Computer Graphics -Sampling

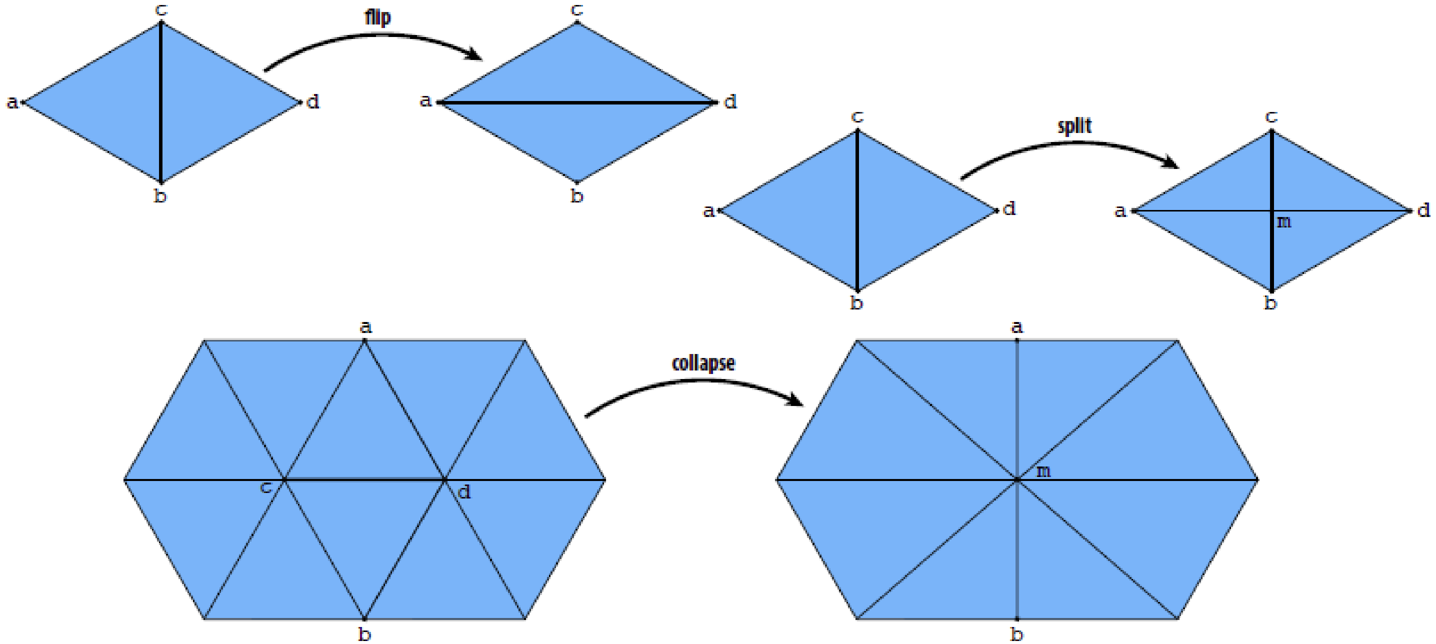
Junjie Cao @ DLUT

Spring 2016

http://jjcao.github.io/ComputerGraphics/

Processing geometry with halfedges

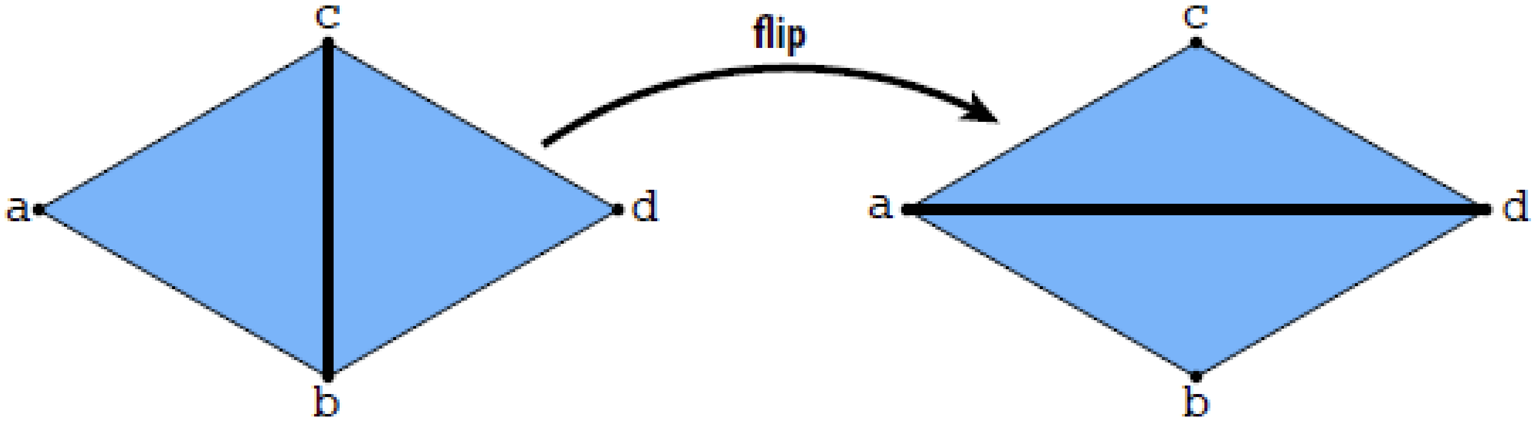
- Remember key feature of linked list: insert/delete elements
- Same story with halfedge mesh (“linked list on steroids”)
- Several atomic operations for triangle meshes:



- How? Allocate/delete elements; reassigning pointers.
- (Should be careful to preserve manifoldness!)

Edge Flip

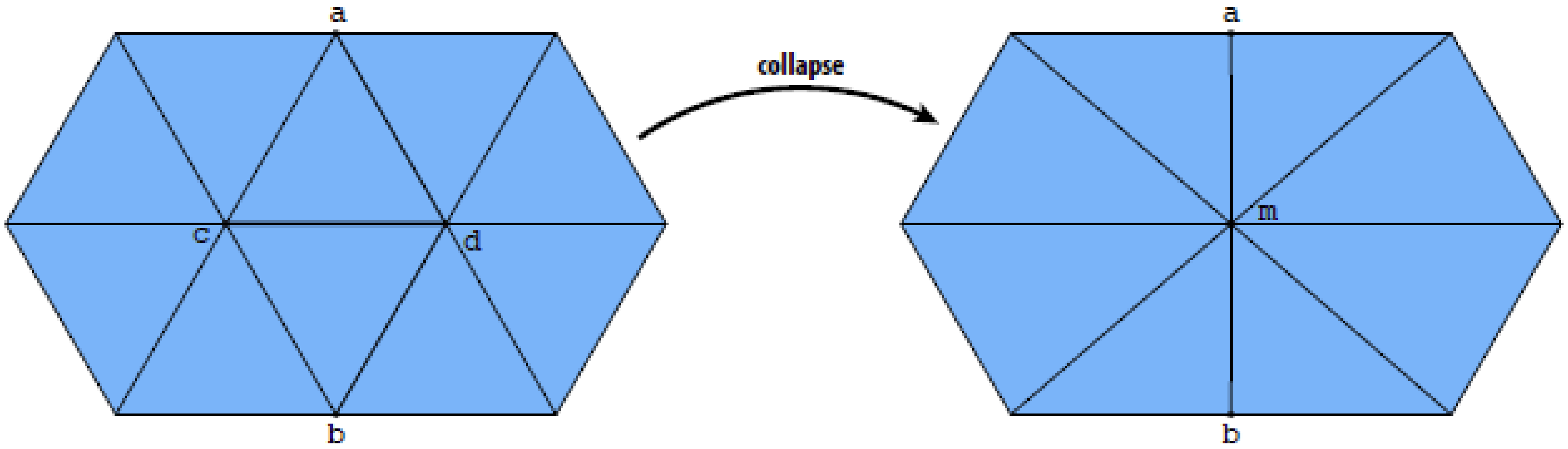
- Triangles (a,b,c), (b,d,c) become (a,d,c), (a,b,d):



- Long list of pointer reassignments (edge->halfedge = ...)
- However, no elements created/destroyed.
- Q: What happens if we flip twice?
- (Challenge: can you implement edge flip such that pointers
- are unchanged after two flips?)

Edge Collapse

- Replace edge (c,d) with a single vertex m:

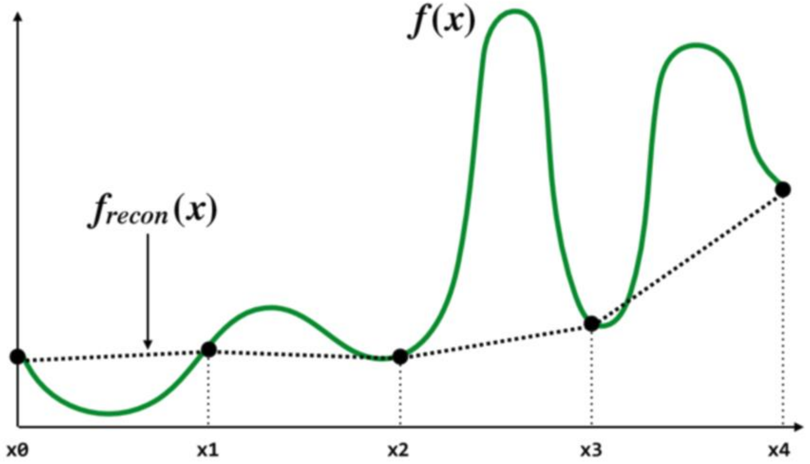


- Now have to *delete* elements.
- Still lots of pointer assignments!
- Q: How would we implement this with a polygon soup?
- Any other good way to do it? (E.g., different data structure?)

Ok, but what can we actually *do* with these operations?

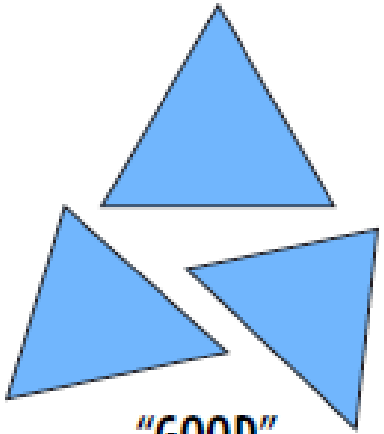
Remeshing as resampling

- Remember our discussion of *aliasing*
- Bad sampling makes signal appear different than it really is
- E.g., undersampled curve looks flat
- Geometry is no different!
  - undersampling destroys features
  - oversampling destroys performance



What makes a “good” triangle mesh?

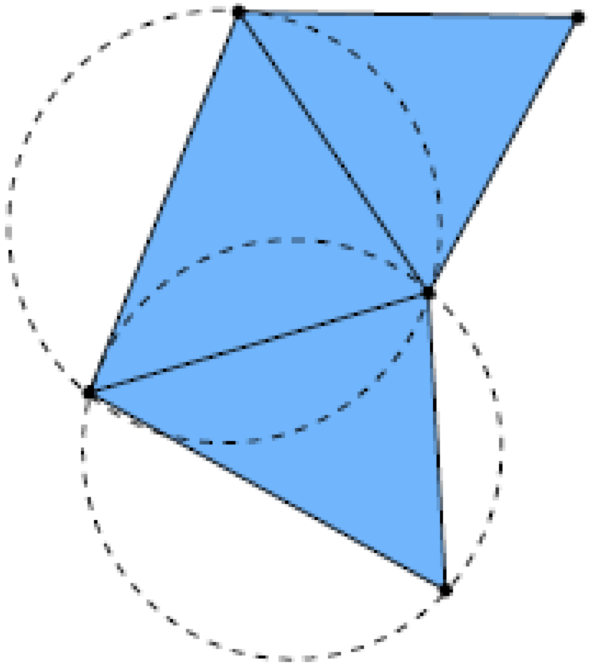
- One rule of thumb: *triangle shape*



“GOOD”

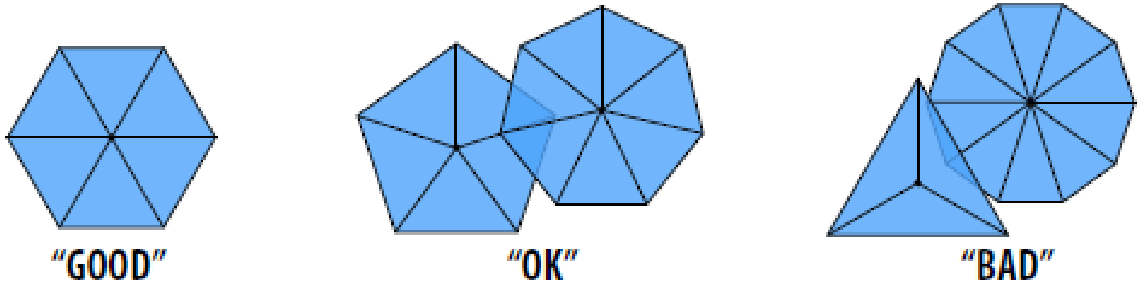
“BAD”

- More specific condition: *Delaunay*
- “Circumcircle interiors contain no vertices.”
- Not *always* a good condition, but often*
- especially important for simulation
- for approximation, long triangles may be better

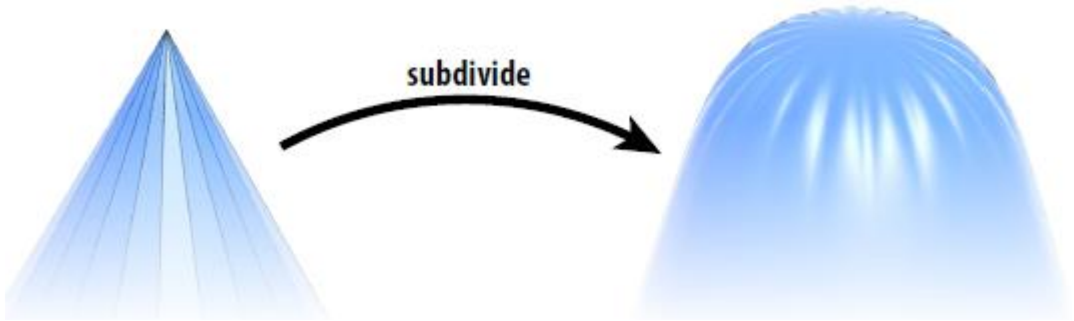


What else constitutes a good mesh?

- Another rule of thumb: *regular vertex degree*
- Ideal for triangle meshes: make every vertex valence 6:



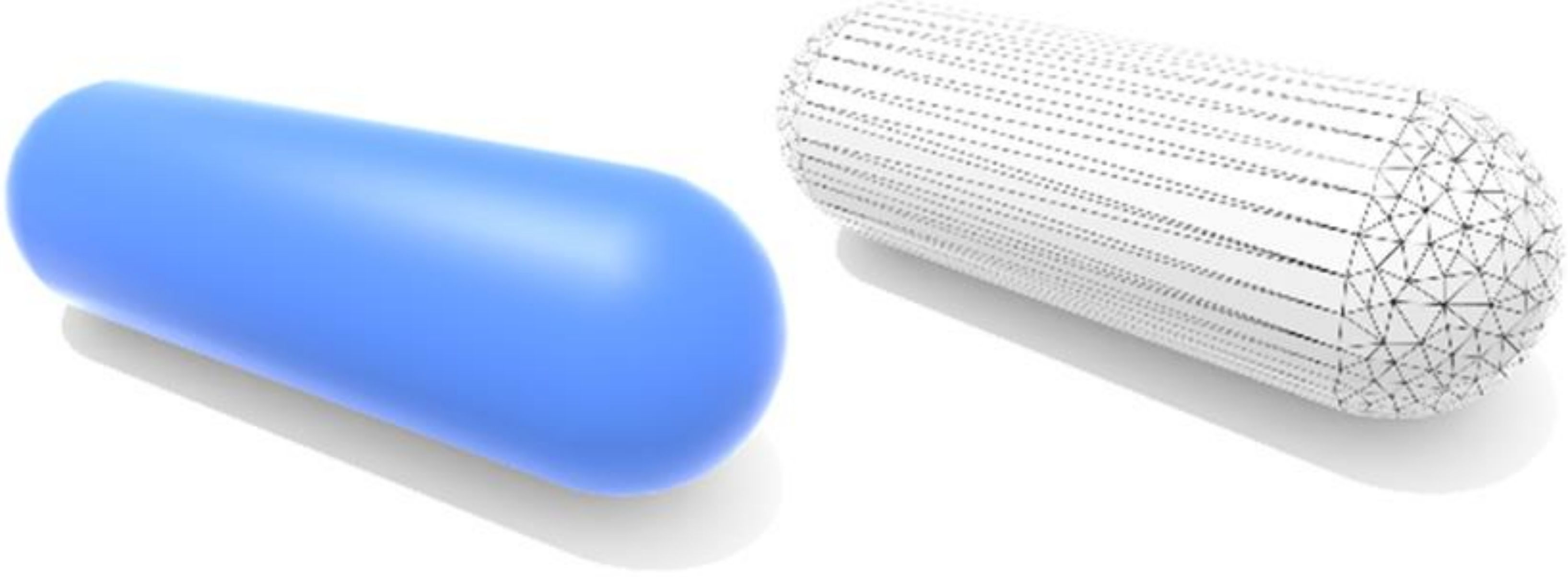
- Why? Better triangle shape, important for (e.g.) subdivision:



- FACT: Can’t have perfect valence everywhere! (except on torus)

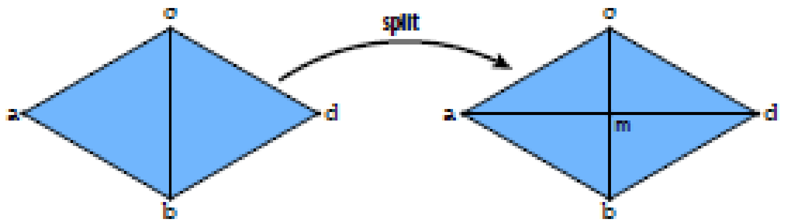
What *else* makes a "good" mesh?

- Keep only elements that contribute *information* about shape
- Add additional information where, e.g., *curvature* is large
- Balance with element quality
  - Delaunay, "round" triangles, regular degree, etc.

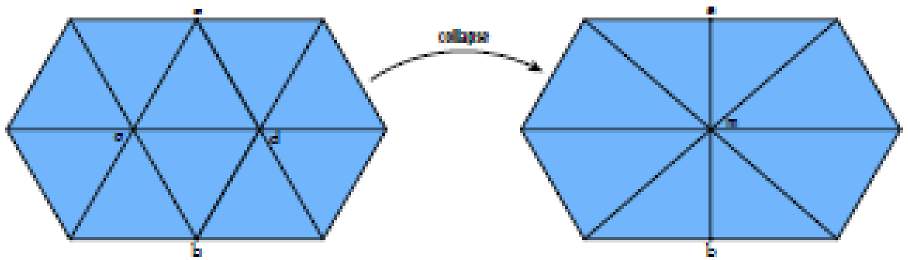


How do we resample? Already know how!

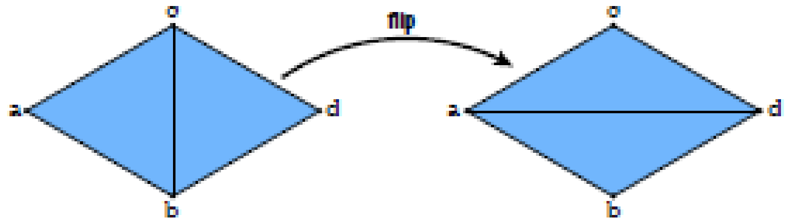
- **Edge split is (local) upsampling:**



- **Edge collapse is (local) downsampling:**



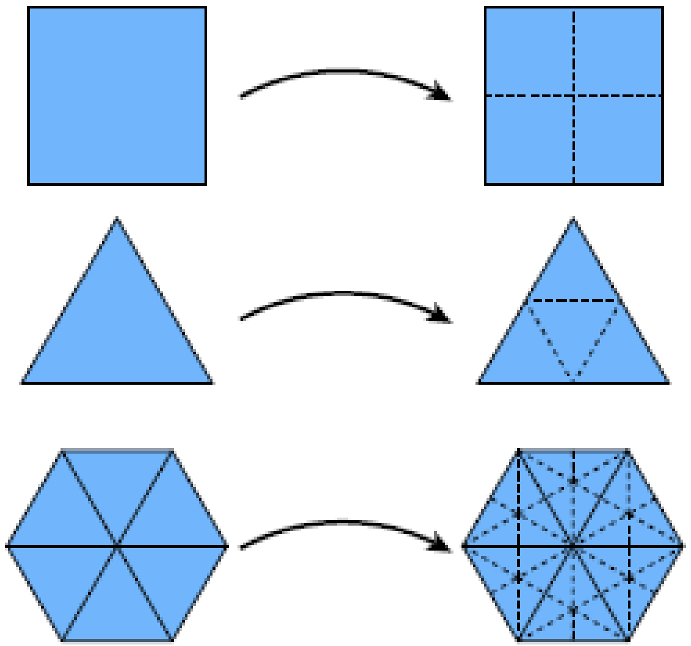
- **Edge flip is (local) resampling:**



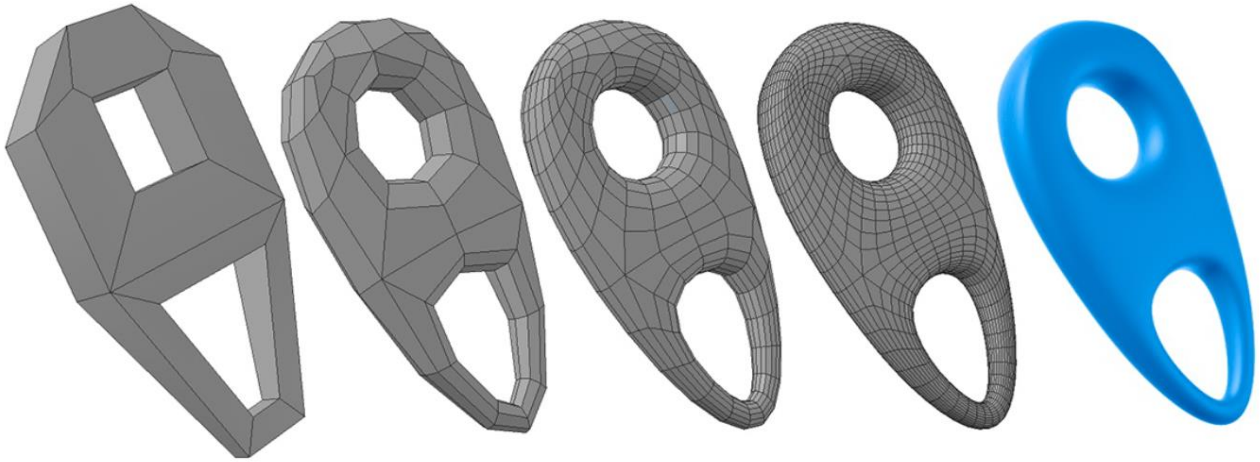
- **Still need to intelligently decide *which* edges to modify!**

Which edges should we split to upsample the whole mesh?

Subdivision as Upsampling

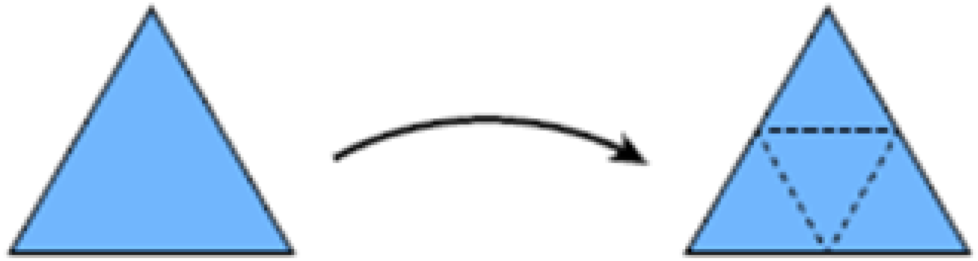


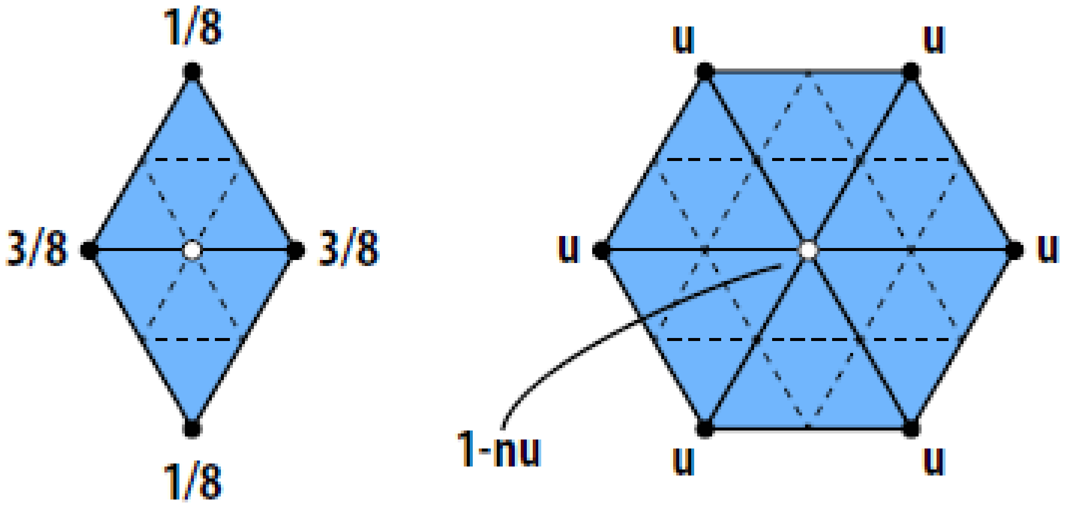
- **Repeatedly split each element into smaller pieces**
- **Replace vertex positions with weighted average of neighbors**
- **Main considerations:**
  - **interpolating vs. approximating**
  - **limit surface continuity (*C1*, *C2*, ...)**
  - **behavior at irregular vertices**
- **Many options:**
  - **Quad: Catmull-Clark**
  - **Triangle: Loop, Butterfly, Sqrt(3)**



Loop Subdivision

- **Fairly common subdivision rule for triangles**
- **Curvature is continuous away from irregular vertices ("*C2*")**
- ***Approximating*, not interpolating**
- **Algorithm:**
  - **Split each triangle into four**
  - **Assign new vertex positions according to weights:**

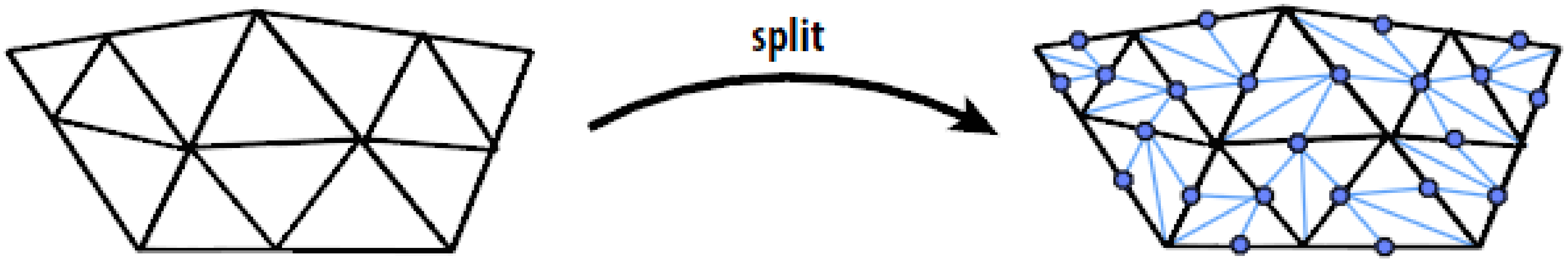




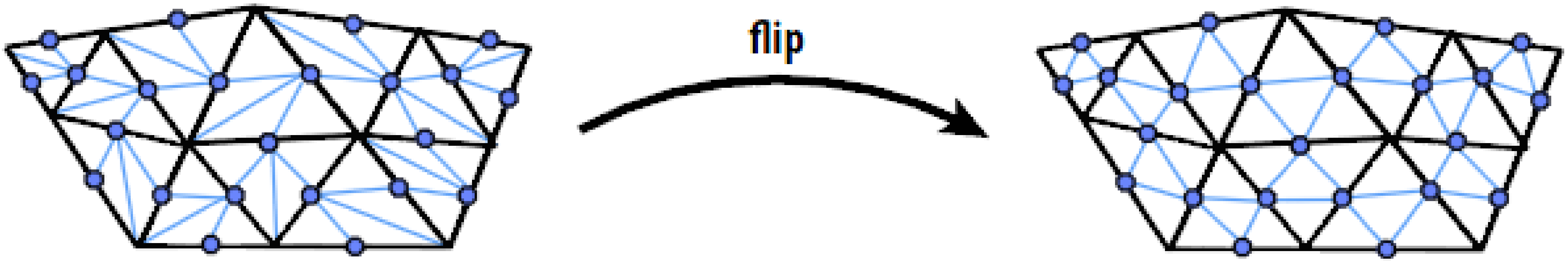

```
n: vertex degree
u: 3/16 if n=3, 3/(8n) otherwise
```

Loop Subdivision via Edge Operations

- **First, split edges of original mesh in *any* order:**



- **Next, flip new edges that touch a new & old vertex:**



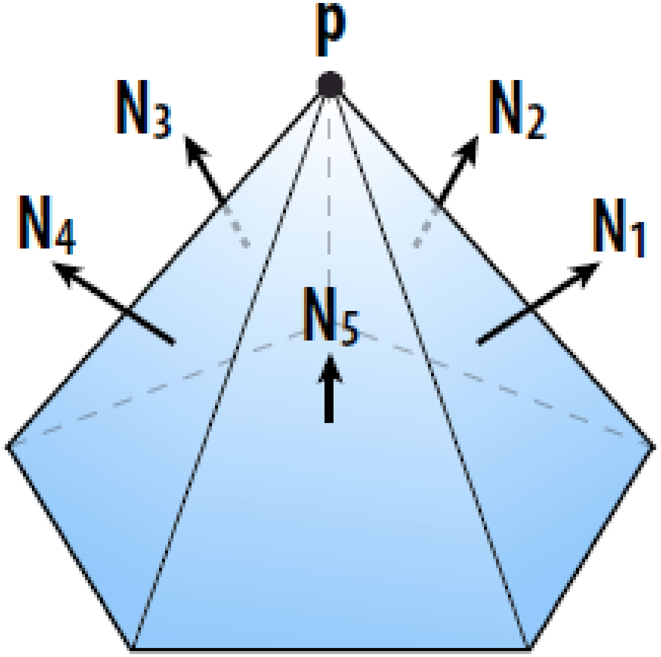
What if we want fewer triangles?

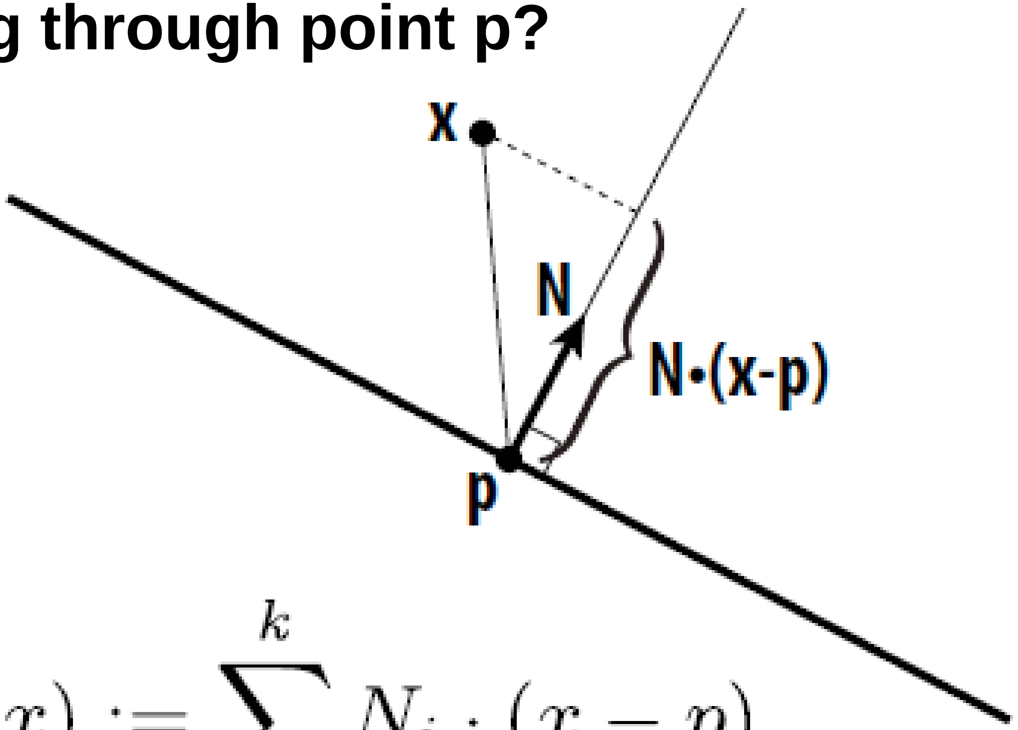
Simplification via Quadric Error Metric

- One popular scheme: iteratively collapse edges
- Which edges? Assign score with *quadric error metric**
  - approximate distance to surface as sum of distance to aggregated triangles
  - iteratively collapse edge with smallest score
  - greedy algorithm... great results!

Quadric Error Metric

- **Approximate distance to a collection of triangles**
- **Distance is sum of point-to-plane distances**
  - **Q: Distance to plane w/ normal N passing through point p?**
  - **A: d(x) = N•x - N•p = N•(x-p)**
- **Sum of distances:**





$$d(x) := \sum_{i=1}^{k} N_i \cdot (x - p)$$

Quadric Error - Homogeneous Coordinates

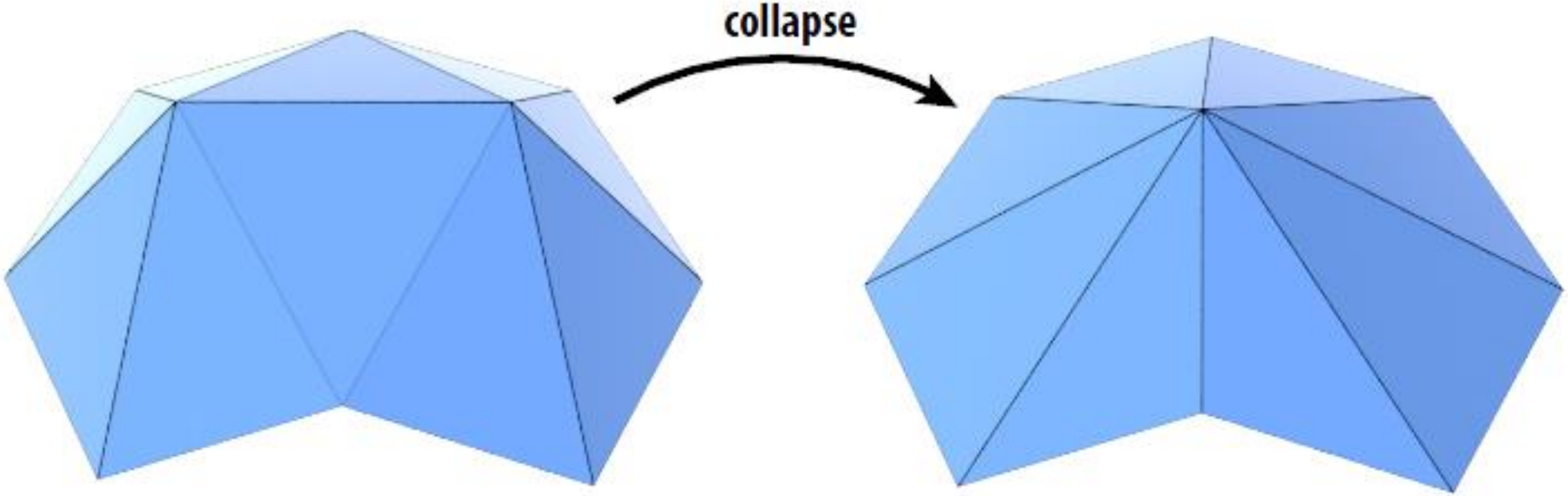
- **Suppose in coordinates we have**
  - **a query point (x,y,z)**
  - **a normal (a,b,c)**
  - **an offset d := -(x,y,z) • (a,b,c)**
- **Then in homogeneous coordinates, let**
  - **u := (x,y,z,1)**
  - **v := (a,b,c,d)**

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

- ***Signed* distance to plane is then just u•v = ax+by+cz+d**
- ***Squared* distance is (uTv)2 = uT(vvT)u =: uTQu**
- **Key idea: *matrix Q encodes distance to plane***
- **Q is symmetric, contains 10 unique coefficients (small storage)**

Quadric Error of Edge Collapse

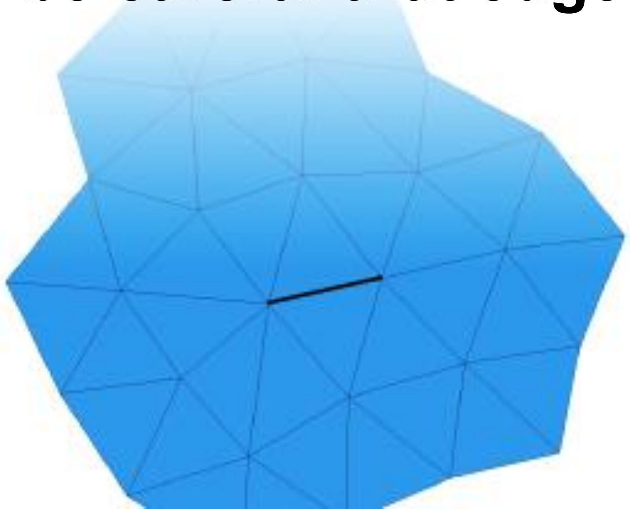
- How much does it cost to collapse an edge?
- Idea: compute edge midpoint, measure quadric error

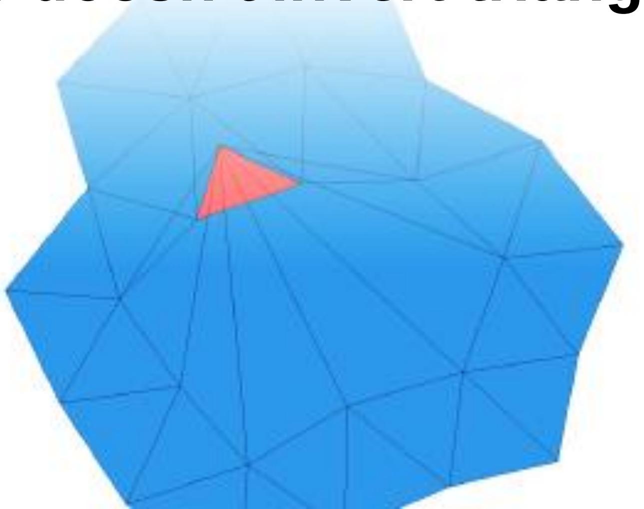


- Better idea: use point that minimizes quadric error as new point!
- (More details in assignment; see also Garland & Heckbert 1997.)

Quadric Error Simplification

- Compute Q for each triangle
- Set Q at each vertex to sum of Qs from incident triangles
- Until we reach target # of triangles:
  - collapse edge (i,j) with smallest cost to get new vertex k
  - add Qi and Qj to get new quadric Qk
  - update cost of any edge touching new vertex k
- Store edges in *priority queue* to keep track of minimum cost
- Should be careful that edge flip doesn't *invert* triangles:

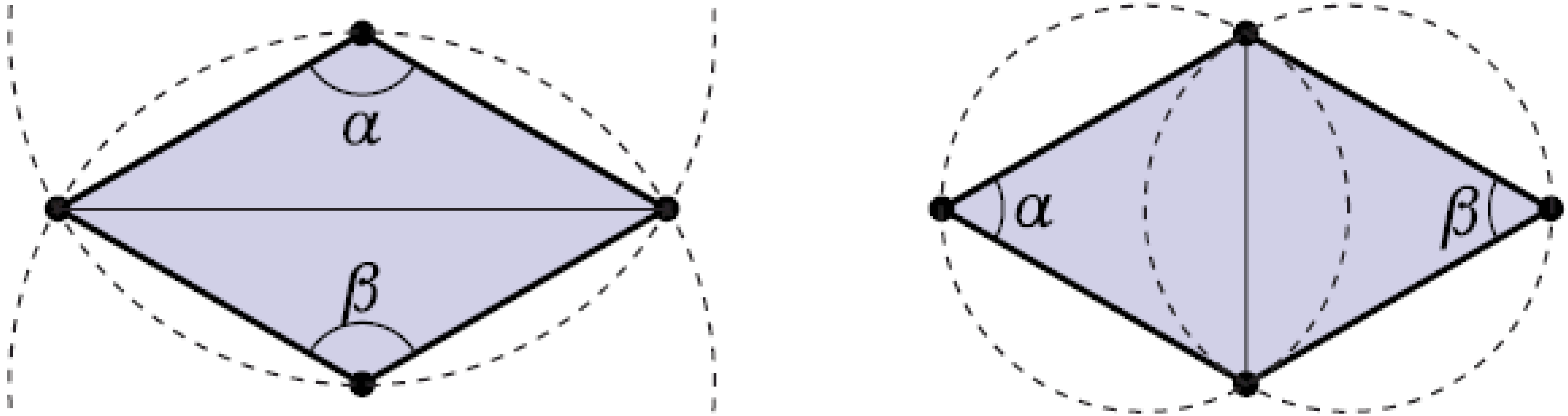




What if we’re happy with the number of triangles, but want to improve quality?

How do we make a mesh "more Delaunay"?

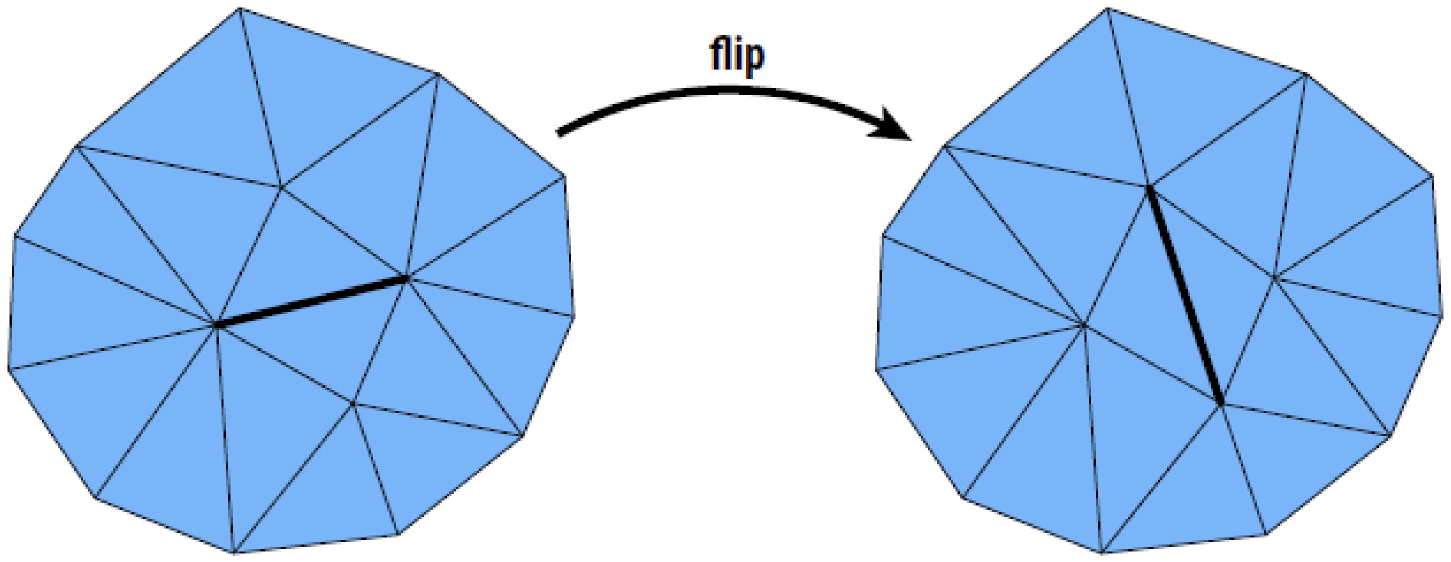
- Already have a good tool: edge flips!
- If α+β > π, flip it!



- FACT: in 2D, flipping edges eventually yields Delaunay mesh
- Theory: worst case O(n2); no longer true for surfaces in 3D.
- Practice: simple, effective way to improve mesh quality

Alternatively: how do we improve degree?

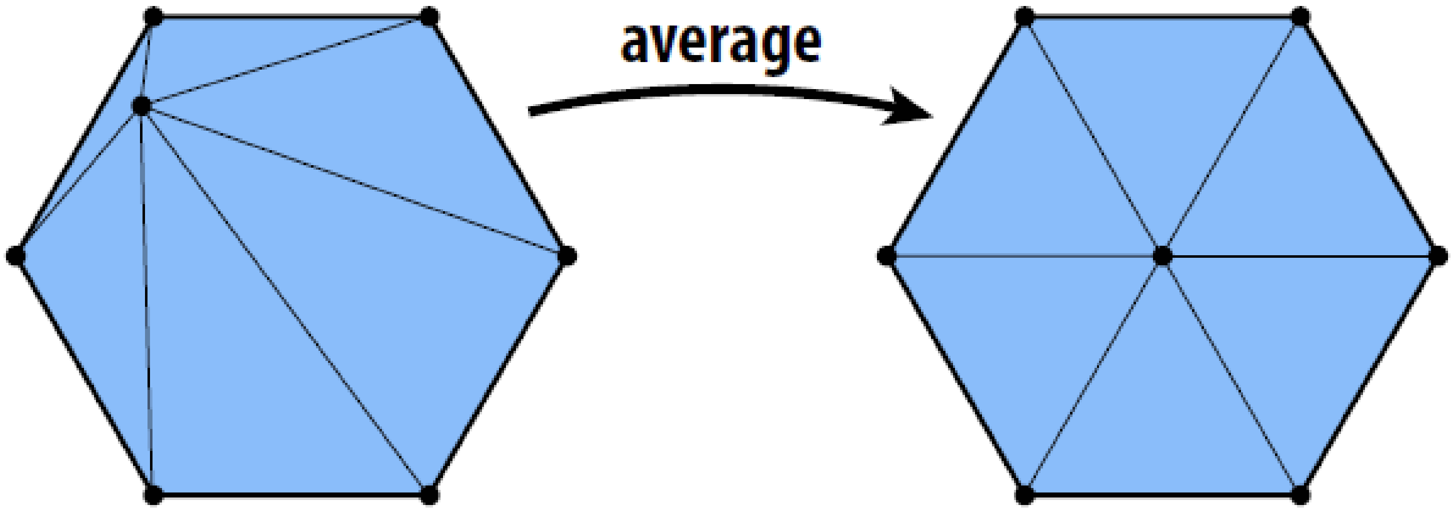
- **Same tool: edge flips!**
- **If total deviation from degree-6 gets smaller, flip it!**



- **FACT: average valence of any triangle mesh is 6**
- **Iterative edge flipping acts like “discrete diffusion” of degree**
- **Again, no (known) guarantees; works well in practice**

How do we make a triangles “more round”?

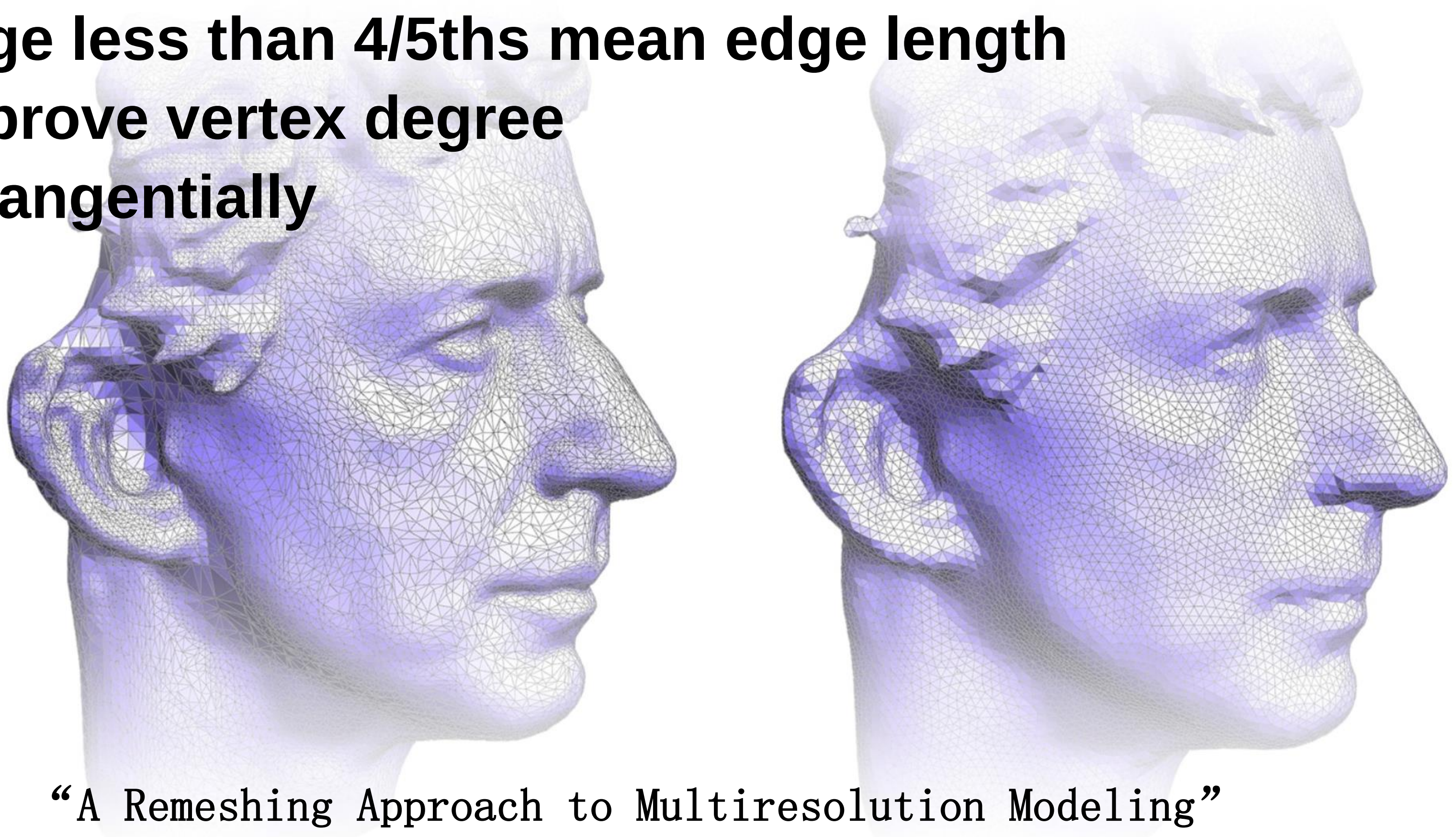
- Delaunay doesn’t mean triangles are “round” (angles near 60°)
- Can often improve shape by centering vertices:



- Simple version of technique called “Laplacian smoothing”.*
- On surface: move only in *tangent* direction
- How? Remove normal component from update vector.

Isotropic Remeshing Algorithm*

- Try to make triangles uniform shape & size
- Repeat four steps:
  - Split any edge over 4/3rds mean edge legth
  - Collapse any edge less than 4/5ths mean edge length
  - Flip edges to improve vertex degree
  - Center vertices tangentially



*Based on Botsch & Kobbelt, “A Remeshing Approach to Multiresolution Modeling”

Thank you