

Implicit Surfaces

- Define surface as where some scalar function of x,y,z is zero:
 - {x,y,z | F(x,y,z)=0}
- Interior (can only do closed surfaces!) is where function is negative
 - {x,y,z | F(x,y,z)<0}
- Outside is where it's positive
 - {x,y,z | F(x,y,z)>0}
- ♦ Ground is F=y
- Example: F=x²+y²+z²-1 is the unit sphere

Testing Implicit Surfaces

- Interference is simple:
 - Is F(x,y,z)<0?
- Collision is a little trickier:
 - Assume constant velocity x(t+h)=x(t)+hv
 - Then solve for h: F(x(t+h))=0
 - This is the same as ray-tracing implicit surfaces...
 - But if moving, then need to solve F(x(t+h), t+h)=0
 - Try to bound when collision can occur (find a sign change in F) then use secant search

Implicit Surface Normals

- Outward normal at surface is just
- Most obvious thing to use for normal at a point inside the object (or anywhere in space) is the same formula
 - Gradient is steepest-descent direction, so hopefully points to closest spot on surface: direction to closest surface point is parallel to normal there
 - We really want the implicit function to be monotone as we move towards/away from the surface

cs533d-term1-2005

 ∇F

n =

Building Implicit Surfaces

- Planes and spheres are useful, but want to be able to represent (approximate) any object
- Obviously can write down any sort of functions, but want better control
 - Exercise: write down functions for some common shapes (e.g. cylinder?)
- Constructive Solid Geometry (CSG)
 - Look at set operations on two objects
 - [Complement, Union, Intersection, …]
 - Using primitive F()'s, build up one massive F()
 - But only sharp edges...

cs533d-term1-2005

Getting back to particles

- "Metaballs", "blobbies", ...
- Take your particle system, and write an implicit function: $\sum_{i=1}^{n} c(|x-x_i|)$

$$F(x) = \sum_{i} \alpha_{i} f\left(\frac{|x - x_{i}|}{r_{i}}\right) -$$

• Kernel function f is something smooth like a Gaussian

$$f(x) = e^{-x}$$

- Strength α and radius r of each particle (and its position x) are up to you
- Threshold t is also up to you (controls how thick the object is)
- See make_blobbies for one choice...

Problems with these

- They work beautifully for some things!
 - Some machine parts, water droplets, goo, ...
- But, the more complex the surface, the more expensive F() is to evaluate
 - Need to get into more complicated data structures to speed up to acceptable
- Hard to directly approximate any given geometry
- Monotonicity how reliable is the normal?

Signed Distance

- Note infinitely many different F represent the same surface
- What's the nicest F we can pick?
- Obviously want smooth enough for gradient (almost everywhere)
- It would be nice if gradient really did point to closest point on surface
- Really nice (for repulsions etc.) if value indicated how far from surface
- The answer: signed distance

cs533d-term1-2005

Defining Signed Distance

- \blacklozenge Generally use the letter φ instead of F
- υ Magnitude $|\phi(x)|$ is the distance from the surface
 - Note that function is zero only at surface
- υ Sign of $\phi(x)$ indicates inside (<0) or outside(>0)
- υ [examples: plane, sphere, 1d]

cs533d-term1-2005

Closest Point Property

- Gradient is steepest-ascent direction
 - Therefore, in direction of closest point on surface (shortest distance between two points is a straight line)
- The closest point is by definition distance løl away
- υ So closest point on surface from x is

$x - \phi(x) \frac{\nabla \phi}{|\nabla \phi|}$

Unit Gradient Property

- Look along line from closest point on surface to x
- ◆ Value is distance along line
- Therefore directional derivative is 1:

$$\nabla \phi \cdot n = 1$$

- But plug in the formula for n [work out]
- So gradient is unit length: $|\nabla \phi| = 1$

Aside: Eikonal equation

- There's a PDE! $|\nabla \phi| = 1$
 - Called the Eikonal equation
 - Important for all sorts of things
 - Later in the course: figure out signed distance function by solving the PDE...

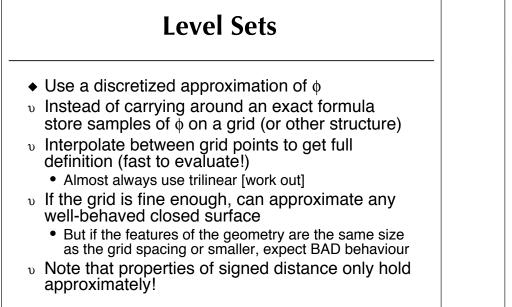
cs533d-term1-2005

13

Aside: Spherical particles

- We have been assuming our particles were just points
- With signed distance, can simulate nonzero radius spheres
 - Sphere of radius r intersects object if and only if φ(x)<r/li>
 - i.e. if and only if $\phi(x)$ -r<0
 - So looks just like points and an "expanded" version of the original implicit surface normals are exactly the same, …

cs533d-term1-2005 14



Building Level Sets

- We'll get into level sets more later on
 - Lots of tools for constructing them from other representations, for sculpting them directly, or simulating them...
- For now: can assume given
- Or CSG: union and intersection with min and max [show 1d]
 - Just do it grid point by grid point
 - Note that weird stuff could happen at sub-grid resolution (with trilinear interpolation)
- Or evaluate from analytical formula

Normals

- We do have a function F defined everywhere (with interpolation)
 - Could take its gradient and normalize
 - But (with trilinear) it's not smooth enough
- Instead use numerical approximation for gradient:

$$g_{i,j,k} = \left(\frac{\phi_{i+1,j,k} - \phi_{i-1,j,k}}{2\Delta x}, \frac{\phi_{i,j+1,k} - \phi_{i,j-1,k}}{2\Delta y}, \frac{\phi_{i,j,k+1} - \phi_{i,j,k-1}}{2\Delta z}\right)$$

- Then, use trilinear interpolation to get (continuous) approximate gradient anywhere
- Or instead apply finite difference formula to 6 trilinearly interpolated points (mathematically equivalent)
- Normalize to get unit-length normal

cs533d-term1-2005

17

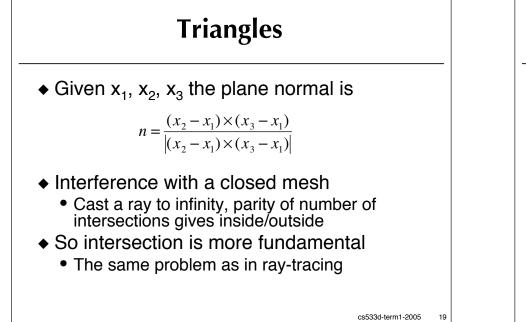
Evaluating outside the grid

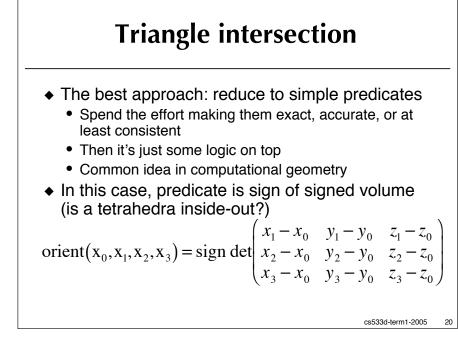
- Check if evaluation point x is outside the grid
- If outside that's enough for interference test
- But repulsion forces etc. may need an actual value
- Most reasonable extrapolation:
 - A = distance to closest point on grid
 - $B = \phi$ at that point
 - Lower bound on distance, correct asymptotically and continuous (if level set doesn't come to boundary of grid):

$$\operatorname{sign}(B)\sqrt{A^2+B^2}$$

• Or upper bound on distance: B + sign(B)A

cs533d-term1-2005 18





Using orient()

◆ Line-triangle

- If line includes x₄ and x₅ then intersection if orient(1,2,4,5)=orient(2,3,4,5)=orient(3,1,4,5)
- I.e. does the line pass to the left (right) of each directed triangle edge?
- If normalized, the values of the determinants give the **barycentric coordinates** of plane intersection point
- Segment-triangle
 - Before checking line as above, also check if orient(1,2,3,4) != orient(1,2,3,5)
 - I.e. are the two endpoints on different sides of the triangle?

cs533d-term1-2005

21

Other Standard Approach

- Find where line intersects plane of triangle
- Check if it's on the segment
- Find if that point is inside the triangle
 - Use barycentric coordinates
- Slightly slower, but worse: less robust
 - round-off error in intermediate result: the intersection point
 - What happens for a triangle mesh?
- Note the predicate approach, even with floatingpoint, can handle meshes well
 - Consistent evaluation of predicates for neighbouring triangles
 Consistent evaluation of predicates for neighbouring
 Constraint evaluation of predicates for neighbouring
 Constraint evaluation of predicates for neighbouring
 Constraint evaluation

Distance to Triangle

- If surface is open, define interference in terms of distance to mesh
- Typical approach: find closest point on triangle, then distance to that point
 - Direction to closest point also parallel to natural normal
- First step: barycentric coordinates
 - Normalized signed volume determinants equivalent to solving least squares problem of closest point in plane
- If coordinates all in [0,1] we're done
- Otherwise negative coords identify possible closest edges
- Find closest points on edges

Testing Against Meshes

- Can check every triangle if only a few, but too slow usually
- Use an acceleration structure:

. . .

- Spatial decomposition: background grid, hash grid, octree, kd-tree, BSP-tree, …
- Bounding volume hierarchy: axis-aligned boxes, spheres, oriented boxes,

cs533d-term1-2005

23

22

Moving Triangles

- Collision detection: find a time at which particle lies inside triangle
- Need a model for what triangle looks like at intermediate times
 - Simplest: vertices move with constant velocity, triangle always just connects them up
- Solve for intermediate time when four points are coplanar (determinant is zero)
 - Gives a cubic equation to solve
- Then check barycentric coordinates at that time
 - See e.g. X. Provot, "Collision and self-collision handling in cloth model dedicated to design garment", Graphics Interface'97

cs533d-term1-2005 25

For Later...

- We now can do all the basic particle vs. object tests for repulsions and collisions
- Once we get into simulating solid objects, we'll need to do object vs. object instead of just particle vs. object
- Core ideas remain the same

cs533d-term1-2005 26