Assignment 2: Advanced Features

COS 426: Computer Graphics (Fall 2022)

Guðni Gunnarsson, Yuanqiao Lin, Yuting Yang

Agenda

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

Logistics

- Midterm is Thursday, 10/13
  - Practice exam is released
  - Next week's precept will be a review session
  - Exercises page

One Primitive A Time

- 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?
- Keep track of new vs old primitives (faces, vertices, half edges)
    - New primitives are always added at the end of their respective arrays

Other Tips

- Count primitives after modifications
  - `console.log` is your friend!
- Draw your operations out
- Check your helper functions and mesh traversal functions
- Applying operations to *selected* primitives

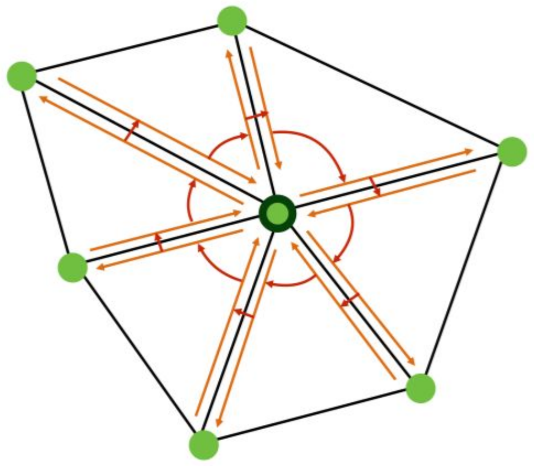
Scale-Dependent Smoothing

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

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

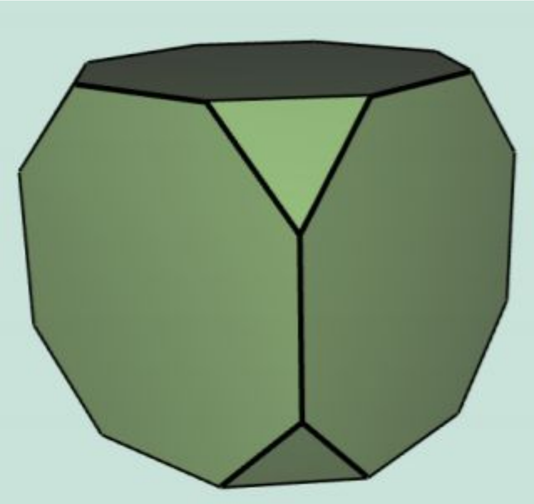
$$A = \frac{1}{N_v} \cdot \sum_{v_i \in V} A_{v_i}$$

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



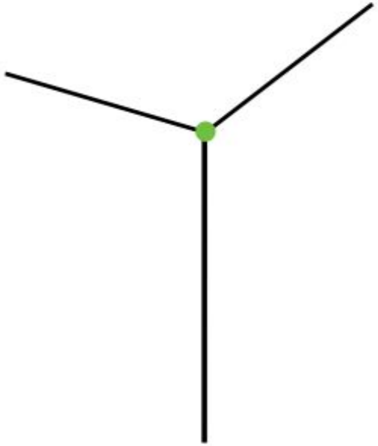
Truncate

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

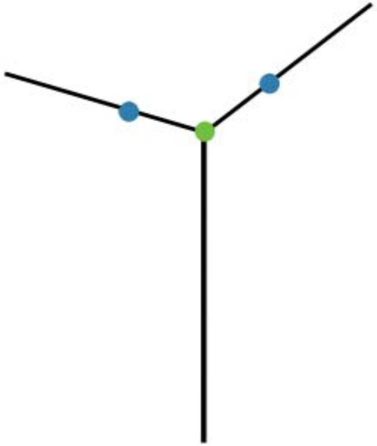


Truncate - Topology

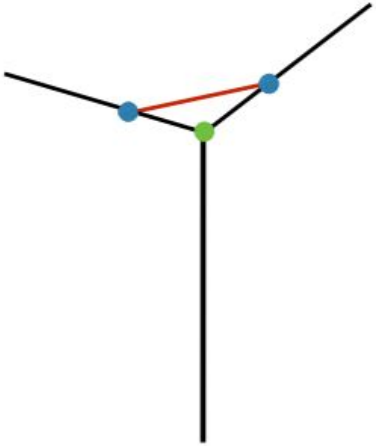
- Consider a vertex with 3 edges
- So we need to add 2 vertices, 1 face



Initial



SplitEdgeMakeVert x 2



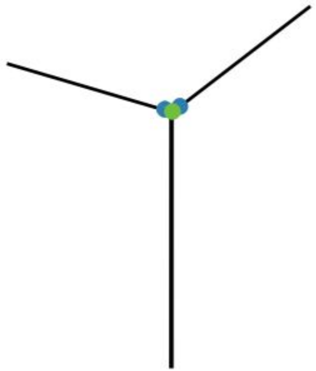
SplitFaceMakeEdge

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

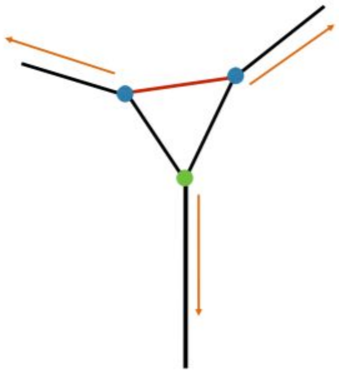
They are moved apart for easier visualization.

Truncate - Geometry

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



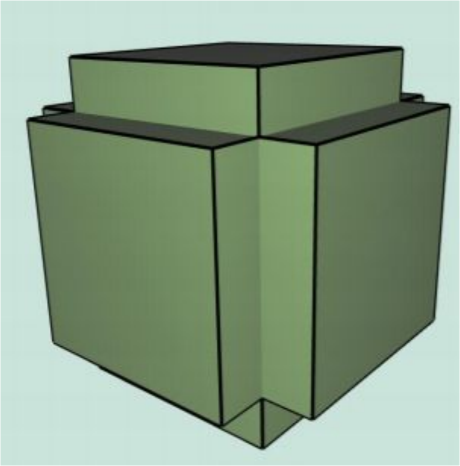
After Making Face



Apply Offsets

Extrude

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



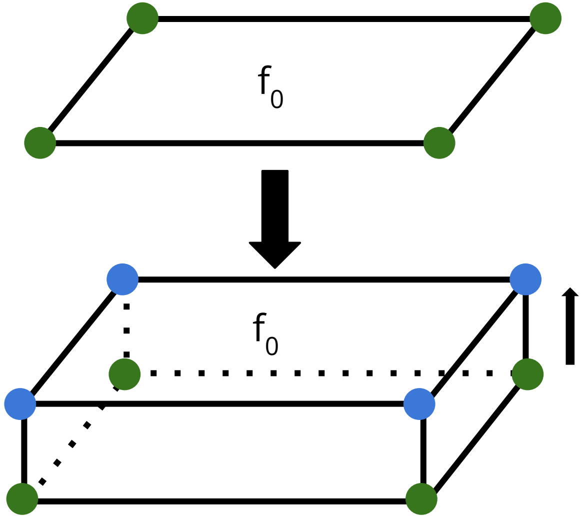
Extrude - Topology

- 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!!!

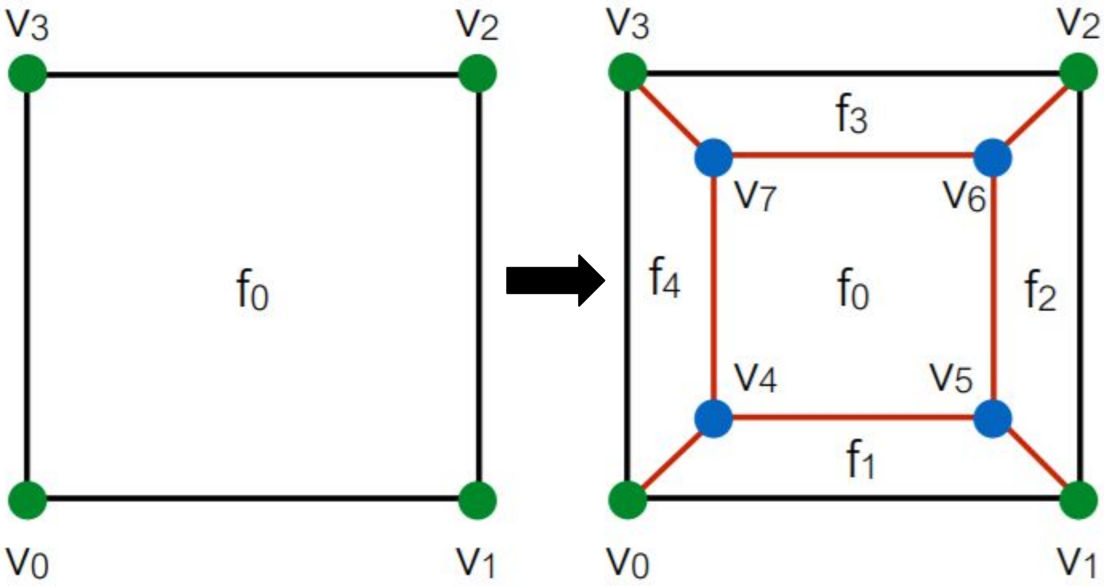
Extrude - Topology

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

3D View:

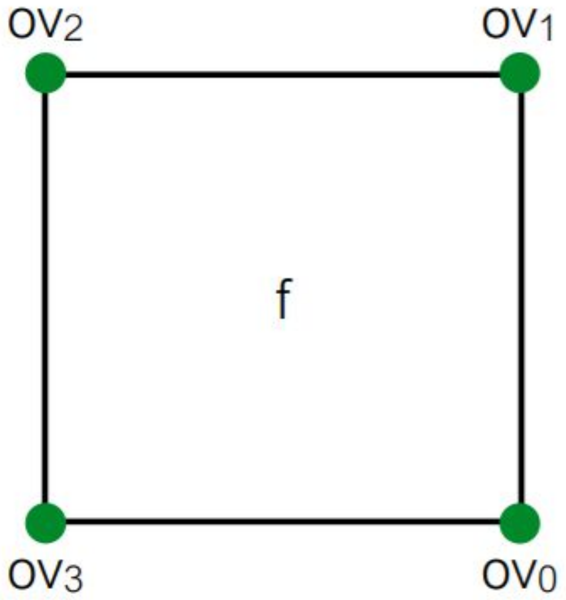


Topological View:



Extrude - Topology

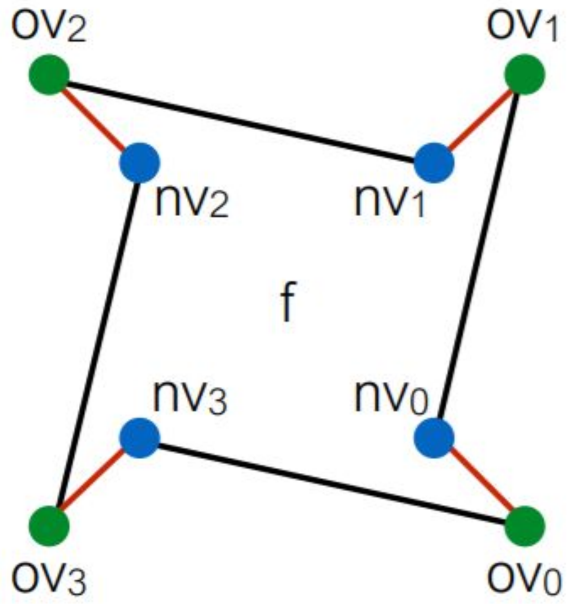
- Denote **o**v for **o**ld vert and **n**v for **n**ew vert



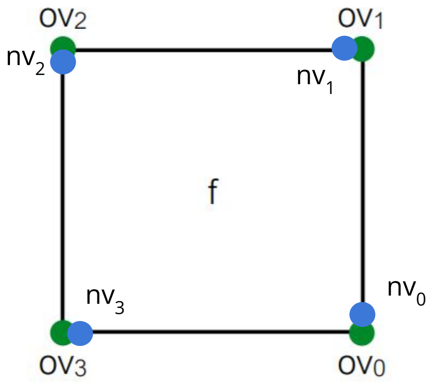
Extrude - Topology

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

Topological View:



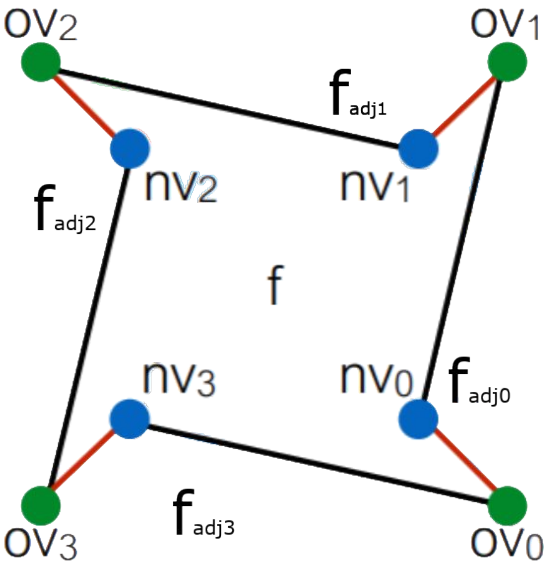
Reality:



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?

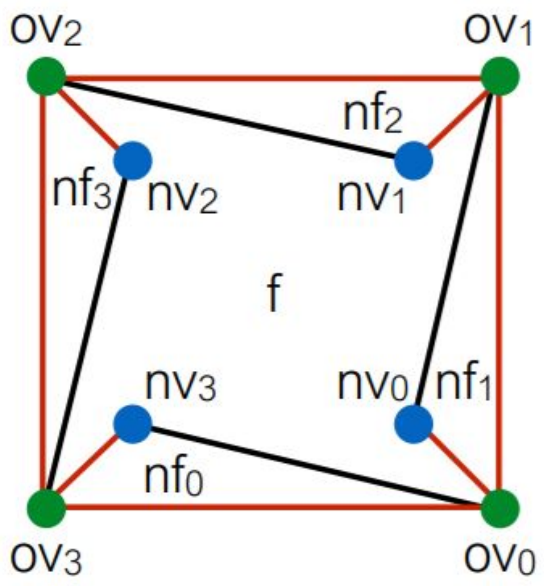
Topological View:



Extrude - Topology

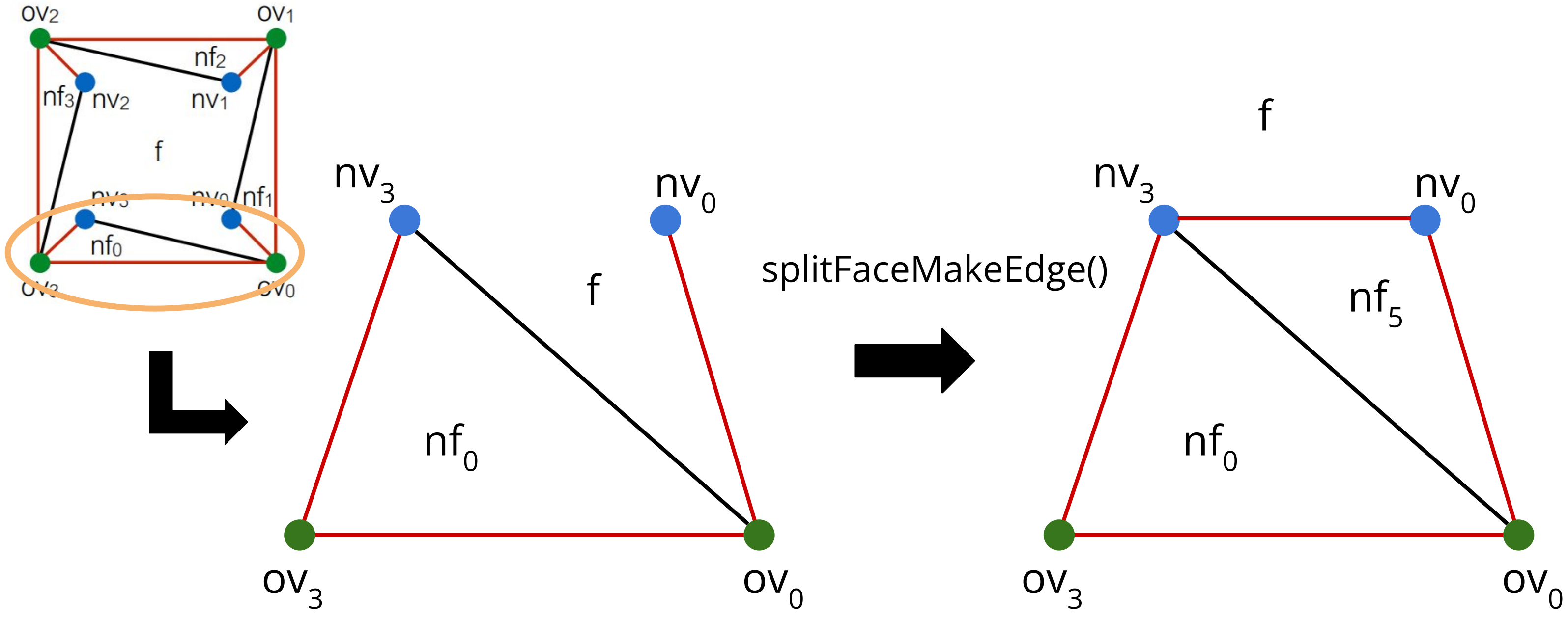
- Then, split 4 **adjacent** faces
  - SplitFaceMakeEdge x 4
  - Between which 2 vertices should we split the face each time?
  - Which vertex would we like on which face at the end?

Topological View:



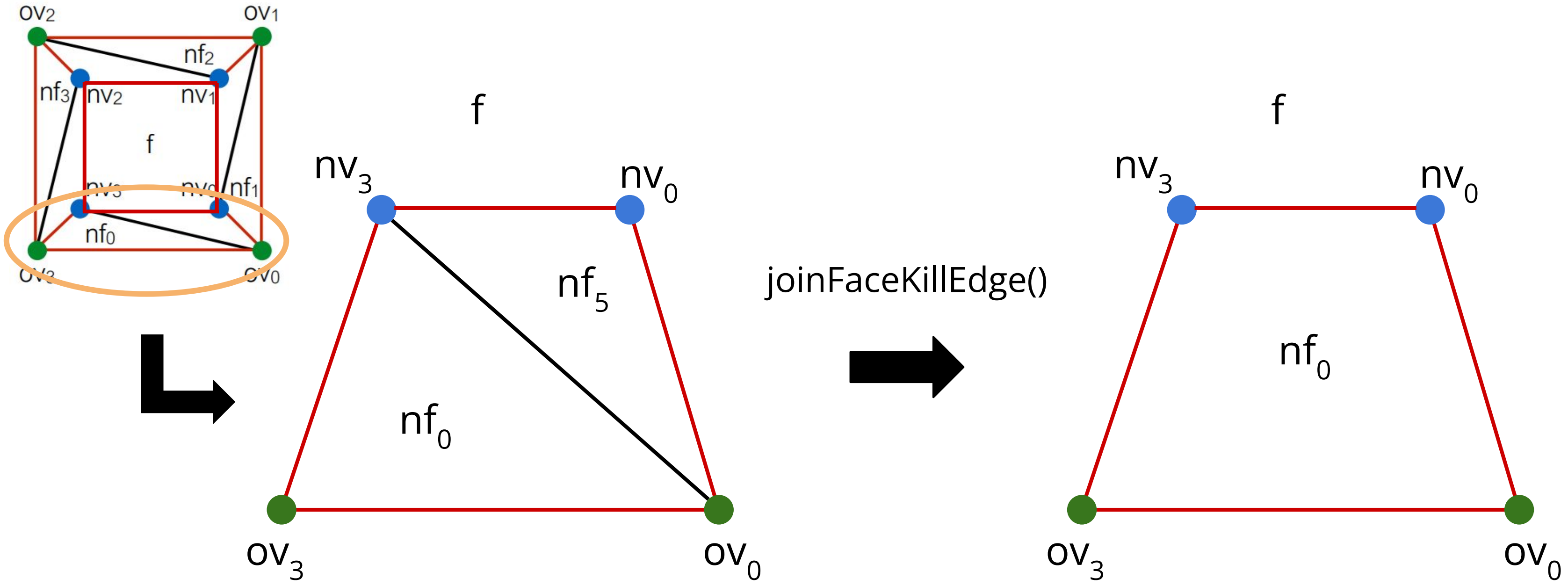
Extrude - Topology

- We want to connect the new vertices



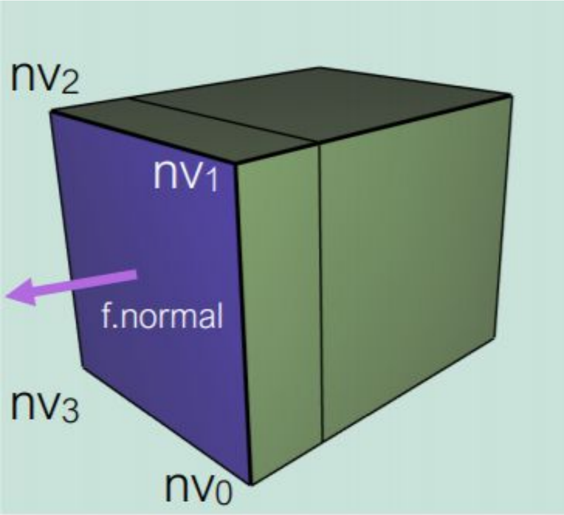
Extrude - Topology

- Now join the two new faces



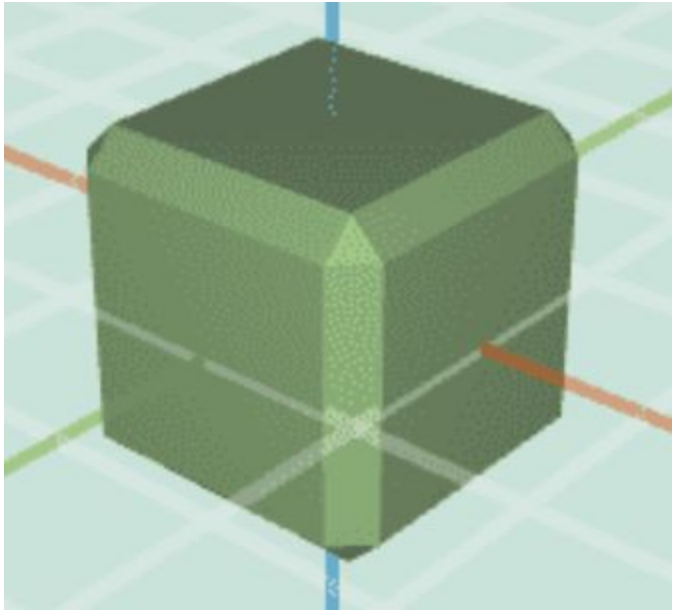
Extrude - Geometry

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



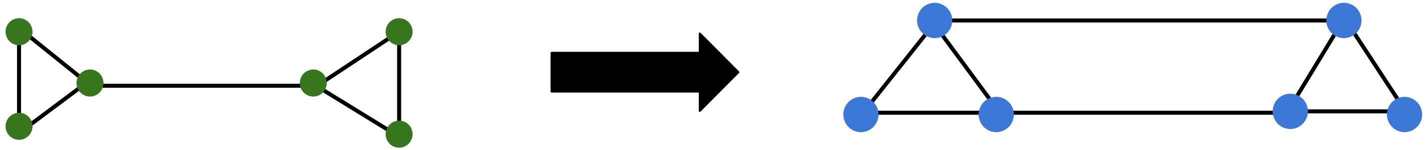
Bevel

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



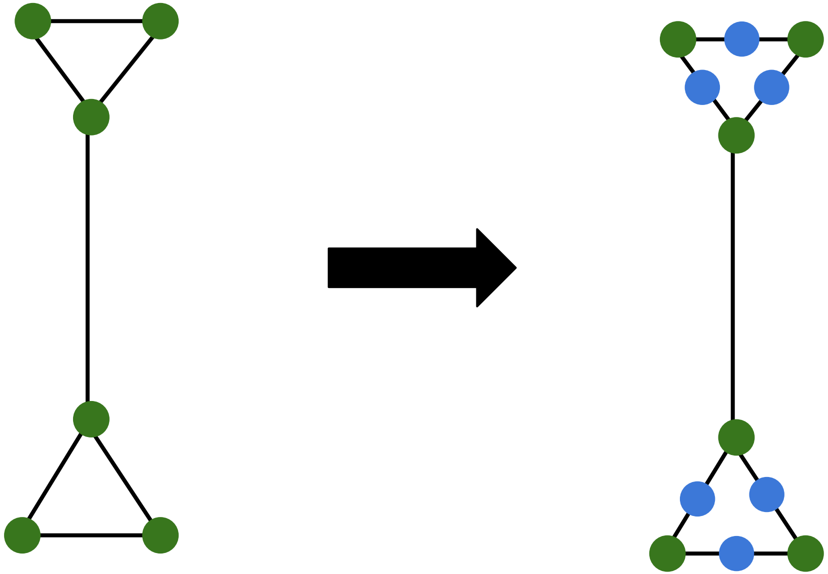
Bevel - Topology

- A good place to start is Vertex => Face, aka truncate
- Now we want to convert edges to faces
  - Let’s consider one edge



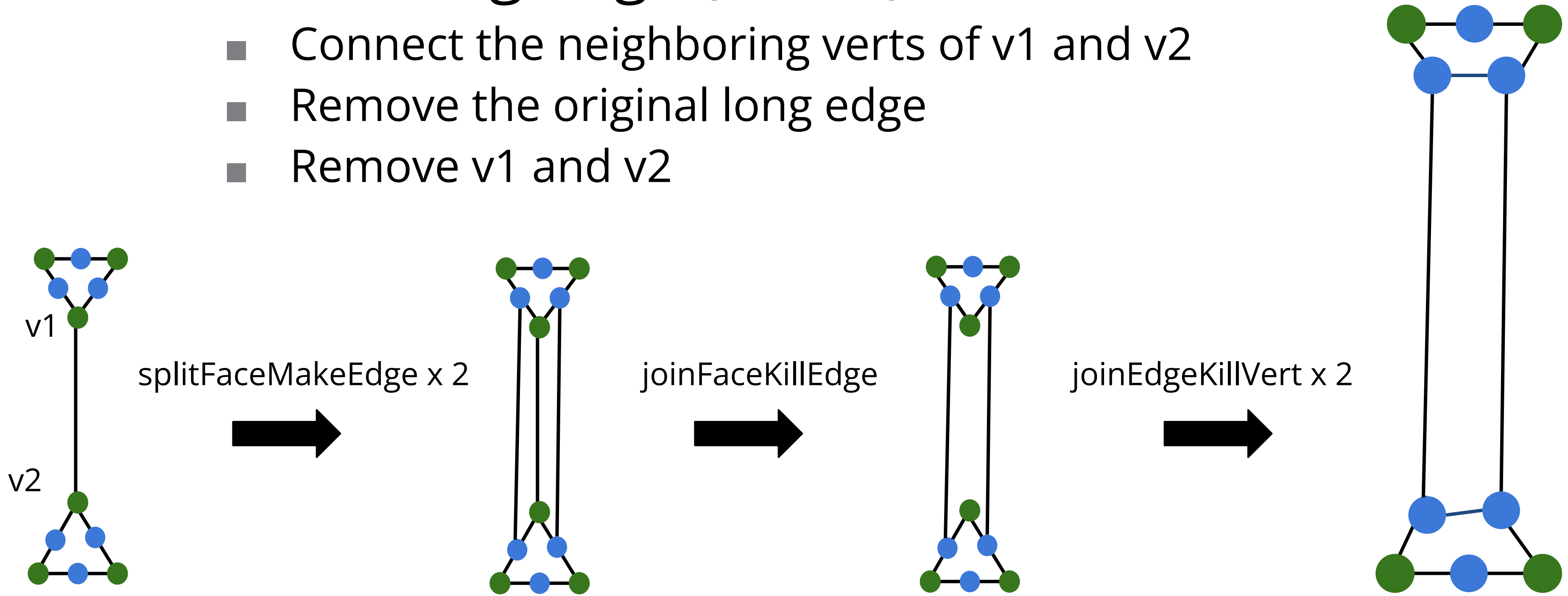
Bevel - Topology

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



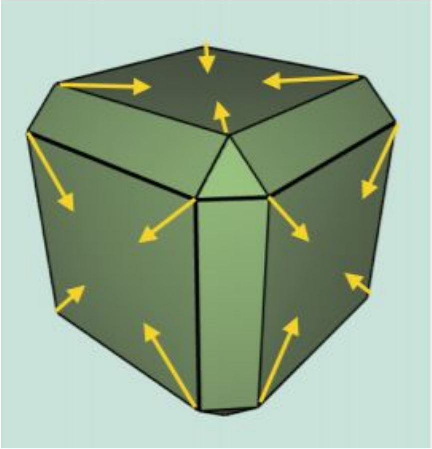
Bevel - Topology

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



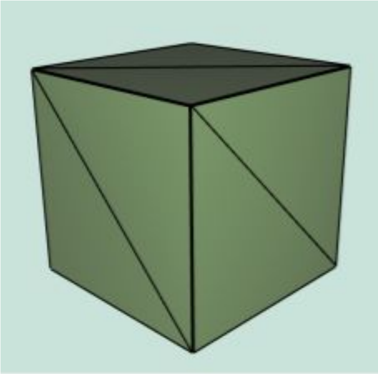
Bevel - Geometry

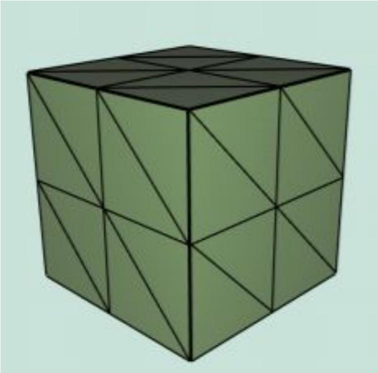
- Simply move each vertex closer to the centroid of its corresponding face based on the factor parameter



Triangle Topology

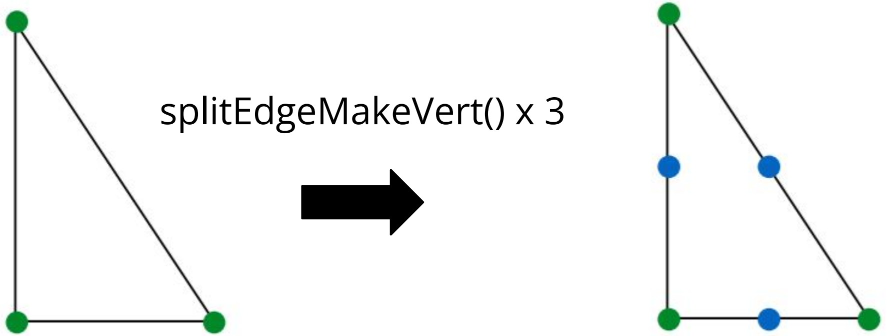
- Splits each selected face in the mesh into four triangles
- First, split all n-gons into triangles
  - Filters.triangulate()





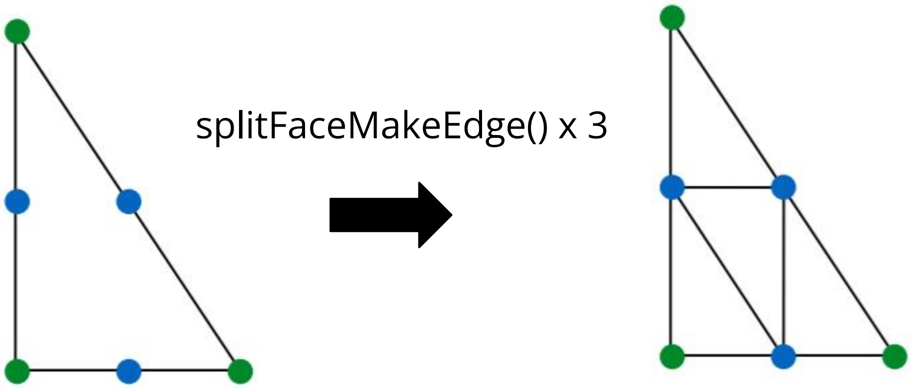
TriTop - Topology

- Split all edges
  - For each face, add 3 vertices and 3 faces
  - Create a list of all half edges beforehand
    - When you split a half edge, opposite will be split, so you need to keep track - avoid double splitting



TriTop - Topology

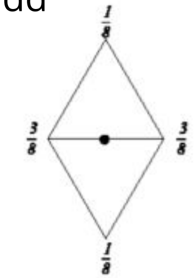
- Join new vertices around a face
  - Keep track of new indices by index - new ones are always added to end of verts array
- Do edge splits and join verts in separate loops



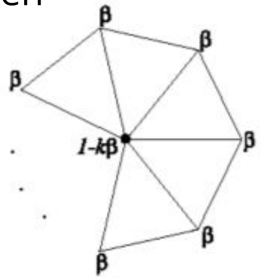
TriTop - Loop Subdivision

- 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

Odd



Even



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

These weights are w/ respect to the old vertices!

TriTop - Loop Subdivision

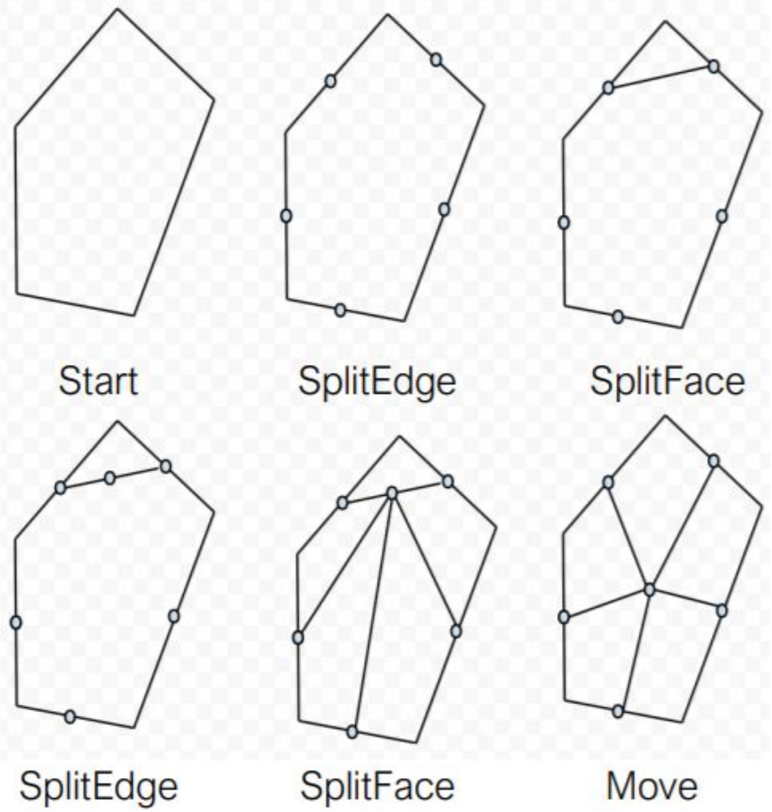
- On boundary edges, use a different mask:

a. Masks for odd vertices   b. Masks for even vertices

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

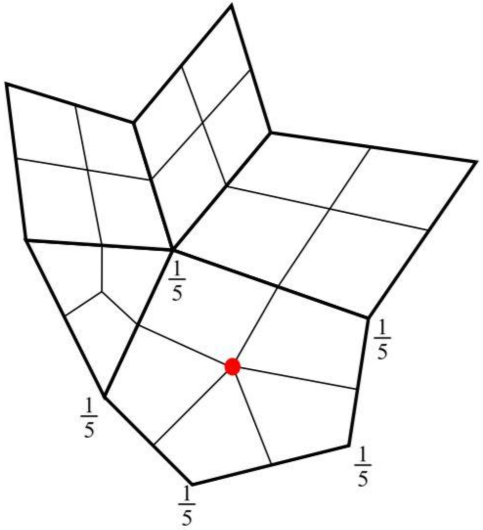
Quad Subdivision

- 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

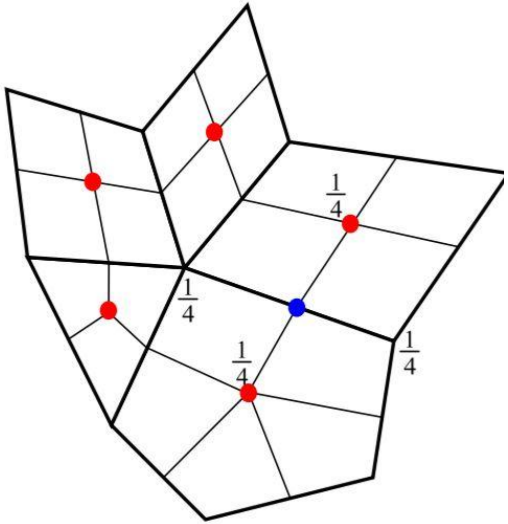


Catmull-Clark Subdivision

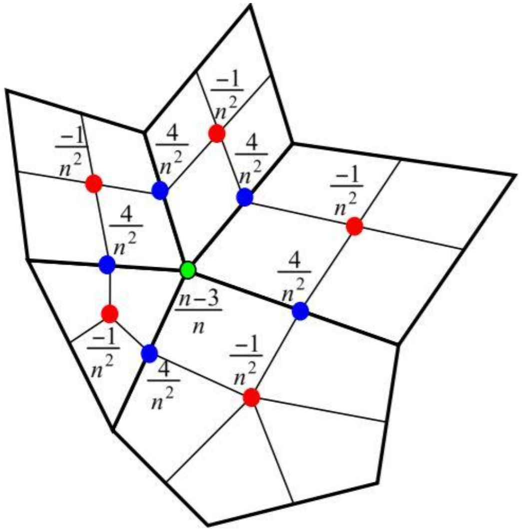
Centroids



MidPoints



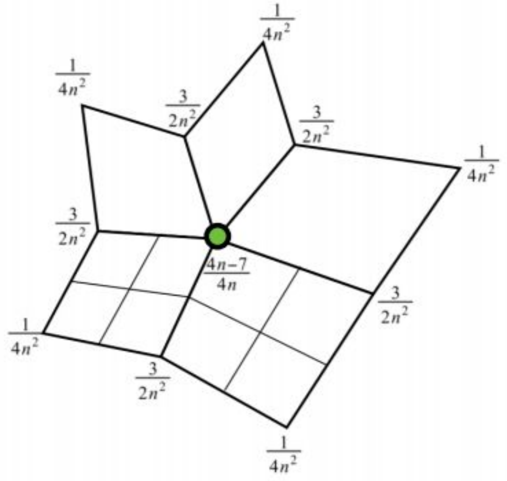
Old Vertices



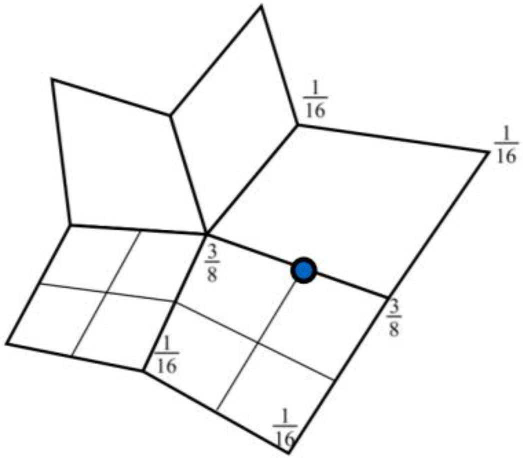
n = number of neighbors of vert

Catmull-Clark Subdivision

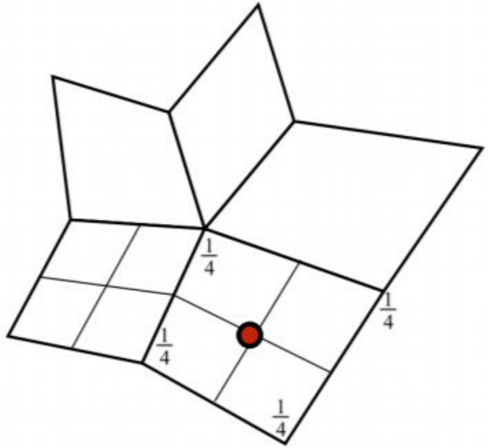
Old Vertices



MidPoints



Centroids



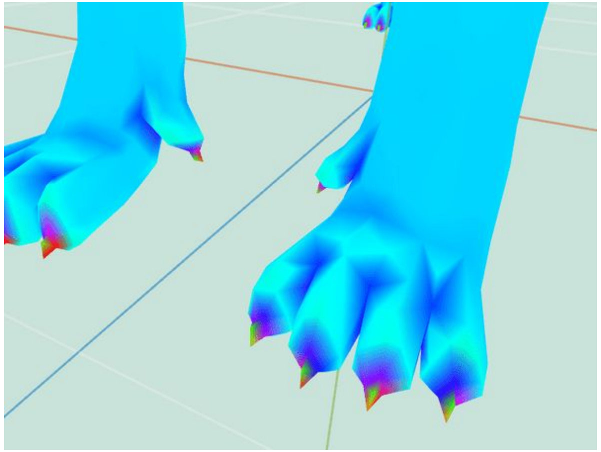
n = number of neighbors of vert

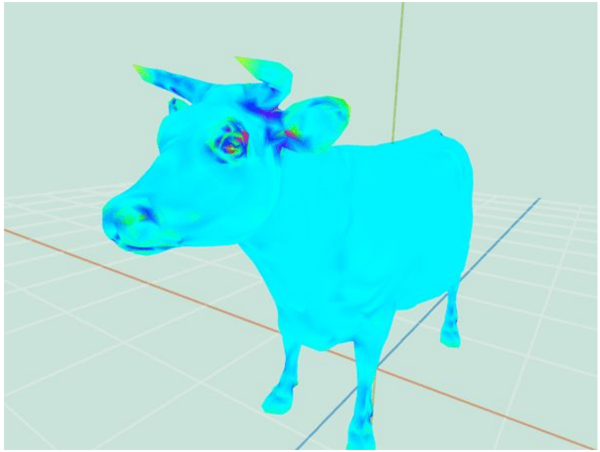
Catmull-Clark Subdivision

- Boundaries: Same boundary weights as loop, but more complicated when dealing with boundary faces.
    - Details are included in the assignment description

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
    - Gaussian curvature = angular deflection / area associated with vertex
    - Area associated with vertex = Sum of area of faces neighboring vertex
- (This makes for really good art submissions!)

Q&A