Physically Based Animation

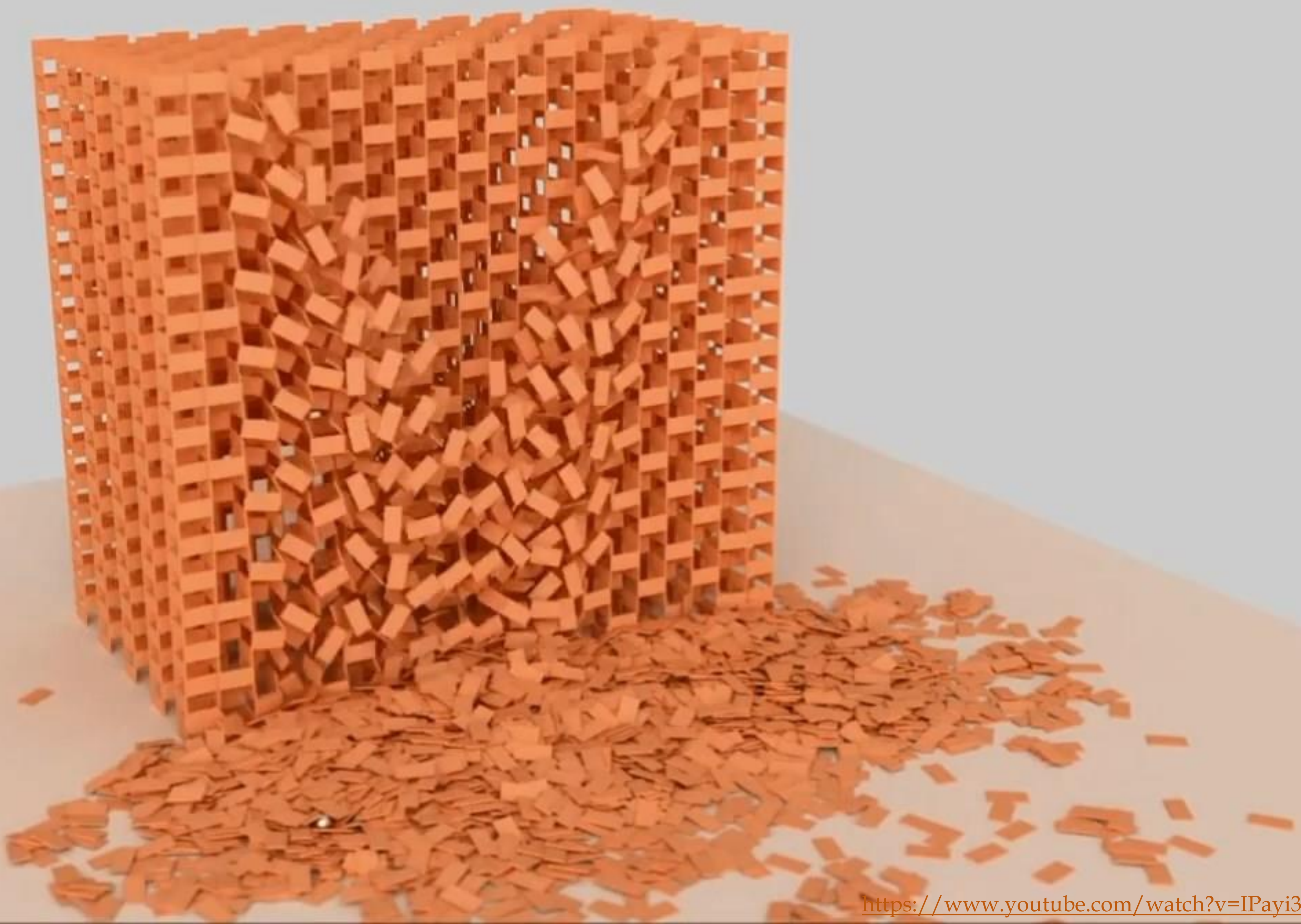
National Cheng Kung University, Fall 2017

Instructor: Shih-Chin Weng 翁士欽 (Style.Me)

Motivations

- The goal of animation is bringing character to life
  - It is about *timing and spacing*
- Few animation principles (like secondary motion, following through, etc.) are inspired from physical effects
- But using keyframe animation for physical effects is quite hard to match the timing and movements of interacting objects
- Physically-based animation (or modeling) is about using the principles of physics to model dynamically evolving phenomena

Bullet Physics - Over 50,000 Planks



https://www.youtube.com/watch?v=IPayi38vQws

View-Dependent Adaptive Cloth Simulation [Koh et al., SCA'14]

How Disney's 'Moana' created its amazing water effects

https://www.youtube.com/watch?v=2tbrocVCb3I

Foundations of Physically-Based Animation

- Model
  - A set of rules govering how something behaves (e.g. classical physics)
- Simulation
  - Predict the evolution over time
  - Time integration
  - Collision detection & response

Dynamics & Numerical Integration

Classical Physics

- The physics before the advent of quantum mechanics
  - The related physical laws are ***deterministic***
- The job of classical physics is to *predict future behavior*
  - It also needs to be revertible (able to trace the history)
  - The ***state*** is everything we need to predict the future by using governing laws
- System: a collection of objects (i.e., particles, fields, waves, etc.)
- Dynamical system: a system changes with *time*

Particle System

- Particles are objects
  - have mass, position and velocity
  - respond to force
  - have no spatial extent
- Physical properties
  - Position: $\mathbf{x}(t)$
  - Velocity: $\mathbf{v}(\mathrm{t}) = \frac{d\mathbf{x}}{dt} = \dot{\mathbf{x}}$
  - Acceleration: $\mathbf{a}(t) = \frac{d\mathbf{v}}{dt} = \dot{\mathbf{v}} = \frac{d^2\mathbf{x}}{dt^2} = \ddot{\mathbf{x}}$
- How do we update acceleration at any instant of time?
  - Newton 2nd law: $\mathbf{f} = m\mathbf{a}$

Evolve the Particle System

```
for t in duration:
# compute net force in another separated loop?
        for p in particles:
                f = compute_net_force(p)
                a = f / p.mass
                p.velocity += a * dt
                p.position += p.velocity * dt
```

- ***State* (x, v)**: all the things we need to update the position
  - $v' = v + \Delta v = v + (a * \Delta t)$
  - $x' = x + \Delta x = x + (v' * \Delta t\,)$
- The *state* of the *n* particles in $\mathbb{R}^3$ is:

$$(\mathbf{x}_1, \mathbf{v}_1, \mathbf{x}_2, \mathbf{v}_2, \ldots, \mathbf{x}_\mathrm{N}, \mathbf{v}_\mathrm{N}), \qquad \boldsymbol{x_i}, \boldsymbol{v_i} \in \mathbb{R}^3$$

  - The space of state is called phase space sometimes

Dynamical System

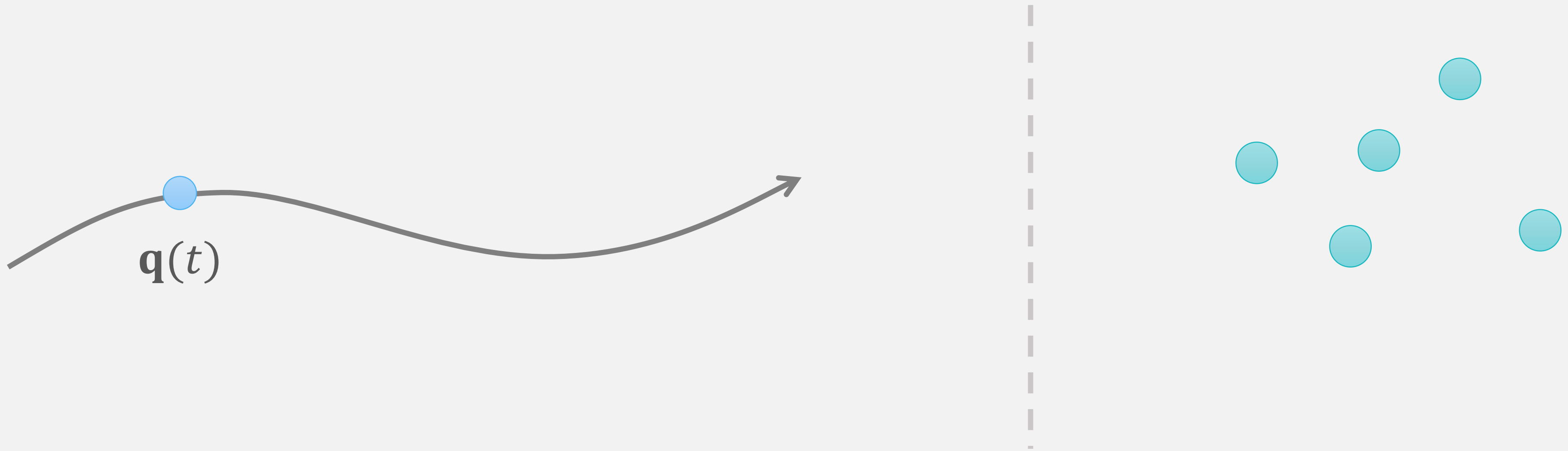
- *Configuration*: $\mathbf{q}(t)$
  - This is what we want to solve (e.g. the *state*) in generalized perspective
- Velocity (time derivative): $\dot{\mathbf{q}} \coloneqq \frac{d}{dt}\mathbf{q}$
- Mass: M
- Forces: F
- Constraints: $\mathrm{c}(\mathbf{q}, \dot{\mathbf{q}}, t) = 0$

Generalized Coordinates

- Often describing systems with many moving pieces
- The configuration (state vector) is a single vector with all states:

  $$\mathbf{q}(t) = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$$

- The motion of the system is the trajectory of *state* vector

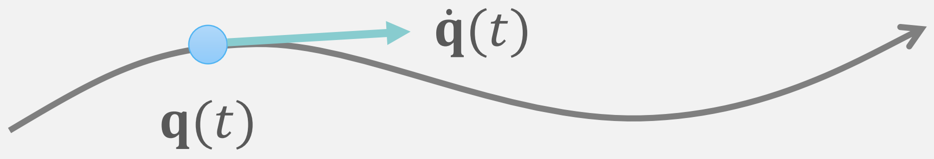


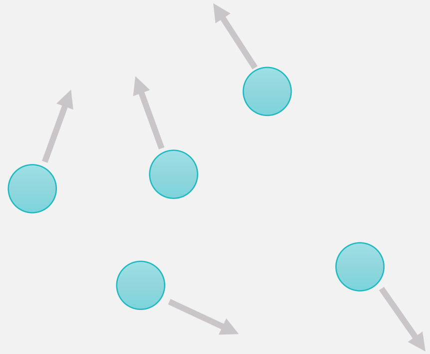
Generalized Velocity

- The time derivative of generalized coordinates:

  $\dot{\mathbf{q}} = (\dot{x}_1, \dot{x}_2, \dots, \dot{x}_n)$

  - Tells us how the system evolves in small amount of time





Why Using Numerical Integration?

$$\dot{q}(t) = \frac{d}{dt} q(t) = f\big(q(t)\big)$$

$$\begin{aligned} q(t+h) &= q(t) + \int_t^{t+h} \dot{q}(t) dt \\ &= q(t) + \int_t^{t+h} f(q(t)) dt \end{aligned}$$

no analytic form usually!

Numerical Integration

- Key idea: replace derivatives with differences

$$\frac{d}{dt}q(t) = f(q(t))$$

$$\Downarrow$$

new configuration (unknown)

current configuration

$$\frac{q_{k+1} - q_k}{h} = f(q(t_?))$$

evaluate $\boldsymbol{q}$ *at* $\boldsymbol{t_{k+1}}$ *or* $\boldsymbol{t_k}$*?*

"time step", i.e., the time interval between $q_{k+1} - q_k$

Ordinary Differential Equation (ODE)

- *'Ordinary'* means that it only depends on time

$$\frac{d}{dt}u(t) = -au, \qquad a > 0$$

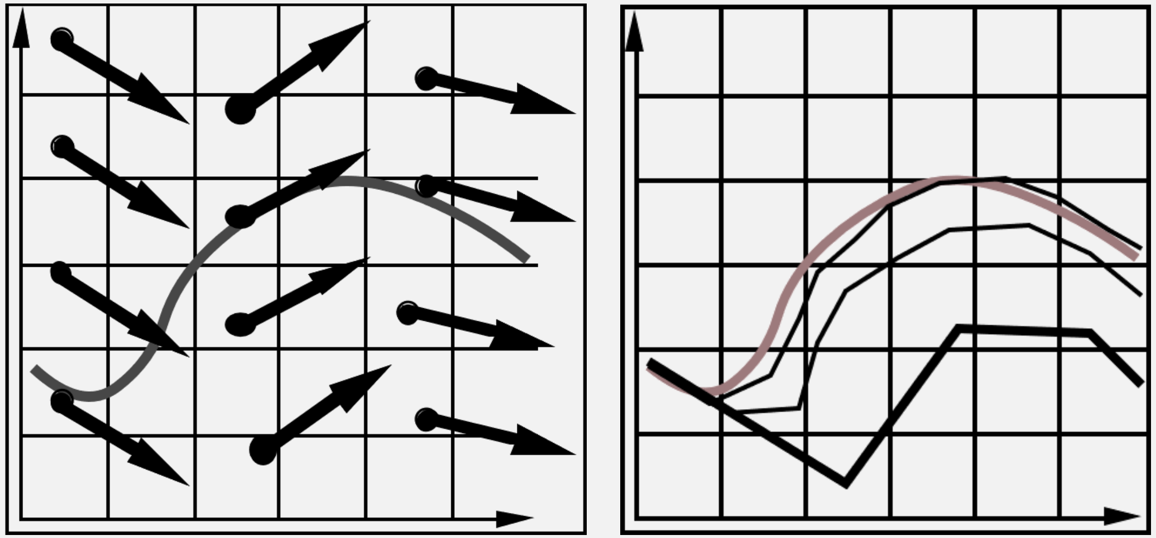
the exact solution is: $u(t) = e^{-at}$

it should decay: when $t \to \infty, \; u \to 0$

Forward Euler Method

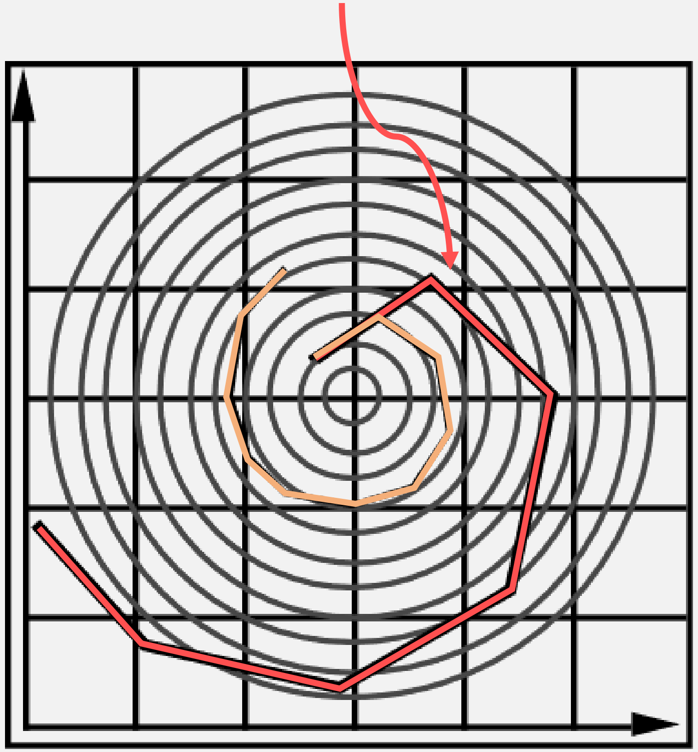
$$q_{k+1} - q_k = h * f(q_k)$$

- Evaluate velocity at current configuration
  - Easy to implement, it's straight forward
- Unstable with large step size h
  - $h \uparrow \Rightarrow error \uparrow$
  - Overshooting or 'blowing up'



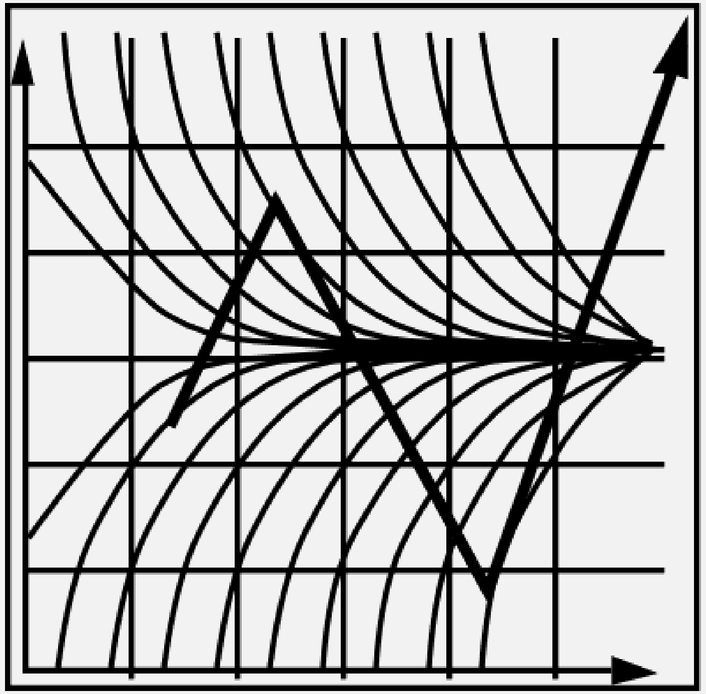
Two Problems of Forward Euler

*Shrinking the step size will slow the drift, but **NEVER** eliminate it!*



Inaccuracy

$$h \uparrow \Rightarrow error \uparrow$$



Instability

Physically Based Modeling, Witkin and Baraff, SIG'01 Course Notes

Example of Forward Euler Integration

$$\dot{u} = -au, \qquad a > 0$$

$$u_{k+1} - u_k = h(-au_k)$$

$$u_{k+1} = (1 - ha)u_k$$

after n steps…

$$\boxed{u_n = (1 - ha)^n u_0}$$

converges **ONLY** if $|1 - ha| < 1$

if a is large, h needs to be very small!

How Accurate is Forward Euler?

- The Taylor expansion of $x(t)$ at t_0 is
  - $x(t_0 + h) = x(t_0) + \dot{x}(t_0)h + \ddot{x}(t_0)\frac{h^2}{2!} + \cdots + \frac{\partial^n x(t_0)}{\partial t^n}\frac{h^n}{n!} + \cdots$
  - The ***order*** of integration schemes: how many levels of derivatives represent in the expansion
- Forward Euler is a *first-order* method and its error term is
  - $\ddot{x}(t_0)\frac{h^2}{2!} + \cdots + \frac{\partial^n x(t_0)}{\partial t^n}\frac{h^n}{n!} + \cdots \approx O(h^2)$
  - It's only correct if all derivatives beyond the first is zero!
  - In general, an order n method will have $O(h^{n+1})$ error
- Reduce the step size to $h/2$, the error becomes $1/4$
  - But the accumulated error is still linearly proportional to h (since we need double the accumulation steps)

Runge-Kutta Methods

- Key idea: use the *slope* at more than one point to extrapolate the value at the future time step

$$\dot{q}(t) = \frac{d}{dt}q(t) = f\big(t, q(t)\big)$$

approximate by using quadrature

$$q(t+h) = q(t) + \int_t^{t+h} \dot{q}(t)dt \approx q(t) + \boxed{h\sum_i w_i f\big(t + v_i h, q(t + v_i h)\big)} \qquad v_i \in \mathbb{R}$$

Second Order Runge-Kutta

- Use previous slope k_1 to approximate the temporary q for k_2

$$k_1 = f(t_n, q_n)$$

$$k_2 = f(t_n + \alpha h, q_n + \beta h k_1) \Rightarrow \; k_2 \approx f(t_n, q_n) + \alpha h \frac{\partial f}{\partial t}(t_n, q_n) + \beta h k_1 \frac{\partial f}{\partial q}(t_n, q_n) + O(h^2)$$

$$q_{n+1} = q_n + h(w_1 k_1 + w_2 k_2)$$

$$\Downarrow$$

$$q_{n+1} = q_n + h w_1 f(t_n, q_n) + h w_2 \left(f(t_n, q_n) + \alpha h \frac{\partial f}{\partial t}(t_n, q_n) + \beta h \frac{\partial f}{\partial q} \frac{\partial q}{\partial t} (t_n, q_n) \right)$$

$$= q_n + h(w_1 + w_2) f(t_n, q_n) + h^2 w_2 \left(\alpha f_t + \beta f_q f \right)$$

Second Order Runge-Kutta (Cont.)

- Match the coefficients to the Taylor expansion:

$$q(t+h) = q(t) + h\frac{dq}{dt} + \frac{h^2}{2}\frac{d^2q}{dt^2} + O(h^3)$$

$$\boxed{\frac{d^2q}{dt^2} = \frac{df(t,q(t))}{dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial q}\frac{\partial q}{\partial t} = f_t + f_q f}$$

$$= q(t) + hf(t_n, q_n) + \frac{h^2}{2}\left(f_t + f_q f\right)(t_n, q_n) + O(h^3)$$

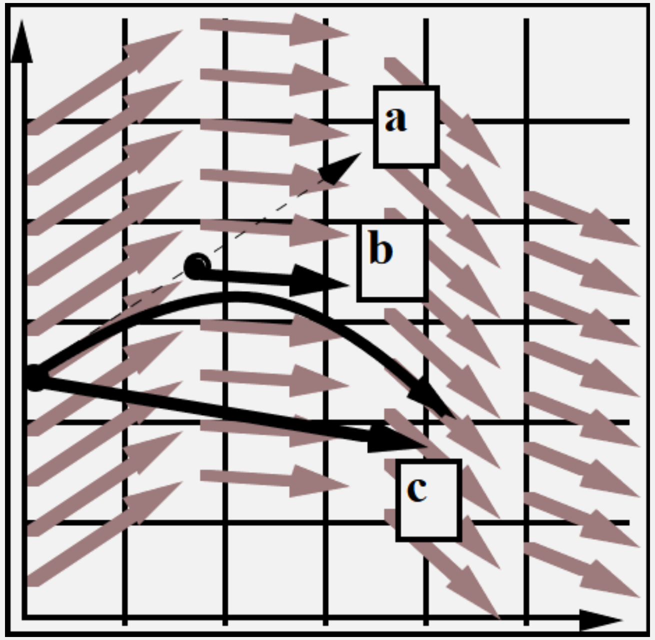
$$q_{n+1} = q_n + h(w_1 + w_2)f(t_n, q_n) + h^2 w_2\left(\alpha f_t + \beta f_q f\right)(t_n, q_n)$$

- Get $\boxed{w_1 + w_2 = 1,\ \alpha w_2 = \frac{1}{2}, \beta w_2 = \frac{1}{2}}$
  - There are infinitely many choices!
  - Classical 2nd order RK2: $w_1, w_2 = \frac{1}{2}, \alpha, \beta = 1$

Midpoint Method (RK2)

$$q_{n+1} = q_n + h(w_1 + w_2)f(t_n, q_n) + h^2 w_2\left(\alpha f_t + \beta f_q f\right)(t_n, q_n) + O(h^3)$$

- By choosing $\alpha = \beta = \frac{1}{2},\ w_1 = 0,\ w_2 = 1$
- Then $q_{n+1} = q_n + hf(t_n, q_n) + \frac{h^2}{2}\left(f_t + f_q f\right)(t_n, q_n)$
- Integration steps:
  a. Compute an Euler step
  $$hk_1 = h * f(t_k, q_k)$$
  b. Evaluate f at the midpoint
  $$k_2 = f\left(t_k + \frac{h}{2}, q_k + \frac{hk_1}{2}\right)$$
  c. Take full step with the slope at the midpoint
  $$q_{k+1} = q_k + h * k_2$$



Symplectic Euler

- Key ideas
  - Update velocity using current configuration first
  - Use that new velocity to update next configuration
- Used by many real-time physics engines like Bullet
  - Also known as *semi-explicit Euler, semi-implicit Euler, Euler-Cromer and Newton-Stormer-Verlet* (so many names …)
- Benefits: stable and preserving energy

Symplectic Euler

- Key ideas
  - Update velocity using current configuration first
  - Use that new velocity to update next configuration
- Used by many real-time physics engines like Bullet
  - Also known as *semi-explicit Euler, semi-i*... *Newton-Stormer-Verlet* (so many nam...
- Benefits: stab...

Read More

Integration Basics, Glenn Fiedler.

Adaptive Step Sizes

- What is ‘good’ step size for explicit integrator?
  - The largest step size with acceptable integration error
- Vary step size h over the course of solving ODE
  - Compute two estimates of $x(t_0 + h)$
    1. x_a: take one full step size
    2. x_b: take two steps with size h/2
  - Measure the error $e = |x_a - x_b|$
  - Scale the step size w.r.t. the error bound and acceptable error

Adaptive Step Sizes (Cont.)

- Suppose the error bound is $O(h^2)$ and the acceptable error is 10^{-4} per step
- If current error is 10^{-8}, we could increase step size to $100h$

$$\left(\frac{10^{-4}}{10^{-8}}\right)^{\frac{1}{2}} h = 100h$$

- If current error is 10^{-3}, we have to decrease the step size to $0.316h$

$$\left(\frac{10^{-4}}{10^{-3}}\right)^{\frac{1}{2}} h \approx 0.316h$$

Backward Euler Method

$$q_{k+1} - q_k = h * f(q_{k+1})$$

- Evaluate function f at *new* (i.e., k+1) configuration
- Need to solve a linear system for each time step
- Unconditionally stable, even if h is ∞
  - But this does **NOT** mean it is accurate!
- Numercial dissipation
  - Converges to zero when h is large
  - Energy lost

Example of Backward Euler Integration

$$\dot{u} = -au, \qquad a > 0$$

$$u_{k+1} - u_k = h(-au_{k+1})$$

$$u_{k+1} = \frac{1}{1+ah} u_k$$

after n steps…

$$u_n = \left(\frac{1}{1+ah}\right)^n u_0$$

$a, h > 0 \Rightarrow 1 + ah > 1$, always converges!

'Stiff' System

- Suppose we are trying to fix a particle movement in y-axis by

$$\dot{\mathbf{X}}(\mathbf{t}) = \frac{d}{dt}\begin{pmatrix}\mathbf{x}(t)\\ \mathbf{y}(t)\end{pmatrix} = \begin{pmatrix}-\mathbf{x}(t)\\ -k\mathbf{y}(t)\end{pmatrix}$$

  - k is a large positive number => a quick decay along y-axis

$$\mathbf{X}_{\mathbf{new}} = \mathbf{X}_{\mathbf{0}} + h\dot{\mathbf{X}}(t_0) = \begin{pmatrix}(1-h)\mathbf{x}_{\mathbf{0}}\\ (1-hk)\mathbf{y}_{\mathbf{0}}\end{pmatrix}$$

- It converges if $|1 - hk| < 1 \Rightarrow hk < 2$
  - Where k is large, thus h has to be ***small***
- *Stiff:* numerically unstable, unless the step size has to be small

Numerical Integration of Particle System

- Suppose $\ddot{\mathbf{x}}(t) = f\big(\mathbf{x}(t), \dot{\mathbf{x}}(t)\big)$
- From Newton 2nd law of motion:

$$\frac{d}{dt}\begin{pmatrix}\mathbf{x}\\ \mathbf{v}\end{pmatrix} = \begin{pmatrix}\mathbf{v}(t)\\ f\big(\mathbf{x}(t), \mathbf{v}(t)\big)\end{pmatrix}$$

which t? at current or new configuration?

Forward Euler

$$\begin{pmatrix} \Delta x \\ \Delta v \end{pmatrix} = h \begin{pmatrix} v_n \\ f(x_n, v_n) \end{pmatrix}$$

$$x_{n+1} = x + \Delta x = x + hv_n$$
$$v_{n+1} = v_0 + hf(x_n, v_n)$$

- Force function could be:
  - Unary forces like gravity and drag
  - n-ary forces like springs
  - Forces of spatial interaction
    - Attraction and respulsion between particles

Backward Euler

$$\frac{d}{dt}\begin{pmatrix} x \\ v \end{pmatrix} = \begin{pmatrix} v(t) \\ f(x(t), v(t)) \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} \Delta x \\ \Delta v \end{pmatrix} = h \begin{pmatrix} v_n + \Delta v \\ f(x_n + \Delta x, v_n + \Delta v) \end{pmatrix}$$

- Suppose $\ddot{x}(t) = f(x(t), \dot{x}(t))$ and assume function f is smooth, then f could be approximated by Taylor expansion:

$$\begin{aligned} f(x_n + \Delta x, v_n + \Delta v) &= f(\boldsymbol{x_n}, \boldsymbol{v_n}) + \frac{\partial f}{\partial \boldsymbol{x}} \Delta x + \frac{\partial f}{\partial \boldsymbol{v}} \Delta v \\ &= f(\boldsymbol{x_n}, \boldsymbol{v_n}) + \frac{\partial f}{\partial \boldsymbol{x}} h(v_n + \Delta v) + \frac{\partial f}{\partial \boldsymbol{v}} \Delta v \end{aligned}$$

- Plug into the linear system and solve Δv:

$$\left(I - h^2 \frac{\partial f}{\partial x} - h \frac{\partial f}{\partial v} \right) \Delta v = f(x_n, v_n) + \frac{\partial f}{\partial x} h v_n \qquad then\ x_{n+1} = x_n + \Delta x = x_n + h(v_n + \Delta v)$$

Forward vs. Backward Euler

- Forward Euler is fast, simple, but not stable
  - Need to reduce the time step to avoid overshootings
- Backward Euler is unconditionaly stable with prices
  - Need to solve system of equations in each time step
  - Temporal details disappear, due to numerical disippation
    - Large time step would lose more energy?
- For 'stiff' system, backward Euler is more stable than forward Euler
- Total evaluation time is:

$$\boxed{\textit{number of time steps} \times \textit{the computation time in each step}}$$

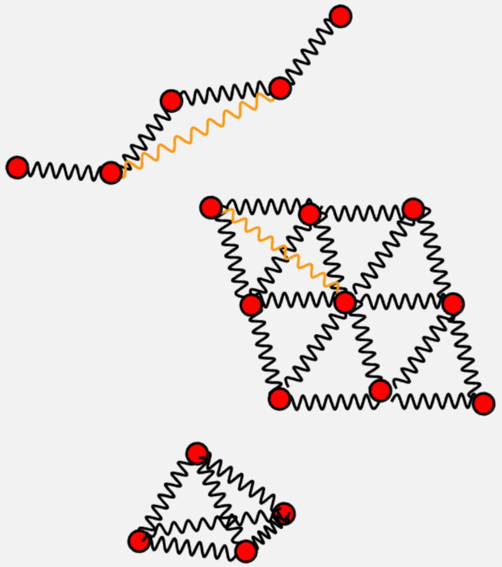
Mass-Spring System

Deformable Objects



Mass-Spring Structure

- A set of point masses connected by springs
- Each spring follows Hook's law and Newton's 3rd law of motion
- Structure for deformable objects:
  - Hair/rope: chain
    - Additional springs for bending and torsional resistance needed
  - Cloth: triangle mesh
    - Need extra springs to model bending
    - Low bending resistance but quite stiff w.r.t. stretching
  - Soft-body: tetrahedral mesh

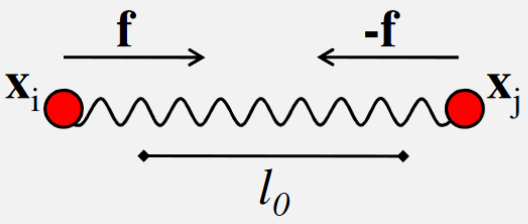


Physical Formulation

- Spring force

stiffness *rest length*

$$\mathbf{f}_\mathrm{i}^s = \mathbf{f}^s(\mathbf{x}_\mathrm{i}, \mathbf{x}_\mathrm{j}) = k_s \frac{\mathbf{x}_\mathrm{j} - \mathbf{x}_\mathrm{i}}{|\mathbf{x}_\mathrm{j} - \mathbf{x}_\mathrm{i}|}\left(|\mathbf{x}_\mathrm{j} - \mathbf{x}_\mathrm{i}| - l_0\right)$$



- Damping force

damping

$$\mathbf{f}_\mathrm{i}^\mathbf{d} = \mathbf{f}^\mathbf{d}(\mathbf{x}_\mathrm{i}, \mathbf{v}_\mathrm{i}, \mathbf{x}_\mathrm{j}, \mathbf{v}_\mathrm{j}) = k_d \left[(\mathbf{v}_\mathrm{j} - \mathbf{v}_\mathrm{i}) \cdot \left(\frac{\mathbf{x}_\mathrm{j} - \mathbf{x}_\mathrm{i}}{|\mathbf{x}_\mathrm{j} - \mathbf{x}_\mathrm{i}|}\right)\right]\left(\frac{\mathbf{x}_\mathrm{j} - \mathbf{x}_\mathrm{i}}{|\mathbf{x}_\mathrm{j} - \mathbf{x}_\mathrm{i}|}\right)$$

- Unified spring force: $\mathbf{f}_\mathrm{i} = \mathbf{f}(\mathbf{x}_\mathrm{i}, \mathbf{v}_\mathrm{i}, \mathbf{x}_\mathrm{j}, \mathbf{v}_\mathrm{j}) = \mathbf{f}^s(\mathbf{x}_\mathrm{i}, \mathbf{x}_\mathrm{j}) + \mathbf{f}^\mathbf{d}(\mathbf{x}_\mathrm{i}, \mathbf{v}_\mathrm{i}, \mathbf{x}_\mathrm{j}, \mathbf{v}_\mathrm{j}) = -\mathbf{f}_\mathrm{j}$

Implicit Integration

forces of n particles: $\mathbf{f}: \mathbb{R}^{3n} \rightarrow \mathbb{R}^{3n}$

Mass matrix: $\mathbf{M} \in \mathbb{R}^{3n\times 3n}$

(diagonal matrix)

positions: $\mathbf{x} \in \mathbb{R}^{3n}$

$$\mathbf{Mv^{t+1}} = \mathbf{Mv^{t}} + h * \mathbf{f}\left(\mathbf{x^{t+1}}\right)$$

$$\Downarrow \quad \mathbf{x^{t+1}} = \mathbf{x^{t}} + h * \mathbf{v^{t+1}}$$

$$\mathbf{Mv^{t+1}} = \mathbf{Mv^{t}} + h * \mathbf{f}\left(\mathbf{x^{t}} + h * \mathbf{v^{t+1}}\right)$$

$$\Downarrow \quad \textit{Taylor expansion}$$

$$\mathbf{Mv^{t+1}} = \mathbf{Mv^{t}} + h\left[\mathbf{f}(\mathbf{x^{t}}) + \frac{\partial \mathbf{f}(\mathbf{x^{t}})}{\partial \mathbf{x}} * (h * \mathbf{v^{t+1}})\right]$$

Implicit Integration (Cont.)

$$\mathbf{M}\mathbf{v}^{\mathbf{t+1}} = \mathbf{M}\mathbf{v}^{\mathbf{t}} + h\left[\mathbf{f}(\mathbf{x}^{\mathbf{t}}) + \frac{\partial \mathbf{f}(\mathbf{x}^{\mathbf{t}})}{\partial \mathbf{x}} * (h * \mathbf{v}^{\mathbf{t+1}})\right]$$

$$= \mathbf{M}\mathbf{v}^{\mathbf{t}} + h\mathbf{f}(\mathbf{x}^{\mathbf{t}}) + h^2\mathbf{K}\mathbf{v}^{\mathbf{t+1}}$$

$$(\mathbf{M} - h^2\mathbf{K})\mathbf{v}^{\mathbf{t+1}} = \mathbf{M}\mathbf{v}^{\mathbf{t}} + h\mathbf{f}(\mathbf{x}^{\mathbf{t}})$$

How to construct matrix $\mathbf{K}$*?*

$$\mathbf{x}^{\mathbf{t+1}} = \mathbf{x}^{\mathbf{t}} + h * \mathbf{v}^{\mathbf{t+1}}$$

Jacobian Matrix K

- Contains the derivatives of all *3n* force components w.r.t. all *3n* position components of the particles
  - Also called *tangent stiffness* matrix
- A spring force between particles *i and j* constructs *four* 3x3 sub-matrices:
  - Derivatives w.r.t. positions $\mathbf{x_i}$ *and* $\mathbf{x_j}$

$$\mathbf{K_{i,i}} = \frac{\partial}{\partial \mathbf{x_i}} \mathbf{f}^s(\mathbf{x_i}, \mathbf{x_j}) = k_s \frac{\partial}{\partial \mathbf{x_i}} \left((\mathbf{x_j} - \mathbf{x_i}) - l_0 \frac{\mathbf{x_j} - \mathbf{x_i}}{|\mathbf{x_j} - \mathbf{x_i}|} \right)$$

$$= k_s \left(-\mathbf{I} + l_0 \left[\frac{\mathbf{I} - (\mathbf{x_j} - \mathbf{x_i})(\mathbf{x_j} - \mathbf{x_i})^{\mathbf{T}}}{|\mathbf{x_j} - \mathbf{x_i}|^3} \right] \right)$$

$$= -\mathbf{K_{i,j}} = \mathbf{K_{j,j}} = -\mathbf{K_{j,i}}$$

$$\left(\frac{f}{g}\right)' = \frac{f'g - fg'}{g^2}$$

$$\frac{\partial |\mathbf{x}|}{\partial \mathbf{x}} = \left(\frac{\mathbf{x}}{|\mathbf{x}|}\right)^T = \hat{\mathbf{x}}^T$$

$$\frac{\partial \hat{\mathbf{x}}}{\partial \mathbf{x}} = \frac{\mathbf{I} \cdot |\mathbf{x}| - \mathbf{x} \cdot \hat{\mathbf{x}}^{\mathbf{T}}}{|\mathbf{x}|^2} = \frac{\mathbf{I} - \hat{\mathbf{x}} \cdot \hat{\mathbf{x}}^{\mathbf{T}}}{|\mathbf{x}|}$$

Read More: The Matrix Cookbook

Solve System Equations

$$(\mathbf{M} - h^2\mathbf{K})\mathbf{v}^{\mathbf{t+1}} = \mathbf{M}\mathbf{v}^{\mathbf{t}} + h\mathbf{f}(\mathbf{x}^{\mathbf{t}})$$

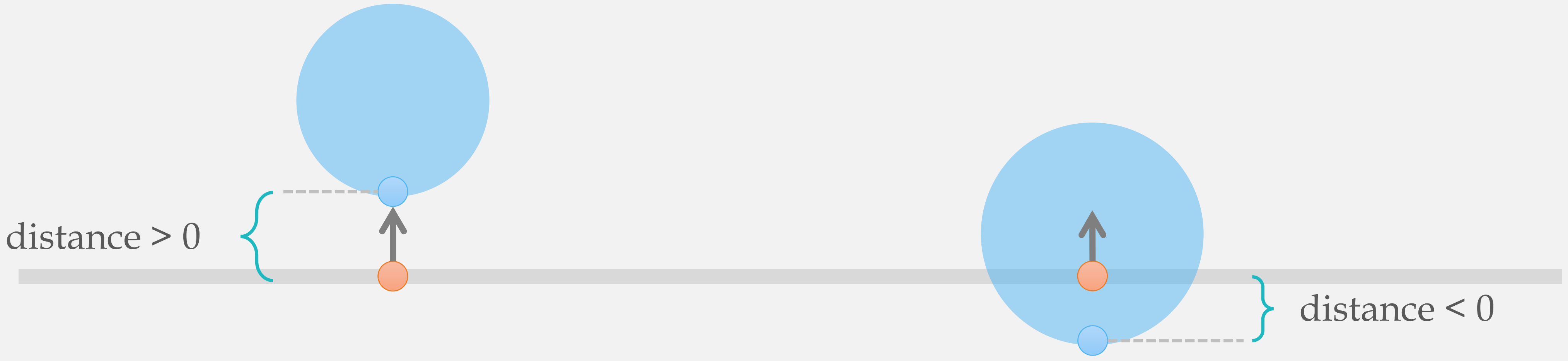
- For each time step
  - K is reset to zero at first
  - Then add each sub-matrix $\mathbf{K}_{\mathbf{i,j}}$ of each spring force to $\mathbf{K}$
- $\mathbf{K}$ changes at each time step
  - Using direct solver is not practical
  - Solving the system with iterative methods like *Conjugate Gradients* is more common

Three Phases of Collision Handling

- ***Collision detection***
  - Detect whether there is a collision or not
- ***Collision determination***
  - Find the collision point **p** and other infomation like contact normal
- ***Collision response***
  - Update simulation state of each entity such as position, velocity, etc.

Collision Detection

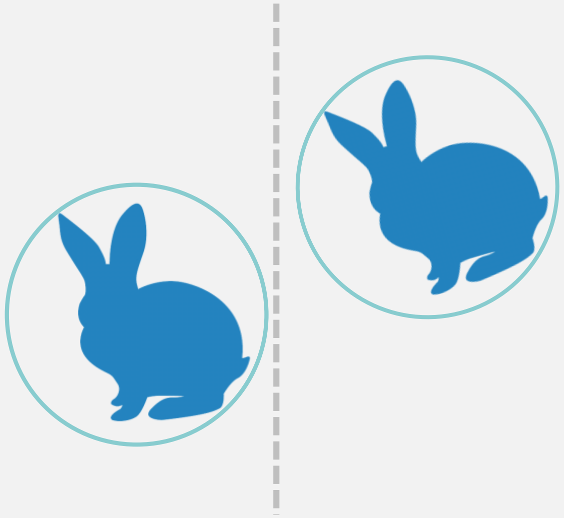
- Broad-phase
  - Using bound volumes to determine if there are any potential intersections
- Narrow-phase
  - Compute closest point, normal and distance/penetration



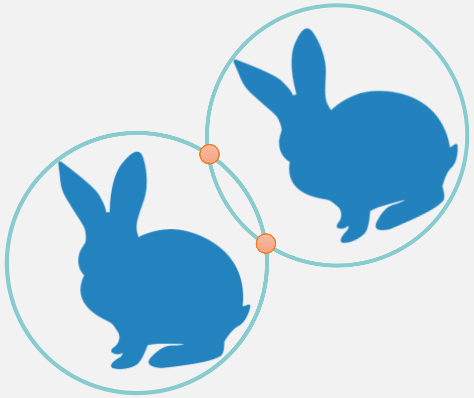
Recap

Types of Bounding Volumes

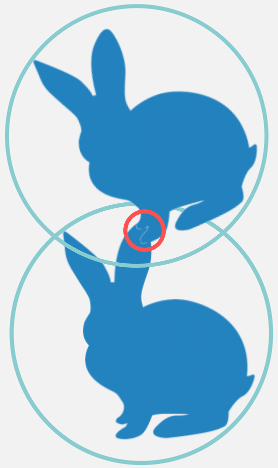
Collision Detection (Cont.)



Board-Phase: not collide



Board-Phase: collide
Narrow-Phase: no intersection

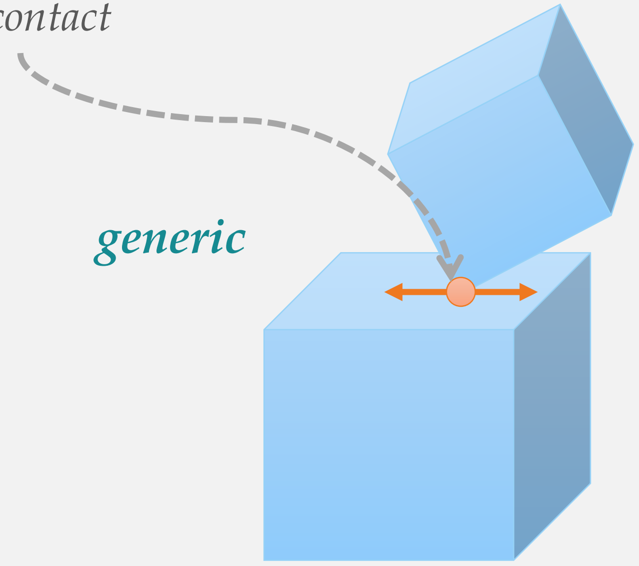


Board-Phase: collide
Narrow-Phase: intersection

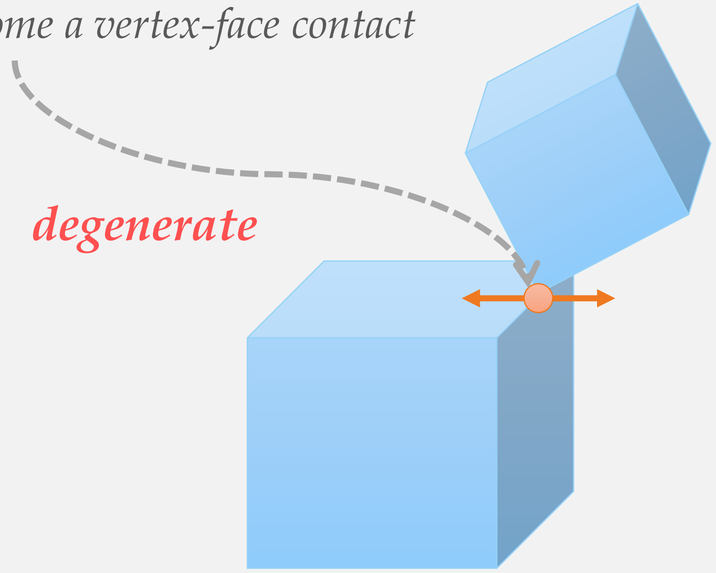
Type of Collisions

- *Genericity:* objects in 'general position'
  - A tiny perturbation does not change the nature of its interaction with the stuff around it
  - *Degenerate:* the objects are **NOT** in general position

when moving each object in the contact plane,
it's still a vertex-face contact



small movement would make such
vertex-edge contact become a vertex-face contact



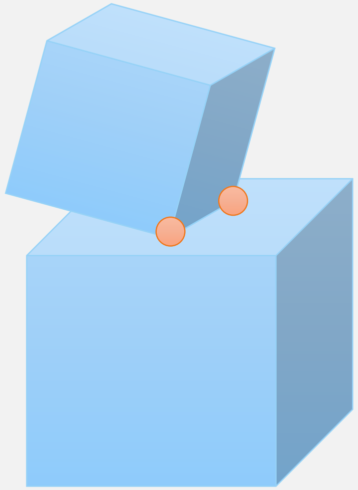
Type of Collisions (Cont.)

- There are 9 combinations of collision tests between geometry primitives (e.g. vertex, edge and face)
- But there are only two *generic* types of collisions:
  - Vertex-face
  - Edge-edge
- All other types of collisions are all *degenerate*, which means a very small perturbation will remove them as possibilities

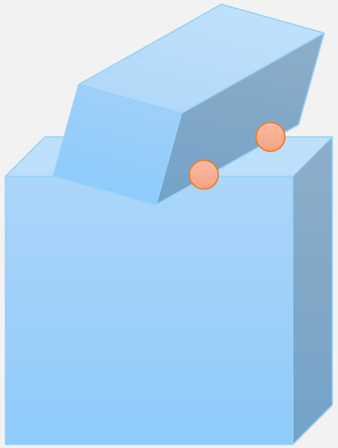
Type of Collisions (Cont.)

- Even degenerate collisions have generic collisons occurring with them
  - Thus it's critical to handle genric collisions
- An edge-face collision occurs when there are

two vertex-face contacts

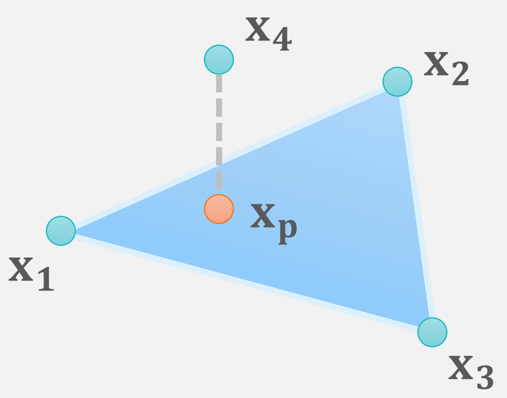


two edge-edge contacts



Vertex-Face Test

- Suppose all faces have been triangularized
- Find the closest point x_p on the triangle to point x_4



$$\mathbf{x_p} = w_1\mathbf{x_1} + w_2\mathbf{x_2} + w_3\mathbf{x_3} = \mathbf{x_3} + w_1\vec{\mathbf{x}}_{\mathbf{13}} + w_2\vec{\mathbf{x}}_{\mathbf{23}}$$

where $w_1 + w_2 + w_3 = 1,\quad \vec{\mathbf{x}}_{\mathbf{ij}} = \mathbf{x_j} - \mathbf{x_i}$

$$\vec{\mathbf{x}}_{\mathbf{4p}} = \vec{\mathbf{x}}_{\mathbf{43}} - w_1\vec{\mathbf{x}}_{\mathbf{13}} - w_2\vec{\mathbf{x}}_{\mathbf{23}} = \mathbf{0}$$

$$\begin{bmatrix}\vec{\mathbf{x}}_{\mathbf{13}} & \vec{\mathbf{x}}_{\mathbf{23}}\end{bmatrix}\begin{bmatrix}w_1\\ w_2\end{bmatrix} = \begin{bmatrix}\vec{\mathbf{x}}_{\mathbf{43}}\end{bmatrix}$$

Underdetermined! Use normal equation $A^TAx = A^Tb \Rightarrow$

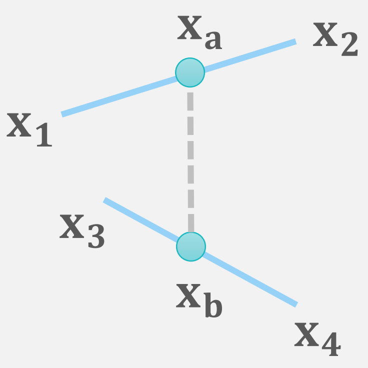
$$\begin{bmatrix}\vec{\mathbf{x}}_{\mathbf{13}}\\ \vec{\mathbf{x}}_{\mathbf{23}}\end{bmatrix}\begin{bmatrix}\vec{\mathbf{x}}_{\mathbf{13}} & \vec{\mathbf{x}}_{\mathbf{23}}\end{bmatrix}\begin{bmatrix}w_1\\ w_2\end{bmatrix} = \begin{bmatrix}\vec{\mathbf{x}}_{\mathbf{13}}\\ \vec{\mathbf{x}}_{\mathbf{23}}\end{bmatrix}\begin{bmatrix}\vec{\mathbf{x}}_{\mathbf{43}}\end{bmatrix}$$

$$\begin{bmatrix}\vec{\mathbf{x}}_{\mathbf{13}}\cdot\vec{\mathbf{x}}_{\mathbf{13}} & \vec{\mathbf{x}}_{\mathbf{13}}\cdot\vec{\mathbf{x}}_{\mathbf{23}}\\ \vec{\mathbf{x}}_{\mathbf{23}}\cdot\vec{\mathbf{x}}_{\mathbf{13}} & \vec{\mathbf{x}}_{\mathbf{23}}\cdot\vec{\mathbf{x}}_{\mathbf{23}}\end{bmatrix}\begin{bmatrix}w_1\\ w_2\end{bmatrix} = \begin{bmatrix}\vec{\mathbf{x}}_{\mathbf{13}}\cdot\vec{\mathbf{x}}_{\mathbf{43}}\\ \vec{\mathbf{x}}_{\mathbf{23}}\cdot\vec{\mathbf{x}}_{\mathbf{43}}\end{bmatrix}$$

Edge-Edge Test

$$\mathbf{x_a} = \mathbf{x_1} + a\vec{\mathbf{x}}_{21}, \qquad \mathbf{x_b} = \mathbf{x_3} + b\vec{\mathbf{x}}_{43}, \qquad a, b \in [0,1]$$

$$\mathbf{x_b} - \mathbf{x_a} = \vec{\mathbf{x}}_{31} + b\vec{\mathbf{x}}_{43} - a\vec{\mathbf{x}}_{21} \;\Rightarrow \begin{bmatrix}\vec{\mathbf{x}}_{21} & -\vec{\mathbf{x}}_{43}\end{bmatrix}\begin{bmatrix}a \\ b\end{bmatrix} = \begin{bmatrix}\vec{\mathbf{x}}_{31}\end{bmatrix}$$



Apply normal equation $A^T A x = A^T b$ to find the closest point

$$\begin{bmatrix}\vec{\mathbf{x}}_{21} \\ -\vec{\mathbf{x}}_{43}\end{bmatrix}\begin{bmatrix}\vec{\mathbf{x}}_{21} & -\vec{\mathbf{x}}_{43}\end{bmatrix}\begin{bmatrix}a \\ b\end{bmatrix} = \begin{bmatrix}\vec{\mathbf{x}}_{21} \\ -\vec{\mathbf{x}}_{43}\end{bmatrix}\begin{bmatrix}\vec{\mathbf{x}}_{31}\end{bmatrix}$$

$$\begin{bmatrix}\vec{\mathbf{x}}_{21}\cdot\vec{\mathbf{x}}_{21} & -\vec{\mathbf{x}}_{21}\cdot\vec{\mathbf{x}}_{43} \\ -\vec{\mathbf{x}}_{43}\cdot\vec{\mathbf{x}}_{43} & \vec{\mathbf{x}}_{43}\cdot\vec{\mathbf{x}}_{43}\end{bmatrix}\begin{bmatrix}a \\ b\end{bmatrix} = \begin{bmatrix}\vec{\mathbf{x}}_{21}\cdot\vec{\mathbf{x}}_{31} \\ -\vec{\mathbf{x}}_{43}\cdot\vec{\mathbf{x}}_{31}\end{bmatrix}$$

Continuous Intersection Tests

- Assume all vertices move with constant velocity over the time step
  - Suppose contact occurs at time t
  - $\mathbf{x_i}(t)$ is the vertex $\mathbf{x_i}$ at time *t*, $\mathbf{v_i}$ is the velocity of $\mathbf{x_i}$
  - $\vec{\mathbf{x}}_{\mathbf{ij}}(t) = \mathbf{x_{ij}} + t\vec{\mathbf{v}}_{\mathbf{ij}}$
- For Vertex-Face contact, vertices $\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}$ are coplanar
  - $\mathbf{x_4}$ is contained in the triangle: $\mathbf{x_4}(\mathbf{t}) = u\vec{\mathbf{x}}_{\mathbf{31}}(t) + v\vec{\mathbf{x}}_{\mathbf{21}}(t)$
    - This is a non-linear system and hard to solve
    - But we could apply use orthogonality:

a ***cubic*** equation for *t*,
solve it and plug it back to solve *u*, *v*

$$\vec{\mathbf{x}}_{\mathbf{41}}(t) \cdot \vec{\mathbf{N}} = \vec{\mathbf{x}}_{\mathbf{41}}(t) \cdot \left(\vec{\mathbf{x}}_{\mathbf{21}}(t) \times \vec{\mathbf{x}}_{\mathbf{31}}(t)\right)$$

$$= (\vec{\mathbf{x}}_{\mathbf{41}} + t\vec{\mathbf{v}}_{\mathbf{41}}) \cdot \left((\vec{\mathbf{x}}_{\mathbf{21}} + t\vec{\mathbf{v}}_{\mathbf{21}}) \times (\vec{\mathbf{x}}_{\mathbf{31}} + t\,\vec{\mathbf{v}}_{\mathbf{31}})\right) = 0$$

- For Edge-Edge contant, 4 points are coplanar

$$\left(\vec{\mathbf{x}}_{\mathbf{21}}(t) \times \vec{\mathbf{x}}_{\mathbf{43}}(t)\right) \cdot \vec{\mathbf{x}}_{\mathbf{13}}(t) = \mathbf{0}$$

Collision Response

- Given a surface particle $\mathbf{q}$ of object A is inside a tetrahedron $\mathbf{t}$ of an object B
- In the case of self-collision, two objects are identical
  - It's far from trivial to find a stable way to resolve such collision
- Decide the future position of the penetrated vertex $\mathbf{q}$
  - Move $\mathbf{q}$ to the closest surface point of object B
    - Not recommended due to stability issue
  - Move $\mathbf{q}$ back to where it penetrated object B
    - Backward ray casting along the surface normal at $\mathbf{q}$
- Then the collision response force is defined as $\mathbf{f}_{coll} = k(\mathbf{q}' - \mathbf{q})$
  - k is stiffness coefficient

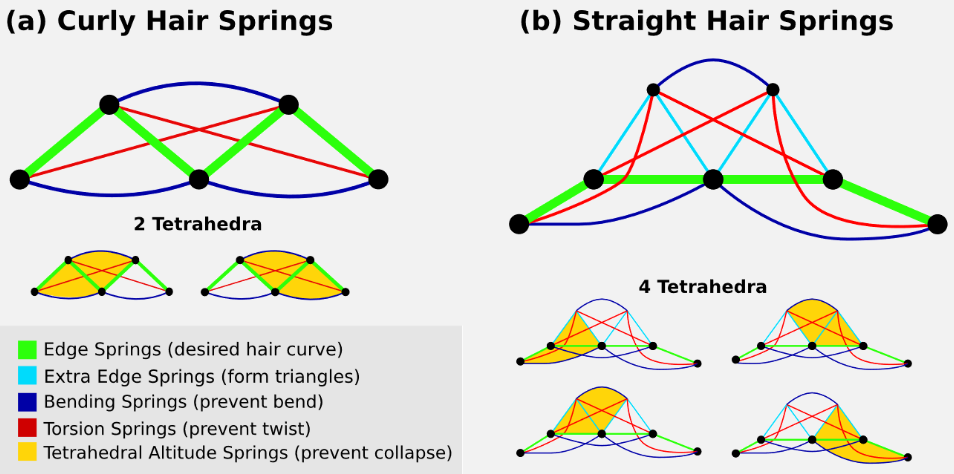
Practical Issues

- The object behavior depends on the spring network
- Hard to get desired behavior by tweaking the stiffness of relevant springs
- Simulate inextendibility would make system *‘stiff’*
- Can’t capture volumetric effects directly

Mass-Spring Model for Hair Simulation

- Hair is properties
  - Bending
  - Inextensibility
  - Torsion

[Iben et al., SCA'13]



[Selle et al., SIG'08]

Hair Troubles in Disney's *Tangled*

https://www.youtube.com/watch?v=9K-Gv4XVb10

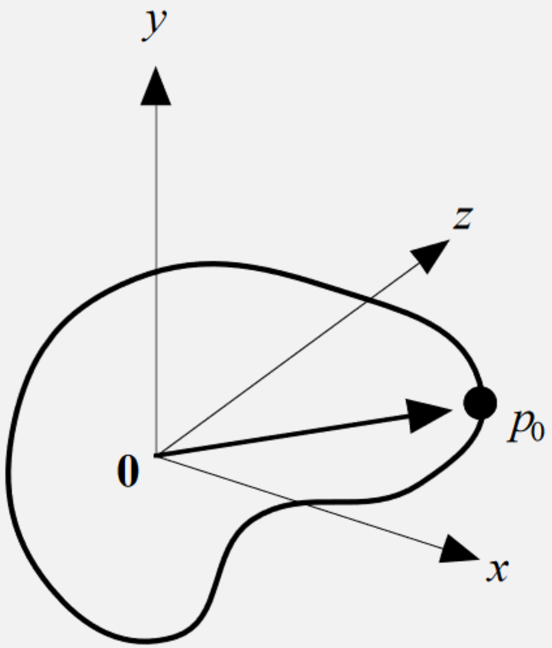
Rigid Body Dynamics

Rigid Body

- Collection of particles
- No deformation: the distance between any two particles is constant
- Rigid motion: rotation + translation => 6 DOFs

Orientation

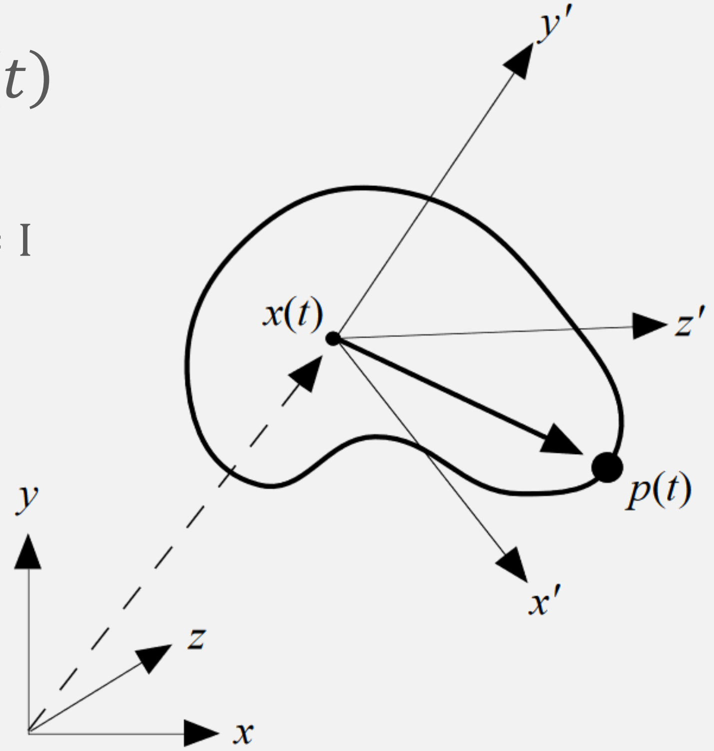
body space



$$p(t) = R(t)p_0 + x(t)$$

$$R(t)^T R(t) = R(t)R(t)^T = I$$

world space



Center of Mass and Body Velocity

- The coordinate of body's geometric center is
  - (0, 0, 0) in body space

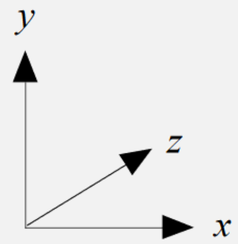
$$\sum m_i \boldsymbol{p_i}(0) = \boldsymbol{0}$$

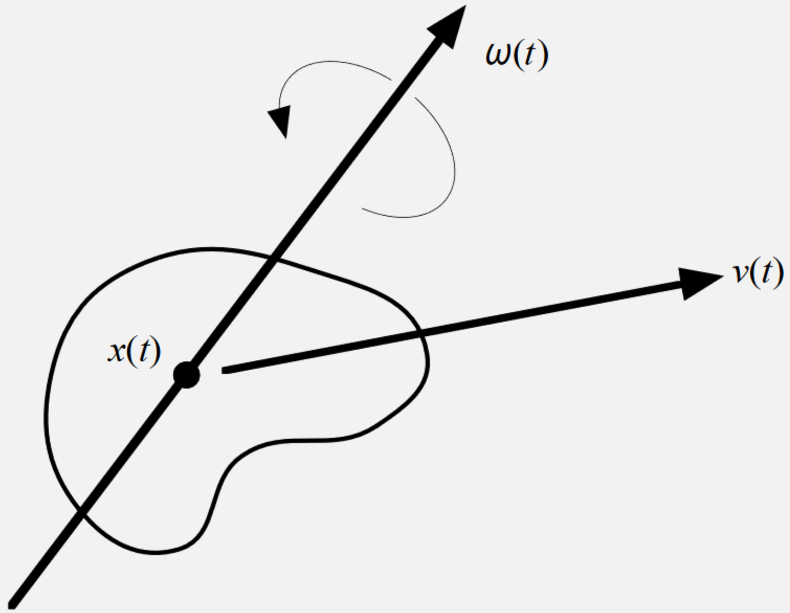
  - $x(t)$ in world space

$$x(t) = \frac{\sum m_i \boldsymbol{p_i}(\boldsymbol{t})}{\sum m_i}$$

- The body velocity in world space

$$v(t) = \frac{dx(t)}{dt} = \dot{x}(t)$$





Orientation

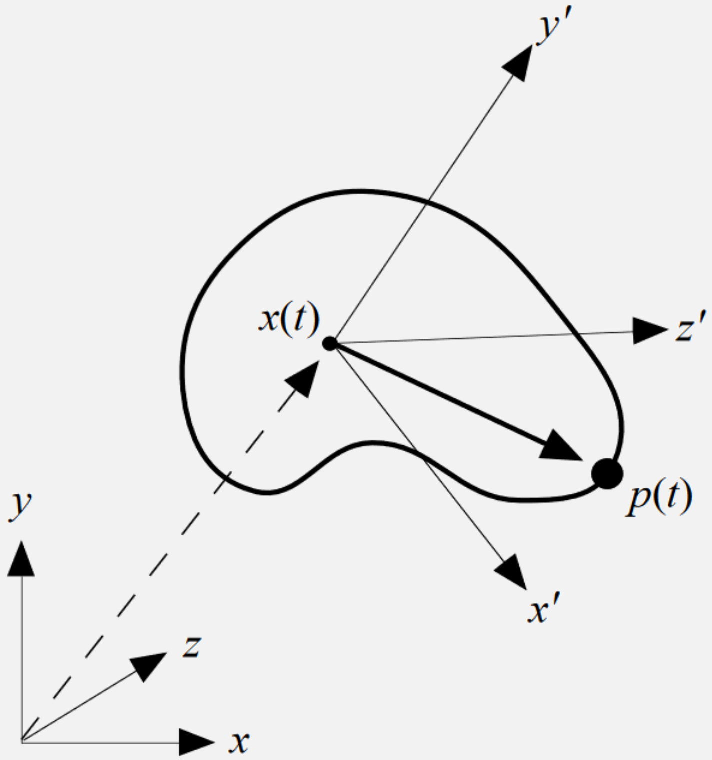
- The columns of $R(t)$:
  - The directions of the transformed x, y, z axes in body space at time t

$$p(t) = R(t)p(t) + x(t)$$

$$R(t)\begin{pmatrix}1\\0\\0\end{pmatrix} = \begin{pmatrix}r_{xx}\\r_{xy}\\r_{xz}\end{pmatrix} = x'$$

$$\Downarrow$$

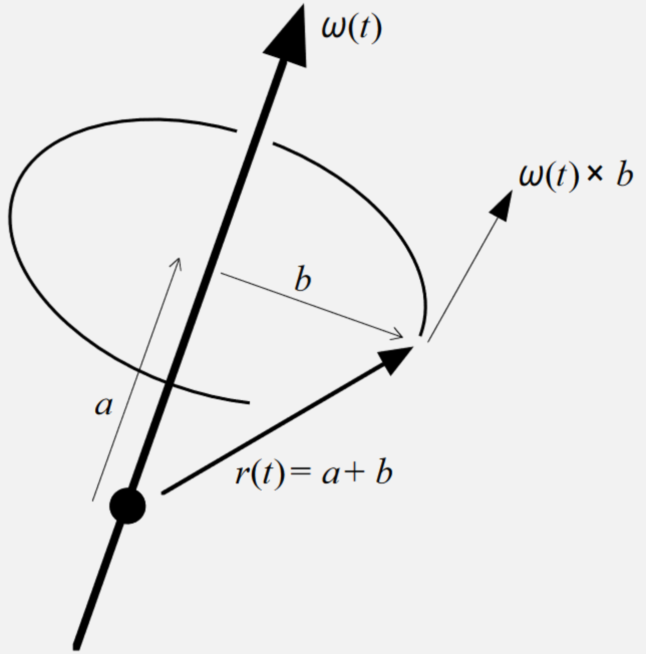
$$R(t) = \begin{pmatrix}r_{xx} & r_{yx} & r_{zx}\\r_{xy} & r_{yy} & r_{zy}\\r_{xz} & r_{yz} & r_{zz}\end{pmatrix}$$



Angular Velocity

- $\omega(t)$ is a vector represents the axis of spin
  - The magnitude $|\omega(t)|$ is the speed of spin
- How are $R(t)$ and $\omega(t)$ related?
  - $r(t) = a + b$, is a vector fixed to the body
    - translation-invariant
      - Its time derivative is only related to rotation
    - moving in a circle of radius b
  - $\dot{r}(t) = \omega(t) \times b = \omega(t) \times b + \underbrace{\boxed{\omega(t) \times a}}_{=0}$

$$= \omega(t) \times (a + b) = \omega(t) \times r(t)$$



Skew Matrix

$\forall \boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3$

$$\boldsymbol{a} \times \boldsymbol{b} = \begin{pmatrix} a_y b_z - b_y a_z \\ -a_x b_z + b_x a_z \\ a_x b_y - b_x a_y \end{pmatrix} = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} = \boldsymbol{a}^* \boldsymbol{b}$$

Change of Orientation

- The columns of $\dot{R}(t)$
  - The velocity with which the x, y, z axes are being transformed
- Given $\dot{r}(t) = \omega(t) \times r(t)$, and apply it to x, y, z-axis:

$$\dot{R}(t) = \left(\omega(t) \times \begin{pmatrix} r_{xx} \\ r_{xy} \\ r_{xz} \end{pmatrix} \quad \omega(t) \times \begin{pmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \end{pmatrix} \quad \omega(t) \times \begin{pmatrix} r_{zx} \\ r_{zy} \\ r_{zz} \end{pmatrix} \right)$$

$$= \omega^*(t) \left(\begin{pmatrix} r_{xx} \\ r_{xy} \\ r_{xz} \end{pmatrix} \quad \begin{pmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \end{pmatrix} \quad \begin{pmatrix} r_{zx} \\ r_{zy} \\ r_{zz} \end{pmatrix} \right) = \omega^*(t) R(t)$$

$$\boxed{\dot{R}(t) = \omega(t)^* R(t)}$$

Velocity of a Particle

$$r_i(t) = R(t)r_{0i} + x(t)$$

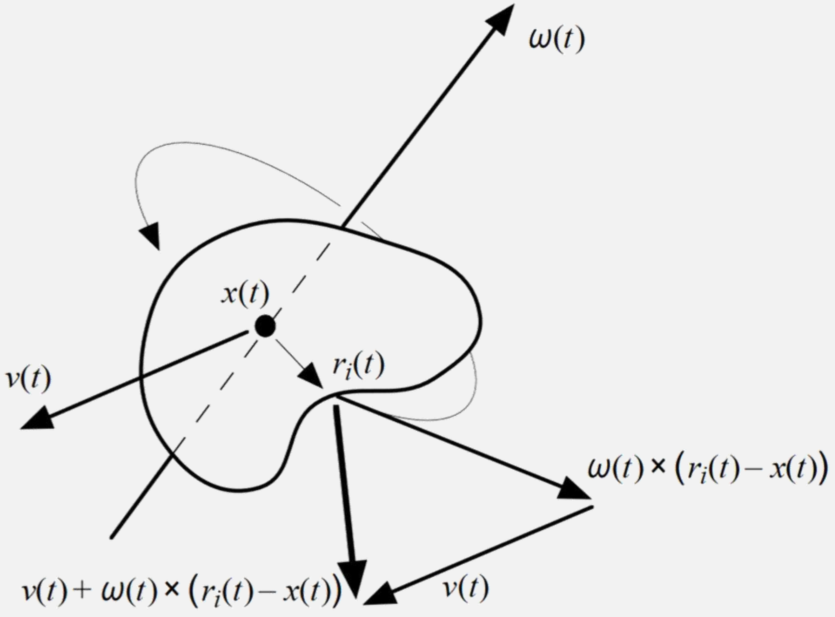
$\Downarrow$ *time derivative*

$$\dot{r}_i(t) = \dot{R}(t)r_{0i} + \dot{x}(t) = \omega(t) \times R(t)r_{0i} + v(t)$$

$$= \omega(t) \times (R(t)r_{0i} + x(t) - x(t)) + v(t)$$

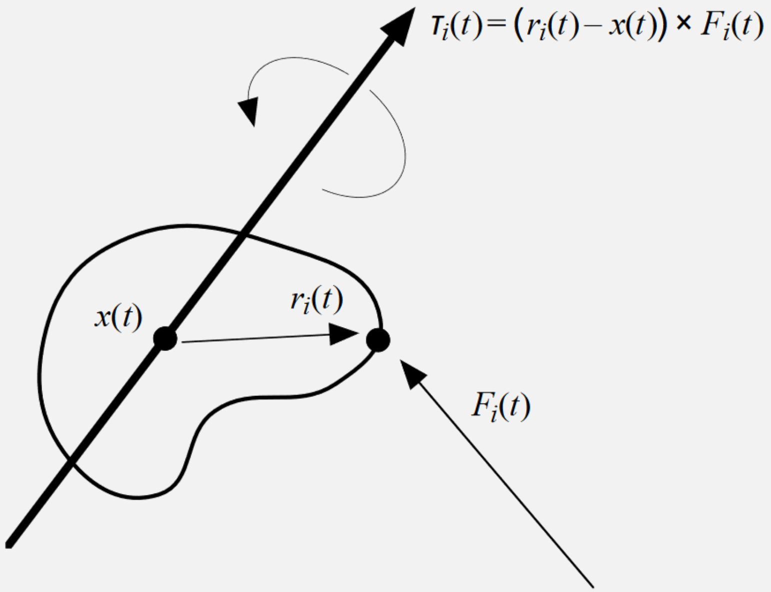
$$= \omega(t) \times (r_i(t) - x(t)) + v(t)$$

= *angular component* + *linear component*



Force and Torque

- Conceptual model: external force acting on a particular *particle* of the body
- The total external force: $F(t) = \sum F_i(t)$
  - Convey no information about where the various forces acting on
- The external torque acting on the ith particle
  - $\tau_i(t) = (r_i(t) - x(t)) \times F_i(t)$
- Total external torque provide some information about the distribution of the forces $F_i(t)$ over the body
  - $\tau(t) = \sum \tau_i(t) = \sum (r_i(t) - x(t)) \times F_i(t)$



Linear Momentum

- For single particle p, its linear momentum is $p_i = m_i v_i$
- While the total linear momentum of a rigid body at time t is

$$\begin{aligned} P(t) &= \sum_i m_i \dot{r}_i(t) \\ &= \sum_i \left(m_i v(t) + m_i \omega_i(t) \times \left(r_i(t) - x(t) \right) \right) \\ &= \sum_i m_i v(t) + \omega_i(t) \times \overbrace{\boxed{\sum_i m_i \left(r_i(t) - x(t) \right)}}^{=0} \\ &= M v(t) \end{aligned}$$

- The change in linear momentum is the total force exerting on the body

$$\dot{v}(t) = \frac{\dot{P}(t)}{M} \Rightarrow \dot{P}(t) = F(t)$$

Angular Momentum

- Angular momentum of a single particle is $L_i = r_i \times (m_i v_i) = r_i \times p_i$
- All particles within the body has the same angular velocity ω, thus their linearly velocity $v_i = \omega \times r_i$
- The total angular momentum is

$$
\begin{aligned}
L(t) &= \sum_i r_i(t) \times m_i v_i(t) \\
&= \sum_i r_i(t) \times m_i(\omega \times r_i(t)) \\
&= \sum_i -m_i r_i(t) \times (r_i(t) \times \omega(t)) \\
&= -m_i r_i^*(t) r_i^*(t)\omega = I(t)\omega(t)
\end{aligned}
$$

The Inertia Tensor

$\boxed{\text{Let } r_i' = r_i(t) - x(t)}$

$$I(t) = \sum \begin{pmatrix} m_i\left(r_{iy}'^2 + r_{iz}'^2\right) & -m_i r_{ix}' r_{iy}' & -m_i r_{ix}' r_{iz}' \\ -m_i r_{iy}' r_{ix}' & m_i\left(r_{ix}'^2 + r_{iz}'^2\right) & -m_i r_{iy}' r_{iz}' \\ -m_i r_{iz}' r_{ix}' & -m_i r_{iz}' r_{iy}' & m_i\left(r_{ix}'^2 + r_{iy}'^2\right) \end{pmatrix}$$

$$= \sum m_i r_i'^T r_i' \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \boxed{\begin{pmatrix} m_i r_{ix}'^2 & m_i r_{ix}' r_{iy}' & m_i r_{ix}' r_{iz}' \\ m_i r_{iy}' r_{ix}' & m_i r_{iy}'^2 & m_i r_{iy}' r_{iz}' \\ m_i r_{iz}' r_{ix}' & m_i r_{iz}' r_{iy}' & m_i r_{iz}'^2 \end{pmatrix}}$$

$$\begin{pmatrix} r_{ix}' \\ r_{iy}' \\ r_{iz}' \end{pmatrix} \begin{pmatrix} r_{ix}' & r_{iy}' & r_{iz}' \end{pmatrix} = r' r'^T$$

$$= \sum m_i \left(\left(r_i'^T r_i' \right) \mathbf{1} - r' r'^T \right)$$

$$\Downarrow$$

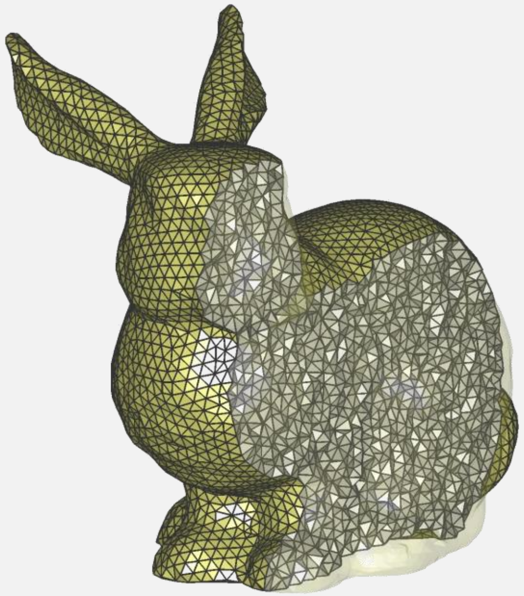
$$I(t) = R(t) \left(\sum m_i \left(\left(r_{0i}^T r_{0i} \right) \mathbf{1} - r_{0i} r_{0i}^T \right) \right) R(t)^T = R(t) I_{body} R(t)^T$$

precompute it!

$\boxed{r_i' = R(t) r_{0i}, \qquad R(t)R(t)^T = \mathbf{1}}$

Approximating I_{body}

- Goal: discreteize the mesh volume and accumulate internia tensor
- Bounding boxes, spheres
  - Simple but may not fit to geometry properly
- Convex decomposition
  - Tetrahedralized mesh



Forward Euler Integration

$$\frac{d}{dt}\mathbf{X}(t) = \frac{d}{dt}\begin{pmatrix} x(t) \\ R(t) \\ P(t) \\ L(t) \end{pmatrix} = \begin{pmatrix} v(t) \\ \omega(t)^{*}R(t) \\ F(t) \\ \tau(t) \end{pmatrix}$$

$$v(t) = P(t)/M$$

$$I(t) = R(t)I_{body}R(t)^{T}$$

$$\omega(t) = I(t)^{-1}L(t)$$

initialization

$$M \leftarrow \sum_i m_i$$

$$\bar{x}_0 \leftarrow \frac{\sum_i x_{0i}}{M}$$

$$r_{0i} \leftarrow x_{0i} - \bar{x}_0$$

$$I_{body}^{-1} \leftarrow \left(-\sum_i m_i r_{0i}^{*} r_{0i}^{*}\right)^{-1}$$

$$initlaize\ \mathbf{x}, \mathbf{v}, \mathbf{R}, \mathbf{L}$$

$$I^{-1} \leftarrow R I_{body}^{-1} R^{T}$$

$$\omega \leftarrow I^{-1}L$$

simulation loop

$$F \leftarrow \sum_i f_i$$

$$\tau \leftarrow \sum_i r_i \times f_i$$

$$x \leftarrow x + v\Delta t$$

$$v \leftarrow v + \frac{F}{M}\Delta t$$

$$R \leftarrow R + \omega^{*}R\Delta t$$

$$L \leftarrow L + \tau\Delta t$$

$$I^{-1} \leftarrow R I_0^{-1} R^{T}$$

$$\omega \leftarrow I^{-1}L$$

$$r_i \leftarrow R r_{0i}$$

$$x_i \leftarrow x + r_i$$

$$v_i \leftarrow v + \omega \times r_i$$

Rotation Matrix vs. Unit Quaternion

- Most import reason to avoid rotation matrix is the *numerical drift* in

$$\dot{R}(t) = \omega^*(t)R(t)$$

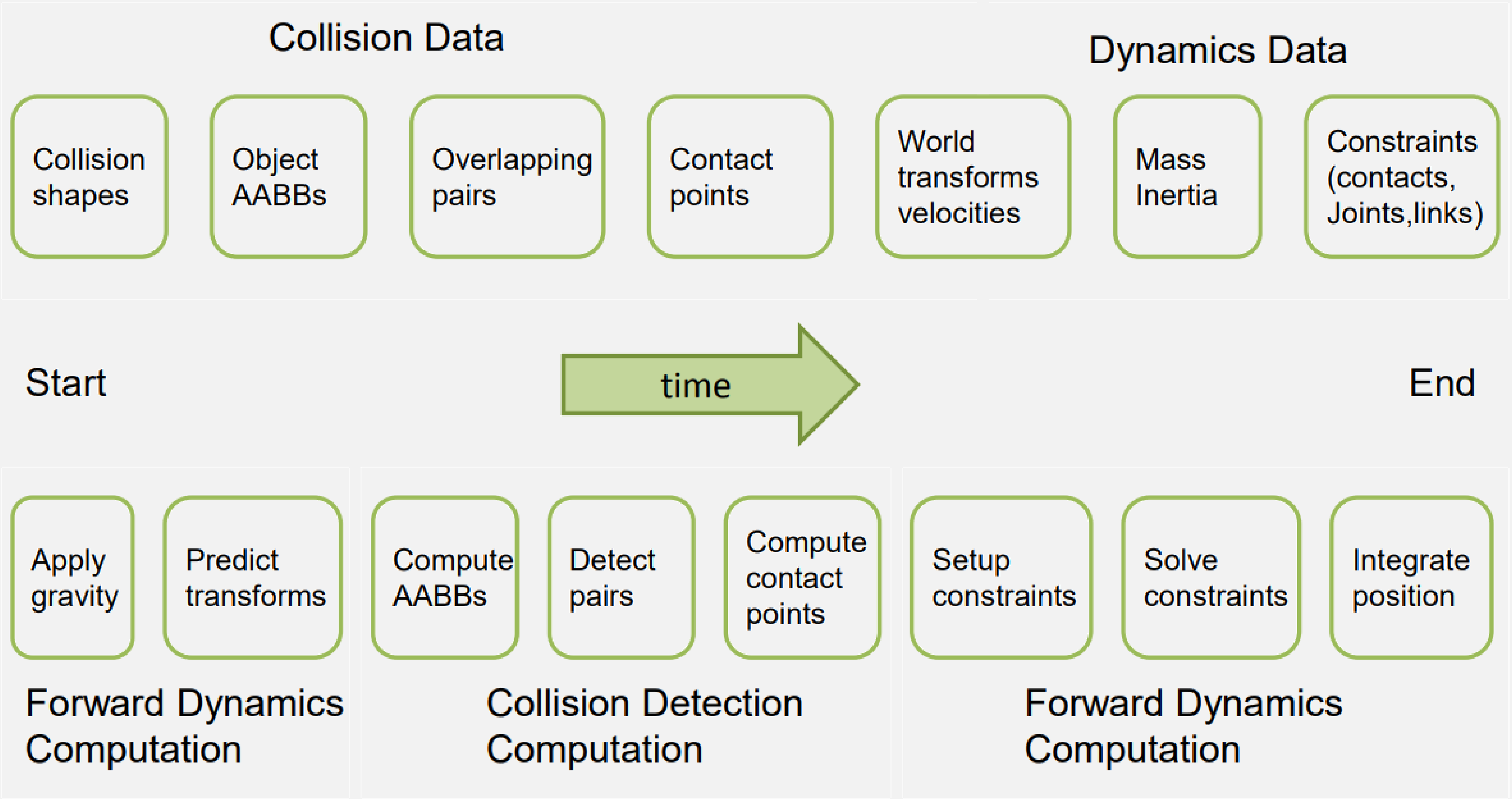
  - The numerical errors in the 9 coefficients in $R(t)$
  - Unit quaternion has far less drift because it only has one extra variable
    - The the drift could be resolved by renormalization

$$\dot{q}(t) = \frac{1}{2}\big(0, \omega(t)\big) * q(t)$$

Types of Rigid Body Contact

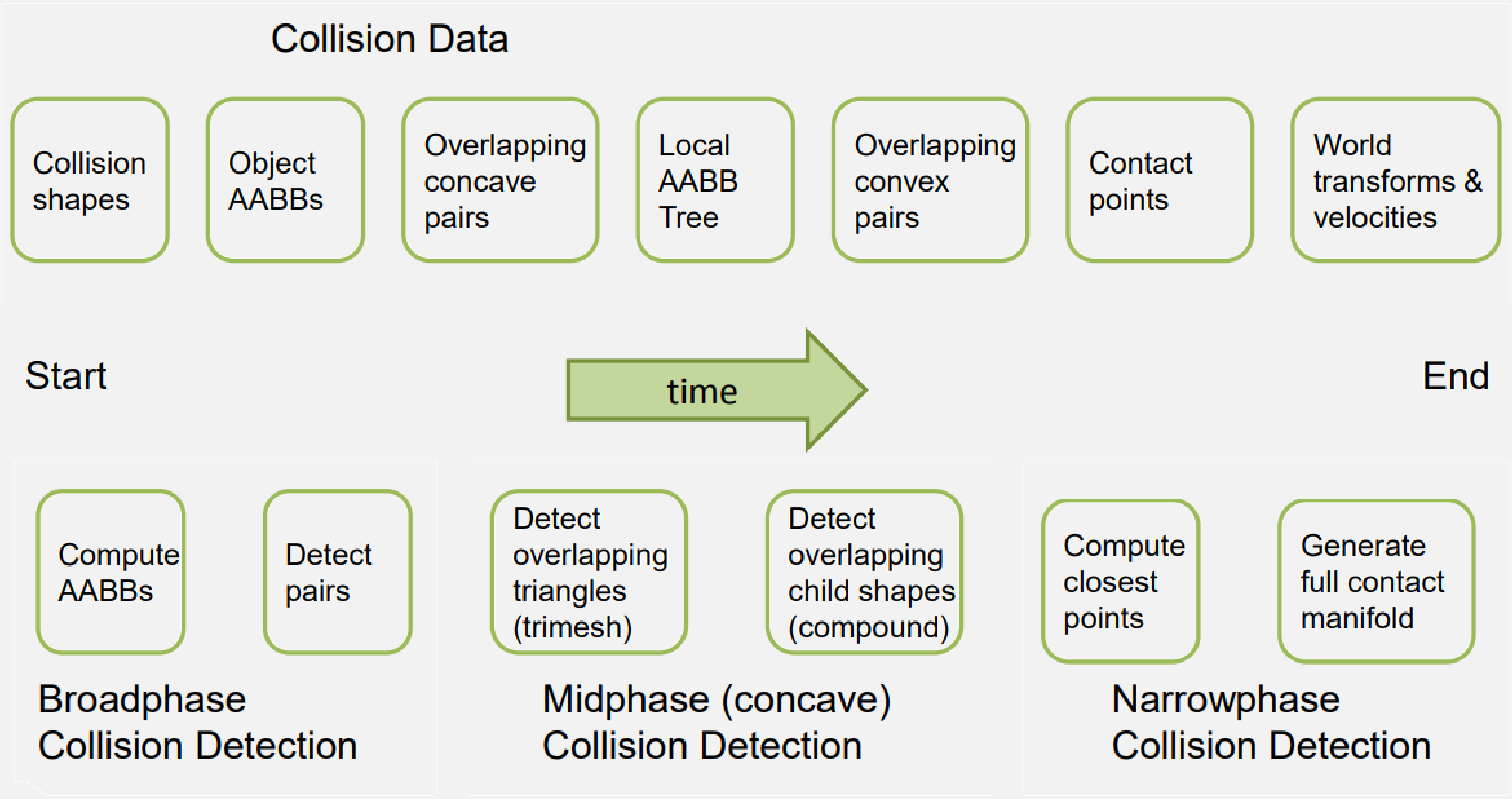
- Rigid bodies are non-flexible $\Rightarrow$ no inter-penetration
- At the instant of contact, particles have to change the velocity abruptly
- There are two types of collision contacts
  - ***Colliding contact***
    - Two bodies are in contact at point p, and their velocity towards each other
    - Requires an instantaneous change in velocity
  - ***Resting contact***
    - Bodies are resting on one another at some point p
    - Compute a *contact force* that prevents the particle from accelerating downwards

Multi Physics Pipeline in Bullet



[Erwin'10]

Collision Detection Pipeline in Bullet

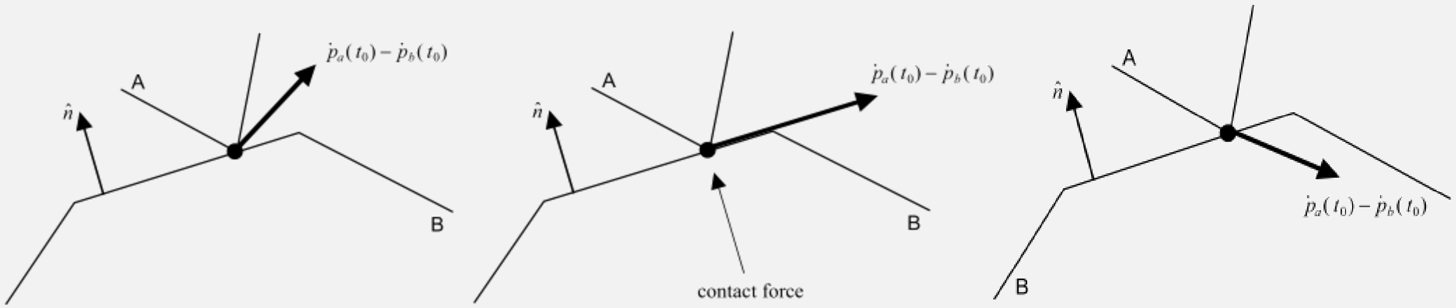


[Erwin'10]

Handling Colliding Contact

- Goal: find the changes of velocities of two bodies
  - Only consider two generic cases: vertex-face and edge-edge contacts
- Contact normal $\hat{\mathbf{n}}$
  - vertex-face: the outwards pointing unit normal of the contact face
  - edge-edge: the cross product of two edges
- Relative velocity: $v_{rel} = \hat{\mathbf{n}}(t_0) \cdot (\dot{\mathbf{p}}_\mathbf{a}(t_0) - \dot{\mathbf{p}}_\mathbf{b}(t_0))$
  - $\dot{\mathbf{p}}_\mathbf{i}(t_0) = \mathbf{v}_\mathbf{i}(t_0) + \boldsymbol{\omega}_\mathbf{i}(t_0) \times \left(\mathbf{p}_\mathbf{i}(t_0) - \mathbf{x}_\mathbf{i}(t_0)\right)$

$$v_{rel} \begin{cases} > 0 & \textit{moving apart} \\ = 0 & \textit{resting contact} \\ < 0 & \textit{colliding} \end{cases}$$



Collision Impulse J

- Impulse $\mathbf{J} = \mathbf{F}\Delta t$
  - Concept: apply very large force in a very short amount of time (like delta function)
  - Result in a finite change in momentum $M\Delta\mathbf{v}$
  - Impulses are always exchanged along contact normal $\hat{\mathbf{n}}(t_0)$, $\mathbf{J} = j\hat{\mathbf{n}}$
- Impulse could change both linear and angular momentum of the object

$$\boxed{\begin{aligned}\Delta\mathbf{P} &= \mathbf{J}\\ \Delta\mathbf{L} &= \mathbf{r_a} \times \mathbf{J} = j(\mathbf{r_a} \times \hat{\mathbf{n}})\end{aligned}} \Rightarrow \begin{aligned}\Delta\mathbf{v_a} &= \frac{\Delta\mathbf{P}}{m_a} = \frac{j\hat{\mathbf{n}}}{m_a}\\ \Delta\boldsymbol{\omega}_\mathbf{a} &= \mathbf{I}^{-1}\Delta\mathbf{L} = j\mathbf{I}^{-1}(\boldsymbol{r}_a \times \hat{\mathbf{n}})\end{aligned}$$

$$\begin{aligned}\dot{\mathbf{p}}_\mathbf{a}^{+} &= (\mathbf{v_a} + \Delta\mathbf{v_a}) + (\boldsymbol{\omega}_\mathbf{a} + \Delta\boldsymbol{\omega}_\mathbf{a}) \times \mathbf{r_a}\\ &= \mathbf{v_a} + \boldsymbol{\omega}_\mathbf{a} \times \mathbf{r_a} + \Delta\mathbf{v_a} + \Delta\boldsymbol{\omega}_\mathbf{a} \times \mathbf{r_a}\\ &= \dot{\mathbf{p}}_\mathbf{a} + \Delta\mathbf{v_a} + \Delta\boldsymbol{\omega}_\mathbf{a} \times \mathbf{r_a} = \dot{\mathbf{p}}_\mathbf{a} + j\left(\frac{\hat{\mathbf{n}}}{m_a} + \mathbf{I}_\mathbf{a}^{-1}(\mathbf{r_a} \times \hat{\mathbf{n}}) \times \mathbf{r_a}\right)\end{aligned}$$

Handling Colliding Contact (Cont.)

- For colliding contacts, the relative velocity after collision v_{rel}^{+} has to be

$$v_{rel}^{+} = \hat{\mathbf{n}} \cdot (\dot{\mathbf{p}}_a^{+} - \dot{\mathbf{p}}_b^{+}) \geq 0, \qquad v_{rel}^{+} = -\varepsilon v_{rel}^{-}, \qquad \varepsilon \in \mathbb{R}$$

  - ε is the restitution coefficient: $\varepsilon = 1$ for fully elastic, while $\varepsilon = 0$ for fully inelastic

$$v_{rel}^{+} = \hat{\mathbf{n}} \cdot \left(\dot{\mathbf{p}}_{\mathbf{a}} + j\left(\frac{\hat{\mathbf{n}}}{m_a} + \mathbf{I}_{\mathbf{a}}^{-1}(\mathbf{r}_{\mathbf{a}} \times \hat{\mathbf{n}}) \times \mathbf{r}_{\mathbf{a}}\right) - \dot{\mathbf{p}}_b - j\left(\frac{\hat{\mathbf{n}}}{m_b} + \mathbf{I}_{\mathbf{b}}^{-1}(\mathbf{r}_b \times \hat{\mathbf{n}}) \times \mathbf{r}_b\right)\right)$$

$$= v_{rel}^{-} + \left(j\left(\frac{1}{m_a} + \mathbf{I}_a^{-1}(\mathbf{r}_{\mathbf{a}} \times \hat{\mathbf{n}}) \times \mathbf{r}_{\mathbf{a}} \cdot \hat{\mathbf{n}}\right) - j\left(\frac{1}{m_b} + \mathbf{I}_b^{-1}(\mathbf{r}_b \times \hat{\mathbf{n}}) \times \mathbf{r}_b \cdot \hat{\mathbf{n}}\right)\right) = -\varepsilon v_{rel}^{-}$$

solve

$$j = \frac{-(1+\varepsilon)v_{rel}^{-}}{\frac{1}{m_a} + \mathbf{I}_{\mathbf{a}}^{-1}(\mathbf{r}_{\mathbf{a}} \times \hat{\mathbf{n}}) \times \mathbf{r}_{\mathbf{a}} \cdot \hat{\mathbf{n}} + \frac{1}{m_b} + \mathbf{I}_b^{-1}(\mathbf{r}_b \times \hat{\mathbf{n}}) \times \mathbf{r}_b \cdot \hat{\mathbf{n}}}$$

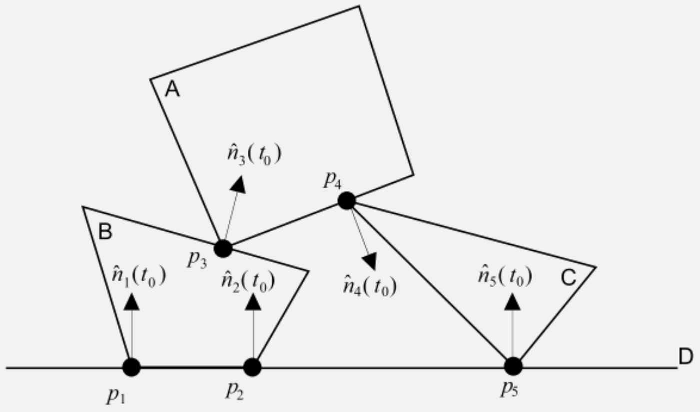
update

$$\mathbf{v}_{\mathbf{a}}^{+} = \mathbf{v}_{\mathbf{a}} + \frac{j\hat{\mathbf{n}}}{m_a}, \qquad \mathbf{v}_b^{+} = \mathbf{v}_b + \frac{j\hat{\mathbf{n}}}{m_b}$$

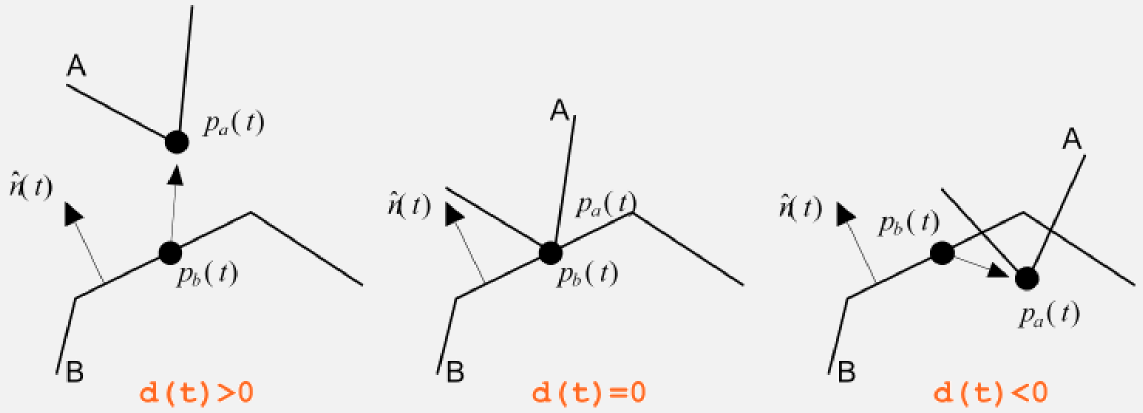
$$\boldsymbol{\omega}_a^{+} = \boldsymbol{\omega}_{\mathbf{a}} + j\mathbf{I}_{\mathbf{a}}^{-1}(\boldsymbol{r}_a \times \hat{\mathbf{n}}), \qquad \boldsymbol{\omega}_b^{+} = \boldsymbol{\omega}_b + j\mathbf{I}_b^{-1}(\boldsymbol{r}_b \times \hat{\mathbf{n}})$$

Handling Resting Contacts

- Goal: determaine all *contact force* at the same time
  - Contact force is along contact normal: $\mathbf{f_i} = f_i\hat{\mathbf{n}}_\mathbf{i}(t_0)$
  - Requirements for contact forces
    1. Must prevent inter-penetration
    2. Repulsive: push bodies apart, never act like glue
    3. Become zero if the bodies begin to separate



- Separation distance $d_i(t) = \hat{\mathbf{n}}_\mathbf{i}(t) \cdot \left(\mathbf{p_a}(\mathbf{t}) - \mathbf{p_b}(\mathbf{t})\right)$
  - The separation between A and B near $\mathbf{p_a}(\mathbf{t})$
    - $d_i(t) = 0$: at i-th contact point
    - $d_i(t) > 0$: lost contact
    - $d_i(t) < 0$: inter-penetrate



Handling Resting Contacts (Cont.)

- At the contact moment $\mathrm{d_i}(\mathrm{t_0}) = 0$, we want to keep it from decreasing at time t_0 (e.g. $\dot{\mathrm{d}}_\mathrm{i}(\mathrm{t_0}) \geq 0$)
- The separation velocity at time t

$$\dot{d}_i(t) = \dot{\hat{\mathbf{n}}}_\mathbf{i}(t) \cdot \left(\mathbf{p_a}(t) - \mathbf{p_b}(t)\right) + \hat{\mathbf{n}}_\mathbf{i}(t) \cdot (\dot{\mathbf{p}}_\mathbf{a}(t) - \dot{\mathbf{p}}_\mathbf{b}(t))$$

  - At time t_0 when contact occurs
    - $\mathbf{p_a}(t_0) = \mathbf{p_b}(t_0) \Rightarrow \dot{d}_i(t_0) = \hat{\mathbf{n}}_\mathbf{i}(t_0) \cdot \left(\dot{\mathbf{p}}_\mathbf{a}(t_0) - \dot{\mathbf{p}}_\mathbf{b}(t_0)\right)$
- For resting contact $d_i(t_0) = \dot{d}_i(t_0) = 0$, and the second time derivative is:

$$\ddot{d}_i(t_0) = \ddot{\hat{\mathbf{n}}}(t_0) \cdot \underbrace{\boxed{\left(\mathbf{p_a}(t_0) - \mathbf{p_b}(t_0)\right)}}_{=0} + 2\dot{\hat{\mathbf{n}}}_\mathbf{i}(t_0) \cdot \left(\dot{\mathbf{p}}_\mathbf{a}(t_0) - \dot{\mathbf{p}}_\mathbf{b}(t_0)\right) + \hat{\mathbf{n}}_\mathbf{i}(t_0) \cdot \left(\ddot{\mathbf{p}}_\mathbf{a}(t_0) - \ddot{\mathbf{p}}_\mathbf{b}(t_0)\right)$$

$$= 2\dot{\hat{\mathbf{n}}}_\mathbf{i}(t_0) \cdot \left(\dot{\mathbf{p}}_\mathbf{a}(t_0) - \dot{\mathbf{p}}_\mathbf{b}(t_0)\right) + \hat{\mathbf{n}}_\mathbf{i}(t_0) \cdot \left(\ddot{\mathbf{p}}_\mathbf{a}(t_0) - \ddot{\mathbf{p}}_\mathbf{b}(t_0)\right)$$

Solve Contact Forces

- $\ddot{d}_i(t_0)$ measures how the two bodies are accelerating towards each other
- Formulate the requirements of contact forces
  1. $\ddot{d}_i(t_0) \geq 0$: prevent inter-penetration
  2. $f_i \geq 0$: be repulsive
  3. $f_i \ddot{d}_i(t_0) = 0$: become zero if the bodies begin to separate
- To find the contact forces satisfying above requirements, we express each $\ddot{d}_i(t_0)$ as a function of the unknown $f_i{}'s$

$$\ddot{d}_i(t_0) = a_{i1} f_1 + a_{i2} f_2 + \cdots + a_{in} f_n + b_i$$

$$\begin{pmatrix} \ddot{d}_1(t_0) \\ \vdots \\ \ddot{d}_n(t_0) \end{pmatrix} = A \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} + \begin{pmatrix} b_i \\ \vdots \\ b_n \end{pmatrix}$$

WEE GAFFES

SOMETIMES THE COMPUTER MAKES MISTAKES.

IT'S NOT OUR FAULT.


GUARDIANS
OF THE GALAXY
VOL. 2
SONY
WALKMAN
FRAMESTORE

References

- Real-Time Physics, SIGGRAPH Course Notes, 2008.
- David Baraff and Andrew Witkin, Physically Based Modeling, SIGGRAPH Course Notes, 2001.
- Interactive Simulation of Rigid Body Dynamics in Computer Graphics, STAR, EUROGRAPHICS, 2012.
- Donald House, John C. Keyser, Foundations of Physically Based Modeling and Animation, CRC Press, 2017.