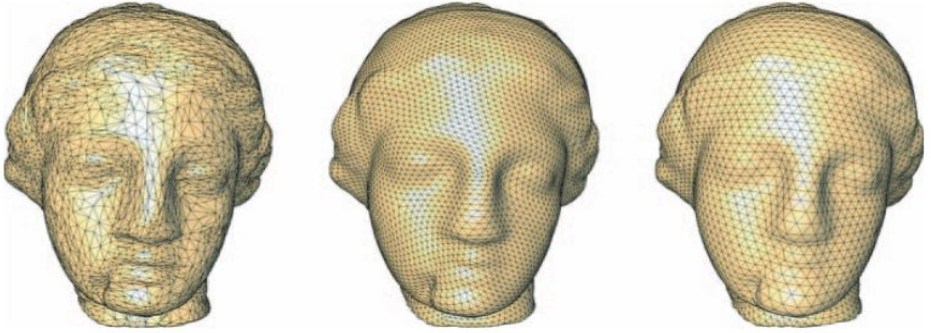
Connectivity Regularization

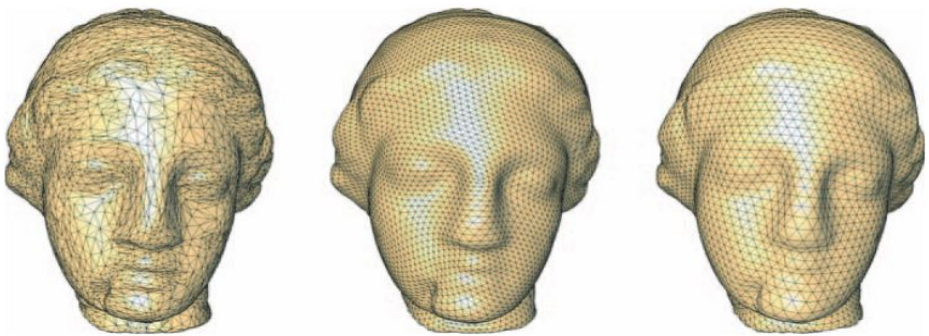
Final Presentation



Regularity of meshes

- Regular meshes: all vertices have a constant number of neighbors

- Simpler Connectivity graph
  → Efficient traversal



Irregular semi-regular regular

Approach

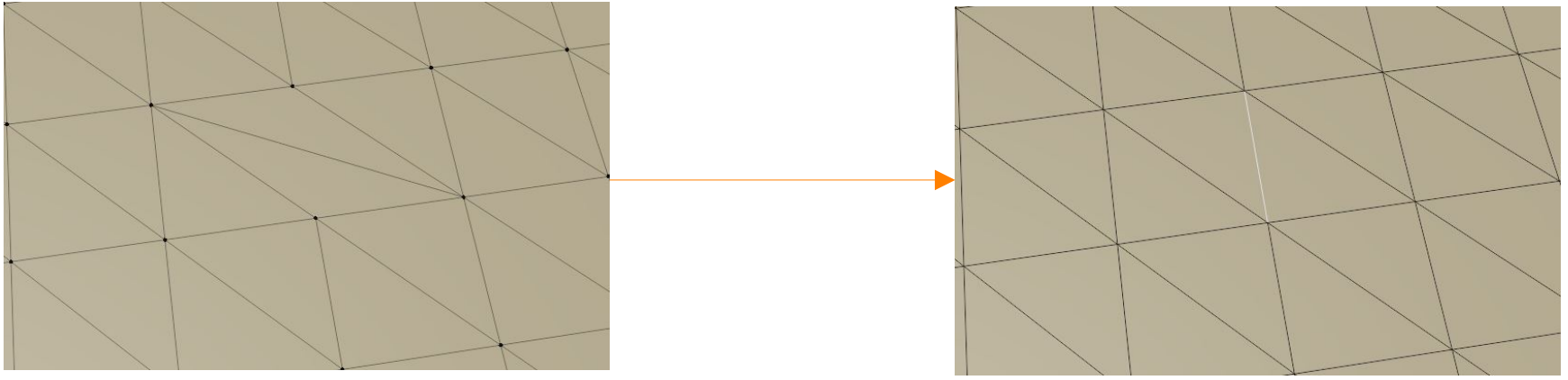
- Local Operations:
  - Edge Flips, Splits and Collapses
  - Angle-based Smoothing

Algorithm

Step 1: Perform all *basic* Edge Flips

- Flip edges, that improve the degree error of the mesh
- Optimal degree of a vertex d(v) is 4 for boundary vertices and 6 for other ones

Improve Vertex degree

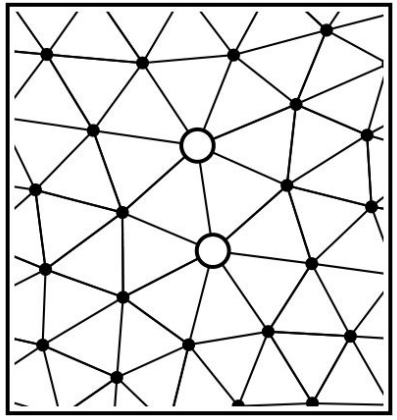


Improve inner angles

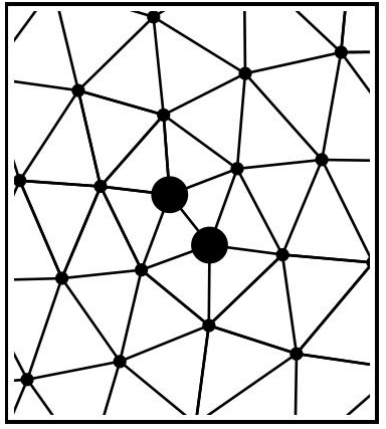
Algorithm

Step 2:

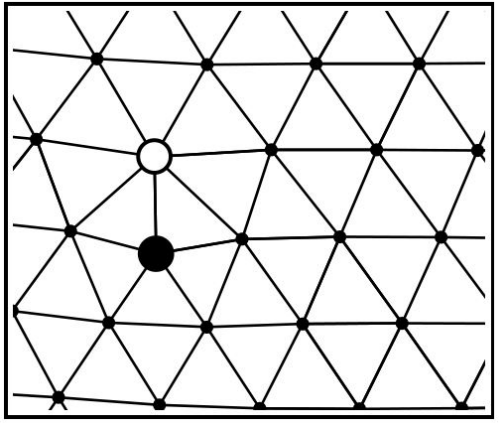
- Categorize edges:



Long



Short

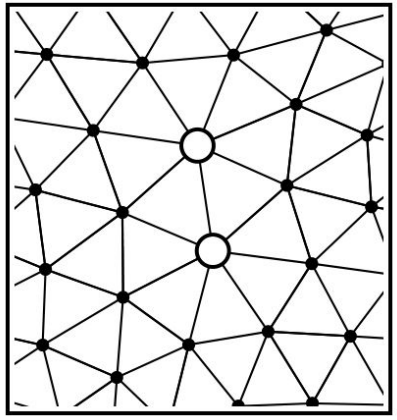


Drifting

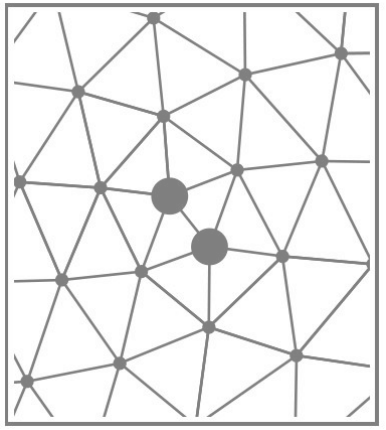
Algorithm

Step 2:

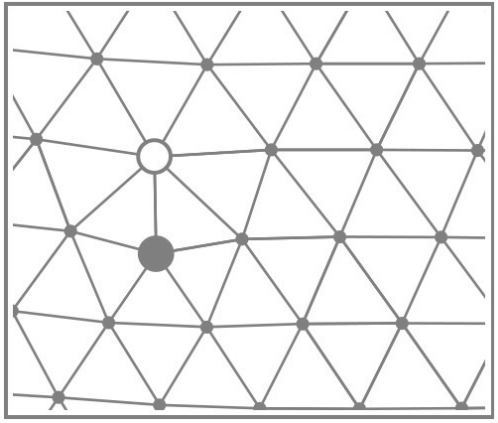
- Long edges: Split and perform basic Edge Flips



Long



Short

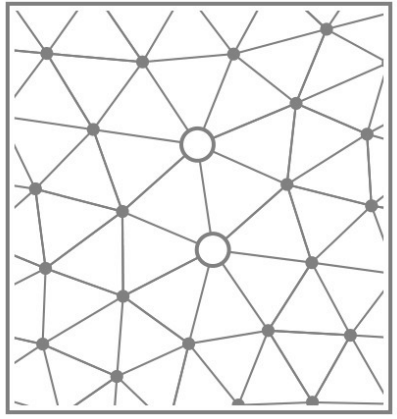


Drifting

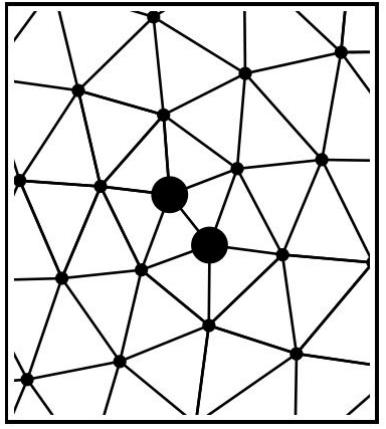
Algorithm

Step 2:

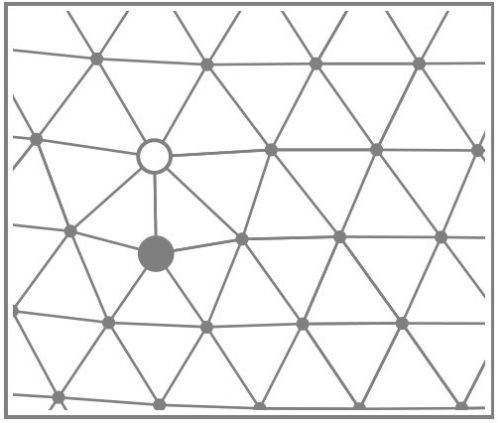
- Short Edges: Collapse and perform basic Edge Flips



Long



Short

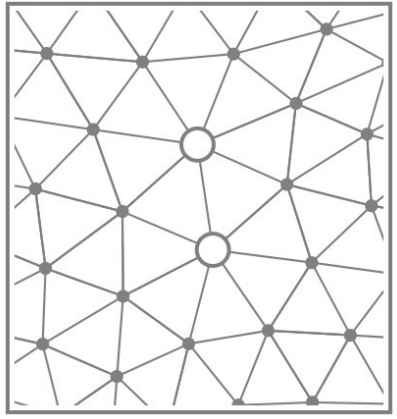


Drifting

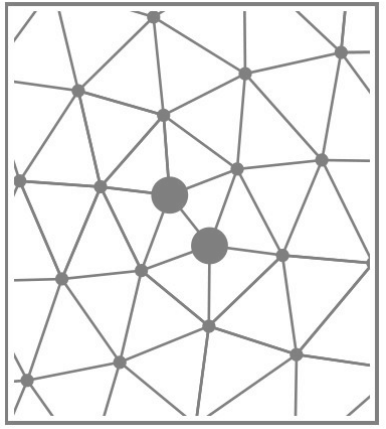
Algorithm

Step 2:

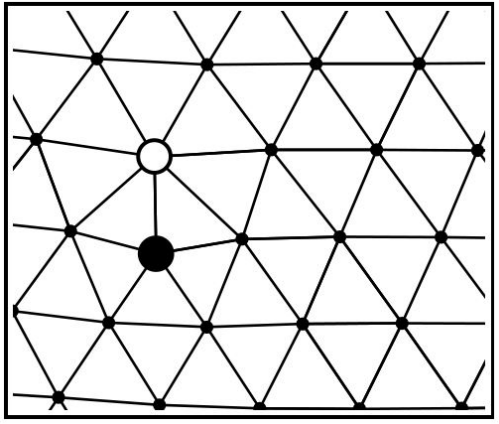
- Drifting edges: move along mesh until they meet another irregular vertex



Long



Short

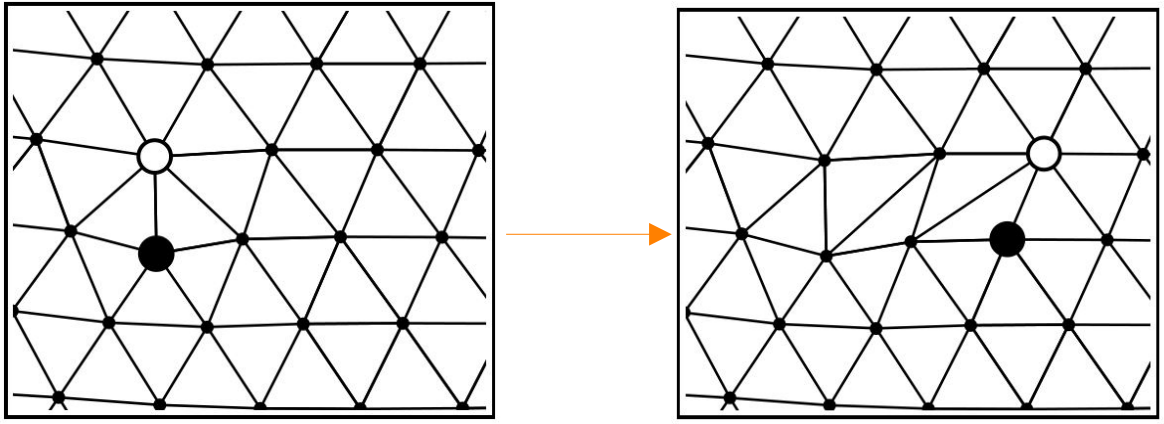


Drifting

Algorithm

Step 2:

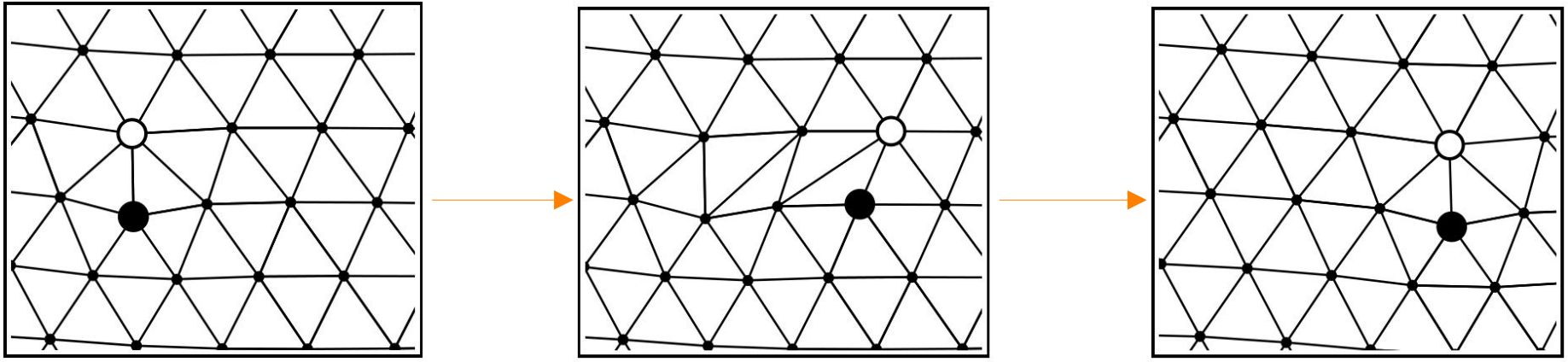
- Drifting edges: move along mesh until they meet another irregular vertex



Algorithm

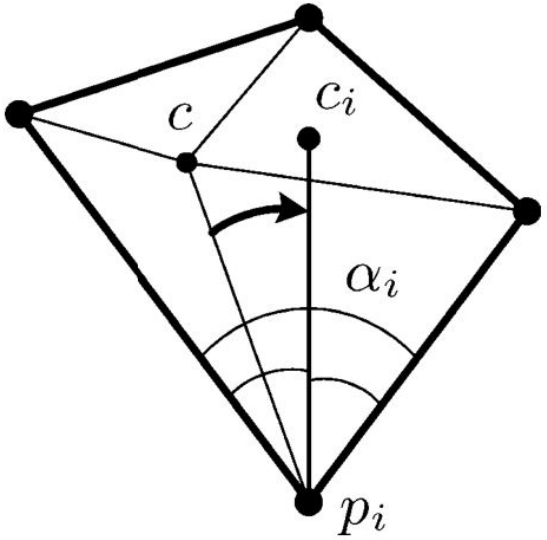
Step 2:

- Apply angle-based smoothing to involved vertices after each operation



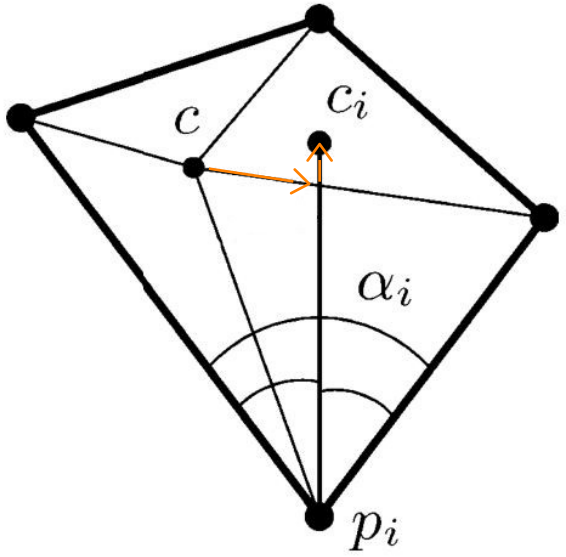
Angle based smoothing

- Calculate new position of vertex c for all neighbor vertices p by rotating edge p → c
- Move c to average of calculated positions



Angle based smoothing

- Adaptation for 3D meshes:
  - Calculate new position by moving c along the other edges
  - Extend to preserve distance



Result

- Few irregular vertices
- Most irregular vertices surrounded by regular ones

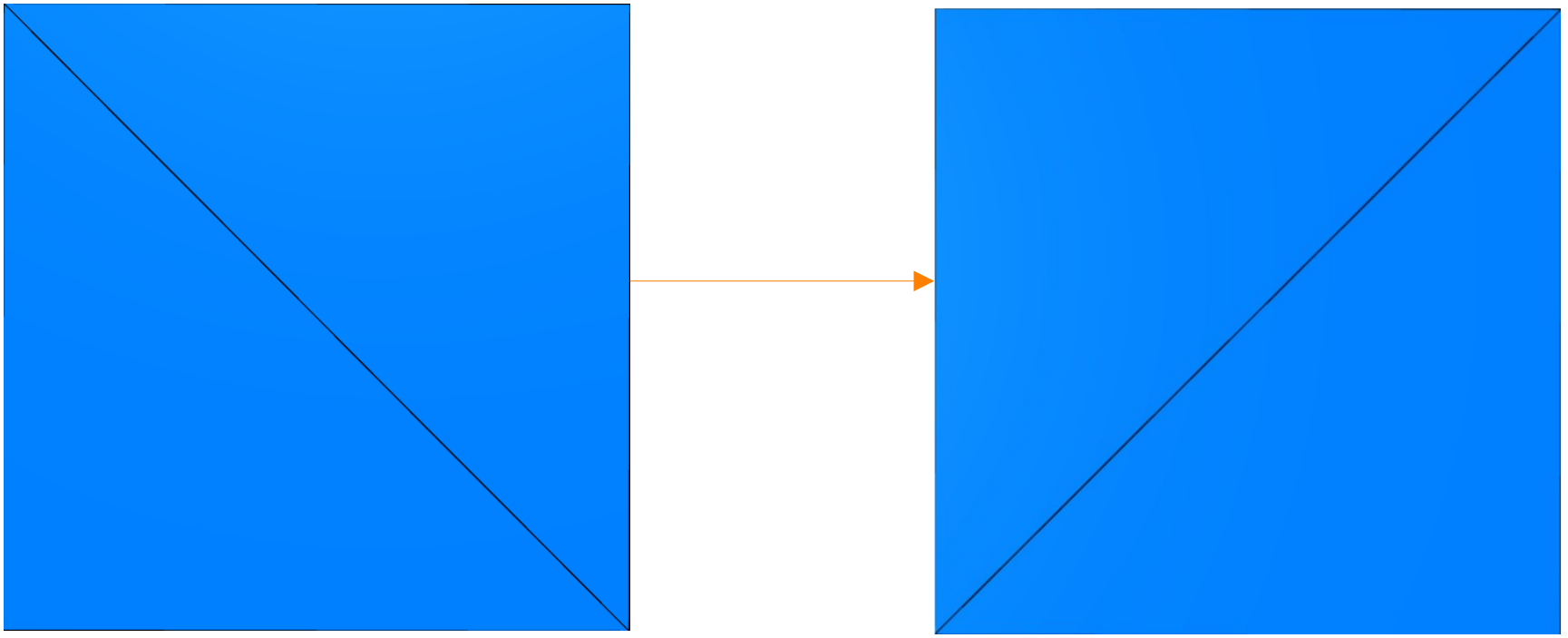
Implementation

- Typescript
- Used course halfedge data structure as basis

Edge Flip

- Lots of pointer reassignments
- Use angles between faces and edges to determine if flip is possible beforehand

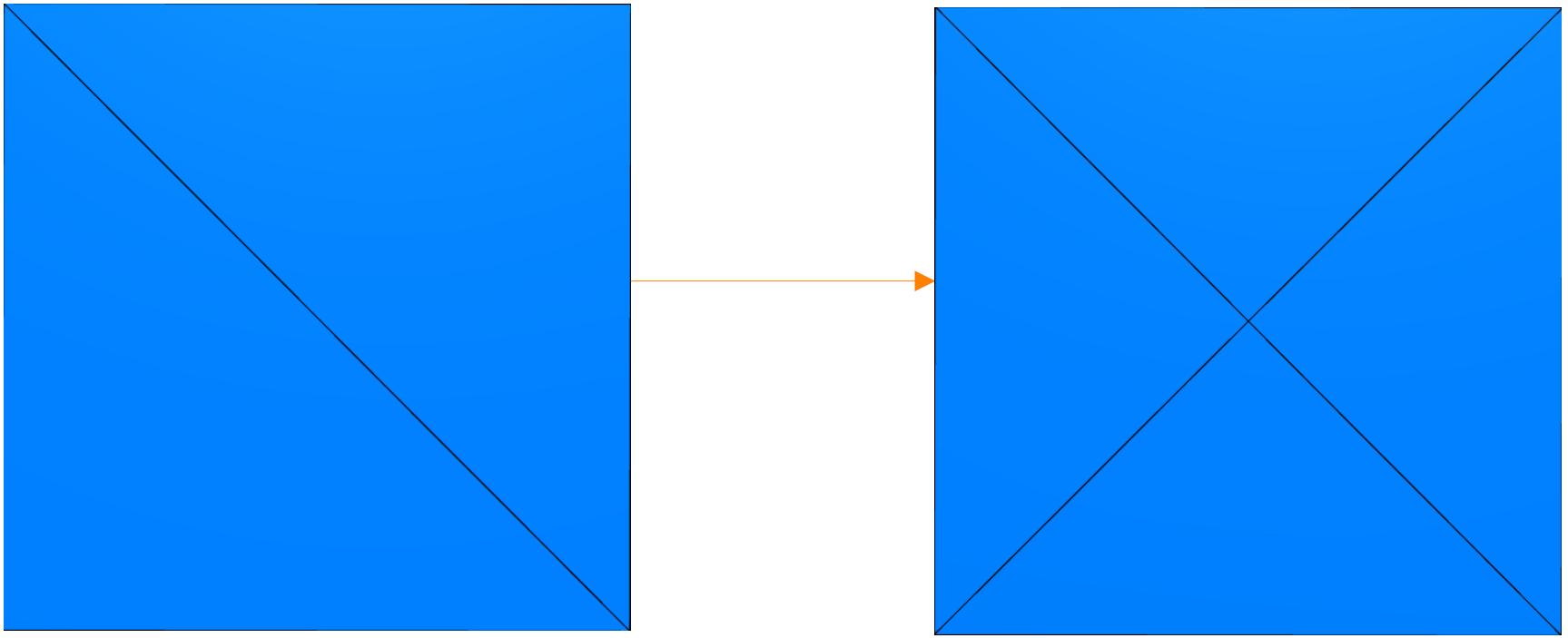
Edge Flip



Edge Split

- Adds:
  - 1 Vertex
  - 6 Halfedges
  - 3 Edges
  - 2 Faces
- Lots of pointer reassignments

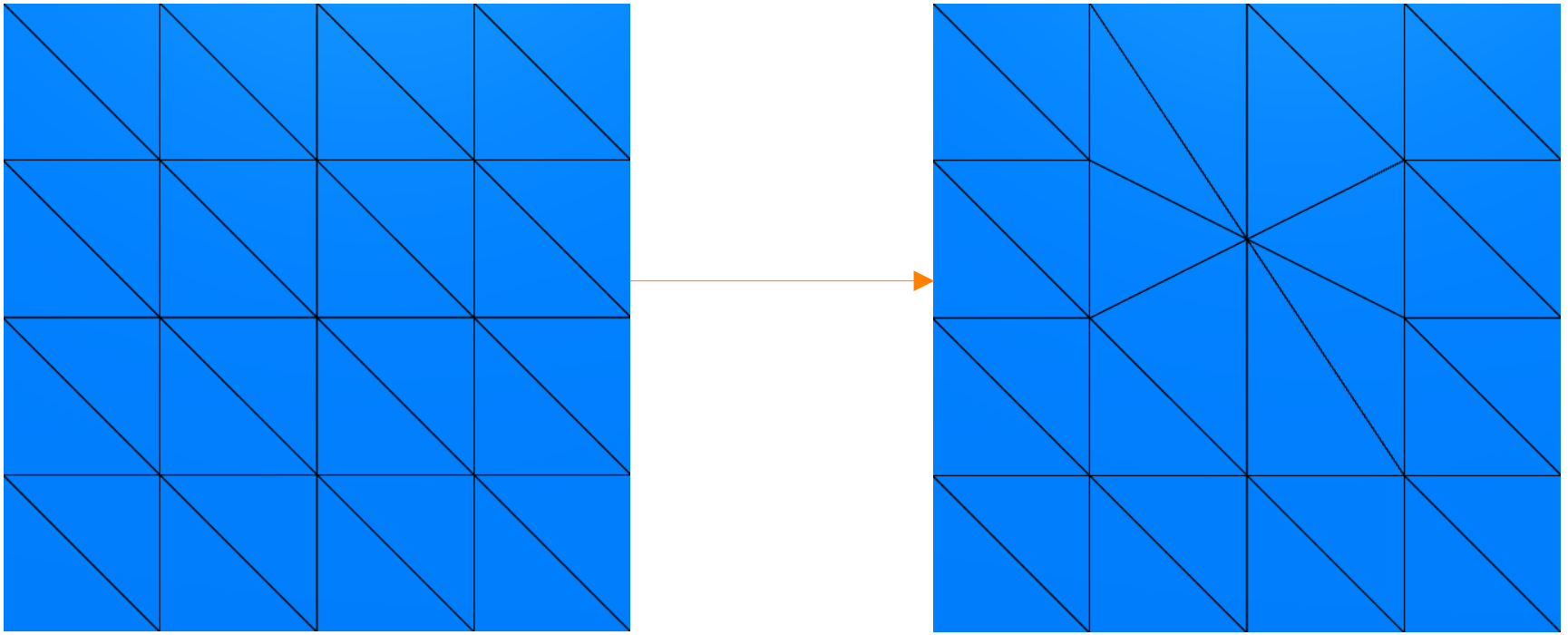
Edge Split



Edge Collapse

- Removes:
  - 1 Vertex
  - 2 Halfedges
  - 1 Edge
  - 2 Faces
- Lots of pointer reassignments

Edge Collapse

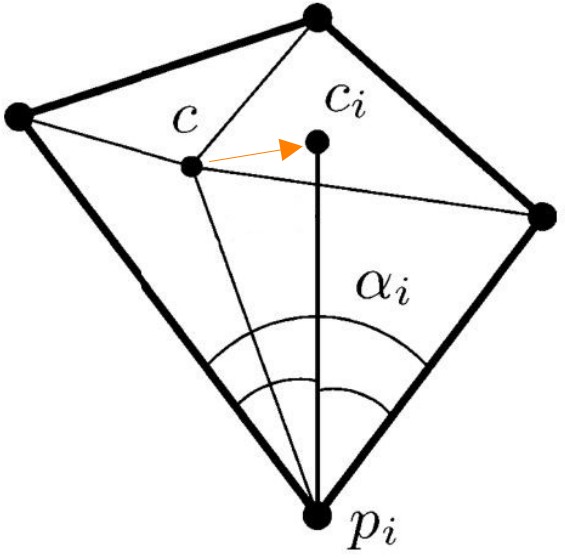


Angle based smoothing

- One function that calculates the new position
- One function to update the position/s
- Parameters:
  - Intensity
  - Rounds

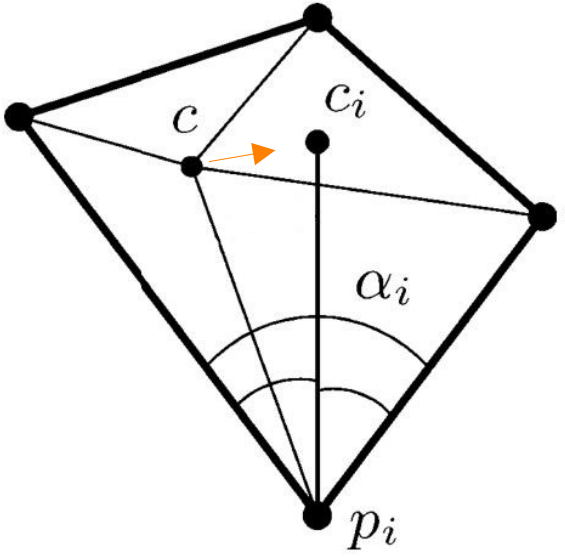
Angle based smoothing

Intensity:

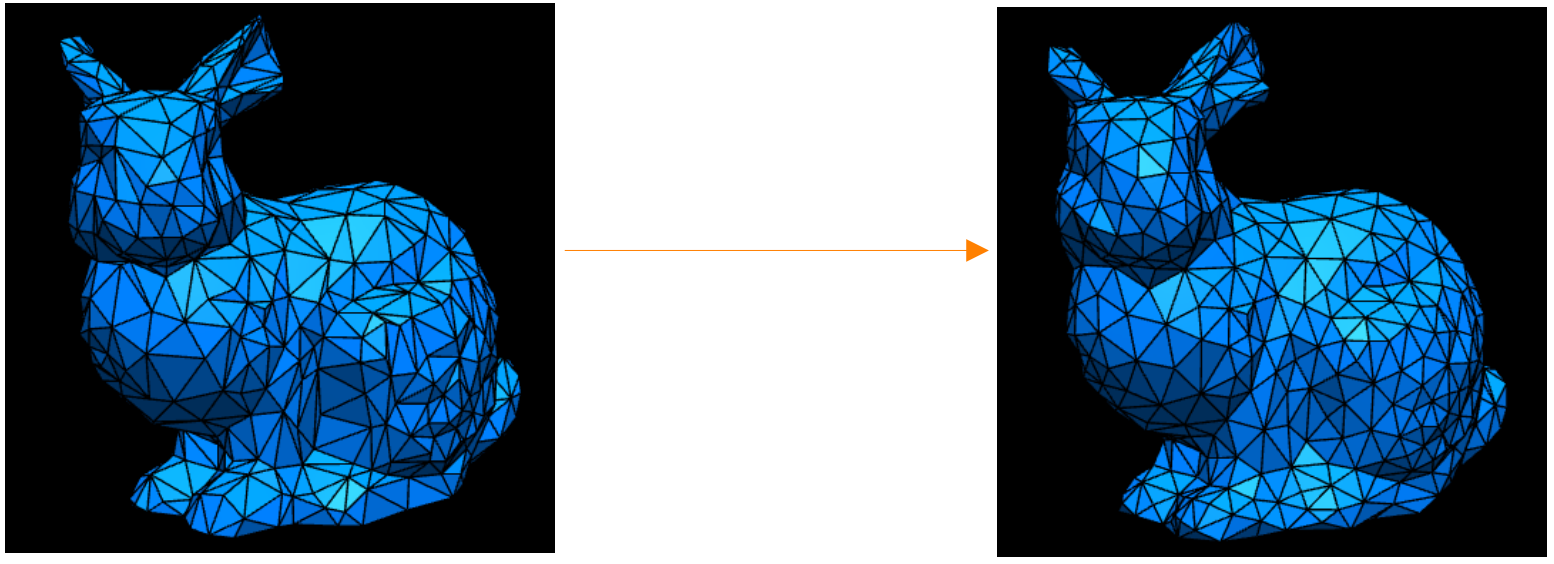


Angle based smoothing

Intensity:



Angle based smoothing



Basic Edge Flips

- Simple loop
- Check degree of vertices before/after flip
- Flip if it improves

Handle Long/Short Edges

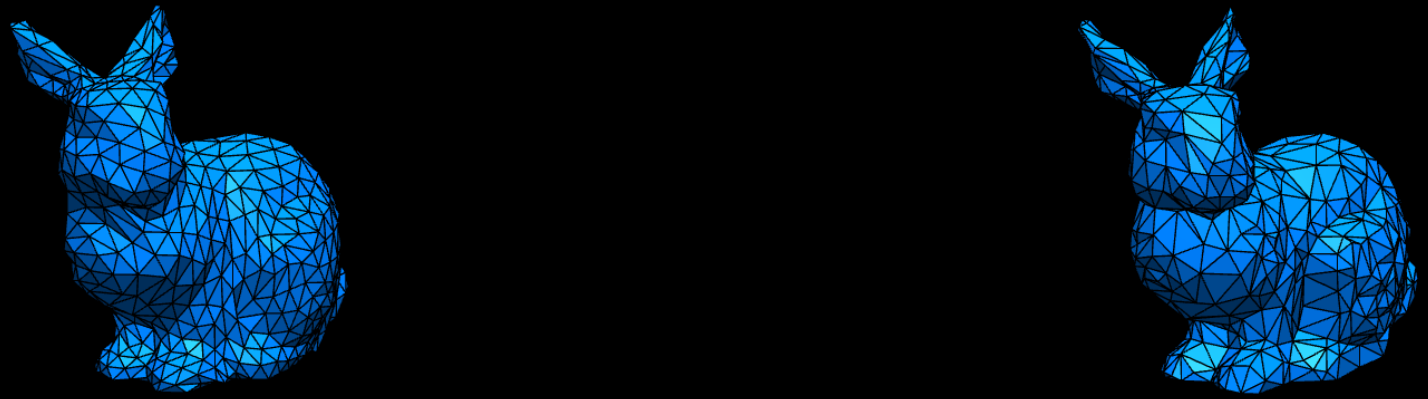
- Simple loop
- Check degree of edge vertices to categorize
- Handle according to scheme

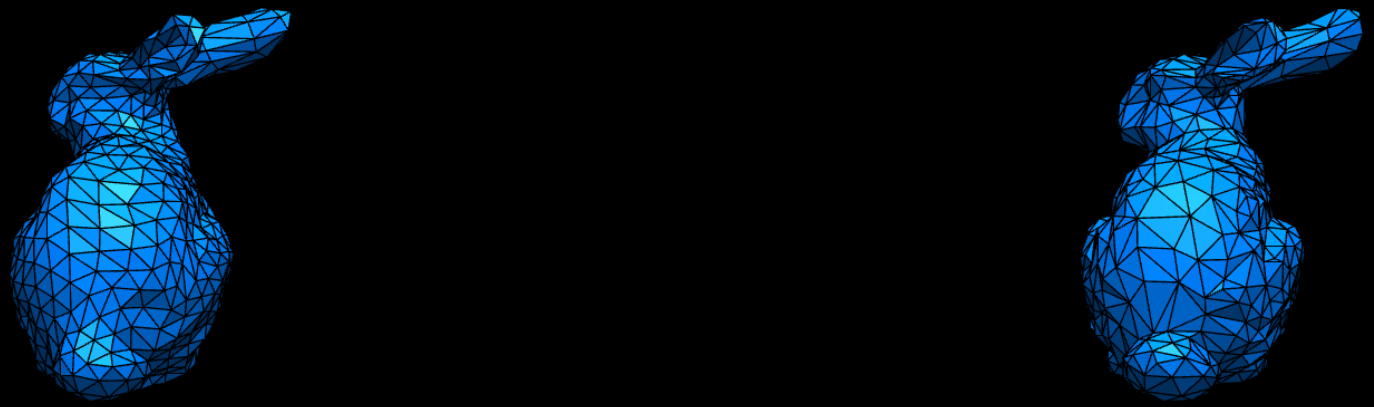
Handle Drifting Edges

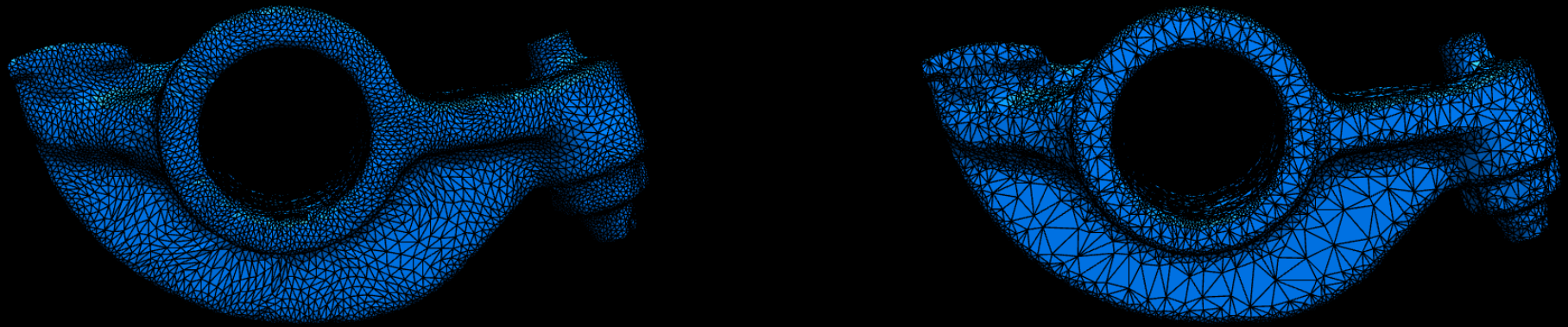
- Check degree of edge vertices to categorize
- Check for both directions to find the nearest irregular vertices
- Check if flips are possible
- Flip and handle result

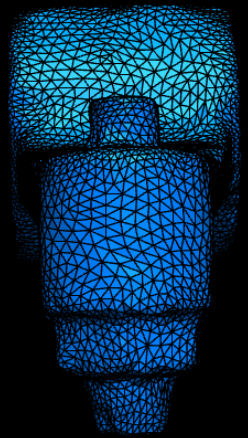
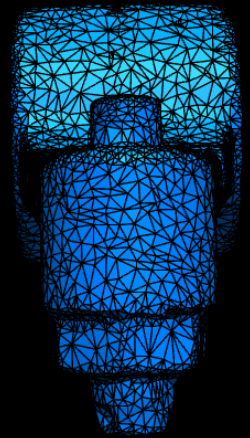
Final State

- Basic edge flips
- Easy long/short edges
- Drifting edges
- Angle based smoothing after edge operations
- (optional smooth steps)







Demo

Challenges

- Edge operations:
  - lots of pointer reassignments
  - preserve correct connectivity
  - hard to debug

Challenges

- Drifting edges:
  - correct traversal of connectivity
  - finding possible flips

Possible Improvements

- Find all possible edge flips and assign priority
- Better shape preservation