COS 426 : Precept 5
Working with Half-Edge

Agenda

- How to tackle implementation of more advanced features
- Specific discussion
  - Truncate
  - Extrude
  - Triangle Subdivision
  - Quad Subdivision(?)
  - Smoothing(?)

How do I start?

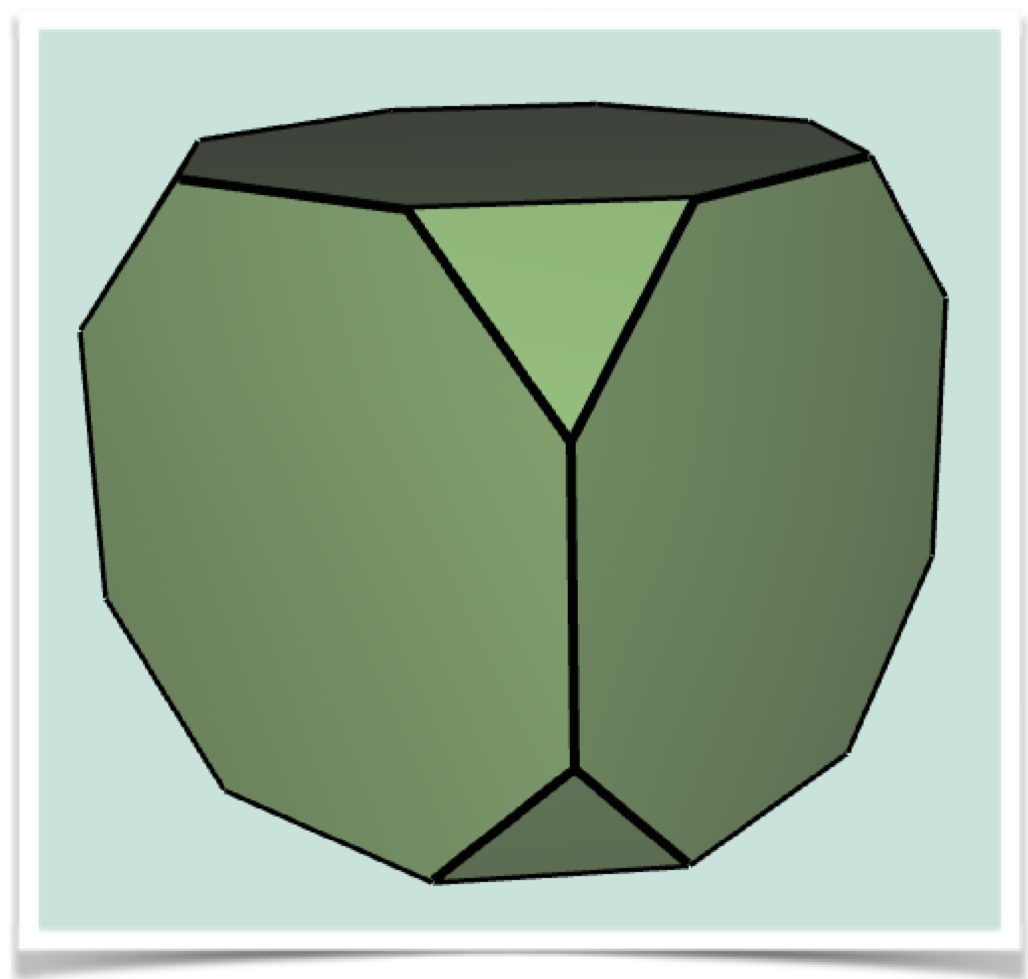
- Some of the operations are tricky to implement!
- Think locally - independence of operations
  - Modifying a vertex/edge/face should not influence other primitives
- Start small
  - Just work on one primitive at a time
- Decouple topology and geometry
  - What are necessary topological changes?
  - What are necessary geometrical changes?
  - Apply geometrical change after topological

Caution is advised

- Need to think ahead
  - What data might change?
  - Do you need to store it beforehand?
- Pen and paper!
  - Draw things out, make sure you understand what is happening
- Count!
  - After applying your operation how many new vertices you expect to see?

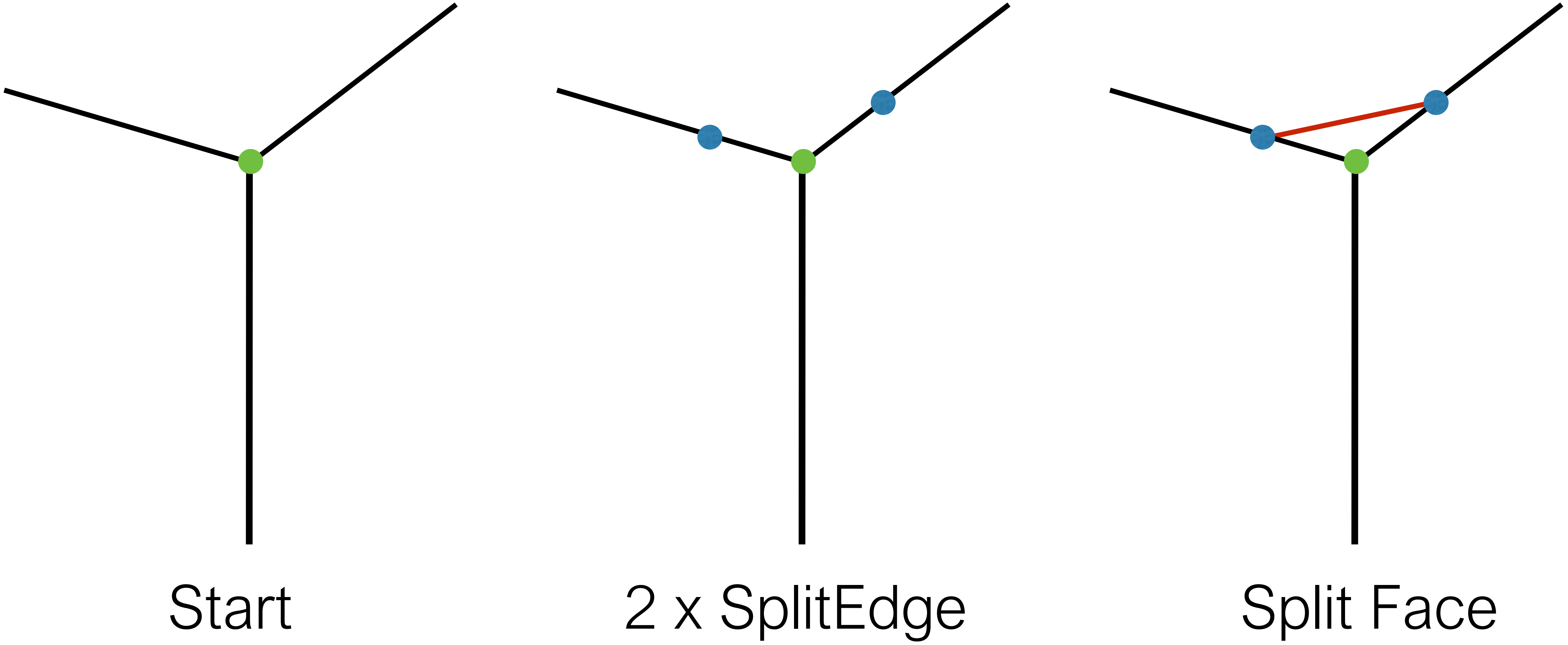
Truncate

- Corners of the shape are cutoff
- Main primitive
  - Vertex
- How many new vertices?
  - +2 per vertex
- How many new faces?
  - +1 per vertex



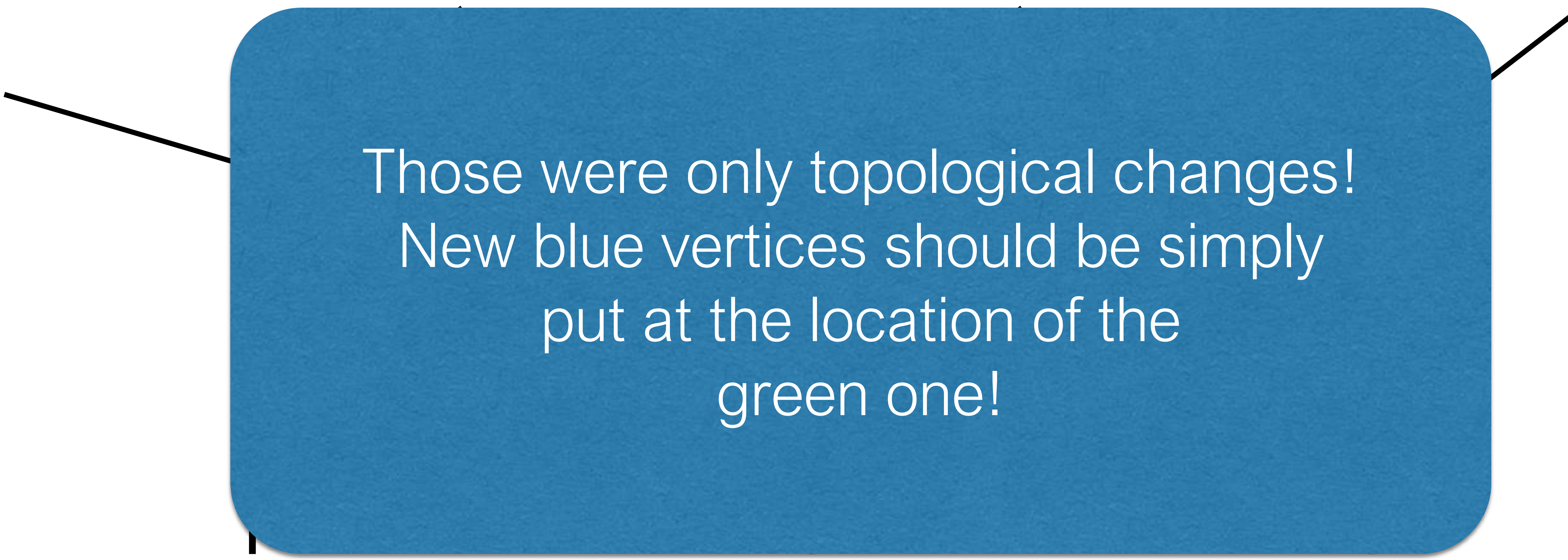
Truncate - topology

- Start locally - just consider single vertex
- Need to add two new vertices, and a single new face



Truncate - topology

- Start locally - just consider single vertex
- Need to add two new vertices, and a single new face

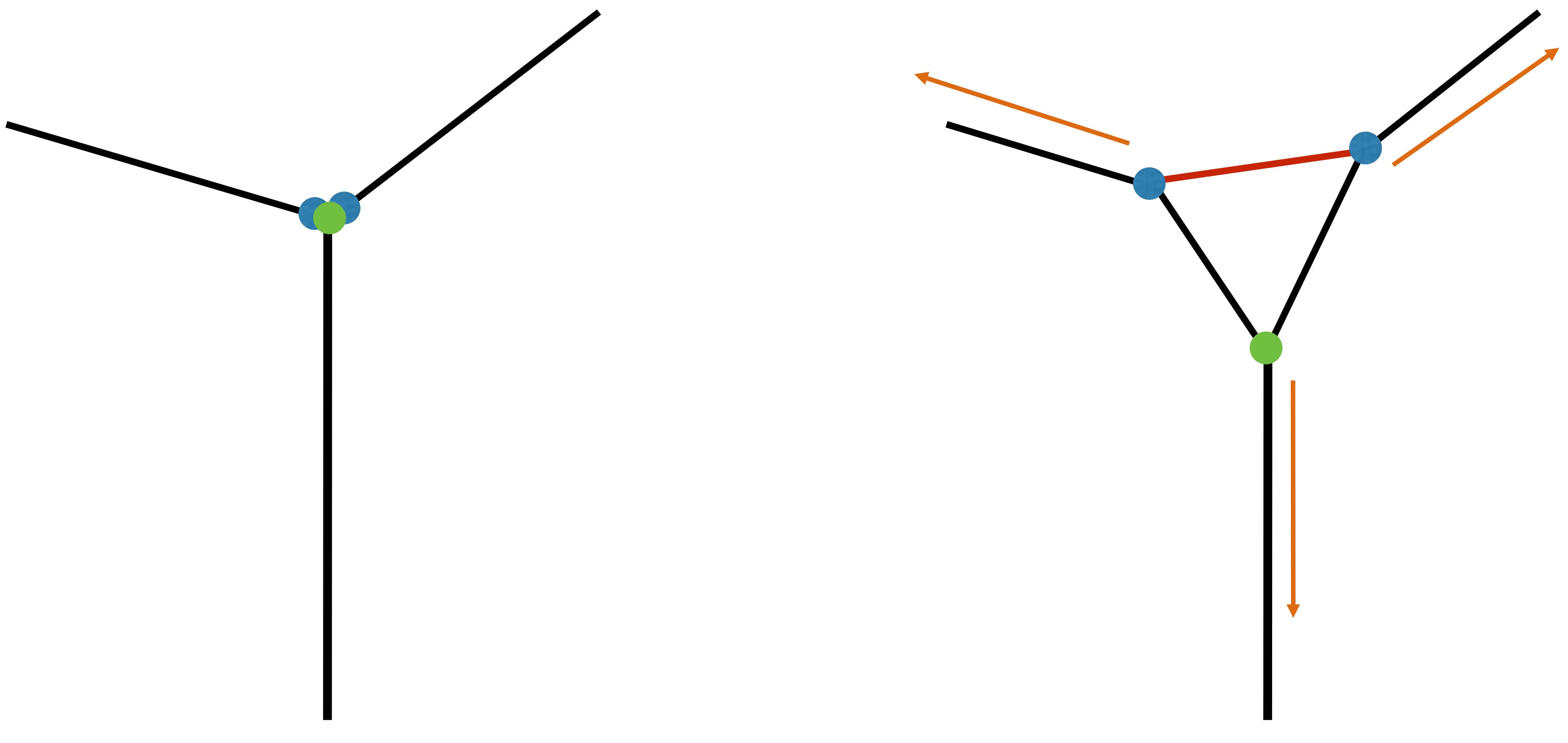


Those were only topological changes!
New blue vertices should be simply
put at the location of the
green one!

Start 2 x SplitEdge Split Face

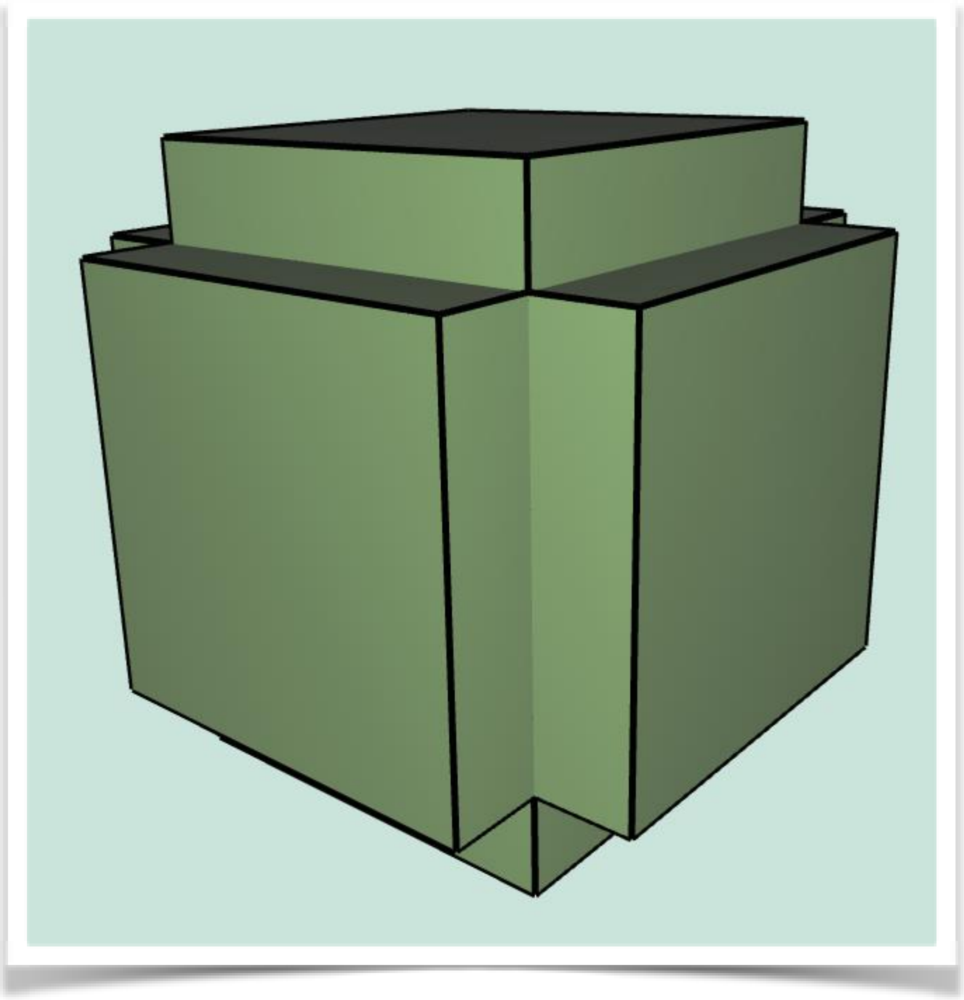
Truncate - geometry

- We need to move vertices along halfedges
  - You may want to store the respective offset vectors per vertex before hand
  - As you modify one vertex lengths of edges will change!



Extrude

- Each face is moved along its normal, with new faces stitched to original face position
- Main primitive
  - Face
- How many new vertices?
  - +n per n-gon
- How many new faces?
  - +n per n-gon



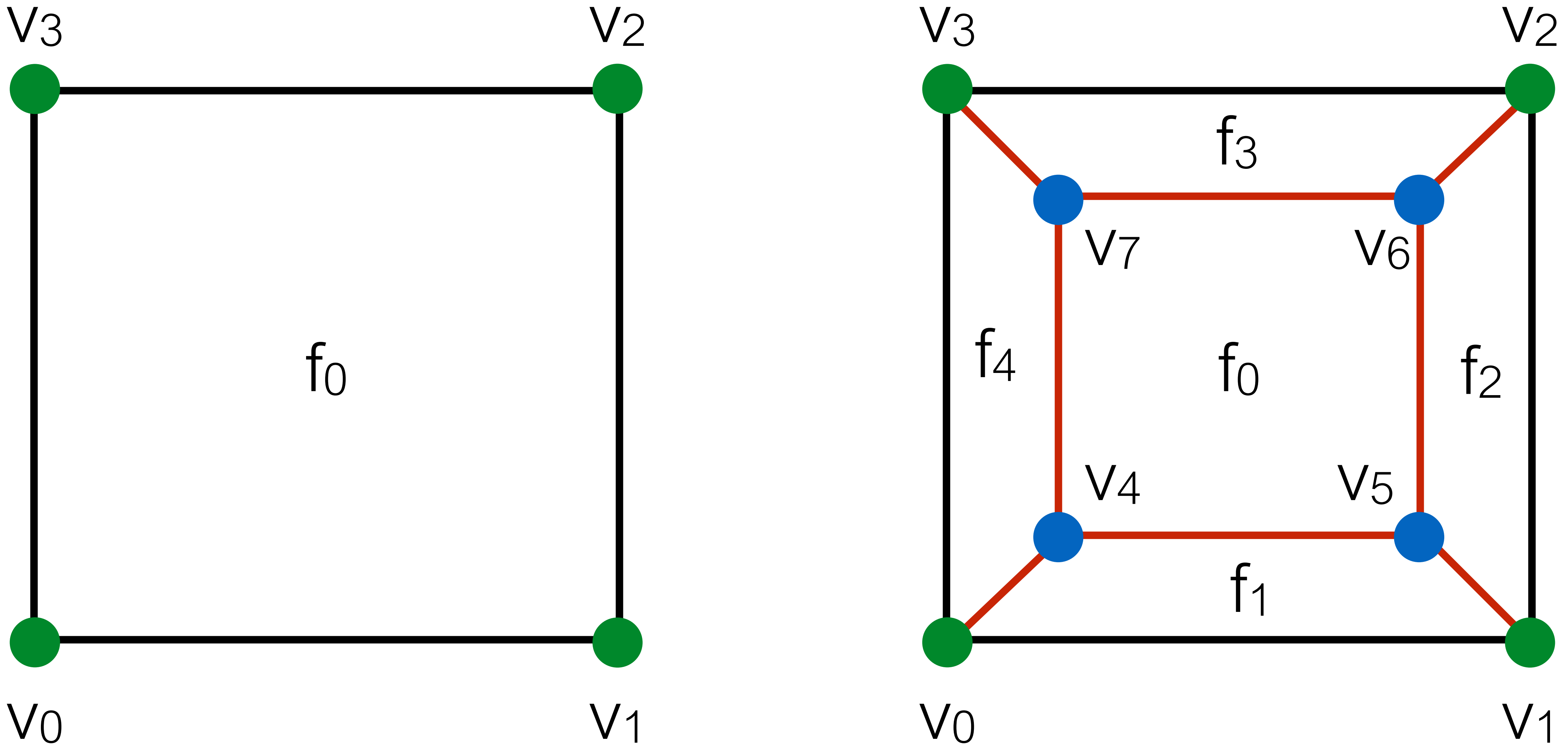
Extrude - topology

- Again, following figures are for illustration only, new vertices should be added at a location of the old ones!

Extrude - topology

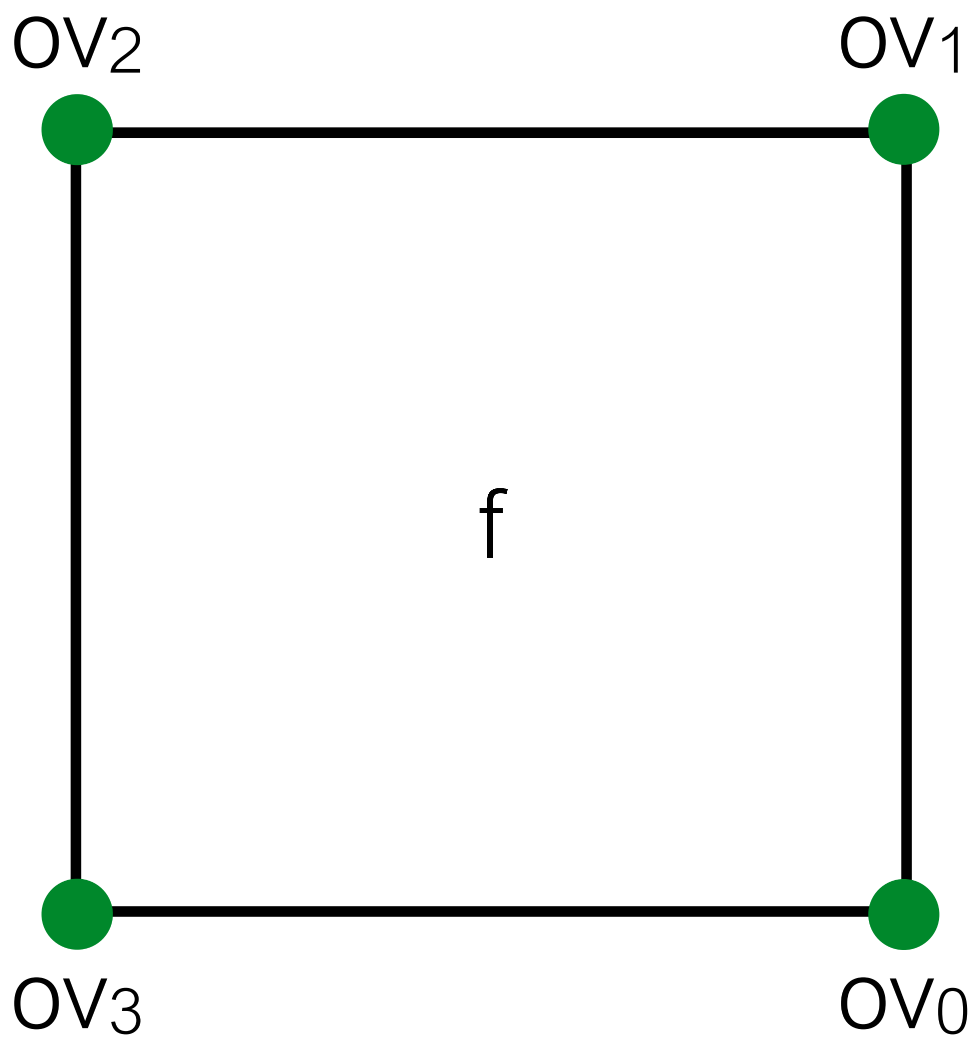
- Extrude is bit harder - you need to perform adding new geometry and relinking manually.

- Desired:



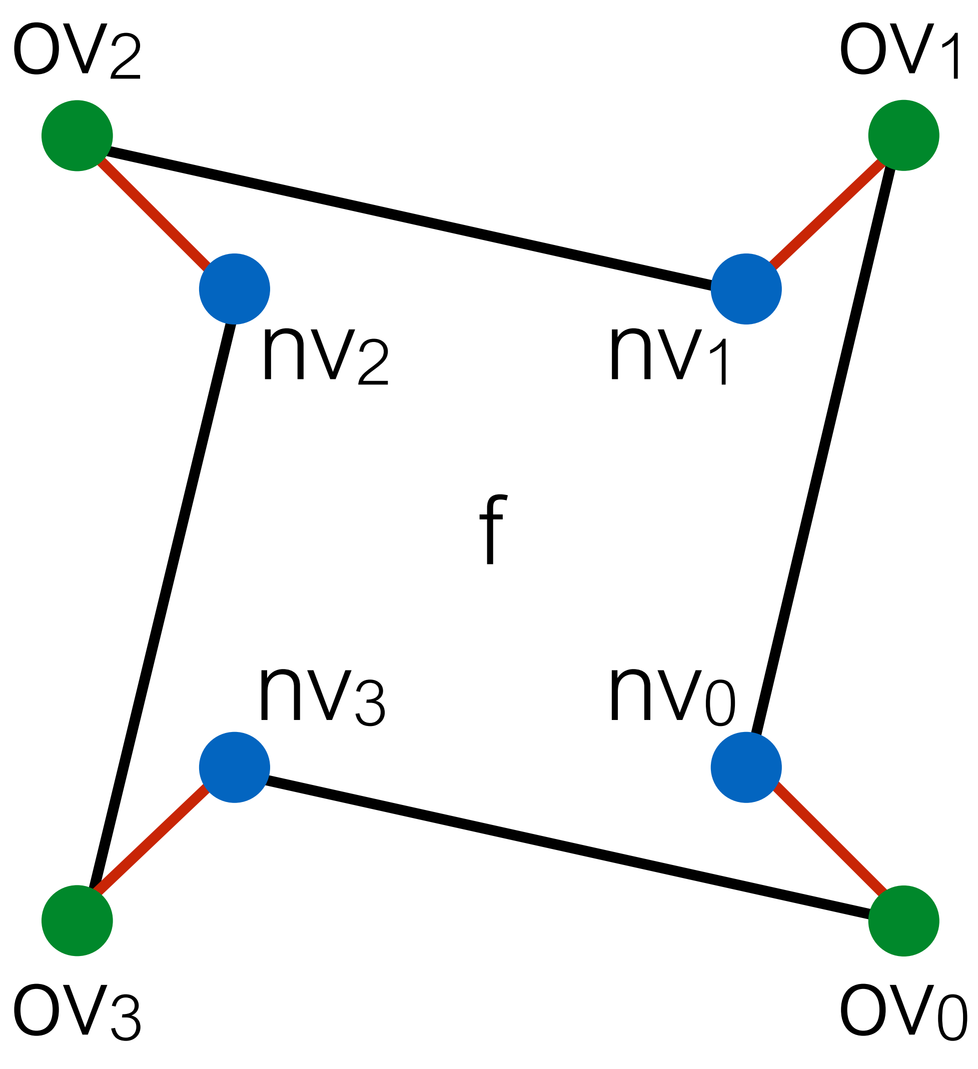
Extrude - topology

- Let's change notation a bit, introduce old and new vertices



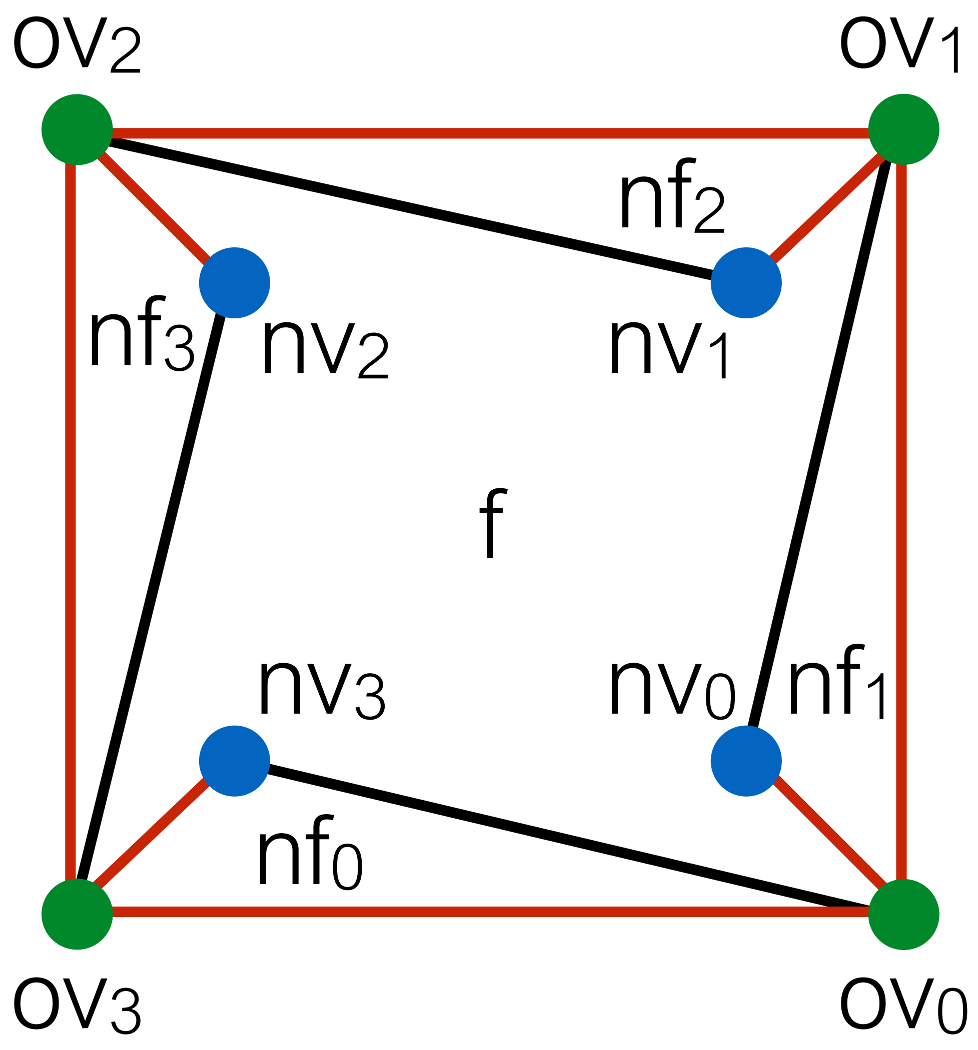
Extrude - topology

- Let's change notation a bit, introduce old and new vertices



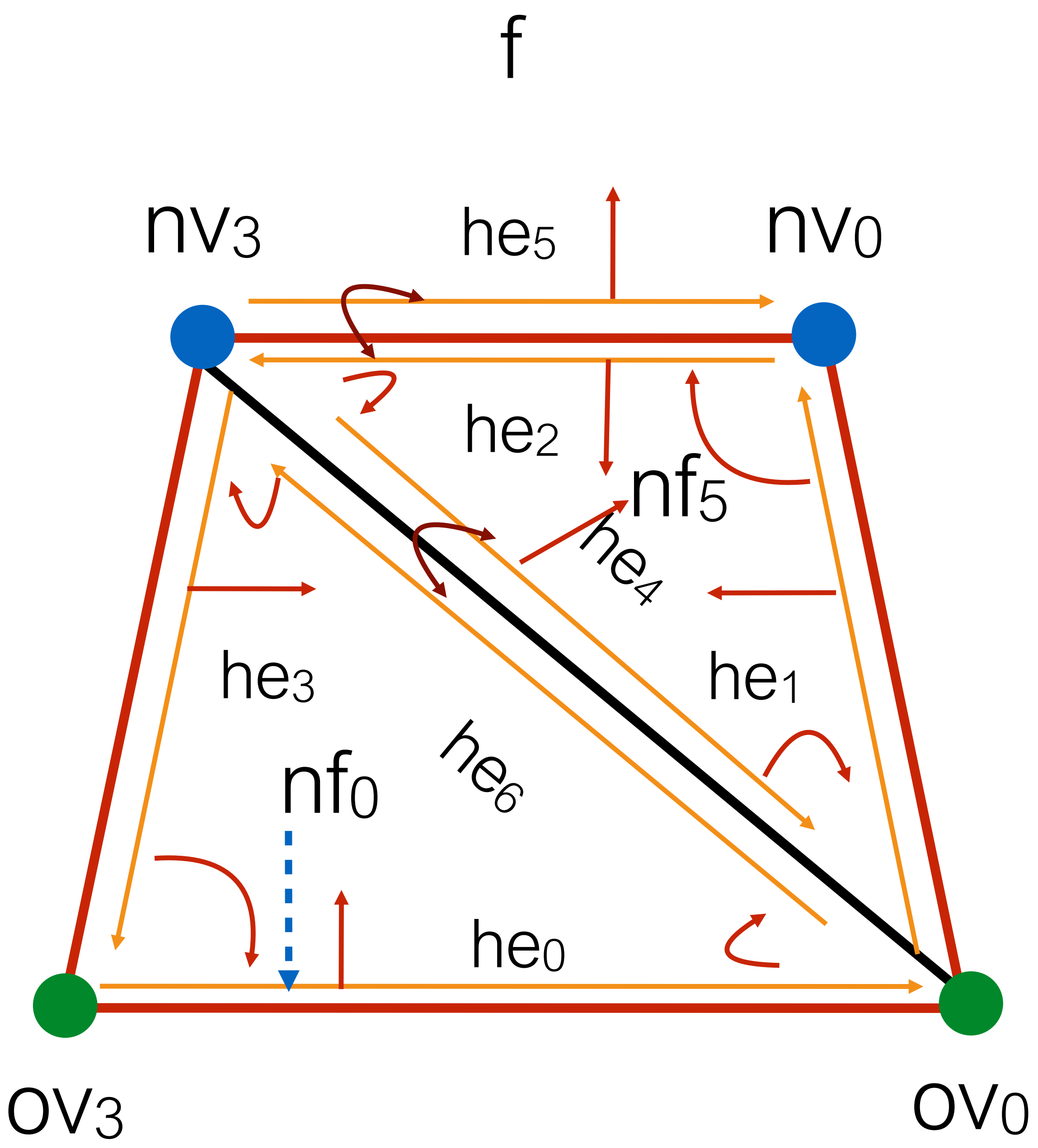
$$nv_i = \text{splitEdgeMakeVert}(ov_i, ov_{i+1}, 0);$$

Extrude - topology



nf_i = splitFaceMakeEdge();

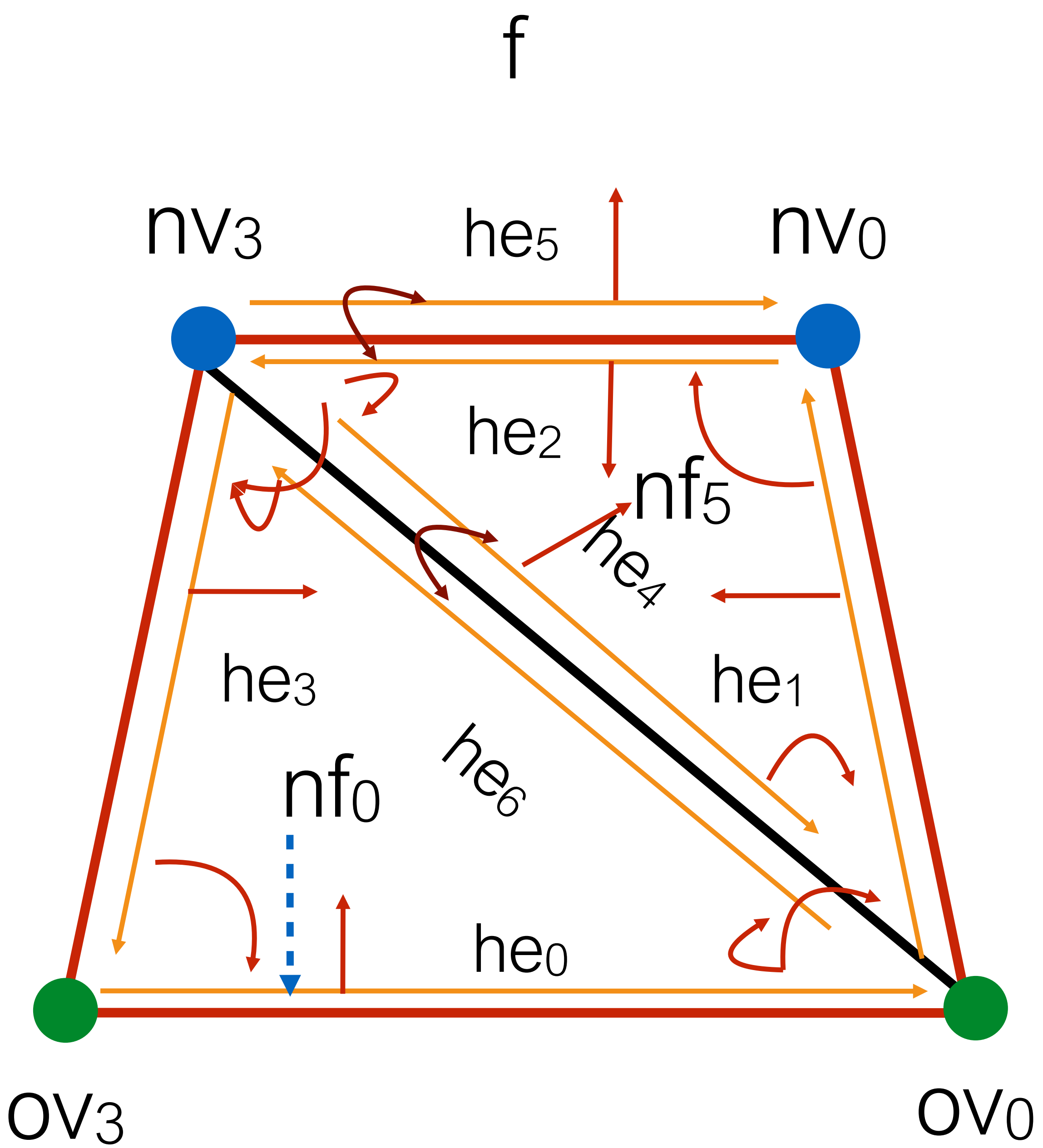
Extrude - topology



Want to connect up the new vertices

nf5 = splitFaceMakeEdge(
f, nv0, nv3);

Extrude - topology



Want to delete old edge

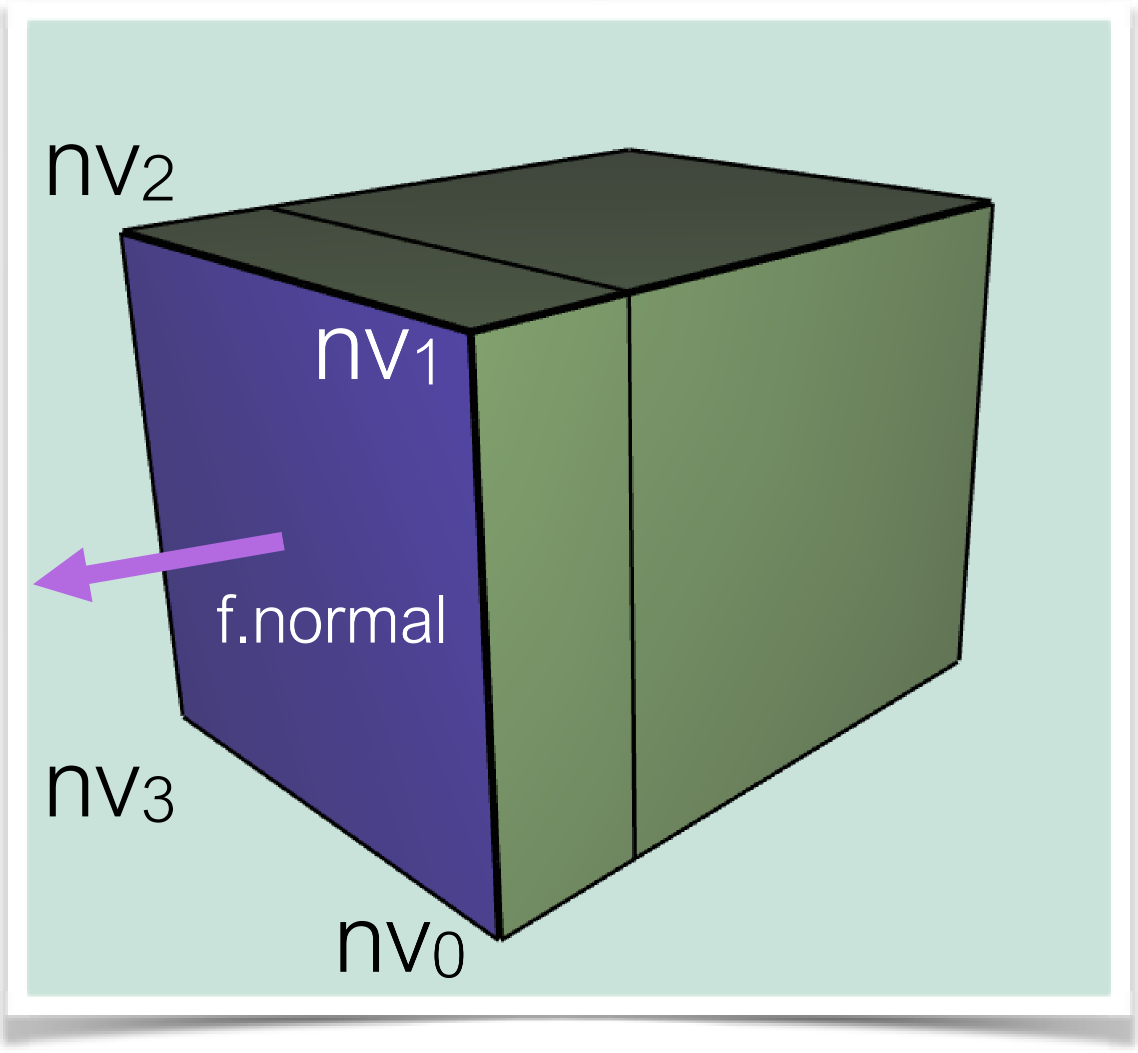
Should be stored before hand

he_4 = old_halfedges[0];

joinFaceKillEdgeSimple(he_6);

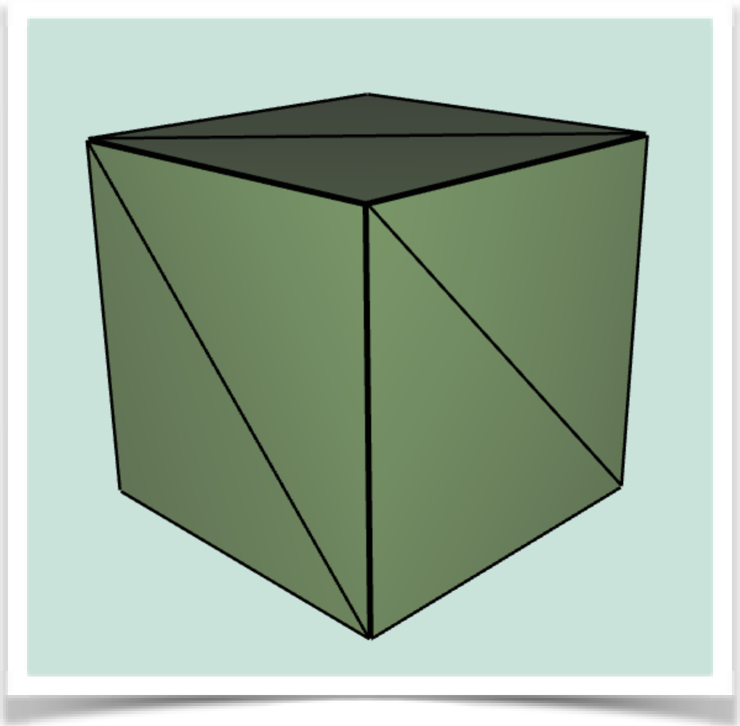
Extrude - geometry

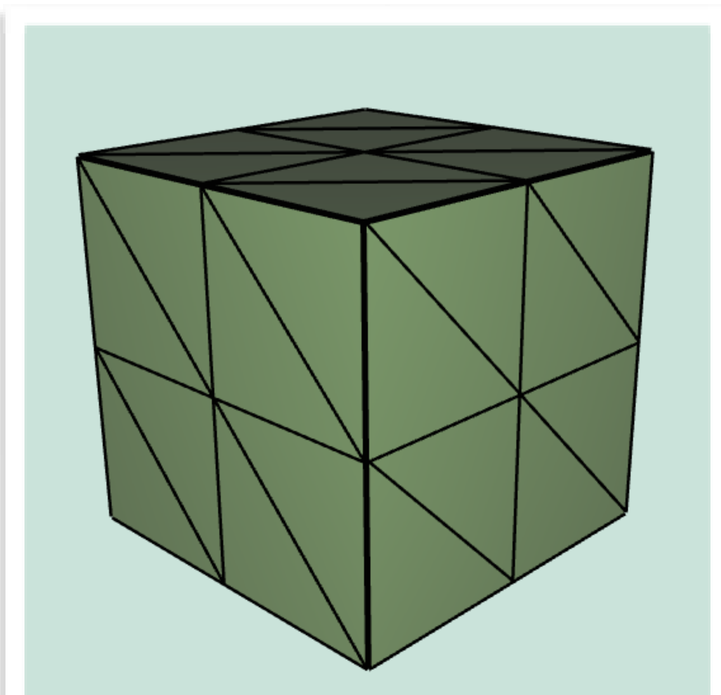
- Actually, very simple
- Move each nv_i by `factor * f.normal`



Triangle Topology

- Each face becomes 4 faces, by splitting all edges in half
- Assumes all triangles!
  - Call your Filters.triangulate();
- Main primitive
  - Face
- How many new vertices?
  - +1 per edge
- How many new faces?
  - +3 per face



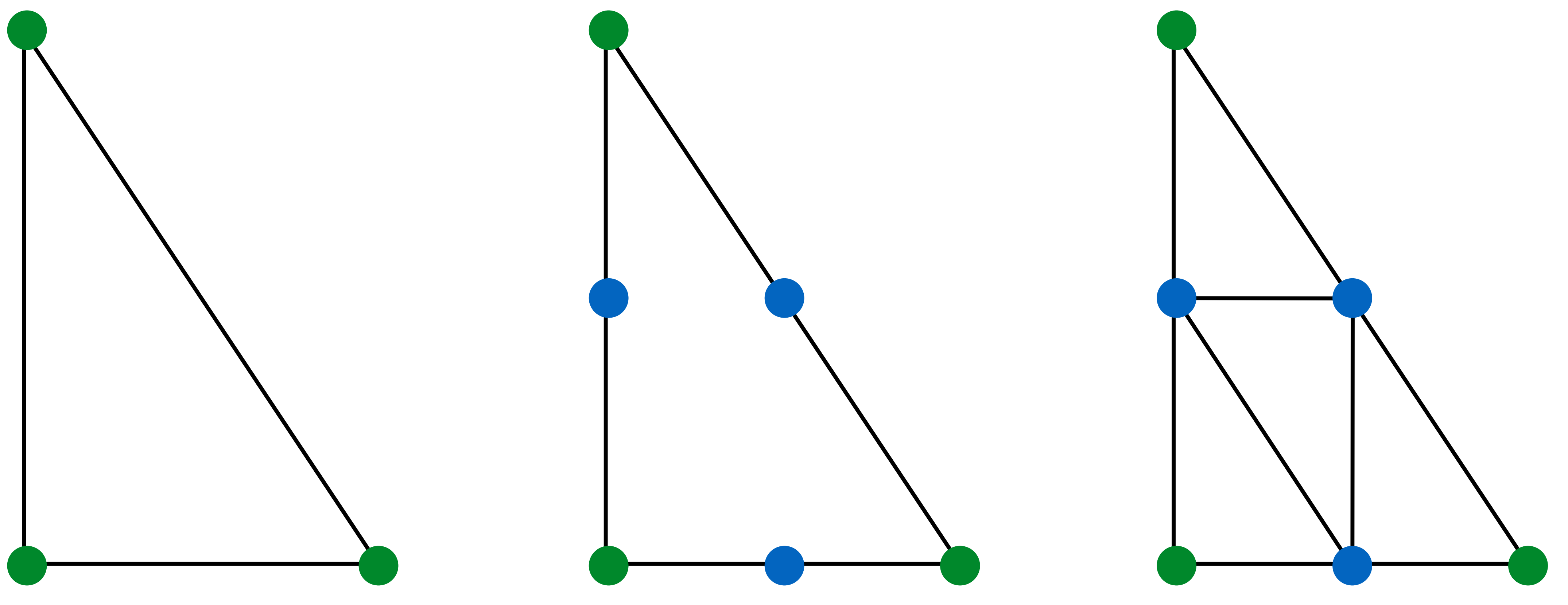


TriTop - topology

- Need to split all edges!
- Create list of half edges
  - Half of them, when splitting halfedge, opposite will also be split
- Join new vertices around a face
  - Determine whether a vertex is old or new by index in vertices array
  - All new will be added to the end of the array!

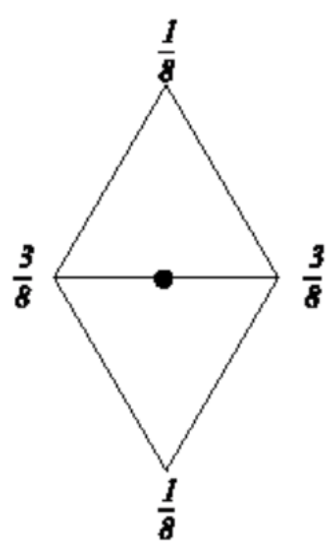
TriTop - topology

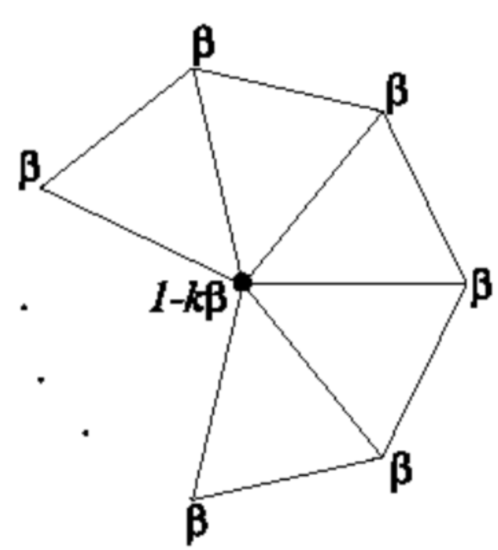
- SplitEdge for each half edge in pre-computed list
- SplitFace per each face, joining new vertices



TriTop - geometry

- None - we're done!
- For Loop Subdivision - store array of new positions for each vertex, where you will write positions calculated according to weight rules
- After done with topology, update positions!





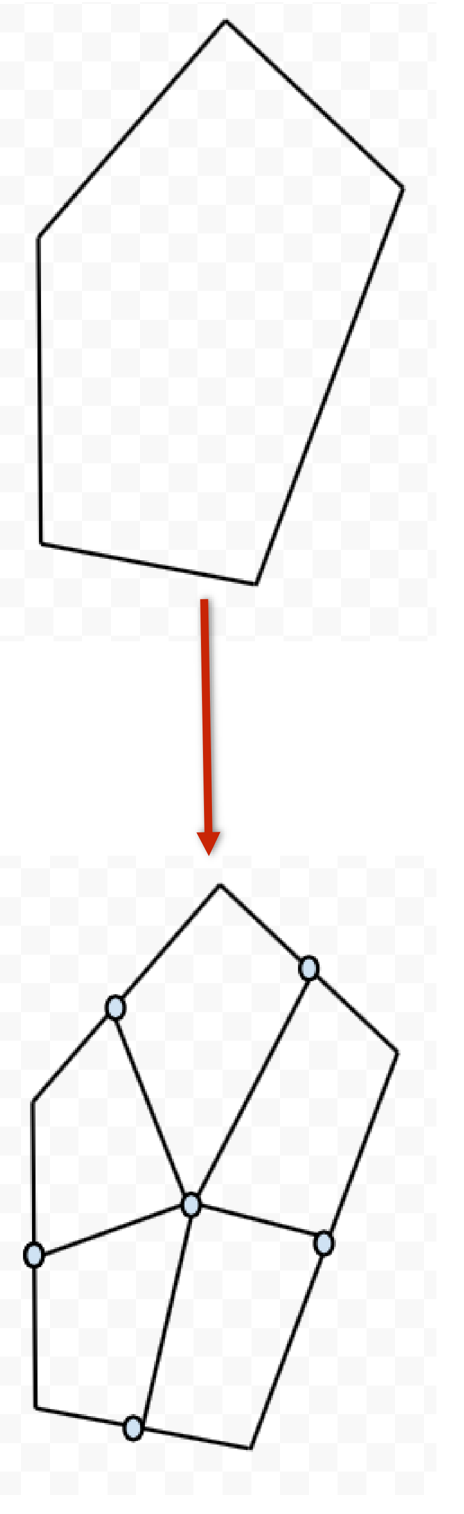
$$\beta = \begin{cases} \frac{3}{8n} & n > 3 \\ \frac{3}{16} & n = 3 \end{cases}$$

Optional features

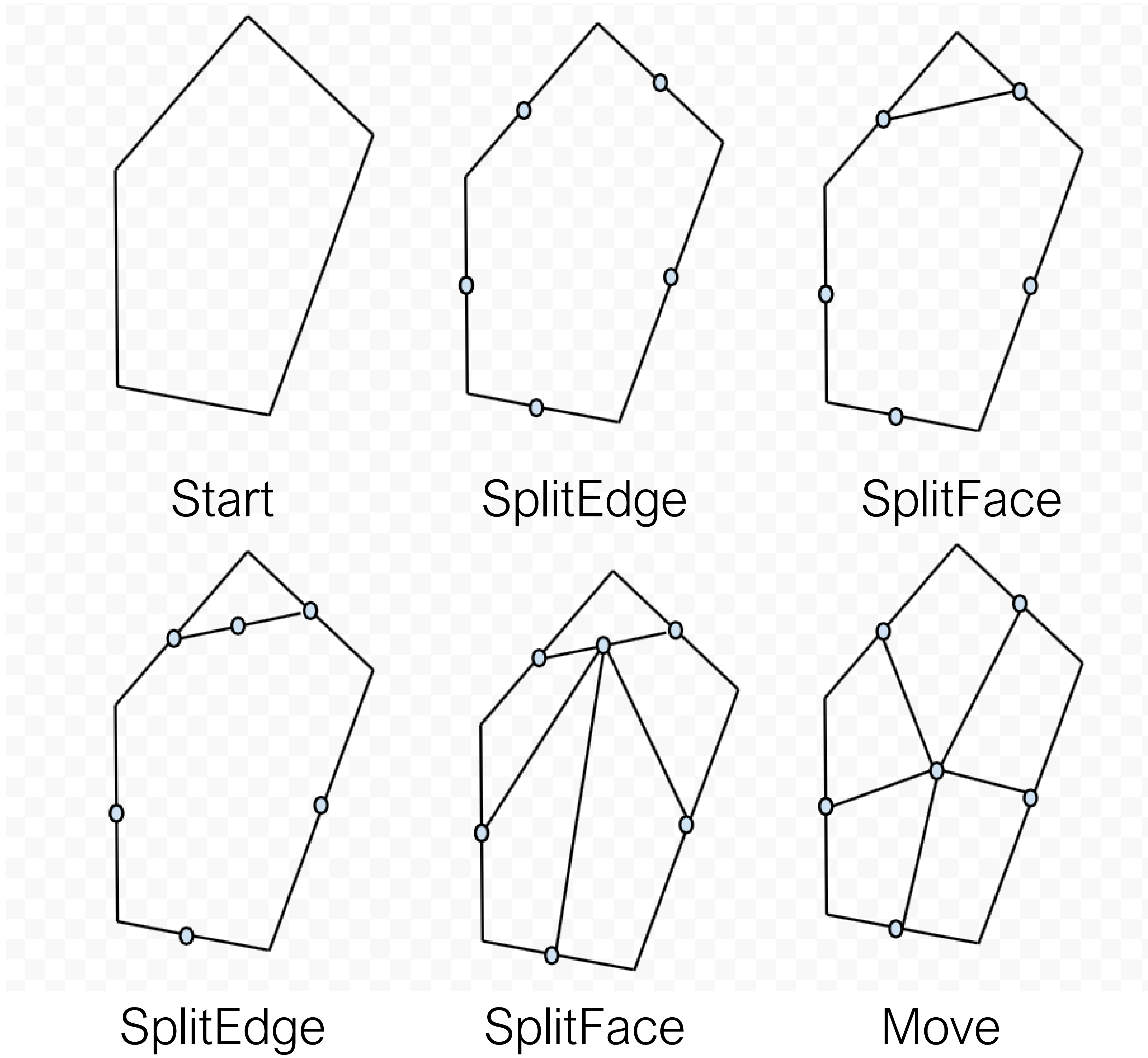
- Quad Subdivision
- Scale-dependent and implicit smoothing
- We will just gloss over those

Quad Topology

- n-gon to quad split
  - Split each edge (SplitEdge)
  - Join 2 new vertices (SplitFace)
  - Split newly create edge (SplitEdge)
  - Join rest of new vertices (SplitFace)
  - Move to interior vertex to centroid location

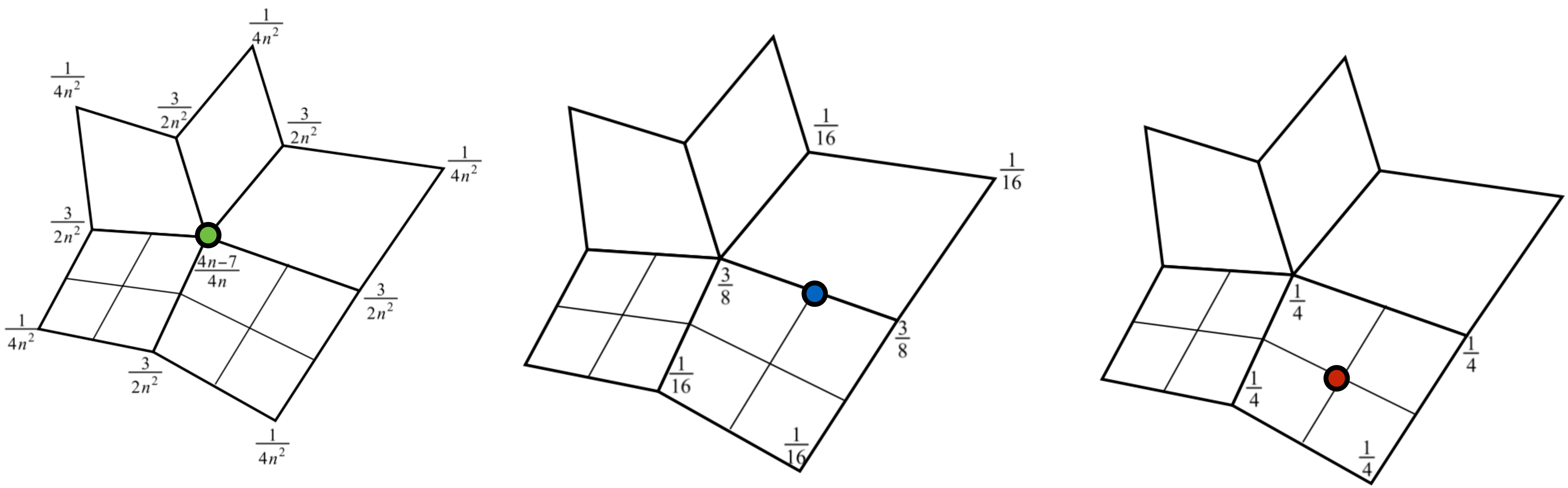


Quad Topology



Quad Subdivision

- Three classes
  - Old vertices
  - Midpoints
  - Centroids

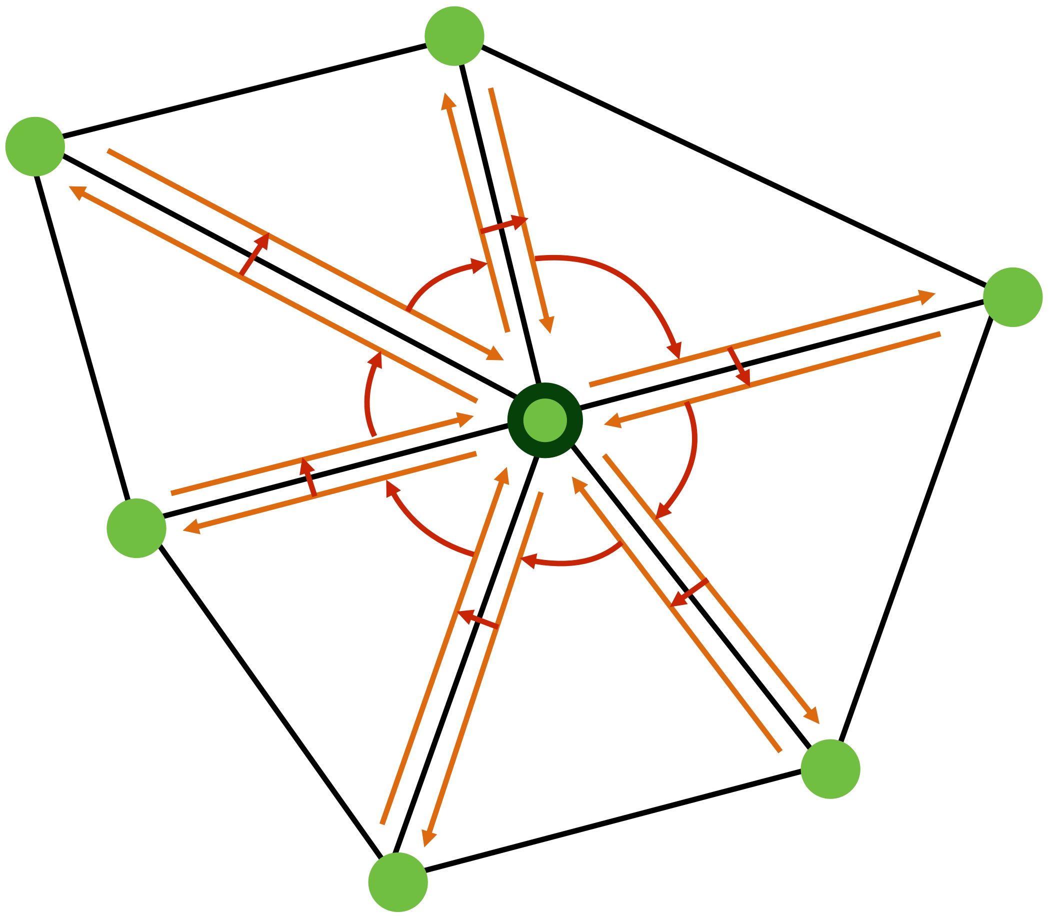


Scott Schaefer

Smoothing

- Repeating uniform Laplacian smoothing
- $L \cdot V = \sum_{v_i \in 1ring} v_i - v \cdot N_{v_{1ring}}$

```
original_he = vertex.he;
he = original_he;
avg_pos.set( 0, 0, 0 );
do {
  avg_pos.add(he.vertex);
  he = he.opposite.next;
} while ( he != original_he)
avg_pos.add(-vertex*num_neigh);
new_pos = vertex + avg_pos*delta;
```



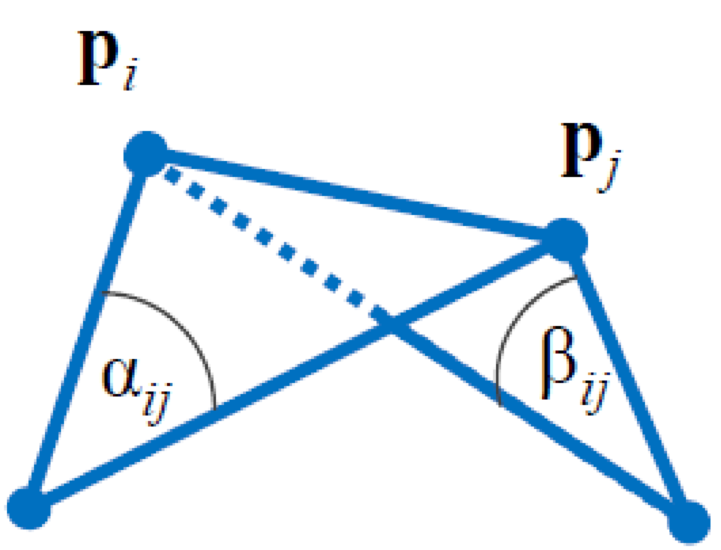
Smoothing

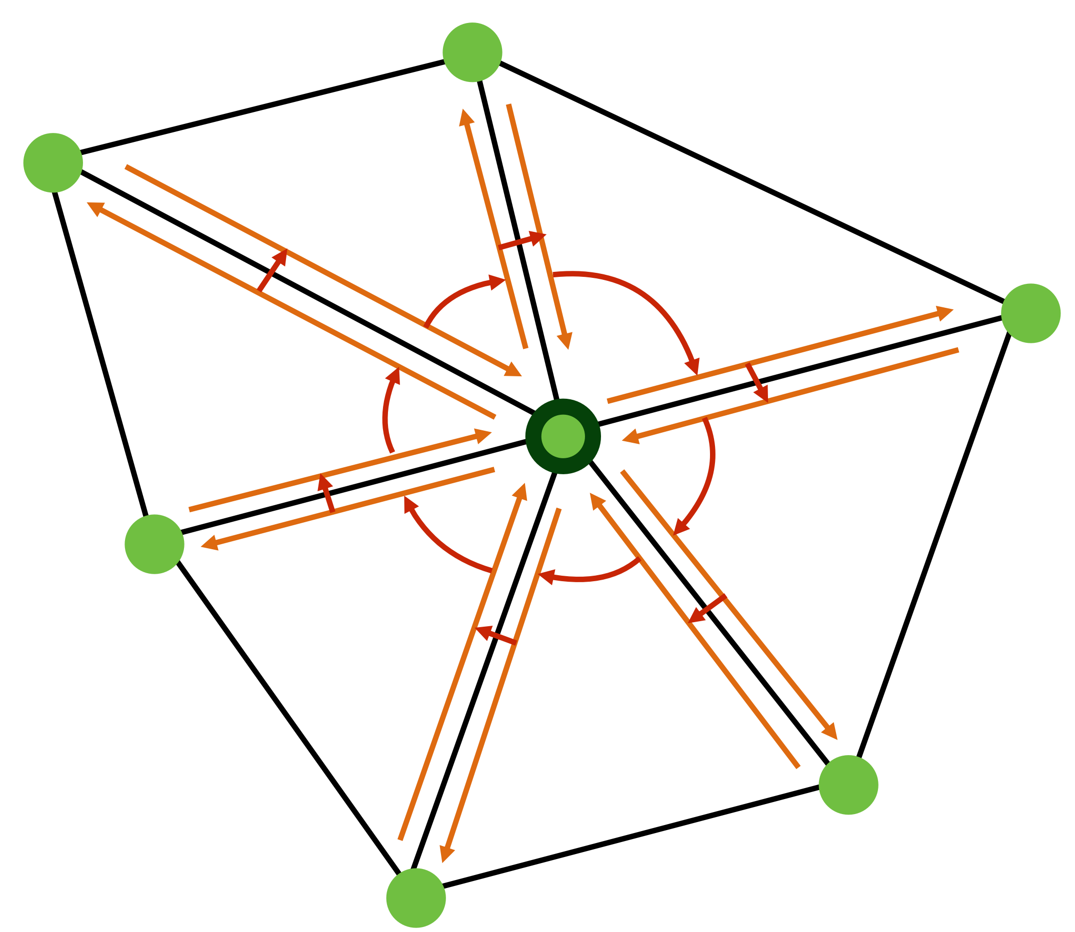
- Cotan Laplacian smoothing
- $L \cdot V = \sum_{v_i \in 1ring} w_i \cdot v_i - v \cdot \sum_{v_i \in 1ring} w_i$

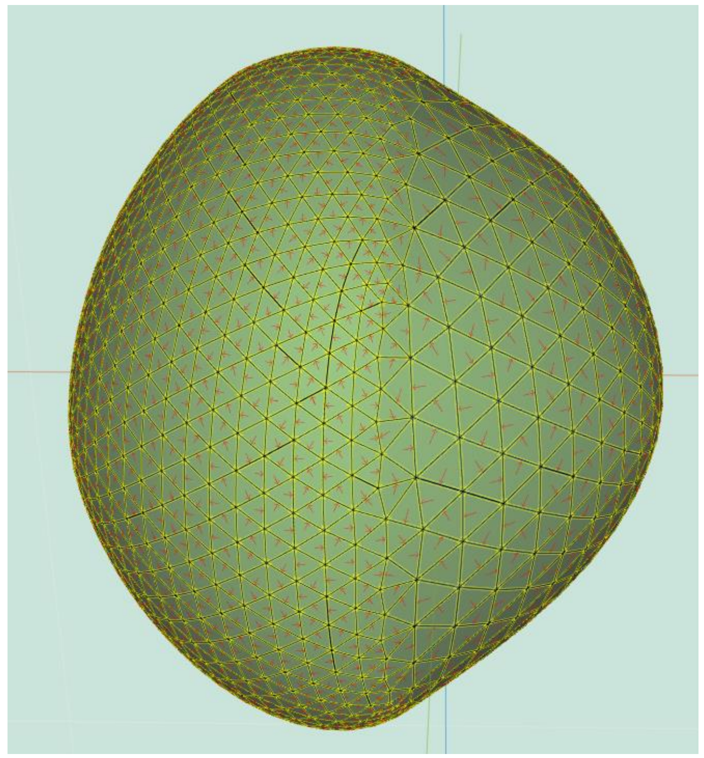
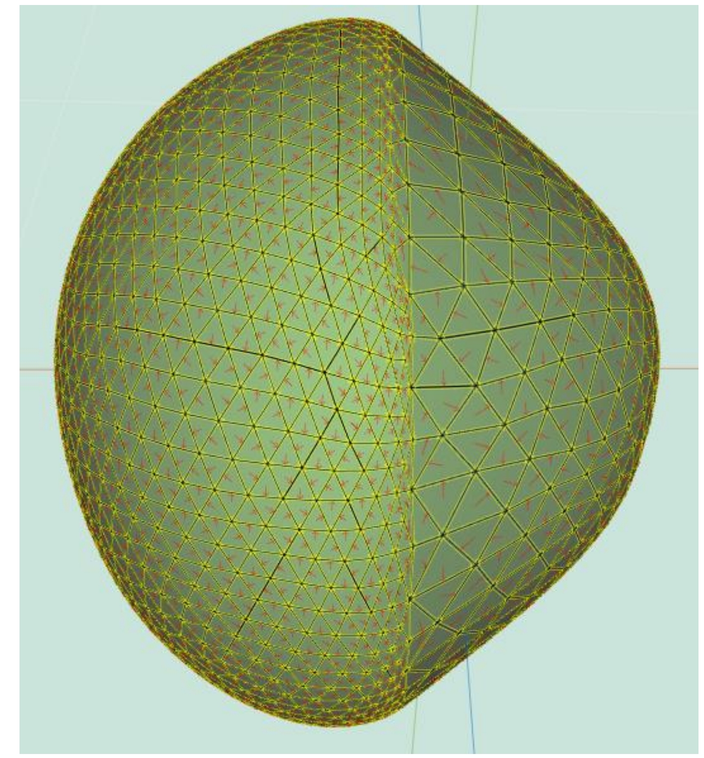
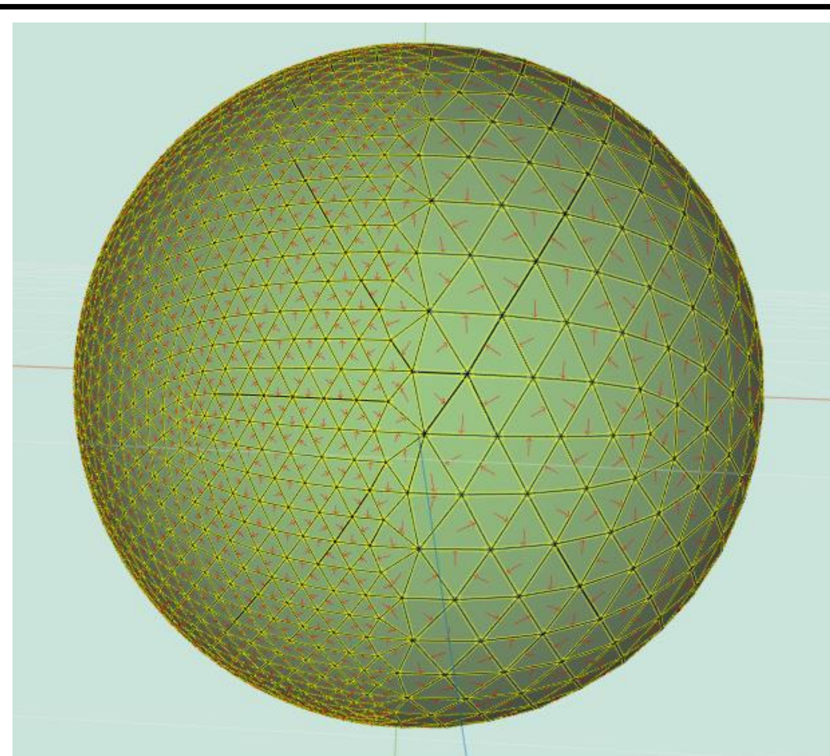
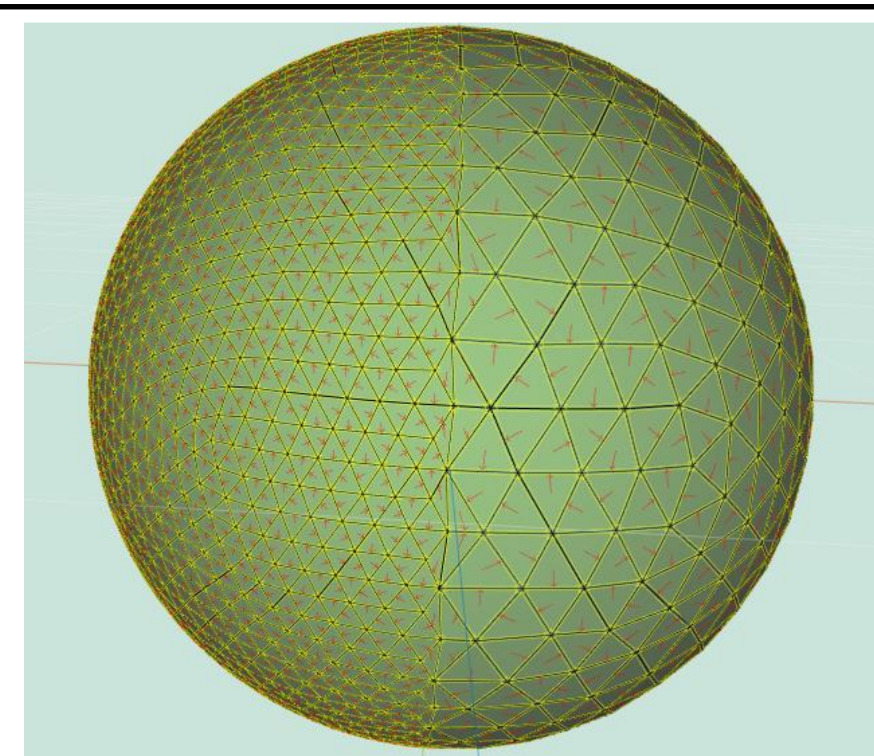
avg_pos.add(he.vertex); ⟶ avg_pos.add(w*he.vertex);

num_neigh ⟶ total_w

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





	Uniform	Curvature-flow
Not scale dependent		
Scale dependent		

Smoothing

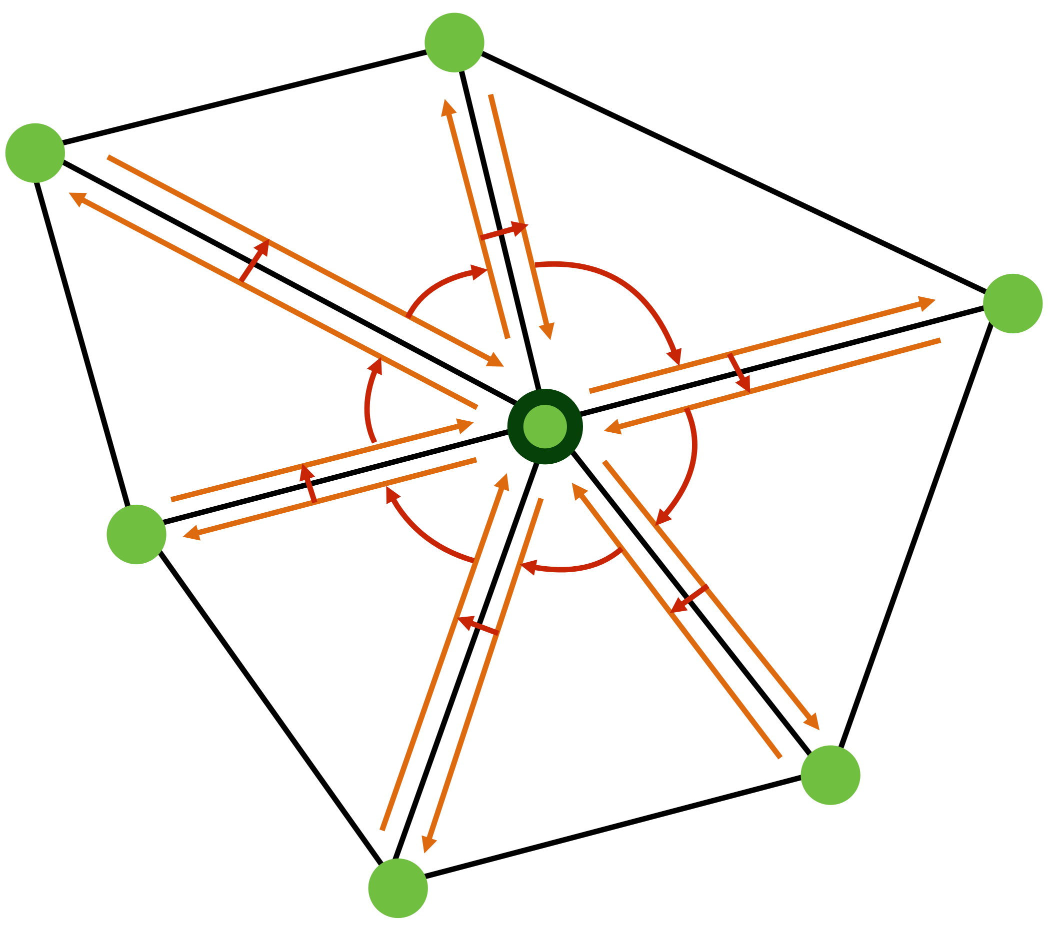
- Scale-dependent smoothing

$$v_{new} = v_{old} + (L \cdot v_{old}) \cdot \delta \longrightarrow v_{new} = v_{old} + (L \cdot v_{old}) \cdot \delta \cdot \frac{A}{A_v}$$

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

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

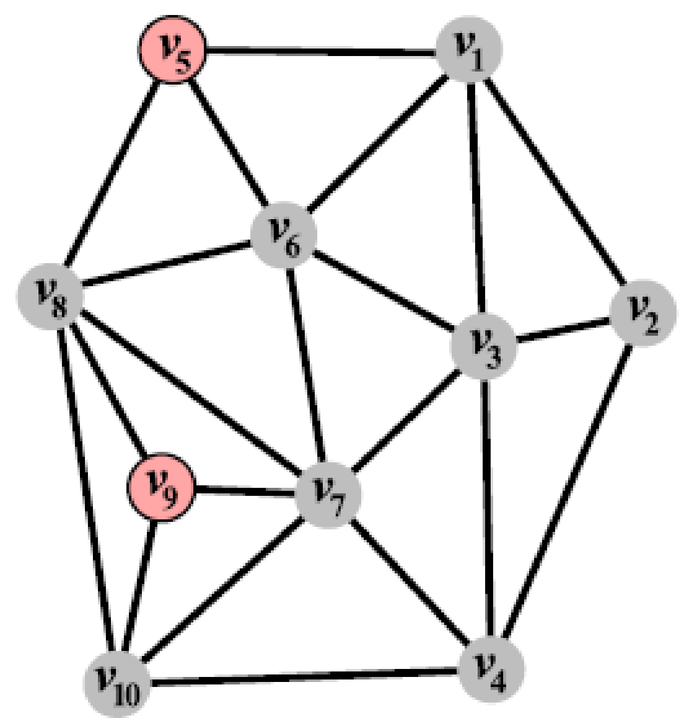
$$A = \frac{3}{N_v} \cdot \sum_{f_i \in F} area(f_i)$$



Smoothing

- Implicit smoothing
  - Matricial form

$$L_{ij} = \begin{cases} -w_{ij} & i \neq j \\ \Sigma_{j \in 1_{ring_i}} w_{ij} & i = j \\ 0 & else \end{cases}$$



4	−1	−1		−1	−1				
−1	3	−1	−1						
−1	−1	5	−1		−1	−1			
	−1	−1	4			−1			−1
−1				3	−1		−1		
−1		−1		−1	5	−1	−1		
		−1	−1		−1	6	−1	−1	−1
				−1	−1	−1	5	−1	−1
						−1	−1	3	−1
			−1			−1	−1	−1	4

  - w_{ij} can be uniform or cotan
  - Scale dependency: diagonal matrix M of the “mass” ($\frac{A}{A_v}$)

$$L_{scale\ dependent} = M \cdot L$$

Smoothing

- $v_{new} = v_{old} + (L \cdot v_{old}) \cdot \delta \longrightarrow v_{old} = v_{new} - (L \cdot v_{new}) \cdot \delta$

$$v_{new} = (I - L \cdot \delta)^{-1} \cdot v_{old}$$

```
matLDecomp = math.lup(matL);
resX = math.lusolve(matLDecomp,allXs);
resY = math.lusolve(matLDecomp,allYs);
resZ = math.lusolve(matLDecomp,allZs);
```

- You would probably want to use matrix.subset and math.range