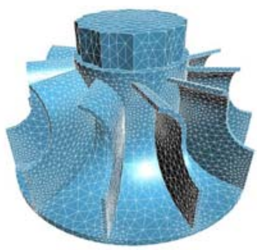
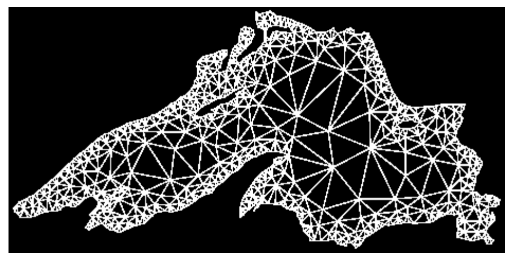
University of
British Columbia

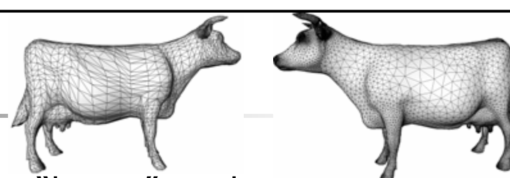
Mesh Generation

Introduction & Meshing in 2D

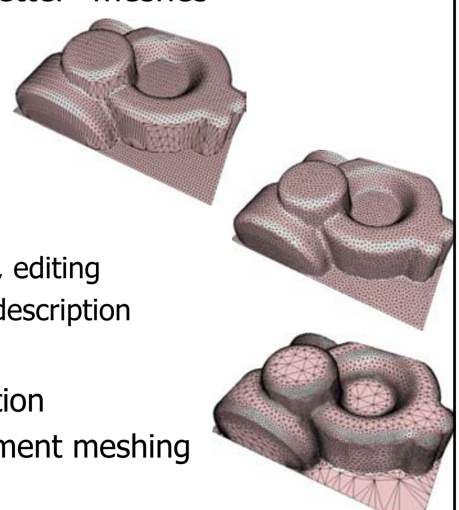




Motivation

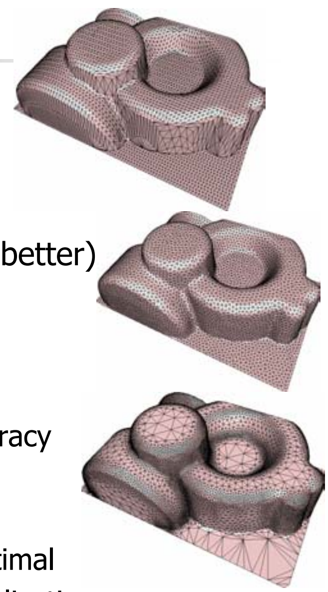


- Remeshing = creating "better" meshes
  - Element sizing
  - Element shape
  - Approximation quality
- Applications
  - Computer Graphics
    - Applications: rendering, editing
    - Compact but accurate description
  - Engineering
    - CAD – model description
    - ***Analysis –*** finite element meshing



University of
British Columbia

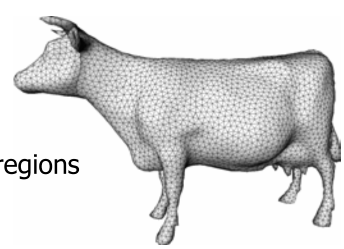
Mesh Properties



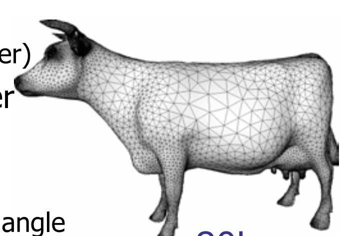
- Depend on application
- Universal
  - Approximation of data
  - Number of elements (less is better)
- Application dependant
  - Element size
    - Graphics – does not matter
    - FEM – defines solution accuracy (depends on equation)
  - Element shape (geometry)
    - Equilateral elements are optimal
    - How crucial depends on application

University of British Columbia

Mesh Properties



5k



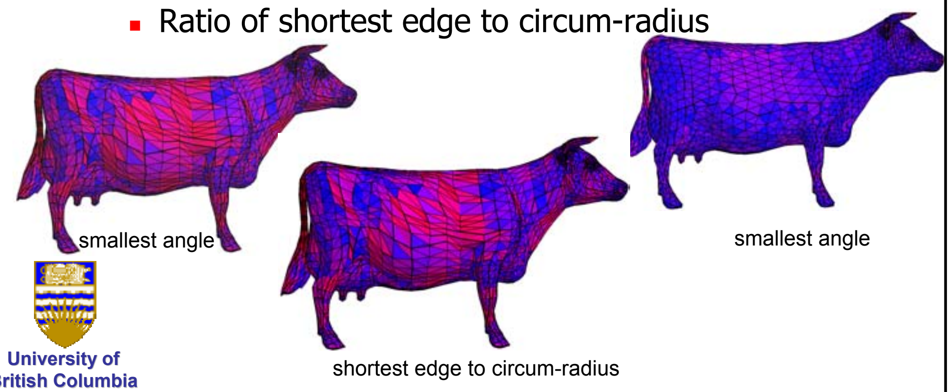
20k

- Approximation of data
  - 2D – accurate approximation of boundary
    - shorter edges in more curved regions
  - 3D – accurate approximation of surface
    - Smaller elements in more curved regions
    - Edges close to surface (more later)
- Sizing function – provided by user
  - In 2D –edge length at any (u,v)
  - 3D –value per vertex
    - barycentric coordinates inside triangle

University of British Columbia

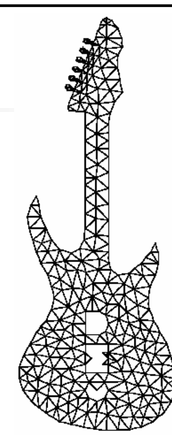
Quality

- Measure "closeness" to equilateral triangle
- Triangle quality measures
  - Ratio of in-radius to circum-radius
  - Smallest angle
  - Ratio of shortest edge to circum-radius



University of British Columbia

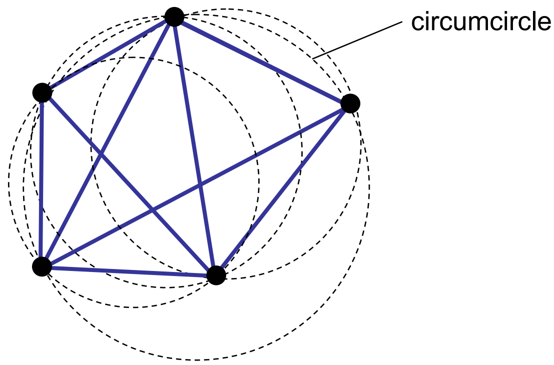
Meshing in 2D



- Input:
  - Planar domain (polygon)
  - Optimal sizing
    - Gradation + budget (e.g. #vertices)
    - Sizing function at each (u,v)
- Output: triangular mesh
- Motivation
  - 2D problems
  - 3D problems reduced to 2D (parameterization)
- Problem components
  - Vertex placement
  - Connectivity construction: Delaunay criterion

University of British Columbia

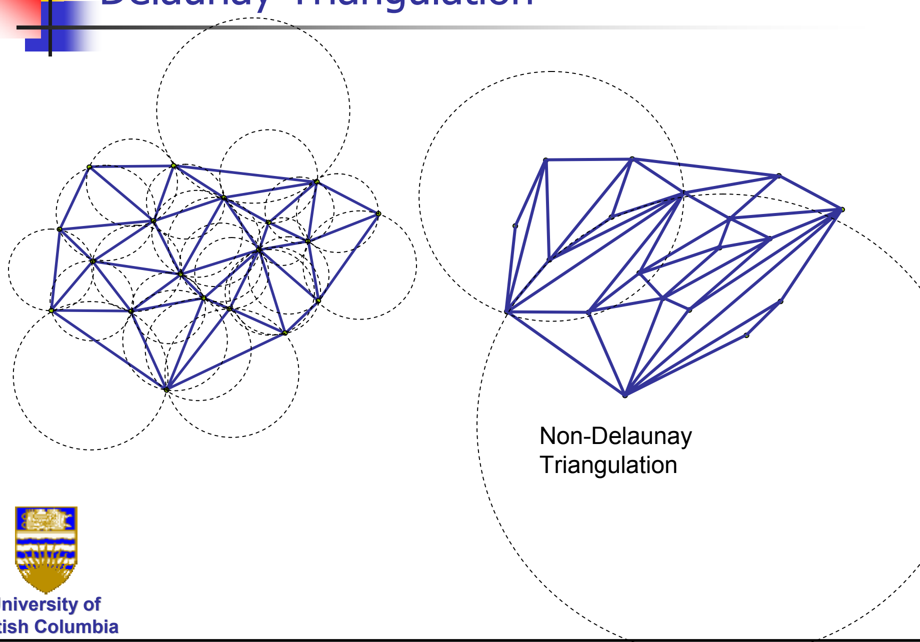
Constructing Connectivity
Delaunay Criterion



Empty Circle Property:
No other vertex is contained within the circumcircle of any triangle

University of
British Columbia

Delaunay Triangulation



University of
British Columbia

Delaunay Triangulation

- Obeys empty-circle property
- Exists for any set of vertices
- Is **unique** (up to degenerate cases)
- Proven to provide best triangles in terms of quality for given vertex positions
- To test – enough to check pairs of triangles sharing common edge

University of British Columbia

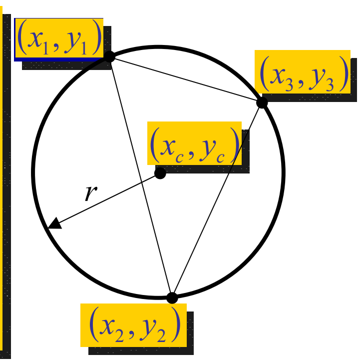
Delaunay - Practicalities

- Computing circumcircle center & radius

$$Ax + By + C = x^2 + y^2$$

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \begin{Bmatrix} A \\ B \\ C \end{Bmatrix} = \begin{bmatrix} x_1^2 + y_1^2 \\ x_2^2 + y_2^2 \\ x_3^2 + y_3^2 \end{bmatrix}$$

$$x_c = \frac{A}{2}, y_c = \frac{B}{2}, r = \sqrt{C + x_c^2 + y_c^2}$$



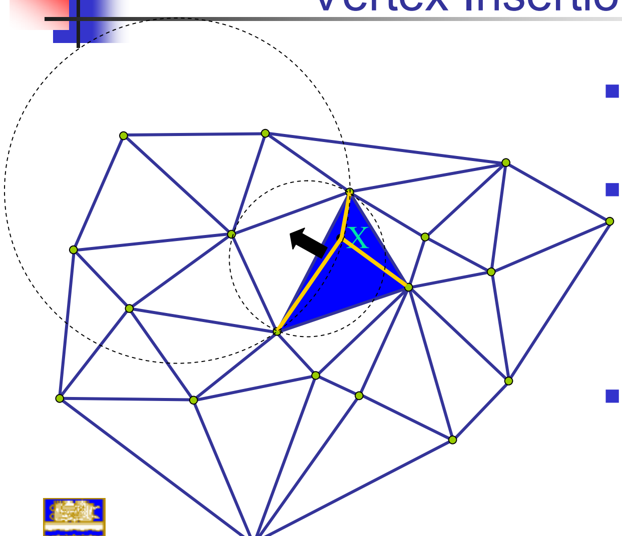
University of British Columbia

Triangulation Methods

- Edge flip algorithm
  - Start with any triangulation of the vertices
  - Test all edges if satisfy Delaunay criterion
    - test triangles on both sides of edge
  - If edge does not satisfy it, flip edge
  - Repeat until all edges satisfy criterion
- Proven to terminate & give Delaunay mesh
- Slow $O(n^2)$
- Alternative – additive construction
  - Keep Delaunay mesh
  - Add one vertex at a time

University of British Columbia

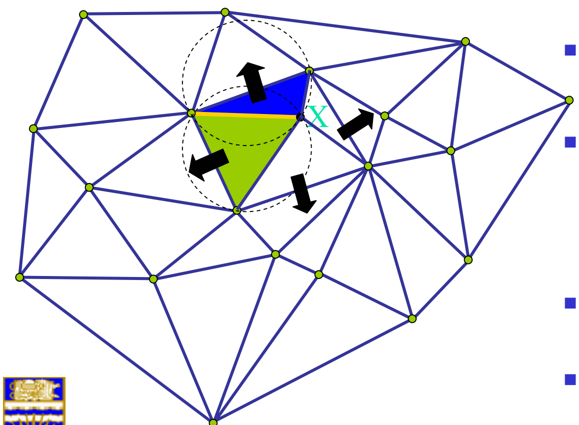
Vertex Insertion



- Locate triangle containing X
- Subdivide triangle
- Recursively check adjoining triangles to ensure empty-circle property

University of British Columbia

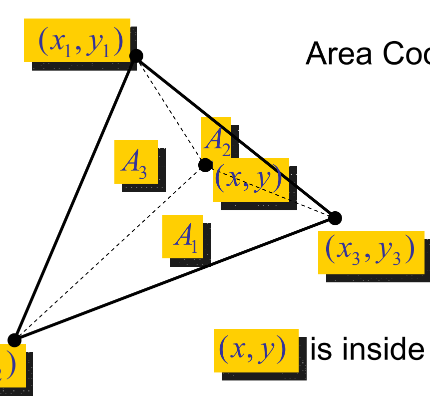
Vertex Insertion



- Locate triangle containing X
- Subdivide triangle
- Recursively check adjoining triangles to ensure empty-circle property
- Swap diagonal if needed
- Expected time O(nlog(n))

Practicalities- Find Triangle

- Search through mesh
- Test inside:



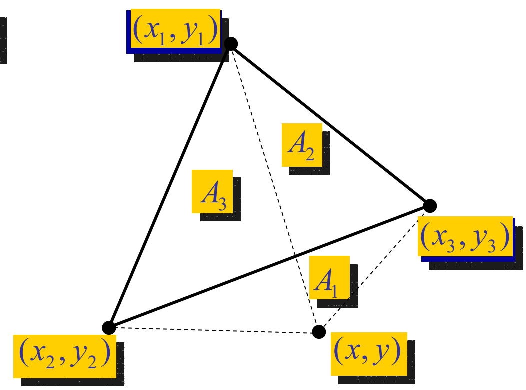
Area Coordinates (A_1, A_2, A_3)

$$A_i = y_{i+2}(x_{i+1} - x) + y_{i+1}(x - x_{i+2}) + y(x_{i+2} - x_{i+1})$$

(x, y) is inside triangle if $A_i > 0$

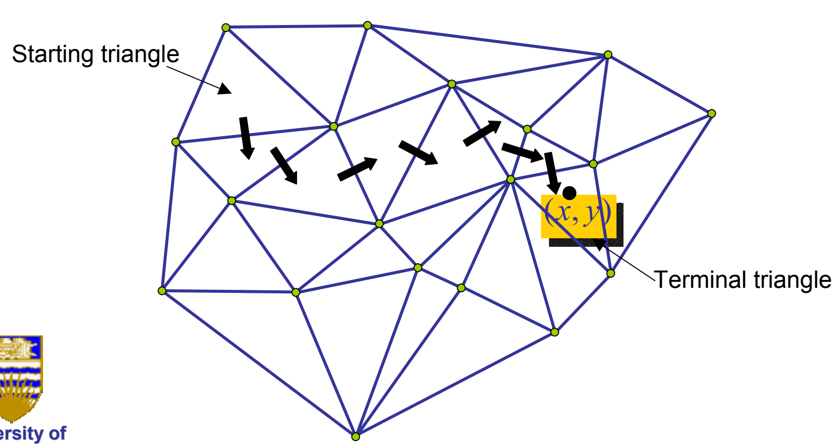
Find Triangle

If $A_i < 0$
then move to adjacent
triangle at edge
$(x_{i+1}, y_{i+1}), (x_{i+2}, y_{i+2})$



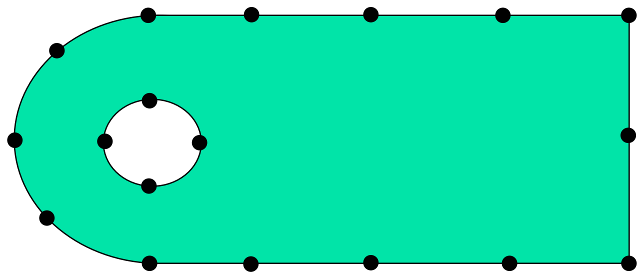
University of
British Columbia

Find Triangle



University of
British Columbia

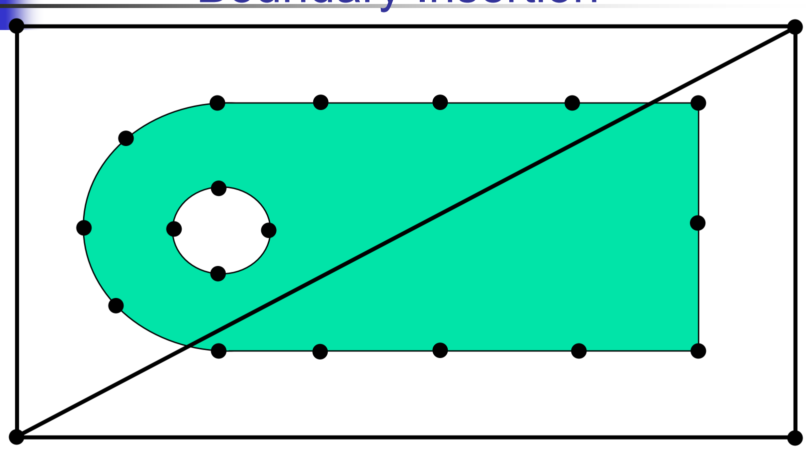
Example: Boundary Insertion



- Place vertices on boundary at cord-length intervals based on sizing

University of British Columbia

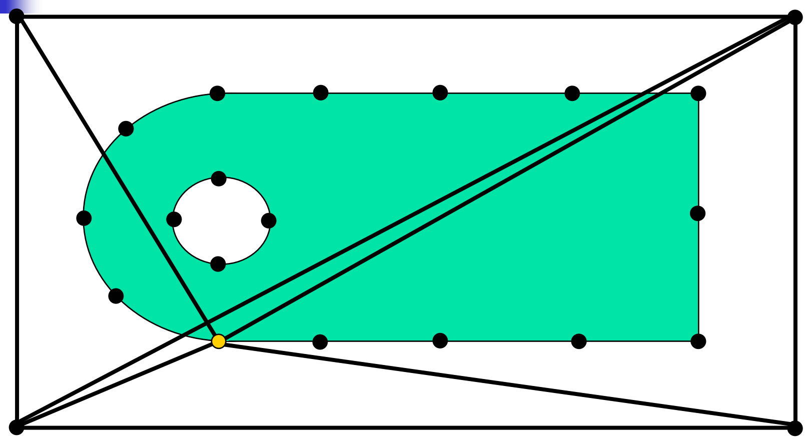
Boundary Insertion



- Create bounding triangles

University of British Columbia

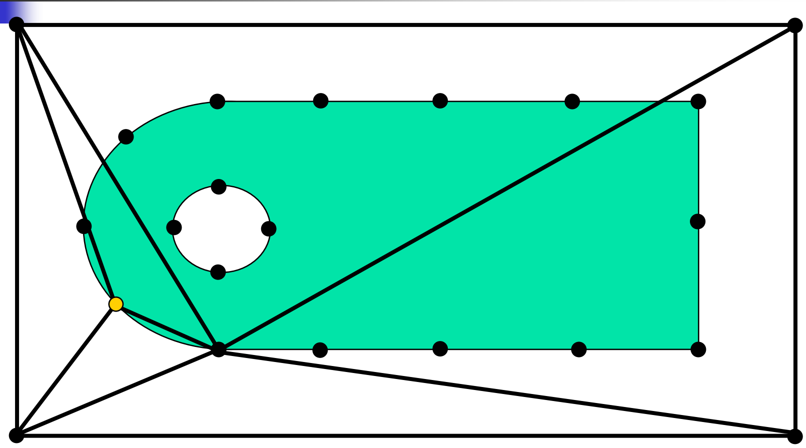
Boundary Insertion



- Insert vertices using Delaunay method

University of British Columbia

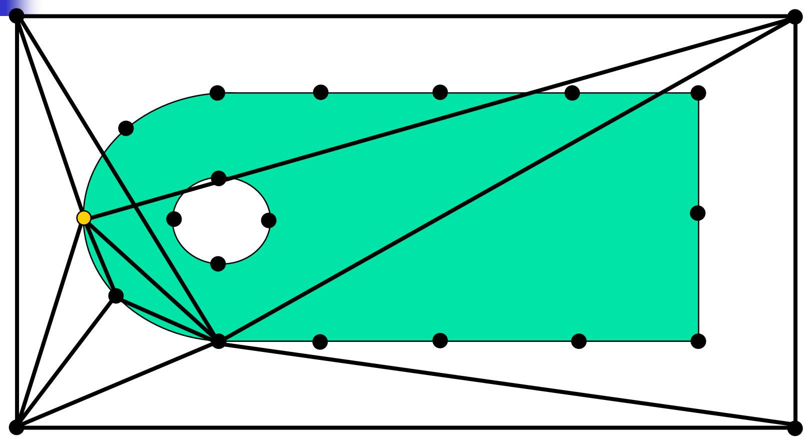
Boundary Insertion



- Insert vertices using Delaunay method

University of British Columbia

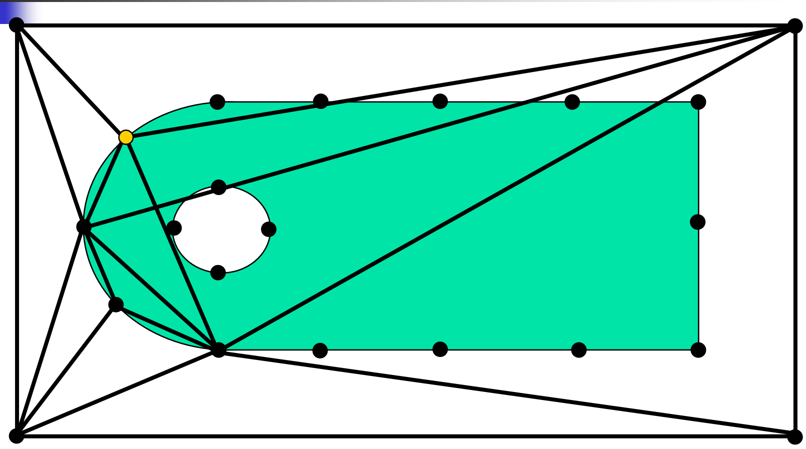
Boundary Insertion



- Insert vertices using Delaunay method

University of British Columbia

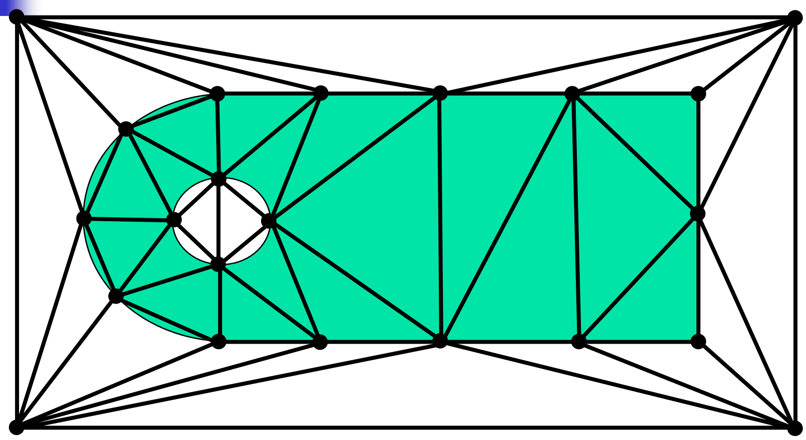
Boundary Insertion



- Insert vertices using Delaunay method

University of British Columbia

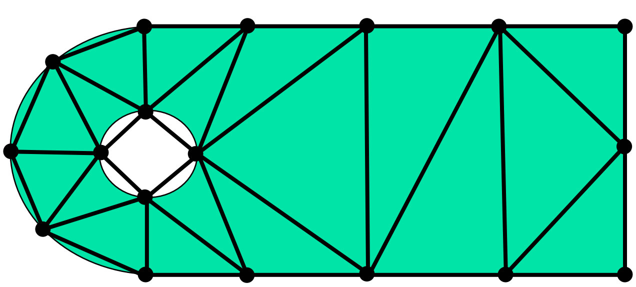
Boundary Insertion



- Insert vertices using Delaunay method

University of
British Columbia

Boundary Insertion



- Delete outside triangles

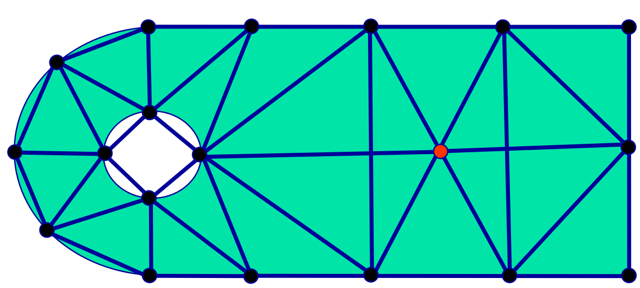
University of
British Columbia

Geometry: Placing Vertices

- Beforehand – domain sampling
- After:
  - Refinement
    - Split any edge longer than specified
      - (typically > 1.5 or 2 of required)
  - Coarsening
    - Collapse short edges
  - Smoothing
- Can combine both
- After approach is simpler to implement
- Without "smart" smoothing - poor distribution
- Each vertex added to mesh as described

University of British Columbia

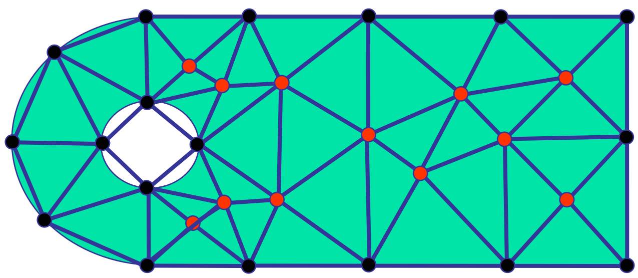
Refinement - Edge Split



- Split long edges – one at a time (Why?)
  - Typically start from longest
- Add to mesh (one by one)

University of British Columbia

Coarsening - Edge Collapse



- Short edges – result of splitting nearby edges
- Collapse – check if any flips needed

University of British Columbia

Smoothing

- Perform a few iterations of smoothing (e.g. Laplacian) with boundary vertices fixed
  - Other weights better oriented towards good spacing exist
- Note: domain not convex – test for foldovers
- After smoothing perform edge flips on edges not satisfying Delaunay requirements

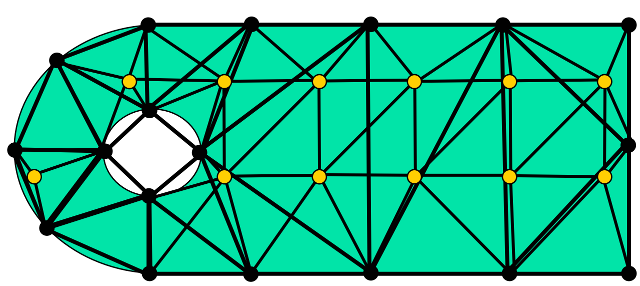
University of British Columbia

Sampling

- Sampling provides initial placement for vertices before any triangulation exists
- Similar to quantization:
  - have "budget" (estimated number of vertices)
  - Need to optimally spread thru the domain
- Choices
  - Grid – place vertices on uniform 2D grid
  - Random / Smart random
  - Centroidal Voronoi diagram

University of British Columbia

Uniform Sampling

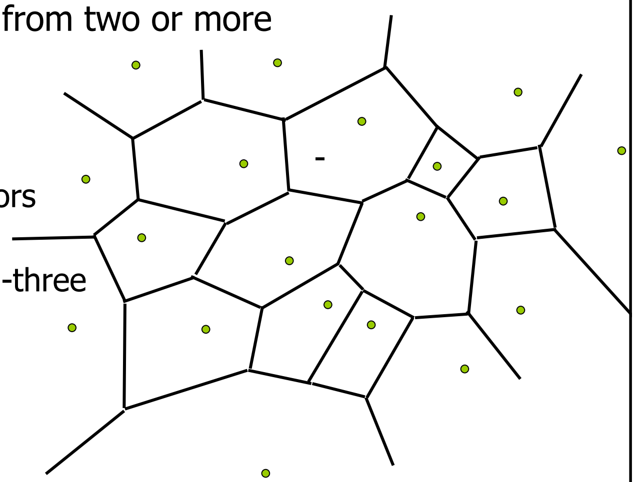


- Place vertices on regular grid
- Artifacts on boundary
- 90/45/45 triangles
  - Very simple
  - Non-Uniform distribution (why?)

University of British Columbia

Voronoi Diagram

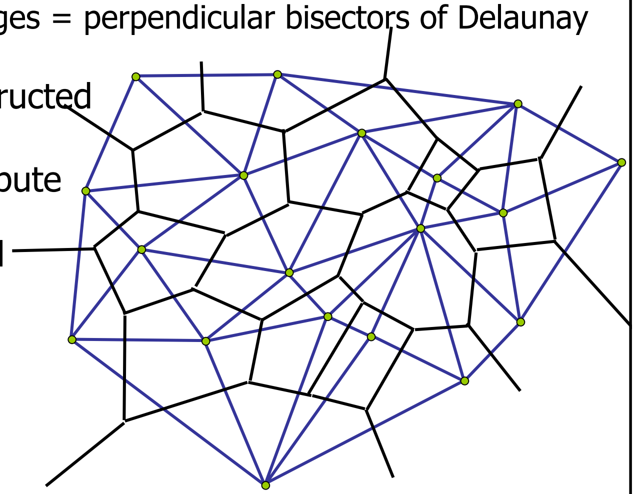
- Voronoi Diagram for given set of vertices: union of all locations at equal distance from two or more vertices
- Consists of
  - straight lines vertex bisectors
  - bisector intersections -three or more lines
  - (why?)



University of British Columbia

Voronoi Diagram

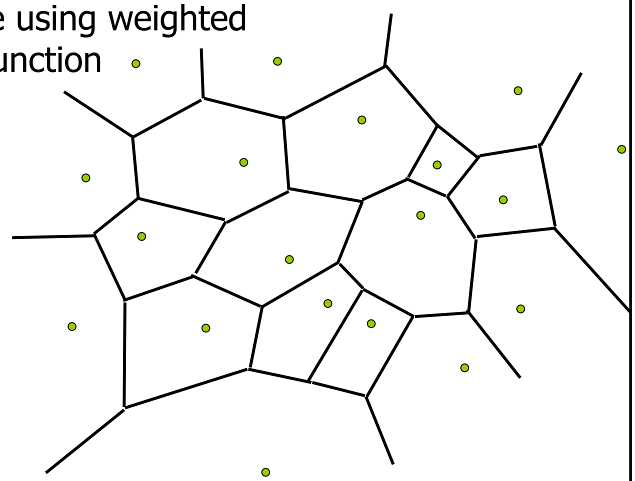
- Dual to Delaunay Triangulation
  - Vertices correspond to faces
  - Voronoi edges = perpendicular bisectors of Delaunay edges
- Can be constructed directly
- Easier – compute Dealunay & compute dual



University of British Columbia

Voronoi Diagram

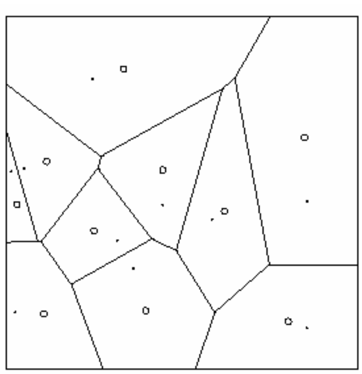
- Diagram partitions space into regions "closer" to one vertex than other
- Can define using weighted distance function



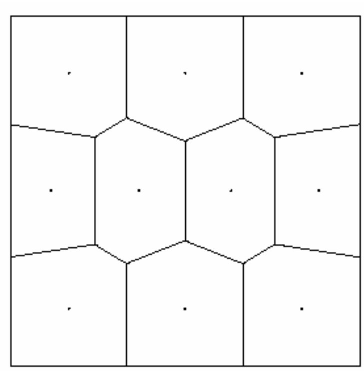
University of British Columbia

Centroidal Voronoi diagram

Vertices (sites) coincide with centroids (center of mass)



Ordinary Voronoi diagram



Centroidal Voronoi diagram

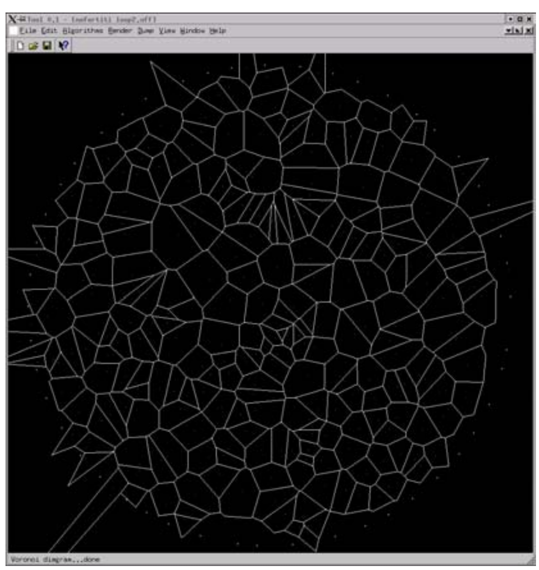
University of British Columbia

Centroidal VD

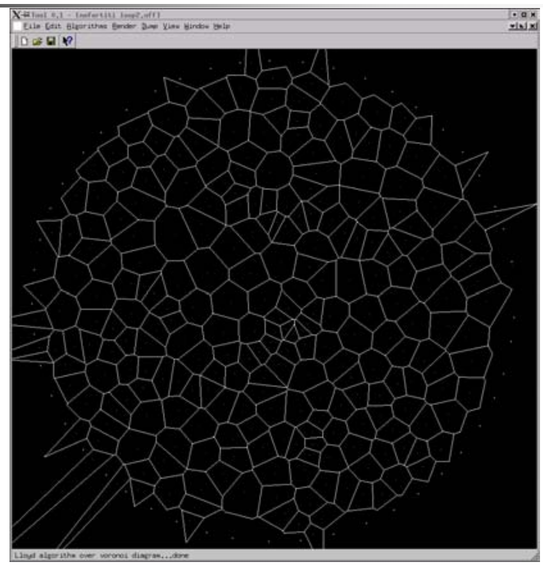
- To compute use Lloyd iterations:
  - Start with set of sites
  - Do
    - Compute VD
    - Compute centers of mass for each Voronoi cell
    - If sites = centers of mass
      - Stop
    - Set sites to centers of mass
    - Repeat
- Guaranteed to converge
- Provides optimal repartitioning of density among vertices

University of British Columbia

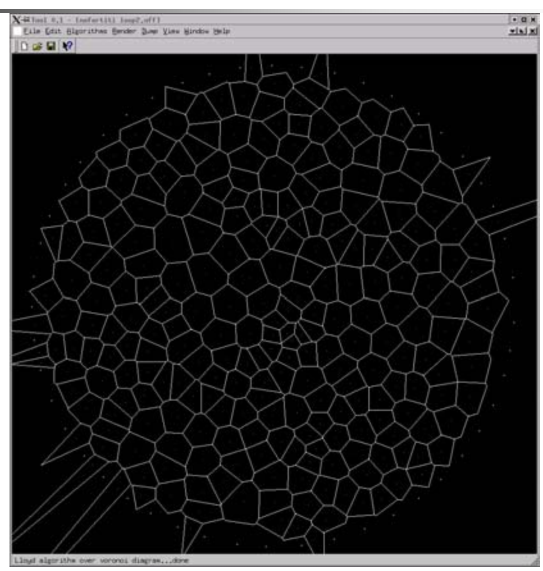
Lloyd & uniform density



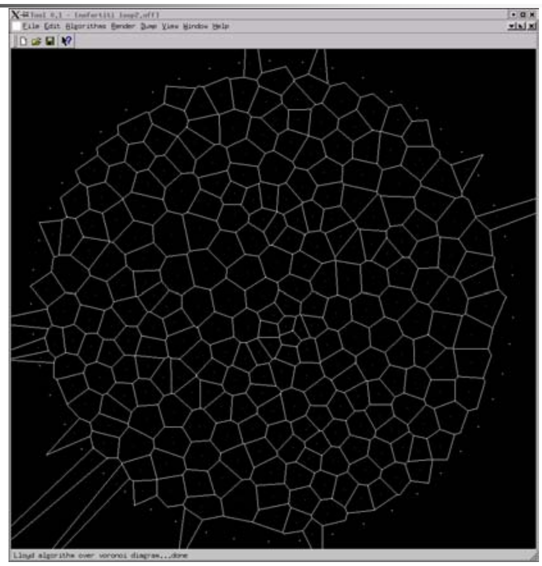
University of British Columbia



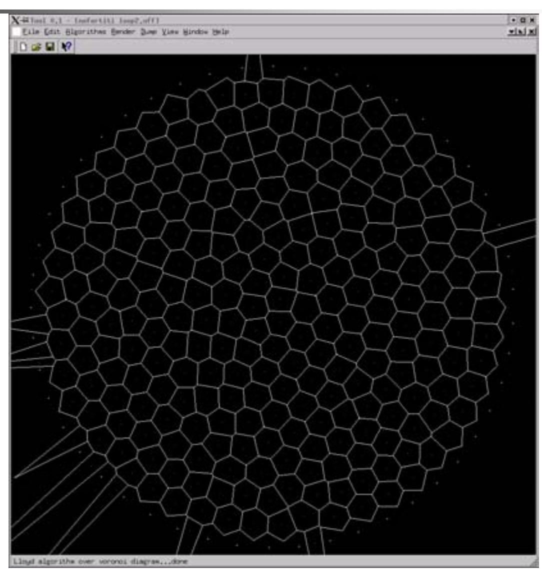
University of British Columbia



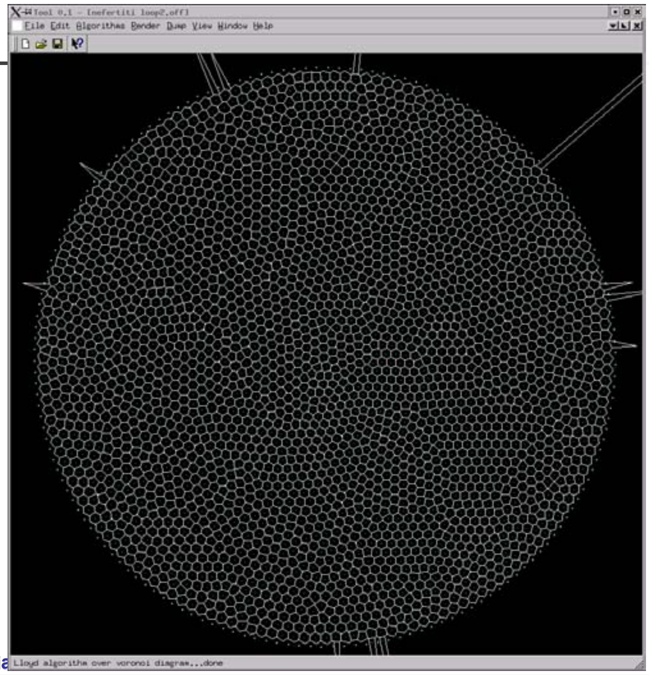
University of British Columbia



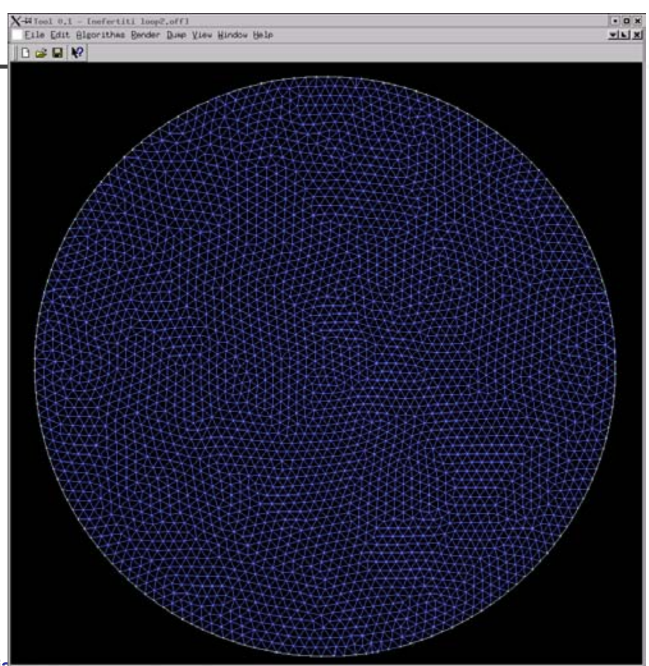
University of
British Columbia



University of
British Columbia



University of British Columbia



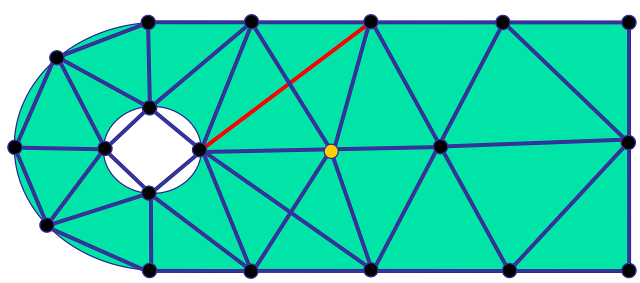
University of British Columbia

Improving Quality

- "Good" sizing does not guarantee good element shape
- Solution: Insert additional vertices
- Insertion criterion - use measures above (min angle, edge ratio,..)
- Insertion strategies
  - Edge split
  - Circumcenters

University of British Columbia

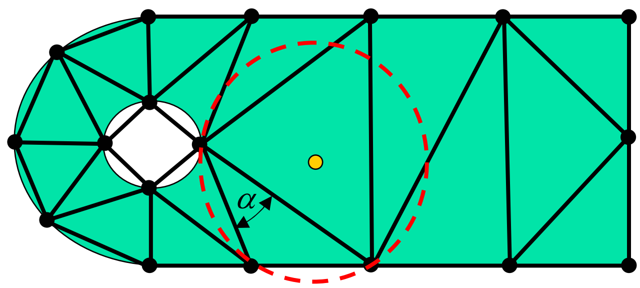
Quality - Edge Split



- Split edge if opposite very obtuse angle
- Start from longest
  - Start from worst triangle
  - If offender edge belongs to triangle with even longer edge, consider it (recursively) first
  - Why?

University of British Columbia

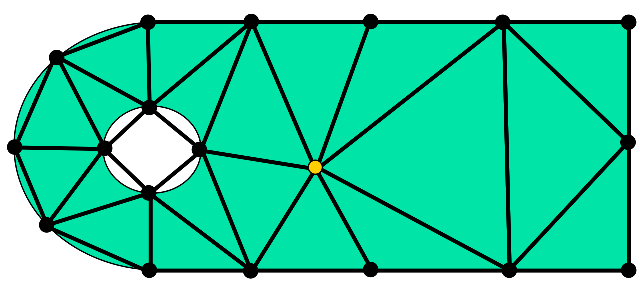
Quality: Vertex Insertion – Circumcenter



- Vertices introduced at triangle circumcenters
- "Guaranteed Quality"
- Continue till minimal angle > predefined minimum

University of British Columbia

Vertex Insertion - Circumcenter



- Add vertices on boundary if circumcenter is outside (if can)
- Performs better than edge split
- More complex to implement

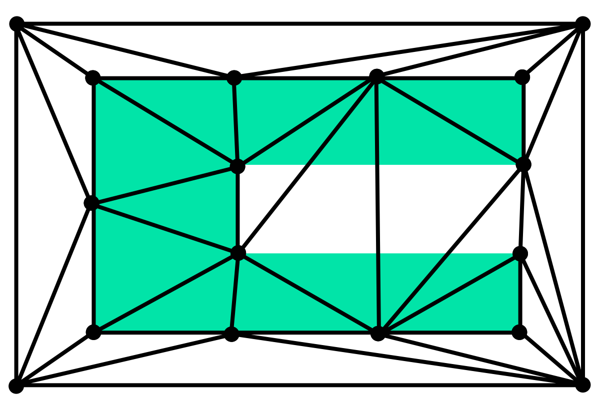
University of British Columbia

Meshing 2D

- Place vertices on boundary
- Use sampling for initial placement inside
- Construct Delaunay triangulation
- Iterate
  - Refinement
  - Coarsening
  - Smoothing
  - Each time perform necessary edge flips
- Last component: boundary recovery

University of British Columbia

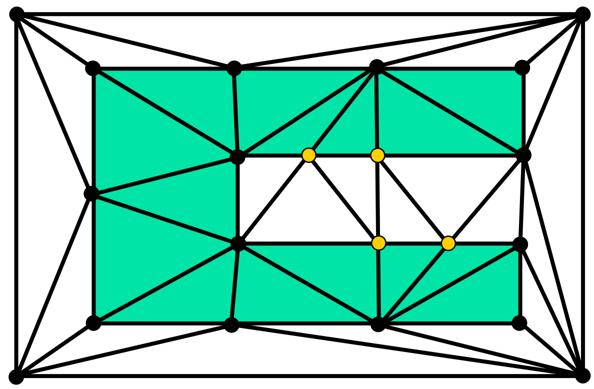
Boundary Recovery



- Delaunay triangulation does not have to obey polygon boundary

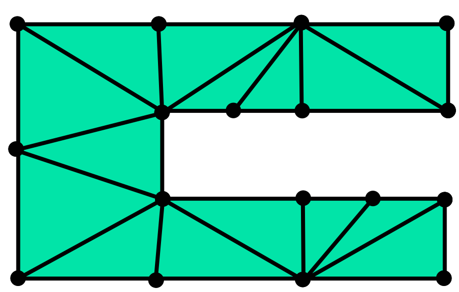
University of British Columbia

Boundary Recovery



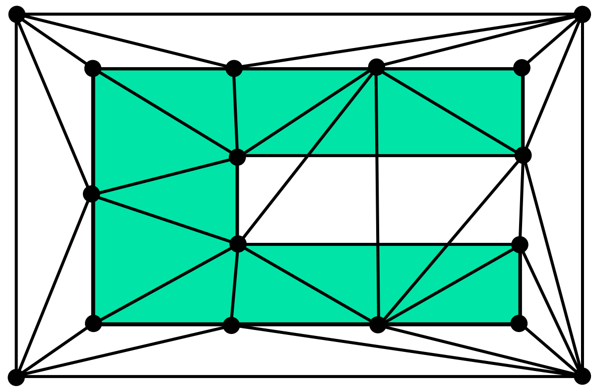
- Boundary Conforming – add vertices at intersections
- Repeat if necessary

Boundary Conforming Delauanay



University of British Columbia

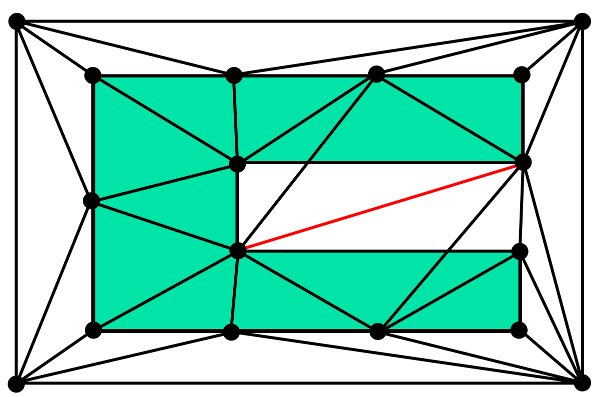
Boundary Recovery - Constrained



- Not always can add boundary vertices (shared edges)

University of British Columbia

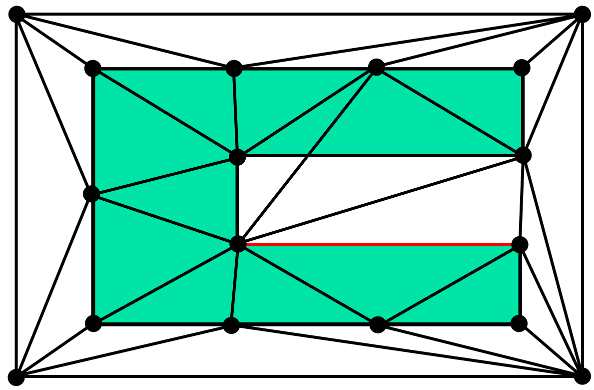
Boundary Recovery - Constrained



- Swap edges between adjacent pairs of triangles
- Repeat till recover the boundary

University of British Columbia

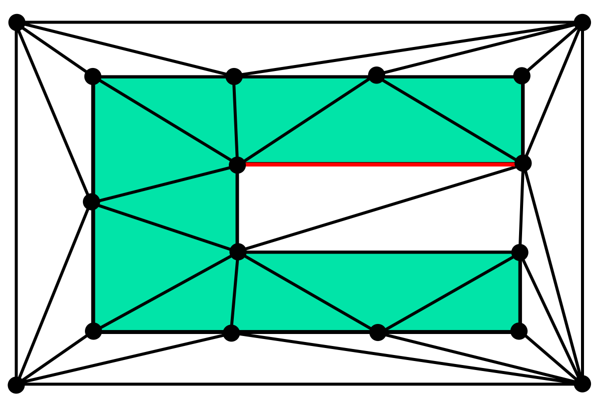
Boundary Recovery - Constrained



- Swap edges between adjacent pairs of triangles
- Repeat till recover the boundary

University of British Columbia

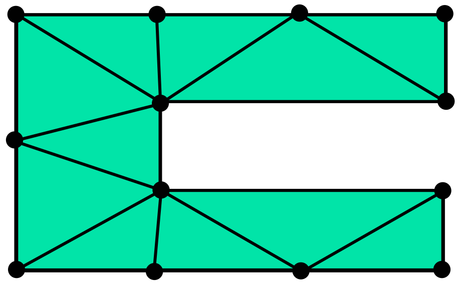
Boundary Recovery - Constrained



- Swap edges between adjacent pairs of triangles
- Repeat till recover the boundary

University of British Columbia

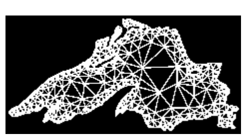
Boundary Recovery - Constrained



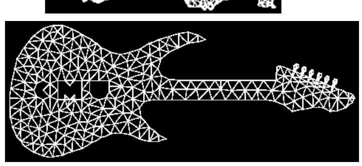
- Does not maintain Delaunay criterion !!!

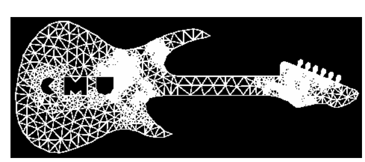
University of
British Columbia

Examples



Boundary Conforming Meshes with different minimal angle constraints (0º, 5º & 15º)





University of
British Columbia