COS 426 : Precept 5

Working with Half-Edge

Agenda

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

How do I start?

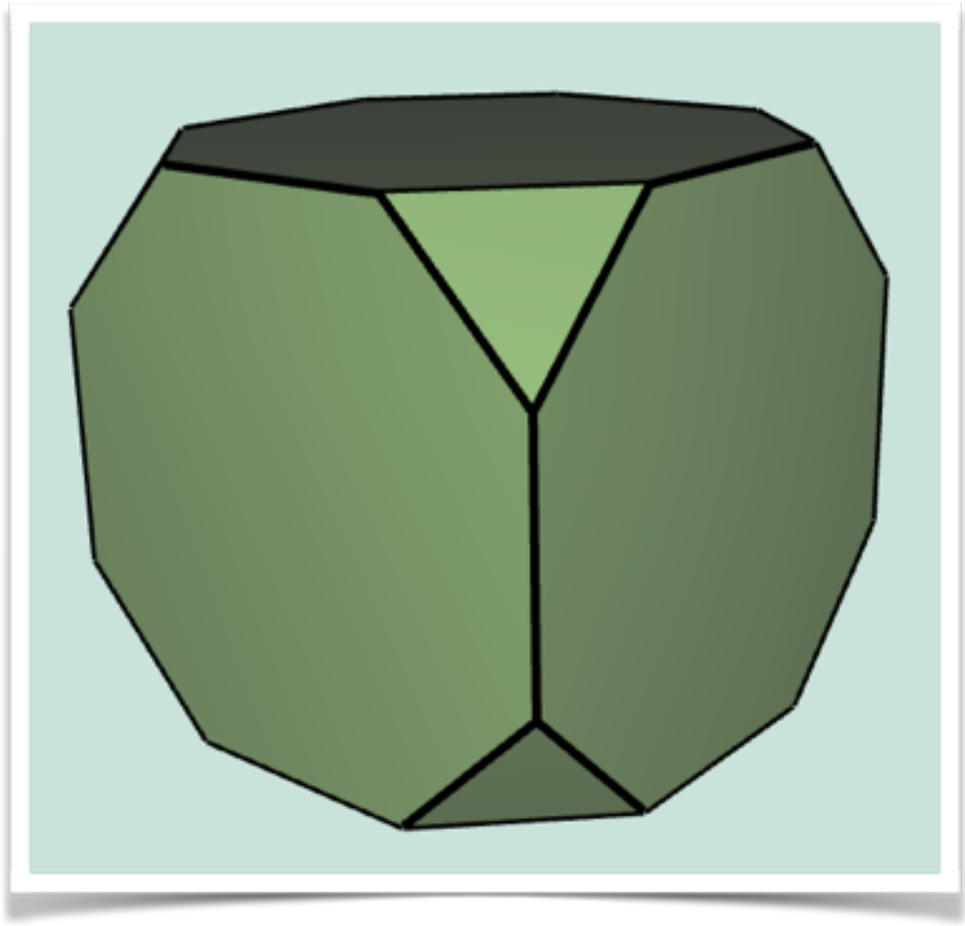
- Some of the operations are tricky to implement!
- Think locally - independence of operations
  - Modifying a vertex/edge/face should not influence other primitives
- Start small
  - Just work on one primitive at a time
- Decouple topology and geometry
  - What are necessary topological changes?
  -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

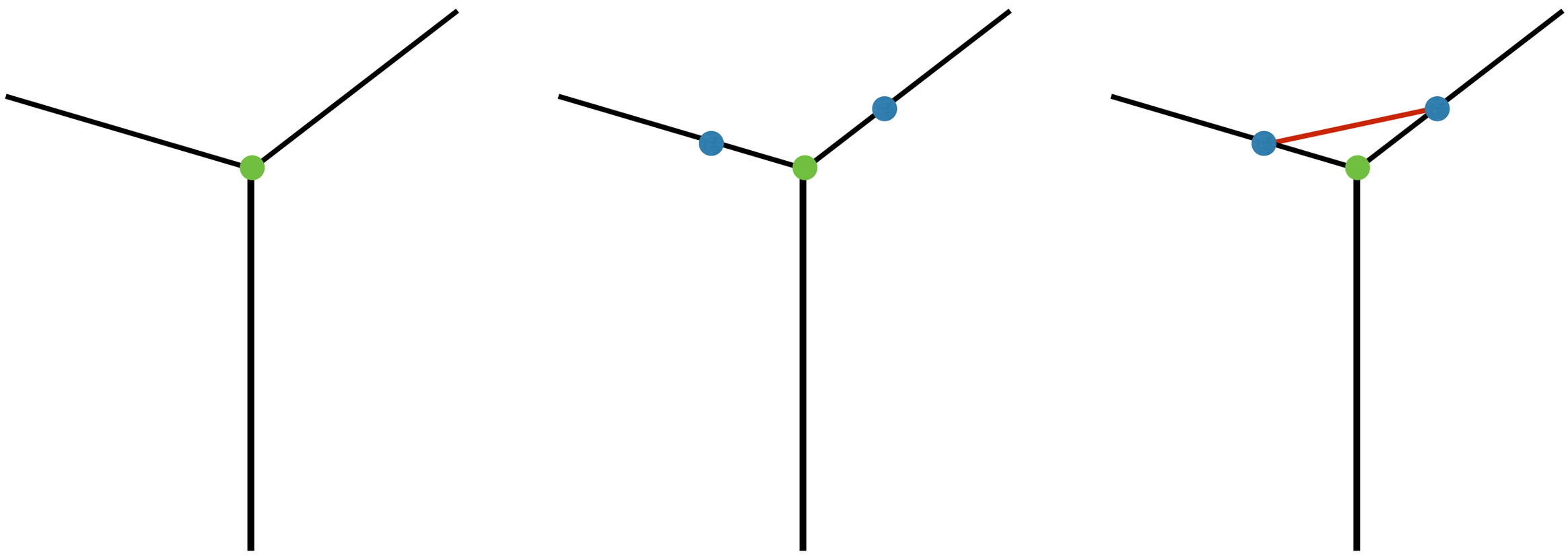
Truncate

- 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



Start | 2 x SplitEdge | Split 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!

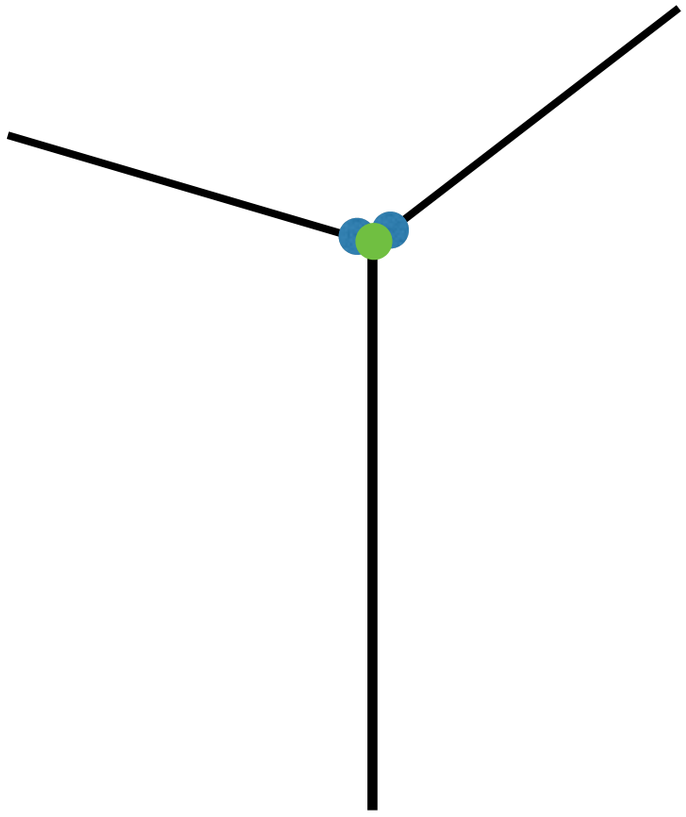
Start

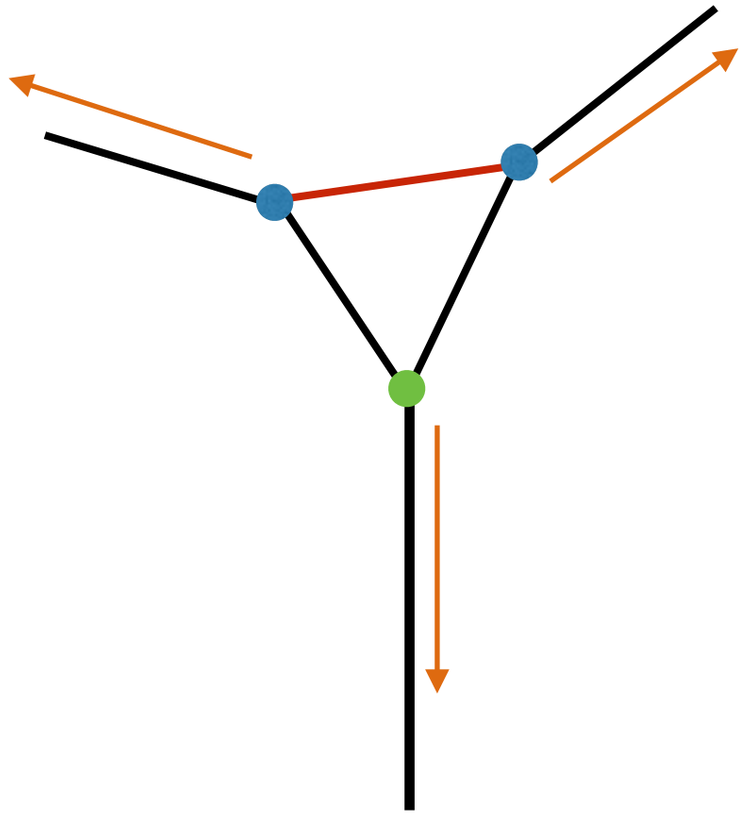
2 x SplitEdge

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

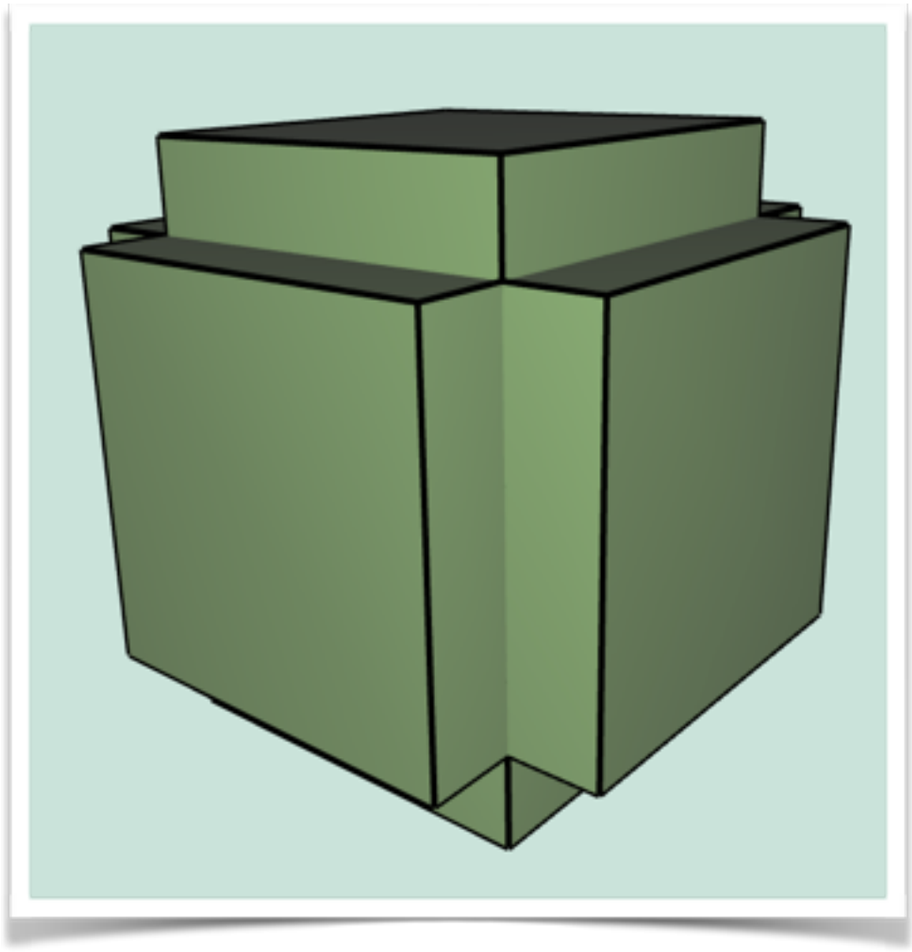
    - Face

- How many new vertices?

    - +n per n-gon

- How many new faces?

    - +n per n-gon



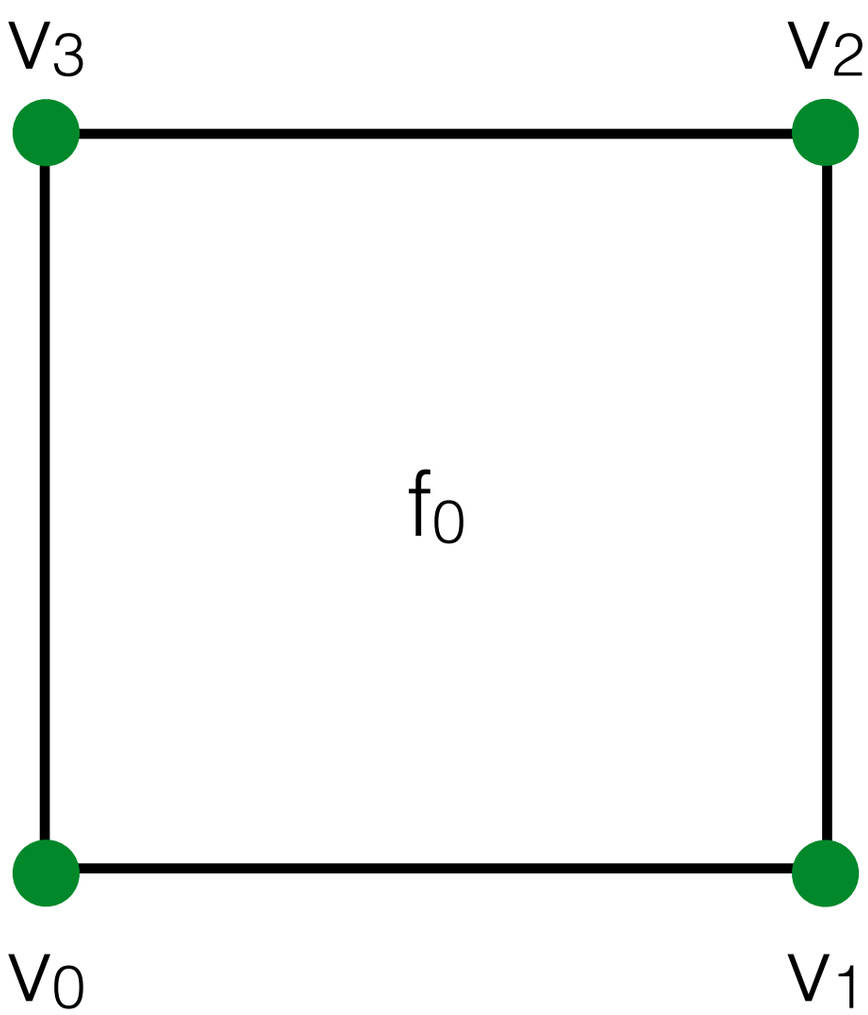
Extrude - topology

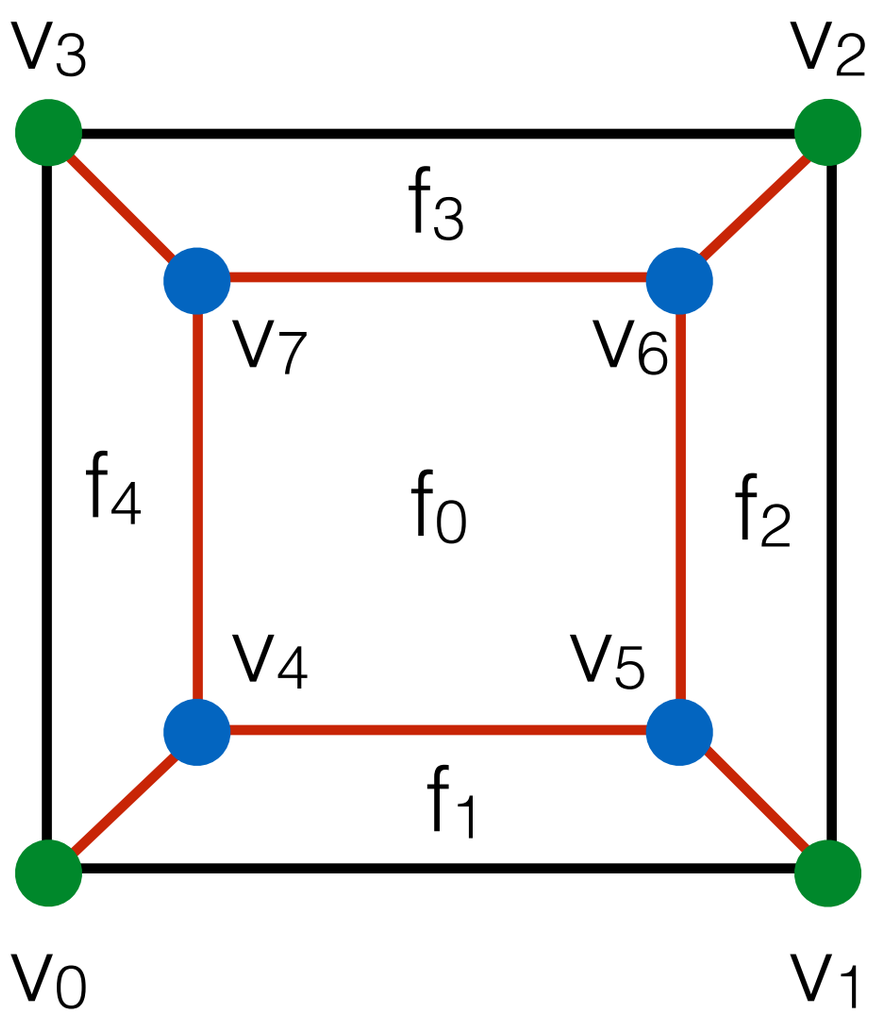
- Again, following figures are for illustration only, new vertices should be added at a location of the old ones!

Extrude - topology

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

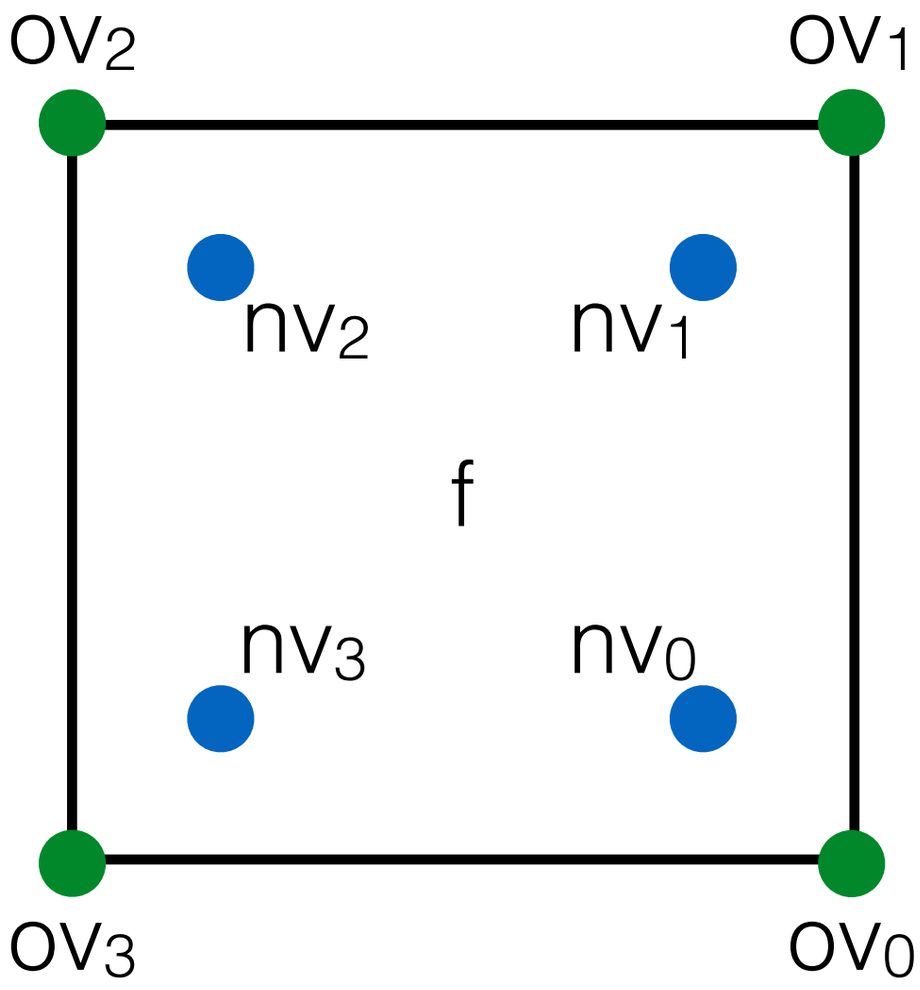
- Desired:





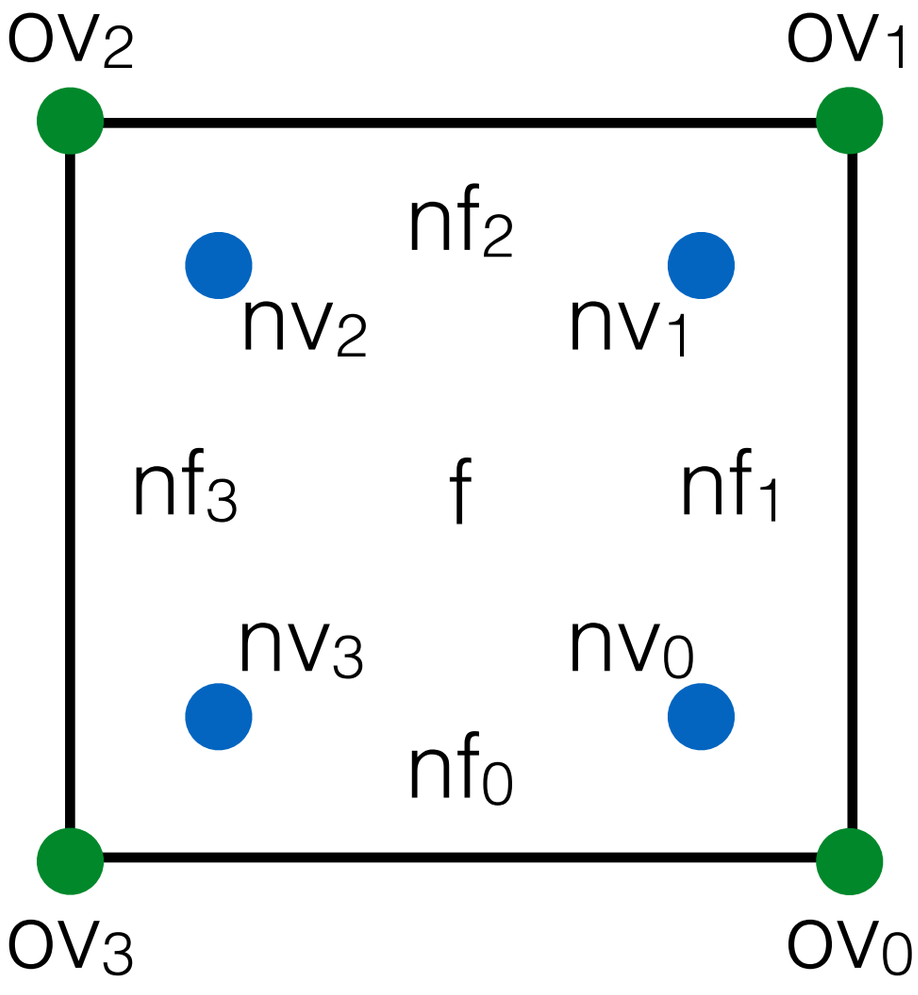
Extrude - topology

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



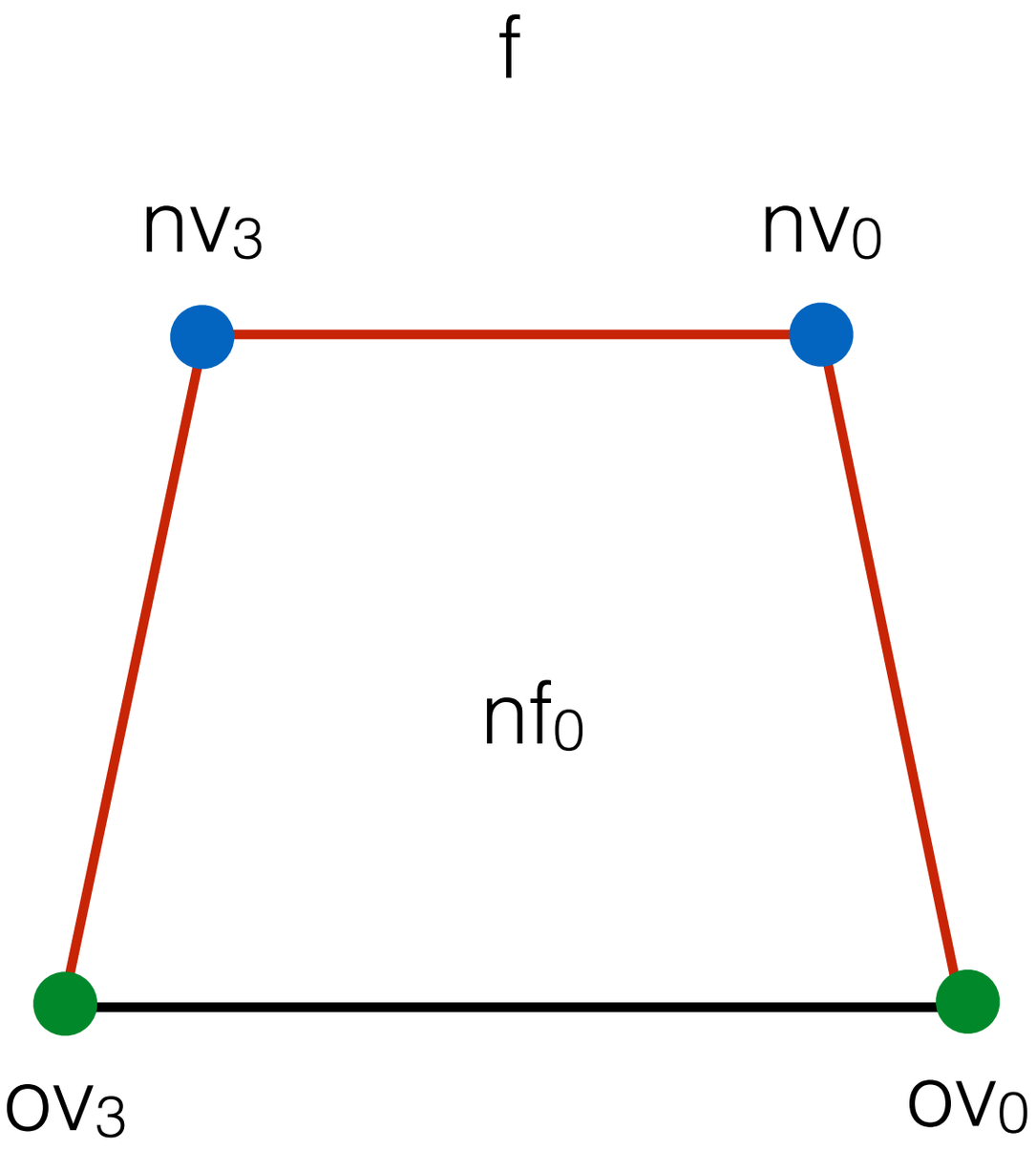
nv_i = addVertex(ov_i.position);

Extrude - topology

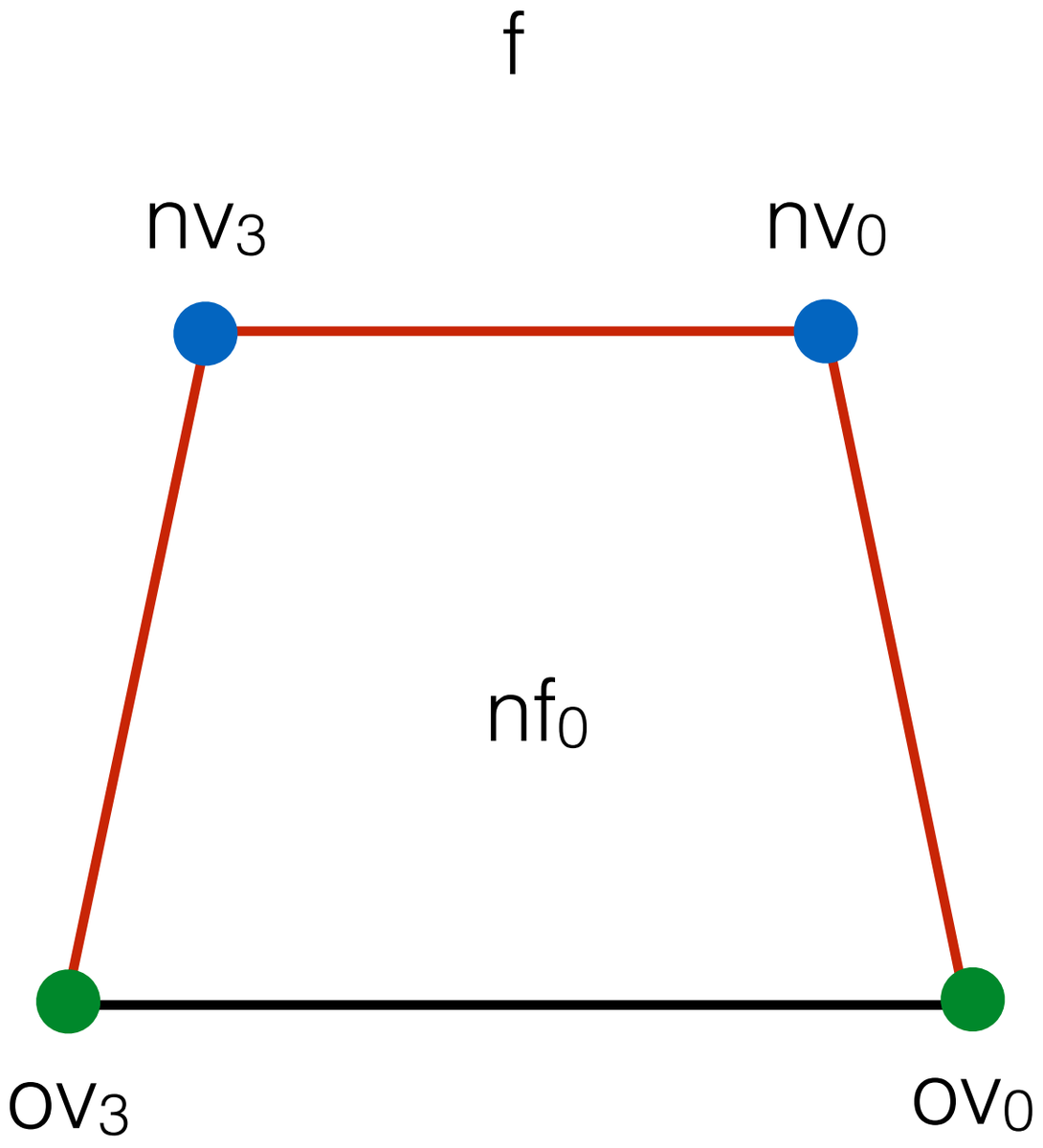


nf_i = addFace();

Extrude - topology



Extrude - topology

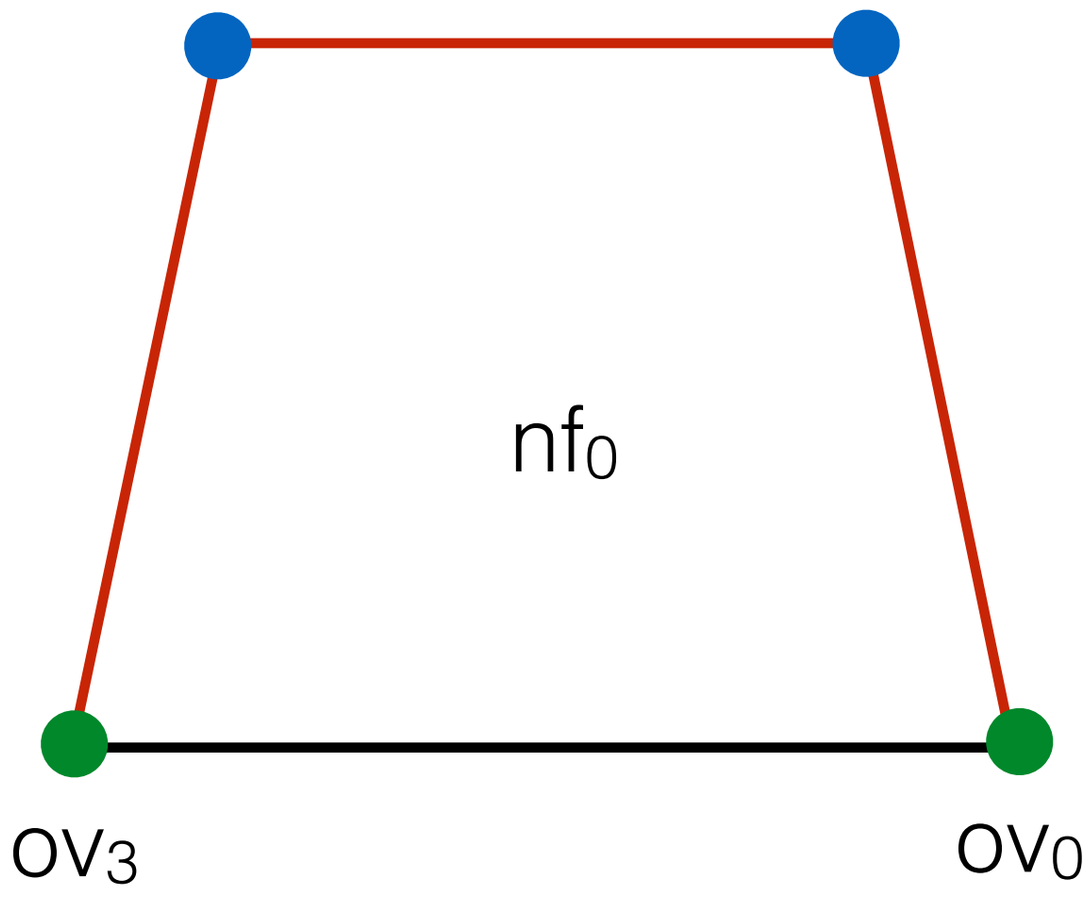


he_0 = old_halfedges[0];

Extrude - topology

Should be stored before hand, such that old_halfedges[0] points at ov_0

f

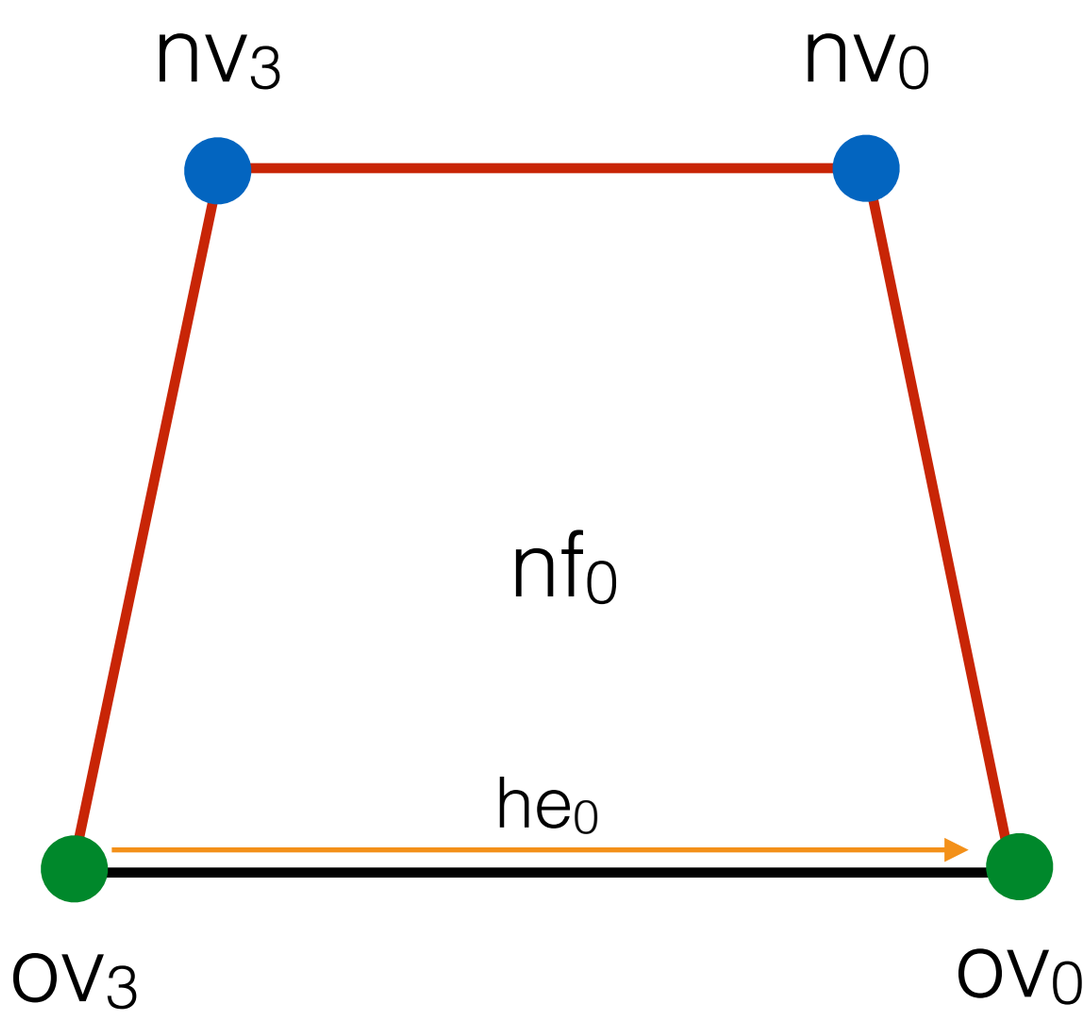


he_0 = old_halfedges[0];

Extrude - topology

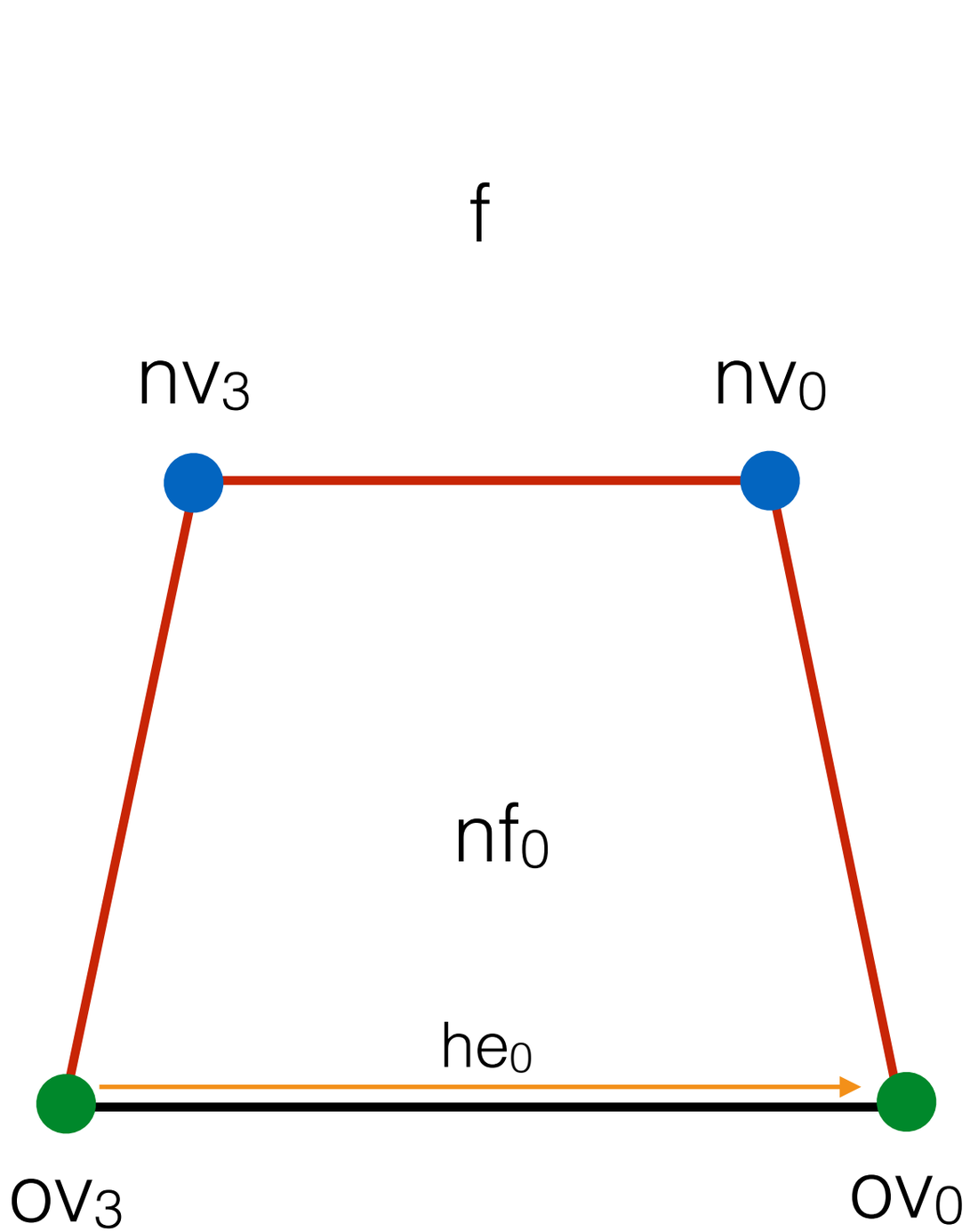
Should be stored before hand, such that old_halfedges[0] points at ov_0

he_0 = old_halfedges[0];



Extrude - topology

Should be stored before hand, such that old_halfedges[0] points at ov_0

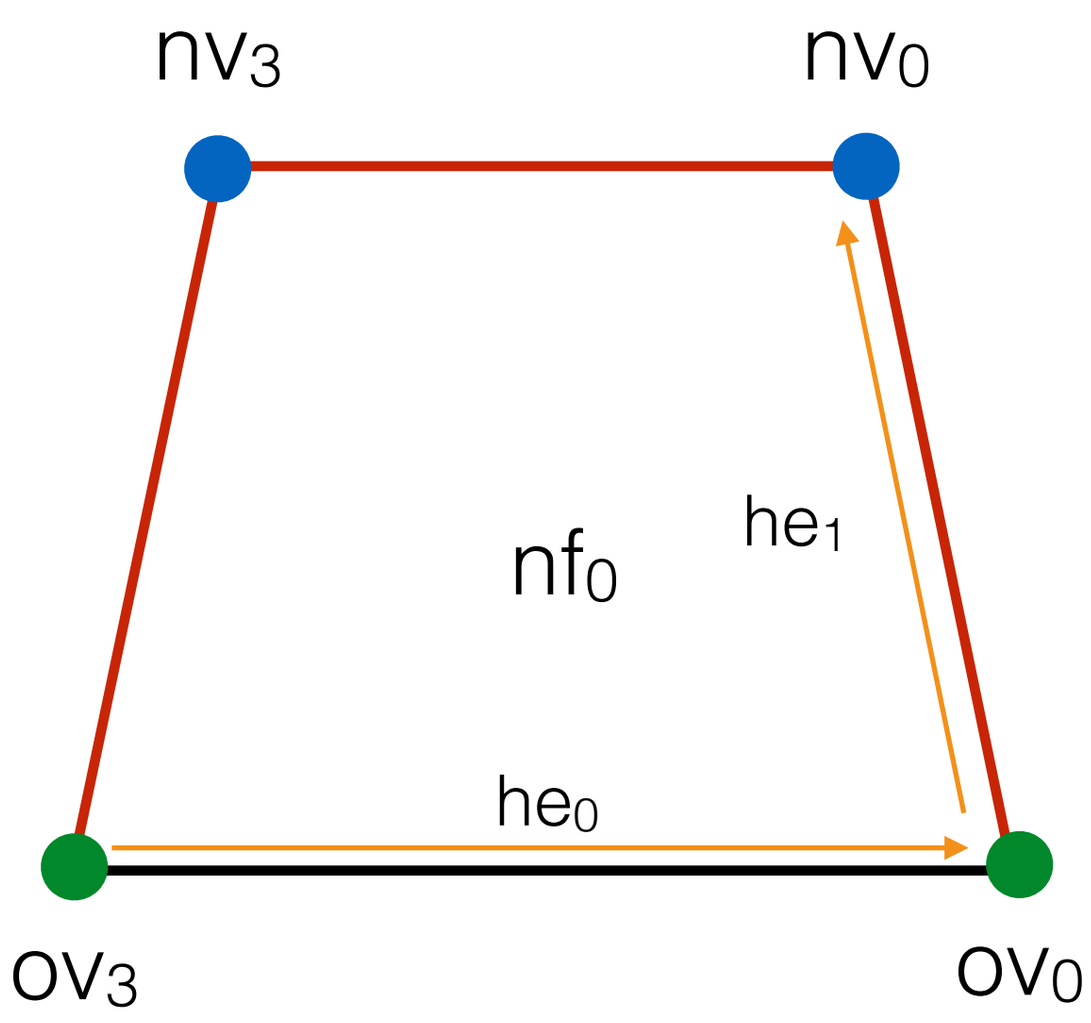


he_0 = old_halfedges[0];

Add half edges in counter clockwise order

Extrude - topology

Should be stored before hand, such that old_halfedges[0] points at ov_0

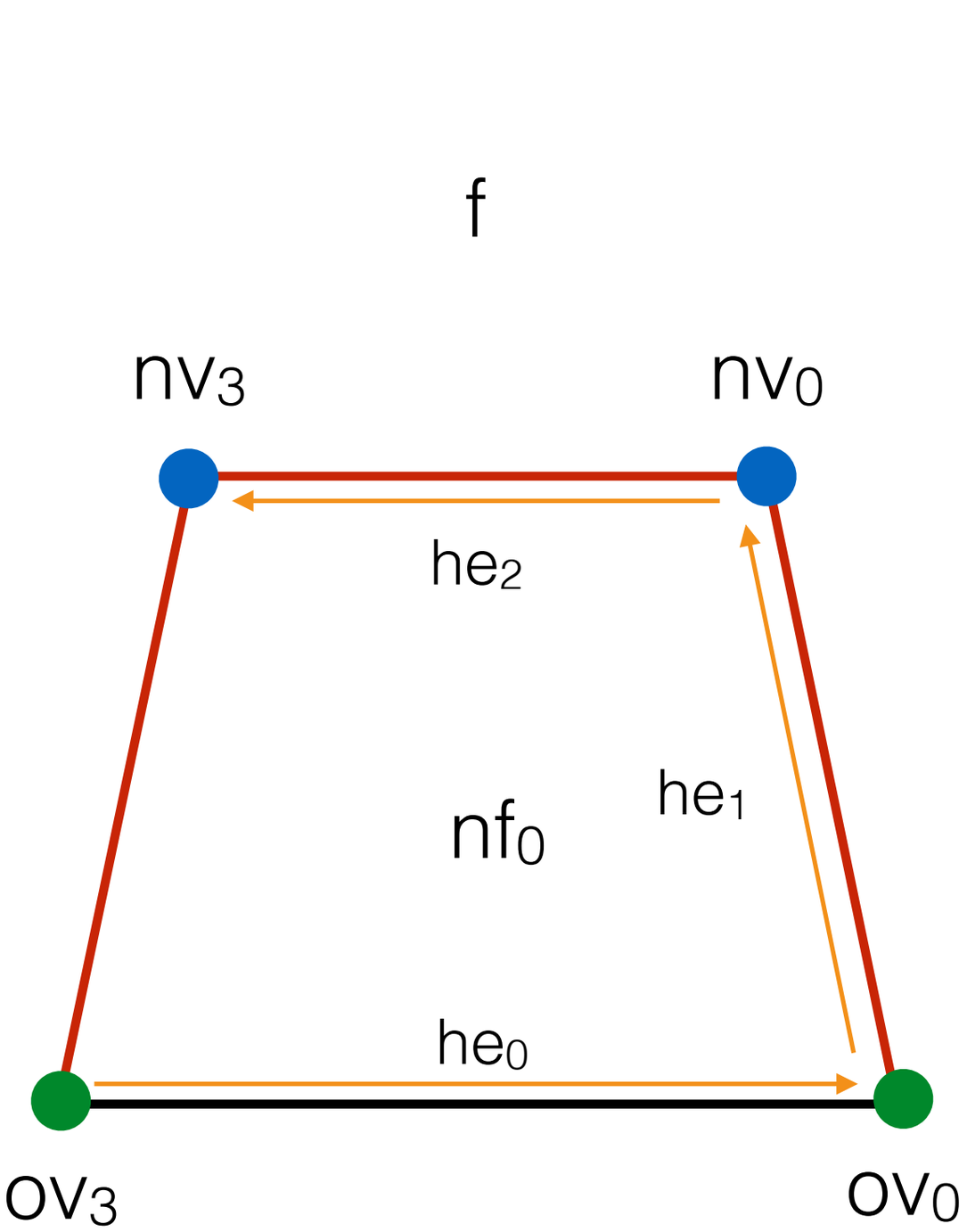


he_0 = old_halfedges[0];

Add half edges in counter clockwise order

Extrude - topology

Should be stored before hand, such that old_halfedges[0] points at ov_0

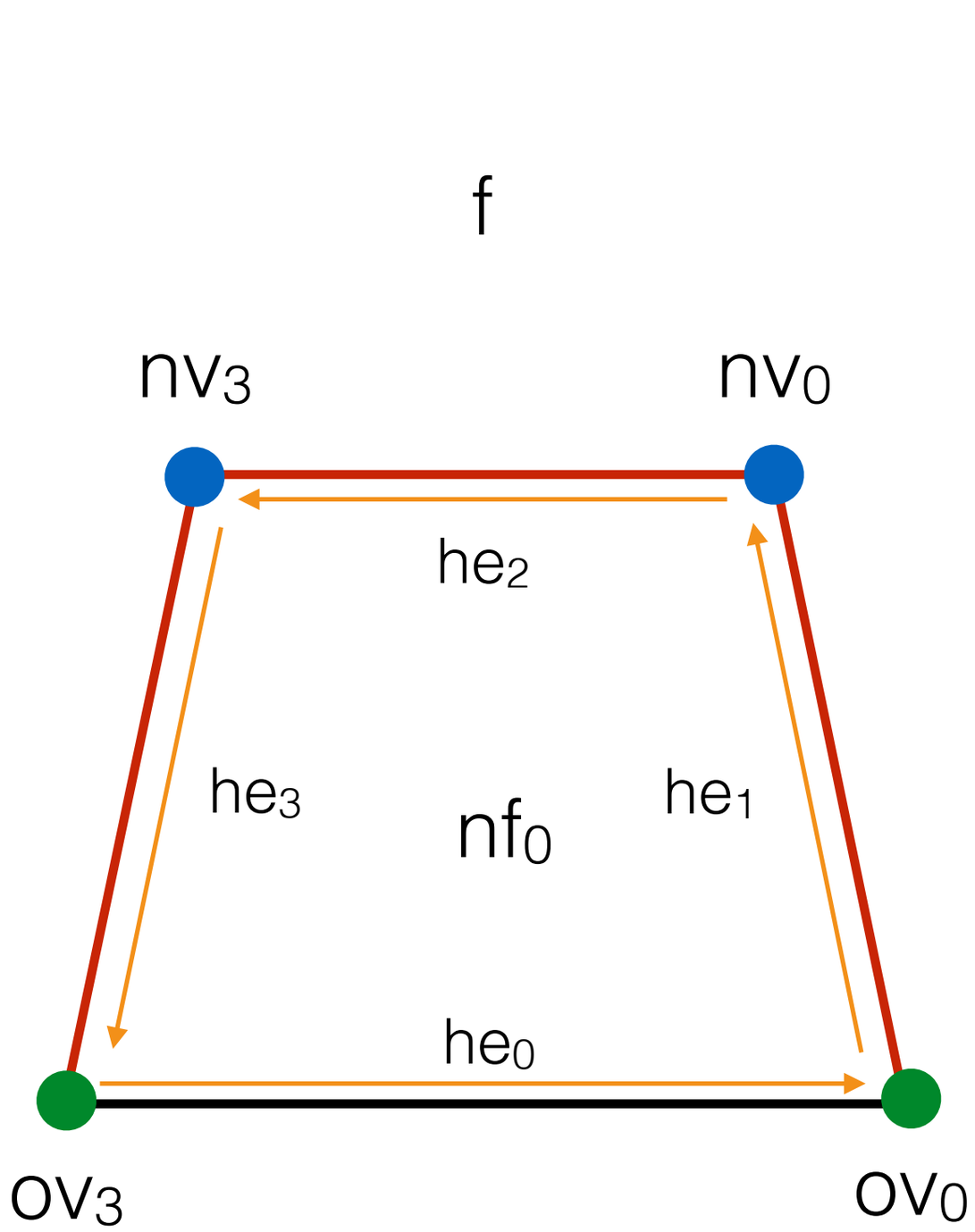


he_0 = old_halfedges[0];

Add half edges in counter clockwise order

Extrude - topology

Should be stored before hand, such that old_halfedges[0] points at ov_0

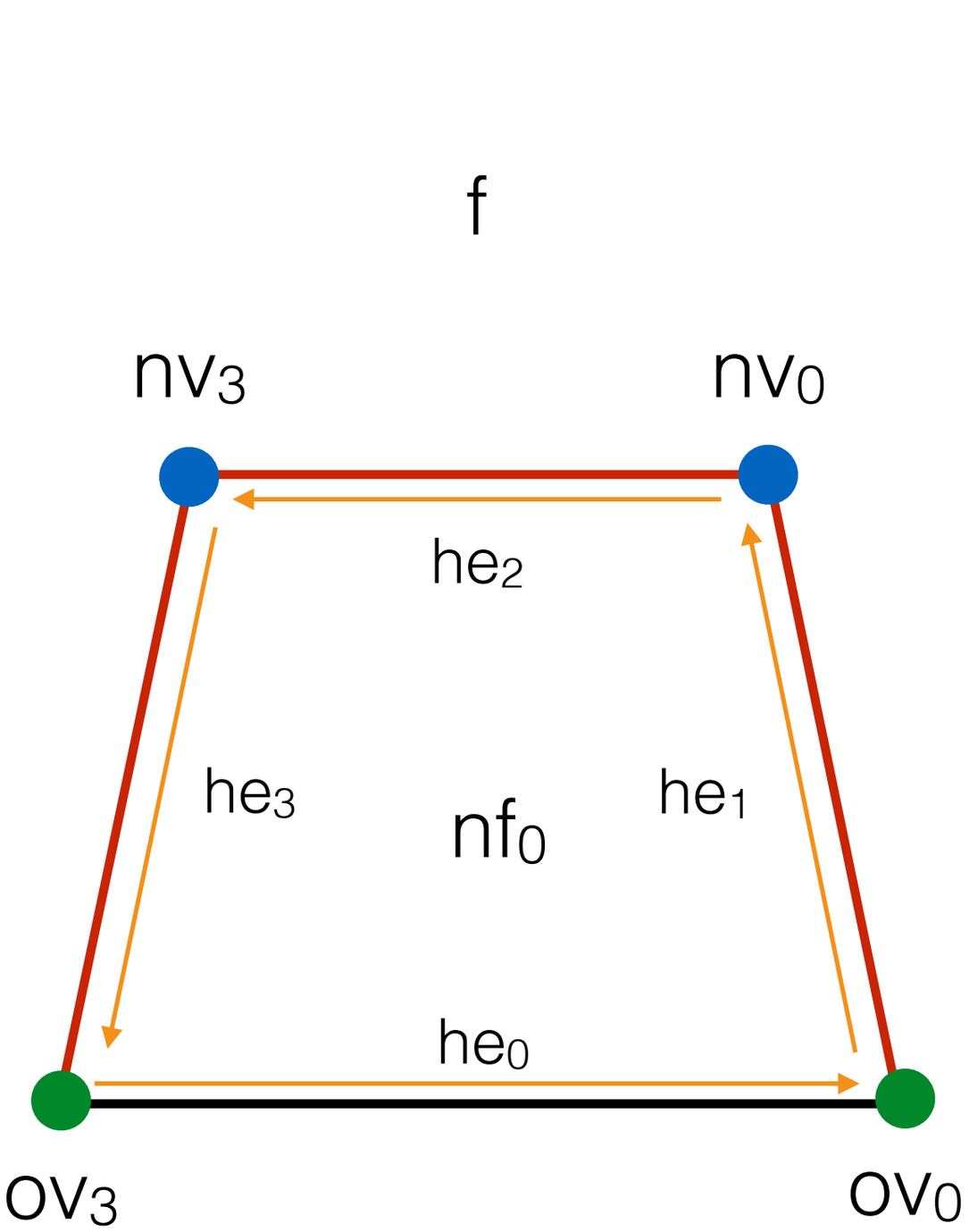


he_0 = old_halfedges[0];

Add half edges in counter clockwise order

Extrude - topology

Should be stored before hand, such that old_halfedges[0] points at ov_0

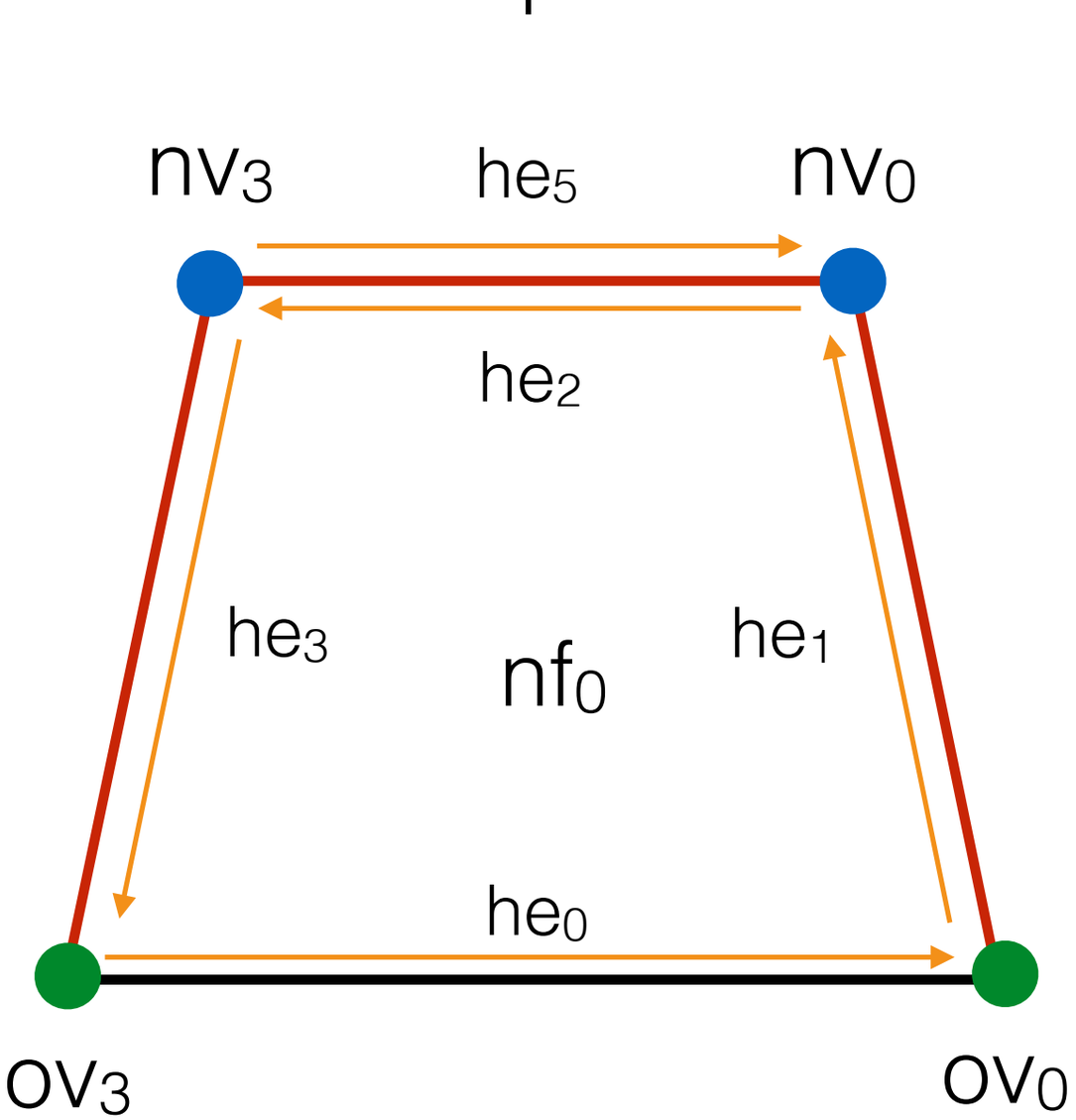


he_0 = old_halfedges[0];

Add half edges in counter clockwise order

Add half edge opposite to he_2

Extrude - topology



Should be stored before hand, such that old_halfedges[0] points at ov_0

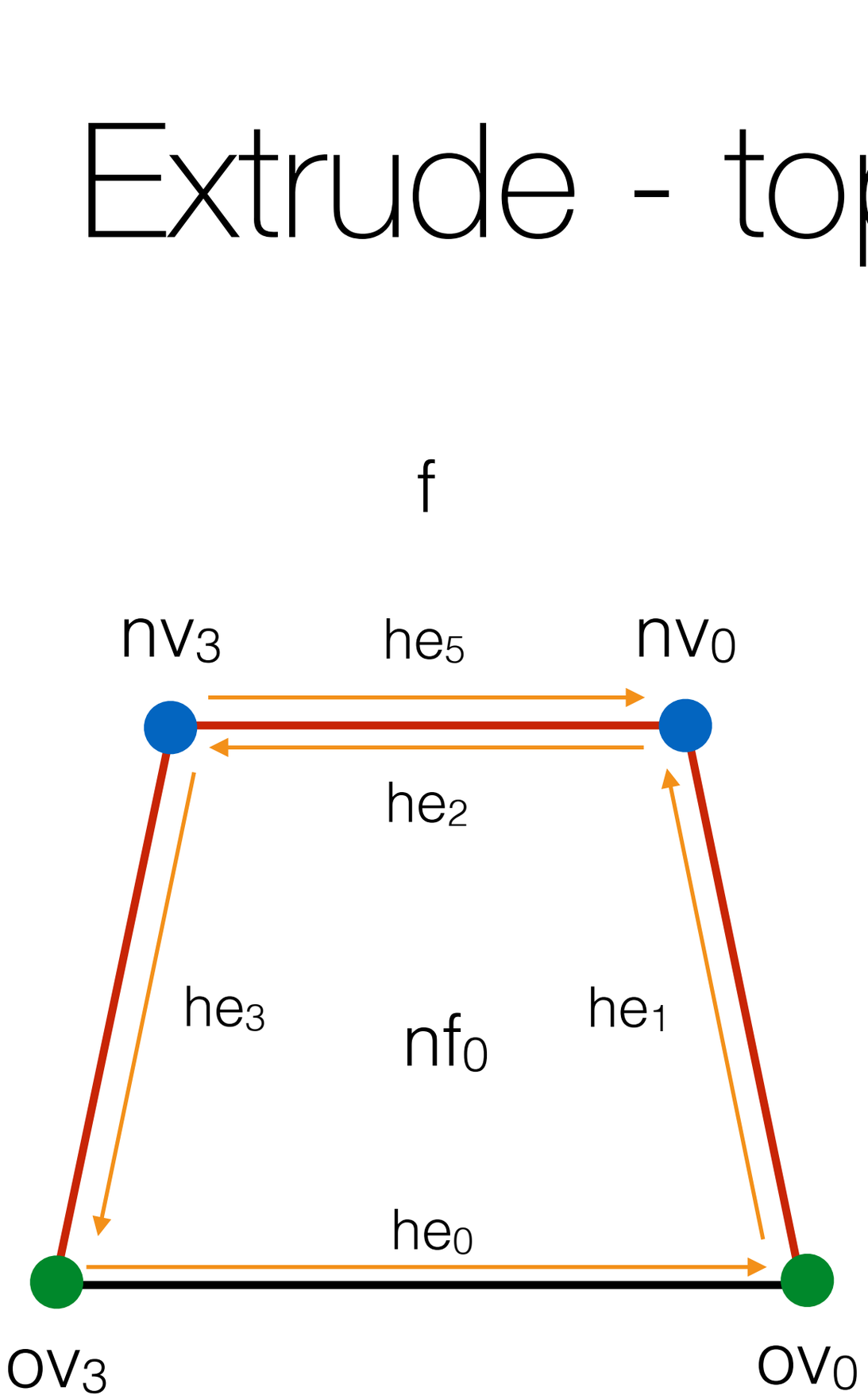
he_0 = old_halfedges[0];

Add half edges in counter clockwise order

Add half edge opposite to he_2

Extrude - topology

Should be stored before hand, such that old_halfedges[0] points at ov_0



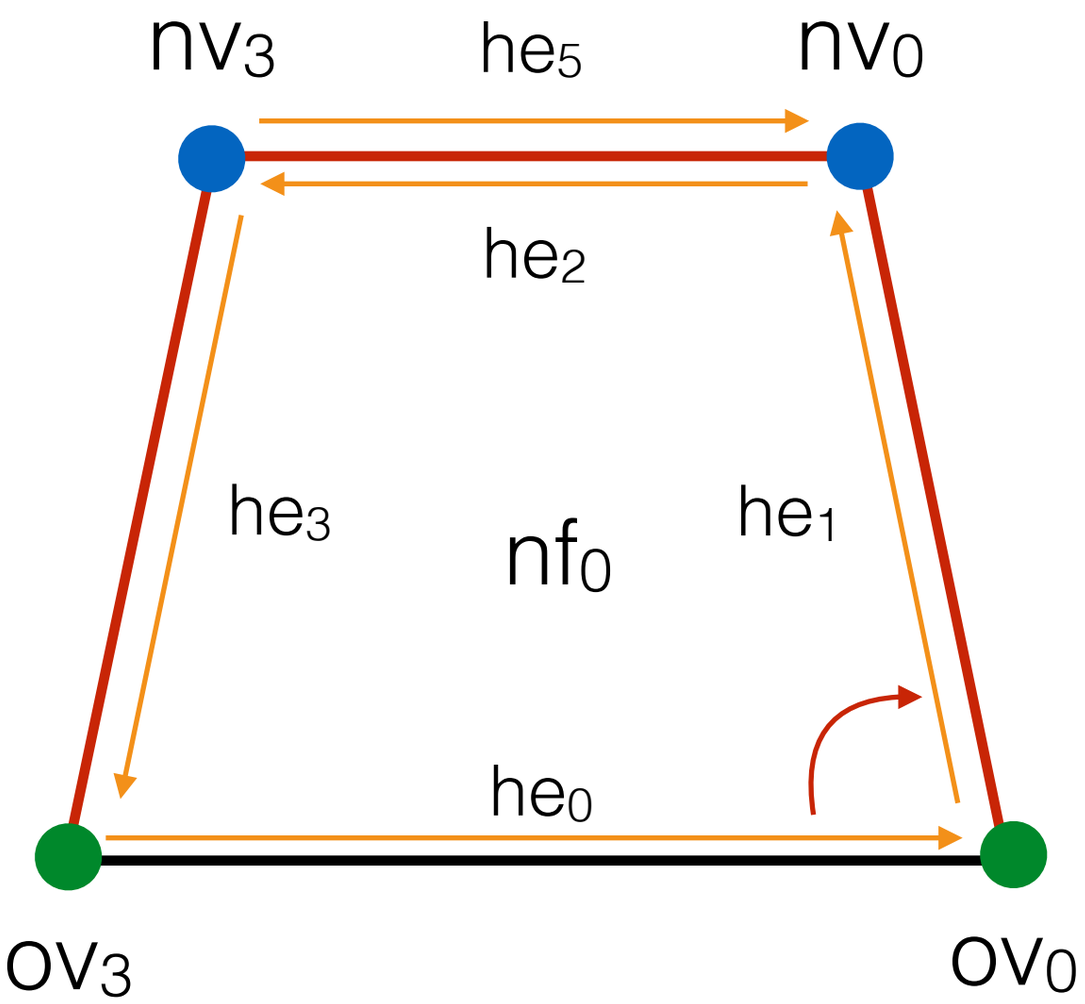
he_0 = old_halfedges[0];

Add half edges in counter clockwise order

Add half edge opposite to he_2

Relink next

Extrude - topology



Should be stored before hand, such that old_halfedges[0] points at ov_0

he_0 = old_halfedges[0];

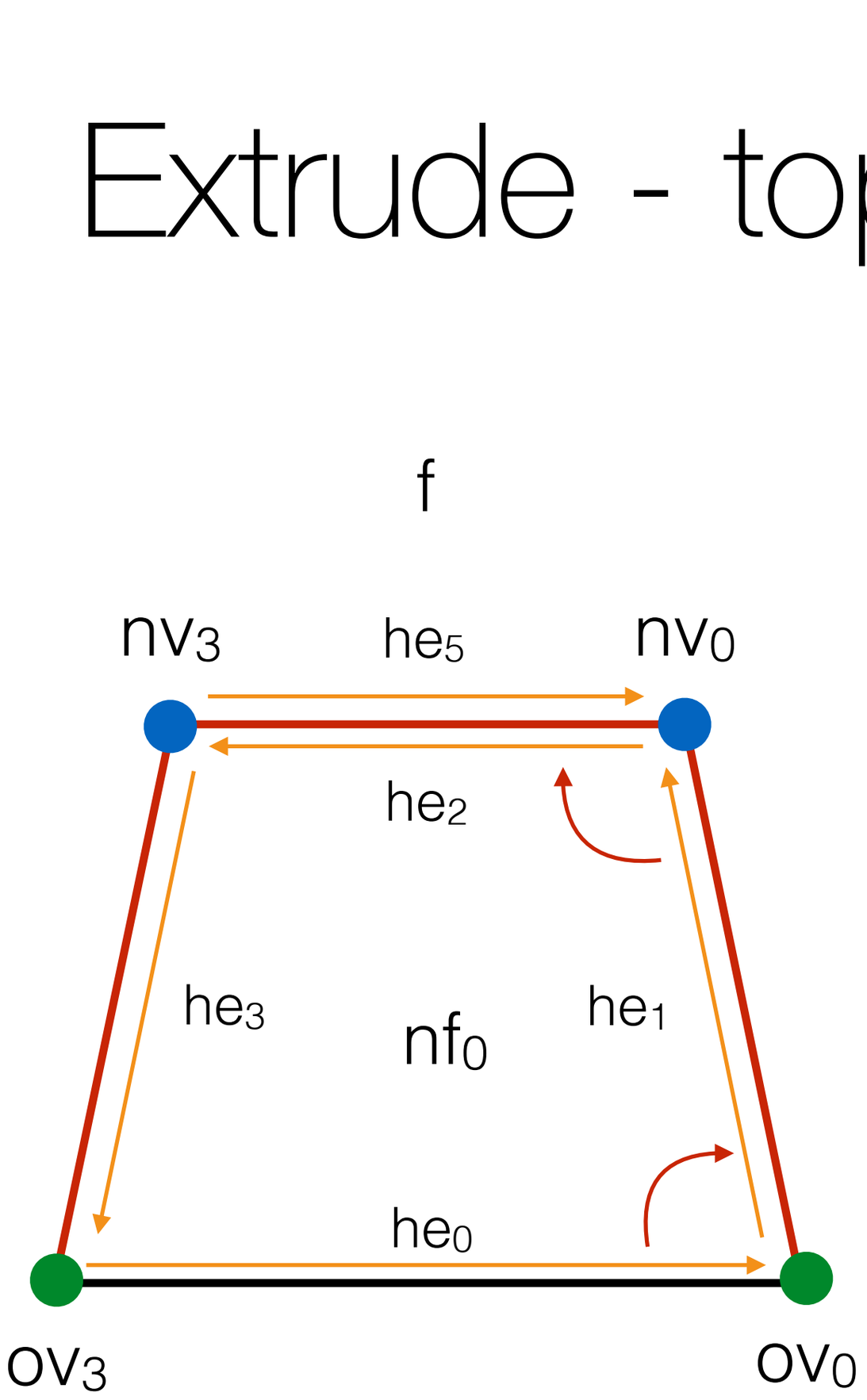
Add half edges in counter clockwise order

Add half edge opposite to he_2

Relink next

Extrude - topology

Should be stored before hand, such that old_halfedges[0] points at ov_0



he_0 = old_halfedges[0];

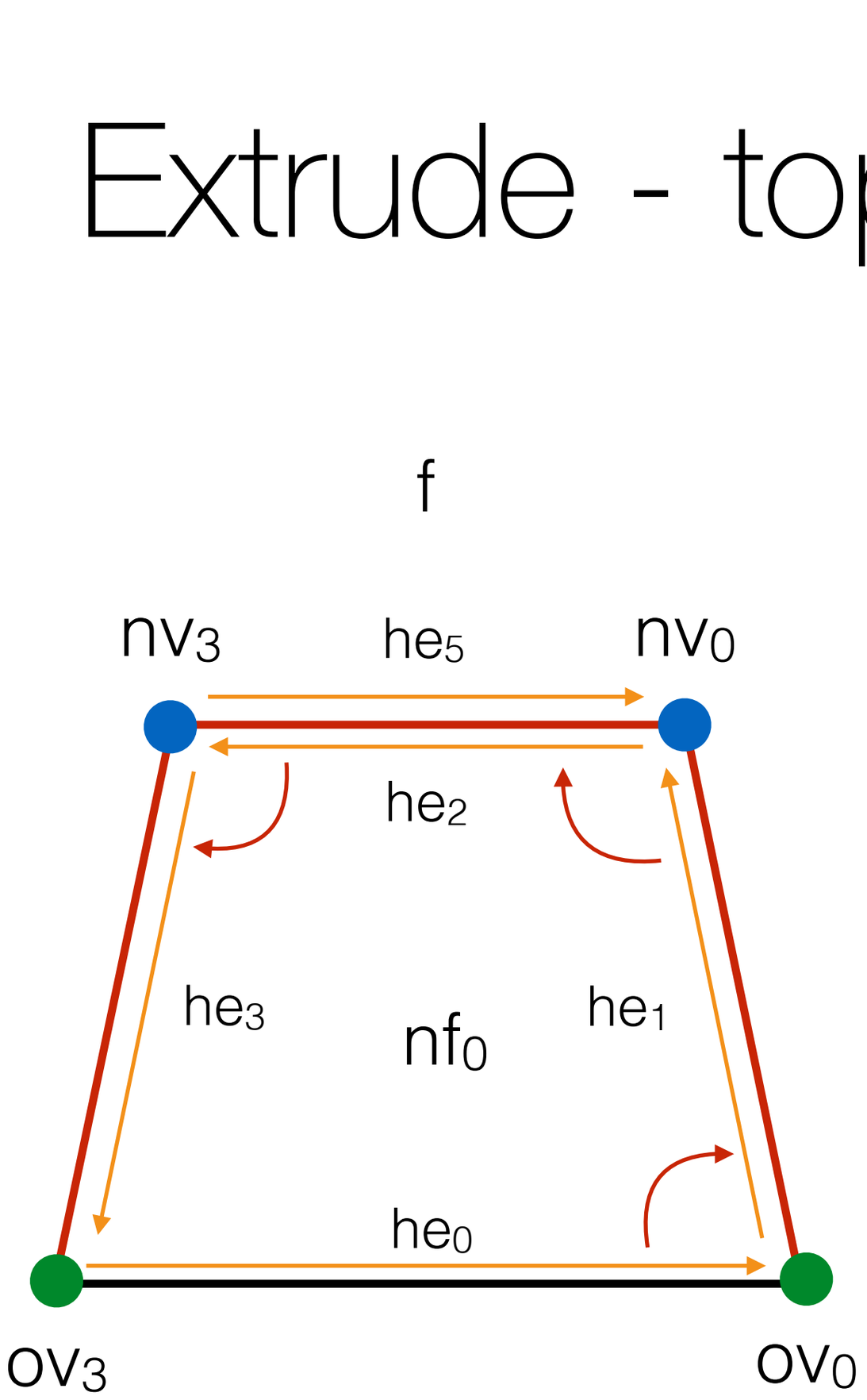
Add half edges in counter clockwise order

Add half edge opposite to he_2

Relink next

Extrude - topology

Should be stored before hand, such that old_halfedges[0] points at ov_0



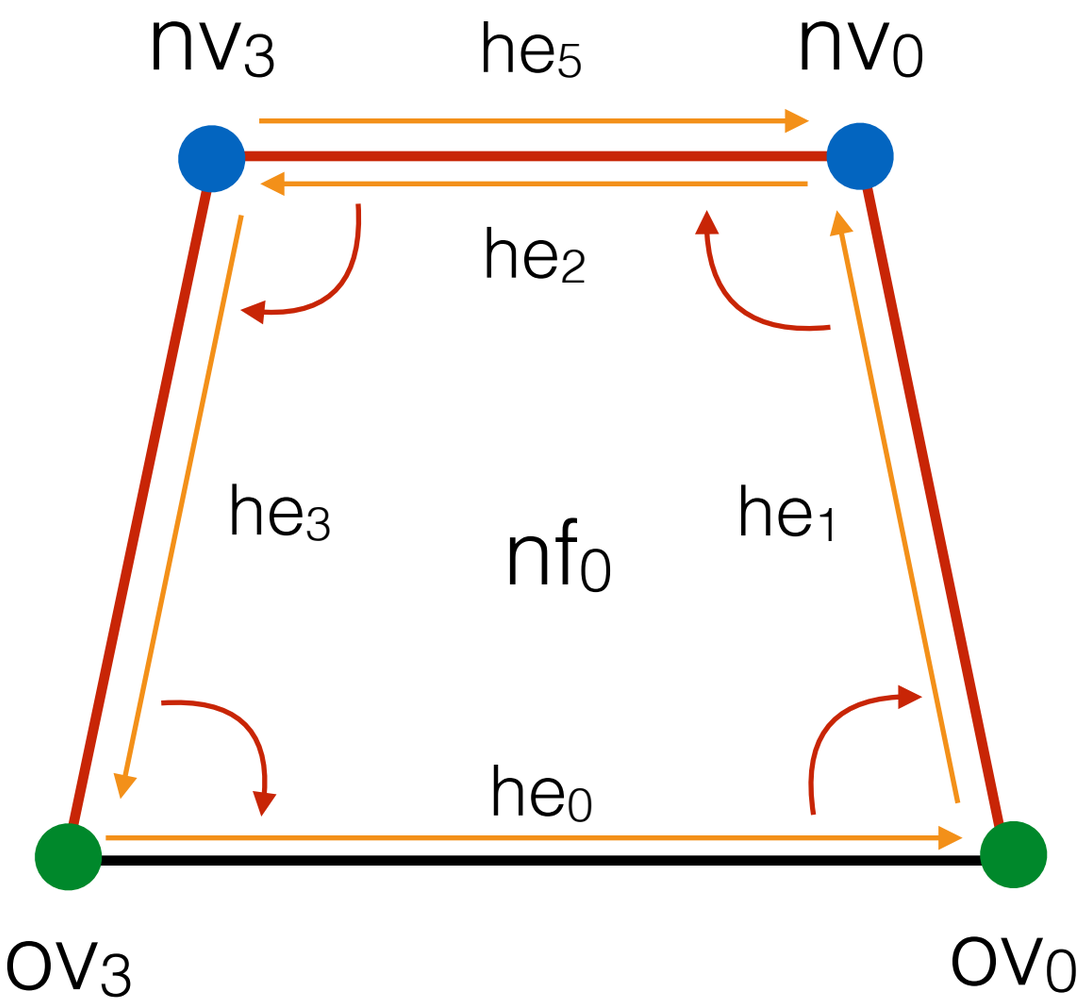
he_0 = old_halfedges[0];

Add half edges in counter clockwise order

Add half edge opposite to he_2

Relink next

Extrude - topology



Should be stored before hand, such that old_halfedges[0] points at ov_0

he_0 = old_halfedges[0];

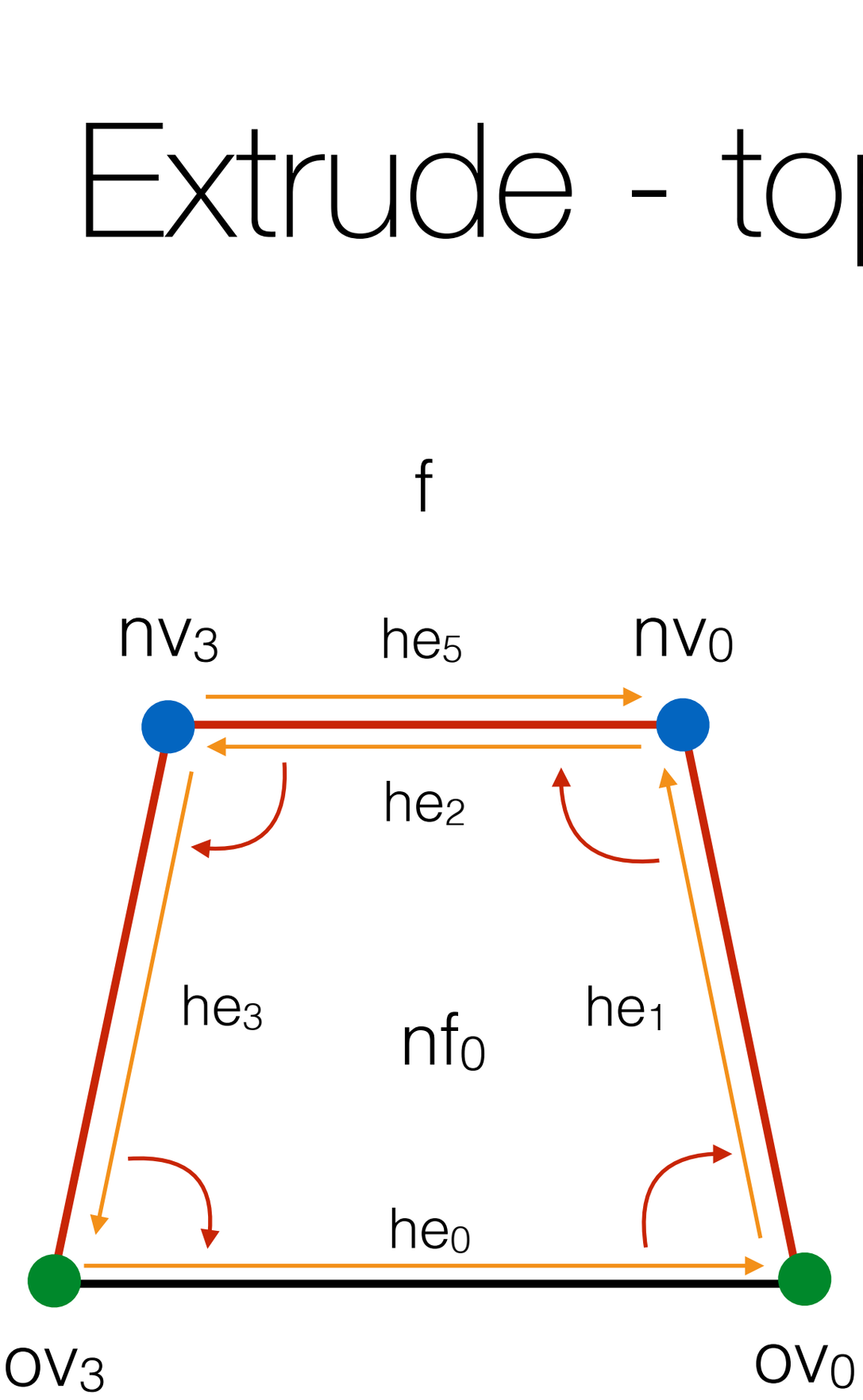
Add half edges in counter clockwise order

Add half edge opposite to he_2

Relink next

Extrude - topology

Should be stored before hand, such that old_halfedges[0] points at ov_0



he_0 = old_halfedges[0];

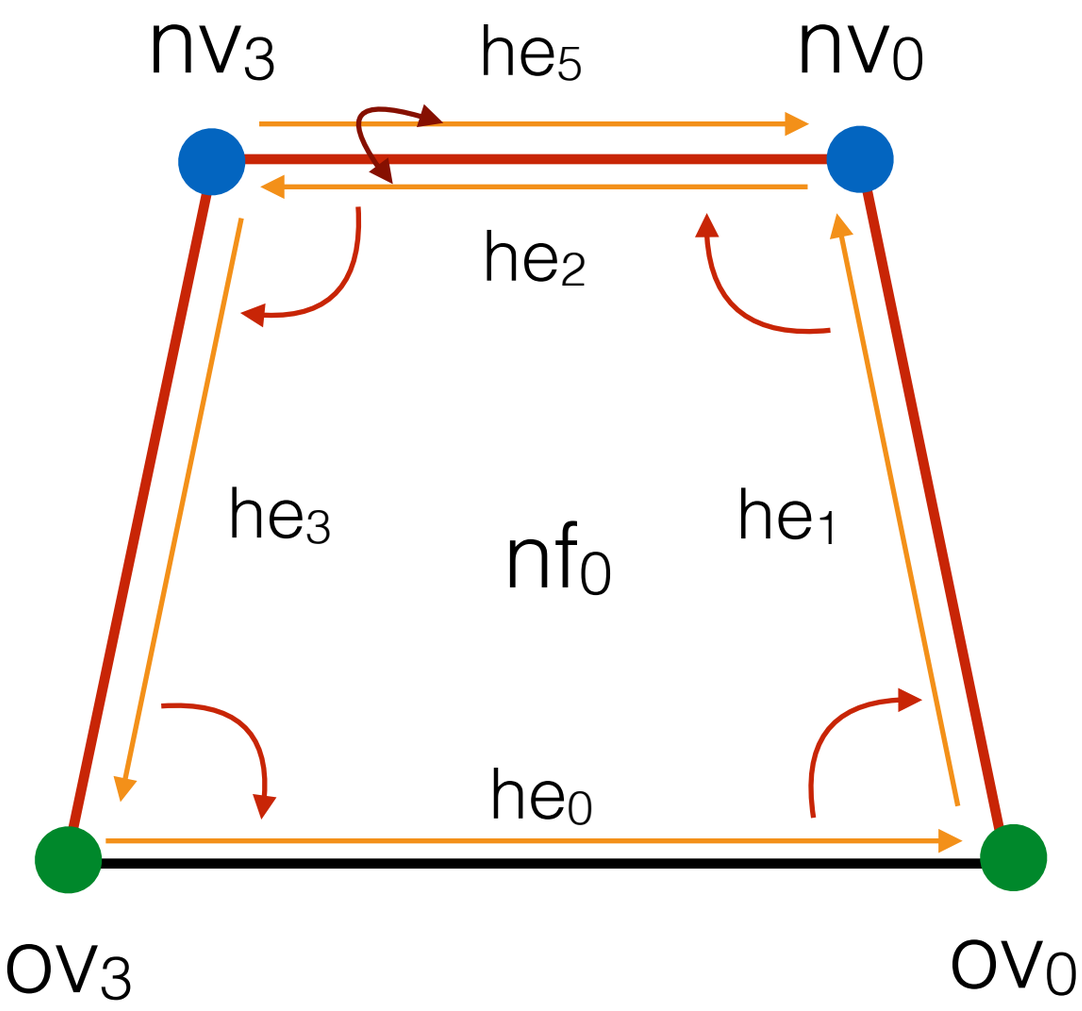
Add half edges in counter clockwise order

Add half edge opposite to he_2

Relink next

Relink opposite

Extrude - topology



Should be stored before hand, such that old_halfedges[0] points at ov_0

he_0 = old_halfedges[0];

Add half edges in counter clockwise order

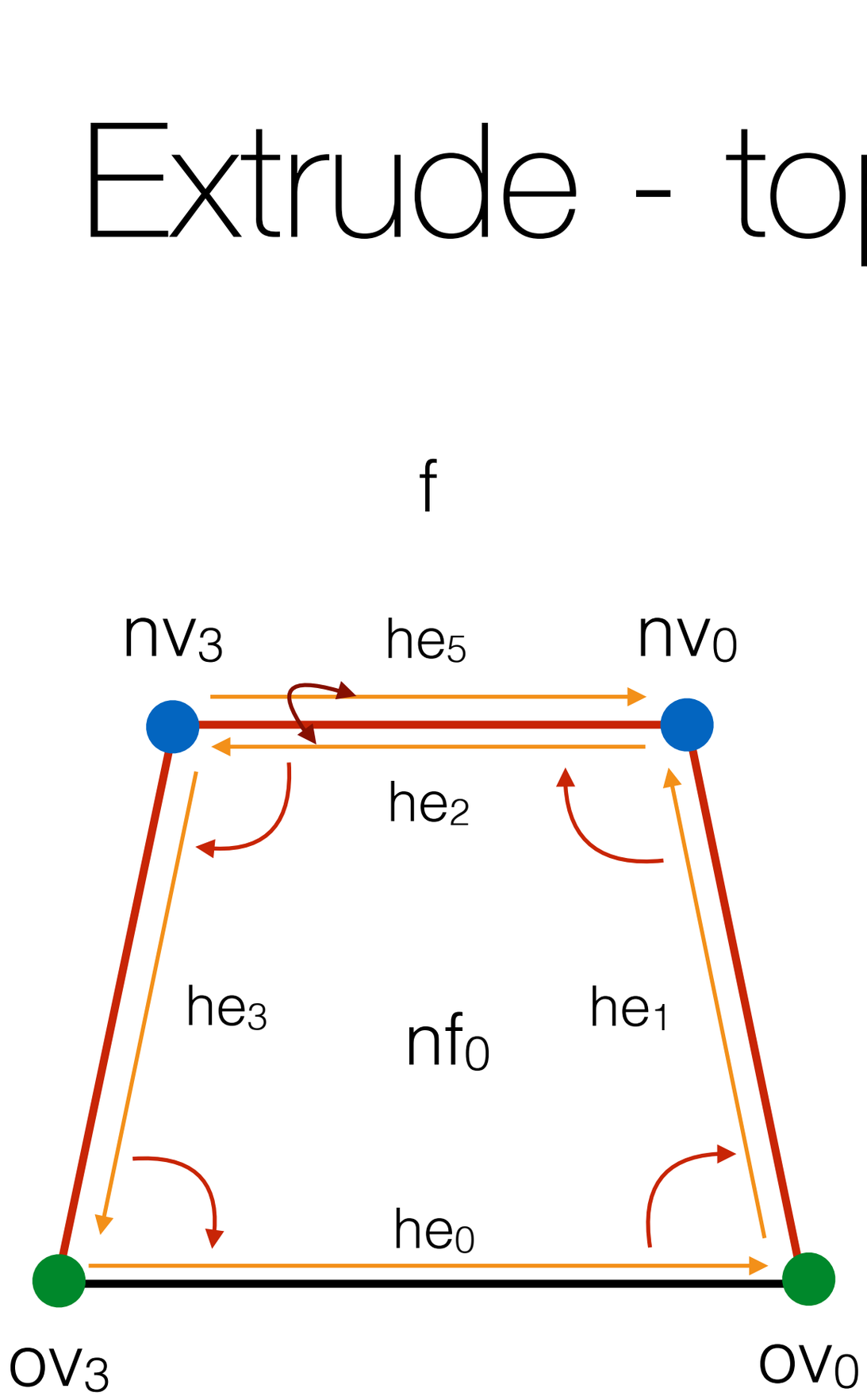
Add half edge opposite to he_2

Relink next

Relink opposite

Extrude - topology

Should be stored before hand, such that old_halfedges[0] points at ov_0



he_0 = old_halfedges[0];

Add half edges in counter clockwise order

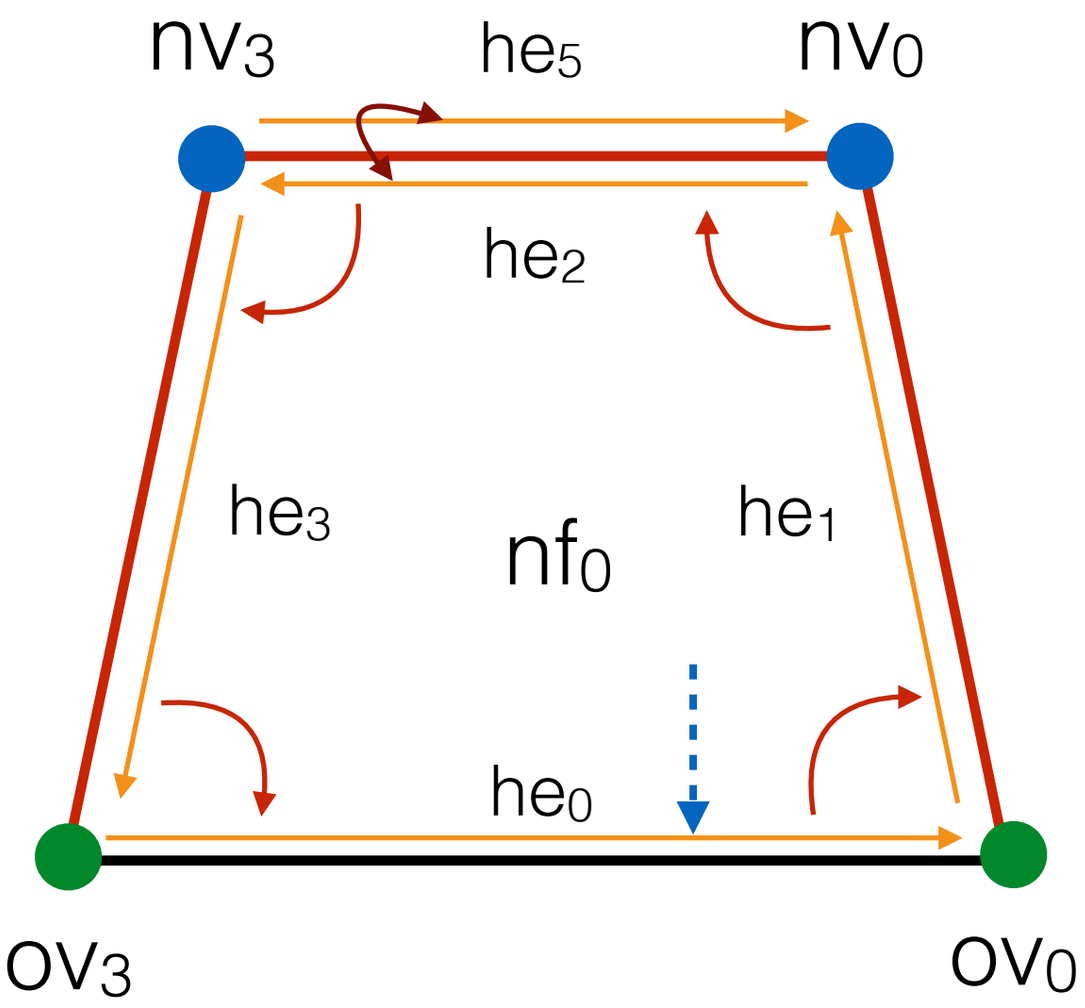
Add half edge opposite to he_2

Relink next

Relink opposite

Relink faces

Extrude - topology



Should be stored before hand, such that old_halfedges[0] points at ov_0

he_0 = old_halfedges[0];

Add half edges in counter clockwise order

Add half edge opposite to he_2

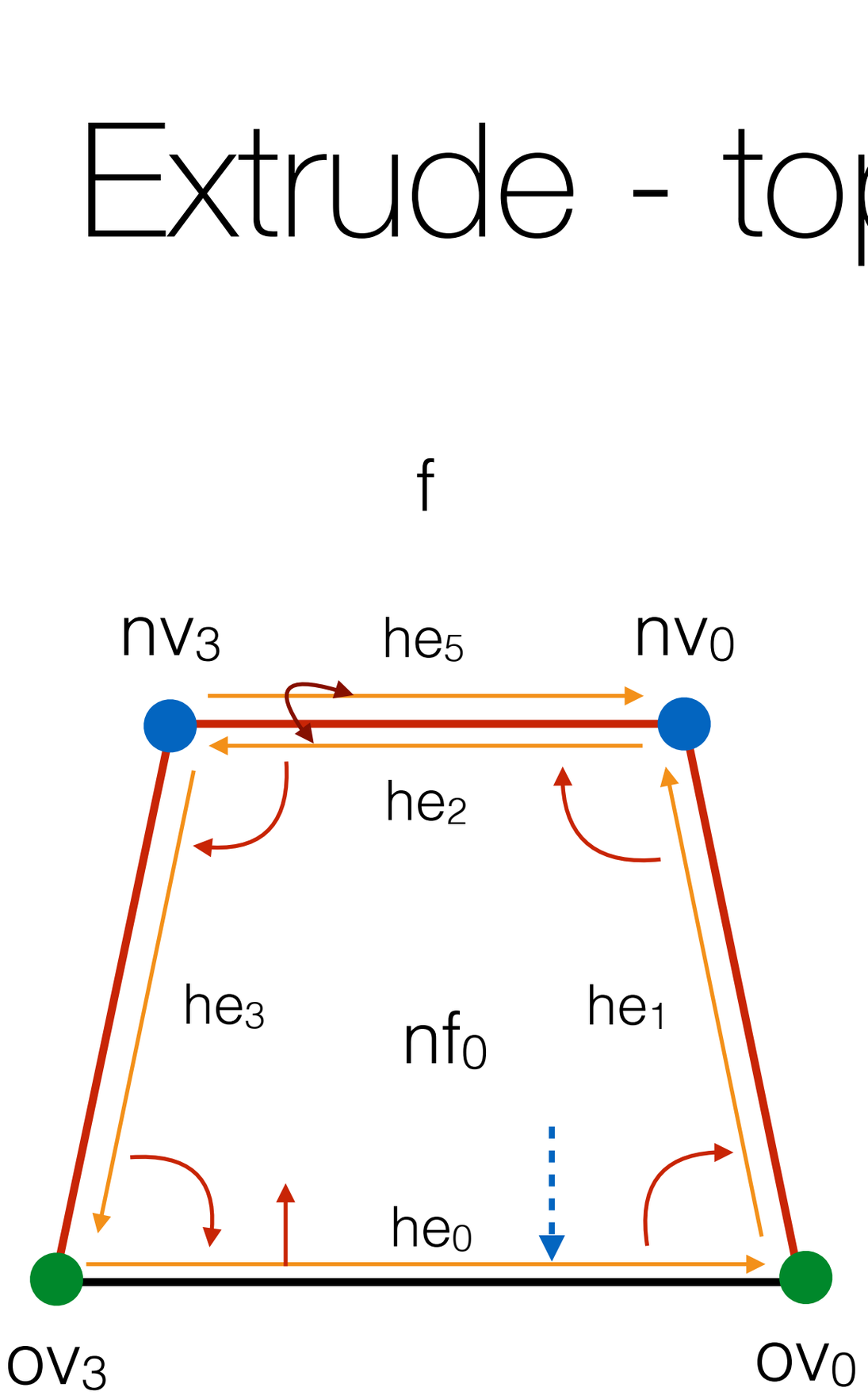
Relink next

Relink opposite

Relink faces

Extrude - topology

Should be stored before hand, such that old_halfedges[0] points at ov_0



he_0 = old_halfedges[0];

Add half edges in counter clockwise order

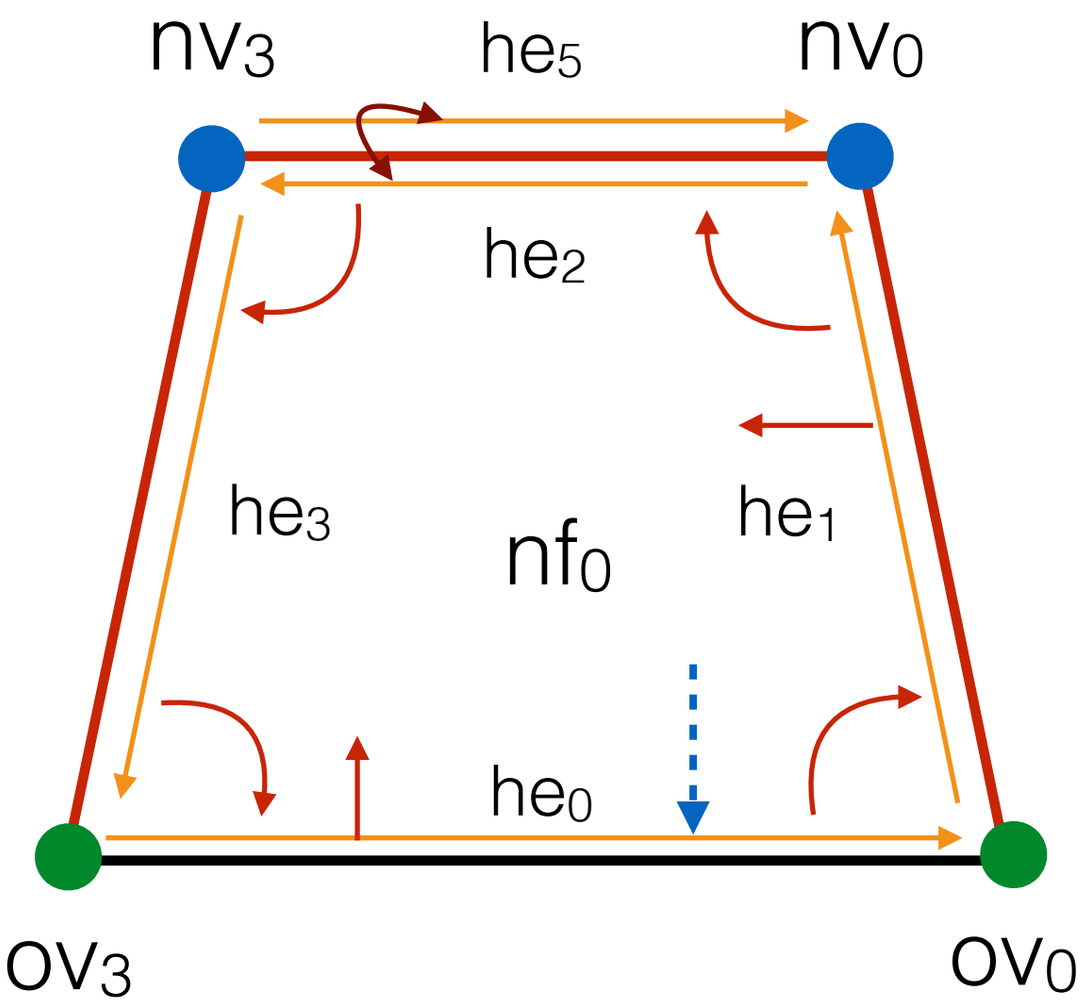
Add half edge opposite to he_2

Relink next

Relink opposite

Relink faces

Extrude - topology



Should be stored before hand, such that old_halfedges[0] points at ov_0

he_0 = old_halfedges[0];

Add half edges in counter clockwise order

Add half edge opposite to he_2

Relink next

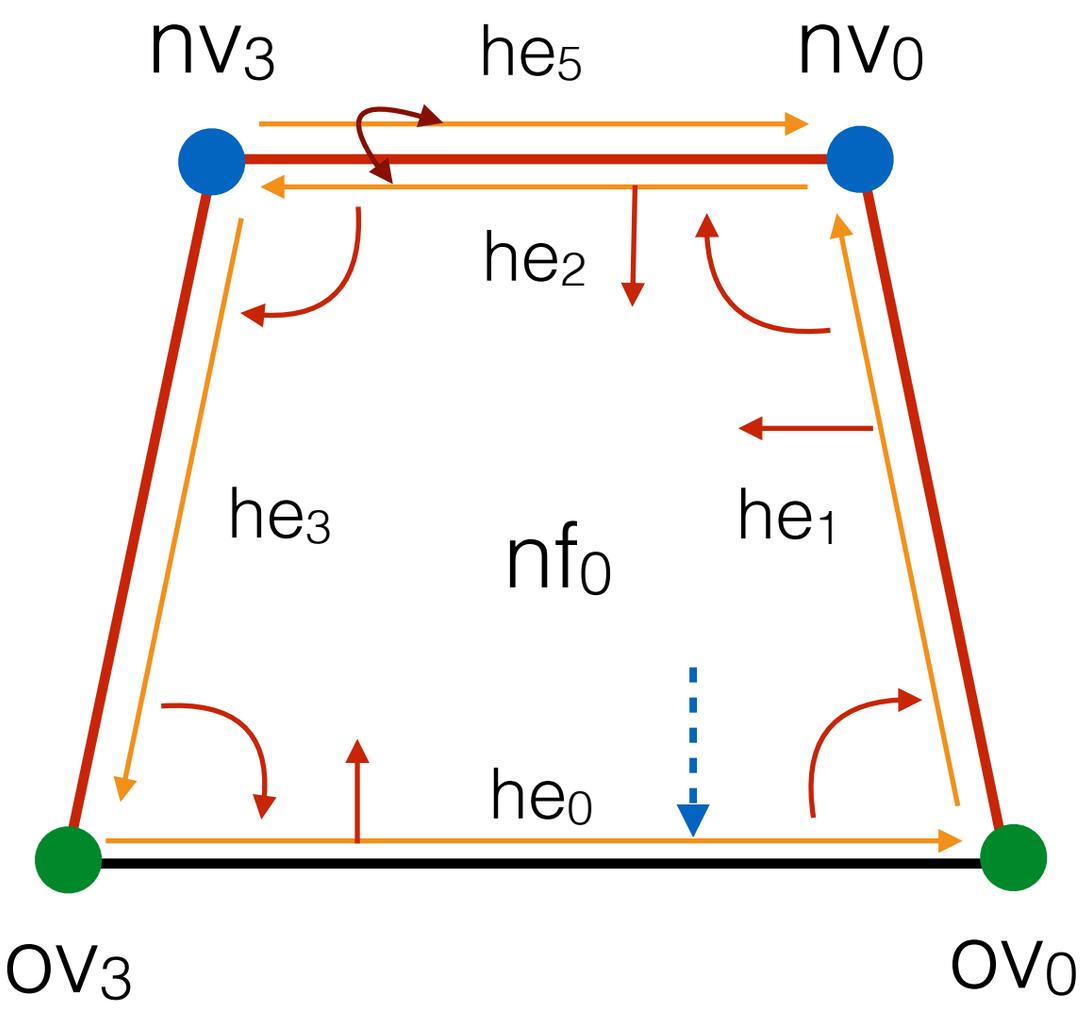
Relink opposite

Relink faces

Extrude - topology

Should be stored before hand, such that old_halfedges[0] points at ov_0

f



he_0 = old_halfedges[0];

Add half edges in counter clockwise order

Add half edge opposite to he_2

Relink next

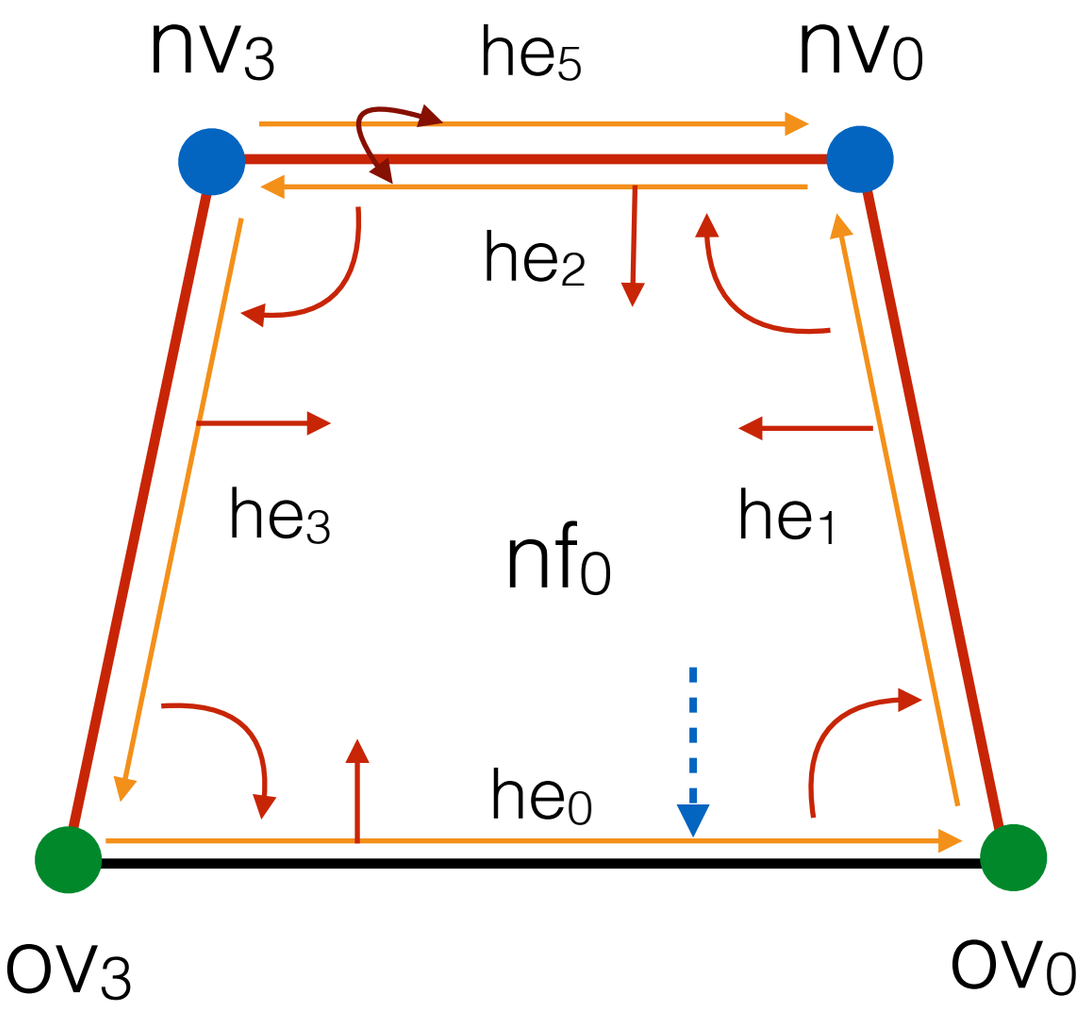
Relink opposite

Relink faces

Extrude - topology

Should be stored before hand, such that old_halfedges[0] points at ov_0

f



he_0 = old_halfedges[0];

Add half edges in counter clockwise order

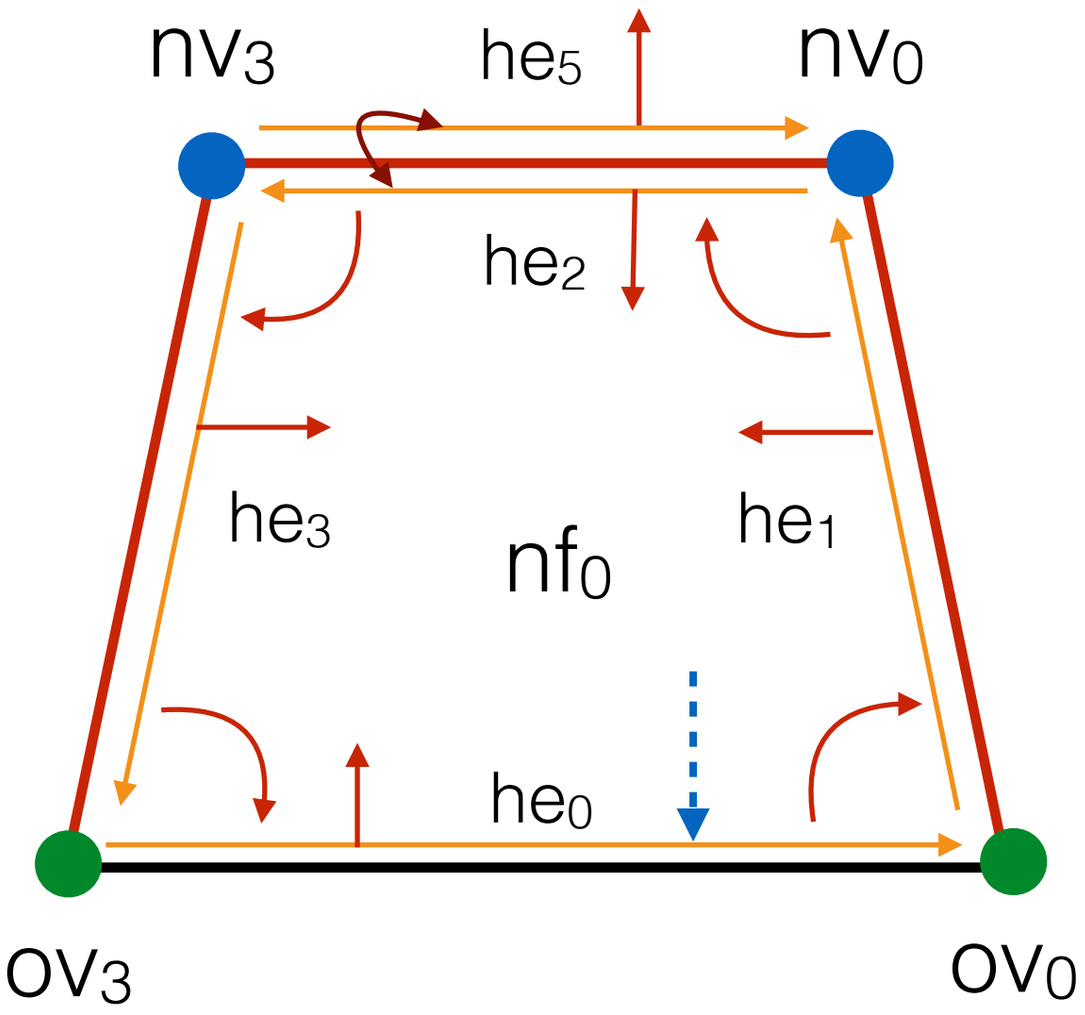
Add half edge opposite to he_2

Relink next

Relink opposite

Relink faces

Extrude - topology



Should be stored before hand, such that old_halfedges[0] points at ov_0

he_0 = old_halfedges[0];

Add half edges in counter clockwise order

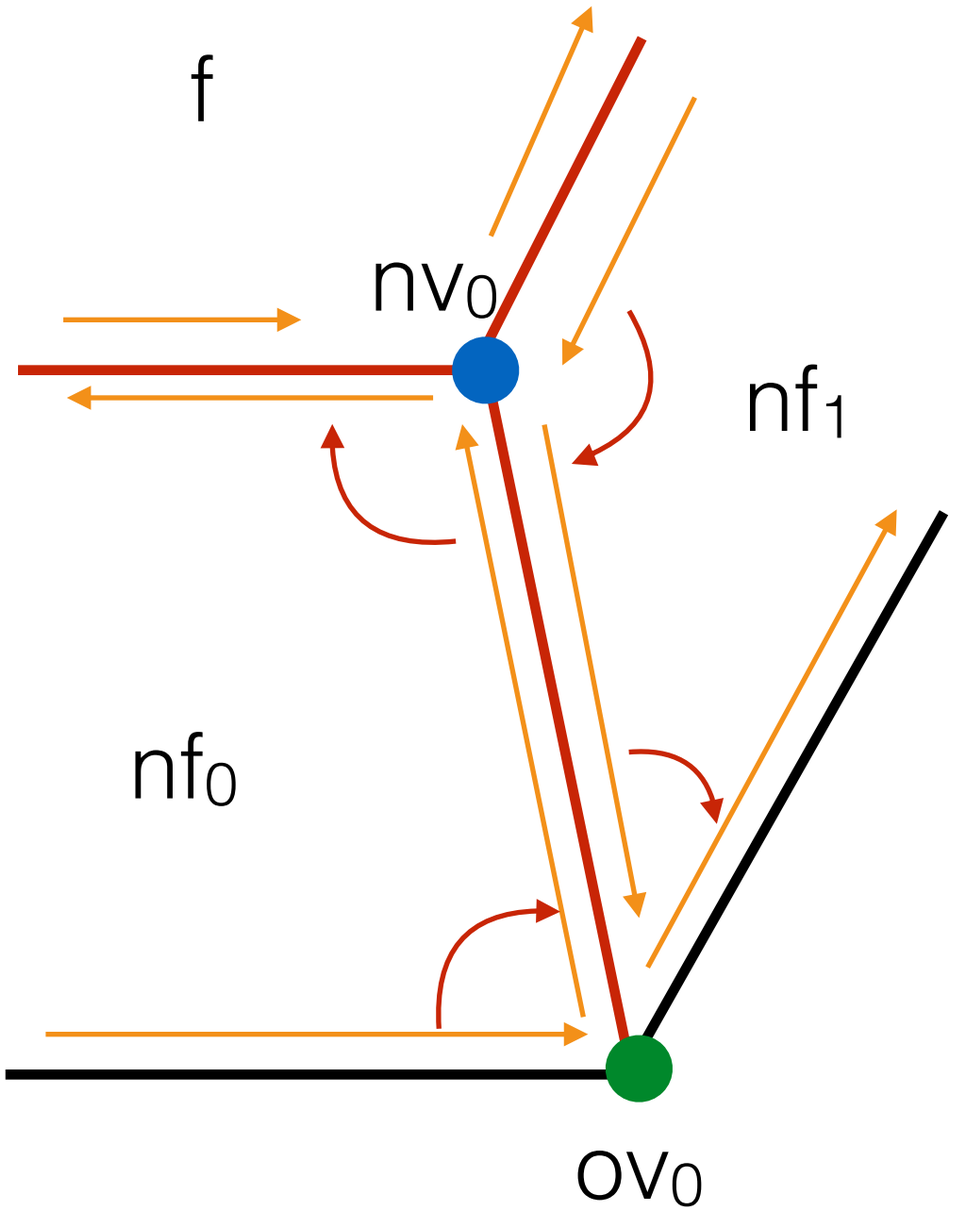
Add half edge opposite to he_2

Relink next

Relink opposite

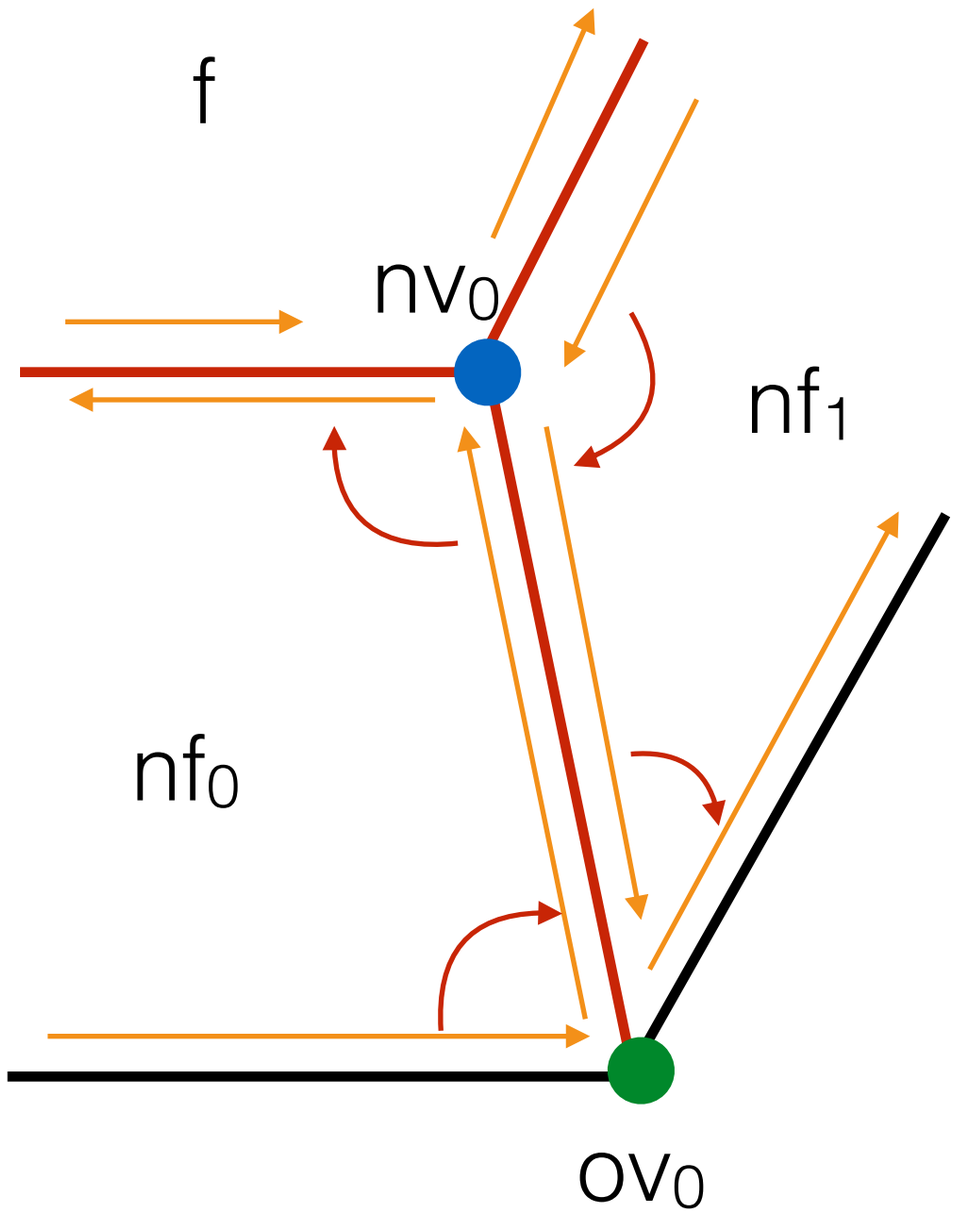
Relink faces

Extrude - topology



Missing next links around face.
Missing opposite link on edges.

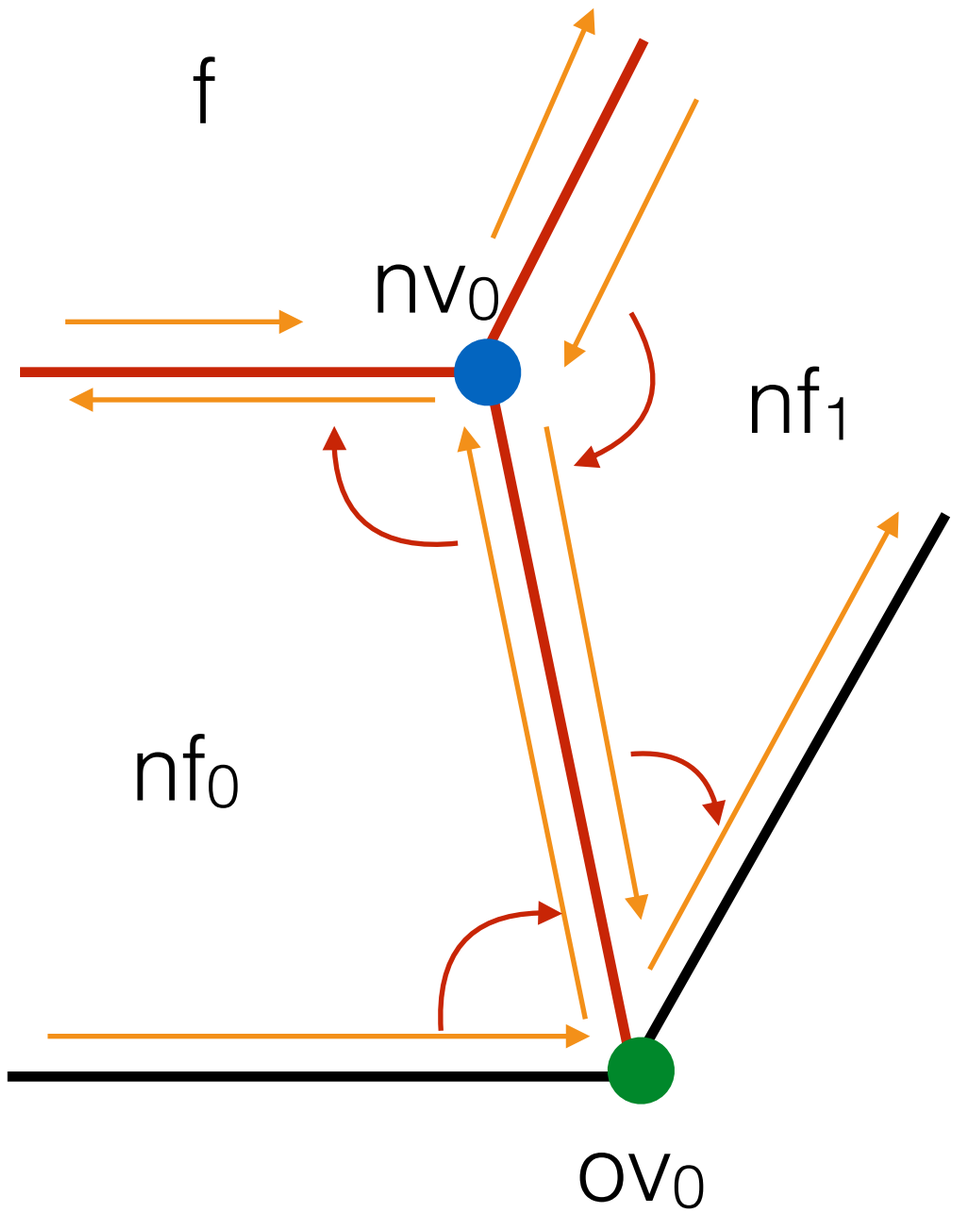
Extrude - topology



Missing next links around face.
Missing opposite link on edges.

Store references in separate arrays

Extrude - topology

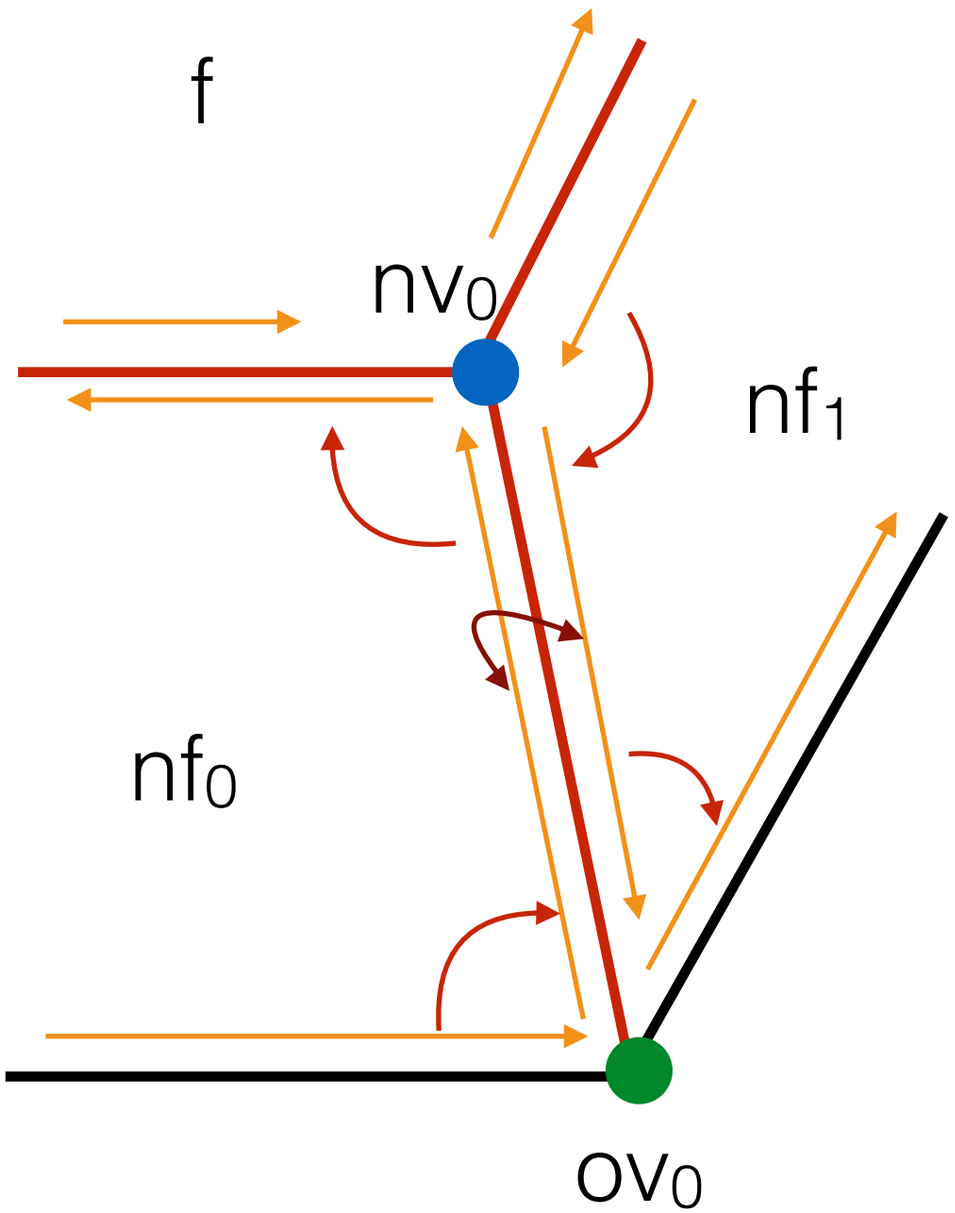


Missing next links around face.
Missing opposite link on edges.

Store references in separate arrays

Relink

Extrude - topology

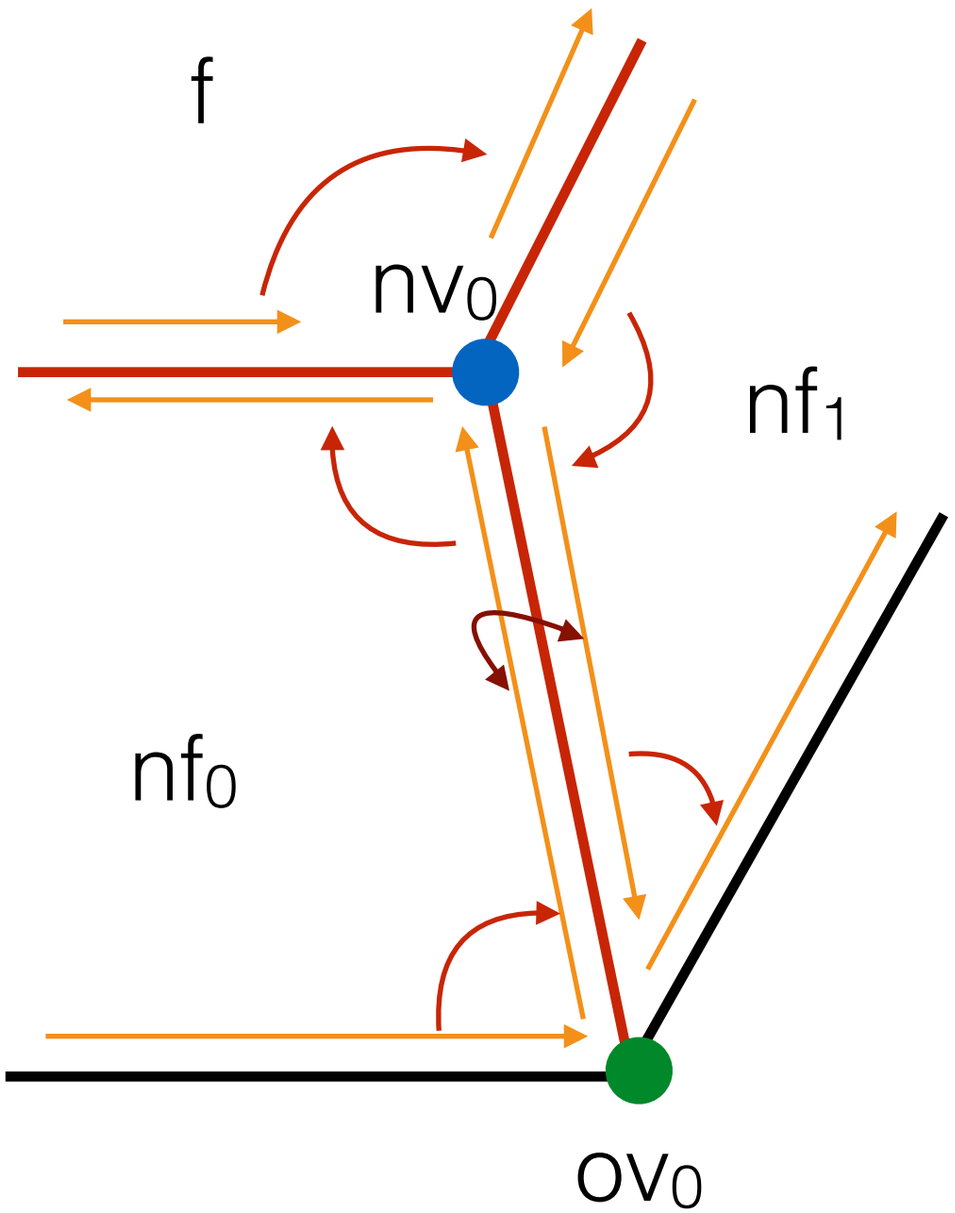


Missing next links around face.
Missing opposite link on edges.

Store references in separate arrays

Relink

Extrude - topology



Missing next links around face.
Missing opposite link on edges.

Store references in separate arrays

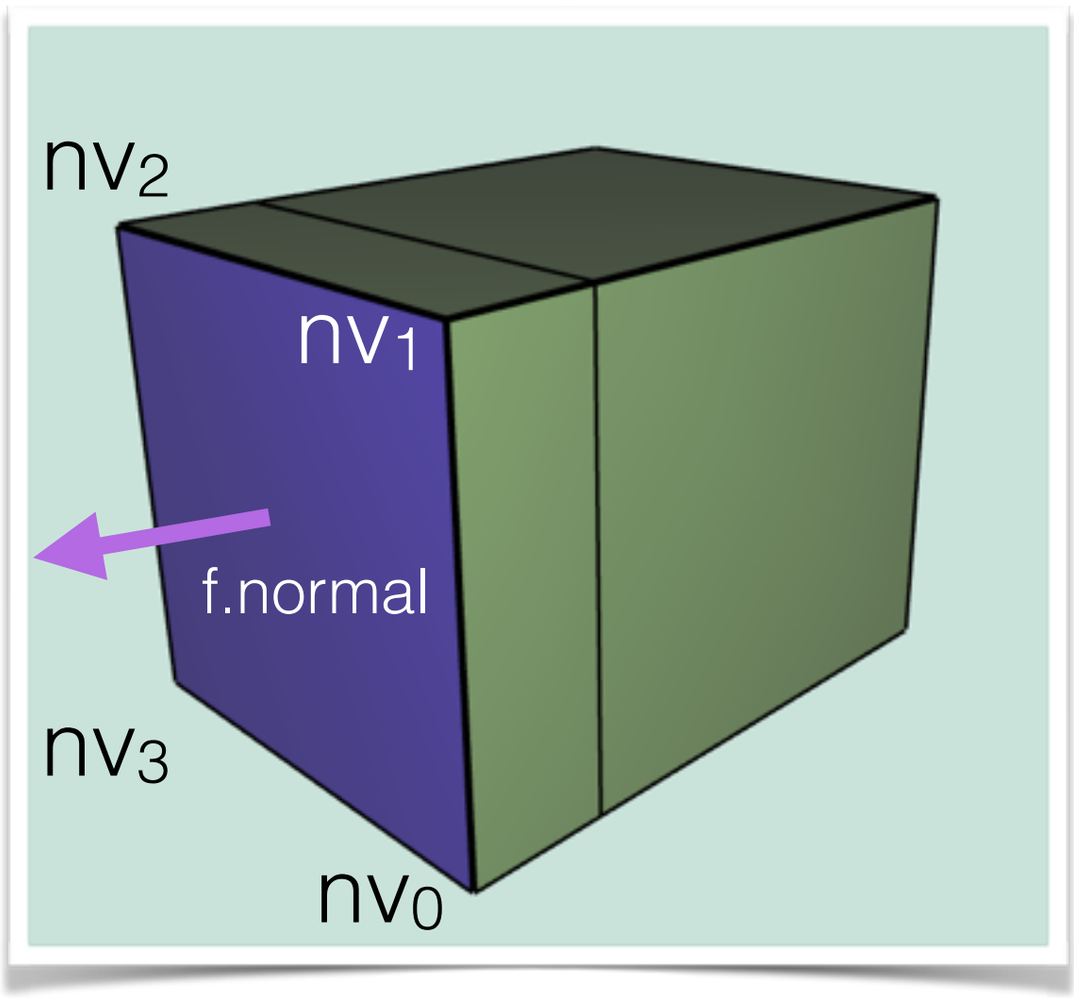
Relink

Extrude - geometry

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

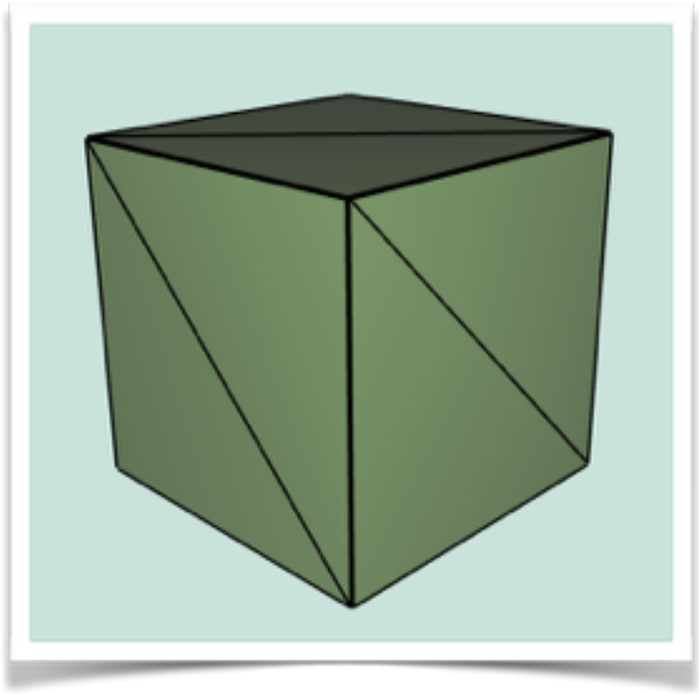
Extrude - geometry

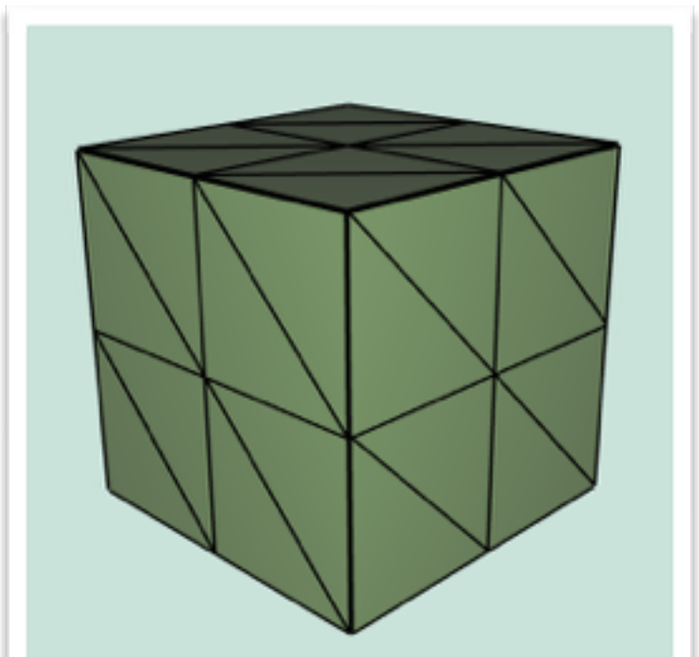
- Actually, very simple
- Move each nv_i by `factor * f.normal`



Triangle Subdivision

- 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



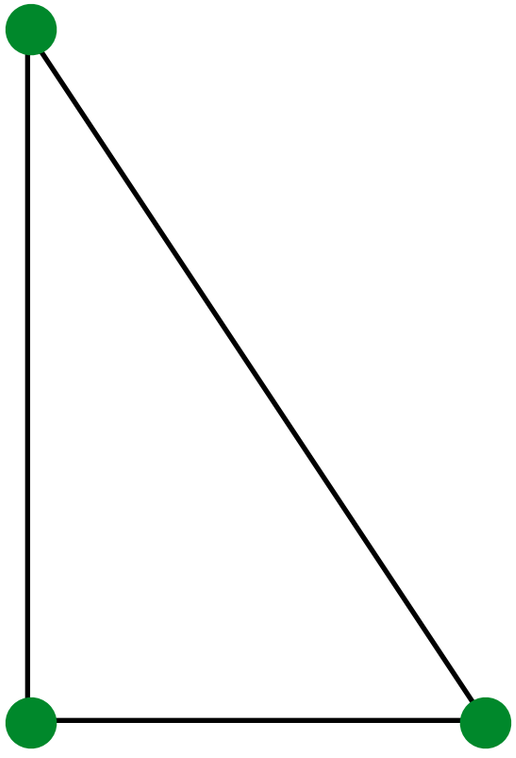


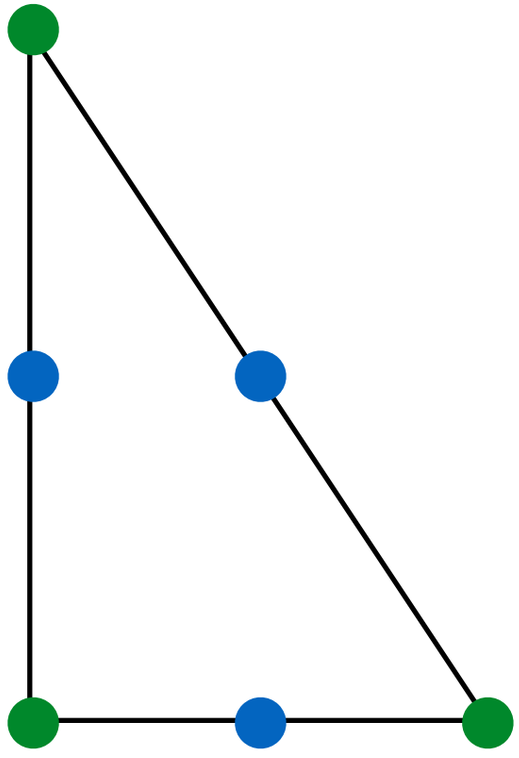
TriSub - topology

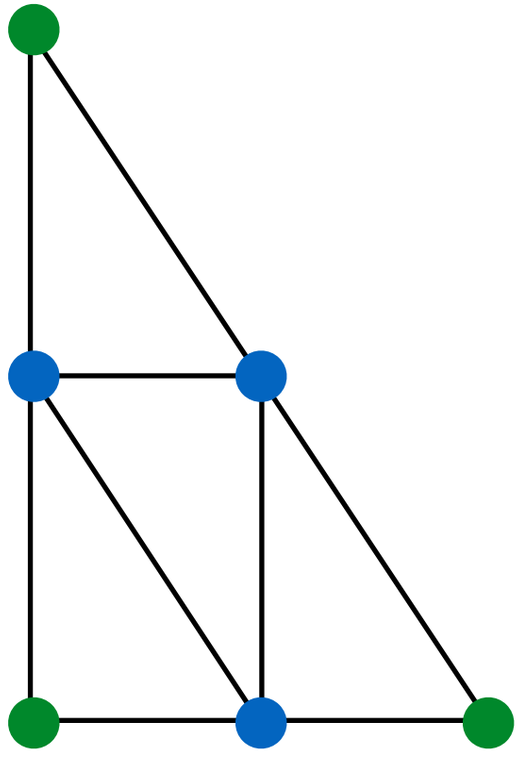
- Need to split all edges!
- Create list of half edges
  - Half of them, when splitting halfedge, opposite will also be split
- Join new vertices around a face
  - Determine whether a vertex is old or new by index in vertices array
  - All new will be added to the end of the array!

TriSub - topology

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





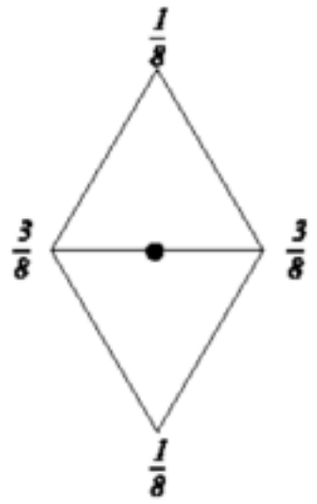


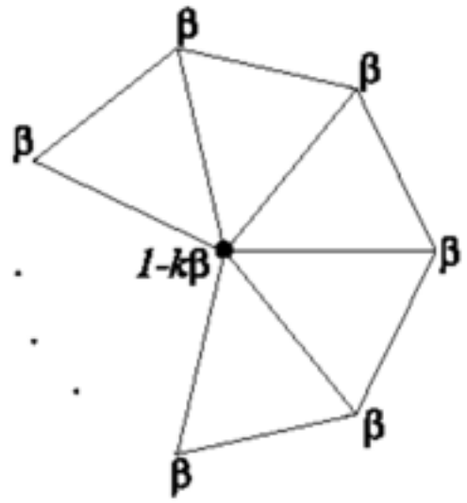
TriSub - geometry

- None - we're done!

- For Loop - store array of new positions for each vertex, where you will write positions calculated according to weight rules

- After done with topology, update positions!





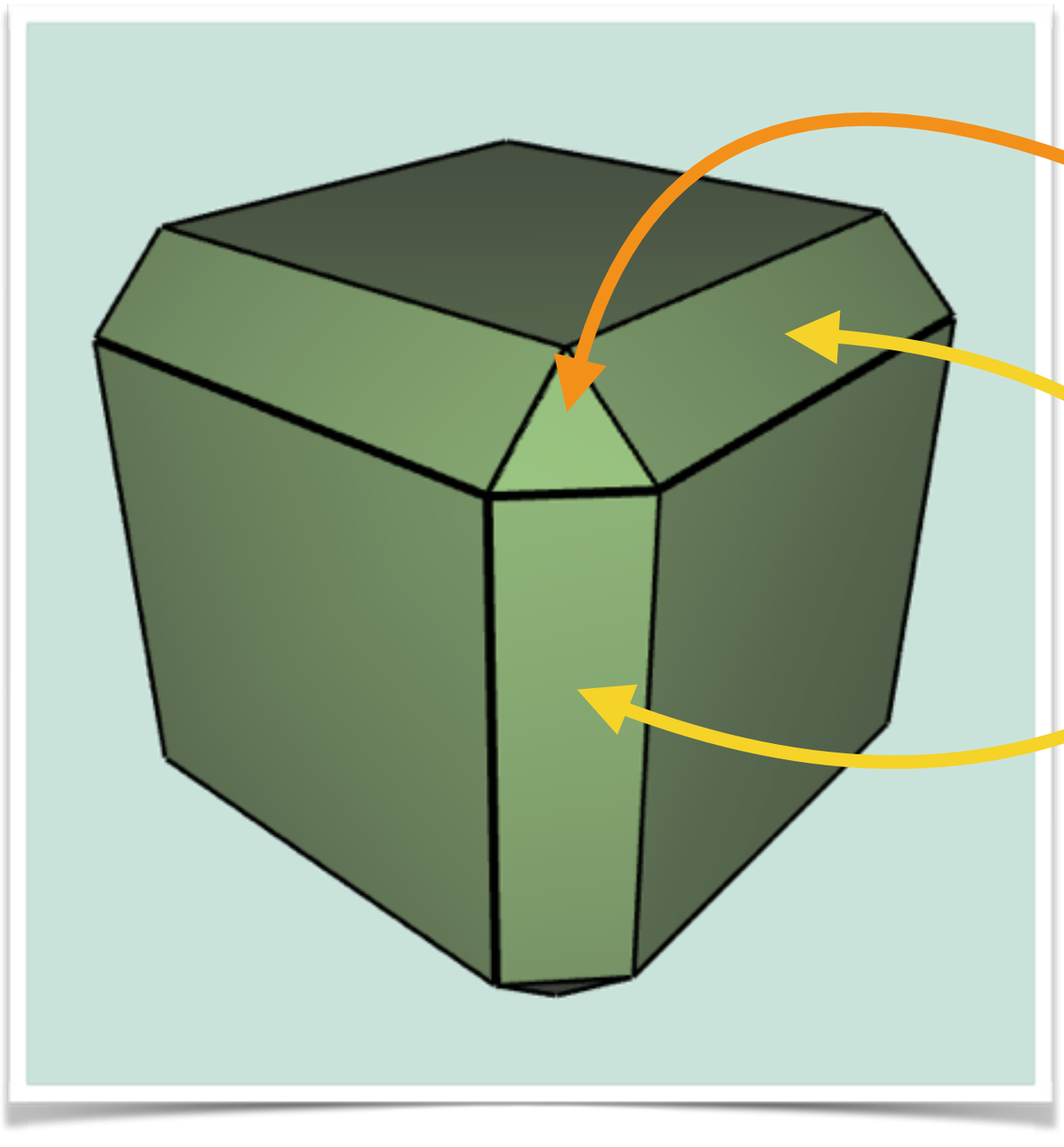
$$\beta = \begin{cases} \frac{3}{8n} & n > 3 \\ \frac{3}{16} & n = 3 \end{cases}$$

Optional features

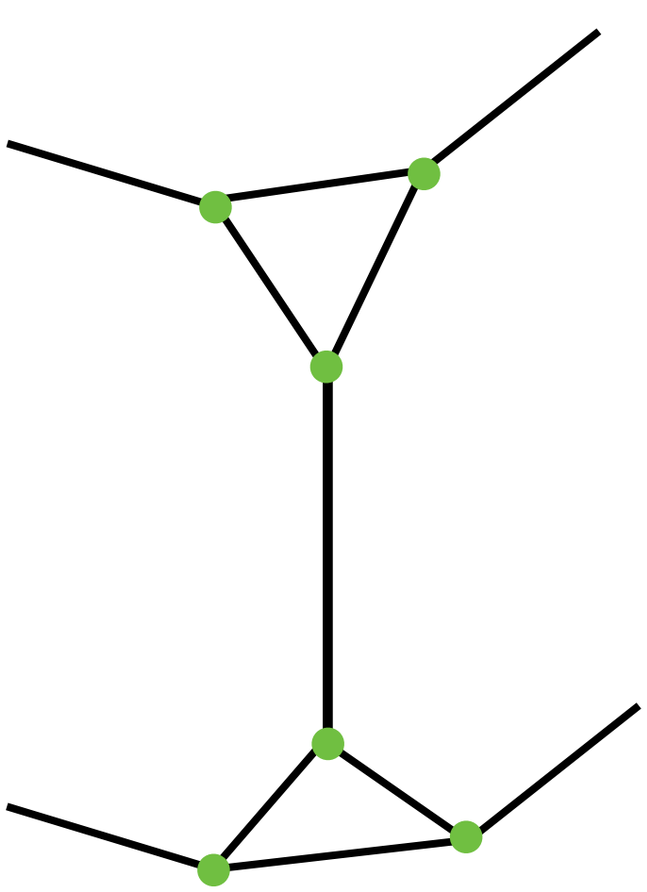
- Bevel
- Quad Subdivision
- We will just gloss over those

Bevel

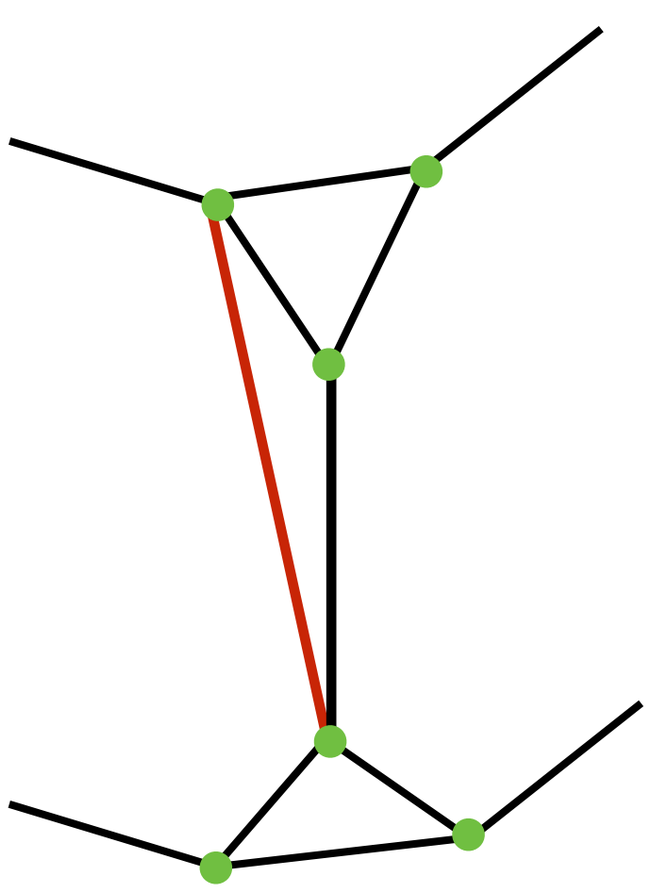
- Let's think about required topology.



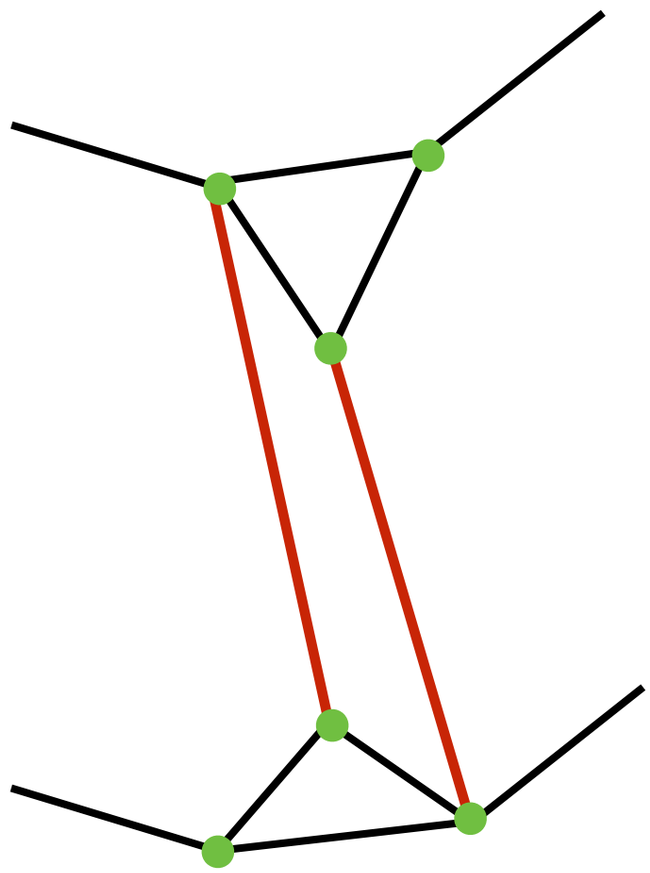
Bevel topology



Start with truncate



Cut a triangle



Relink original edge

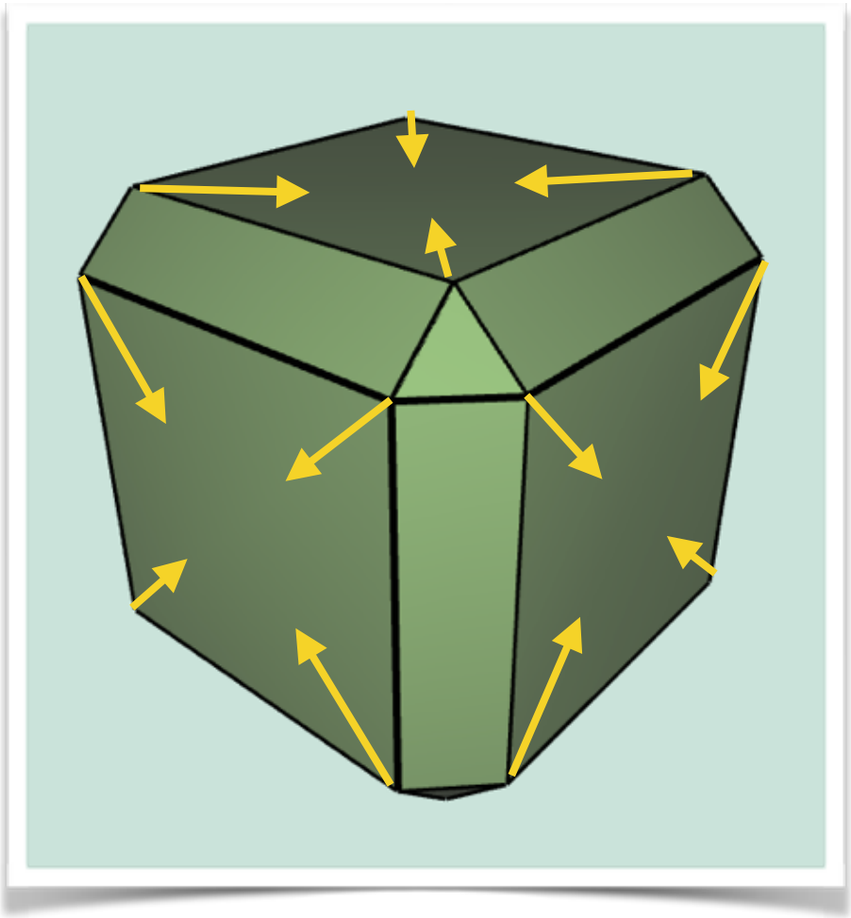
Bevel - topology

- Select half edges that join truncated points
- Caution when selecting half-edges to perform split
  - Make sure you're not double counting
- Moving an edge requires manual relinking

Bevel - geometry

- All new vertices are at location of the respective original vertex

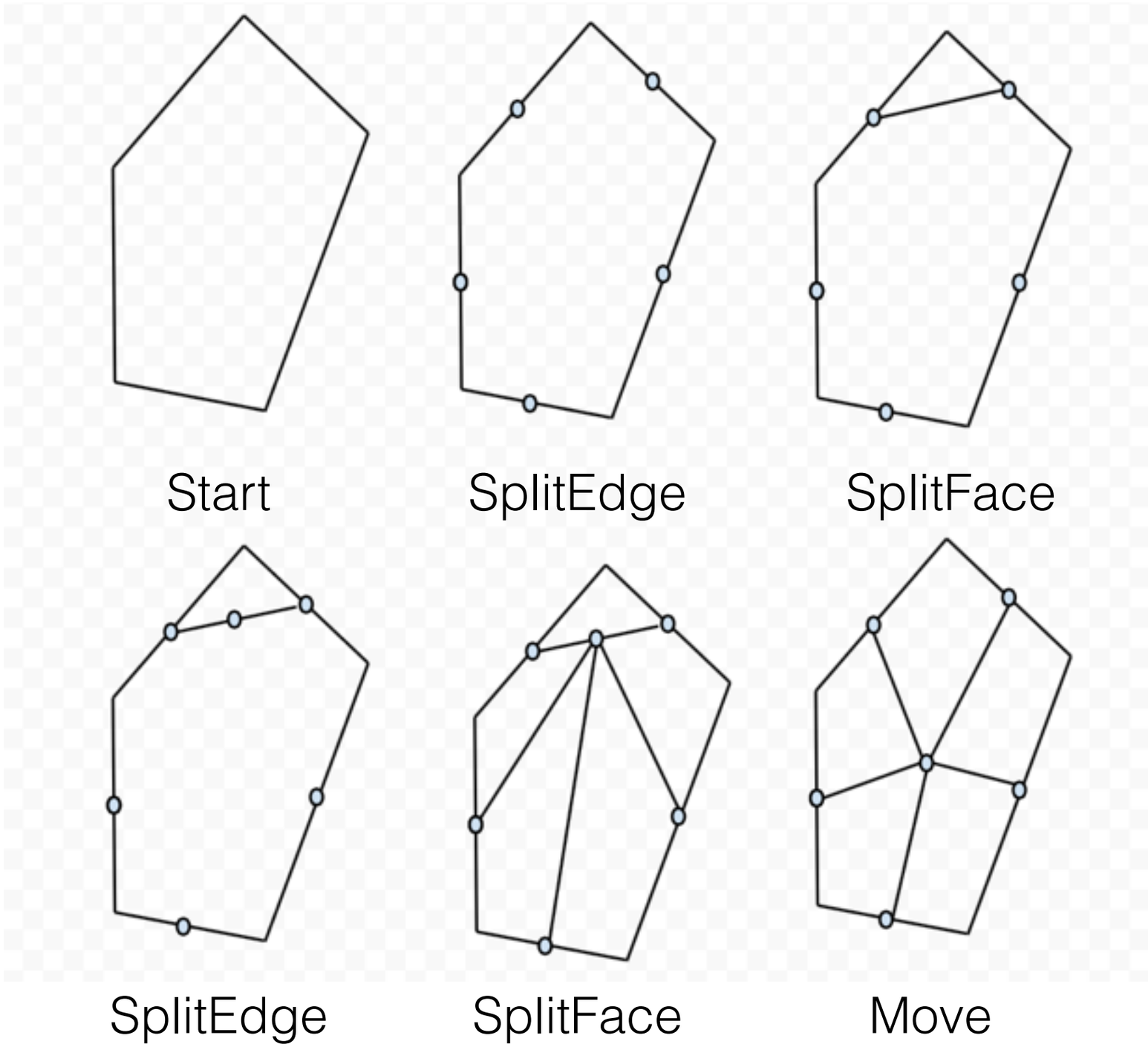
- Can move them towards the centroid of the main face



Quad Subdivision

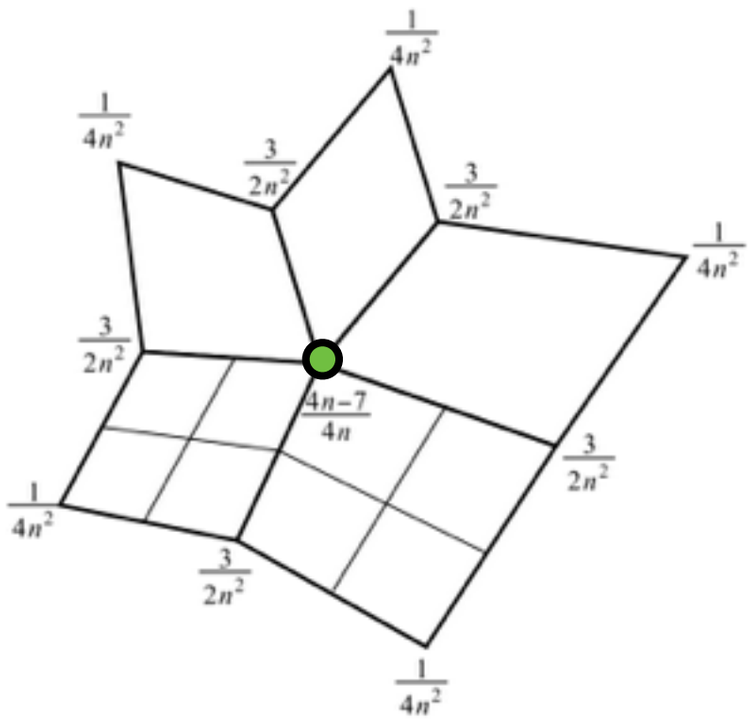
- n-gon to quad split
  - Split each edge (SplitEdge)
  - Join 2 new vertices (SplitFace)
  - Split newly create edge (SplitEdge)
  - Join rest of new vertices (SplitFace)
  - Move to interior vertex to centroid location

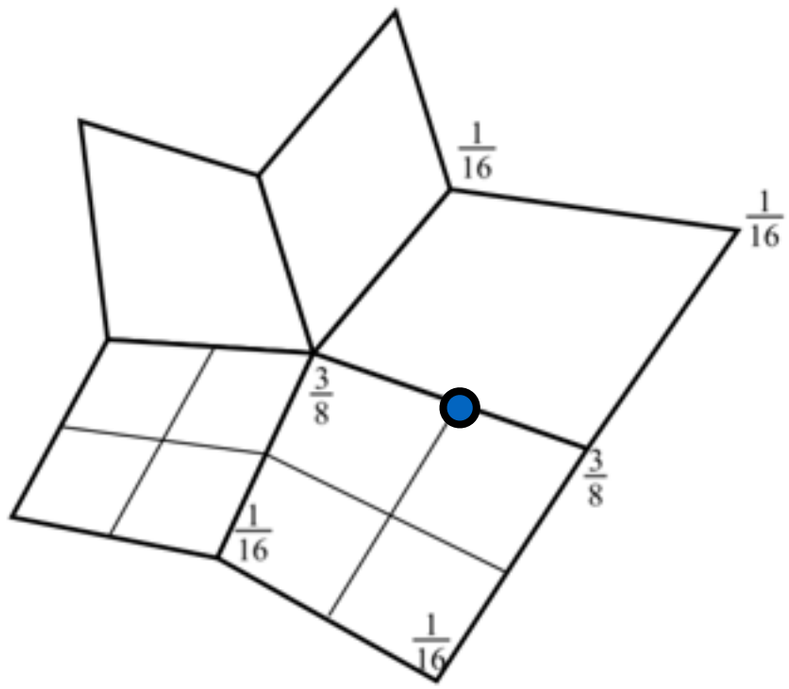
Quad Subdivision

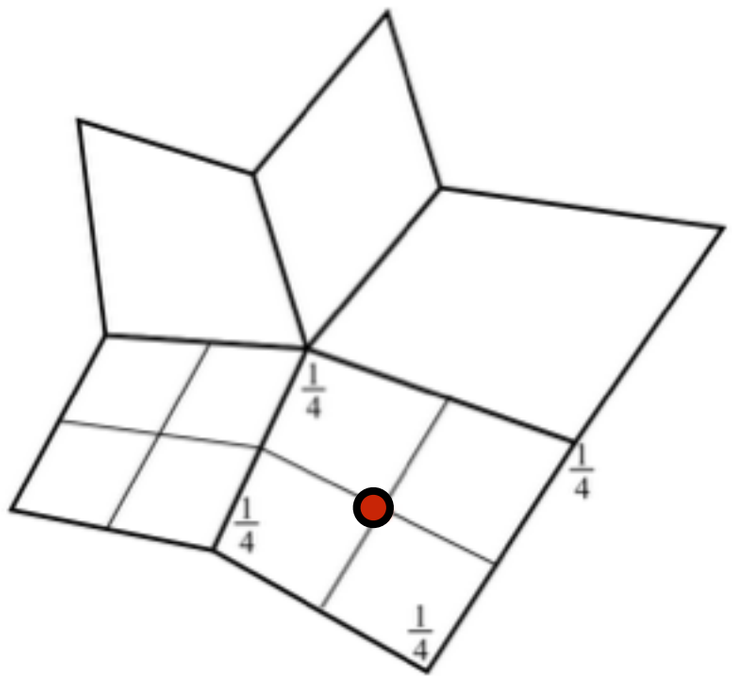


Quad Subdivision

- Three classes
  - Old vertices
  - Midpoints
  - Centroids







Scott Schaefer