Notes

- Assignment 2 going okay?
  - Make sure you understand what needs to be done before the weekend
- Read Guendelman et al, "Nonconvex rigid bodies with stacking", SIGGRAPH'03
- Mistake last class: (forgot a transpose in calculating torque)

$$S^T F_{\text{ext}} = \sum_i \begin{pmatrix} \delta \\ X^{*T} - x_i^{*T} \end{pmatrix} f_i = \begin{pmatrix} \sum_i f_i \\ \sum_i (x_i - X) \times f_i \end{pmatrix} = \begin{pmatrix} F \\ \tau \end{pmatrix}$$

Inertia Tensor Simplified

- Reduce expense of calculating I(t):

$$I(t) = \sum_i m_i (x_i - X)^{*T} (x_i - X)^* = \sum_i m_i \left[(x_i - X)^T (x_i - X)\delta - (x_i - X)(x_i - X)^T \right]$$

- Now use x_i-X=Rp_i and use $R^T R = \delta$

$$I(t) = \sum_i m_i \left[p_i^T R^T R p_i \delta - R p_i p_i^T R^T \right] = R \underbrace{\left(\sum_i m_i \left(p_i^T p_i \delta - p_i p_i^T \right) \right)}_{I_{body}} R^T$$

Inertia Tensor Simplified 2

- So just compute inertia tensor once, for object space configuration
- Then $I(t)=R I_{body} R^T$
- And $I(t)^{-1}=R(I_{body})^{-1}R^T$
  - So precompute inverse too
- In fact, since I is symmetric, know we have an orthogonal eigenbasis Q
- Rotate object-space orientation by Q
  - Then I_{body} is just diagonal!

Degenerate Inertia Tensors

- Inertia tensor can always be inverted unless all the points of the object line up (object is a rod)
  - Or there's only one point
- We don't care though, since we can't track rotation around that axis anyways
  - So diagonalize I, and only invert nonzero elements

Taking the limit

- Letting our decomposition of the object into point masses go to infinity:
  - Instead of sum over particles, integral over object volume
  - Instead of particle mass, density at that point in space

$$\sum_i m_i \operatorname{foo}(x_i) \rightarrow \iiint_x \rho(x) \operatorname{foo}(x) dx$$

Computing Inertia Tensors

- Do the integrals: $I_{body} = \iiint_p \rho \left(p^T p \delta - p p^T \right) dp$
- Lots of "fun"
- You *may* just want to look them up instead
  - E.g. Eric Weisstein's World of Science on the web
- If not…. align axis perpendicular to planes of symmetry (of ρ) in object space
  - Guarantees some off-diagonal zeros
- Example: sphere, uniform density, radius R

$$\begin{pmatrix} \frac{2}{5}MR^2 & 0 & 0 \\ 0 & \frac{2}{5}MR^2 & 0 \\ 0 & 0 & \frac{2}{5}MR^2 \end{pmatrix}$$

Approximating Inertia Tensors

- For complicated geometry, don't really need exact answer
- Could just take the inertia tensor from a simpler geometric figure (will anyone notice?)
- Or numerically approximate integral
  - If we can afford to spend a lot of time precomputing, life is simple
  - Grid approach: sample density…
  - Monte Carlo approach: random samples

Combining Objects

- What if object is union of two simpler objects?
- Integrals are additive
  - But DO NOT USE $I_1(t)+I_2(t)$:
    - World-space formulas (x-X) use the X for the object: X_1 and X_2 may be different
    - Simplified I_{body} formula based on having centre of mass at origin
  - Let's work it out from the integral of I(t)
- Combined mass: $M=M_1+M_2$
- Centre of mass of combined object:

$$X = \frac{\int_{\Omega_1 \cup \Omega_2} \rho x}{\int_{\Omega_1 \cup \Omega_2} \rho} = \frac{M_1 X_1 + M_2 X_2}{M}$$

Combined Inertia Tensor

$$\begin{aligned}
I(t) &= \int_{\Omega_1 \cup \Omega_2} \rho (x-X)^{*T}(x-X)^* \\
&= \int_{\Omega_1} \rho (x - X_1 + X_1 - X)^{*T}(x - X_1 + X_1 - X)^* + \int_{\Omega_2} \cdots \\
&= \int_{\Omega_1} \rho (x-X_1)^{*T}(x-X_1)^* + \int_{\Omega_1} \rho (X_1-X)^{*T}(x-X_1)^* \\
&\quad + \int_{\Omega_1} \rho (x-X_1)^{*T}(X_1-X)^* + \int_{\Omega_1} \rho (X_1-X)^{*T}(X_1-X)^* + \int_{\Omega_2} \cdots \\
&= I_1(t) + (X_1-X)^{*T} \underbrace{\int_{\Omega_1} \rho (x-X_1)^*}_{0} + \underbrace{\int_{\Omega_1} \rho (x-X_1)^{*T}}_{0} (X_1-X)^* \\
&\quad + M_1 (X_1-X)^{*T}(X_1-X)^* + \int_{\Omega_2} \cdots \\
&= I_1(t) + M_1 (X_1-X)^{*T}(X_1-X)^* + I_2(t) + M_2 (X_2-X)^{*T}(X_2-X)^*
\end{aligned}$$

Numerical Integration

- Recall equations of motion

$$\tfrac{d}{dt} V = F/M \qquad \tfrac{d}{dt} L = \mathrm{T}$$

$$\tfrac{d}{dt} X = V \qquad \omega = I(t)^{-1} L$$

$$\tfrac{d}{dt} R = \omega^* R$$

- X and V is just like particle motion
- Angular components trickier:
  R must remain orthogonal, but standard integration will cause it to drift
  - Can use Gram-Schmidt, but expensive and biased

Improving on R

- Instead of 9 numbers for 3 DOF, use a less redundant representation
- Euler angles: 3 numbers
  - But updating with angular velocity is painful
- Quaternions: 4 numbers

What are quaternions?

- Instead of R, use q=(s,x,y,z) with |q|=1
  - Can think of q as a "super complex number" s+xi+yj+zk
  - $i^2=j^2=k^2=-1$, ij=-ji=k, jk=-kj=i, ki=-ik=j
  - Quaternions don't commute! $q_1q_2 \neq q_2q_1$ in general
- Represents "half" a rotation:
  - s=cos(θ/2)
  - $|x,y,z|^2=\sin^2(\theta/2)$
  - Axis of rotation is (x,y,z)
- Conjugate (inverse for unit norm) is

$$\bar{q} = (s, -x, -y, -z)$$

Rotating with quaternions

- Instead of Rp, calculate $q(0,p)\bar{q}$
- Composing a rotation of Δtω to advance a time step:

$$q_{n+1} = \left(\cos\left|\Delta t \frac{\omega}{2}\right|, \sin\left|\Delta t \frac{\omega}{2}\right| \frac{\omega}{|\omega|}\right) q_n$$

- For small Δtω approximate:

$$q_{n+1} = \left(1, \Delta t \frac{\omega}{2}\right) q_n = q_n + \Delta t \frac{(0,\omega)}{2} q_n$$

- From this get the differential equation:

$$\dot{q} = \tfrac{1}{2}(0,\omega)q$$

Integrating Rotation

- Can update like Symplectic Euler, but need to renormalize q after each step
- For reasonable accuracy, limit time step according to rate of rotation
  - Don't try for more than a quarter turn per time step, say
  - Stability is not an issue due to renormalization
- For more accurate methods, see S. R. Buss, "Accurate and efficient simulation of rigid body rotations", JCP 2000

Converting q to R

- Clearly superior to use quaternions for storing and updating orientation
- But, slightly faster to transform points with rotation matrix
- If you need to transform a lot of points (collision detection…) may want to convert q into R
- Basic idea: columns of R are rotated axes $R(1,0,0)^T$, $R(0,1,0)^T$, and $R(0,0,1)^T$
- Do the rotation with q instead.
  - Can simplify and optimize for the zeros - look it up

Gravity

- Force on a point is $m_i g$
- Net force:

$$F = \sum_i m_i g = Mg$$

- Net torque:

$$\tau = \sum_i (x_i - X) \times m_i g = \left(\left(\sum_i m_i x_i\right) - MX\right) \times g = 0$$

Collision Impulses

- Can use same collision detection as deformable objects
  - Since geometry is fixed, may be cheaper
  - E.g. can use level set approximation to geometry
- But applying collision impulses is more complicated than for simple particles
  - Need to take into account angular motion too
- Use same principle though for the colliding points
  - What is the impulse that causes their relative velocity to change as desired?

Frictionless impulse

- Object velocities at point:
  - $v_i = \omega_i \times (x - X_i) + V_i$
- Relative velocity $v = v_1 - v_2$
  - Normal component $v_n = v \cdot n$
- Want post-collision relative normal velocity to be $v_n^{after} = -\varepsilon v_n$
- Apply an impulse $j = j_n n$ in the normal direction to achieve this

$$V_i^{after} = V_i + M_i^{-1} j_i$$
$$L_i^{after} = L_i + (x - X_i) \times j_i$$
$$\omega_i^{after} = \omega_i + I_i(t)^{-1}(x - X_i) \times j_i$$
$$j_i = (-1)^{i+1} j_n n$$

Computing frictionless impulse

$$K_i = \frac{1}{M_i}\delta + (x - X_i)^{*T} I_i^{-1} (x - X_i)^*$$

$$j = \frac{-(1+\varepsilon)v_n}{n^T (K_1 + K_2) n} n$$

Computing friction

- Static friction valid only in "friction cone"

$$|j_T| \le \mu |j_n|$$

- Approach:
  - Calculate static friction impulse (whatever it takes to make relative velocity zero)
  - Check if it's in the friction cone
  - If so, we're done
  - If not, try again with sliding

Computing static friction

$$v^{after} = -\varepsilon v_n n$$

$$j = (K_1 + K_2)^{-1}(-v - \varepsilon v_n n)$$

Sliding friction

- If computed static friction impulse fails friction cone test
- We'll assume sliding direction stays constant during impact: tangential impulse just in the initial relative velocity direction
  - Not true in some situations…

Computing sliding friction

$$T = \frac{v - v_n n}{|v - v_n n|}$$

$$j = j_n n - \mu j_n T$$

$$j_n = \frac{-(1+\varepsilon)v_n}{n^T (K_1 + K_2)(n - \mu T)}$$

Rigid Collision Algorithms

- Use the same collision response algorithm as with particles
  - Identify colliding points as perhaps the deepest penetrating points, or the first points to collide
  - Make sure they are colliding, not separating!
- Problem: multiple contact points
  - Fixing one at a time can cause rattling.
  - Can fix by being more gentle in resolving contacts - negative coefficient of restitution
- Problem: multiple collisions (stacks)
  - Fixing one penetration causes others
  - Solve either by resolving simultaneously or enforcing order of resolution