geometric modeling

- representations
  - conversions
- creation
- manipulation

representations

- how is geometry represented internal to some program
- one can often convert from one rep to another.
  - some operations are easier in some reps

triangle soup

- a list of triangles
- each triangle has 3 vertices
- each vertex has 3 coordinates
- shared vertices may be replicated, or pointed to
- each tri-vert has other attributes
  - e.g normal, texture coordinates
- can associate 2d textures as well as normal maps for rendering
- + easy to render, +easy to acquire

polygon soup

- like triangle soup but can have more verts per poly
- + may be more natural in some settings (architecture)

triangle (poly) meshes

- the base geometry is polygon soup
- lets assume that the geometry is a "surface"
- each edge has two adjacent faces.
- each vertex has a "one-ring" of vertices
- basic idea is to include various pointers in the data set so one can "walk" around the data
  - what are all the vertices "around" a given vertex
  - which poly is across an edge from a given poly
- there are many different such data-structures (fig)
- you will use one such data structure to implement subdiv surface

parametric surfaces

- recall that we could use splines to describe a curve $x(t), y(t)$ with parameter t.
- describe $x(s,t), y(s,t), z(s,t)$ coordinates as a function of s, t
- usually using some piecewise bivariate polynomial form (fig)

- bezier, ubs, nubs, nurbs
- +concise representation of smooth patch
  - +can be adaptively triangulated for rendering as needed
- +allows for somewhat intuitive direct control
- -hard to stitch together multiple patches smoothly to create closed object

subdivision surfaces

- parametric surfaces can be evaluated using repeated subdivision of input grid
- generalize to non-grid mesh (code demo)
- +concise representation of smooth shape
  - +can be adaptively triangulated for rendering as needed
- +allows for somewhat intuitive direct control
- +no need for explicit stitching
- -a bit harder to reason about, or texture map

implicit surfaces

- given some smooth scalar function $f(x, y, z)$
- define surface as points with $f = 0$
  - the sign for non-zeros tells you inside vs out (p:surf2:5, vid)

how to define f

- some algebraic function
  - useful for spheres and other famous canonical shapes
- meta-balls
- define some decreasing function of distance from seed point or segment
- add up functions for all seeds (demo and p:surf2:9)
- +concise representation of smooth shape
- +allows for somewhat intuitive direct control
- +no need for explicit stitching
- +easy to do topology changes
- +easy to find where a ray intersects with surface
- -somewhat hard to turn into polygons (more later)
- +well defined notion of inside and outside (e.g. csg video).
- -hard to do 2d texture map

volume representation

- can have a 3d grid with "density" values at each grid point
- may come from mri scan or from numerical simulation (p:solid:1,3)
- can render slices
- can "interpolate" the values over each cube

- this essentially gives us an scalar function over space
- can render as a 3d semi-transparent fog (vid)
- can interpret use iso-surface as implicit surface(psolid:4)

transformations

- from soup to mesh involves algorithmic stitching
- from parametric to mesh involves (adaptive) direct evaluation
- from mesh to parametric surface involves “optimizing” the fit (ppt)
- from implicit to polygons involves “contouring” (marching cubes vid)
- from closed mesh to volume involves “3d scan convert”

creation

- geometric scanners (lasers, cameras, satellites) (w:cyb)
- typically outputs cloud of points (with colors)
- need to turn this into a mesh (ppt)

simulation

- run a fluid simulator and output a volume rep

procedural

- use some code to generate a shape (ppt)
- fractals (vid)

authoring tool

- software that lets you perform various “tasks” to build up a model.
- often builds stuff up from curves, and basic shapes, plus deformation
- (vid)
- mesh based editing (vids)