1

Notes

2

Triangle intersection

- Many, many ways to do this
- Most robust (and one of the fastest) is to do it based on determinants
  - For vectors a,b,c define $\det(a,b,c) = \begin{vmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{vmatrix} \quad (= a \times b \cdot c)$
  - Det(a,b,c)=±6 volume(tet(a,b,c)), the signed volume of the tetrahedron spanned by edges a,b,c from a common point
  - Sign flips when tetrahedron reflected, or alternatively from right-hand-rule on a×b•c
- Triangle intersection boils down to
  - 2 sign checks: segment crosses plane
  - 3 sign checks: line goes through triangle

3

Triangle Mesh (more)

- Object normal
  - Normalize cross-product of two sides of the triangle
- Distance from single triangle
  - Find barycentric coordinates -- solve a least-squares problem
  - Need to clip to sides of triangle
  - Compute distance from that point
  - Note: also gives direction to closest point
- Distance (and direction) from mesh
  - Compute for all possible triangles, take minimum
  - Trick is to find small list of possible triangles with acceleration structure

4

Implicit Surface

- Simple function, metaballs, or interpolated from 3d grid (“level set”)
  - Recall - for metaballs need acceleration
- Particle inside: $f(x)<0$
- Trajectory cross:
  - Just like ray-tracing - use secant method
- Object normal: $\nabla f/|\nabla f|$
- Distance from surface:
  - If f() is signed distance, then trivial
  - Otherwise, painful, but f() might be good enough for application

Back to particle collisions

- So now we can represent other geometry, how do we do a repulsion velocity field?
  - v(x)=f(distance(x)) * n(x)
  - n(x) is the outward direction (=normal on surface)
  - f is some decreasing function that drops towards zero far away
    - Exponential $f(d)=e^{-k*d}$
    - Or linear drop, truncated to zero: f(d)=max(0,m-k*d)
    - Or more complicated
  - Outward direction is plus/minus direction to closest point
- Aside: useful for more than just collisions - e.g. fire particles streaming out of an object

Force-based repulsions

- Can do exactly the same trick for force-based motion
  - Add repulsion field to F(x)
- Simple, often works, but there are sometimes problems
  - What are you trying to model?
  - Robustness - high velocity impacts can penetrate arbitrarily far
    - High velocity impacts may go straight through thin objects
  - How much of a rebound do you want?

Damped repulsions

- Think of repulsion force as a generalized spring
- Add spring damping:

$$F_{damp} = -D\big(v \cdot n(x)\big)n(x)$$

  - D is some parameter you set
  - n(x) is the outward direction again

Aside: springs and damping

- How do you come up with reasonable values for spring constants and damping constants?
  - And how do you pick good step sizes for differential equation solver (Forward Euler etc.)
- Look at 1D simplified model
  - Ma=F=-Kx-Dv
  - M is the mass, K is like a spring stiffness, D is the damping parameter
- Solve it analytically

Critical Damping

- Three cases:
  - Underdamped (D^2-4MK<0)
    - Oscillation with frequency $\omega \sim \frac{1}{2\pi}\sqrt{K/M}$
    - Characteristic time: $t \sim 2\pi\sqrt{M/K}$
    - Exponentially decays at rate $r = -D/(2M)$
    - Characteristic time: $t \sim 2M/D$
  - Overdamped (D^2-4MK>0)
    - No continued oscillation
    - Exponentially decays at rates $r \sim -K/D, -D/M$
    - Characteristic times: $t \sim D/K, M/D$
  - Critically damped (D^2-4MK=0) $D = 2\sqrt{MK}$
    - No continued oscillation
    - Fastest decay possible at rate $r = -D/(2M)$
    - Characteristic time: $t \sim 2M/D$

Numerical time steps

- Should be proportional to minimum characteristic time
  - Implicit methods like Backwards Euler actually let you take larger steps with stability, but wipe out all hope of accuracy for things with small characteristic time
- For nonlinear multi-dimensional forces, what are K and D?
  - Estimate them by figuring out what is the fastest |F| can change if you modify x or v respectively
  - This is all very approximate, so don't get hung up on getting the "right" answer
  - Will ultimately need a fudge factor anyhow (from experiments)

True Collisions

- Turn attention from repulsions for a while
- Model collision as a discrete event - a bounce
  - Input: incoming velocity, object normal
  - Output: outgoing velocity
- Need some idea of how "elastic" the collision
  - Fully elastic - reflection
  - Fully inelastic - sticks (or slides)
- Let's ignore friction for now
- Let's also ignore how to incorporate it into algorithm for moving particles for now

Newtonian Collisions

- Say object is stationary, normal at point of impact is n
- Incoming particle velocity is v
- Split v into normal and tangential components:

$$v_N = v \cdot n$$

$$v_T = v - v_N n$$

- Newtonian model for outgoing velocity
  - Unchanged tangential component v_T
  - New normal component is $v_N^{new} = -\varepsilon v_N^{old}$
  - The "coefficient of restitution" is ε, ranging from 0 (inelastic) to 1 (perfectly elastic)
- The final outgoing velocity is

$$v^{new} = v_T^{old} - \varepsilon v_N^{old} n$$

Relative velocity in collisions

- What if particle hits a moving object?
- Now process collision in terms of relative velocity
  - $v_{rel}=v_{particle}-v_{object}$
  - Take normal and tangential components of relative velocity
  - Reflect normal part appropriately to get new v_{rel}
  - Then new $v_{particle}=v_{object}+(\text{new } v_{rel})$