Assignment 2: Advanced Features

COS 426: Computer Graphics (Spring 2020)

June Ho Park

Agenda

- Logistical Updates
- General tips on tackling A2
- Going over more advanced features of A2
  - Truncate, Extrude, Bevel
  - Triangle/Quad Topology, Loop/Catmull-Clark subdivision
  - Smoothing, Curvature

Logistics

- A2 is now due on Tuesday, 11:55 PM
- Please fill out A1 Feedback Form!
  - https://forms.gle/SVVV12E422K7nH619
- Midterm is in-class Thursday, 03/12
  - Practice exams will be released this weekend
  - One double-sided A4 cheat sheet, typed or written
  - Next week's precept will be a review session

Final Project Presentation Logistics

- Happening sometime after Dean's Date TBD
- Two sessions in one day
  - One morning, one afternoon
  - All students must attend both sessions, with the exception of exam conflicts
  - Food will be provided

Approach

- Start local
  - Modifications to a primitive shouldn't affect other primitives
- Work with one primitive first

Decouple Topology and Geometry

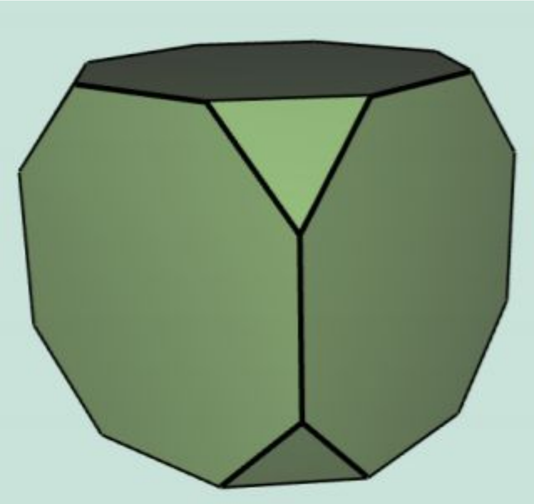
- Topology
  - Relations between structures defining the mesh
    - eg. <u>What</u> vertices do I need to add?
    - eg. Between what vertices should I add an edge?
- Geometry
  - Spatial relationships, shape, form
    - eg. <u>Where</u> on the edge should I insert the vertex?
- Figure out topology first, then geometry

Other Tips

- Caution with Data
  - Do I need to store information about data before modifying them?
- Draw your operations out
- Count primitives after modifications
  - Console Log is your friend!

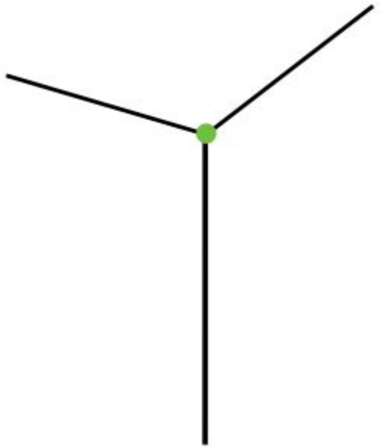
Truncate

- Cut the corners off of a shape
- For every vertex with N edges...
  - Add N-1 vertices
  - Add 1 face

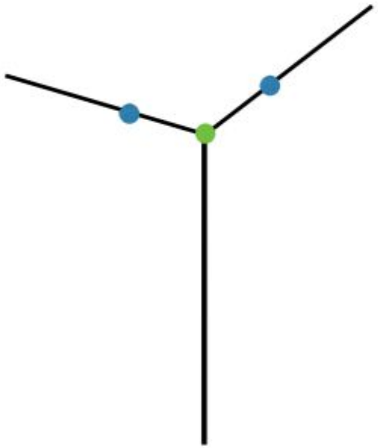


Truncate - Topology

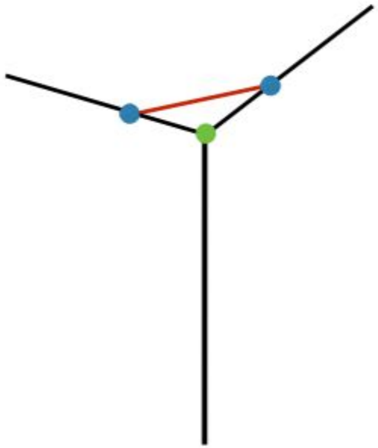
- Consider a vertex with 3 edges
- So we need to add 2 vertices, 1 face



Initial



SplitEdge x 2



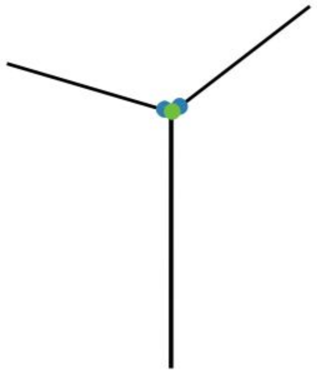
Split Face

Note that the blue vertices should be on top of original vertex in reality.

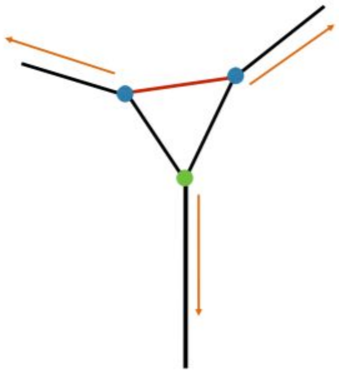
They are moved apart for easier visualization.

Truncate - Geometry

- Now we move vertices along the edges
  - Calculate all offset vectors before applying changes



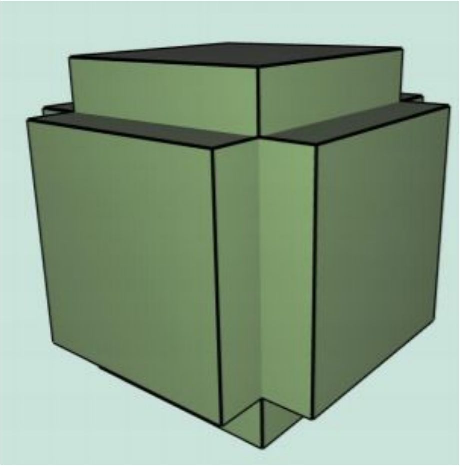
After Making Face



Apply Offsets

Extrude

- Each face is moved along its normal
- For each N-gon face:
    - Add N vertices
    - Add N faces



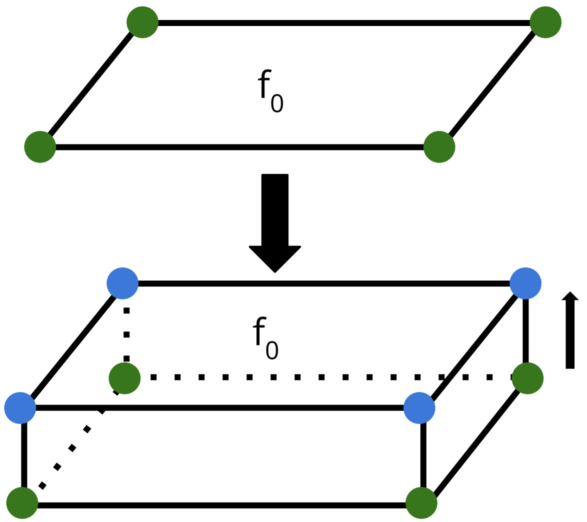
Extrude - Topology

- Note again that the visualizations don't represent accurate spatial relations
- New blue vertices should be directly on top of the old ones at first!!!

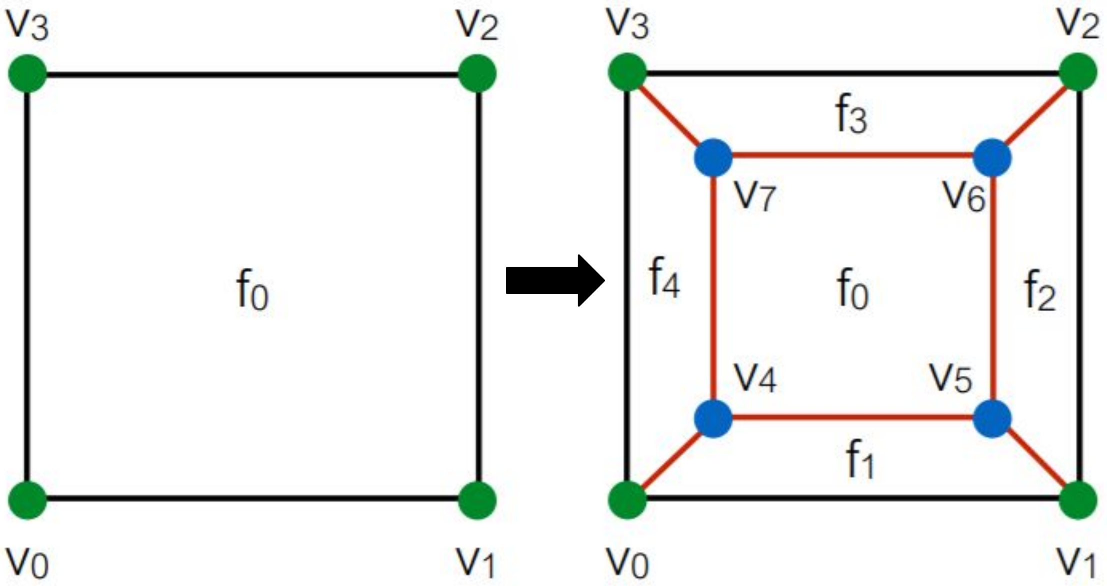
Extrude - Topology

- Let's think about the end result for 1 face

3D View:

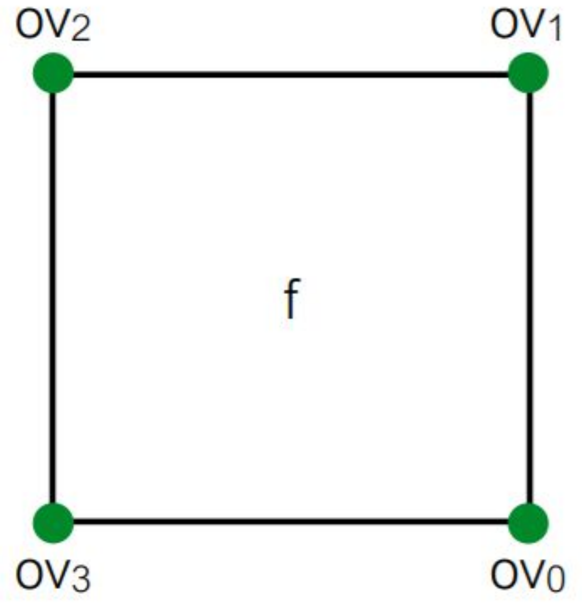


Topological View:



Extrude - Topology

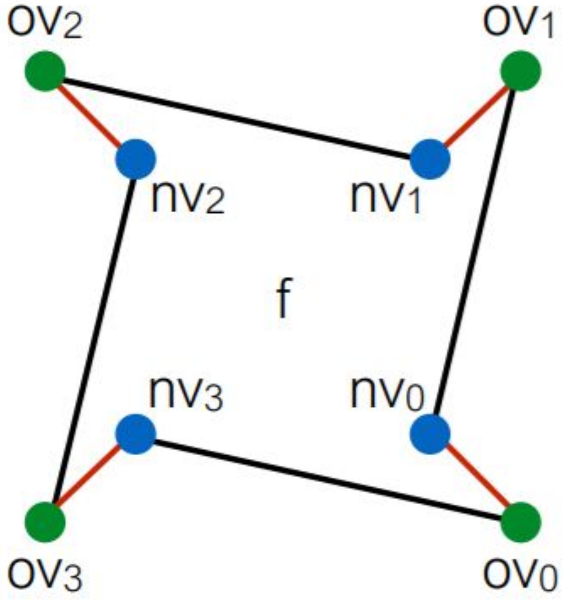
- Denote **o**v for **o**ld vert and **n**v for **n**ew vert



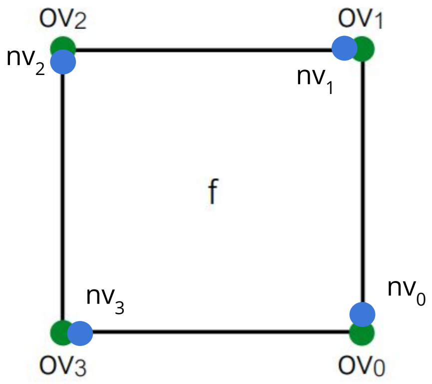
Extrude - Topology

- First, insert 4 new vertices
  - SplitEdgeMakeVert x 4
  - Again, there's no actual movement happening

Topological View:



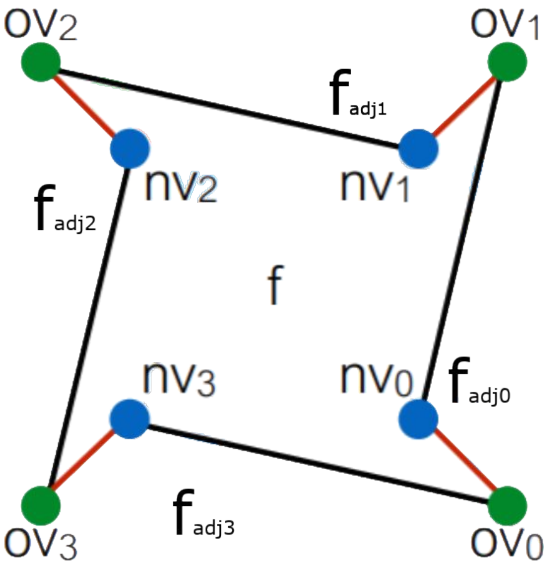
Reality:



Extrude - Topology

- Then, split 4 **adjacent** faces
  - SplitFaceMakeEdge x 4
  - Between which 2 vertices should we split the face each time?
  - Which vertex would we like on which face at the end?

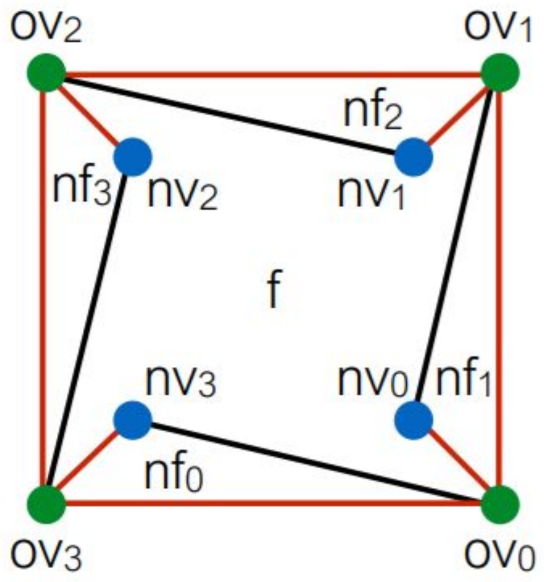
Topological View:



Extrude - Topology

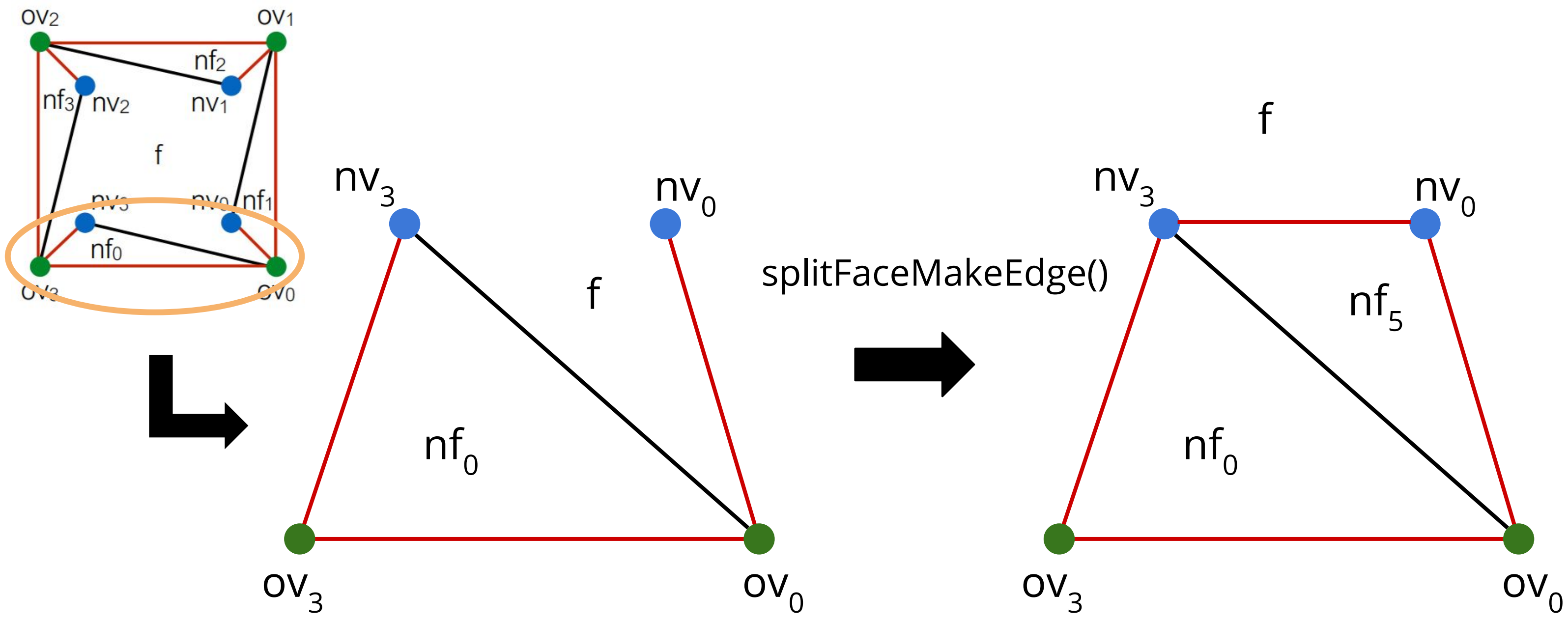
- Then, split 4 **adjacent** faces
  - SplitFaceMakeEdge x 4
  - Between which 2 vertices should we split the face each time?
  - Which vertex would we like on which face at the end?

Topological View:



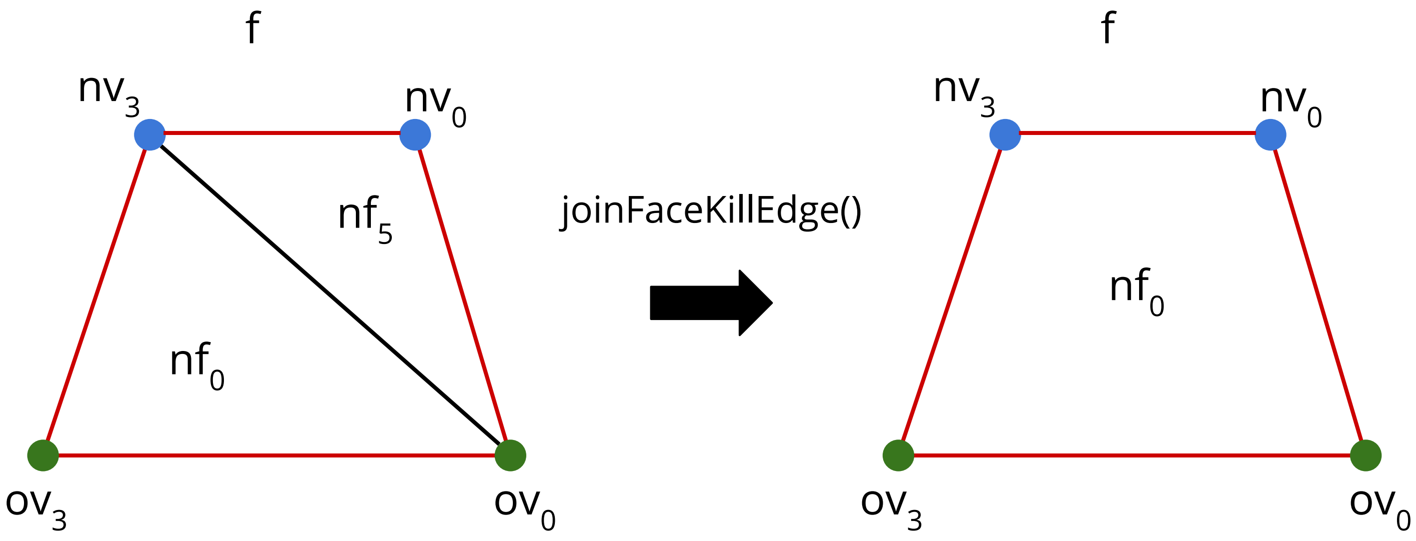
Extrude - Topology

- We want to connect the new vertices



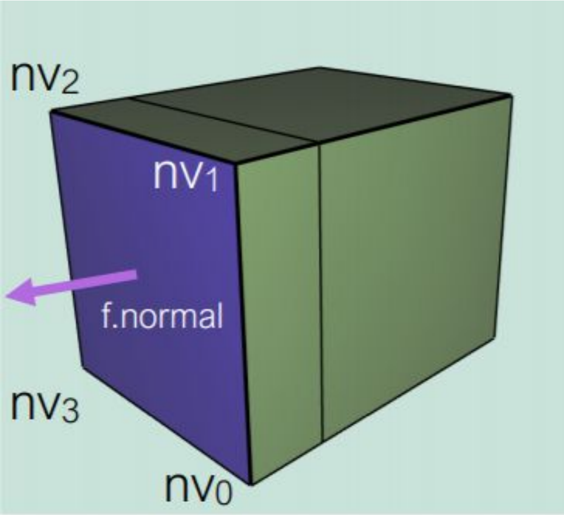
Extrude - Topology

- Now join the two new faces



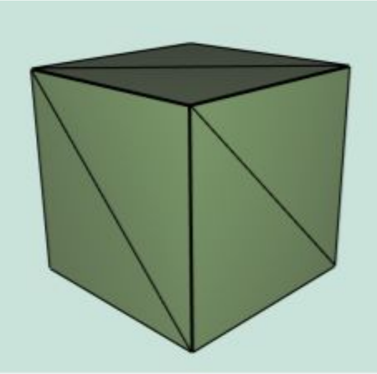
Extrude - Geometry

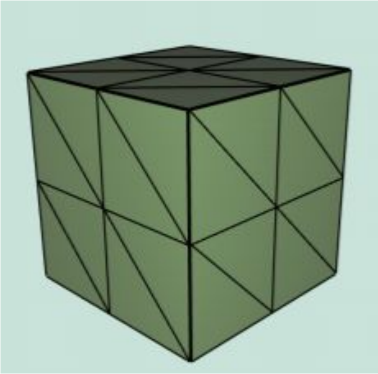
- Simple
  - Move each new vert by factor * f.normal



Triangle Topology

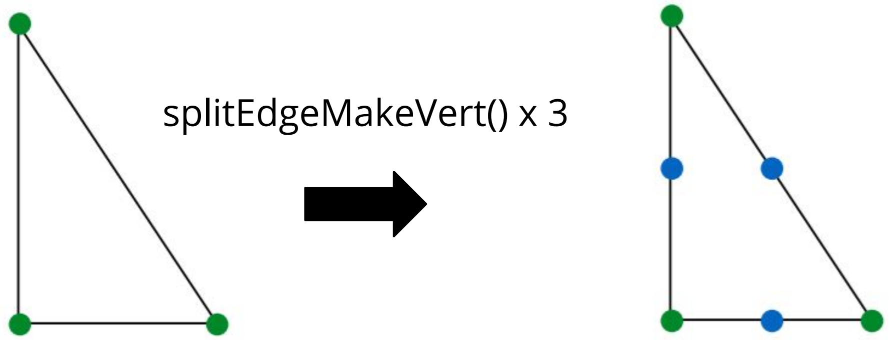
- Call Filters.triangulate()
  - First split all n-gons into triangles
- We want each face to become 4 faces by splitting each edge in half
- For each face:
  - Add 3 vertices
  - Add 3 faces





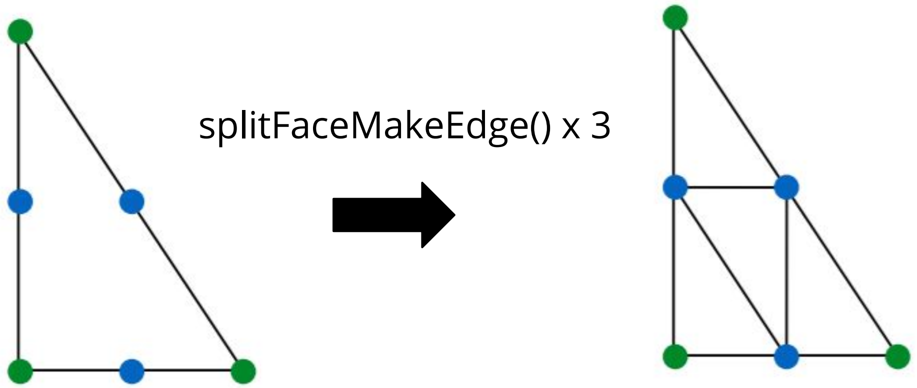
TriTop - Topology

- First, split all edges
  - Create a list of all half edges beforehand
    - Why? When you split a half edge, opposite will be split, so you need to keep track - avoid double splitting



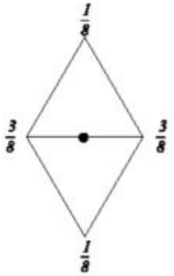
TriTop - Topology

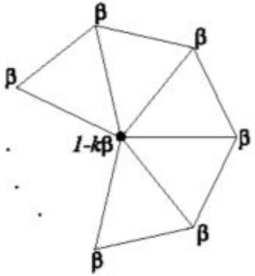
- Join new vertices around a face
  - Keep track of new indices by index - new ones are always added to end of verts array
- Do edge splits and join verts in separate loops



TriTop - Loop Subdivision

- Calculate new positions of vertices as you perform triangle topology
  - Find positions of old verts before adding new verts, and positions of new verts before joining them
- One TriTop is done, update positions



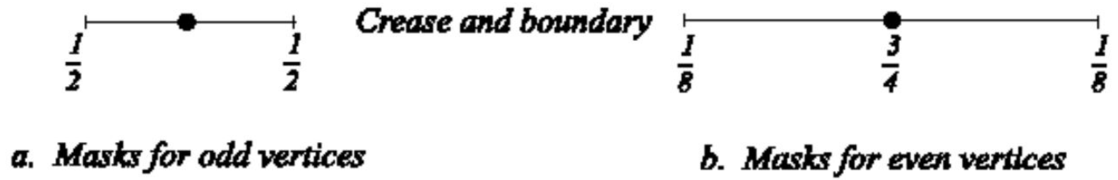


$$\beta = \begin{cases} \frac{3}{8n} & n > 3 \\ \frac{3}{16} & n = 3 \end{cases}$$

These weights are w/ respect to the old vertices!

TriTop - Loop Subdivision

- On boundary edges, use a different mask:

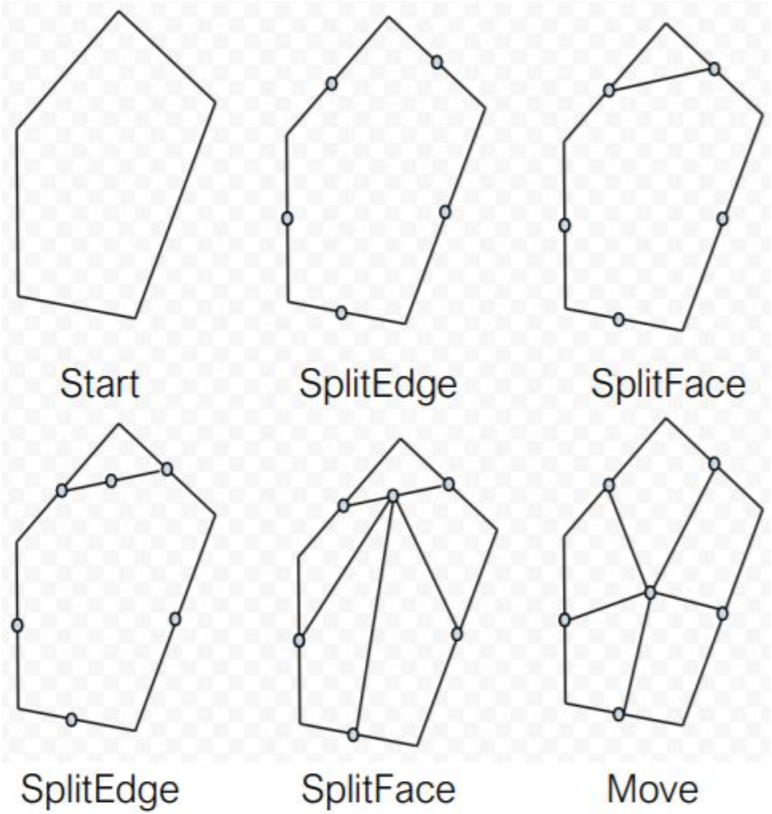


- To prevent degenerate faces, non-selected faces that touch the boundary should receive a TriTop subdivision.

Quad Subdivision - Quad Topology

- Divide an N-gon into quadrilaterals
  - Split each edge
  - Join any 2 new vertices
  - Split this new edge, denote this vert nv_0
  - Join the rest of the new vertices with nv_0
  - Move nv_0 to centroid
- Just as in TriTop, don’t do redundant splits

Quad Topology

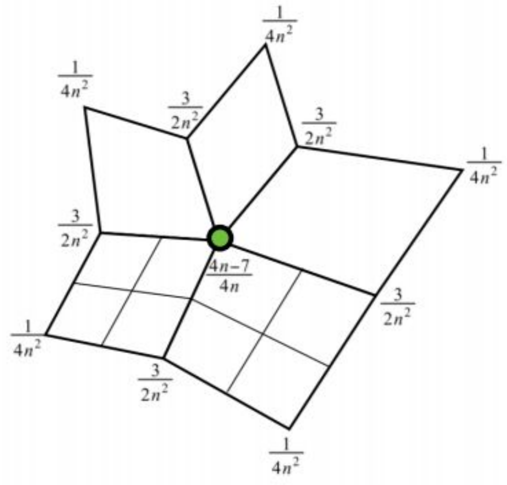


Catmull-Clark Subdivision

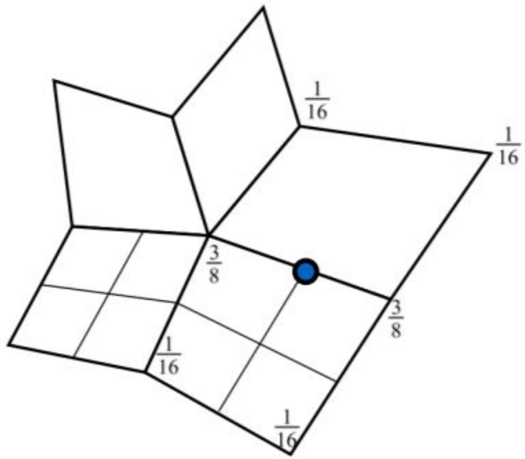
- One possible method:

n = number of neighbors of vert

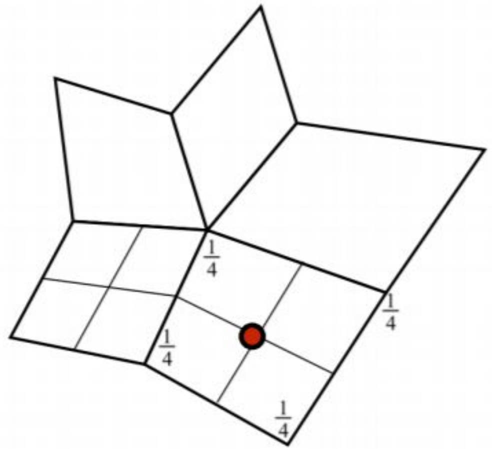
Old Vertices



MidPoints



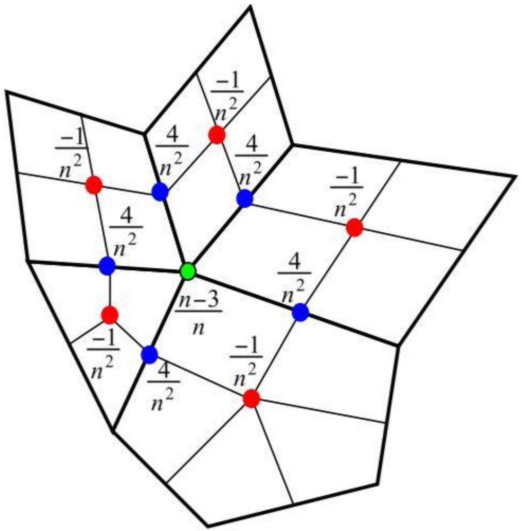
Centroids



Catmull-Clark Subdivision

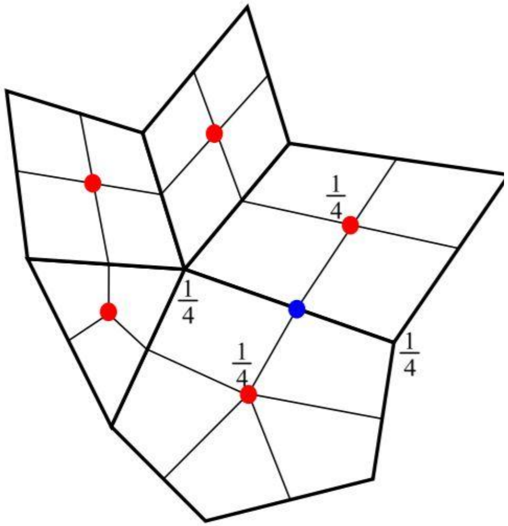
- Another possible method:

Old Vertices

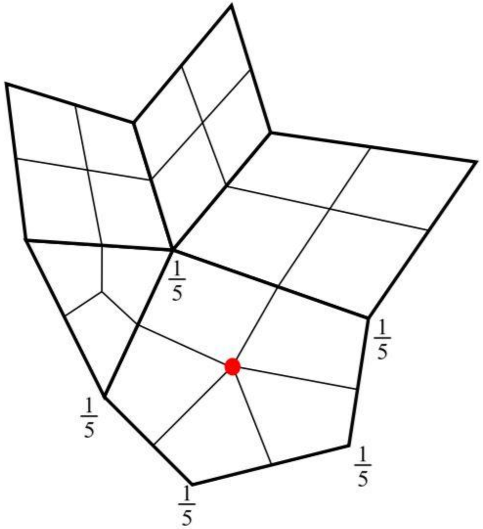


n = number of neighbors of vert

MidPoints



Centroids



Catmull-Clark Subdivision

- And a third:

Old/Even Vertices

- Let F = average of n neighboring face centroids
- Let R = average of n neighboring edge midpoints
- Let p = current position
- The new position is (F + 2R + (n - 3)p) / n

MidPoints

- Receive the average of all their neighbors (centroid vertices, and even vertices)

Centroids

- Receive centroid of their face before the update

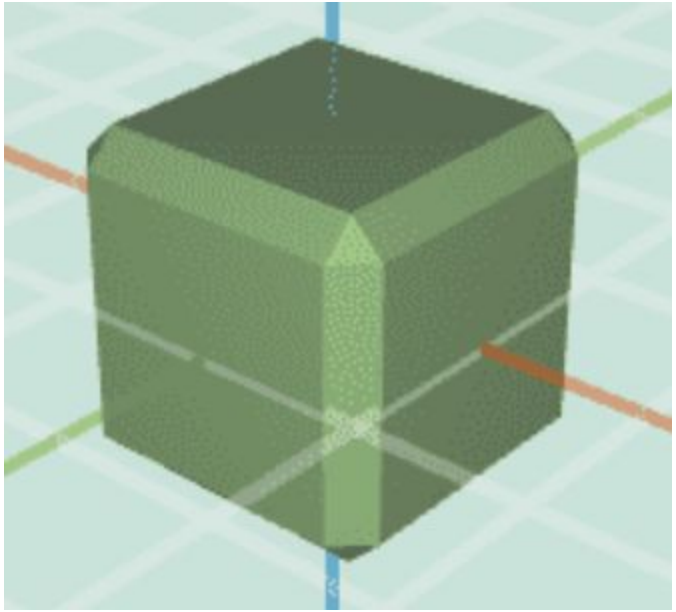
Catmull-Clark Subdivision

- Boundaries?

  Same as Loop... but trickier to implement correctly.

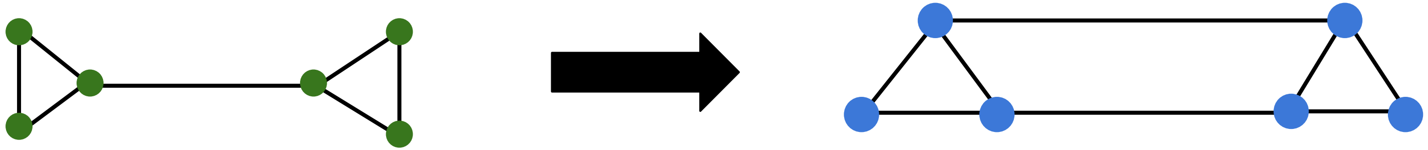
Bevel

- We want to "flatten" corners and edges
    - Each edge "becomes" a face
    - Each vertex "becomes" a face



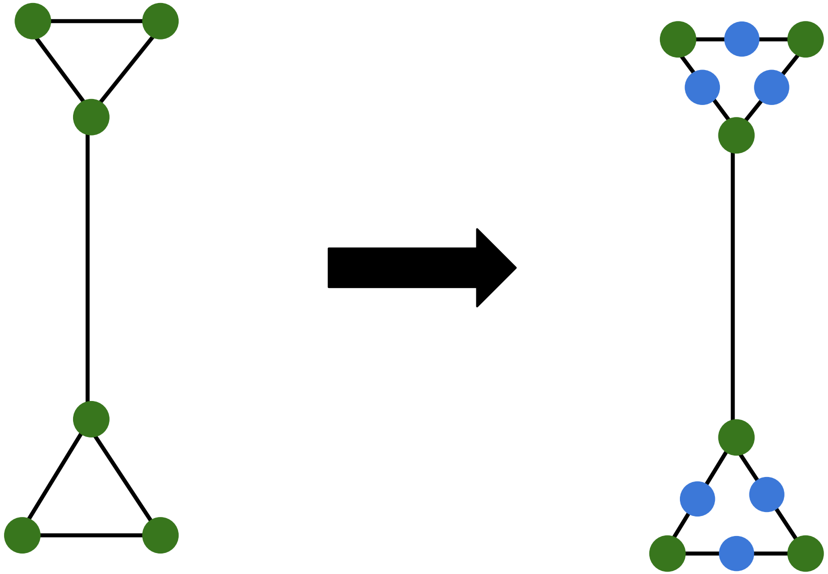
Bevel - Topology

- A good place to start is calling truncate
  - This already flattens each vertex
- Now we want to convert edges to faces
  - Let’s consider one edge



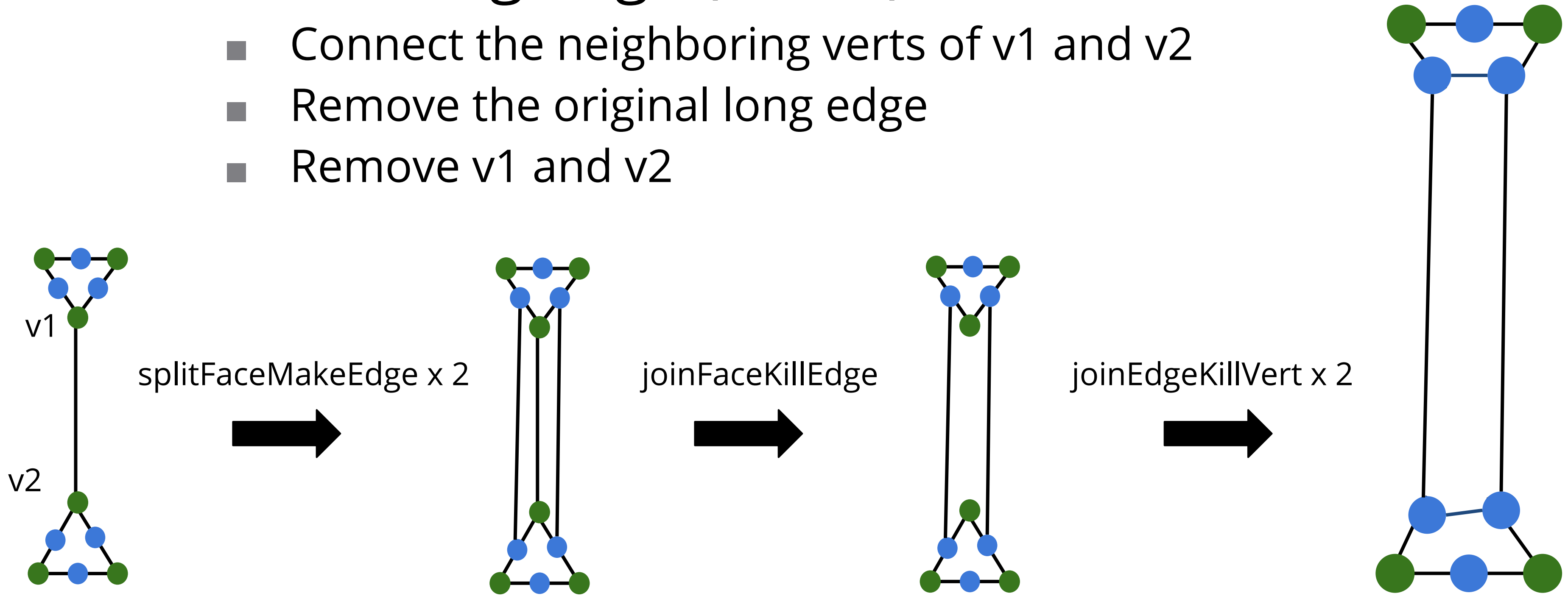
Bevel - Topology

- For each corner face, split all of its edges in half



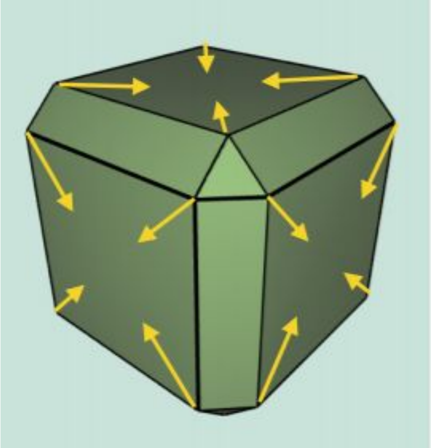
Bevel - Topology

- For each long edge (v1, v2)...
  - Connect the neighboring verts of v1 and v2
  - Remove the original long edge
  - Remove v1 and v2



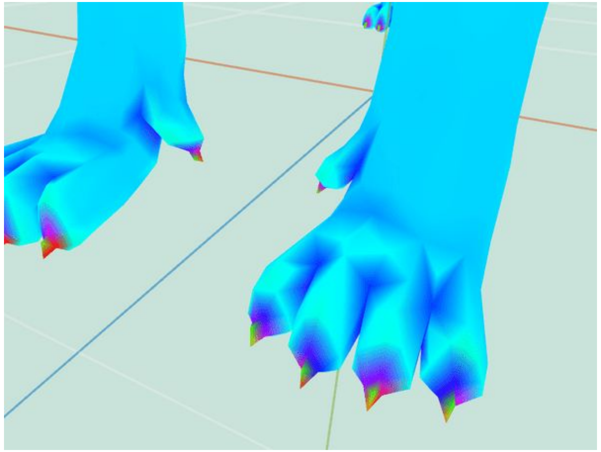
Bevel - Geometry

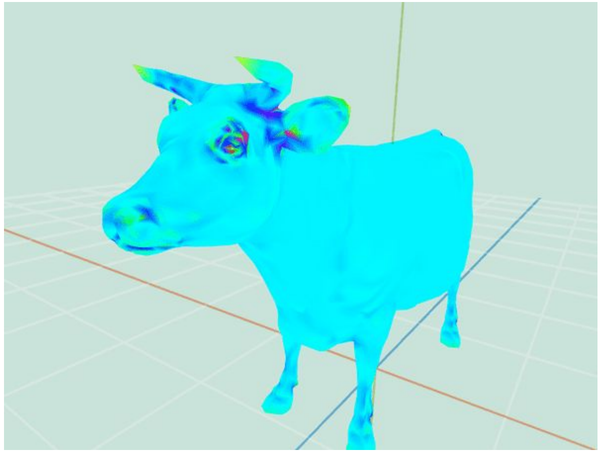
- Simply move each vertex closer to the centroid of its corresponding face



Curvature

- We want to calculate the curvature associated with a vertex
- Then color it based on its curvature



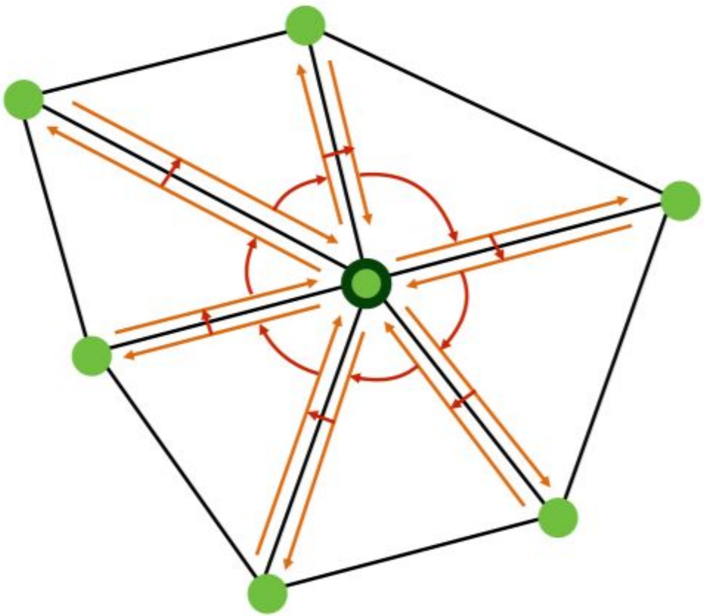


Curvature

- This paper: Akleman, 2006
- Section 2.2 is the most relevant part
  - Area associated with vertex = Sum of area of faces neighboring vertex
- (This makes for really good art submissions!)

Uniform Laplacian Smoothing

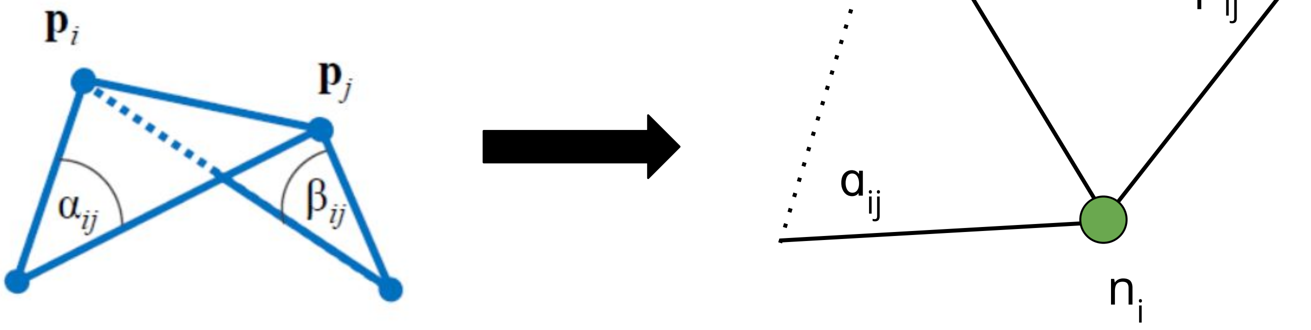
- To update vertex position
  - $v_{new} = v + (\Sigma n_i - N * v) * \delta$
    - n_i = neighbor position
    - N = num neighbors



Curvature-Flow (Cotan) Smoothing

- Mesh must be triangular
- To update vertex position
  - $v_{new} = v + (\Sigma n_i * w_i - v * \Sigma w_i) * \boldsymbol{\delta}$

$$w = \frac{\cot(\alpha_{ij}) + \cot(\beta_{ij})}{2}$$



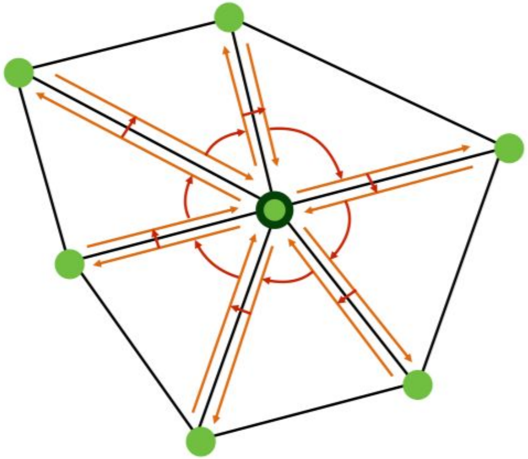
Scale-Dependent Smoothing

- Scale delta to $\delta \cdot \frac{A}{A_v}$ where

$$A_v = \sum_{f_i \in 1ring} area(f_i)$$

$$A = \frac{1}{N_v} \cdot \sum_{v_i \in V} A_{v_i}$$

$$A = \frac{3}{N_v} \cdot \sum_{f_i \in F} area(f_i)$$



Q&A