Notes

- Assignment 1 is due today
  - Make sure I have everything by the time I check my email tomorrow morning
- Assignment 2 goes out today
  - Check the website
  - [mention coefficient of restitution, stability limit]

Moving objects

- Same algorithms, and almost same formulas:
  - Need to look at relative velocity $v_{particle}-v_{object}$ instead of just particle velocity
  - As before, decompose into normal and tangential parts, process the collision, and reassemble a relative velocity
  - Add object velocity to relative velocity to get final particle velocity
- Be careful when particles collide:
  - Same relative Δv but account for equal and opposite forces/impulses with different masses…

Moving Objects…

- Also, be careful with interference/collision detection
  - Want to check for interference at end of time step, so use object positions there
  - Objects moving during time step mean more complicated trajectory intersection for collisions

Almost there!

- We have basic time integration for particles in place now
- Assumed we could just do interference detection, but…
- Detecting collisions over particle trajectories can be dropped in for more robustness - algorithms don't change
  - But use the normal at the collision time
- Speaking of normals, what is the normal inside the object?
- How are we detecting interference/collision?

Geometry

- The plane is easy
  - Interference: y<0
  - Collision: y became negative
  - Normal: constant (0,1,0)
- Let's try to generalize this
  - Look at two types of representation, triangle meshes and level sets
  - Representative of general classes of explicit (parameterized) surfaces and implicit surfaces

Implicit Surfaces

- Define surface as where some scalar function of x,y,z is zero:
  - $\{x,y,z \mid F(x,y,z)=0\}$
- Interior (can only do closed surfaces!) is where function is negative
  - $\{x,y,z \mid F(x,y,z)<0\}$
- Outside is where it's positive
  - $\{x,y,z \mid F(x,y,z)>0\}$
- Ground is F=y
- Unit sphere is $F=x^2+y^2+z^2-1$

Implicit Surface Interference

- 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
    - Could use Newton's method!
  - This is the same as ray-tracing implicit surfaces…
  - But if moving, then need to solve F(x(t+h), t+h)=0
  - This is a little bit harder (Secant?)

Implicit Surface Normals

- Outward normal at surface is just $n = \frac{\nabla F}{|\nabla F|}$
- 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
  - We really want the implicit function to be monotone as we move towards/away from the surface
  - More generally, think of using the normal from the closest point on the surface… (save that thought)

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…

Getting back to particles

- "Metaballs", "blobbies", …
- Take your particle system, and write an implicit function:

$$F(x) = \sum_i \alpha_i f\left(\frac{|x - x_i|}{r_i}\right) - t$$

  - Kernel function f is something smooth like a Gaussian $f(x) = e^{-x^2}$
  - 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)

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

Defining Signed Distance

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

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 $|\phi|$ away
- 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…

Aside: Spherical particles

- We have been assuming our particles were just points
- More general (rigid) objects: next week
- But with signed distance, can simulate nonzero radius spheres
  - Sphere of radius r intersects object if and only if $\phi(x)<r$
  - 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, …

Level Sets

- Use a discretized approximation of ϕ
- Instead of carrying around an exact formula store samples of ϕ on a grid
- Interpolate between grid points to get full definition (fast to evaluate!)
  - Almost always use trilinear [work out]
- If the grid is fine enough, can approximate any closed surface [draw it]
- Note that properties of signed distance only hold approximately!

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
  - E.g. plane, sphere, cube, …

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
  - Normalize to get unit-length normal

Alternatively

- Use the same finite difference formula, but directly at point we're evaluating at
  - Need to trilinearly interpolate 6 points
  - Reuse coefficients
  - Mathematically equivalent; costs are comparable (architecture dependent!) [exercise: check this!]
- Could be useful for…

Evaluating outside the grid

- Usually need to 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 = ϕ at that point
  - Return $\mathrm{sign}(B)\sqrt{A^2 + B^2}$
  - Lower bound on distance, correct asymptotically, continuous.

Explicit Surfaces

- An explicit formula to generate points on surface from 2D parameter space
  - E.g. x(a,b)=(a,0,b) is the plane
  - x is a convex combination of 3 fixed points chosen from a list of triples: triangle mesh
- Interference - does a ray cast to infinity cross surface an odd number of times
  - Or check outward normal at closest point on surface, after finding it!
- Note: can do open surfaces with no interior

Explicit Surfaces…

- Collision - solve $x_{surface}(a,b)=x_{particle}(t)$ for t in collision time step
  - Want first solution if one exists
  - Note: 3 unknowns in general: a,b,t
- Normal: finally something easy
  - Explicit formula from cross-product of partial derivatives

$$n(a,b) = \frac{\partial x/\partial a \times \partial x/\partial b}{\left|\partial x/\partial a \times \partial x/\partial b\right|}$$

Normals not on the surface

- Can take our cue from implicit surfaces
  - Take the direction to the closest point (or a reasonable approximation of it)
  - Note that this kind of looks like figuring out signed distance (or some other implicit surface function)

Triangle Meshes

- Simplest general purpose explicit surface
- Let's start with one triangle
  - Corners x_1, x_2, x_3 means

$$n = \frac{(x_2 - x_1) \times (x_3 - x_1)}{\left|(x_2 - x_1) \times (x_3 - x_1)\right|}$$

  - Can then define implicit function for plane the triangle lies in: $(x - x_1) \cdot n = 0$
    - Actually signed distance if we think of n as pointing outwards…

Segment-triangle intersection

- Important for ray-tracing *(Understatement)*
- Several algorithms out there…
- Here: for checking linear particle trajectory
  - Also checking if inside a closed triangulated object
- First check if segment intersects plane
  - Do endpoint signed distances $(p-x_1) \cdot n$ and $(q-x_1) \cdot n$ have different sign?
- Find plane-intersection point along segment
  - Parameter s such that $(p(1-s)+qs - x_1) \cdot n=0$
  - [work out]
- Find barycentric coordinates of intersection

Barycentric coordinates

- How the triangle is parameterized:

$$\begin{aligned} x(a,b) &= x_1 + a(x_2 - x_1) + b(x_3 - x_1) \\ &= x_1 + au + bv \end{aligned}$$

- a and b are the barycentric coordinates
- If a>=0 and b>=0 and a+b<=1 then we're inside the triangle
- How do we compute them for the plane intersection point?

Computing Barycentric Coords

- We could equate plane-intersection point x_P with parametric point on triangle
  - Problem: 3 equations, 2 unknowns (a,b)
  - If x_P isn't exactly on plane (e.g. round-off error), there will be no solution…
- Least squares!
- Compute closest parameterized point to x_P

$$\min_{a,b} \left| x_P - x_1 - au - bv \right|^2$$

Normal equations

- [derive] $\begin{pmatrix} u \cdot u & u \cdot v \\ u \cdot v & v \cdot v \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} u \cdot (x_P - x_1) \\ v \cdot (x_P - x_1) \end{pmatrix}$
- Determinant formula for solution:

$$\begin{pmatrix} a \\ b \end{pmatrix} = \frac{1}{u^2 v^2 - (u \cdot v)^2} \begin{pmatrix} v^2 & u \cdot v \\ u \cdot v & u^2 \end{pmatrix} \begin{pmatrix} u \cdot (x_P - x_1) \\ v \cdot (x_P - x_1) \end{pmatrix}$$

- Note this formula works for any point in space, not just on the plane…
  - Useful if we want to know closest point in triangle

Round-Off Error

- Always a big concern for collision detection
  - [draw fuzzy triangles in mesh]
  - Particles can fly through the corners/edges of a triangle mesh
- Need to stick in tolerances for all inequalities
  - Tricky part: tuning
  - Very difficult to actually prove what tolerance you should use
  - Other approaches based on consistent primitives - cross-products with edges