



Lecture 12: 几何处理

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

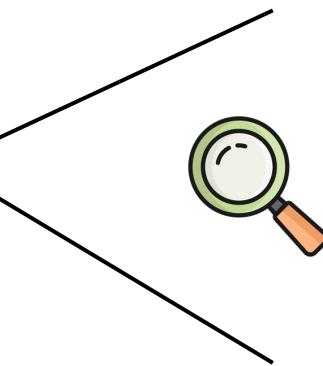
计算机图形学介绍
基于采样的光栅化
空间变换
纹理映射、深度和透明度

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



几何介绍
曲线与曲面
几何处理

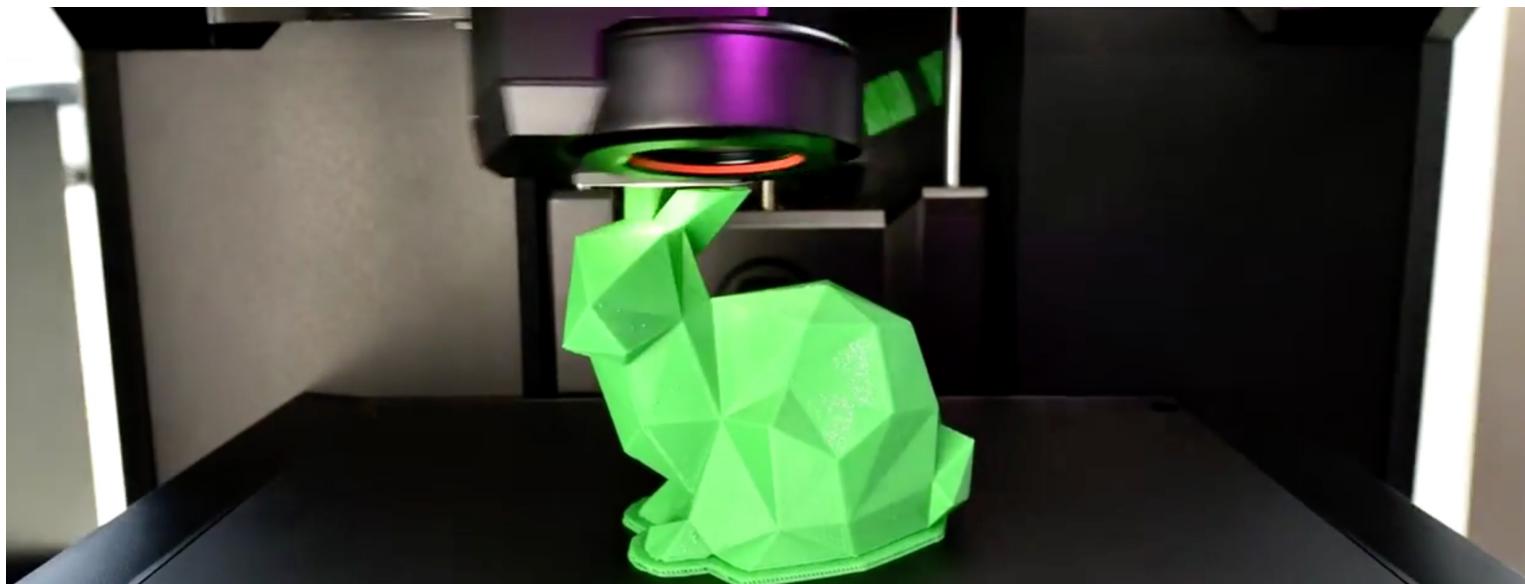


几何处理

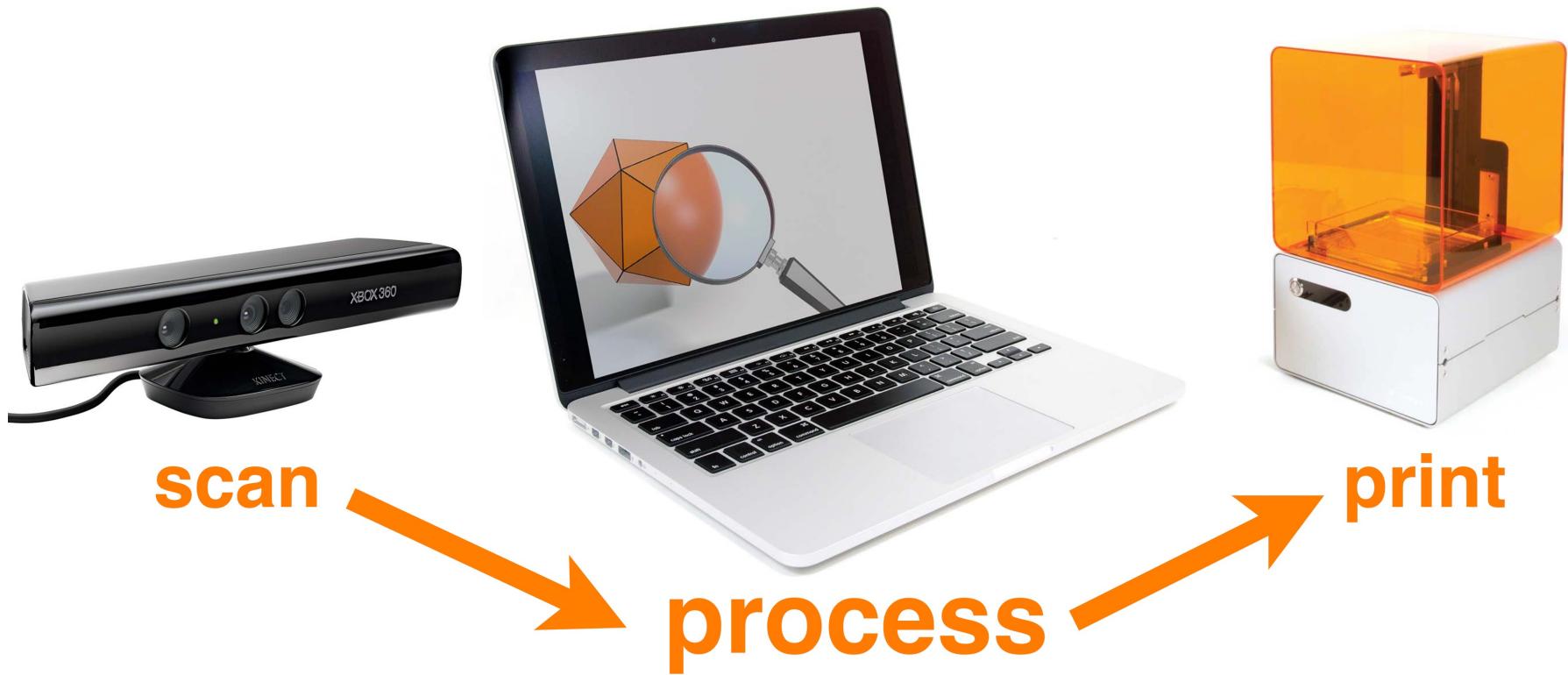
3D Scanning



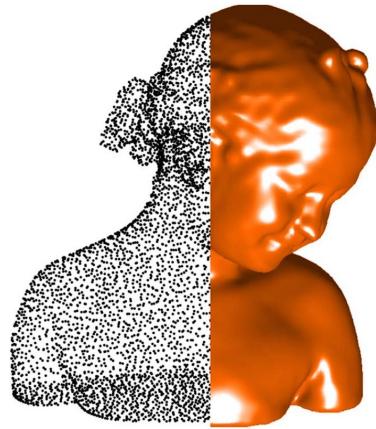
3D Printing



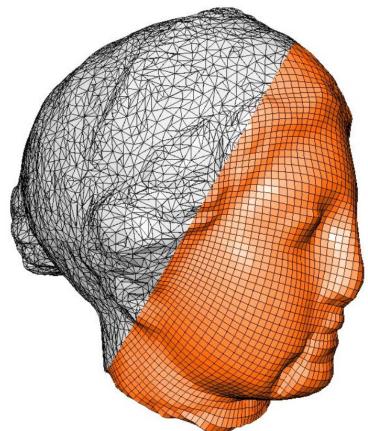
几何处理流程



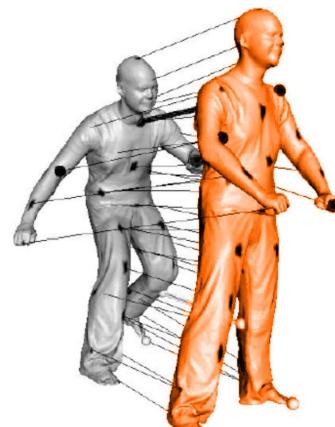
几何处理任务



reconstruction



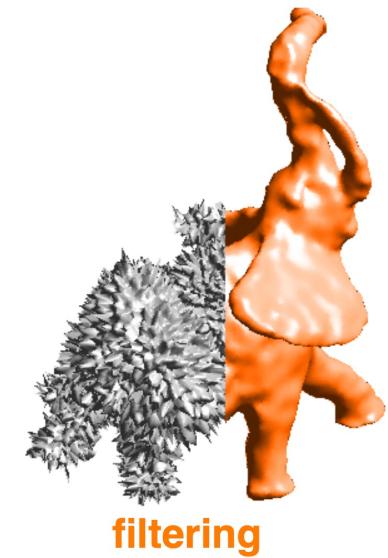
remeshing



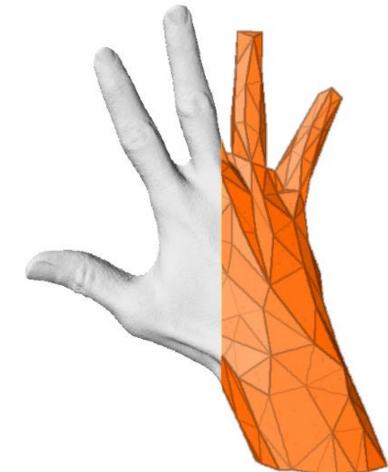
shape analysis



parameterization



filtering



compression

几何处理：重建 Reconstruction

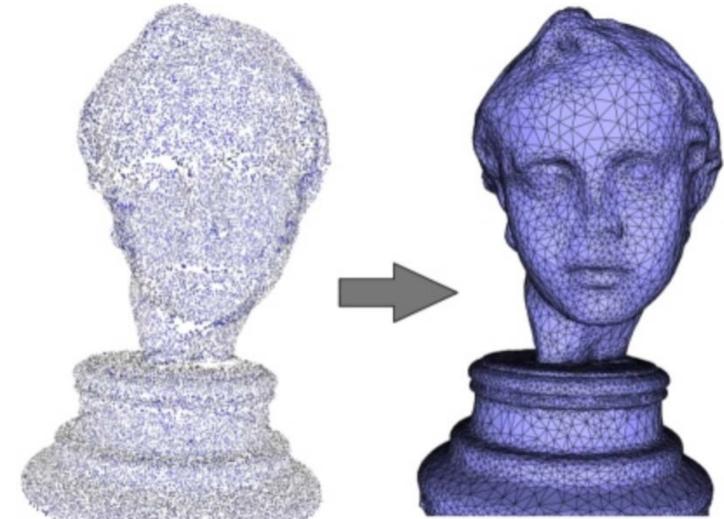
□ 给定几何样本，重建表面

□ 什么是“样本”？有很多可能性：

- 点，点和法线，...
- 图像对/集合（多视角立体）
- 线密度积分（MRI/CT 扫描）

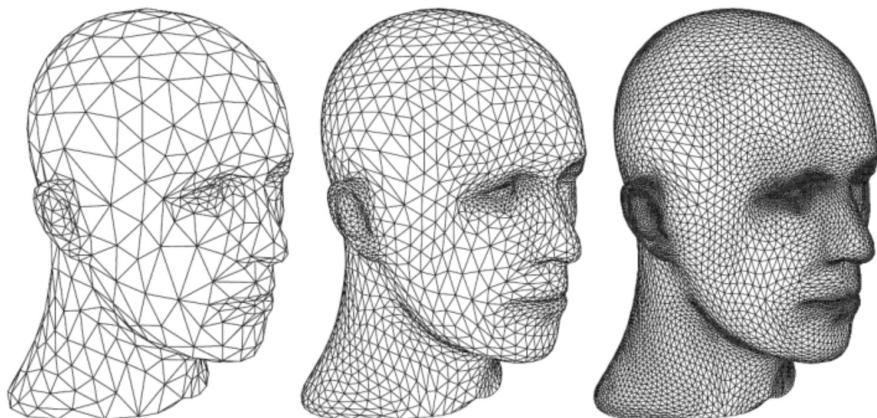
□ 如何获取一个表面？有多种技术：

- 基于轮廓线（视觉包围）
- 基于 Voronoi 图（例如，幂壳：计算点云的 Voronoi 图）
- 基于偏微分方程（例如，泊松重建）
- Radon 变换/等值面化（行进立方体算法）



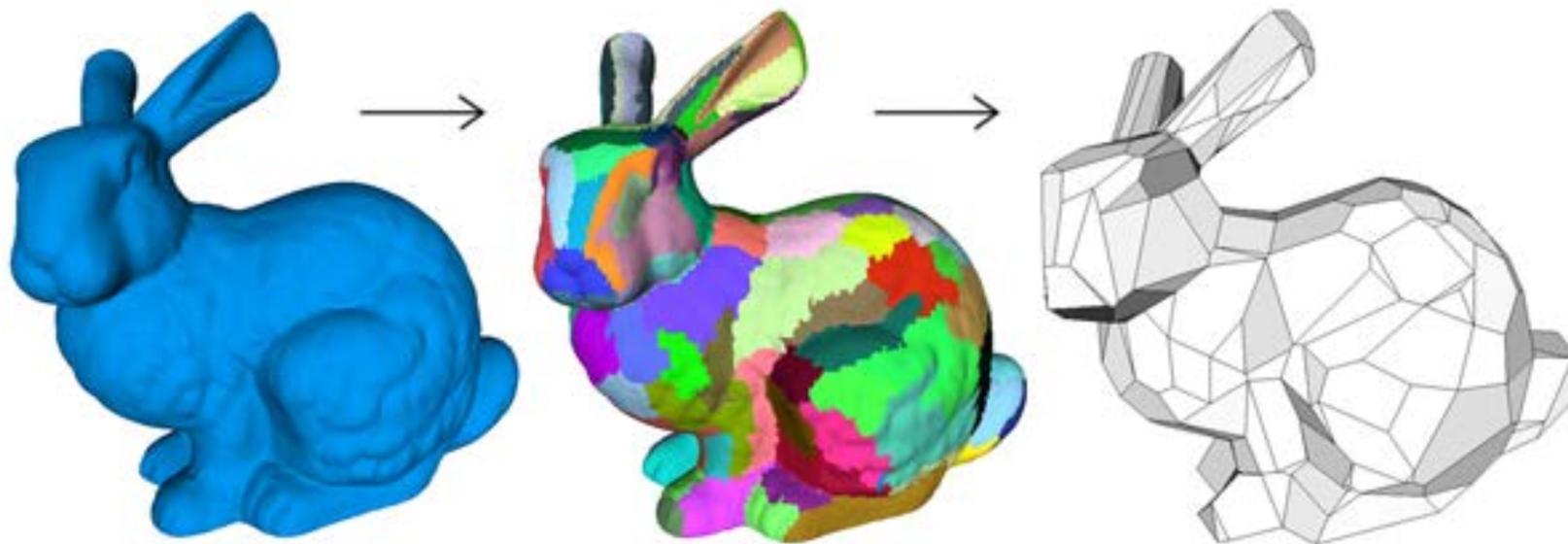
几何处理：上采样 Upsampling

- 通过插值提高分辨率
- 图像：例如，双线性插值、双三次插值
- 多边形网格：
 - 细分 (subdivision)
 - 双边上采样 (bilateral upsampling)



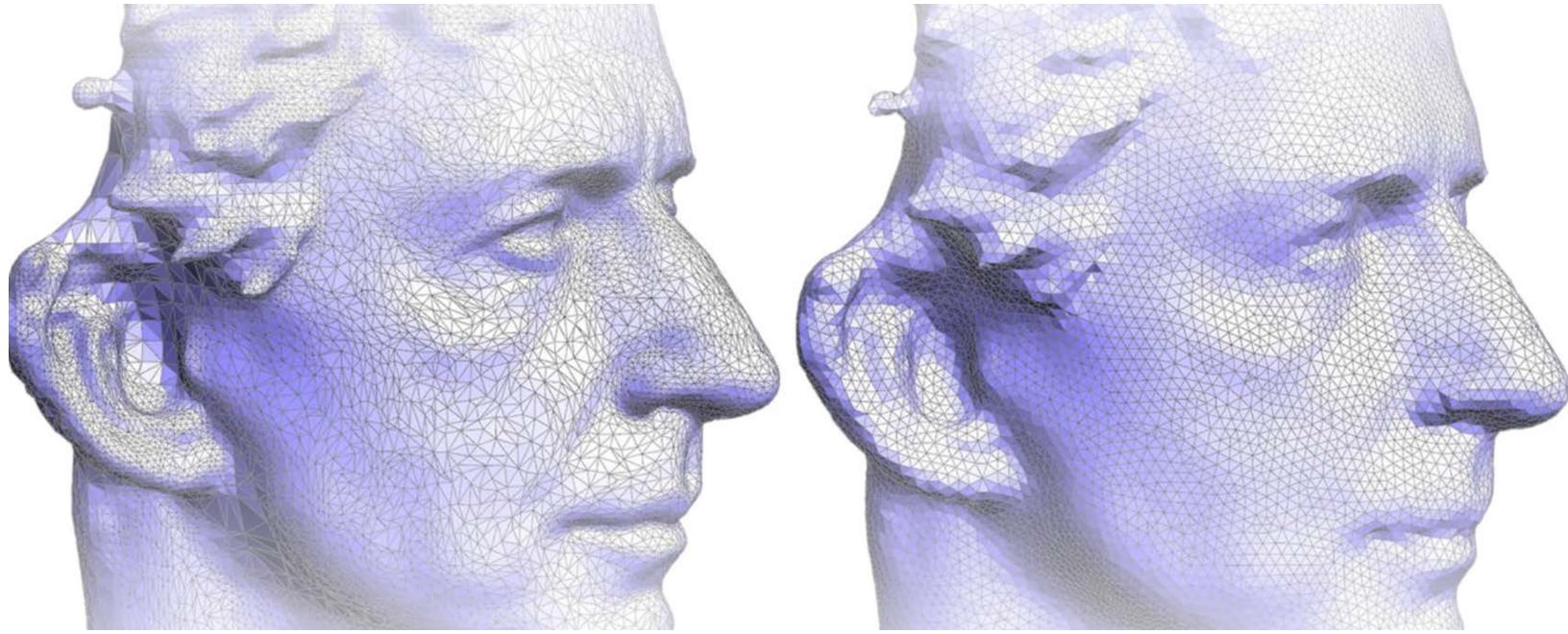
几何处理：下采样 DownSampling

- 降低分辨率；尽量保持形状/外观
- 图像：最近邻插值、双线性插值、双三次插值
- 点云：子采样（只取较少的点）
- 多边形网格：迭代简化，变分形状逼近，.....



几何处理：重采样 Resampling

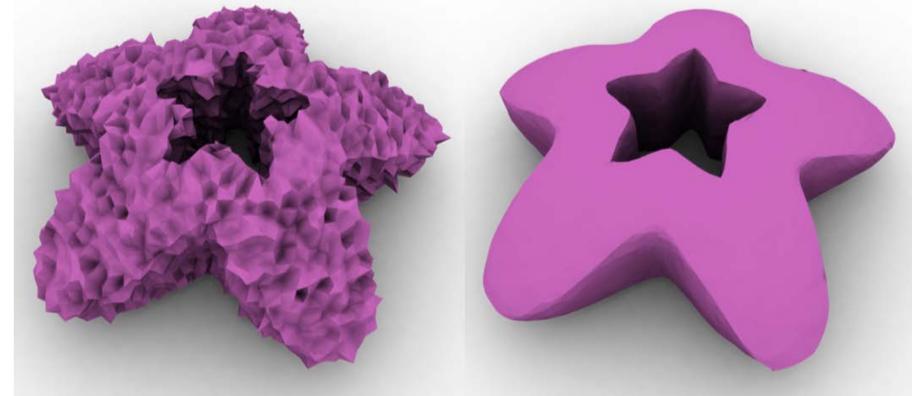
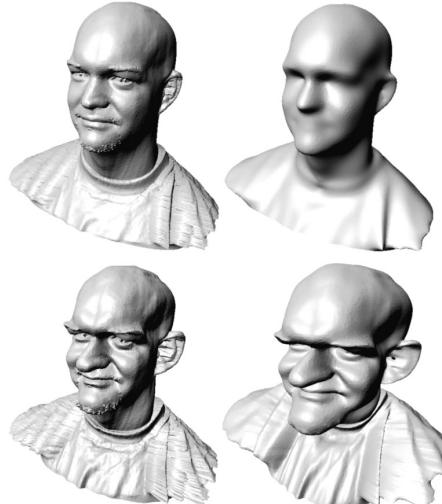
- 修改采样的分布以提高质量
- 图像：不是问题，像素总是存储在规则网格上
- 网格：多边形的形状极其重要！
 - 根据任务有不同的“质量”概念
 - 例如，可视化（更多细节） vs. 求解方程（正三角形）



几何处理：过滤 Filtering

- 去除噪声，或强调重要特征（例如，边缘）
- 图像：模糊、双边滤波（降噪）、边缘检测等
- 多边形网格：

- 曲率流 curvature flow
- 双边滤波 bilateral filter
- 频谱滤波 spectral filter



几何处理：压缩 Compression

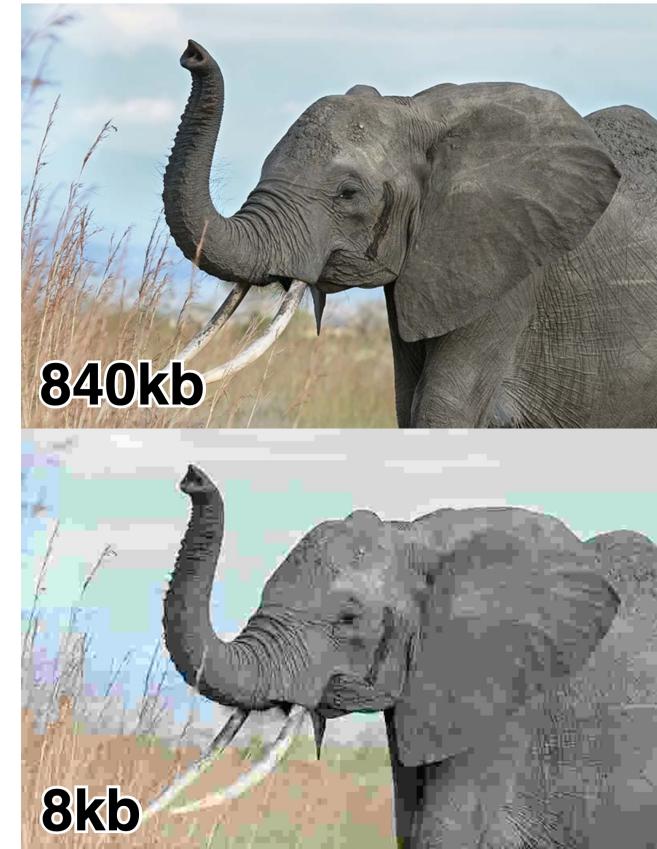
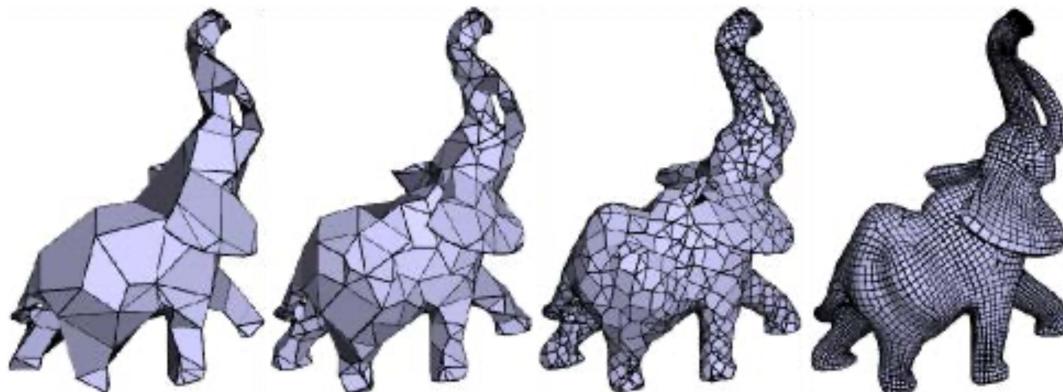
通过消除冗余数据/近似不重要数据来减少存储大小

图像：

- 无损：游程、霍夫曼编码
- 有损：余弦/小波 (JPEG/MPEG)

多边形网格：

- 压缩几何和连通性 (connectivity)
- 多种技术 (有损和无损)

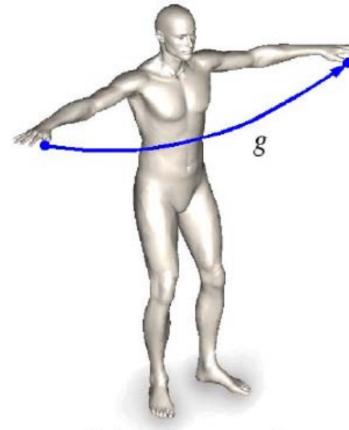
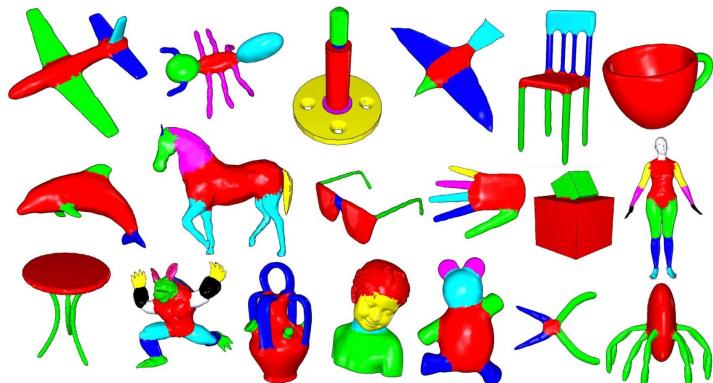


几何处理：形状分析

□识别/理解重要语义特征

□图像：计算机视觉、分割、人脸检测等

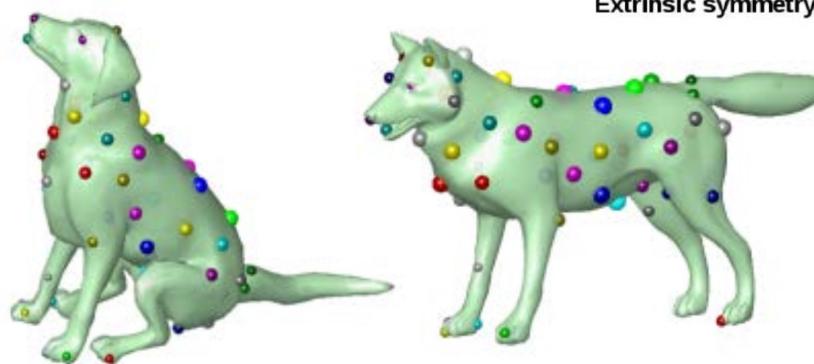
□多边形网格：分割、对应关系、对称性检测等



Extrinsic symmetry



Intrinsic symmetry



Today's topics

□ 几何网格质量分析

□ 网格上采样

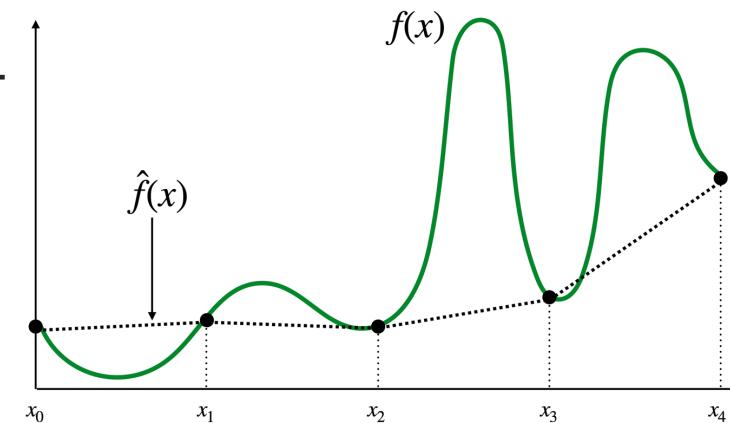
□ 网格下采样 (网格简化)

□ 提升网格质量

重网格化作为重采样

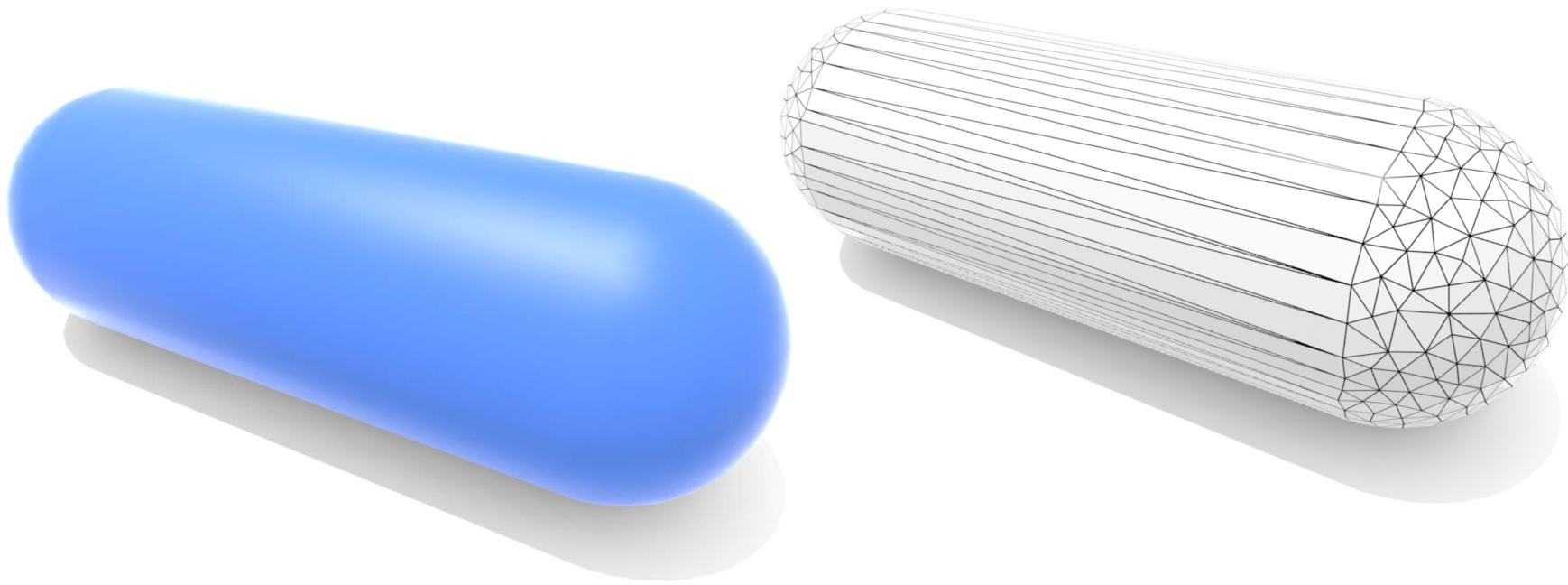
- 口回忆我们关于走样 (aliasing) 的讨论
- 口错误的采样使信号看起来与实际情况不同
- 口比如说，欠采样曲线看起来很平坦
- 口几何也一样！

- 采样不足难以保留特征
- 过采样对性能要求很高



什么构成了一个“好”的网格？

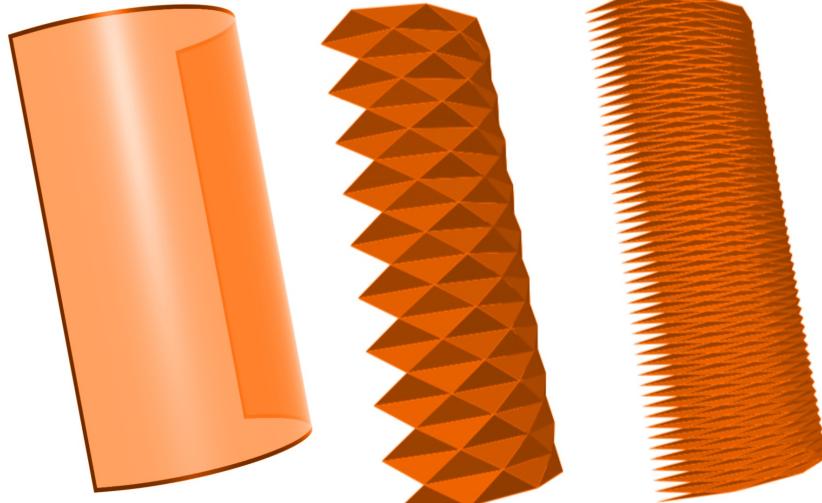
- 一种想法：要尽可能好地模仿原始形状！
- 只保留能够帮助我们了解形状的元素
- 在某些地方，比如曲率很大的地方，需要添加更多的信息



仅靠位置的模拟是不够的！

- 就算网格顶点离模拟表面很近，也不代表就是准确的模拟
- 仍然可能在外观上、面积上等方面出现错误
- 需要考虑其他因素*，例如尽可能精确地模拟表面的法线

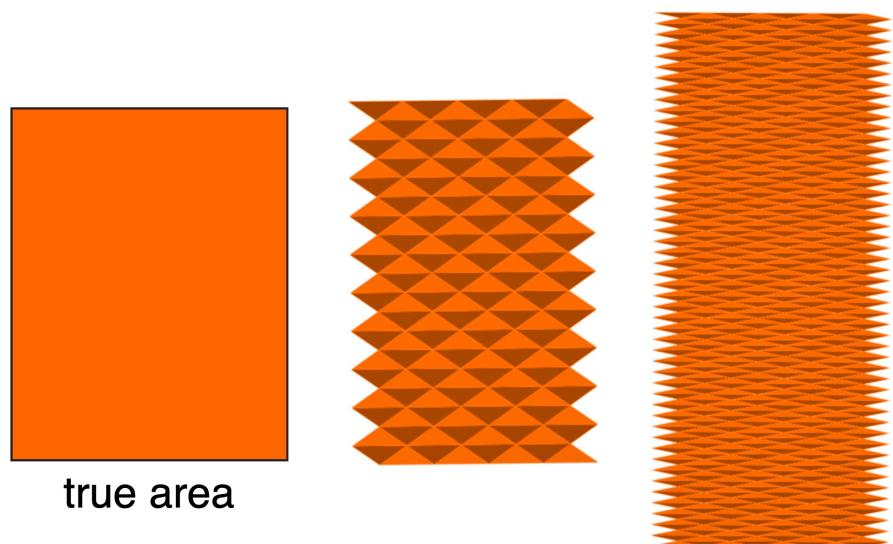
顶点正好在光滑的圆柱体上



smooth cylinder

外观上有差异

光滑圆柱体和网格的平坦化



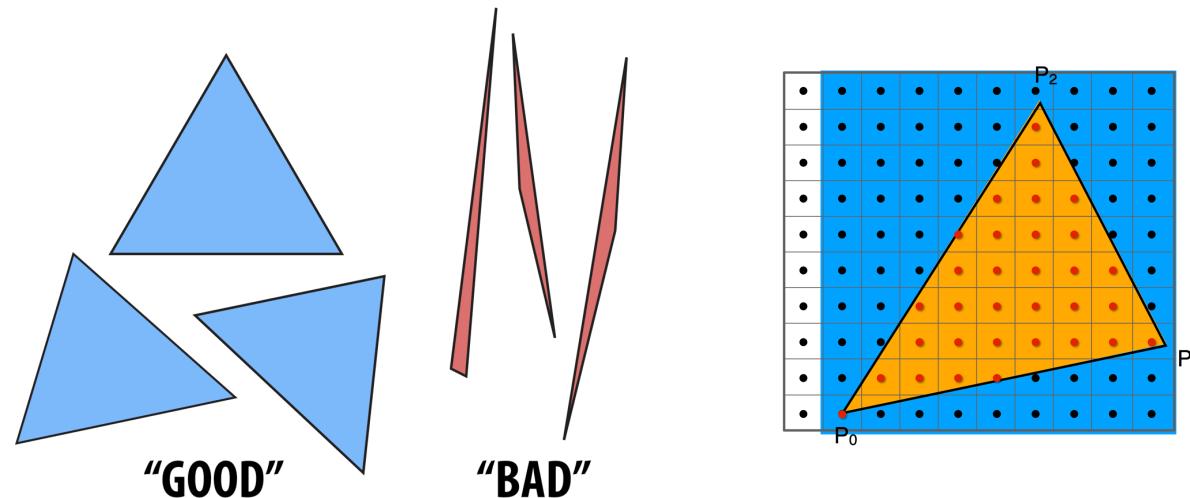
true area

面积上有差异

还有什么能构成一个“好”的网格？

□ 经验法则 1：三角形的形状很重要

- 比如所有的角都接近 60 度
- 更复杂的条件：Delaunay (三角形外接圆内不包含其他点)
 - 提高数值的精确性和稳定性、能最大化最小角度、进行平滑插值...



□ 但并不是所有情况均如此

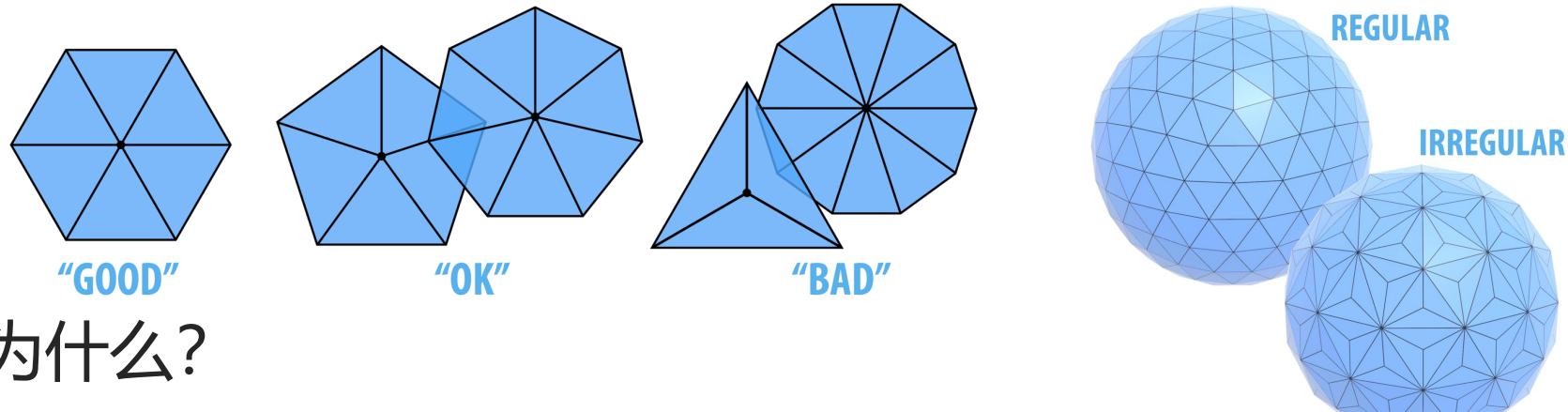
- 过度追求正三角形可能会损失精度
- 有时长和细的三角形就是比较高效，比如网格的精确模拟*

*See Shewchuk, "What is a Good Linear Element"

什么样的三角形网格是好的？

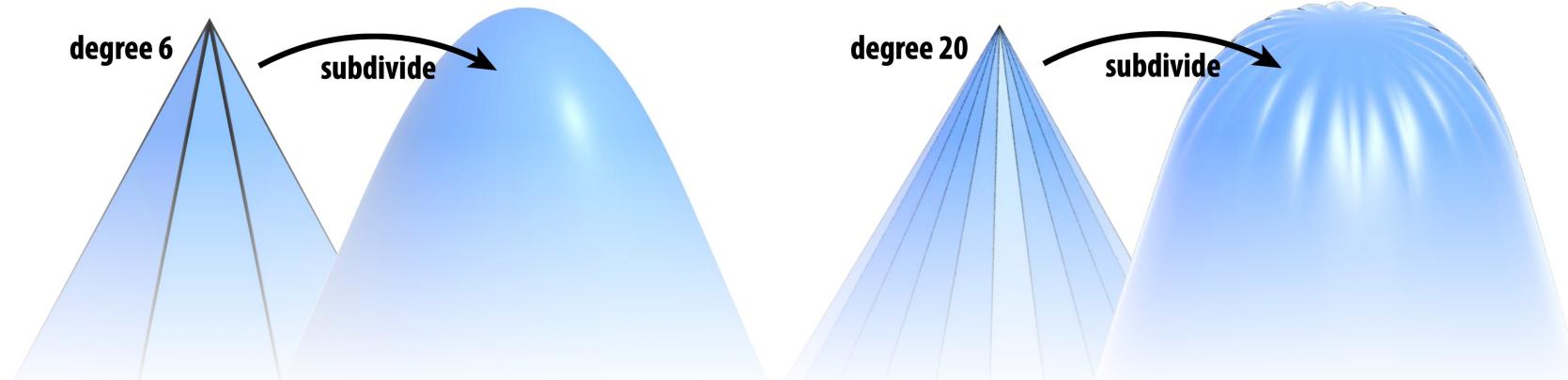
□ 经验规则 2：节点的度也很重要

- 比如三角形网格的节点度接近 6，四边形网格的接近 4



□ 为什么？

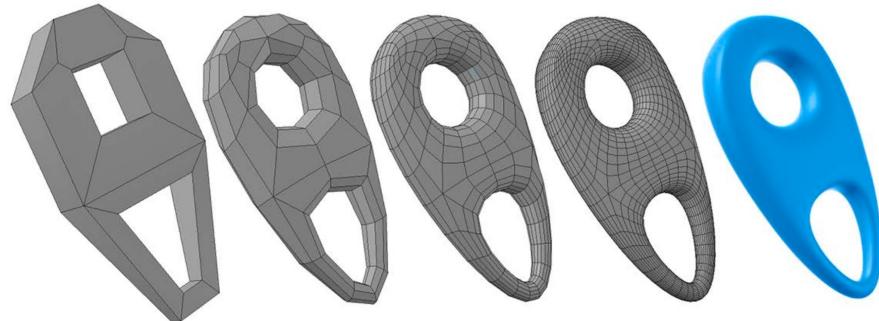
- 更好的多边形形状；更高效的计算；更平滑的细分...



当然，我们不可能让所有地方的节点都有规则的度

如何对网格进行上采样？

通过细分进行上采样



□不断地将每个元素分割成更小的部分

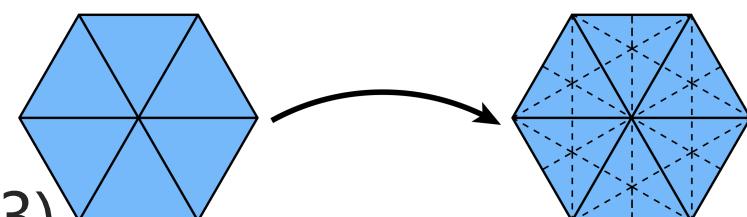
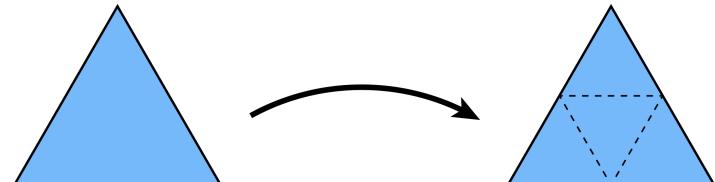
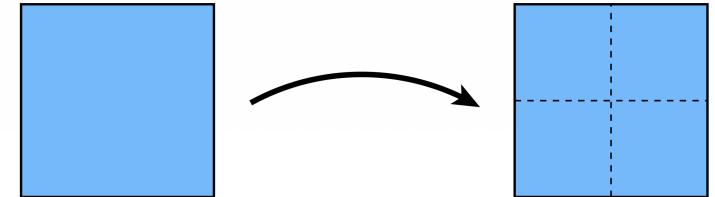
□将顶点位置替换为邻居的加权平均值

□主要的考虑因素：

- 插值与近似
- 限制曲面的连续性 (C^1, C^2, \dots)
- 在不规则顶点处的表现

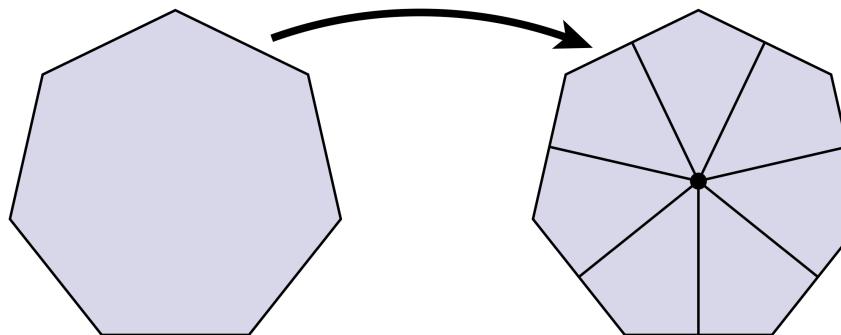
□有许多选择：

- 四边形：Catmull-Clark
- 三角形：Loop, Butterfly, Sqrt(3)

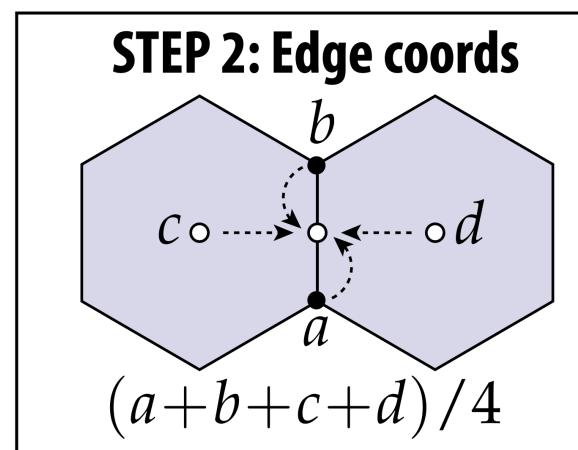
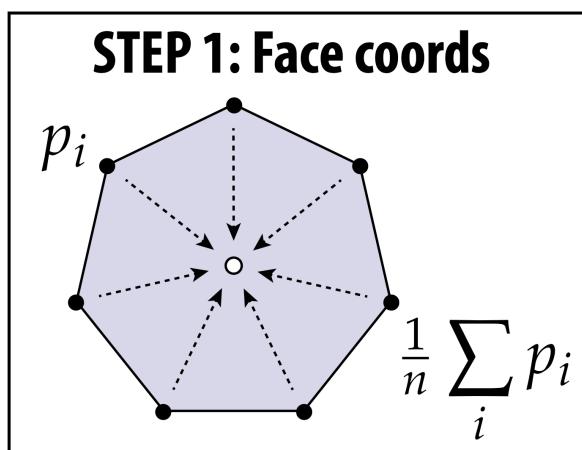


Catmull-Clark 细分 (任意网格)

□ Step 0: 将每个多边形 (任意数量的边) 分成四边形



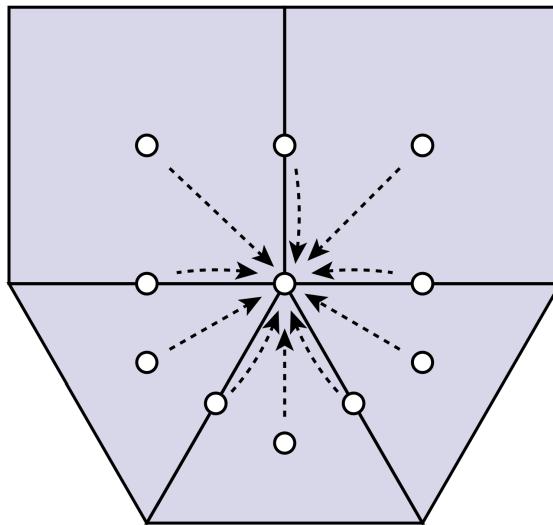
□ 新顶点位置是旧顶点位置的加权组合



Catmull-Clark 细分 (任意网格)

口旧顶点的坐标更新公式

STEP 3: Vertex coords



New vertex coords:

$$\frac{Q + 2R + (n - 3)S}{n}$$

n – vertex degree

Q – average of face coords around vertex

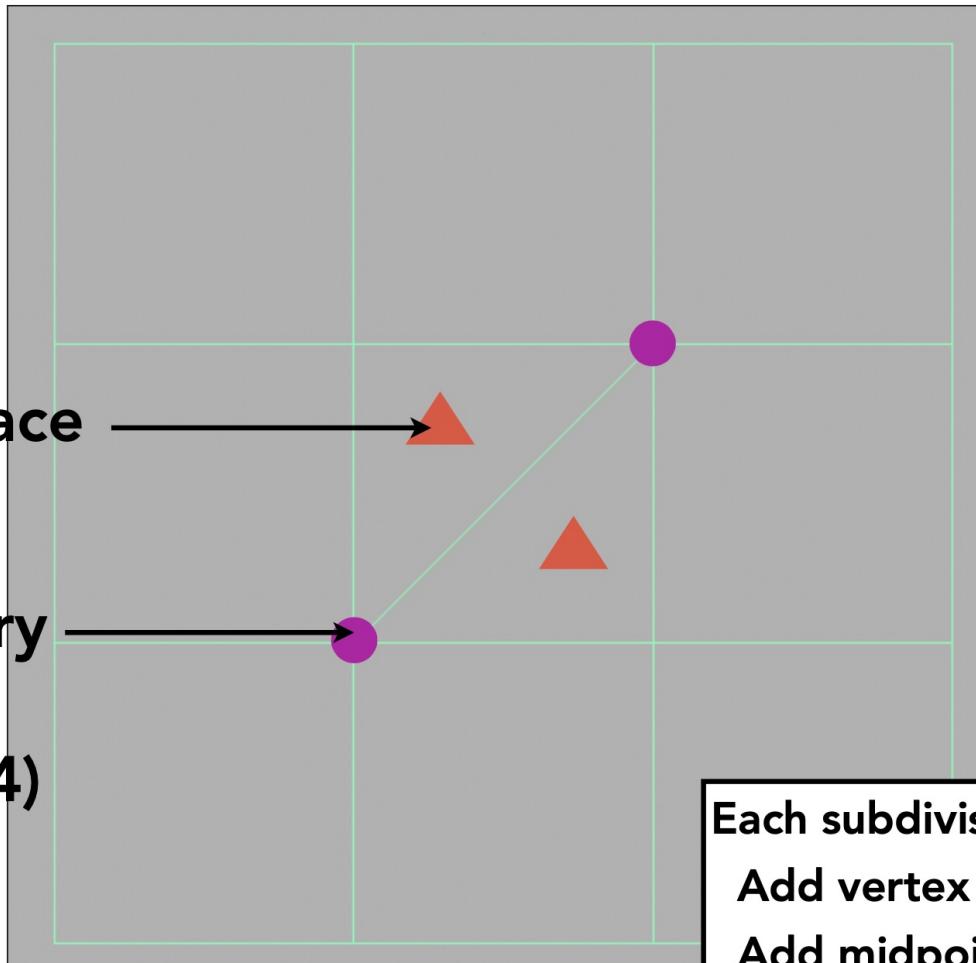
R – average of edge coords around vertex

S – original vertex position

Catmull-Clark 细分

Non-quad face
非四边形面

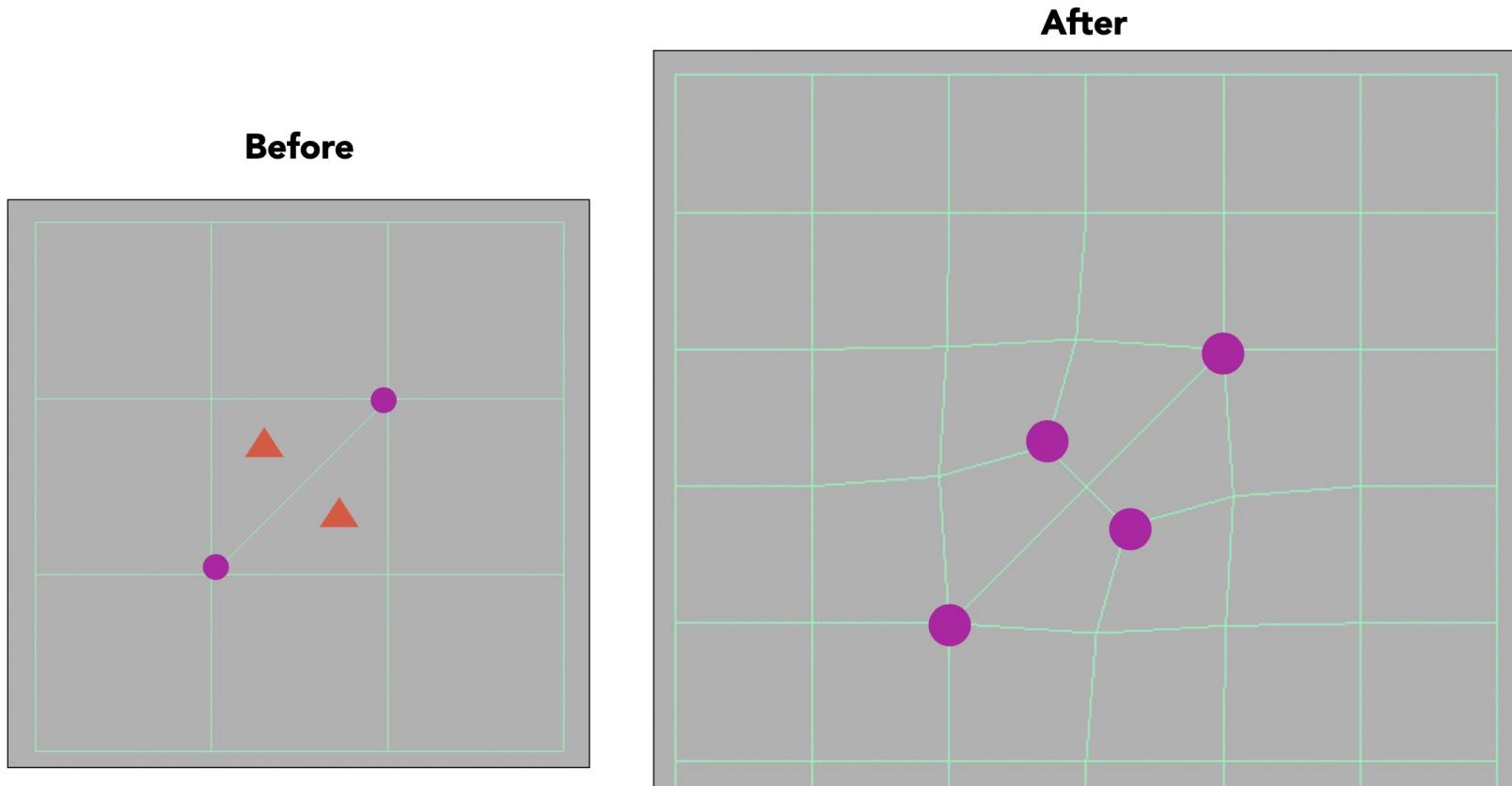
**Extraordinary
vertex**
(degree != 4)
奇异点



Each subdivision step:
Add vertex in each face
Add midpoint on each edge
Connect all new vertices

将边界上的点与面上的点连接

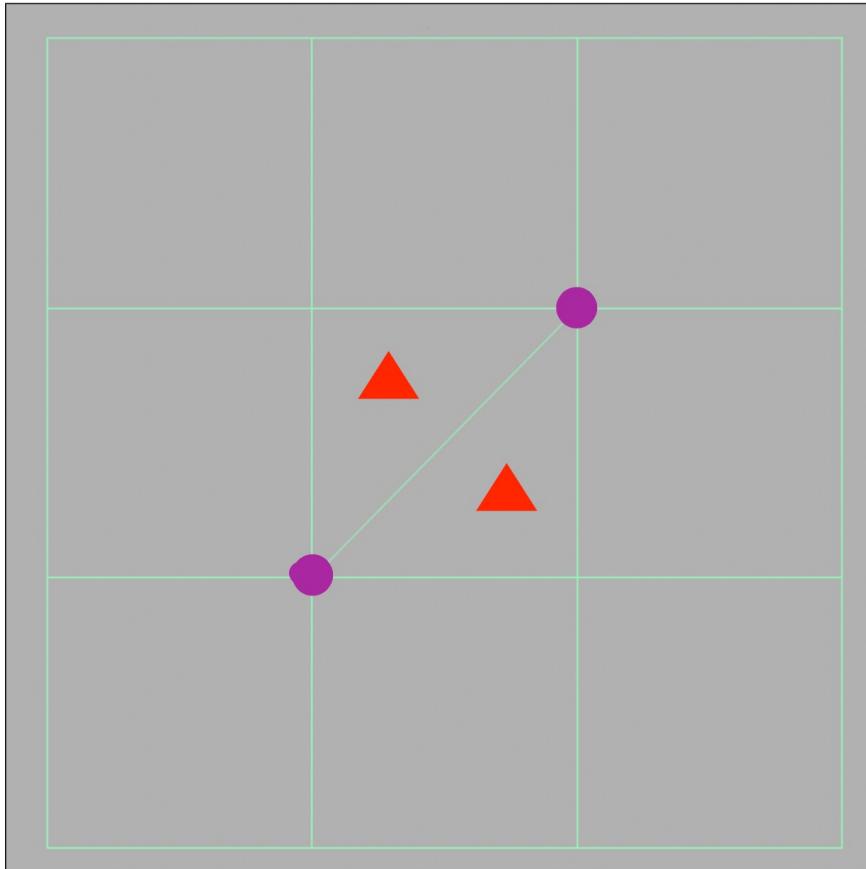
Catmull-Clark 细分



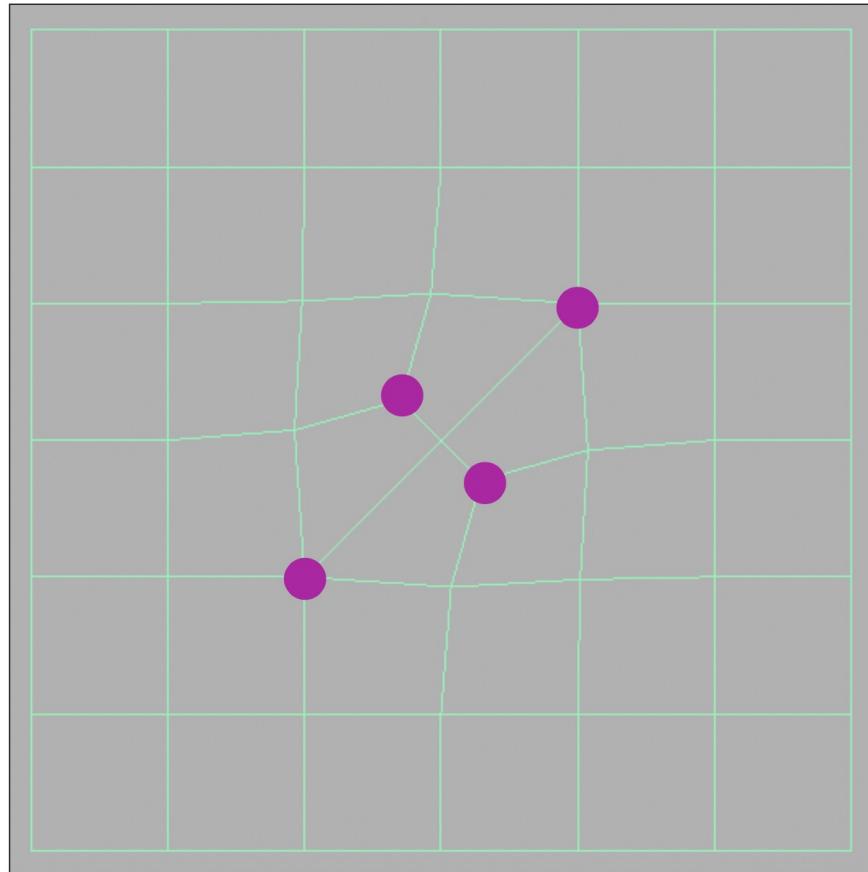
**How many extraordinary vertices after first subdivision?
What are their degrees?
How many non-quad faces?**

不考虑边界情况

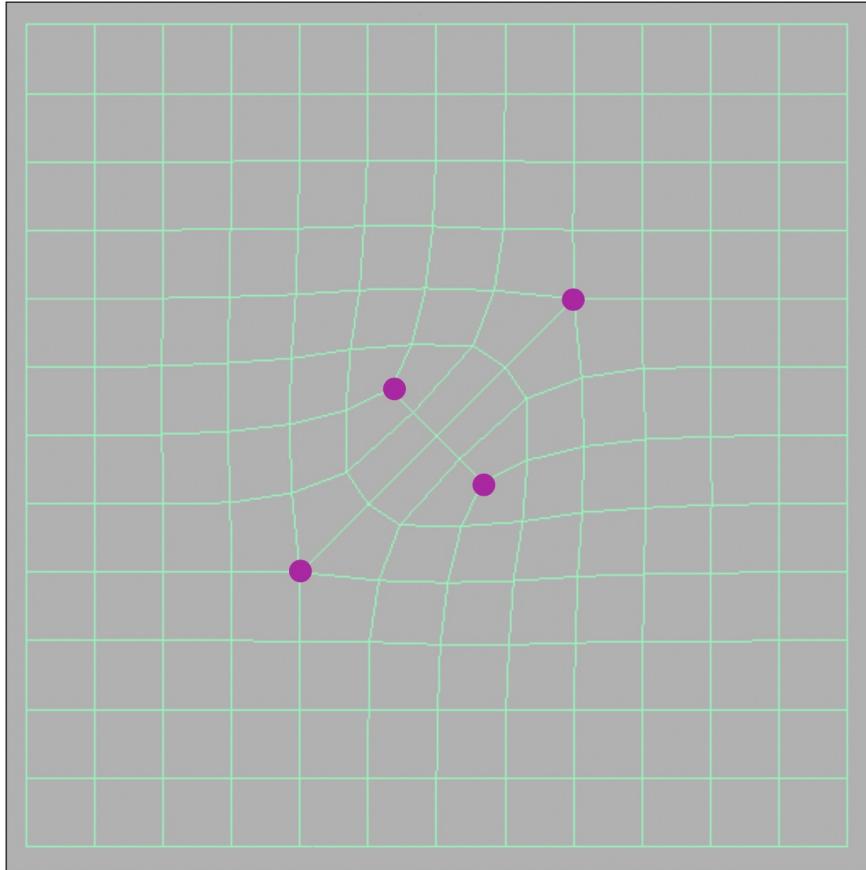
Catmull-Clark 细分



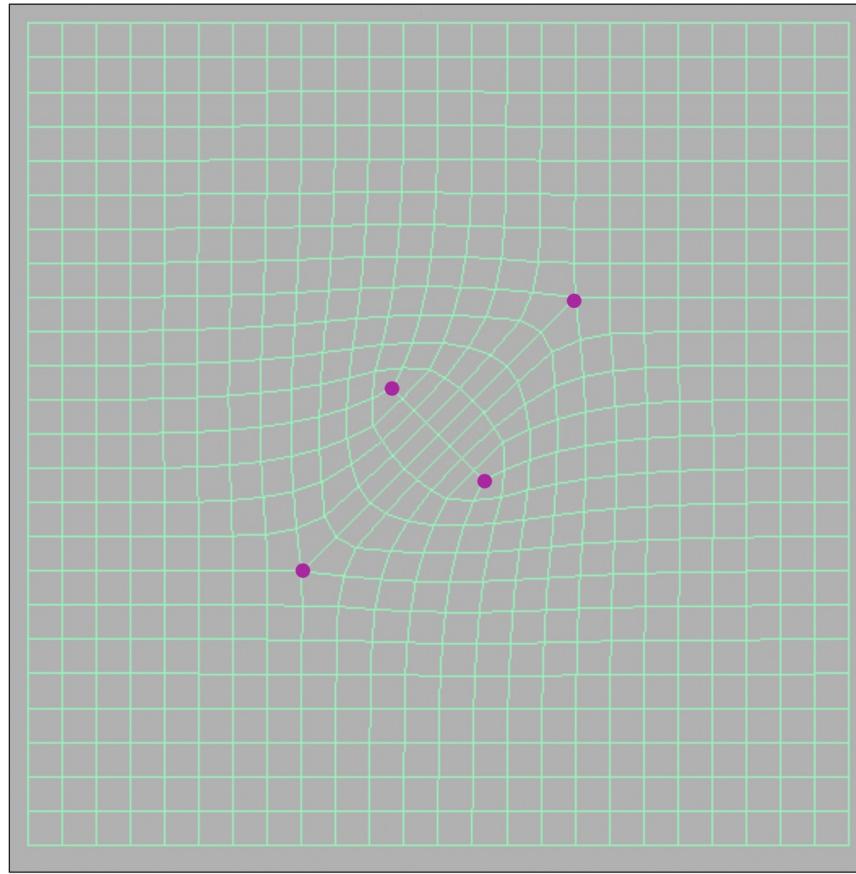
Catmull-Clark 细分



Catmull-Clark 细分



Catmull-Clark 细分



点更新规则 (四边形网格)

口三类点

- 面上的点 (face points)

$$f = \frac{v_1 + v_2 + v_3 + v_4}{4}$$

- 边上的点

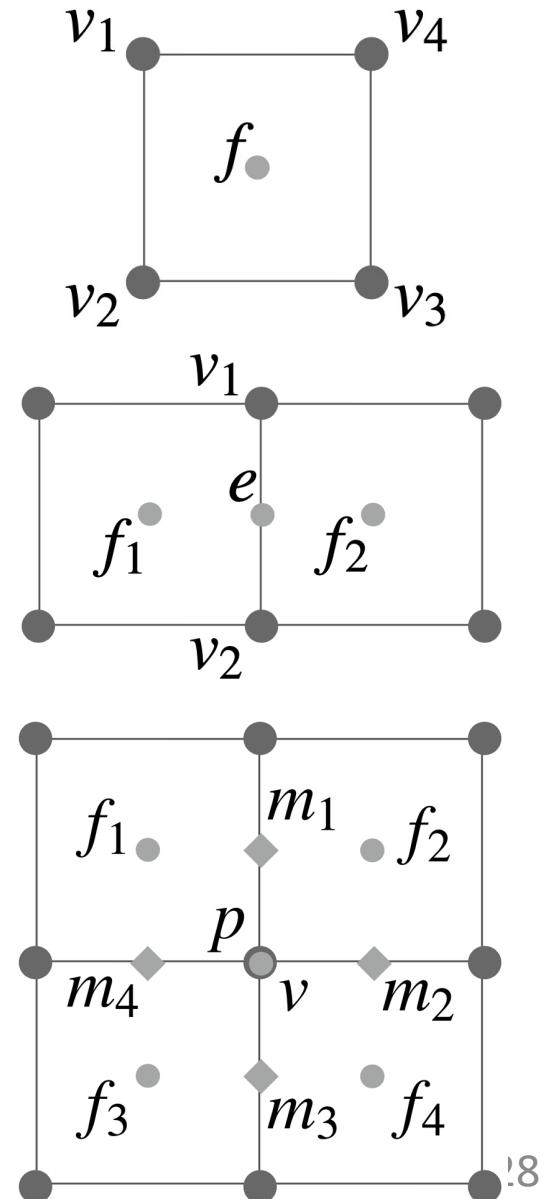
$$e = \frac{v_1 + v_2 + f_1 + f_2}{4}$$

- 旧的点

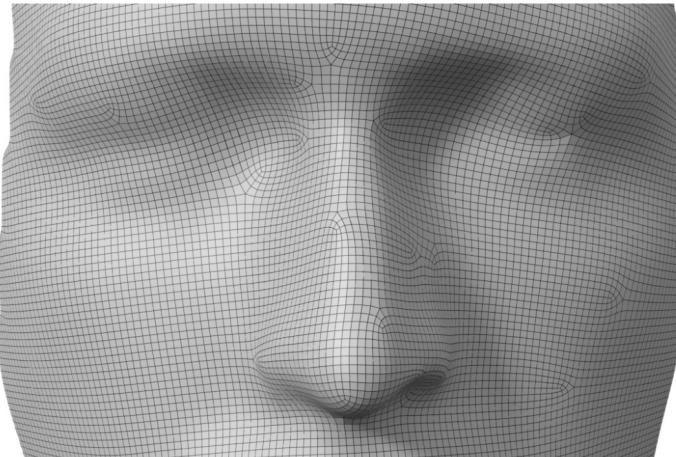
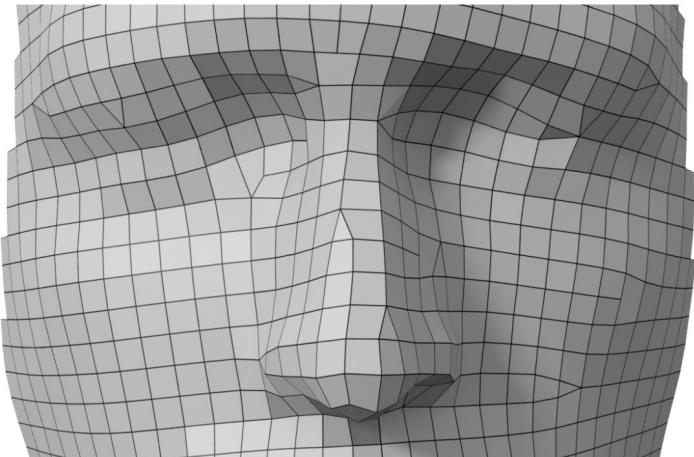
$$v = \frac{f_1 + f_2 + f_3 + f_4 + 2(m_1 + m_2 + m_3 + m_4) + 4p}{16}$$

m midpoint of edge

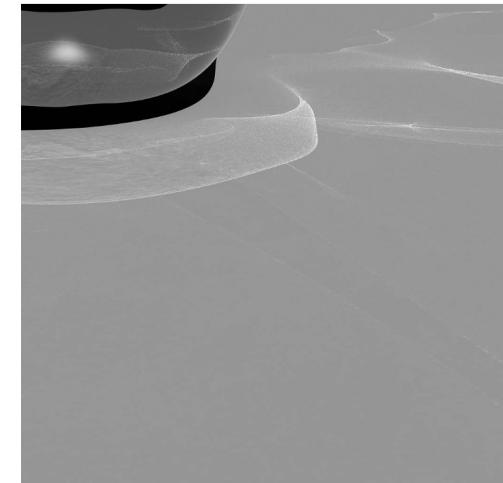
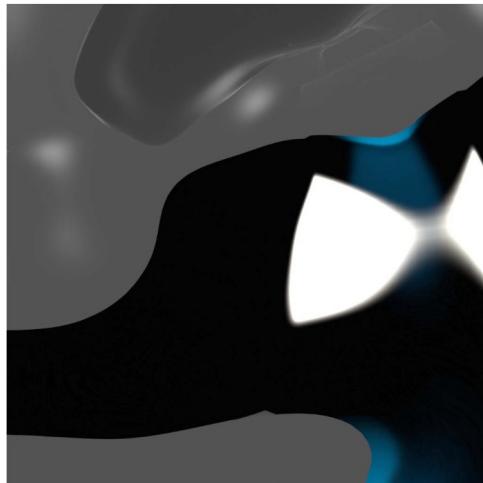
p old “vertex point”



在四边形网格上应用 Catmull-Clark



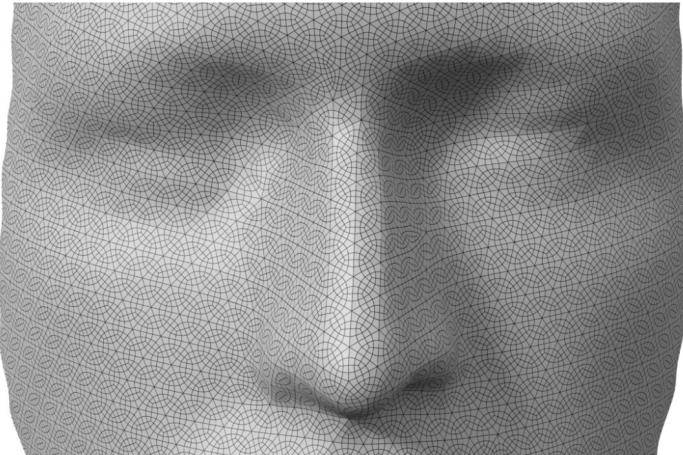
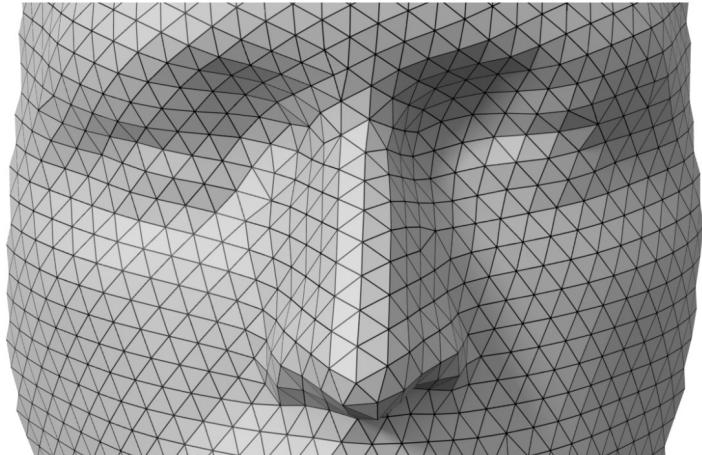
不规则顶点少
=> 表面法线变化平滑



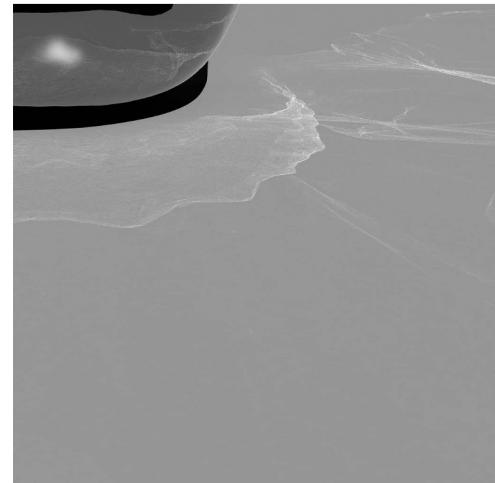
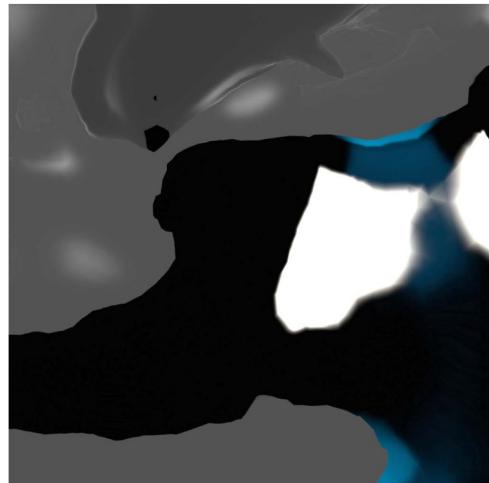
反射线平滑

焦散 (caustics) 平滑

在三角形网格上应用 Catmull-Clark



**不规则顶点多
=> 表面法线变化不规则**



反射线呈锯齿状

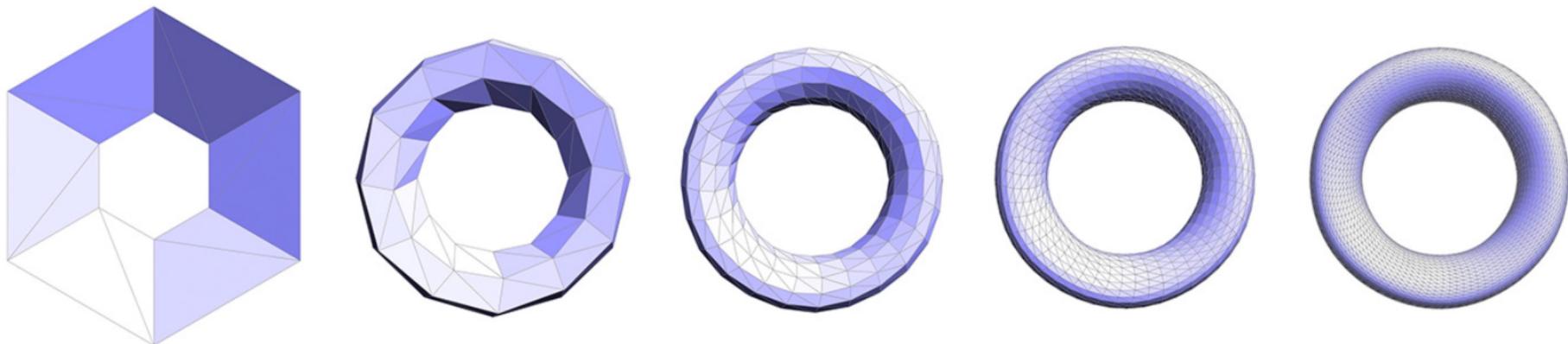
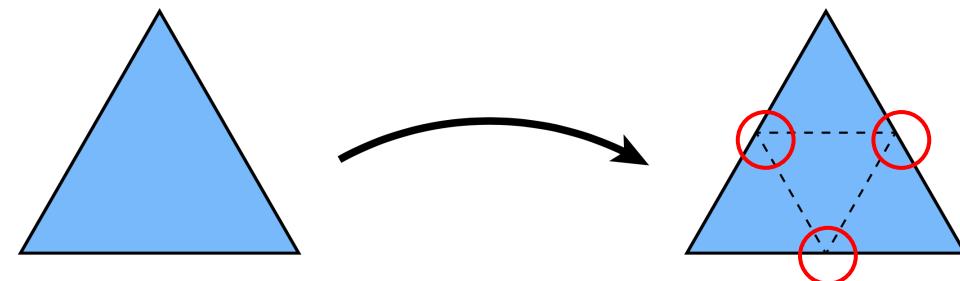
焦散呈锯齿状

Loop 细分

- 针对三角形网格的替代细分方案
- 曲率在不规则顶点之外是连续的 (C^2)

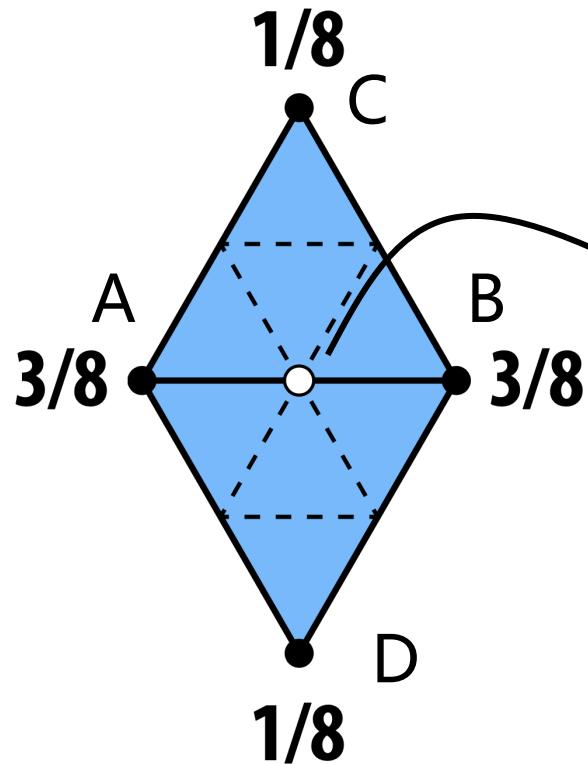
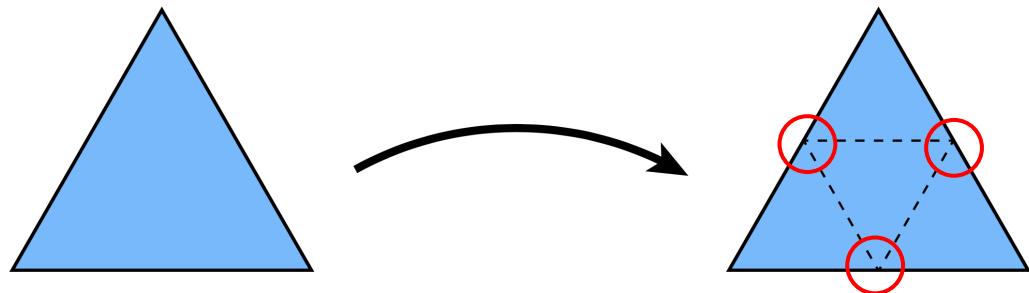
□ 算法：

- 将每个三角形分割成四个
- 根据权重分配新/旧顶点的位置



Loop 细分 – 更新

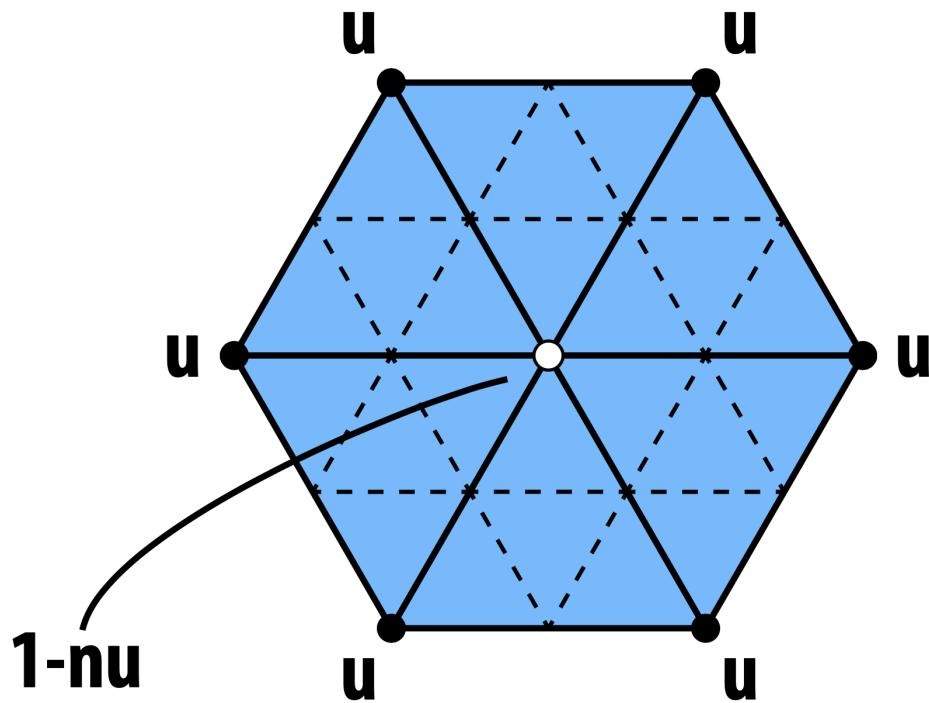
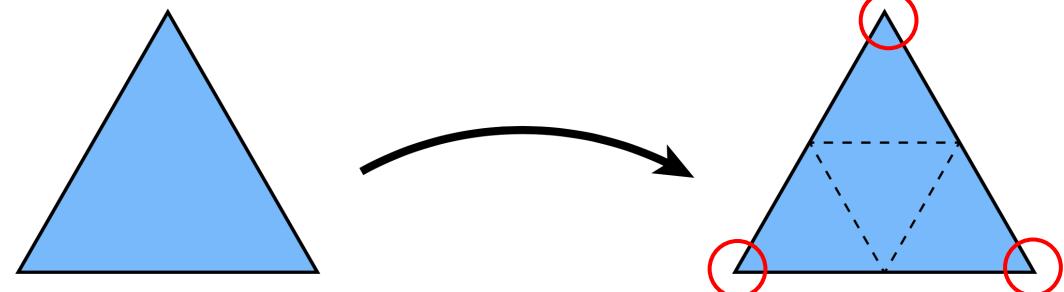
更新新节点的位置



Update to:
 $3/8 * (A + B) + 1/8 * (C + D)$

Loop 细分 – 更新

口更新旧节点的位置

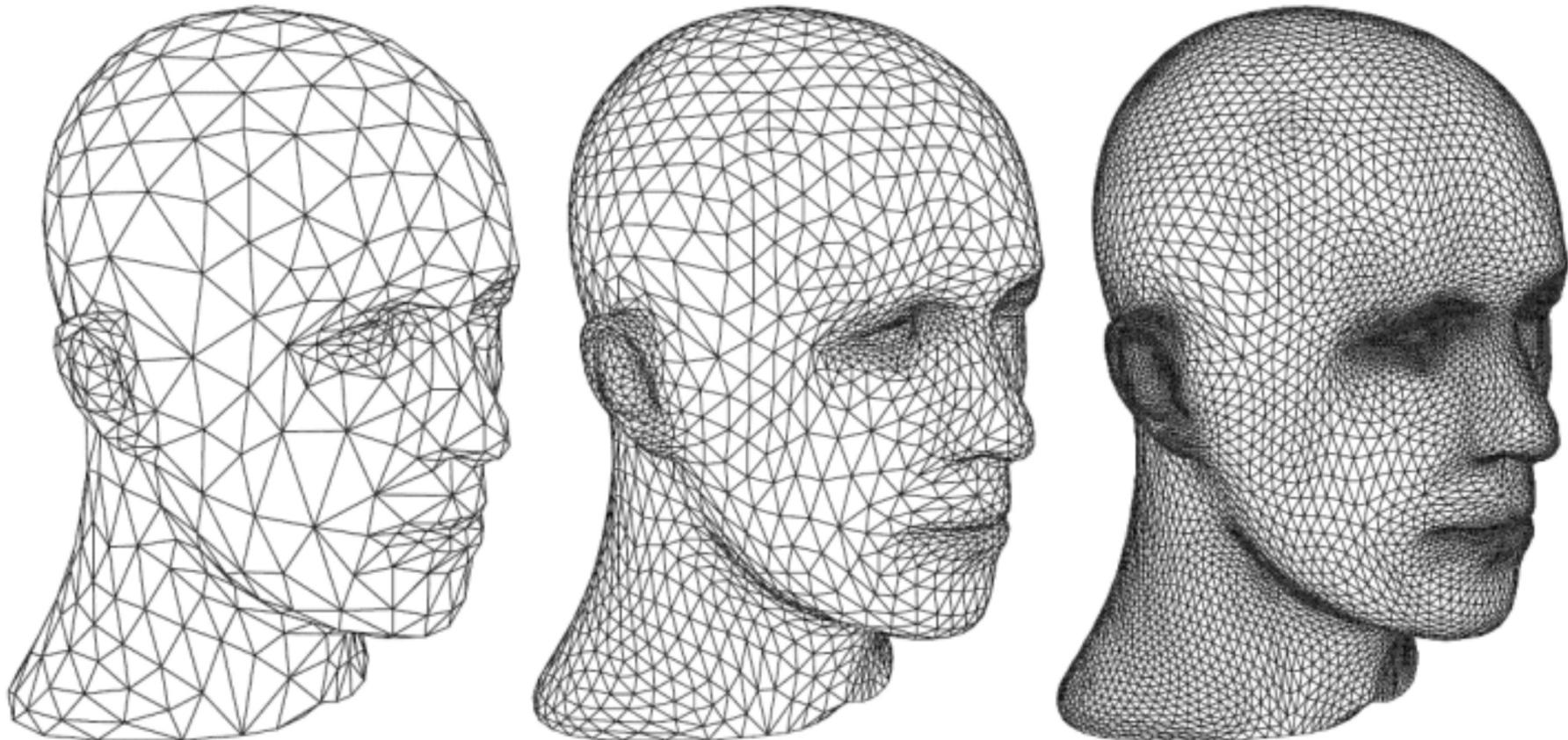


Update to:
 $(1 - n*u) * \text{original_position} + u * \text{neighbor_position_sum}$

n: vertex degree

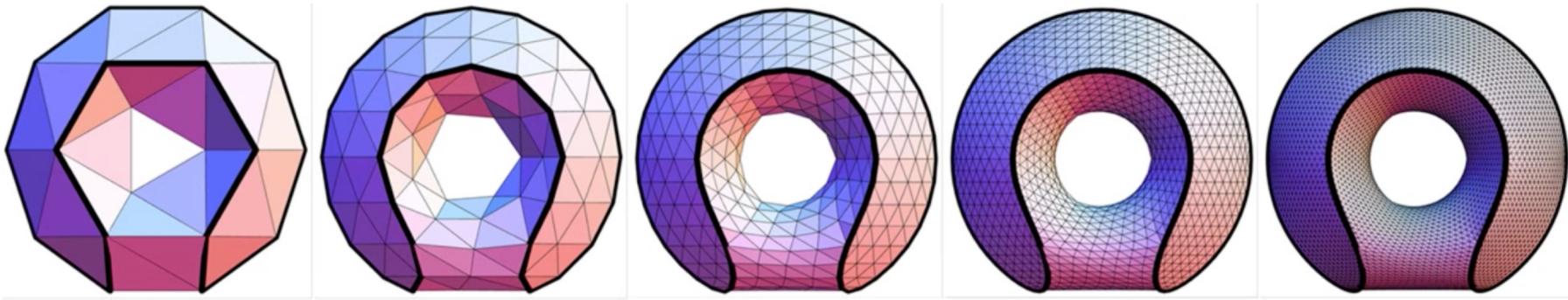
u: $3/16$ if $n=3$, $3/(8n)$ otherwise

Loop 细分 – 结果

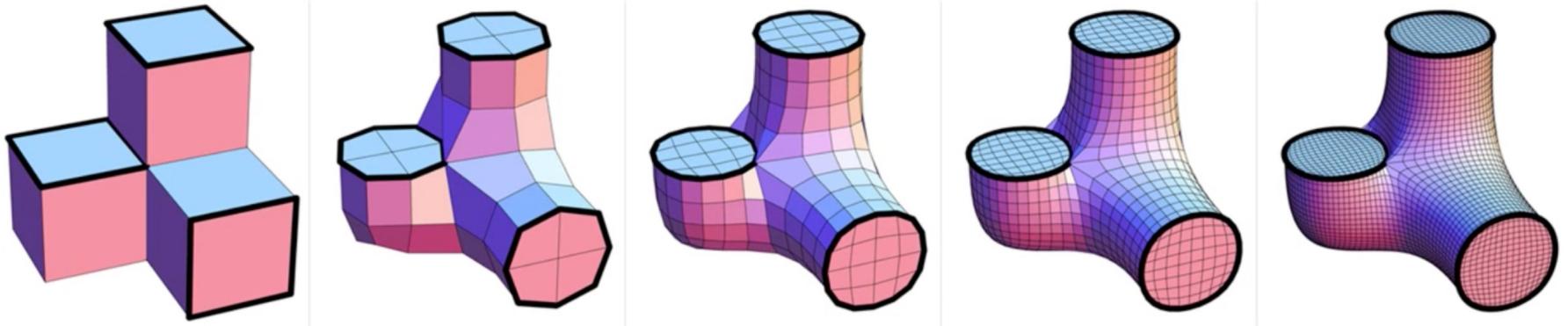


Loop 细分与 Catmull-Clark 细分

Loop 细分 (带有锋利的褶皱)



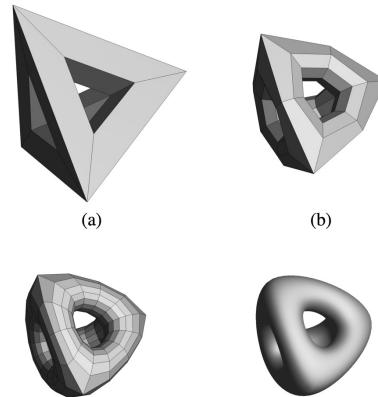
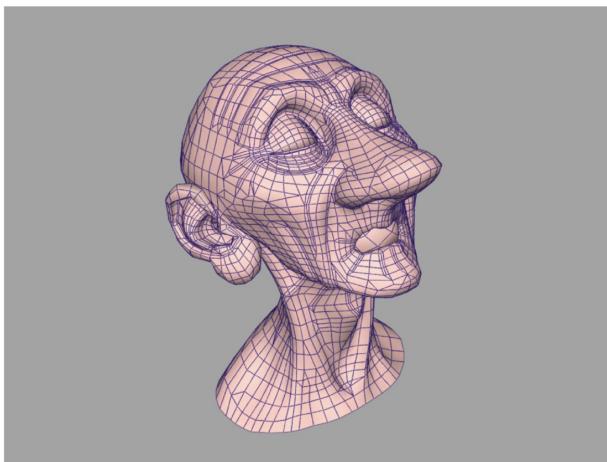
Catmull-Clark 细分 (带有锋利的褶皱)



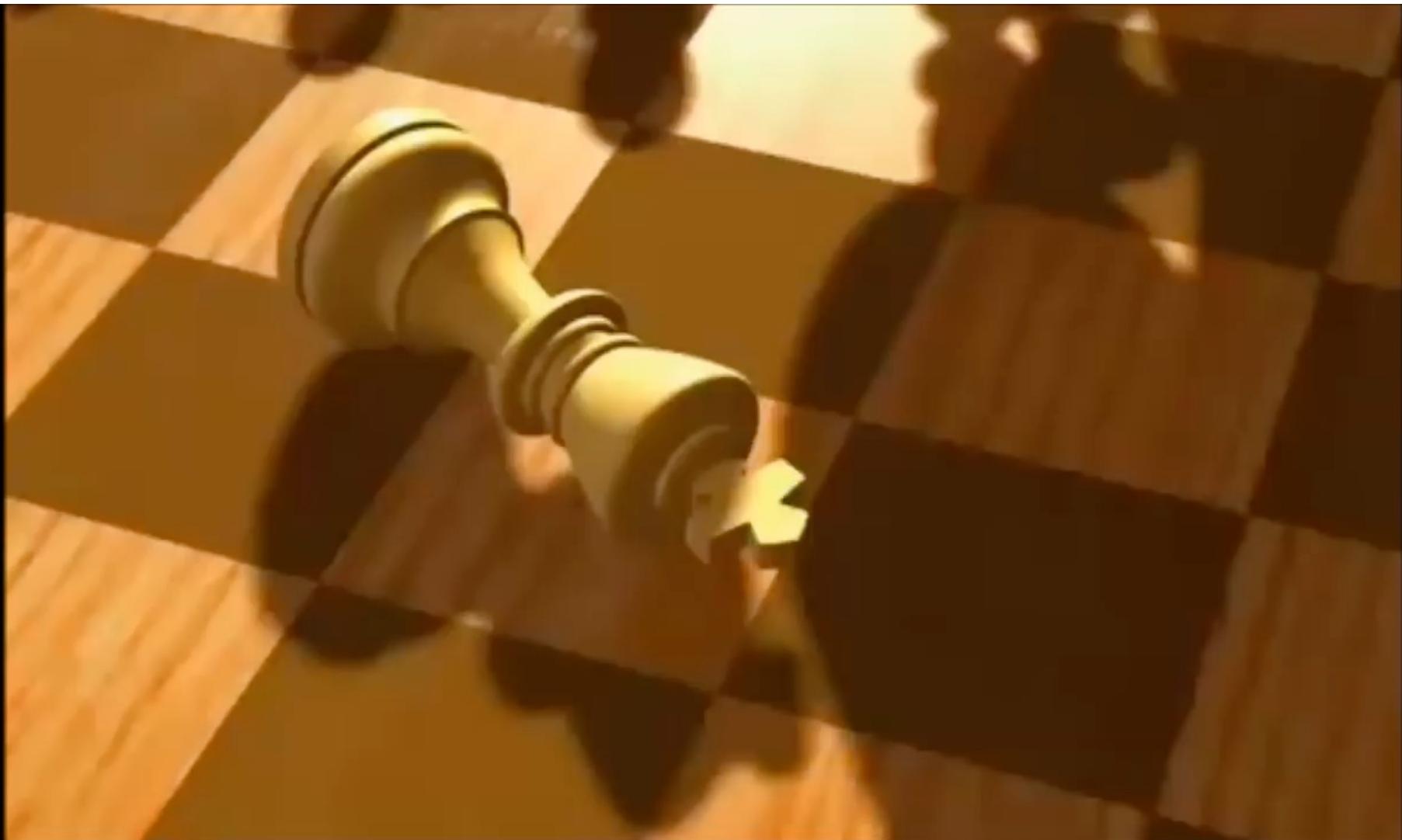
细分应用 – 皮克斯的《格里的游戏》

口细分被应用于角色的各个部分：

- 手和头
- 衣服、领带、鞋子
- ...

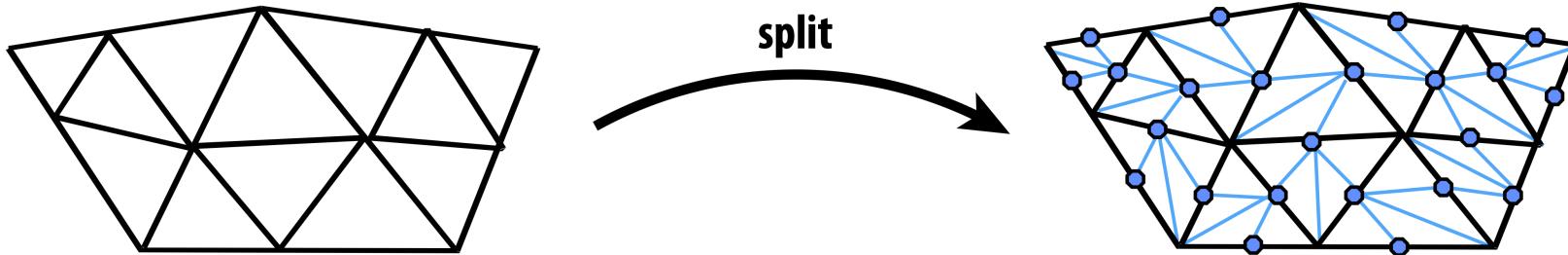


细分应用 – 皮克斯的《格里的游戏》

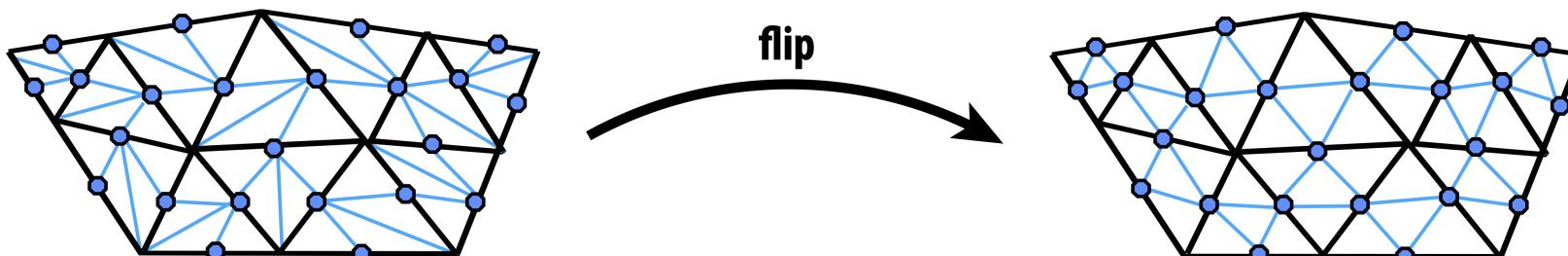


通过边缘操作进行 Loop 细分

□首先，以任意顺序分割原始网格的边缘



□接下来，翻转触及新旧顶点的新边



□不要忘记更新顶点位置！

如果我们想要更少的三角形呢？

网格简化 (Mesh Simplification)

□ 目标：减少多边形的数量

□ 同时尽量保持整体的形状和外观



30,000 triangles



3,000



300



30

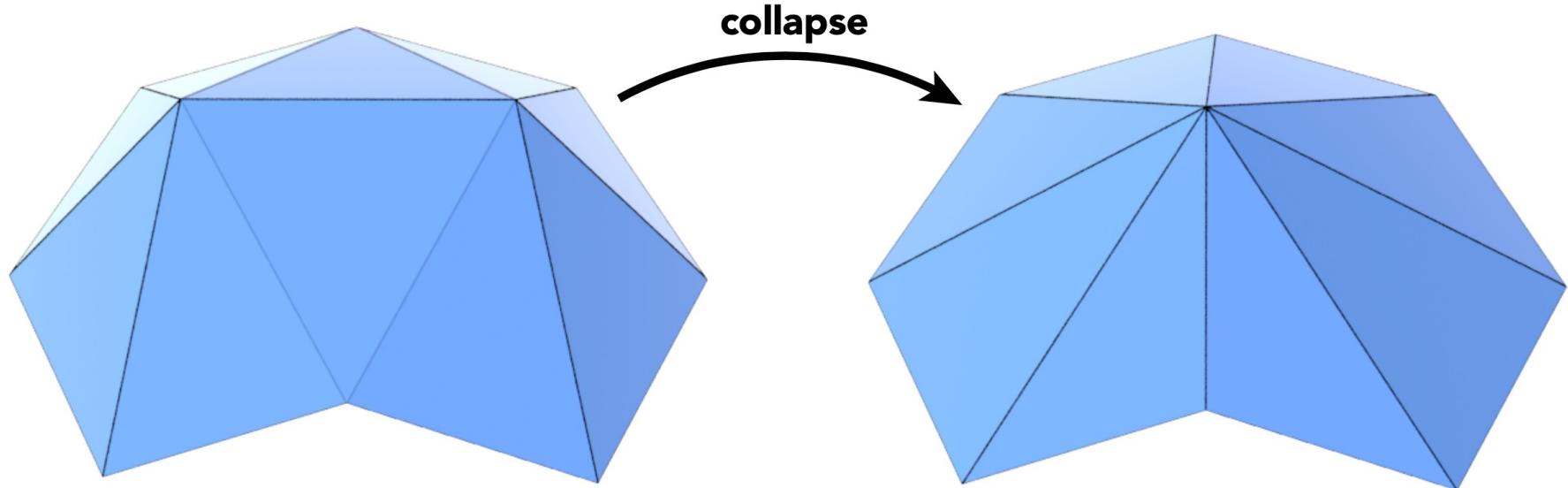


怎么进行计算？

一个常用的算法：边折叠

口边折叠 (edge collapsing) 算法 (贪心算法)

- 为每条边计算一个代价 (衡量折叠此边对整体的影响)
- 折叠具有最小代价的边 (尽可能保持原来的形状)
- 重新计算因折叠被影响的邻居边的代价
- 重复上述过程，直至达到预定数量的多边形



怎么计算每条边的代价？

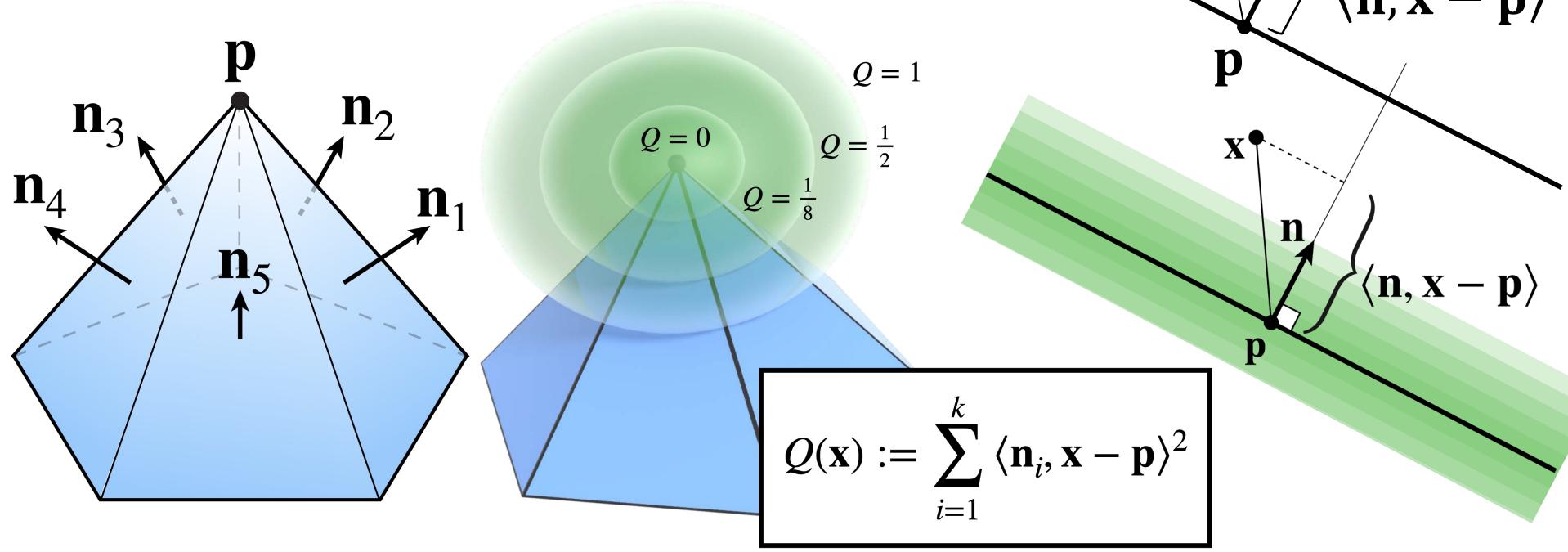
二次误差度量 (Quadric Error Metrics)

□ 计算到一组三角形的距离 (distance to a collection of triangles)

□ Q: 如何计算到通过点 p 且法线为 n 的平面的距离?

□ A: $dist(x) = \langle n, x - p \rangle$

□ 二次误差则为点到平面距离的平方和:



二次误差 – 齐次坐标

□ 假设在坐标中，我们有：

- 一个查询点 $\mathbf{x} = (x, y, z)$
- 一个法向量 $\mathbf{n} = (a, b, c)$
- 一个偏置项 $d := \langle \mathbf{n}, \mathbf{p} \rangle \quad K =$

□ 在齐次坐标中，令

- $\mathbf{u} := (x, y, z, 1), \mathbf{v} := (a, b, c, d)$

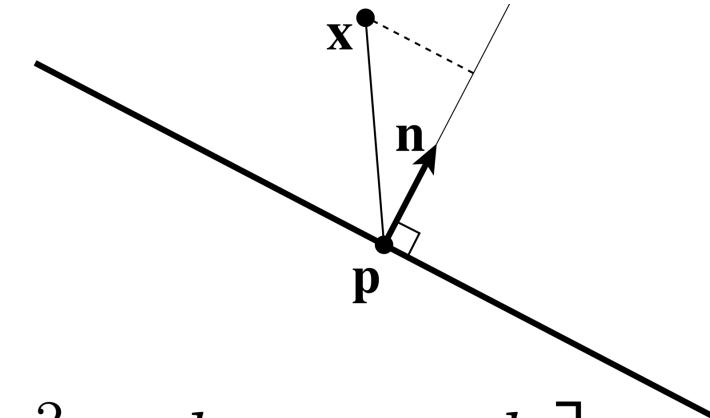
□ 距离平面的**有符号距离**为 $\langle \mathbf{u}, \mathbf{v} \rangle = ax + by + cz + d$

□ 距离平方为 $\langle \mathbf{u}, \mathbf{v} \rangle^2 = \mathbf{u}^T (\mathbf{v} \mathbf{v}^T) \mathbf{u} =: \mathbf{u}^T K \mathbf{u}$

□ 矩阵 $K = \mathbf{v} \mathbf{v}^T$ 编码了到平面的平方距离

Key idea: 多个矩阵 K 的和 \Leftrightarrow 到多个平面的距离之和

$$\mathbf{u}^T K_1 \mathbf{u} + \mathbf{u}^T K_2 \mathbf{u} = \mathbf{u}^T (K_1 + K_2) \mathbf{u}$$



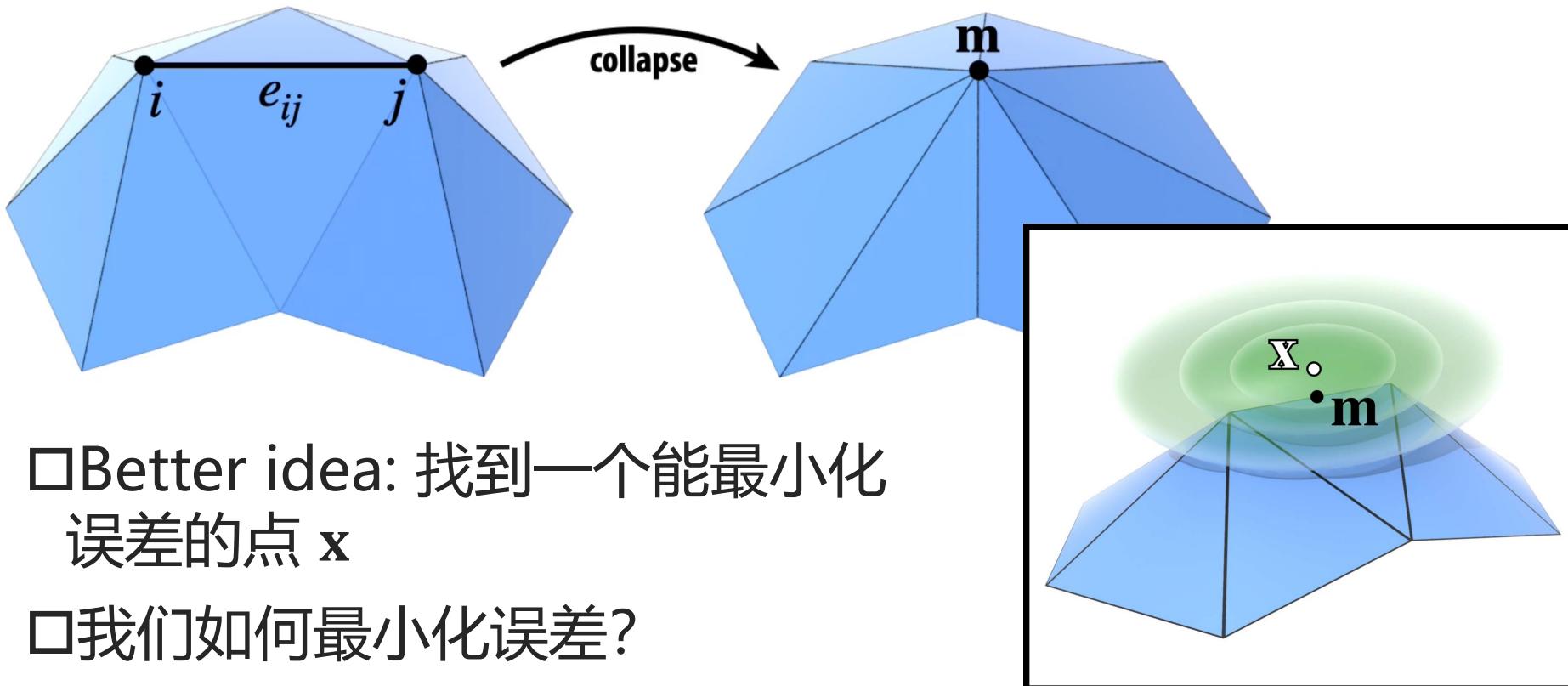
$$K = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$$

边折叠的二次误差

□ 折叠一条边 e_{ij} 的代价是多少？

□ Idea: 计算中点 \mathbf{m} , 测量误差 $Q(\mathbf{m}) = \mathbf{m}^T(K_i + K_j)\mathbf{m}$

□ 误差作为边 e_{ij} 的“分数”，决定其优先级



回顾：最小化二次函数

□ 假设你有一个函数 $f(x) = ax^2 + bx + c$

□ Q：这个函数的图像是什么样的？

□ 也可能是这样的！

□ Q：我们如何找到最小值？

□ A：放大找到函数看起来“平坦”的地方

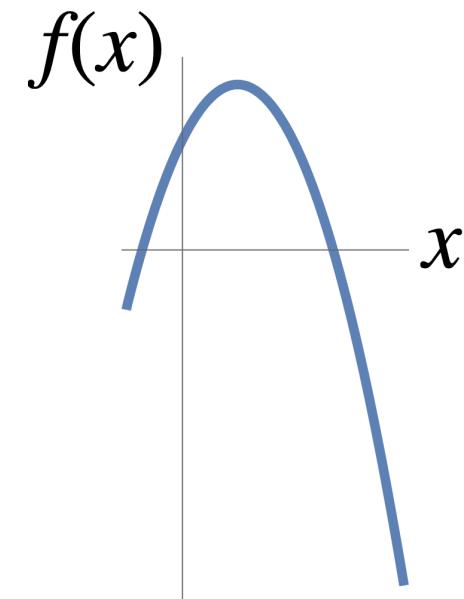
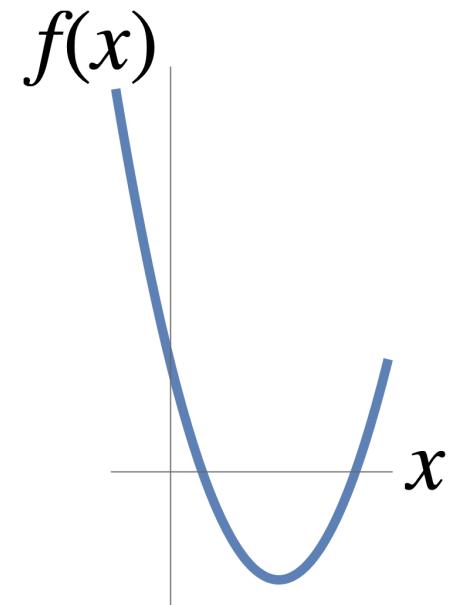
□ 也就是，找到一阶导数为 0 的点

$$f'(x) = 0$$

$$2ax + b = 0$$

$$x = -b/2a$$

□ 对于第二个函数， x 描述的是什么？



最小化二次多项式

□ 最小化具有 n 个变量的二次多项式也是类似的

□ 总是可以用一个**对称矩阵** A 来表示多项式

□ 比如在 2D 中: $f(x, y) = ax^2 + bxy + cy^2 + dx + ey + g$

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} \quad A = \begin{bmatrix} a & b/2 \\ b/2 & c \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} d \\ e \end{bmatrix}$$

$$f(x, y) = \mathbf{x}^T A \mathbf{x} + \mathbf{u}^T \mathbf{x} + g$$

(对任意 n 都有一样的表达式)

□ 问题: 我们如何找到一个临界点 (最小值/最大值/鞍点) ?

□ 答: 将导数设为零! $2A\mathbf{x} + \mathbf{u} = 0$ 与 1D 的结果比较

$$\mathbf{x} = -\frac{1}{2}A^{-1}\mathbf{u} \quad x = -b/2a$$

正定二次型 positive definite quadratic form

□正如 1D 抛物线，临界点 (critical point) 并不总是最小值

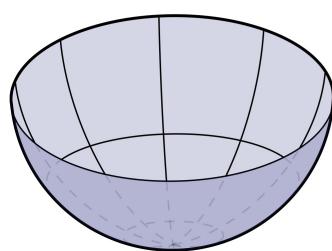
□Q：在 2D, 3D, ..., nD 中，我们如何得到最小值？

□A：当矩阵 A 是正定的时候：

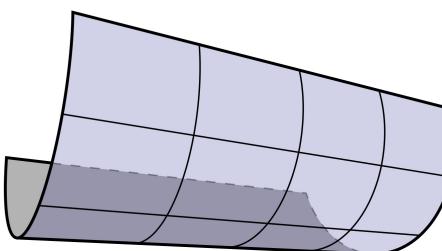
$$\mathbf{x}^T A \mathbf{x} > 0, \forall \mathbf{x}$$

□1D：必须有 $xax = ax^2 > 0$ ，即 $a > 0$

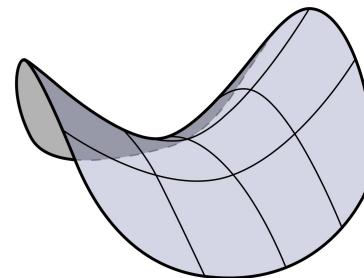
□2D：函数的图像看起来像一个"碗"



positive definite



positive semidefinite



indefinite

□正定性在图形学中极其重要：意味着能通过解线性方程求解最小值。这是许多算法（几何处理、模拟等）的起点

最小化二次误差

□通过最小化二次形式找到边折叠的"最佳"点

$$\min_{\mathbf{u} \in \mathbb{R}^4} \mathbf{u}^T K \mathbf{u}$$

□我们已经知道一个点的第四个（齐次）坐标是1

□所以，我们将二次函数分解为两部分：

$$\begin{aligned} & [\mathbf{x}^T \quad 1] \begin{bmatrix} B & \mathbf{w} \\ \mathbf{w}^T & d^2 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \\ &= \mathbf{x}^T B \mathbf{x} + 2 \mathbf{w}^T \mathbf{x} + d^2 \end{aligned}$$

□现在我们有一个关于未知折叠点 $\mathbf{x} \in \mathbb{R}^3$ 的二次多项式

□可以像之前一样求解最小值：

$$2B\mathbf{x} + 2\mathbf{w} = \mathbf{0} \iff \mathbf{x} = -B^{-1}\mathbf{w}$$

基于二次误差的网格简化

□ 为每个三角形计算 K (到平面的平方距离)

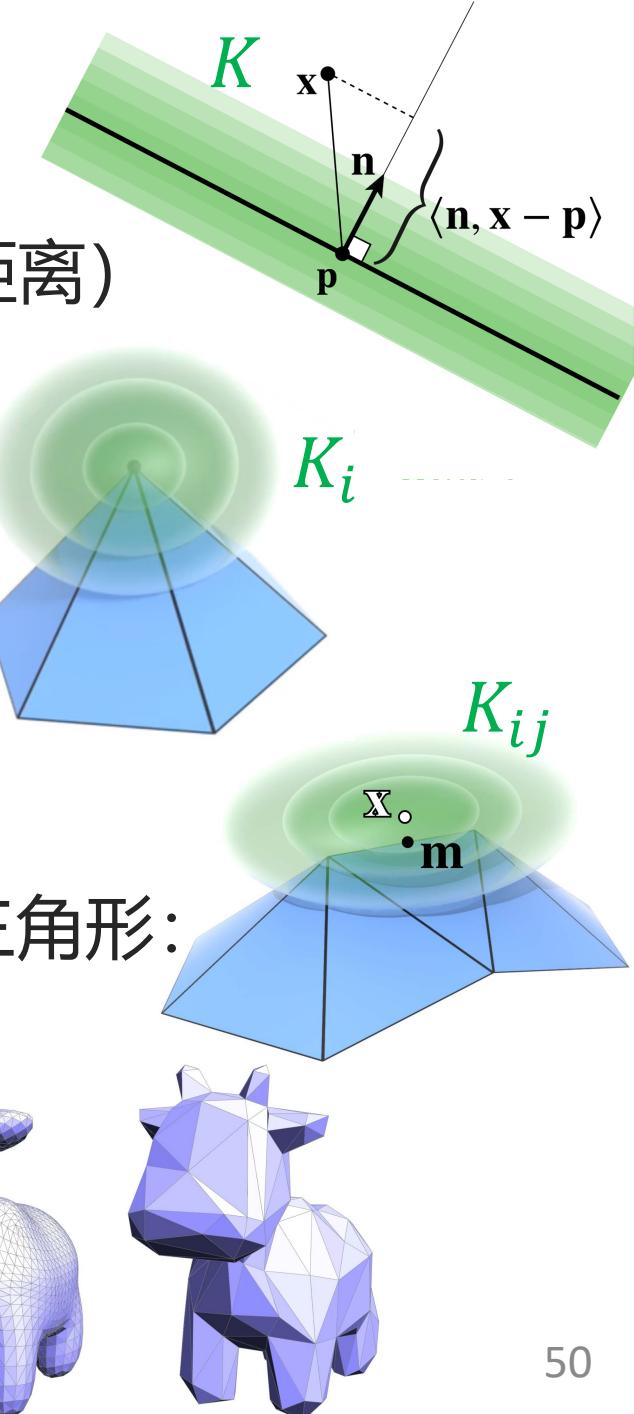
□ 在每个顶点处，将 K_i 设为相邻三角形的 K 之和

□ 对于每一条边 e_{ij} ：

- 设置 $K_{ij} = K_i + K_j$
- 找到最小化误差的点 x ，将误差设置为 $K_{ij}(x)$

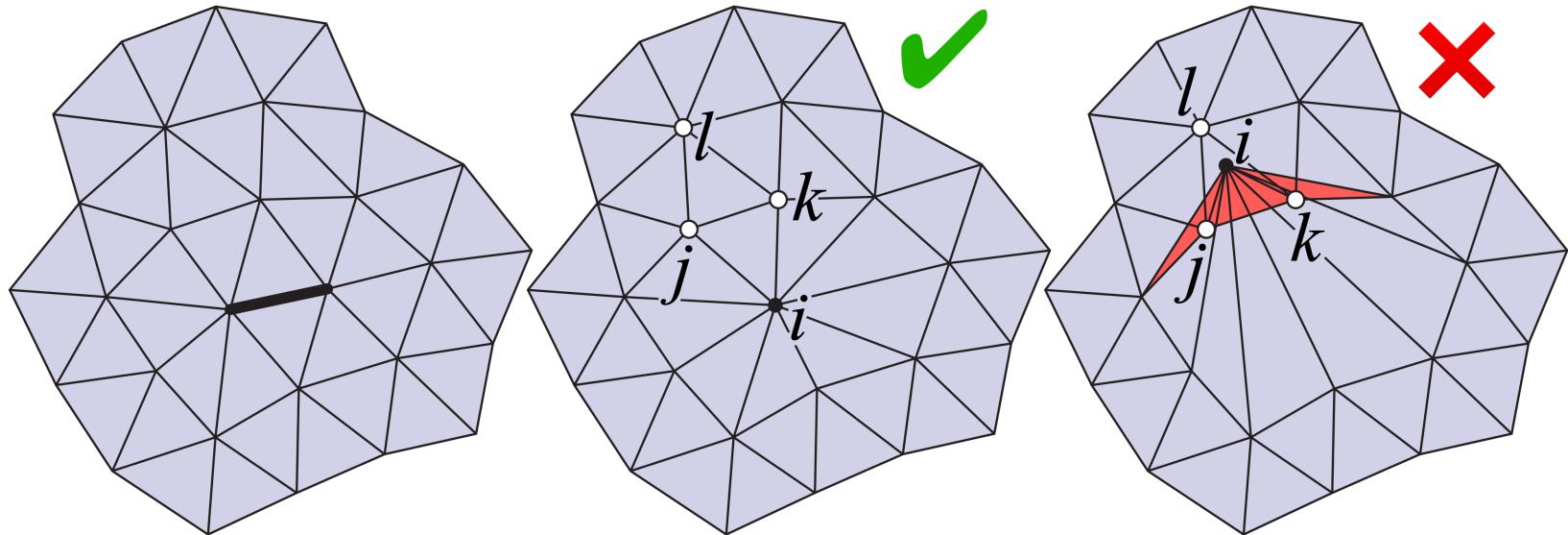
□ 重复下列步骤，直到达到目标数量的三角形：

- 将边 e_{ij} 折叠到最优点 (即具有最小代价的点)
- 在新顶点处设置二次曲面为 K_{ij}
- 更新接触新顶点的边的误差



基于二次误差的网格简化 – 三角形翻转

根据我们放置新顶点的位置，新的三角形可能会"翻转"（法线指向内部而不是外部）：



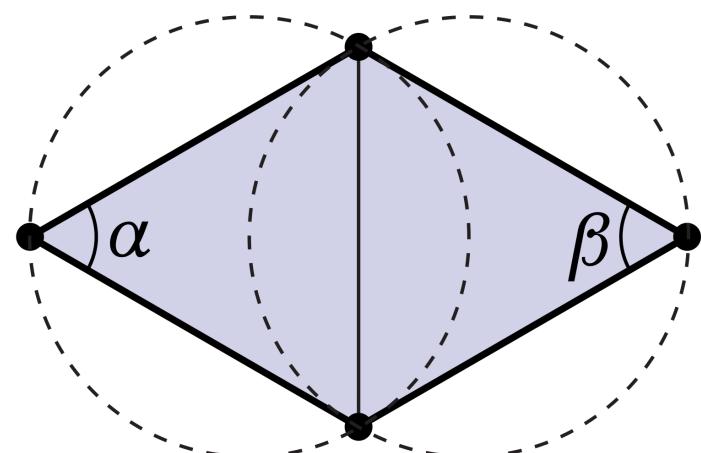
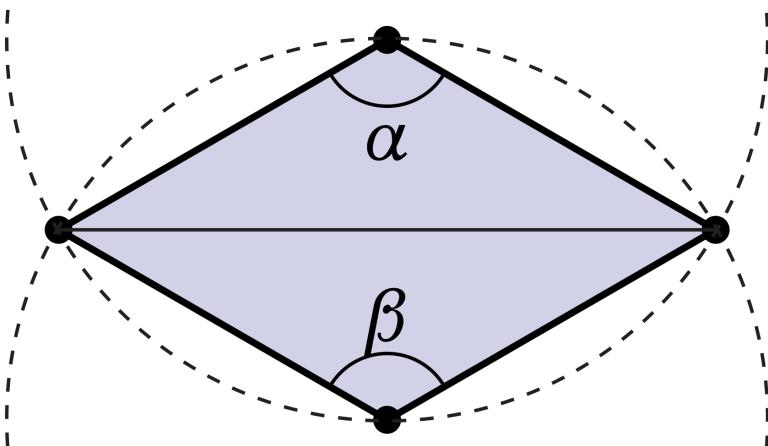
简单的解决方案：对于触及折叠顶点 i 的每个三角形 ijk
考虑法线 N_{ijk} 和 N_{kjl} (kjl 是包含边 jk 的另一个三角形)
如果 $\langle N_{ijk}, N_{kjl} \rangle$ 为负，则不要折叠这条边！

如果我们满意三角形的数量，
但想提高网格质量，该怎么办？

我们如何使一个网格"更 Delaunay"?

□ 我们已经有一个好工具：翻转边缘！

□ 如果 $\alpha + \beta > \pi$, 则反转



□ 事实：在2D中，翻转边缘最终会产生 Delaunay 网格

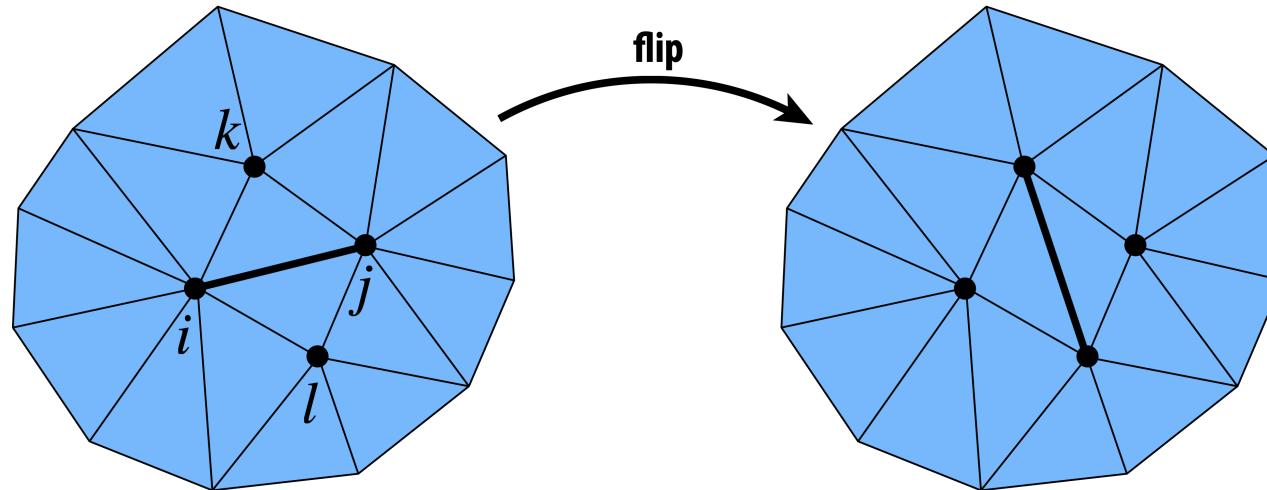
□ 理论：最坏情况 $O(n^2)$ ；对于 3D 表面并不总是有效

□ 实践：简单，有效的方法来提高网格质量

或者，我们如何提高节点的度？

口同样的工具：翻转边缘！

口如果总的偏离度（三角形最优为 6）变小，就翻转它！



$$\text{总偏离度: } |d_i - 6| + |d_j - 6| + |d_k - 6| + |d_l - 6|$$

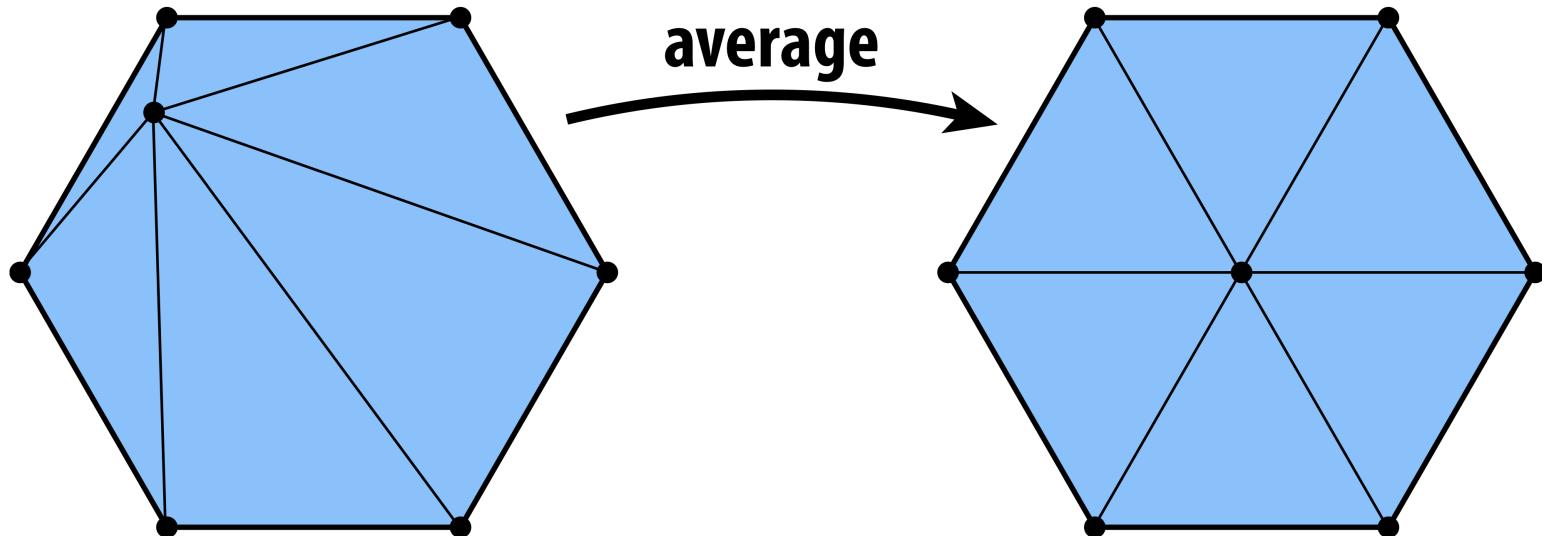
口事实：随着元素数量的增加，平均度数趋近于 6

口迭代翻转边缘能把密集区域的度扩散到其他区域

口没有（已知的）理论保证；在实践中效果良好

我们如何使三角形"更圆润"?

- Delaunay 不能保证三角形是"圆形"（角度接近60°）
- 通过将顶点居中，我们通常可以改善形状：



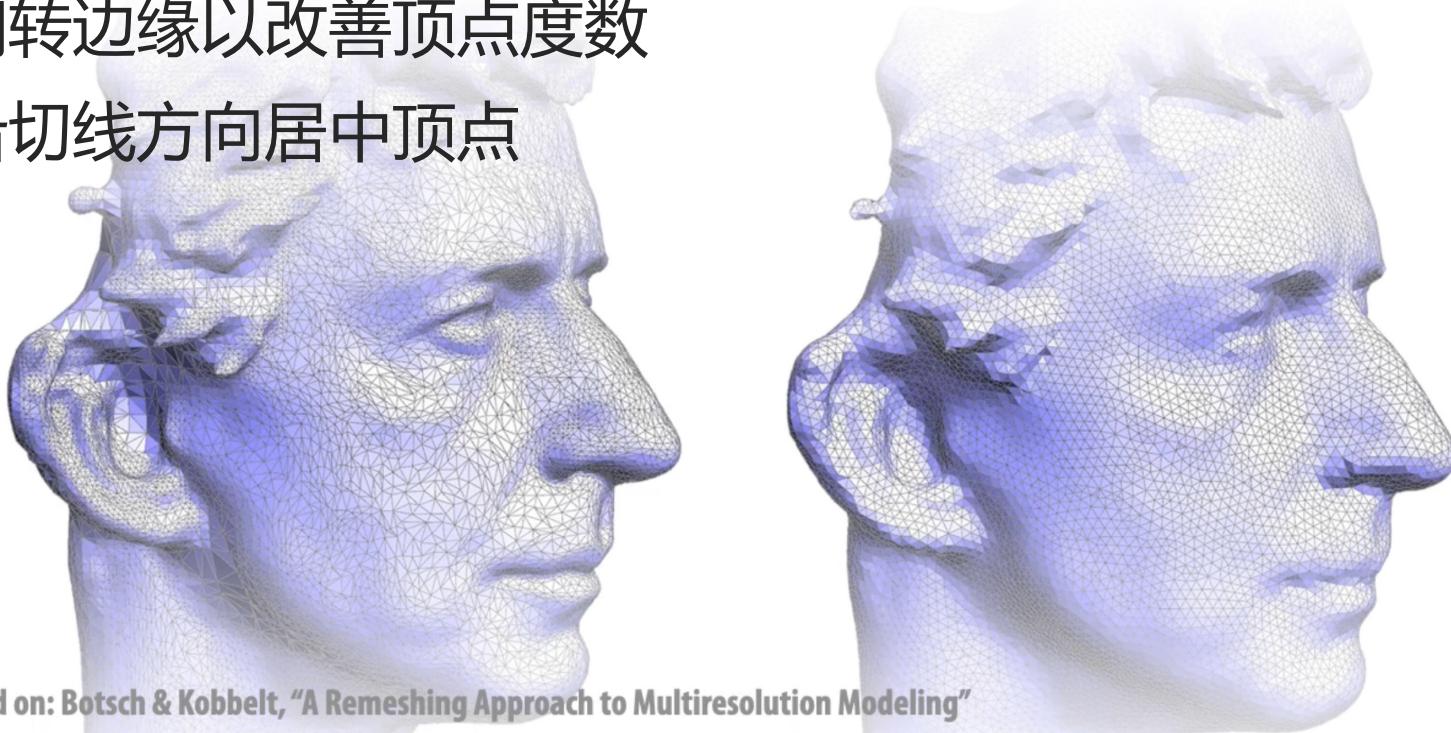
- 这是一种称为" Laplacian 平滑"的简单技术
- 在表面上：只在切线方向移动以居中顶点
- 如何做到？从“移动向量”中减去法向分量

各向同性重网格化算法 Isotropic Remeshing Algorithm

□ 试图使三角形的形状和大小统一

□ 重复四个步骤：

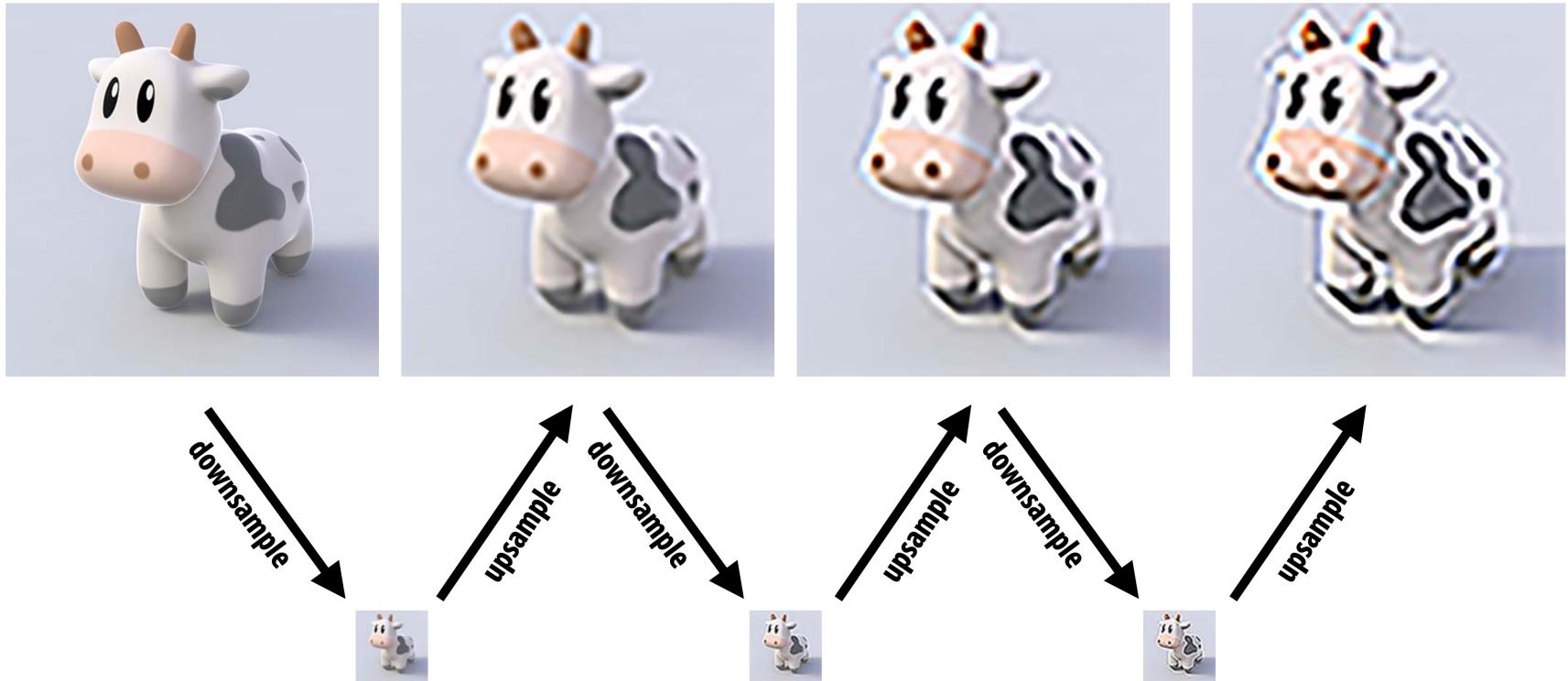
- 分割任何超过 $4/3$ 平均边长的边
- 折叠任何小于 $4/5$ 平均边长的边
- 翻转边缘以改善顶点度数
- 沿切线方向居中顶点



当对一个信号进行重采样时，
可能会出现什么问题？

重采样的风险

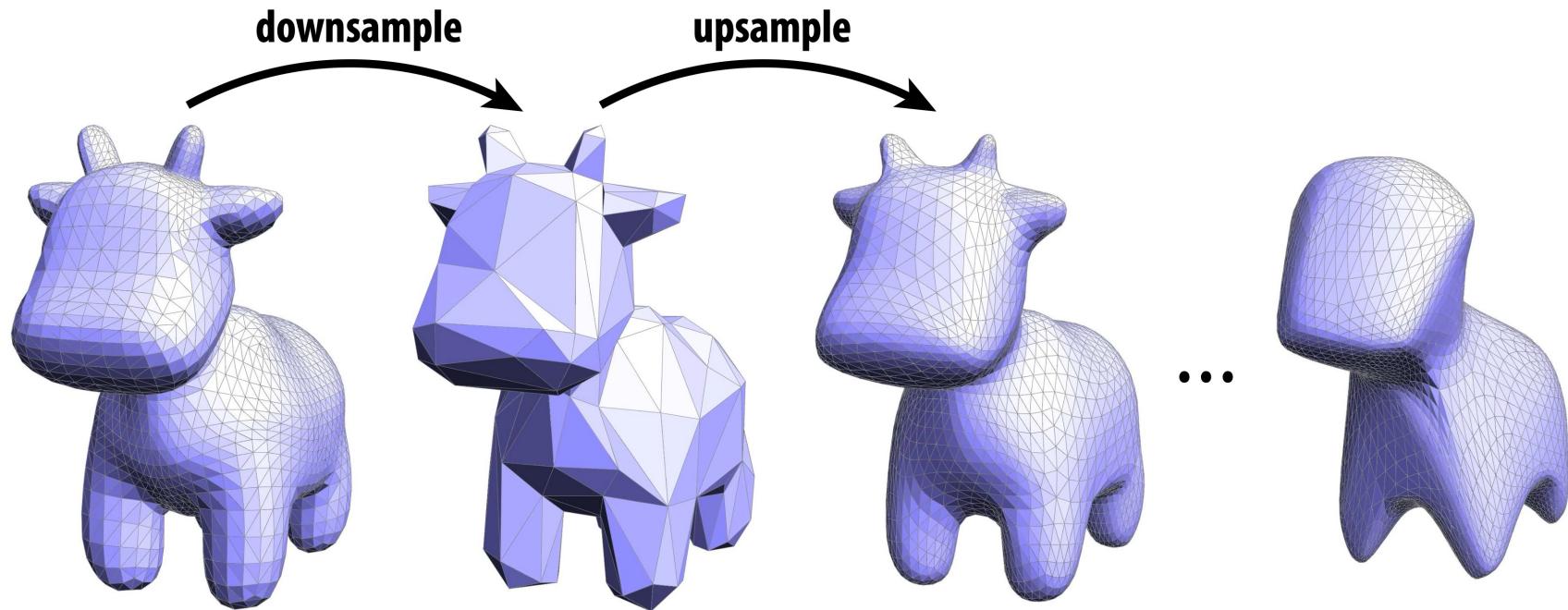
□Q：如果我们反复重采样一张图片会发生什么？



□A：信号质量下降！

重采样的风险

□Q：如果我们反复重采样一个网格会发生什么？

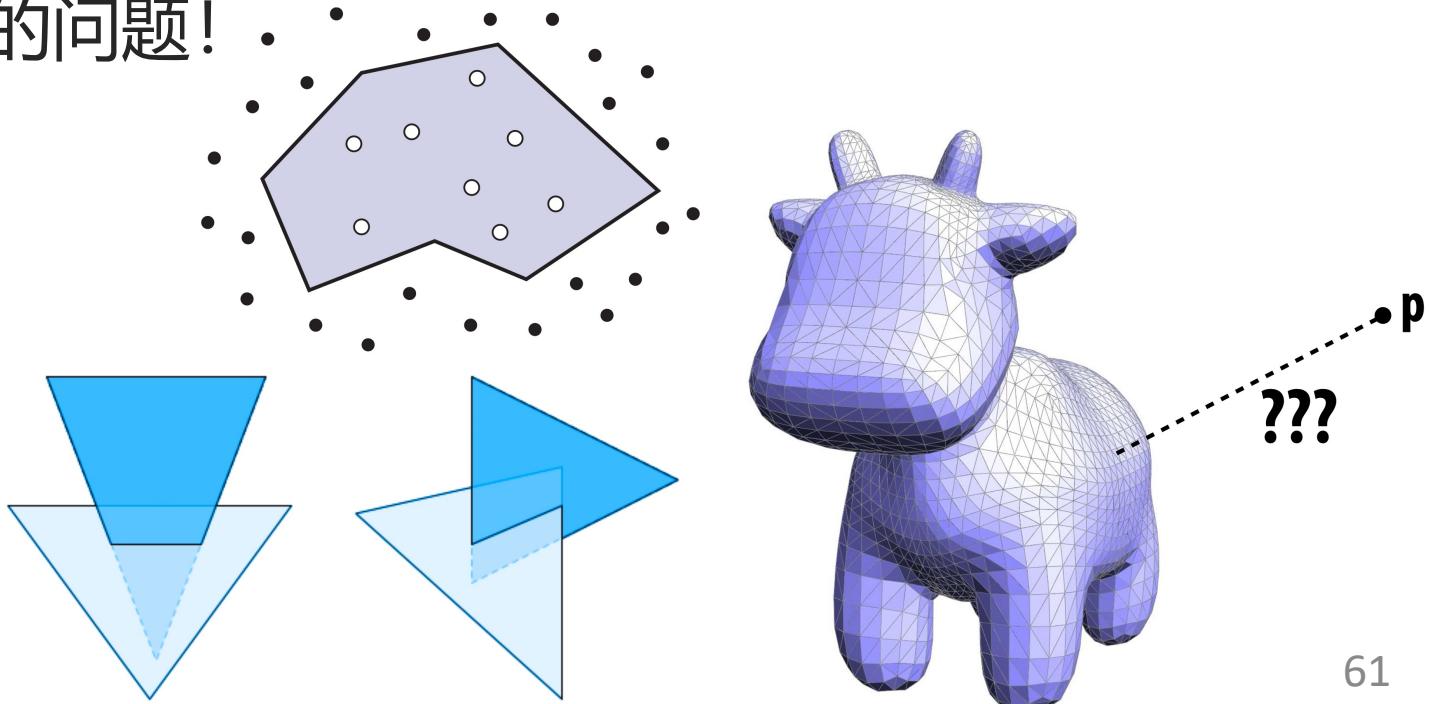


□A：信号质量同样会下降！

等等：我们有原始的信号（网格）
为什么不直接将每个新的采样点投
影到原始网格的最近点上呢？

下节课：几何查询

- 口给定一个空间中的点，我们如何找到一个表面上的最近点？
- 口我们在表面内部还是外部？如何找到两个三角形的交点？
- 口隐式/显式表示是否让这些任务变得更容易？
- 口朴素算法的成本是多少，如何加速对大型网格的查询？
- 口会有非常多的问题！





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