A2.5: MeshEdit

Global Ops Edition

“I hate meshes. I cannot believe how hard this is.

Geometry is hard.”

— David Baraff, Senior Research Scientist, Pixar Animation Studios

Halfedge Data Structure

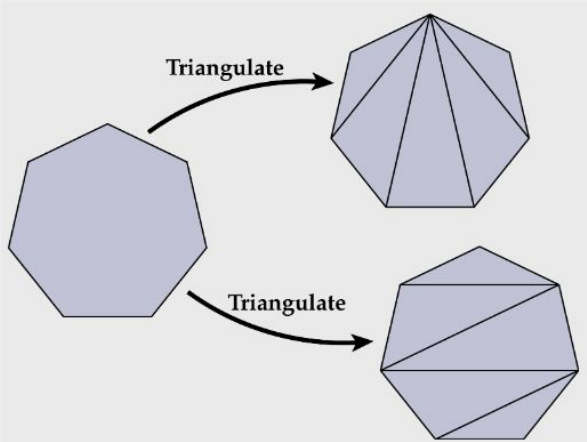
```
class Halfedge {
public:
        //connectivity:
        HalfedgeRef twin; //halfedge on the other side of edge
        HalfedgeRef next; //next halfedge ccw around face
        VertexRef vertex; //vertex this halfedge is leaving
        EdgeRef edge; //edge this halfedge is half of
        FaceRef face; //face this halfedge borders
```

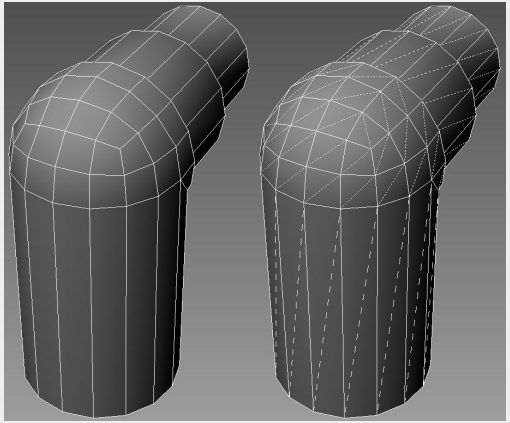
- Triangulation
- Linear Subdivision
- Catmull-Clark Subdivision
- Loop Subdivision
- Isotropic Remeshing
- Simplification

- Triangulation
- Linear Subdivision
- Catmull-Clark Subdivision
- Loop Subdivision
- Isotropic Remeshing
- Simplification

Triangulation

- Goal: split each non-triangular face in mesh into triangles
  - No strict way to triangulate the faces, multiple patterns are acceptable



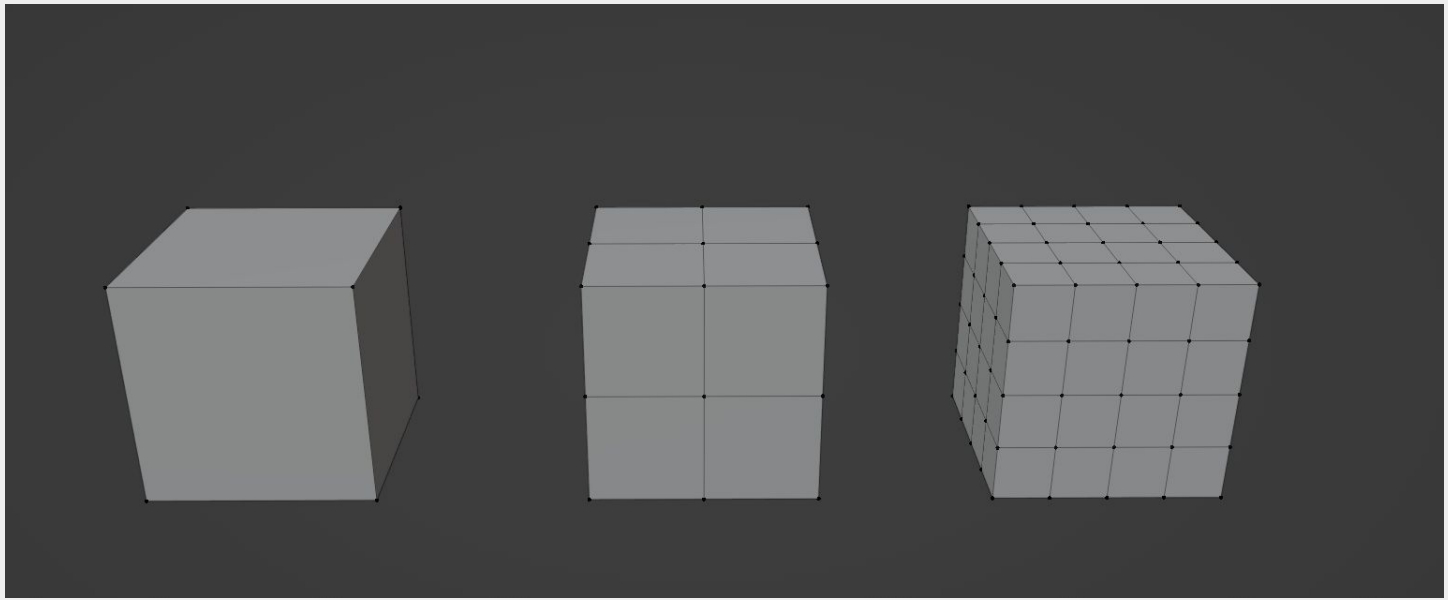


- Most 3D engines (Blender, Unity, Scotty 3D, etc) automatically triangulate meshes before rendering as rasterizing triangles is simpler than rasterizing polygons of higher order
- Triangulated meshes can be used as inputs for other geometric processing algorithms that require tri meshes, like isotropic remeshing

- Triangulation
- Linear Subdivision
- Catmull-Clark Subdivision
- Loop Subdivision
- Isotropic Remeshing
- Simplification

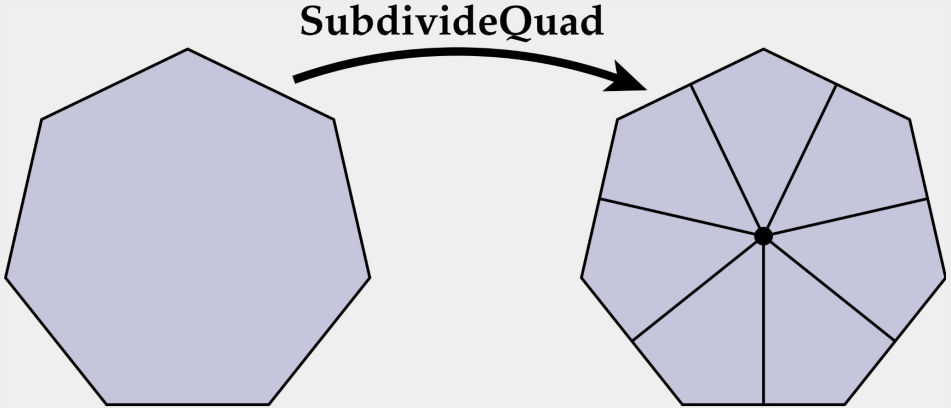
Linear Subdivision

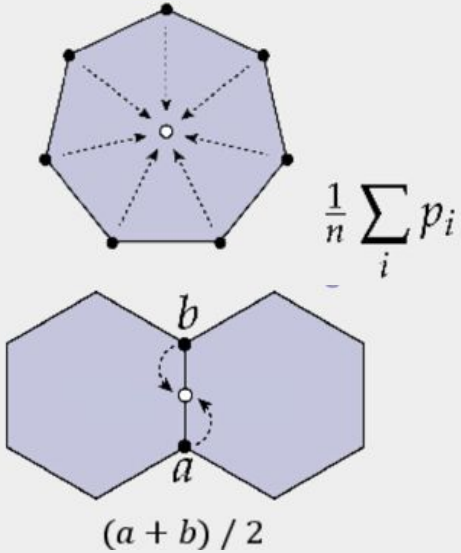
-Goal: Increase fidelity of mesh by splitting existing geometry into quads



Linear Subdivision

- Split each n-gon into quads





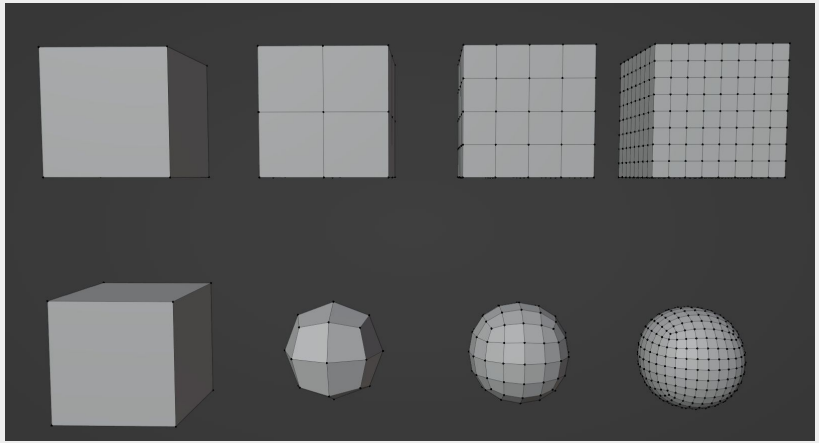
- New central vertex is position is average of vertex positions on the perimeter
- New vertex positions on the edges are on the midpoint of the pre-existing edge

Linear Subdivision

- In contrast to Catmull-Clark, linear subdivision does not modify the vertex positions of the original mesh, resulting in no smoothing

Linear ->

Catmull-Clark->

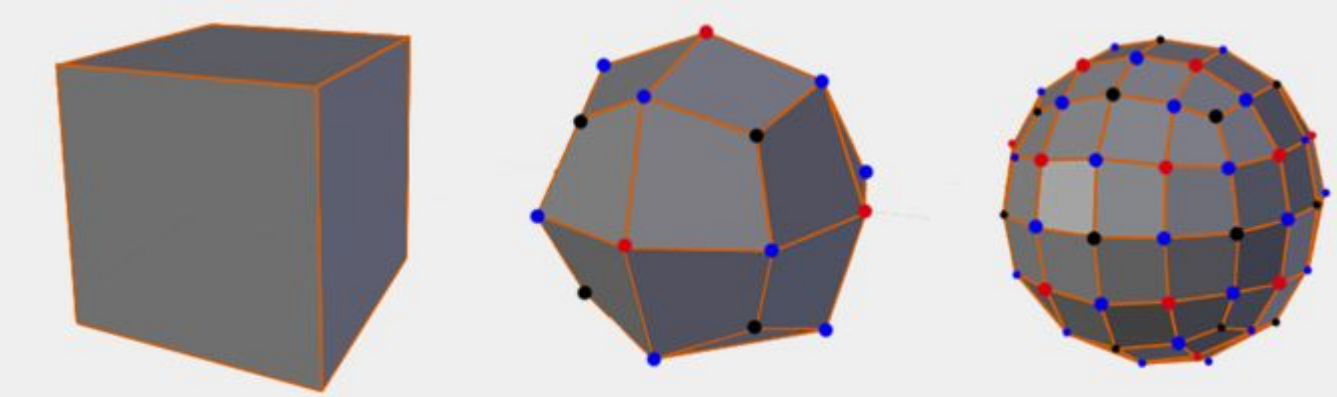


- Useful for when you want to increase fidelity of mesh but preserve its silhouette
    - eg to preprocess mesh before using displacement mapping or procedurally modifying vertex positions with vertex shaders

- Triangulation
- Linear Subdivision
- Catmull-Clark Subdivision
- Loop Subdivision
- Isotropic Remeshing
- Simplification

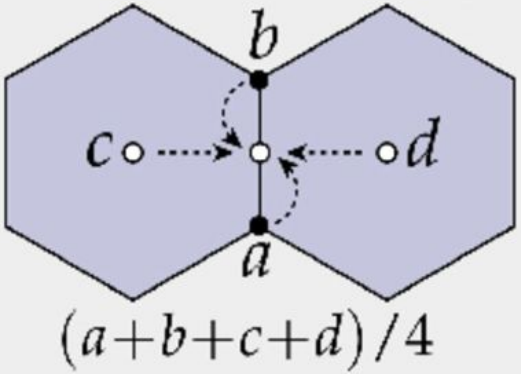
Catmull-Clark Subdivision

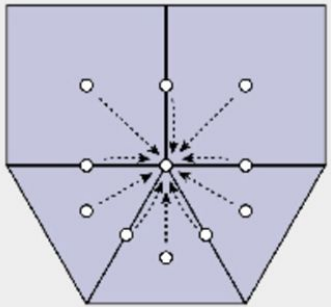
-Goal: Increase fidelity of mesh by splitting existing geometry into quads and smoothing the resulting geometry



Catmull-Clark Subdivision

- Split step is same as linear subdivision
- New edge and vertex coordinates require a bit more math





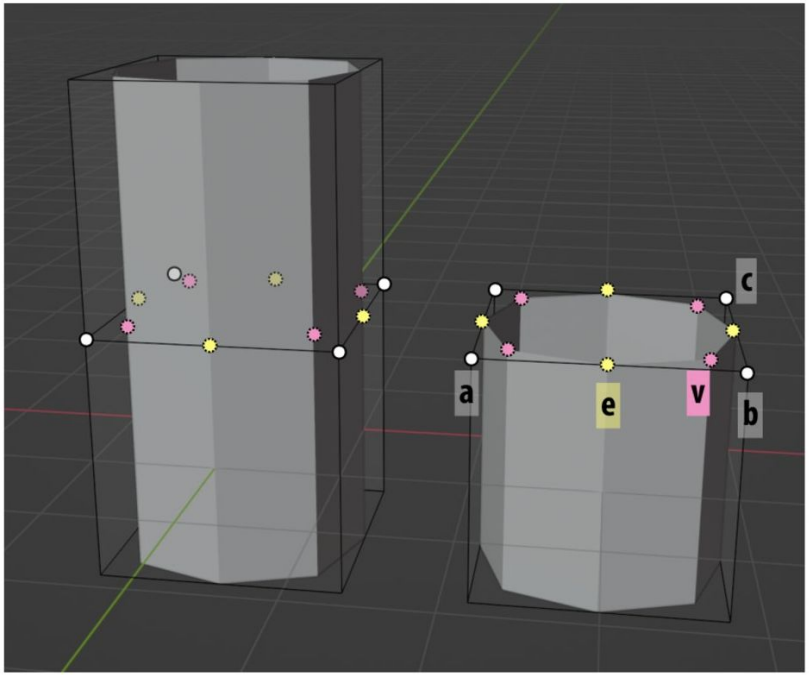
$$\frac{Q + 2R + (n-3)S}{n}$$

Vertex Coords

- n - vertex degree
- Q - average of face coords around vertex
- R - average of edge coords around vertex
- S - original vertex position

Catmull-Clark Subdivision

- Boundary edges



Idea: want the subdivision to "behave the same" if tube gets cut in half

=> edge points: midpoints

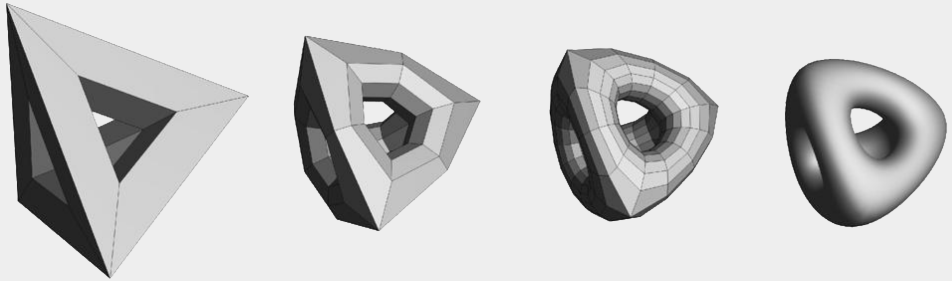
$$e = (a + b) / 2$$

=> vertex positions:

$$v = 1/8a + 1/8c + 3/4\,b$$

Catmull-Clark Subdivision

- Smooths mesh in a appealing way, removes sharpness of low-poly mesh
  - Most effective for more naturalistic meshes, like human characters
  - Nothing in real life is completely sharp – everything has a slight bevel to it
- Oftentimes 3D programs automatically apply multiple Catmull-Clark subdivisions right before rendering
  - So meshes remain computationally cheap and easily malleable when editing, but smooth and high-resolution in the final render



- Triangulation
- Linear Subdivision
- Catmull-Clark Subdivision
- Loop Subdivision
- Isotropic Remeshing
- Simplification

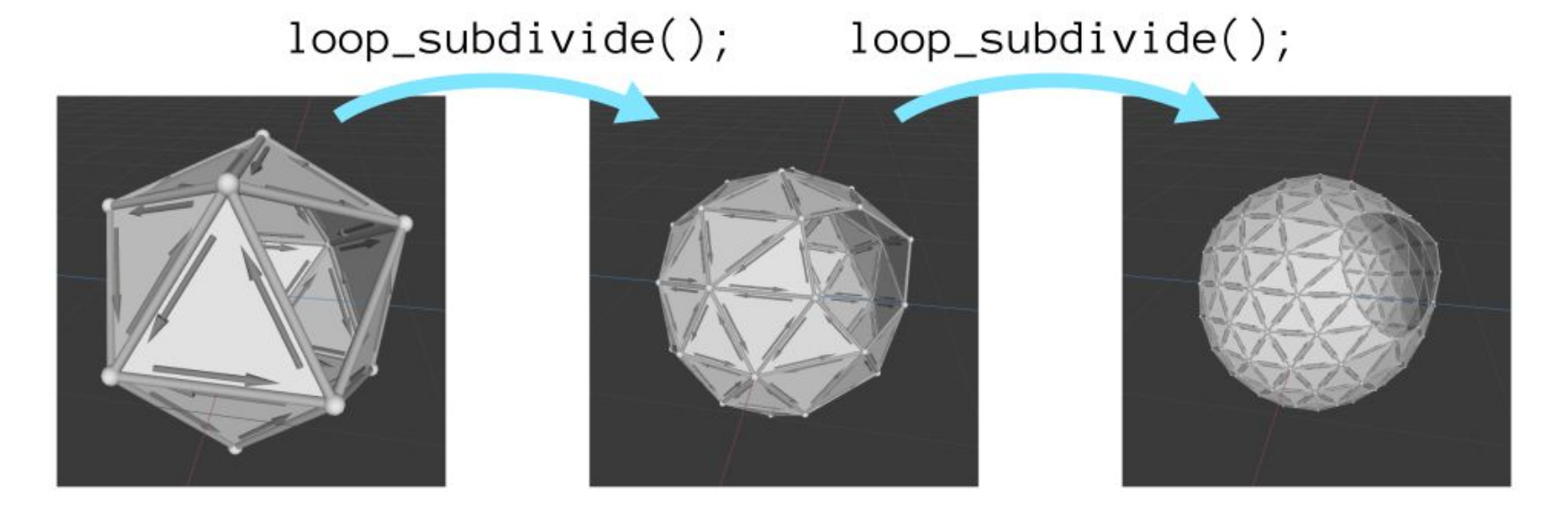
Loop Subdivision

- Catmull-Clark subdivision works quite well, but is there a better subdivision scheme we can use for triangulated meshes?
  - Yes! We can use **loop subdivision**
- We know that all the primitives in a mesh are triangles, so we can just split them into smaller triangles to subdivide the mesh

- How do we achieve this?
  - Split each triangle into 4 smaller triangles
  - Assign coordinates for new vertices
  - Assign coordinates for old vertices

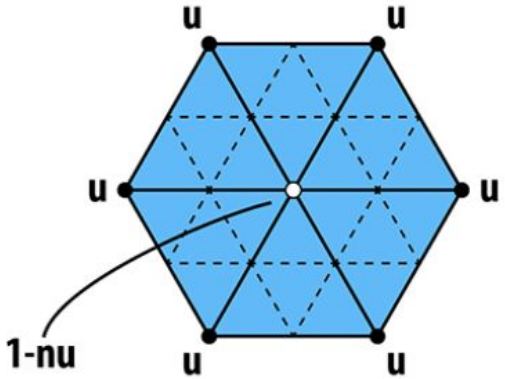
Loop Subdivision

- What does this look like when we actually apply it to a mesh?



Loop Subdivision

- Compute the new positions for every existing vertex in the mesh using the loop subdivision rule
  - Be sure to treat **boundary and interior vertices** appropriately!



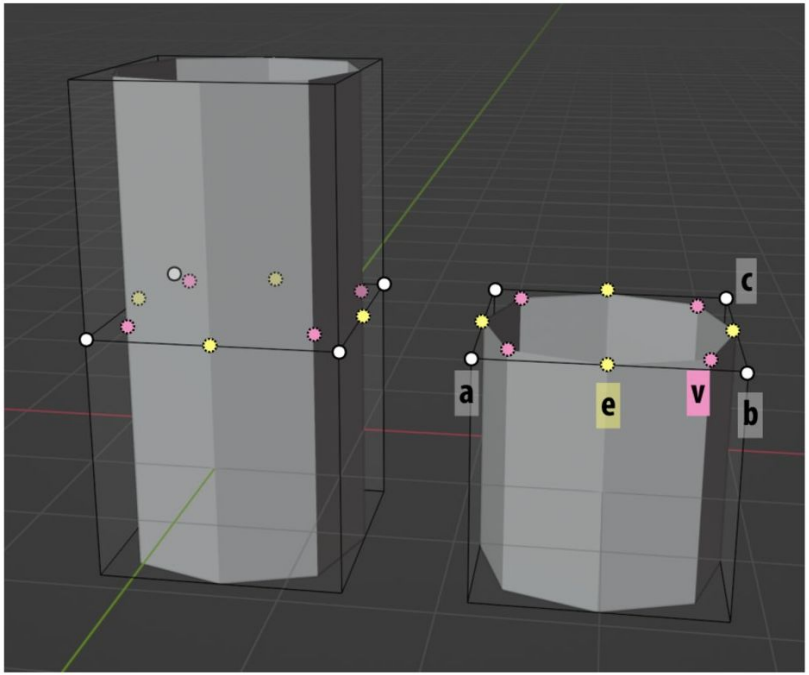
$$v_{new} = (1 - nu) \cdot v_{old} + u \cdot \sum v_{neighbor}$$

n: vertex degree

u: 3/16 if n=3, 3/(8n) otherwise

Loop Subdivision

- Boundary vertices



Idea: want the subdivision to "behave the same" if tube gets cut in half

=> edge points: midpoints

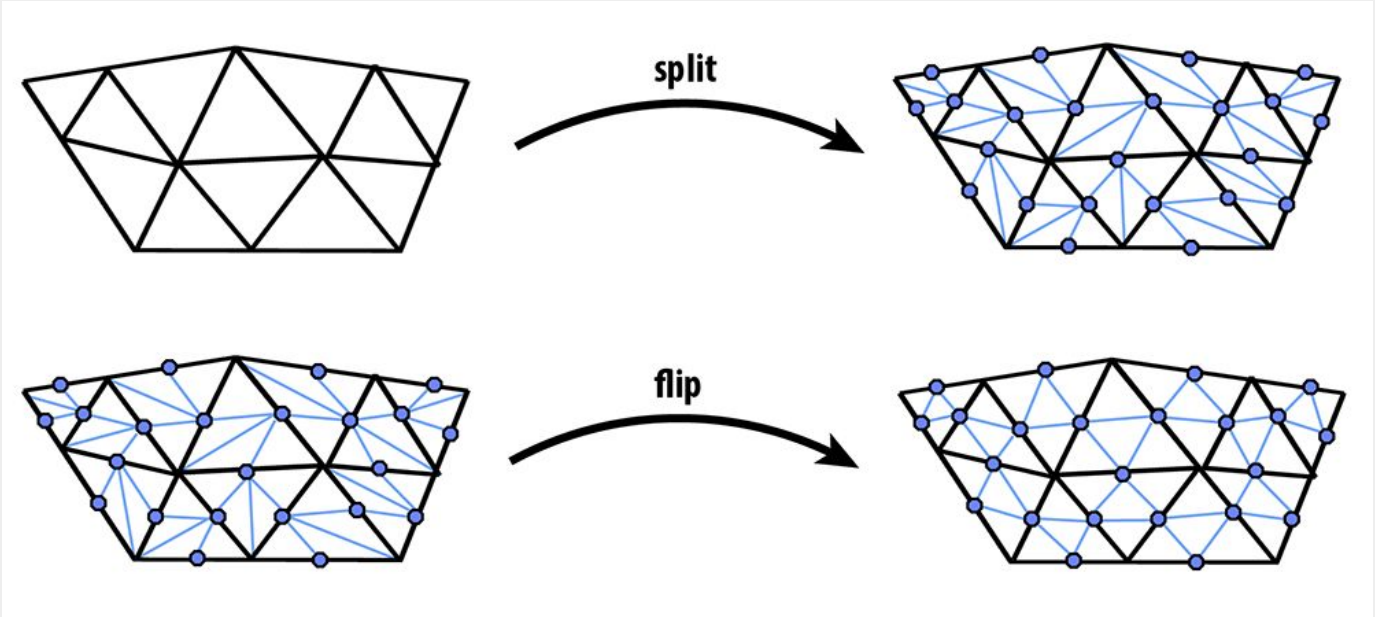
$$e = (a + b) / 2$$

=> vertex positions:

$$v = 1/8a + 1/8c + 3/4\,b$$

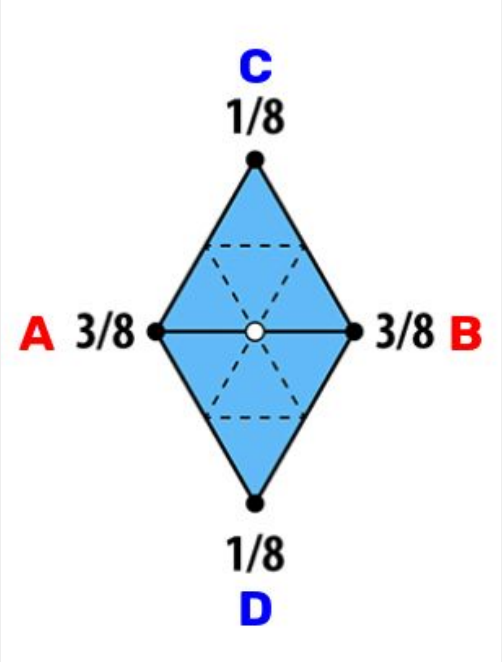
Loop Subdivision

- Split each triangle into 4 smaller triangles
- We can do this using the local ops that we implemented in A2.0
  - Important to keep track of old edges and new edges!



Loop Subdivision

- Compute the new positions for every **new** vertex in the mesh using the loop subdivision rule

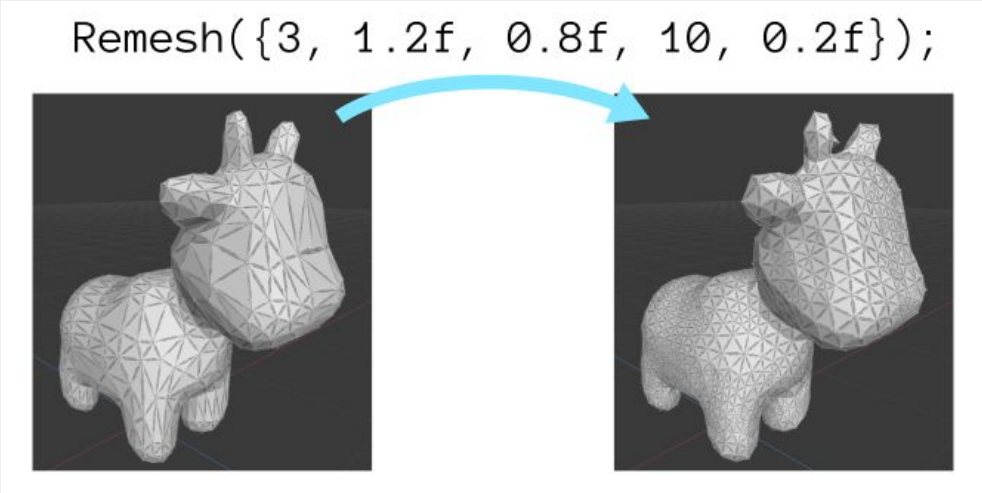


$$v_{new} = \frac{3}{8} \cdot (A + B) + \frac{1}{8} \cdot (C + D)$$

- Triangulation
- Linear Subdivision
- Catmull-Clark Subdivision
- Loop Subdivision
- Isotropic Remeshing
- Simplification

Isotropic Remeshing

- Goal: Produce a mesh with a nicer edge length and angle distribution by repeatedly **splitting edges** that are too long, **collapsing edges** that are too short, and applying **smoothing** (tangent to the surface normal) to vertex positions.



Isotropic Remeshing

- Trying to make mesh more “uniform”
  - i.e. more equilateral triangles, vertex degrees closer to 6, angles close to 60 deg

- How do we achieve this?
  - If an edge is too long, split it.
  - If an edge is too short, collapse it.
  - If flipping an edge improves the degree of neighboring vertices, flip it.
  - Move vertices toward the average of their neighbors.

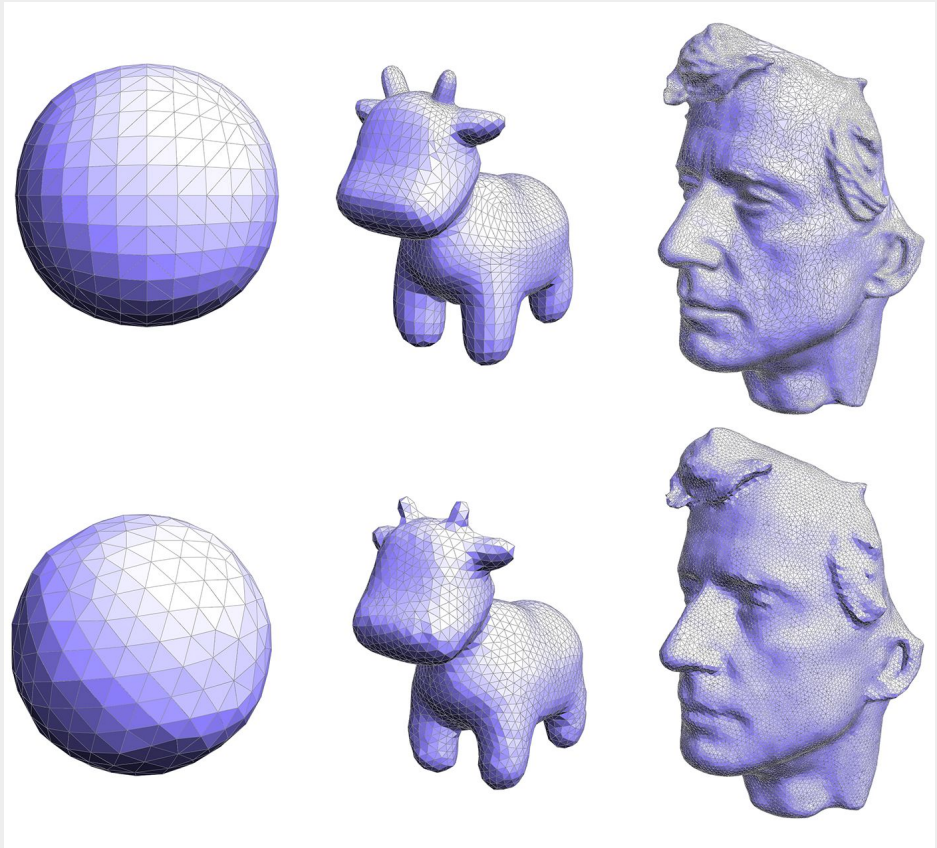
Isotropic Remeshing

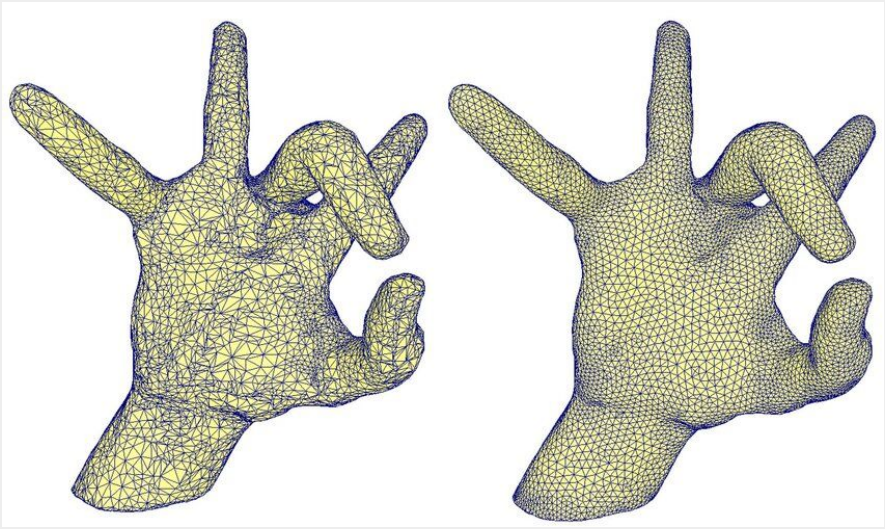
- If an edge is too long, split it. If an edge is too short, collapse it.
  - L = mean edge length
  - Want edges to be 4L/5 < e < 4L/3
  - See `Isotropic_Remesh_Parameters`

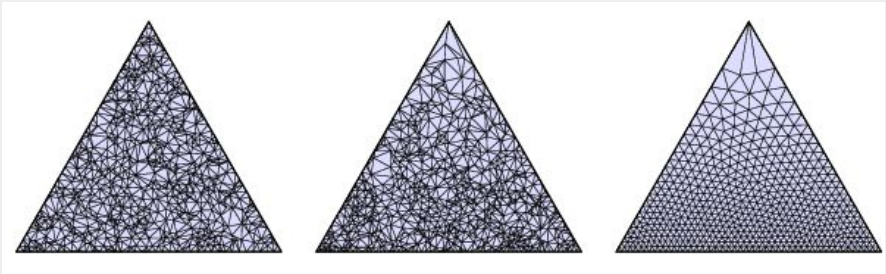
- If flipping an edge improves the degree of neighboring vertices, flip it.
  - Anytime it reduces the total deviation from regular degree (6)
  - Note flipping an edge might destroy it!

- Move vertices toward the average of their neighbors.
  - See `Vertex::neighborhood_center()`
  - First compute new positions, THEN reassign
  - Want to move gently towards center

Isotropic Remeshing



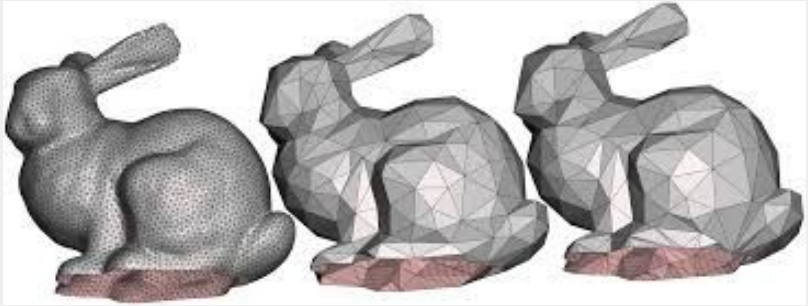


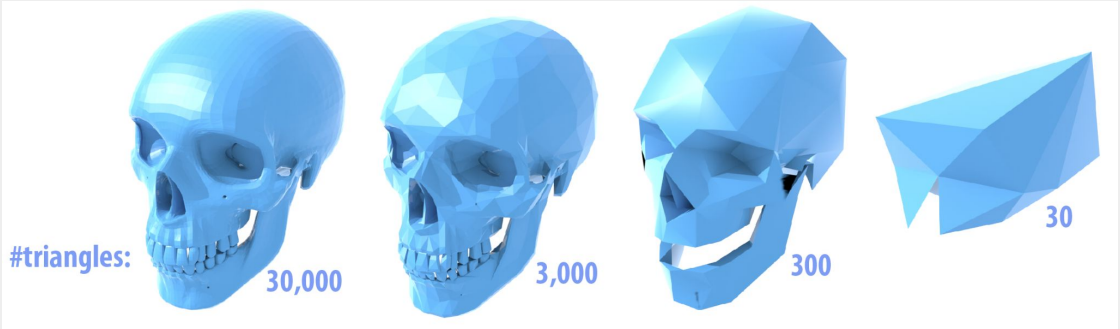


- Triangulation
- Linear Subdivision
- Catmull-Clark Subdivision
- Loop Subdivision
- Isotropic Remeshing
- Simplification

Simplification

- Goal: Reduce the amount of triangles our mesh has while retaining its overall shape





Quadric Error Simplification

- Developed at CMU!
- Iteratively **collapse** edges until we reach the desired number of triangles
- But which edges do we use?

High Level:

- Let’s find edges that, when collapsed, approximate the original mesh as best as possible
- We do this with quadric error matrices!

Quadric Error Simplification

- Quadric Error Matrices:
  - Represent the distance (“error”) from one of the original vertices of the mesh to the edge we are considering collapsing
- We can then take the derivative of this error, set it equal to 0, and find the best vertex position along this collapsed edge that most accurately approximates the original mesh
- Repeat this until we collapse the desired number of edges

- The writeup goes through all the exact math and gives a good guide on how to structure your implementation

Simplification