Notes

Meshing goals

- Robust: doesn't fail on reasonable geometry
- Efficient: as few triangles as possible
  - Easy to refine later if needed
- High quality: triangles should be "well-shaped"
  - Extreme triangles make for poor performance of FEM - particularly large obtuse angles

A few approaches

- **Multiblock methods**:
  - If you can mesh simple parts (conformal mapping), decompose region into block to mesh…
  - Hard to handle general geometry!
- **Advancing front**:
  start at boundary, work inwards
  - Can give very high quality near boundary, fast
  - Can run into problems when fronts meet…
- **Tile-based approach**:
  tile space regularly, cut out geometry
  - Can give optimal quality in the interior, fast
  - But can run into problems at the boundary…
- **Delaunay**

Delaunay triangulation

- Given n points $\{x_i\}$ mesh **convex hull** with triangles
- Triangles localized in the following sense: the **circumcircle** of each triangle is empty
- Dual of Voronoi diagram
  - Voronoi region for point x_i: set of all points closer to x_i than any other point
  - Dual: "rotate" edges, faces become points, points become faces
  - Circumcentres are points where three or more Voronoi regions meet

Nice things about Delaunay

- Always exists
  - Unique up to choice of chords in a polygon inscribed in a circle
    (degenerate Voronoi Diagram)
- Easiest triangulation (in some sense) to construct: O(n log n) or better algorithms
- Maximizes minimum angle
  - Not the best guarantee of quality, but useful

Algorithms

- Incremental insertion:
  - Begin with one big triangle containing all points (deleted at the end)
  - Add points one by one, maintaining Delaunay property
  - Each new point many modify nearby triangles: use a tree to accelerate point location
- Divide-and-conquer
  - Split point set in half
  - Triangulate each half recursively
  - Sew two halves together

Lawson's edge-flipping algorithm

- One particular incremental algorithm
- To insert a new point p:
  - Find triangle containing p
  - Add p by dividing triangle in three
  - "Flip" edges that violate Delaunay property:
    is p in the circumcircle of adjacent triangles?
    If so: flip edge, check newly adjacent triangles

Predicates

- Major problem: degenerate triangles
  - E.g. if boundary contains straight edges
- Floating-point rounding can kill the algorithm
- Handle by reducing to simplest predicates possible
  - And then either compute exactly, or in a consistent way
  - See Shewchuk's Triangle code

Delaunay refinement

- Typically we're only given points on the boundary (and maybe not even that many)
- Need to add new points
- Chew showed one particularly good strategy is to add circumcentres of badly shaped triangles
  - Maintain priority queue of worst triangles
  - Adding circumcentre destroys the triangle, replaces it with better shaped versions
  - On boundary: split edges
- Additionally drive insertion by required "size"
  - E.g. coming from error control of PDE

Mesh improvement

- Additionally can postprocess mesh:
  - Move nodes to more optimal locations
    (if just centroid, called "mesh smoothing")
  - Flip edges to get more balanced valences