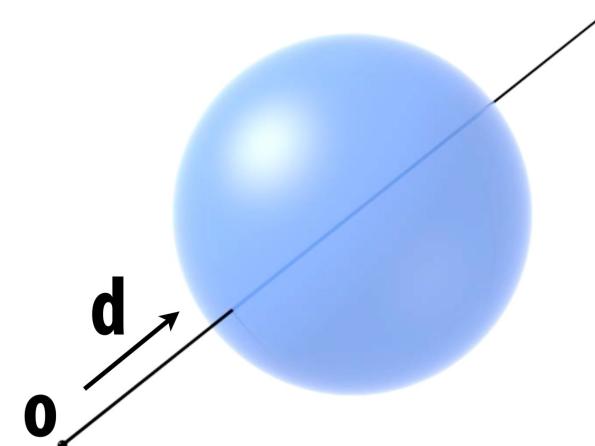
$$\Rightarrow f(\mathbf{r}(t)) = |\mathbf{o} + t\mathbf{d}|^2 - 1$$

$$\underbrace{|\mathbf{d}|^2 t^2}_{a} + \underbrace{2(\mathbf{o} \cdot \mathbf{d})t}_{b} + \underbrace{|\mathbf{o}|^2 - 1}_{c} = 0$$

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

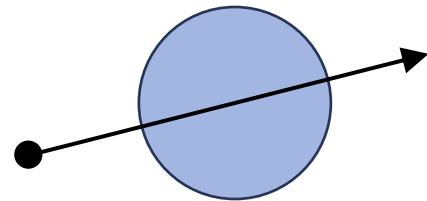
quadratic formula:

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

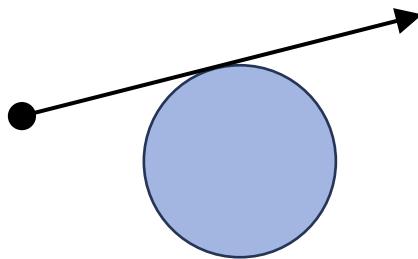


对解的讨论

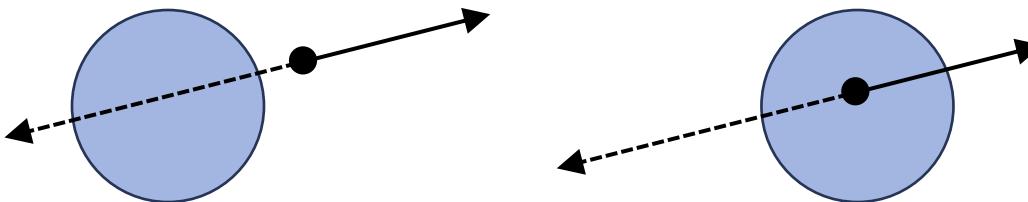
□有两个解



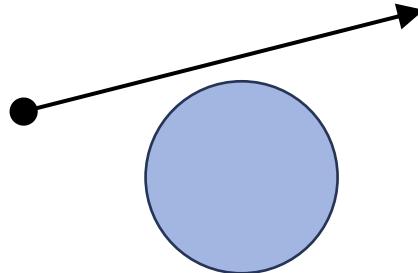
□有一个解



□负数解



□无解



光线与平面相交

□ 假设我们有平面 $\mathbf{N}^T \mathbf{x} = c$

- \mathbf{N} 为单位法线
- c 为偏移量 (offset)

□ 我们如何找到平面与光线的交点?

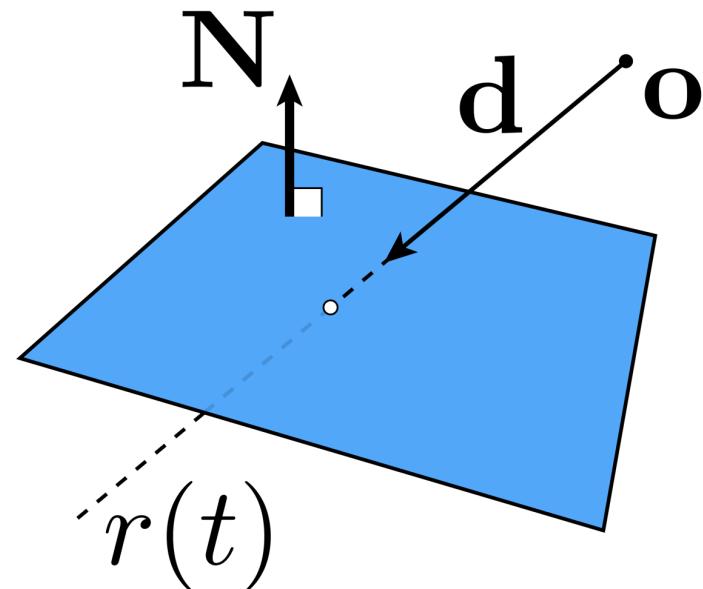
□ 算法: 同样地, 把点 x 替换为光线方程

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

□ 求解 t :

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

□ 将 t 代入光线方程中 $\mathbf{r}(t) = \mathbf{o} + \frac{c - \mathbf{N}^T \mathbf{o}}{\mathbf{N}^T \mathbf{d}} \mathbf{d}$

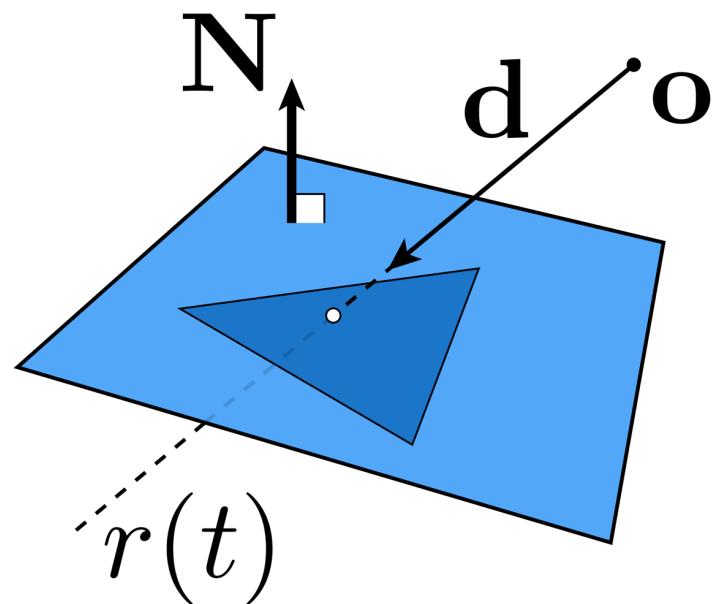


光线与三角形相交

- 三角形在平面中
- 因此，与光线和平面相交类似

- 首先计算光线与平面的交点
- 判断交点是否在三角形内部
- 可使用三次半平面测试，或重心坐标

□ 上述的算法很简单，难的在于性能！



≡ Google Scholar ray triangle intersection methods

Articles About 120,000 results (0.11 sec)

Any time Fast, minimum storage **ray/triangle intersection** CCFA [PDF] utah.edu
Since 2023 T Möller, B Trumbore - ACM SIGGRAPH 2005 Courses, 2005 - dl.acm.org
Since 2022 ... for **triangle** meshes. As we found our **method** to be comparable in speed to previous **methods**,
Since 2019 we believe it is the fastest **ray/triangle intersection** routine for **triangles** which do not have ...
Custom range... ☆ Save Cite Cited by 1696 Related articles All 16 versions

Sort by relevance Optimizing **ray-triangle intersection** via automated search CCF none [PDF] psu.edu
Sort by date A Kensler, P Shirley - ... IEEE Symposium on Interactive Ray ..., 2006 - ieeexplore.ieee.org
Any type ... hood" these **methods** are all taking ... **methods** we optimize **ray-triangle intersection** in two
Review articles different ways. First we do explicit operation counting for the cases of single **rays**, packets of **rays** ...
☆ Save Cite Cited by 67 Related articles All 7 versions

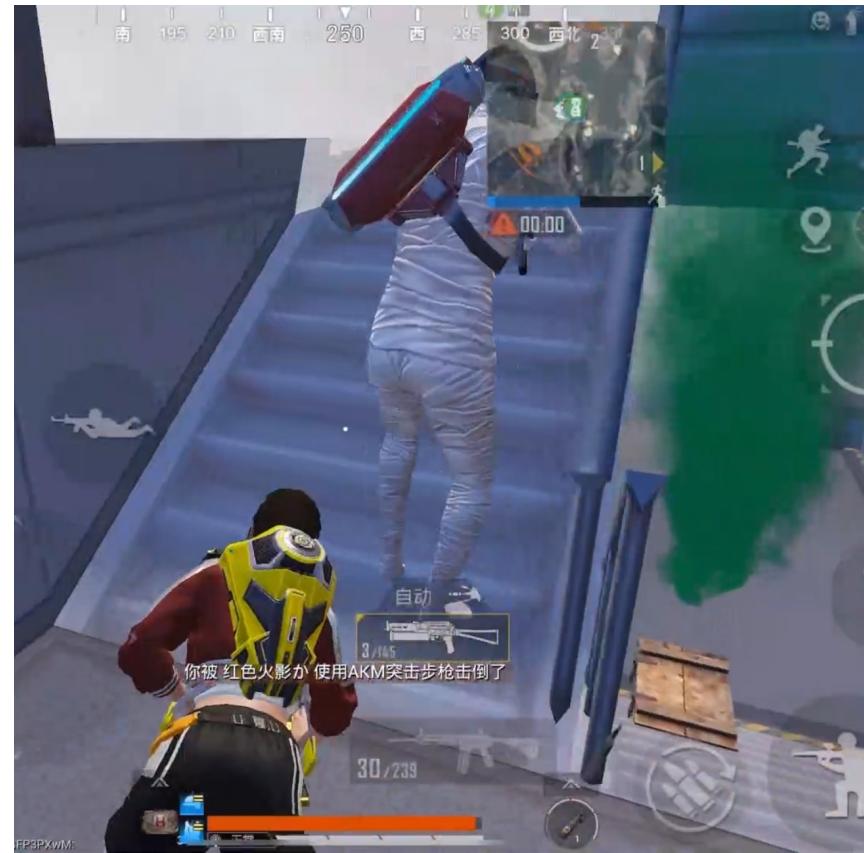
为什么要关心性能？



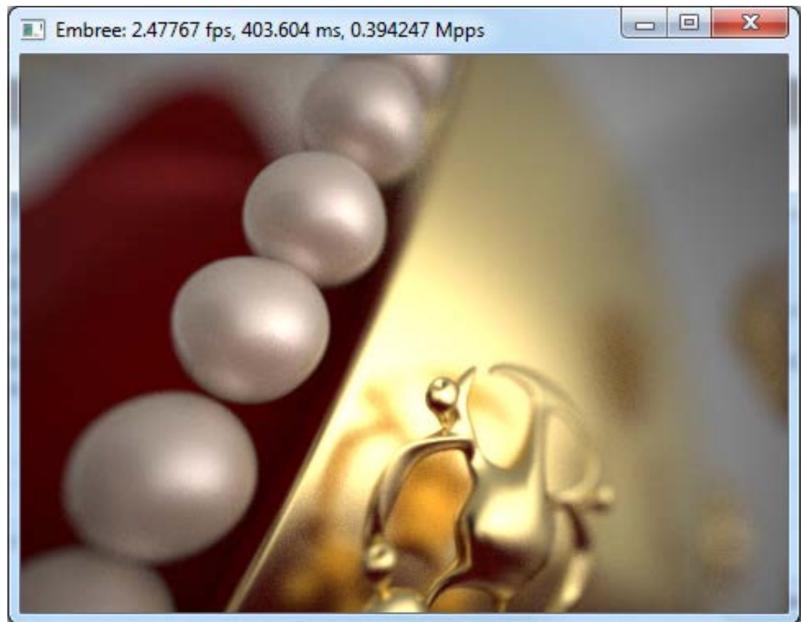
Pixar's "Coco" – 一帧需要渲染 **50个小时** (每秒 24 帧)

性能在游戏中也很重要

和平精英无阴影模式



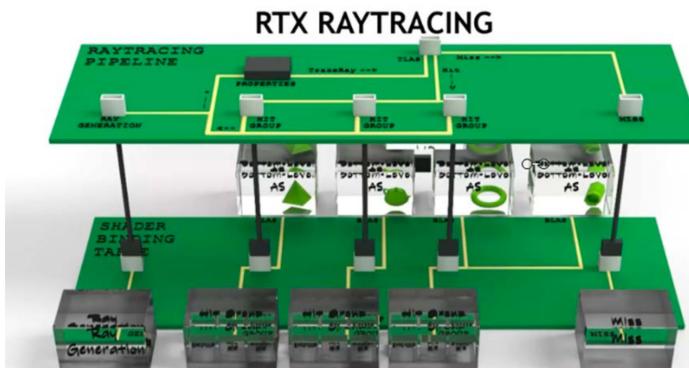
高性能光线跟踪



Intel Embree



NVIDIA OptiX



Hardware-accelerated ray tracing (RTX)

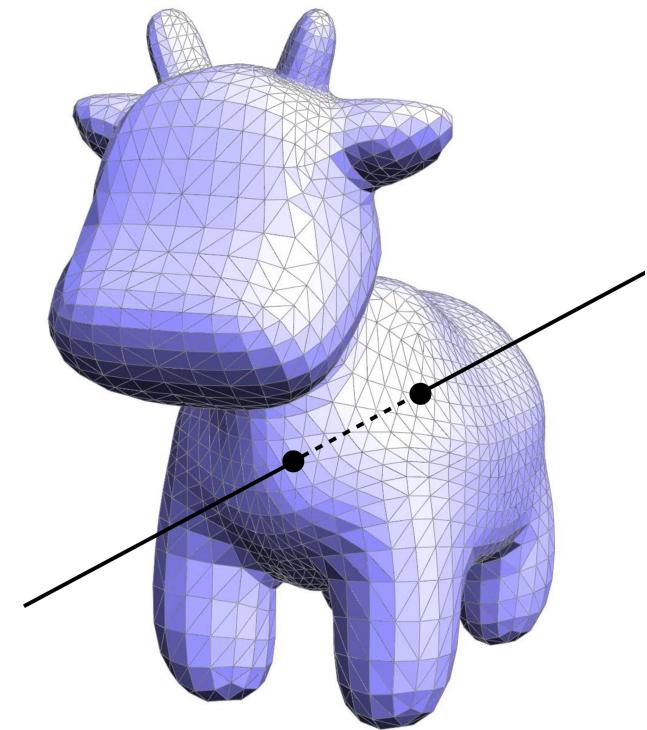
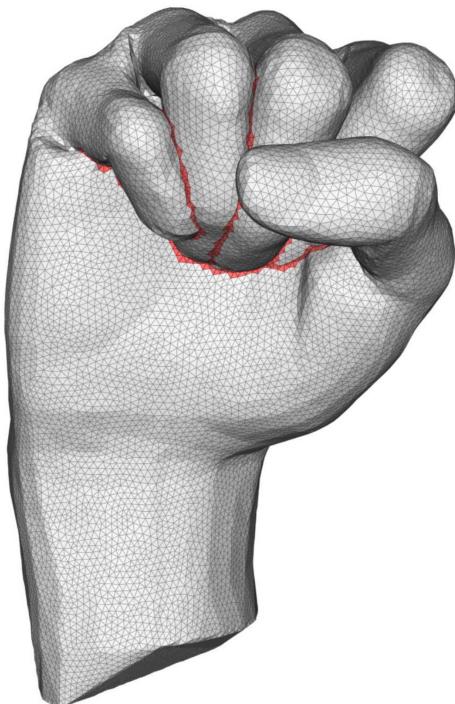
高性能光线跟踪



NVIDIA's "Project Sol" — real time ray tracing demo (RTX)

另一种几何查询：网格与网格相交

- 虽然不与当前主题（光线追踪）直接相关，但也非常重要
- 几何：我们怎么知道网格与自己相交？
- 动画：我们怎么知道发生了碰撞（collision）？



热身：点与点相交

□Q：怎么判断点 $p(p_1, p_2)$ 与点 $a(a_1, a_2)$ 相交？

□A：检查它们是否为同一点

(p_1, p_2)



(a_1, a_2)



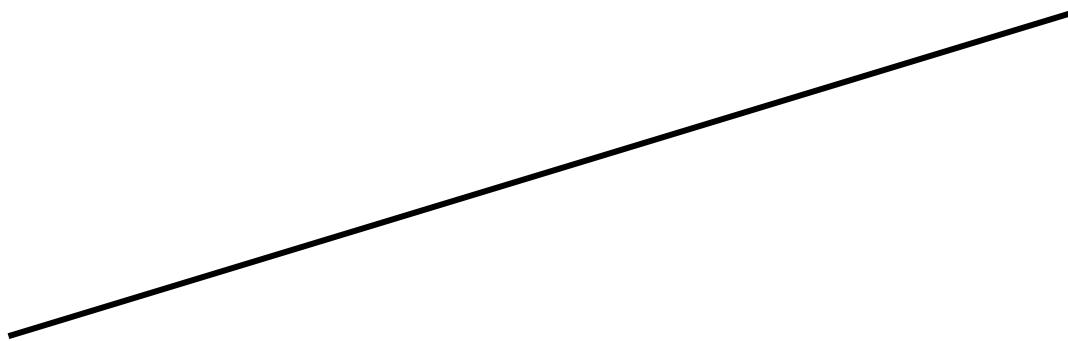
稍微难一点：点和线相交

□Q：怎么判断直线是否穿过某一点？

□A：将其带入到直线方程

p•

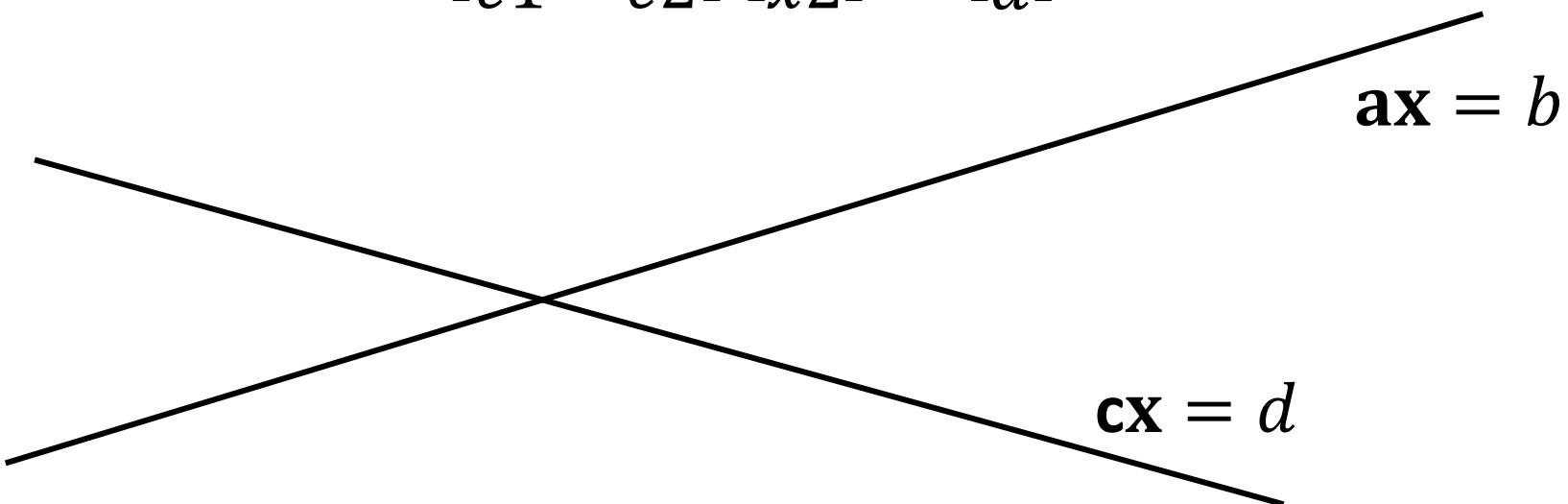
$$\mathbf{N}^T \mathbf{x} = \mathbf{c}$$



直线和直线相交

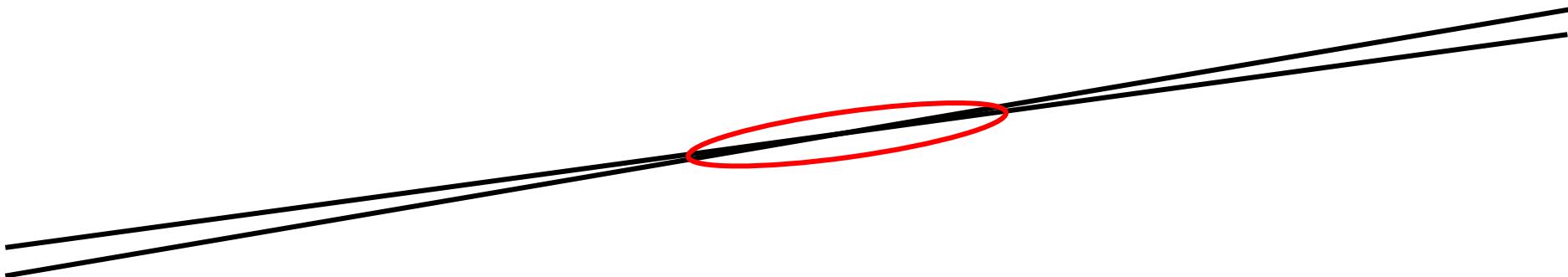
- 两条直线： $\mathbf{ax} = b$ 和 $\mathbf{cx} = d$
- Q：如何找到交叉点？
- A：检查这两条直线是否有共同的解
- 求解线性系统

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



退化的直线与直线相交

- 口在数学上，永远可以求解出精确的交叉点
- 口当在现实中，计算机的精度是有限的！
- 口直线几乎平行时，可能会有多个交叉点（多个交叉的像素）
- 口此类不稳定性在几何处理中非常常见，需要特别小心

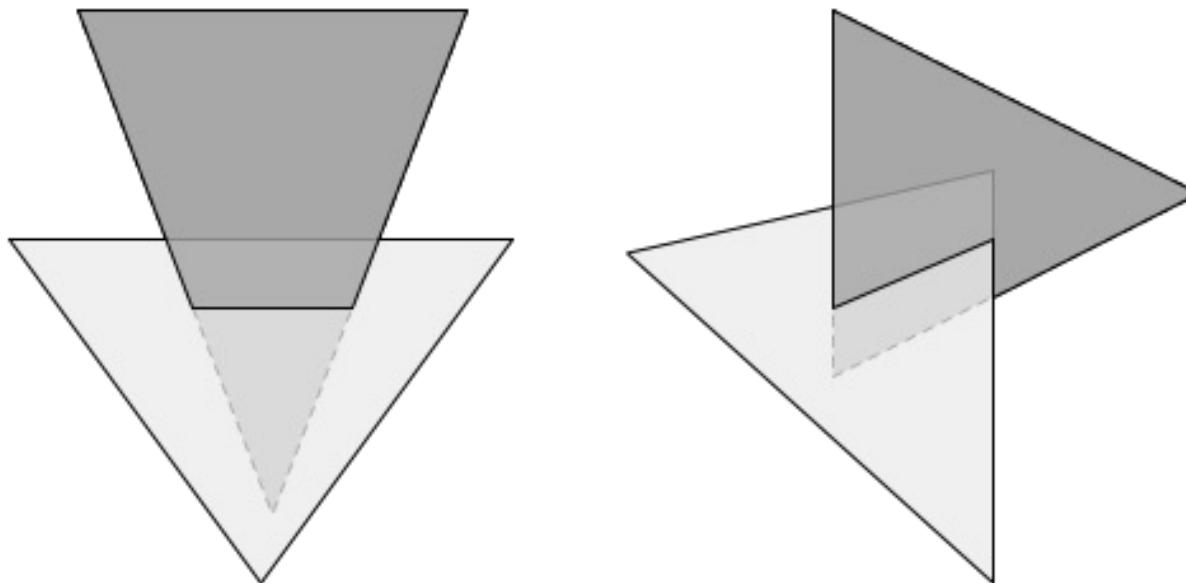


三角形和三角形相交

口同样的，将其分解成我们已有的算法

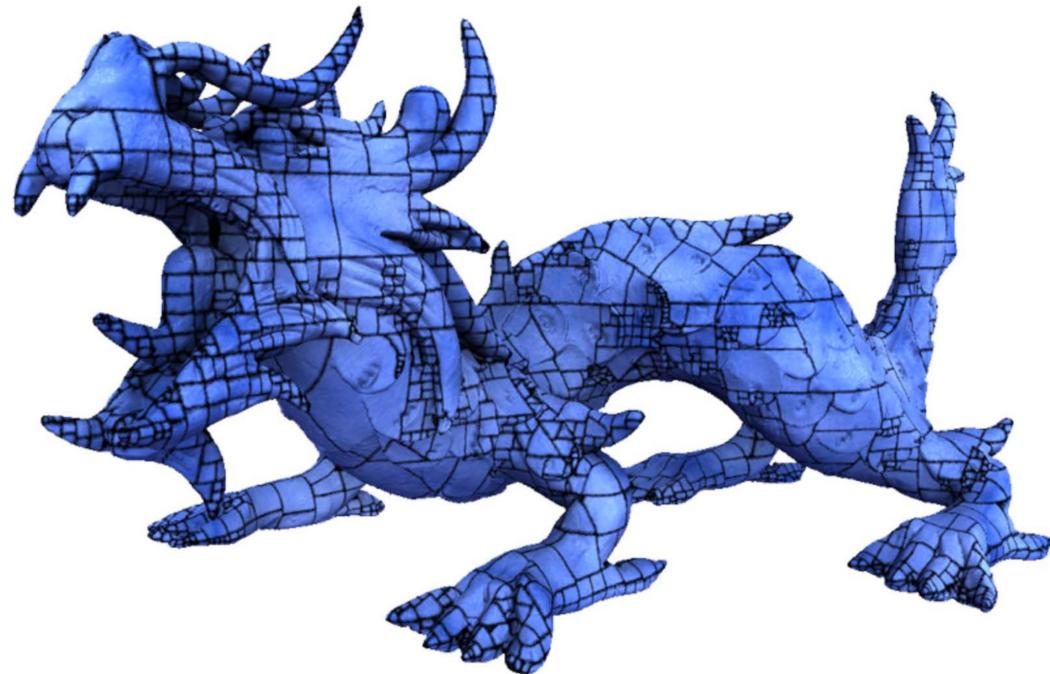
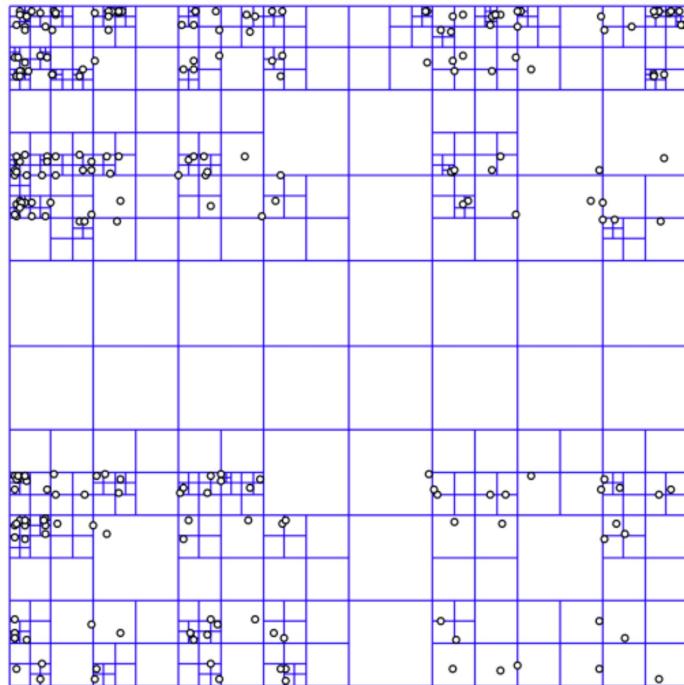
口两个三角形相交：

- 要么一个三角形的两条边与另一条相交（下图左侧）
- 或者两个三角形的一条边与另一个三角形相交（下图右侧）



下节课：空间加速的数据结构

- 每次计算所有元素很慢！
- 类似地，扫描一个线性列表很慢，但二进制搜索很快
- 可以将同样的思维应用于几何查询





中山大學 软件工程学院
SUN YAT-SEN UNIVERSITY SCHOOL OF SOFTWARE ENGINEERING

谢谢

陈壮彬
软件工程学院
chenzhb36@mail.sysu.edu.cn