09 - Implicit Surfaces, then starting on Solids (rather than surfaces)

- Define surface with function over all of space
- f(x, y) = 0 on curve, <0 → inside, >0 → outside
- Good for intersections

Nice properties

Normals, tangents, and curvatures:

- Normals can be defined by partial derivatives of the function used to define the surface (because surface is a level set of the function)
- Consequently can define tangent plane at a particular point from the normal at that point.
- Curvature - rate at which the normal is changing
    - Computation is more involved
    - Principal directions/curvatures - min and max curvature directions at a point
- So normals, tangents, and curvatures are mathematically more easily defined for implicit surfaces than for arbitrary polygonal meshes.

Checking if point lies inside surface

- Can check efficiently: just evaluate f(x, y, z)

Checking for surface intersections

- given f(), g(), find x, y, z at which f(x, y, z) = g(x, y, z) = 0
- => fast collision detection

Efficient boolean operations

- Union?
    - if we take f * g, then if either is zero, then the resulting function is zero. If both positive, we're outside, if one positive, one negative, we're inside. These are correct...
    - But if both negative, we now register as outside. We want to register as inside.
    - What if we just add them together?
        - Then two negatives => still negative
        - Two positives => still positive
        - Negative, positive is the issue.

    - Still, perhaps you can find something useful. (see slides, although they don't really answer this)

Efficient topology changes

- Surface is not represented explicitly, so can care more about movement of "mass" rather than dealing with fixing a polygonal mesh.

Computations in the volume

- Allows for continuity and smoothness
- Suitable for tasks such as reconstruction

Comparison to parametric surfaces

- Implicit
    - Efficient intersections & topology changes
- Parametric
    - Efficient "marching" along surface & rendering

Implicit surface representations

- How do we define implicit functions?

Algebraic surfaces

- Implicit function is polynomial
- Most common form: quadrics
    - f(x, y, z) = up to degree-2 terms of xyz
- Sphere, ellipsoid, paraboloids, hyperboloids
- Can do cubics, quartics…
- Compare to equivalent parametric surface:
    - Tensor product patch of degree m and n curves yields algebraic function with degree 2mn
    - So a bicubic patch (2 * 2 * 2) has degree 8.
    - Even worse for intersection of parametric surfaces
- Another problem: function extends to infinity
    - Must trim to get desired patch (this is difficult)

Voxels

- Regular array of 3D samples (like image)
    - Samples are called voxels ("volume pixels")

    - Can store colors, density…
- We are representing our function in this grid representation, and we want to know where this function would be 0.
    - Applying reconstruction filter (e.g. trilinear sampling) yields f(x, y, z)
    - Isosurface at f(x, y, z) = 0 defines surface.
- Handle this with “iso-surface extraction algorithm”
- One such algorithm: “Marching cubes”
    - Can look at any group of 8 neighboring samples (that form a cube).
    - Does the surface pass through that region?
    - There are only 15 unique arrangements (modulo reorienting the cube) of the cube’s corners being positive/negative.
    - If define and use these 15 rules in a consistent way, can stitch together into continuous surface.
- Storage? O(n^3) for n x n x n grid
    - 1 billion voxels for 1000 x 1000 x 1000

Basis functions
- Implicit function is sum of basis functions
    - Example:
        - $f(P) = a_0 e^{-b_0 d(P, P_0)^2} + a_1 e^{-b d(P, P_1)^2}$ +…
        - Implicit function is sum of Gaussians
        - Basically summing “blobs” centered at various points
- Reconstruction from point sets
    - Have points explicitly labeling what is inside, outside surface
    - Then add up gaussians to smooth out the shape described by the points.

Implicit surfaces summary

- Advantages:
    - Easy to test if point is on surface
    - Computing intersections/unions/differences
    - Easy to handle topological changes
- Disadvantages:
    - Indirect specification of surface
    - Hard to describe sharp features
    - Hard to enumerate points on surface
        - slow rendering

In-class animation break: Ikea shirts animation

Solid Modeling

- Represent solid interiors of objects

Motivation

- Sometimes you get this in scans
- Some applications require solids (medicine, CAD/CAM)

Voxels

- Store properties of solid object with each voxel
  - occupancy
  - color
  - density
  - temperature
  - etc.
- E.g. took a bunch of slices of frozen human corpse, and took pictures

Voxel Processing

- Signal processing (just like images)
  - Reconstruction
  - Resampling
- Typical operations
  - Blur
  - Edge detect
  - Warp
  - etc.
- Often fully analogous to image processing

Voxel boolean operations

- Compare objects voxel by voxel, and do operation on each individual pair of voxels → trivial

Voxel display

- Isosurface rendering
  - Interpolate samples stored on regular grid
  - Isosurface at f(x, y, z) = 0 defines surface (marching cubes)
- Ray casting
  - Integrate density along rays: compositing!
- Extended ray-casting
  - Transfer functions: map voxel values to opacity and material
  - Normals (for lighting) from density gradient

Voxels: summary

- Advantages
  - Simple, intuitive, unambiguous
  - Same complexity for all objects
  - Natural acquisition for some applications
  - Trivial boolean ops
- Disadvantages
  - Approximate
  - Not affine invariant (we'll talk more about what this means… has to do with transformations)
  - Expensive display
  - Large storage requirements

A question of resolution: quadtrees & octrees

- What resolution should be used for voxels?
- Refine resolution of voxels hierarchically
  - More concise and efficient for non-uniform objects

BSP Trees

- "Binary Spatial Partition"
- In 3d, we can define an object by the planes that divide it from the result of the world
- Can sort of see how we could recursively define a tree where the nodes correspond to separating planes

Key properties

- Gives you a nice visibility ordering (will cover more later) → useful for rendering

- Hierarchy of convex regions (useful for collision)

CSG

- “Constructive solid geometry”
- Represent solid object as hierarchy of boolean operations
  - Union
  - Intersection
  - Difference

Sweeps

- Swept volume
  - Sweep one curve along path of another curve
- Surface of revolution
  - Take a curve and rotate it about an axis