CS-184: Computer Graphics

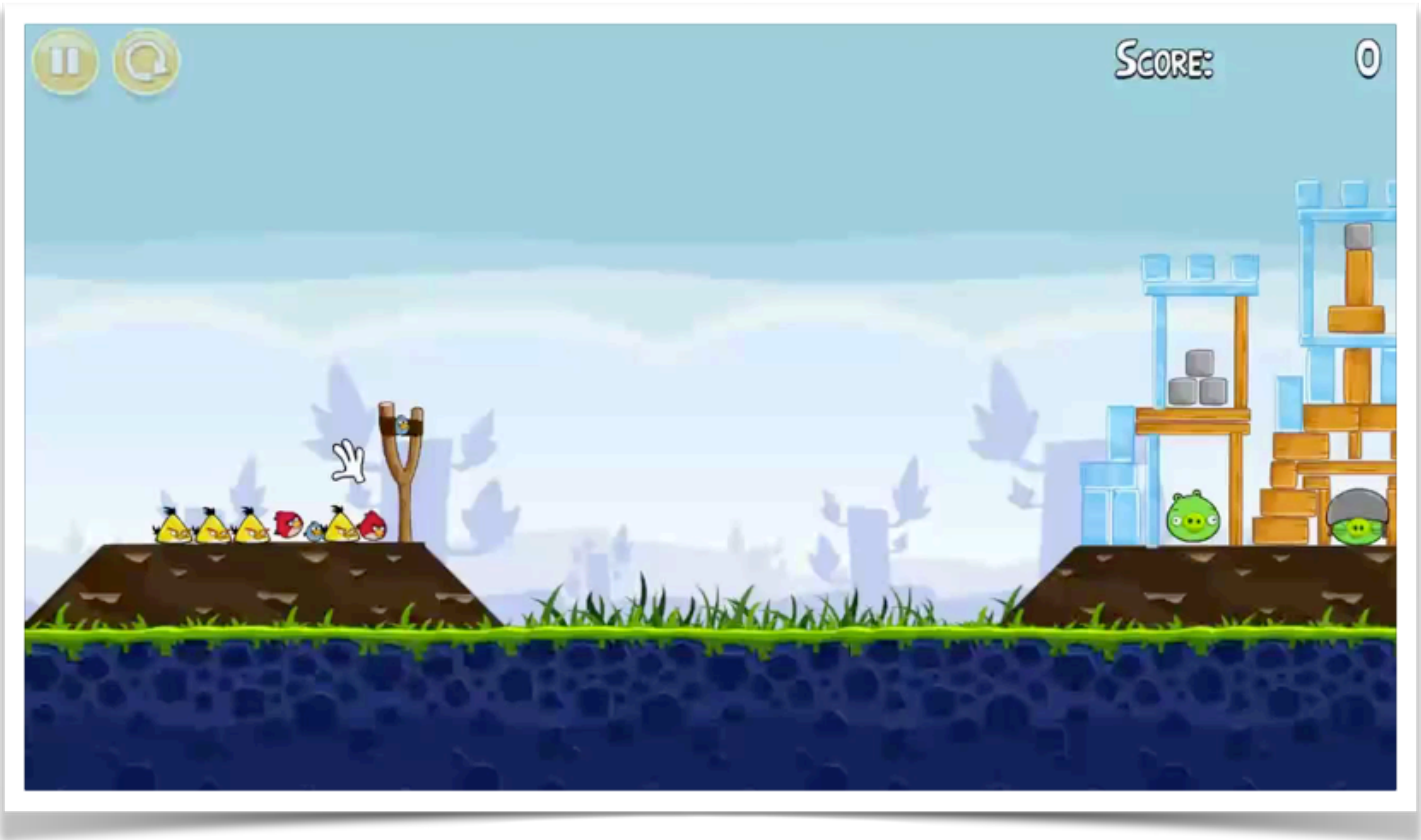
Lecture #25:
Rigid Body Simulations

Tobias Pfaff
537 Soda
(Visual Computing Lab)
tpfaff@berkeley.edu

Reminder

- Final project presentations next week!

“Game Physics”



Types of Materials

- Particles
  - Weakly interacting particles for fluids
  - Non-interacting particles for visuals
- Mass-spring Systems
  - Can model elastic ropes, sheets and bodies
  - Simple model, fast
  - Stiffness / discretization difficult to fine tune
  - Stability problems for stiff materials
- Finite Elements
  - Volume discretization of physical elasticity model
  - More stable and controllable, but complex

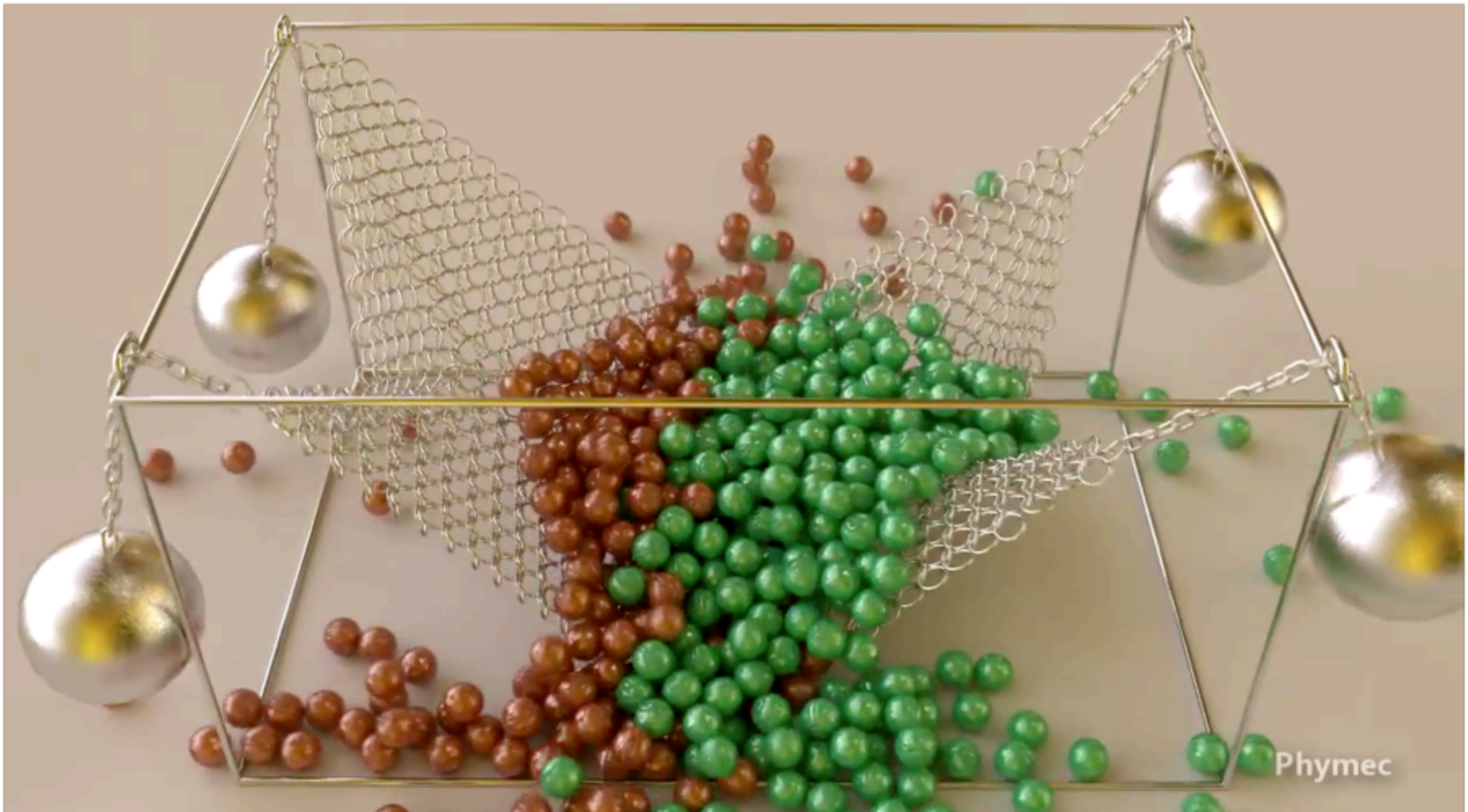
Rigidity

- All materials are elastic to some extent in reality
- Example: try to model metal bar with high stiffness
  - Will enforce inter object distances
  - Propagate deformation, real behavior in the limit
- Problem: high stiffness means large forces, time step will be tiny to keep things stable

Observation

- High stiffness means: vertices should not move w.r.t. each other
  - Effectively removes degrees of freedom from the system
- Obvious: don’t even simulate them in the first place
- New representation: center of mass and orientation
- Need equations of motion for both center of mass and orientation

Complex Dynamics



Bullet Physics Engine / Blender. Video by Phymec

Overview

- Rigid bodies in 2D
  - Orientation
  - Integrating rotational motion
  - Angular momentum
  - Impulses
- Rigid bodies in 3D

Points vs. Rigid Bodies

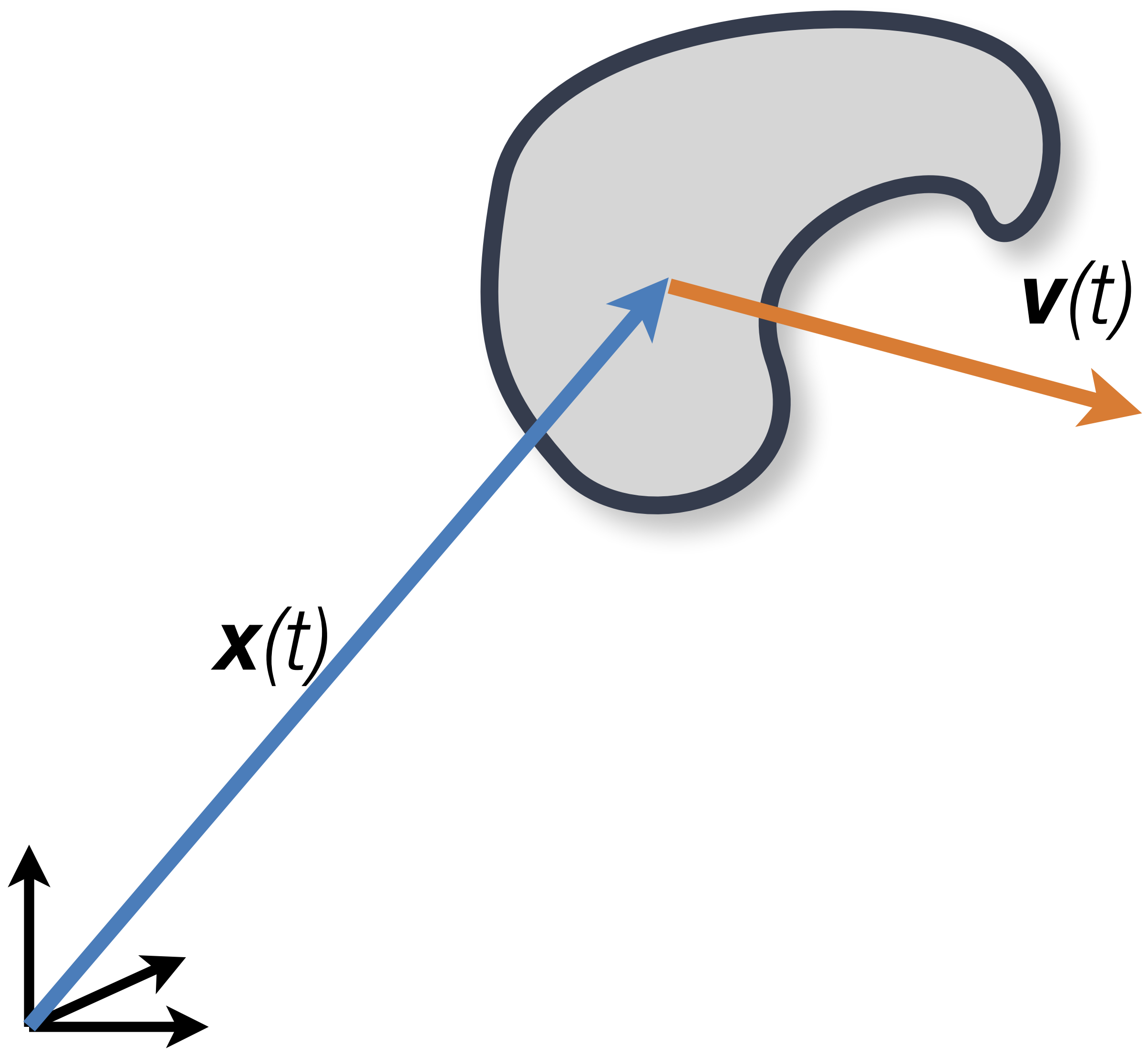
- For particles:
  - Position ***x***
  - Velocity ***v***
- Dynamics:

$$\mathbf{v}(t) = \frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t}$$

$$\mathbf{a}(t) = \frac{\mathrm{d}\mathbf{v}(t)}{\mathrm{d}t}$$

- For a rigid body:
  - Position ***x***
  - ?
  - Velocity ***v***
  - ?

Representation



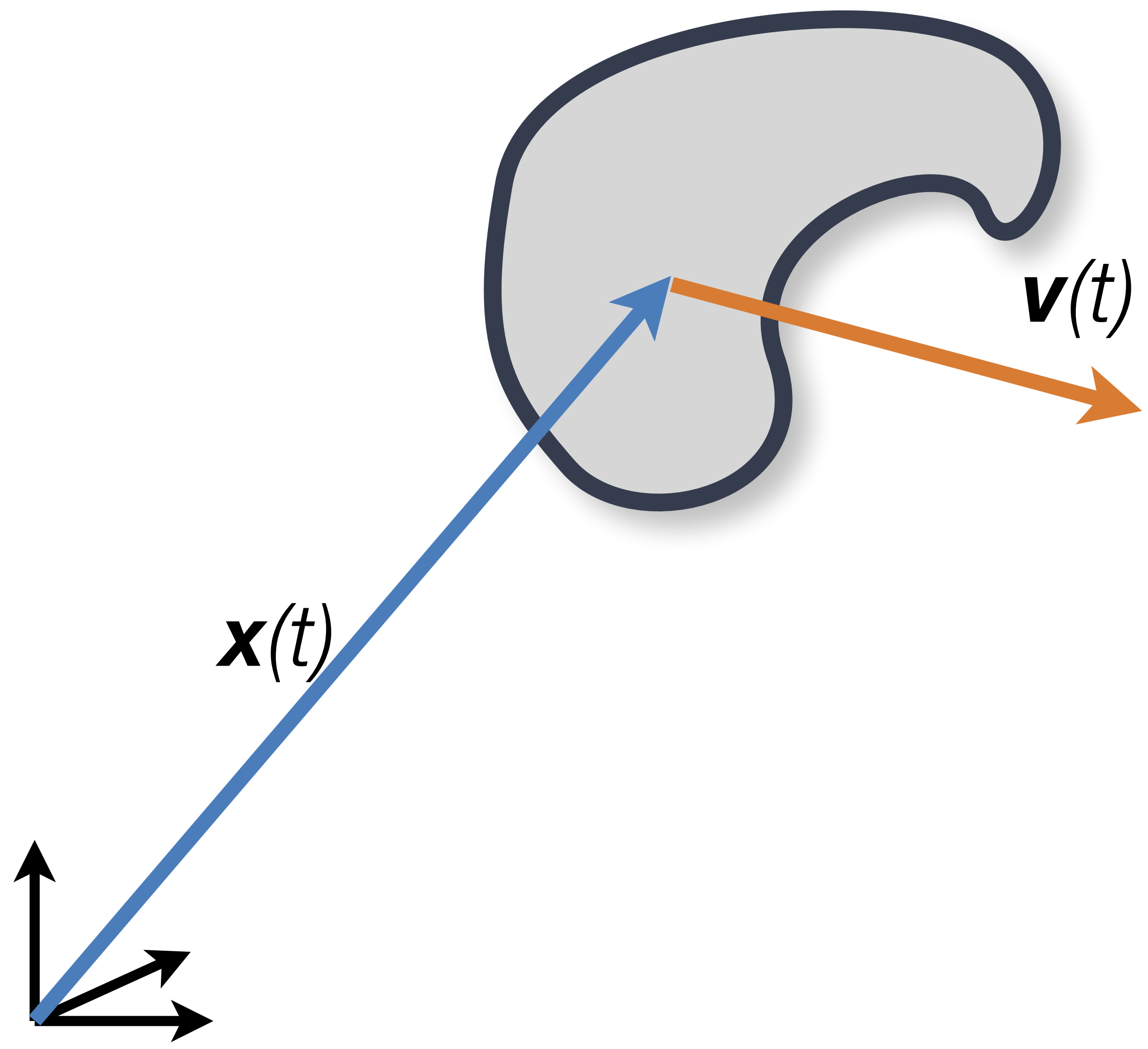
- Reference point on body: center of mass
- Continuous:

$$\mathbf{x}_{cm} = \frac{\int \mathbf{x}\rho(x)\mathrm{d}V}{\int \rho(x)\mathrm{d}V}$$

- Discrete:

$$\mathbf{x}_{cm} = \frac{\sum_i m_i \mathbf{x_i}}{\sum_i m_i}$$

Representation



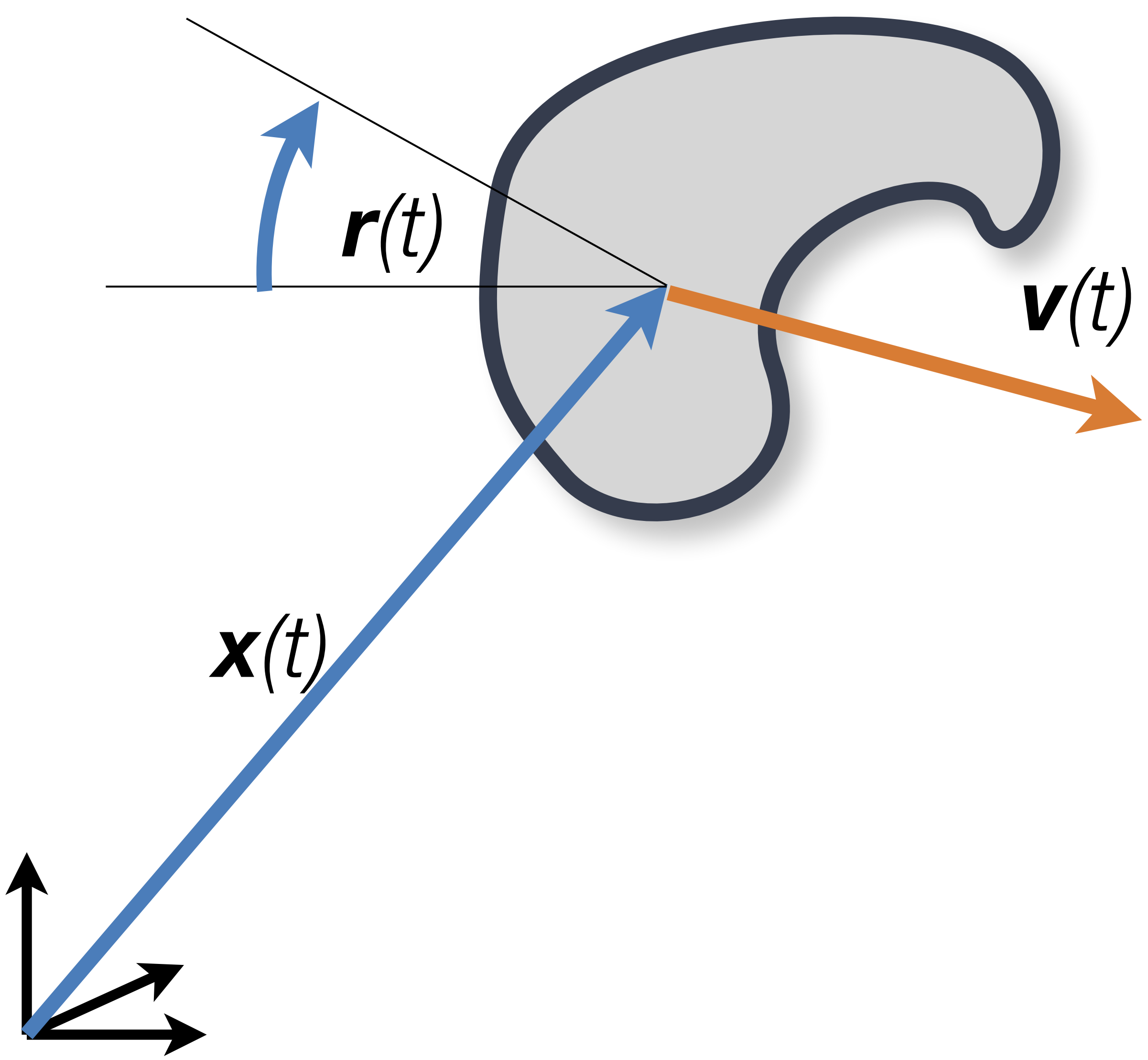
- Center of mass behaves like a mass point with total mass of the body $M = \sum_i m_i$

$$\mathbf{x}_{cm} = \frac{\sum_i m_i \mathbf{x_i}}{M}$$

$$\mathbf{v}_{cm} = \frac{\sum_i m_i \mathbf{v_i}}{M}$$

$$\mathbf{a}_{cm} = \frac{\sum_i m_i \mathbf{a_i}}{M}$$

Representation

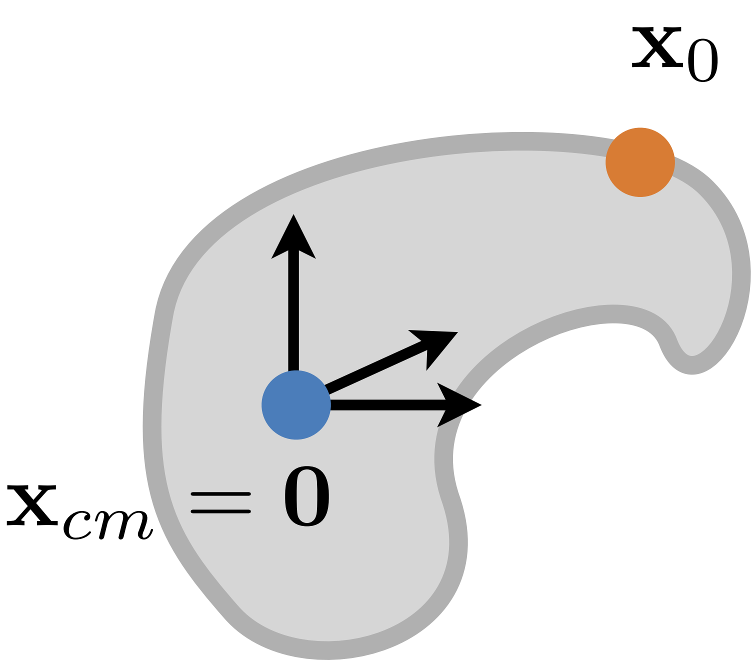


- Orientation: rotation around center of mass
- Point coordinates relative to center of mass (body space)
- Absolute position (world space)

$$\mathbf{x}_{world} = \mathbf{x}_{cm} + Rot_{\mathbf{r}(t)}(\mathbf{x}_0)$$

Representation

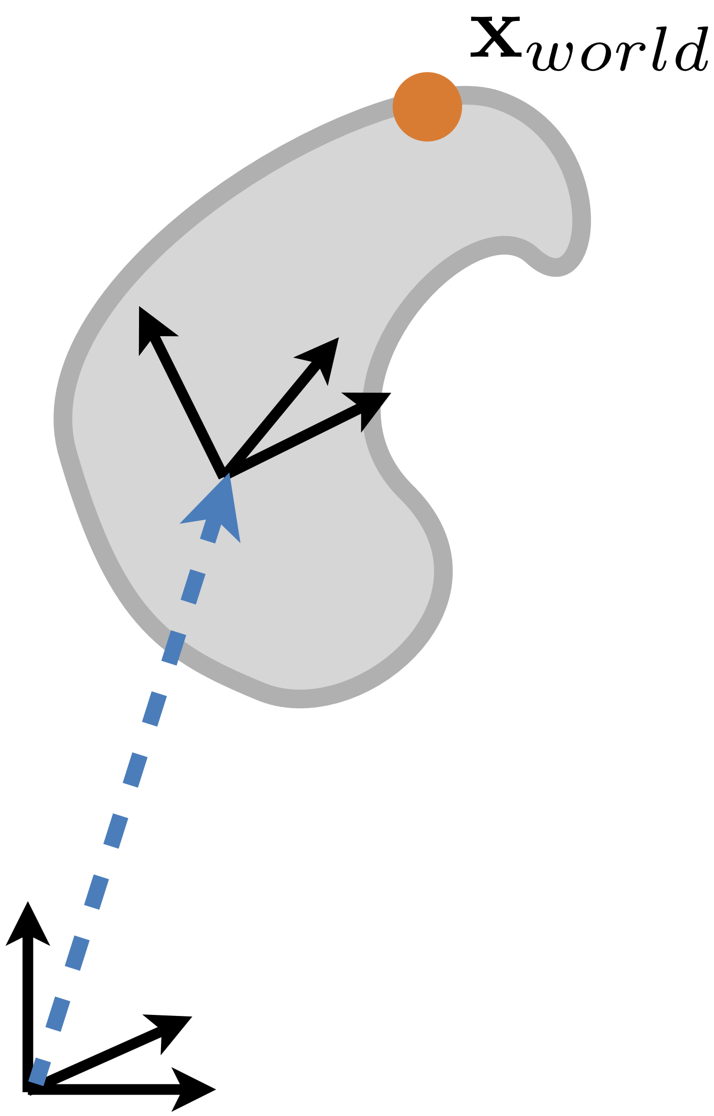
Body space



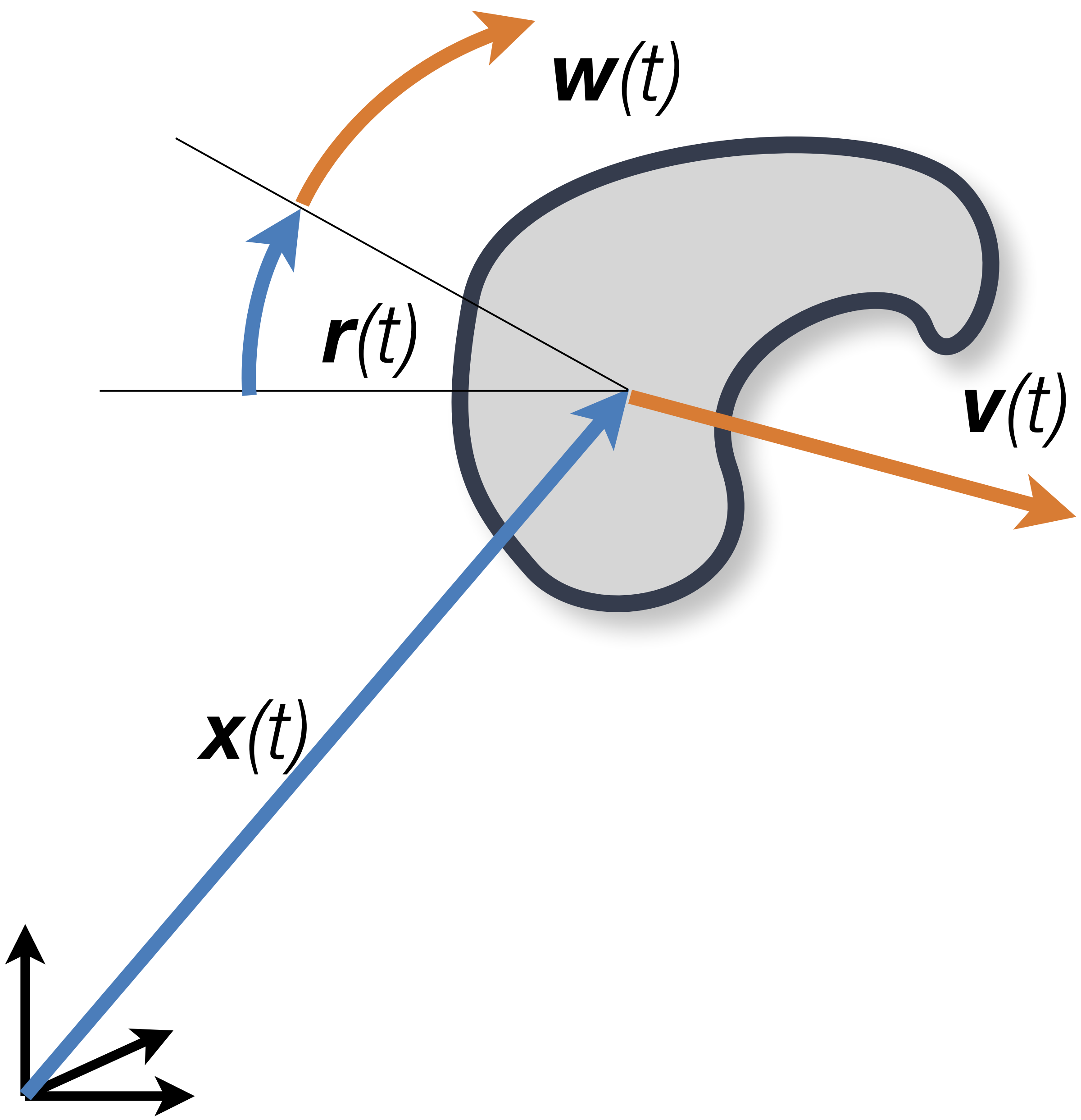
(For convenience, keep center of mass at zero.)

World space

$$\mathbf{x}_{world} = \mathbf{x}_{cm} + Rot_{\mathbf{r}(t)}(\mathbf{x}_0)$$

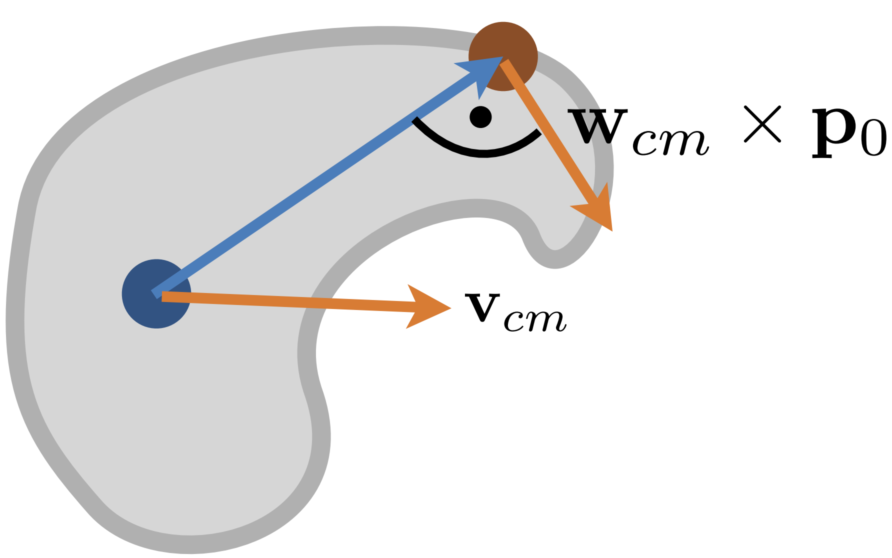


Representation



- Previously: linear velocity
- Now also: angular velocity
- 3 component vector encoding rate of angular change, and axis of rotation
- Total velocity of a point:

$$\mathbf{v}_i = \mathbf{v}_{cm} + \mathbf{w}_{cm} \times \mathbf{p}_0$$



Orientation & Angular Velocity in 2D

- Only rotation around z, so ***w*** is a scalar
- E.g. given in radians
- Velocity of a point is given by:

$$\mathbf{v}_i = \begin{pmatrix} -w \ x_{i,y} \\ w \ x_{i,x} \end{pmatrix}$$

- Apply orientation to point with matrix:

$$\mathrm{Rot}_\alpha = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$$

Points vs. Rigid Bodies

- For particles:
  - Position ***x***
  - Velocity ***v***
- Dynamics:

$$\mathbf{v}(t) = \frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t}$$

$$\mathbf{a}(t) = \frac{\mathrm{d}\mathbf{v}(t)}{\mathrm{d}t}$$

- For a rigid body:
  - Position ***x***
  - Orientation ***r***
  - Linear velocity ***v***
  - Angular velocity ***w***
- Dynamics:
  - **?**

Special Case for 2D

- Same as for point masses, e.g., Euler step:

$$\mathbf{x}_{cm} = \mathbf{x}_{cm} + h\mathbf{v}_{cm}$$

$$\mathbf{r}_{cm} = \mathbf{r}_{cm} + h\mathbf{w}_{cm}$$

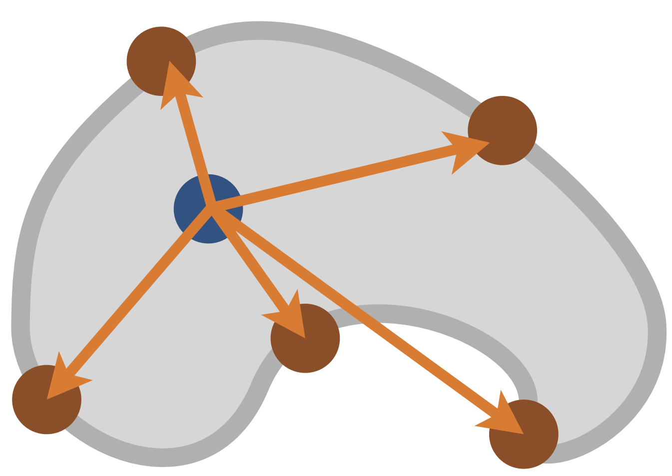
- Works only because we have 1 axis of rotation
- Later on: conserve angular momentum (not angular velocity)
- How to compute accelerations?

Inertia Tensor in 2D

- Equivalent to mass: “resistance to rotation”
  Is pre-computed for reference state
- In 2D, using body coordinates:

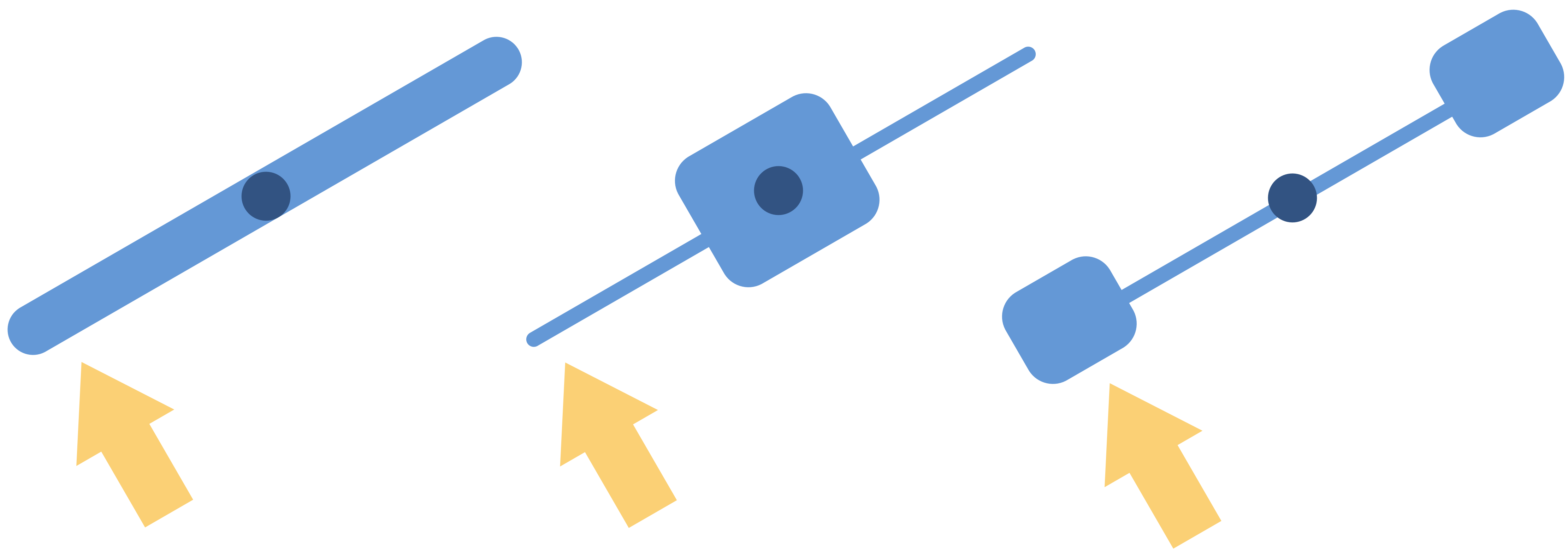
$$i = \sum_n m_n \mathbf{x}_n \cdot \mathbf{x}_n$$

- Example



Inertia Tensor in 2D

- What does *i* look like for these shapes? (Assume equal total mass, and same material density.)
- Or - which shape spins more easily when poked?



Example

- Figure skating
  - Starts in normal pose
  - Rotation in plane
  - Moment of inertia is reduced by pulling in arms

Rotational Dynamics

- Mass points are restricted to move perpendicular to their body space position
- Use cross product for projection
- Newton's 2nd law: $\frac{\mathrm{d}}{\mathrm{d}t}(m_i \mathbf{v}_i) = \mathbf{f}_i$
- Newton's 2nd law, restricted:

$$\mathbf{x}_i \times \frac{\mathrm{d}}{\mathrm{d}t}(m_i \mathbf{v}_i) = \mathbf{x}_i \times \mathbf{f}_i$$

Both sides are vectors parallel to actual axis of rotation

Rotational Dynamics

- From before
$$\mathbf{x}_i \times \frac{\mathrm{d}}{\mathrm{d}t}(m_i \mathbf{v}_i) = \mathbf{x}_i \times \mathbf{f}_i$$
- Move time derivative
$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{x}_i \times m_i \mathbf{v}_i) = \mathbf{x}_i \times \mathbf{f}_i$$
- For whole body
$$\frac{\mathrm{d}}{\mathrm{d}t}\sum_i(\mathbf{x}_i \times m_i \mathbf{v}_i) = \sum_i \mathbf{x}_i \times \mathbf{f}_i$$
- Rename

angular momentum
eq. momentum

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{L} = \mathbf{q}$$

torque
eq. force

Both sides are still vectors
parallel to axis of rotation

Newton's 2nd Law for Rotations

- Angular momentum: $\mathbf{L} = \sum_i \mathbf{x}_i \times m_i \mathbf{v}_i = \mathbf{I}\mathbf{w}$
- Torque $\mathbf{q} = \sum_i \mathbf{x}_i \times \mathbf{f}_i$
- Angular version of Newton's 2nd law: $\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{L} = \mathbf{q}$
- Compute the change of angular velocity over time, for 2D:

$$\mathbf{w}(t+h) = \mathbf{I}^{-1}\ \mathbf{L}(t+h)$$

$$\mathbf{w}(t+h) = \mathbf{w}(t) + h\mathbf{q}/i$$

Can has Angular Momentum Conservation?

- No external forces = no torque
  - angular momentum is constant

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{L} = \mathbf{q}$$

- Cats still turn around in mid-air just fine

Can has Angular Momentum Conservation?

Points vs. Rigid Bodies

- For particles:
  - Position ***x***
  - Velocity ***v***
- Dynamics:

$$\mathbf{v}(t) = \frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t}$$

$$\mathbf{a}(t) = \frac{\mathrm{d}\mathbf{v}(t)}{\mathrm{d}t}$$

- For a rigid body:
  - Position ***x***
  - Orientation ***r***
  - Linear velocity ***v***
  - Angular velocity ***w***
- – Angular dynamics:

$$\mathbf{q}(t) = \sum_i \mathbf{x}_i \times \mathbf{f}_i$$

$$\mathbf{w}(t+h) = \mathbf{w}(h) + h\mathbf{q}/i$$

Simulation Algorithm in 2D

Pre-compute:

$$M \leftarrow \sum_i m_i$$

$$\mathbf{x}'_{cm} \leftarrow \sum_i \mathbf{x}'_i m_i / M$$

$$\mathbf{x}_i \leftarrow \mathbf{x}'_i - \mathbf{x}'_{cm}$$

$$i \leftarrow \sum_i m_i \mathbf{x}_i \cdot \mathbf{x}_i$$

Initialize:

$$\mathbf{x}_{cm}, \mathbf{v}_{cm}$$

$$\mathbf{r}, \mathbf{L}$$

$$\mathbf{w} \leftarrow \mathbf{L}/i$$

External forces

$$\mathbf{t} \leftarrow \sum_i \mathbf{x}_i \times \mathbf{f}_i$$

$$\mathbf{F} \leftarrow \sum_i \mathbf{f}_i$$

Euler step

$$\mathbf{x}_{cm} \leftarrow \mathbf{x}_{cm} + h\mathbf{v}_{cm}$$

$$\mathbf{v}_{cm} \leftarrow \mathbf{v}_{cm} + h\mathbf{F}/M$$

$$\mathbf{r} \leftarrow \mathbf{r} + h\mathbf{w}$$

$$\mathbf{w} \leftarrow \mathbf{w} + h\mathbf{t}/i$$

World position

$$\mathbf{x}_i^{world} \leftarrow \mathbf{x}_{cm} + \mathrm{Rot}_r \mathbf{x}_i$$

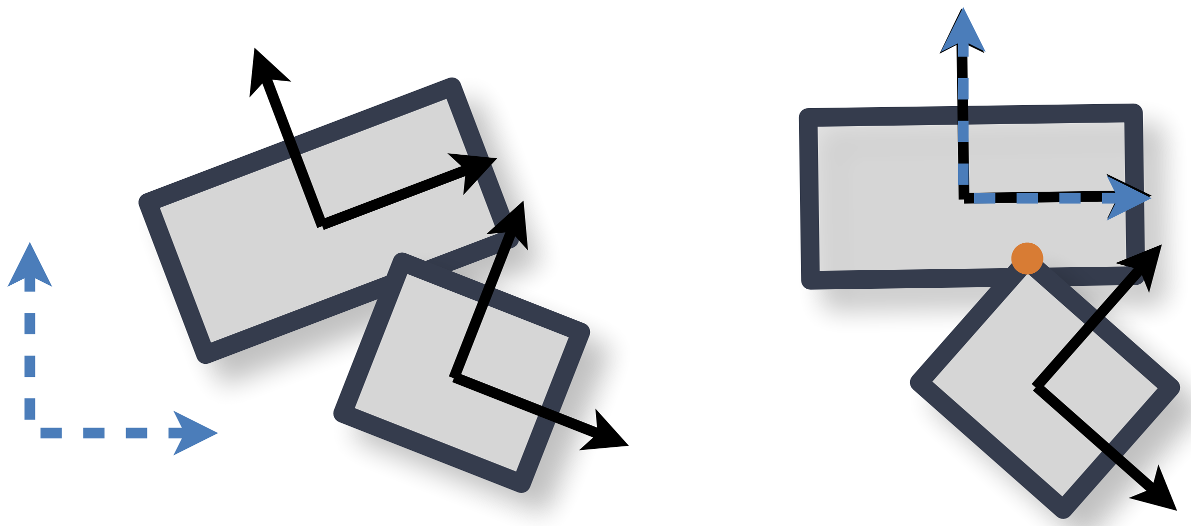
$$\mathbf{v}_i^{world} \leftarrow \mathbf{v}_{cm} + \mathbf{w} \times \mathbf{x}_i$$

Collisions

- What happens during a collision?
  - Body is deformed
  - Elasticity: Deformation energy is released, body bounces back
  - Plasticity: Deformation energy is dissipated, body stays deformed
  - Different materials have different elasticity and plasticity
- Usually happens in a fraction of a second…
  - Hard to simulate explicitly

Collision Detection

- Simple case
  - Simulate boxes
  - Check corner points (or points on surface)
  - For target body, undo translation & rotation
  - Test points for intervals
- In practice:
  polgygon intersections,
  acceleration structures

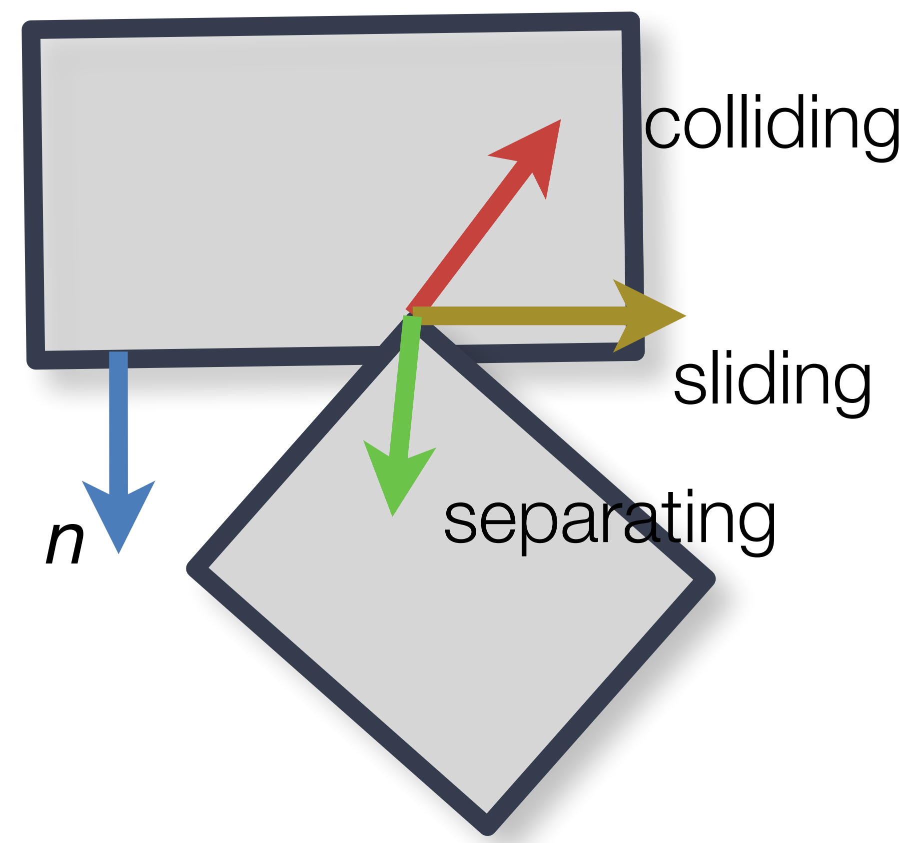


Classifying Contacts

- Velocity of x_i on rigid body: $\mathbf{v}_i = \mathbf{v}_{cm} + \mathbf{w} \times \mathbf{x}_i$
- Retrieve collision normal
- Compute relative velocity:

$$\mathbf{v}_{rel} = \mathbf{n} \cdot (\mathbf{v}_A - \mathbf{v}_B)$$

- 3 Cases:
  - Colliding contact $\mathbf{v}_{rel} < 0$
  - Separating (easy!) $\mathbf{v}_{rel} > 0$
  - Resting contact $\mathbf{v}_{rel} = 0$

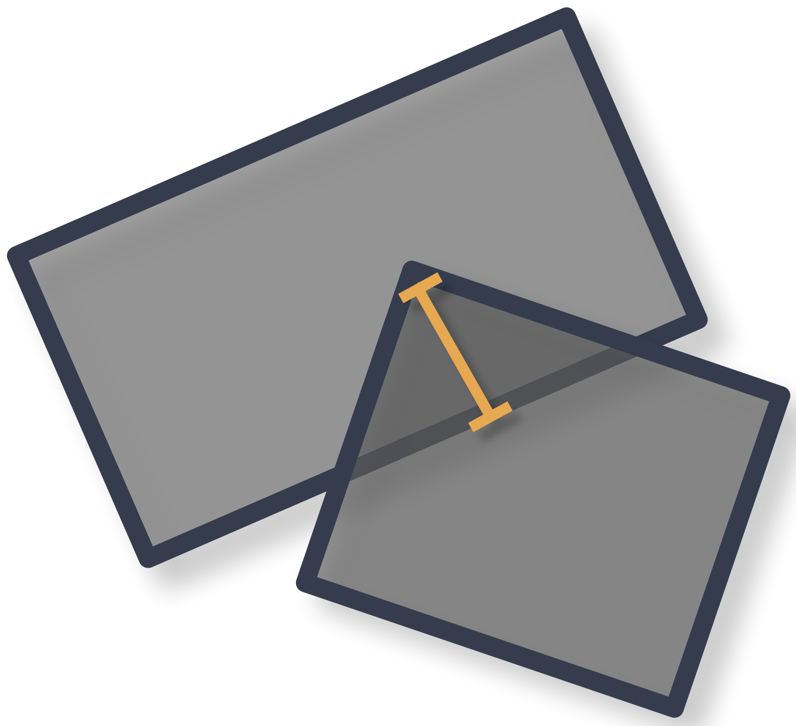


Collision Response

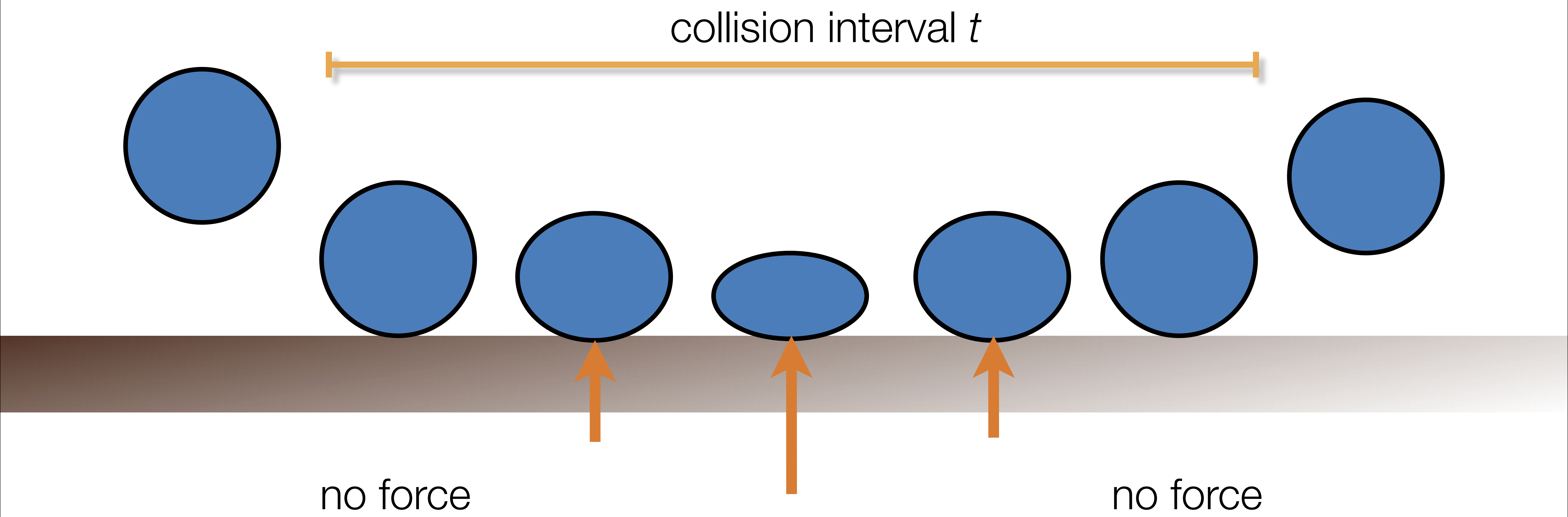
- Compute instantaneous effect of material deformation
- Separate handling of
  - linear motion
  - angular motion
- Make sure the object stop flying into each other...

Impulses

- We could try to model instantaneous deformation with forces, e.g.:
  - Measure penetration distance d
  - Apply force proportional to d
  - Hope that it keeps objects from moving into each other...
- Not a good idea:
  - No guarantees
  - Can cause large forces

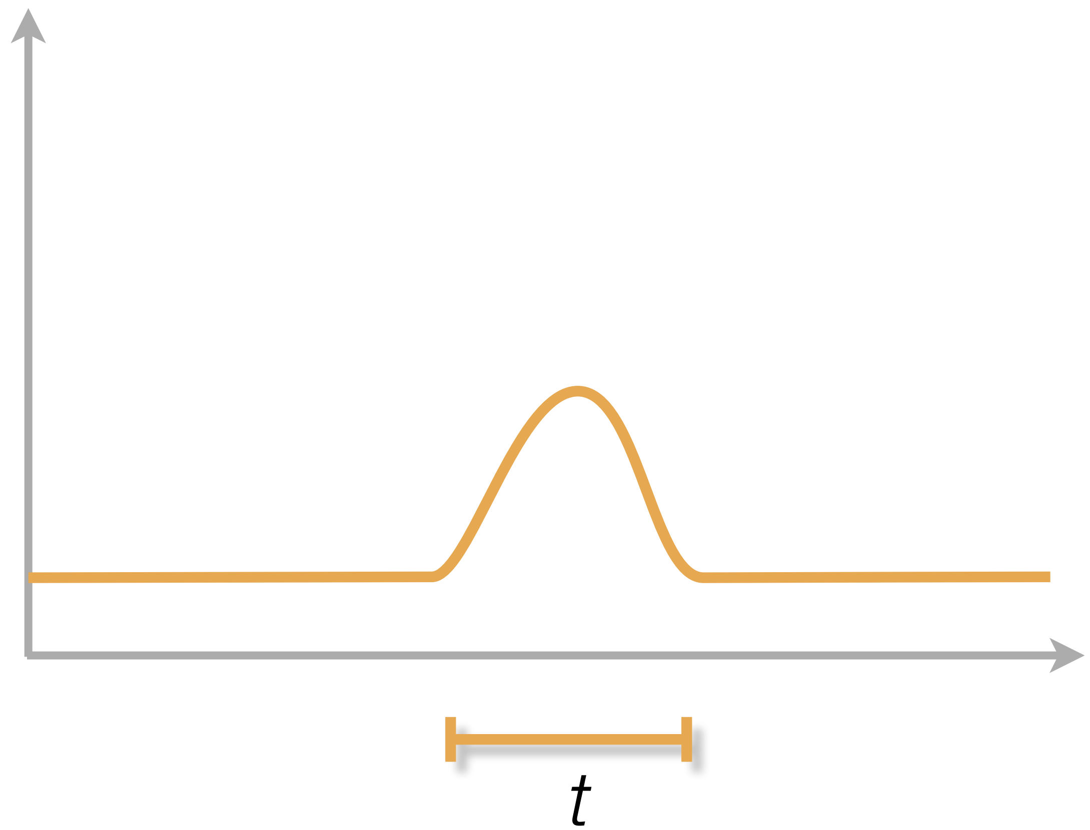


Impulses

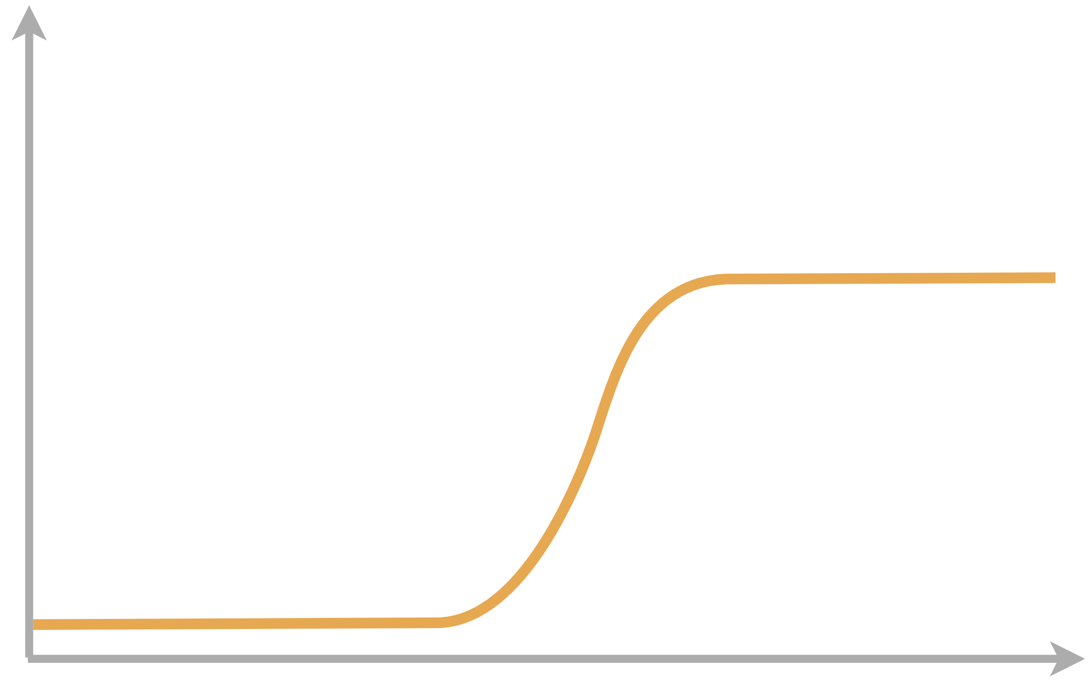


Soft Collision

- Force

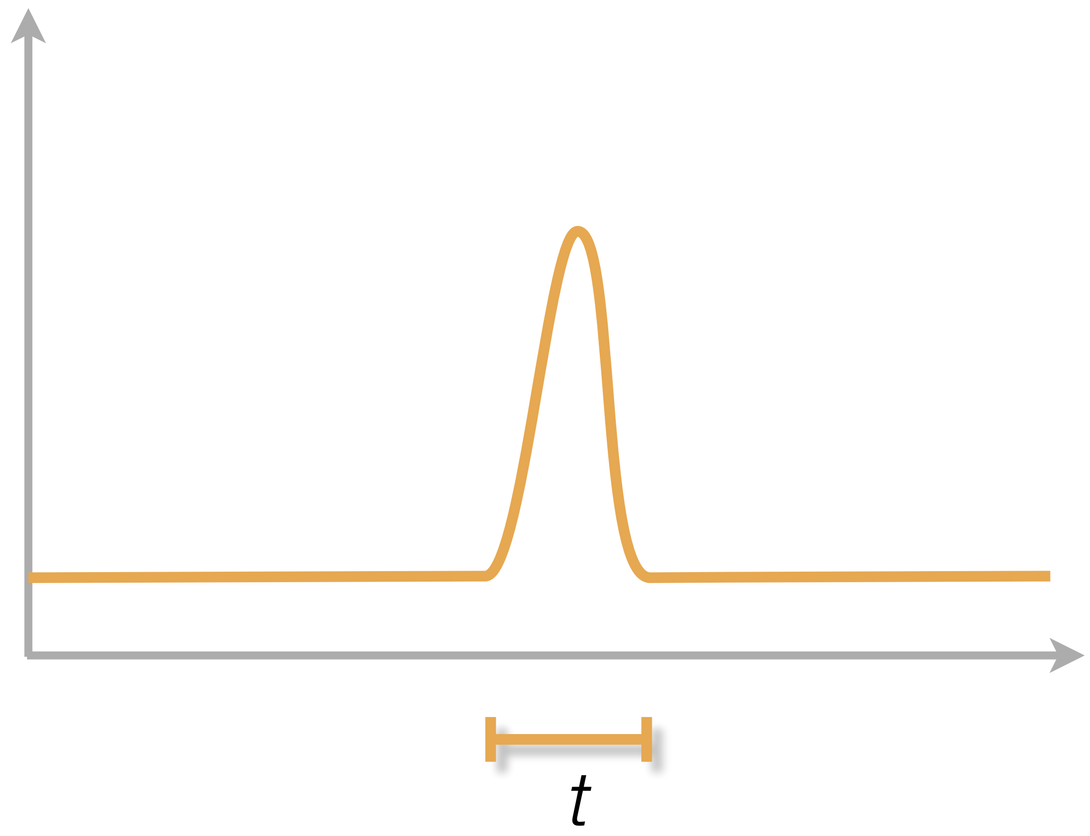


- Velocity

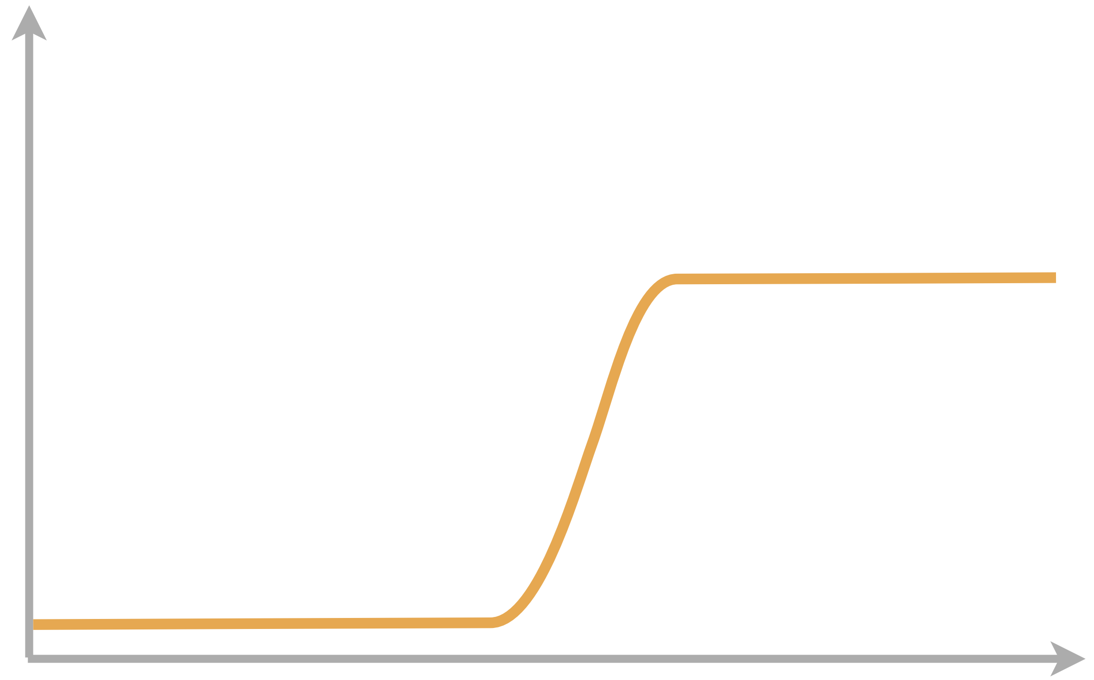


Harder Collision

- Force

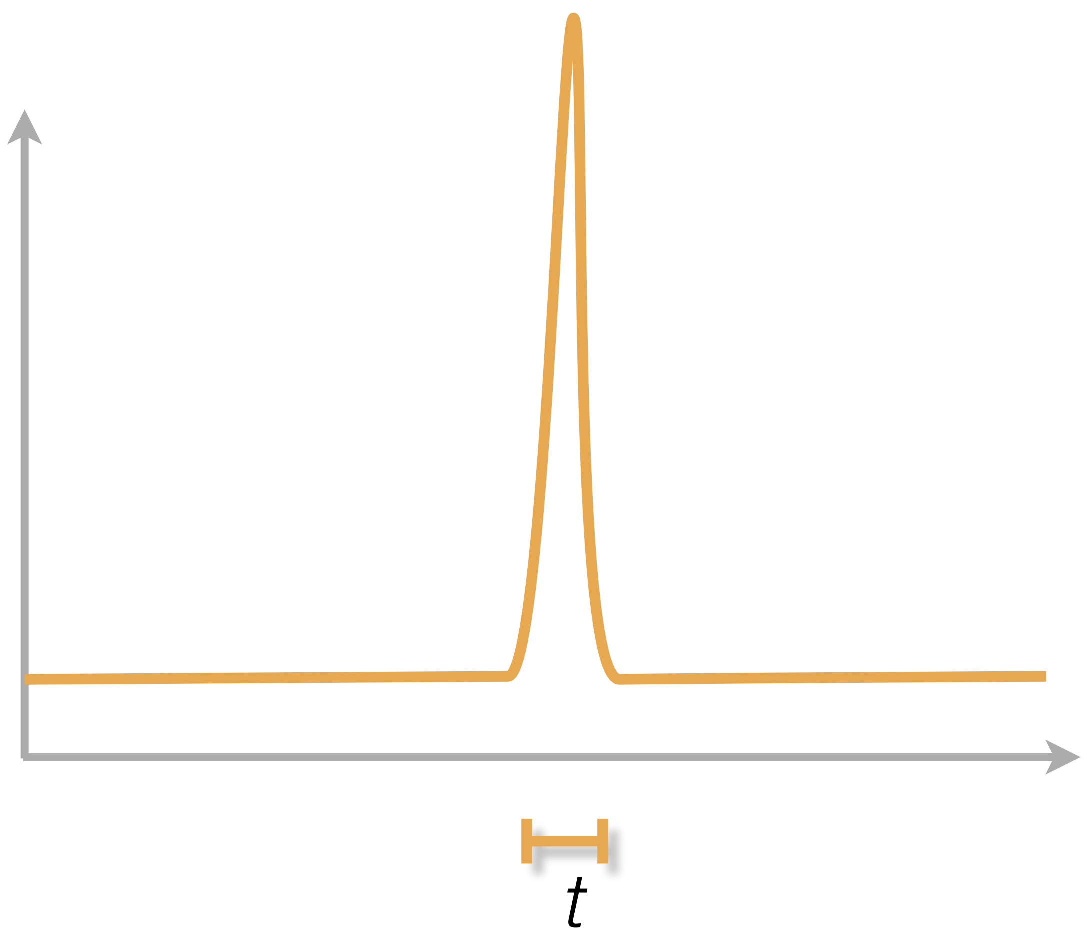


- Velocity

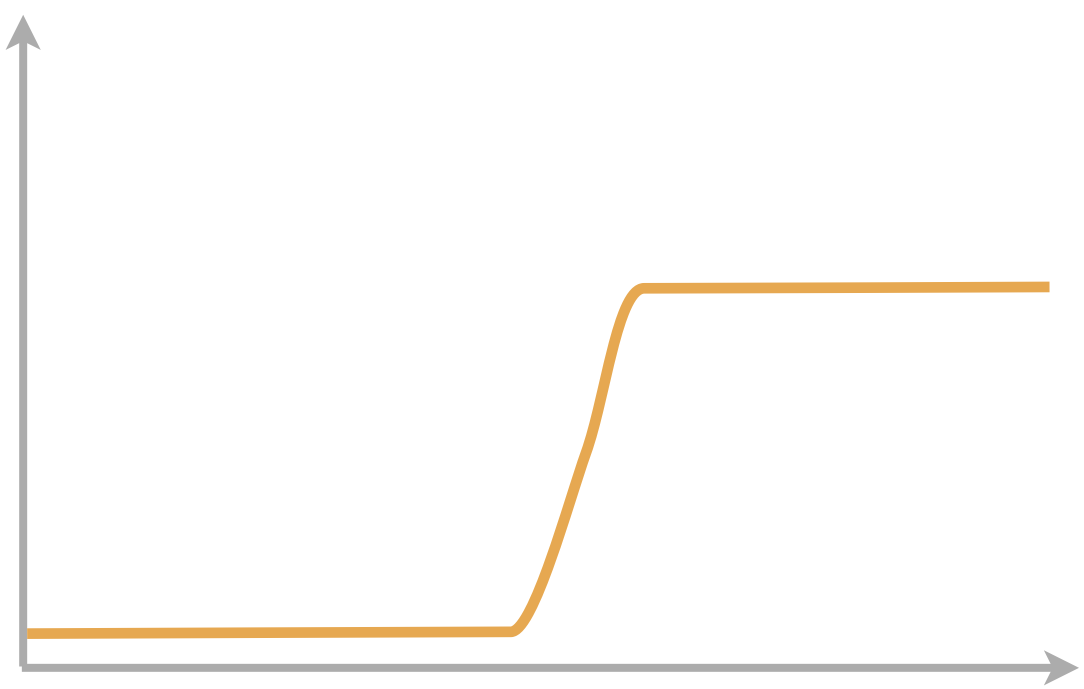


Very Hard Collision

- Force

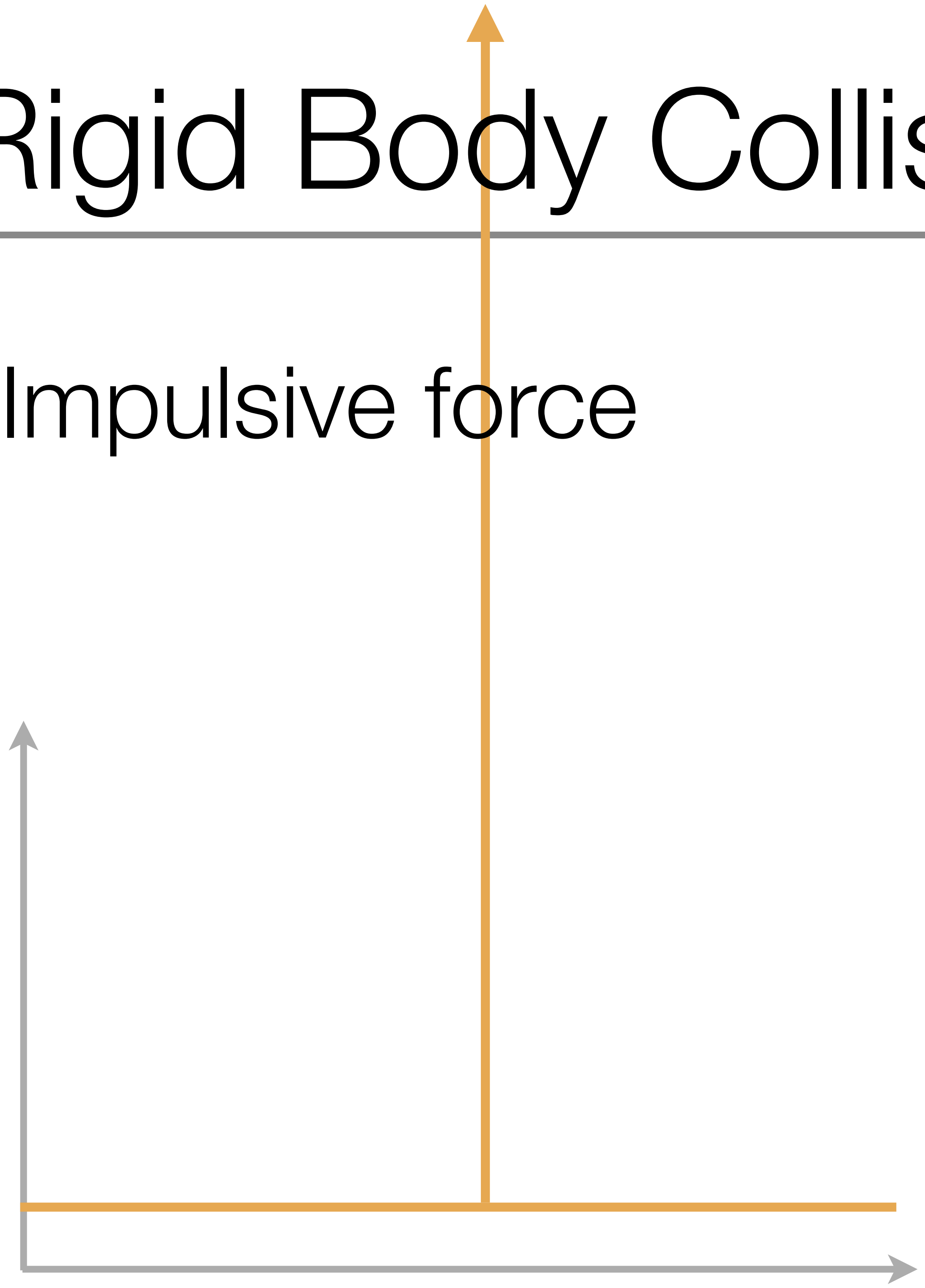


- Velocity



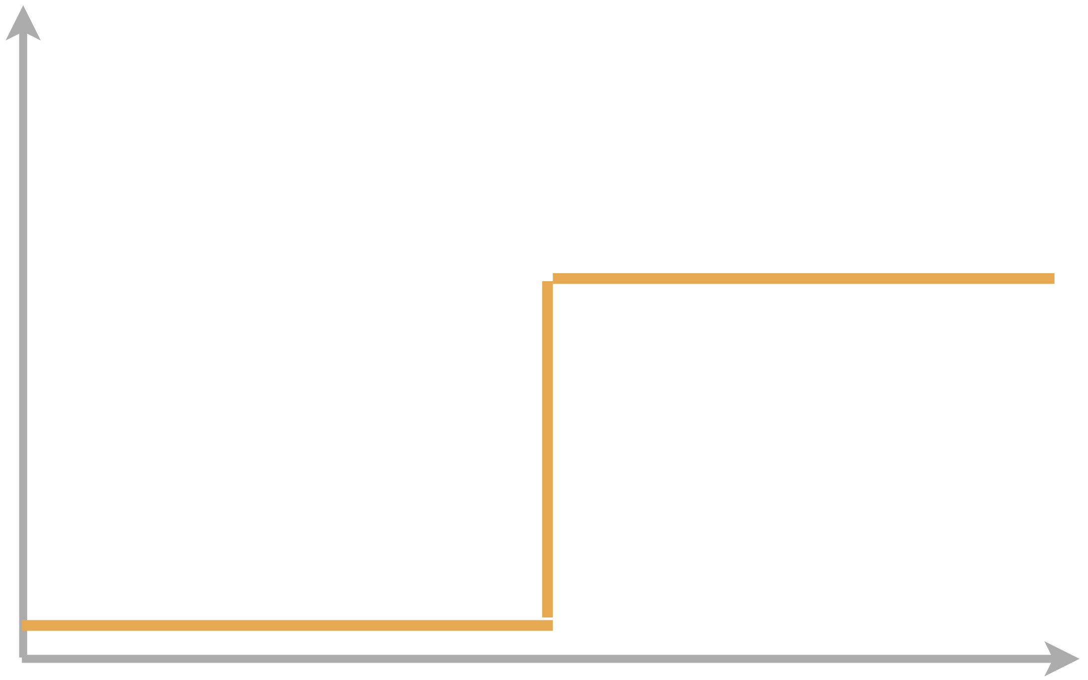
Rigid Body Collision

- Impulsive force



t=0 , infinite force

- Velocity



Impulses

- Fully rigid body would exert infinite elastic force over zero time interval
- Immediate velocity change, units like momentum (not force!)
- To avoid singularity, apply impulses that change velocity directly
- Use: $\mathbf{J} = m\Delta\mathbf{v}$ (no time step!)
- Instead of: $\Delta\mathbf{v} = h\mathbf{F}/m$

So much for 2D...

Rigid Bodies - Moving to 3D

- Positions are easy: 1 new axis
  - Existing integration methods fully hold
- Orientations are quite different
  - Rotation matrices
  - Quaternions (most game engines use this)
  - Exp. matrices

Angular Velocity in 3D

- So far, angular velocity was only the z component of the angular velocity vector
- In 3D, same principle for general vector ***w***
  - Along axis of rotation
  - Speed of rotation is given by norm of ***w***
  - But now all three components are used...

Inertia Tensor in 3D

- Continuos case: $\mathbf{I} = \int_V \rho(\mathbf{x}) \, \left(||\mathbf{x}||^2 - \mathbf{x}\,\mathbf{x}^T \right) \mathrm{d}V$
  (Fun exercise: calculate for a few basic shapes)

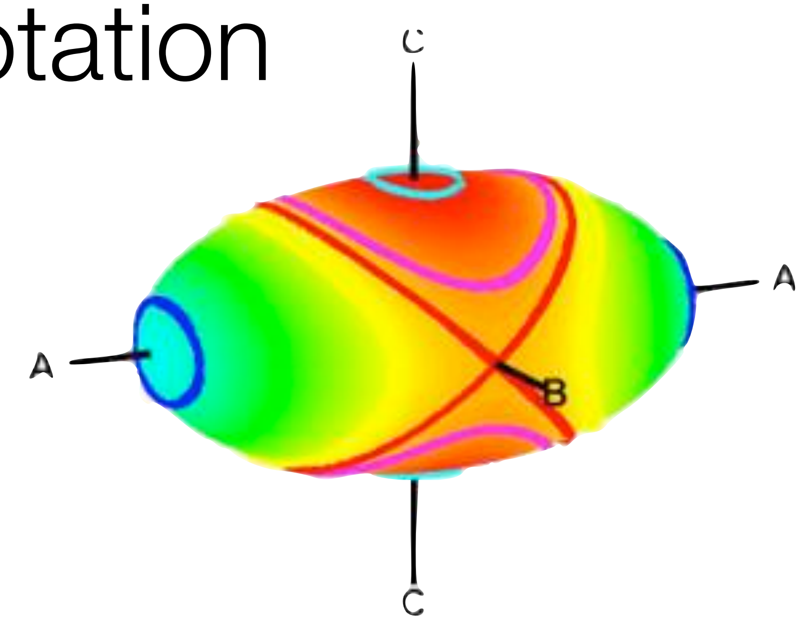
- Discrete: mass-weighted co-variance matrix of body coordinate positions:

$$\mathbf{C} = \sum_n m_n \mathbf{x}_n \mathbf{x}_n^T \qquad \mathbf{I} = \mathbf{Id}\ \mathrm{trace}(\mathbf{C}) - \mathbf{C}$$

  - has to be invertible! No zero eigenvalues...

Example - Axes of Rotation

- Inertia tensor for box has 3 eigenvalues
- Largest & smallest one are stable
- Intermediate one leads to unstable rotation
- Same: ellipse with axes A > B > C



Updating the Inertia Tensor

- In 3D, the inertia tensor depends on the current orientation of the body!
- Luckily, we can compute this from the initial one

$$\mathbf{I}_{current} = \mathrm{Rot}_{\mathbf{r}}\ \mathbf{I}_0\ \mathrm{Rot}_{\mathbf{r}}^{-1} = \mathrm{Rot}_{\mathbf{r}}\ \mathbf{I}_0\ \mathrm{Rot}_{\mathbf{r}}^{T}$$

- Why? Used as $\mathbf{L} = \mathbf{I}\mathbf{w}$
  - > Transform angular velocity into initial orientation, multiply with inertia tensor, transform back
  - Same holds for inverse (used in practice)

Angular Motion in 3D

- Small but important detail: it's not the angular velocity that is constant without forces, but the angular momentum
- Angular velocity can change without external forces and without temporal change of angular momentum
- Happens when:
  - Body has rotational velocity axis that is not a symmetry axis for body (i.e. angular momentum and angular velocity point in different directions)

Newton's 2nd Law for Rotations

- Given forces we can now compute the change of angular velocity over time:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{L} = \mathbf{q} \qquad \mathbf{w} = \mathbf{I}^{-1}\ \mathbf{L}$$

$$\mathbf{L}(t+h) = \mathbf{L}(t) + h\mathbf{q}$$
$$\mathbf{I}^{-1} = \mathrm{Rot}_{\mathbf{r}}\ \mathbf{I}_0^{-1}\ \mathrm{Rot}_{\mathbf{r}}^T$$
$$\mathbf{w}(t+h) = \mathbf{I}^{-1}\ \mathbf{L}(t+h)$$

Note: integrates angular momentum over time, not angular velocity!

Points vs. Rigid Bodies (3D)

- For particles:
  - Position ***x***
  - Velocity ***v***
- Dynamics:

$$\mathbf{v}(t) = \frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t}$$

$$\mathbf{a}(t) = \frac{\mathrm{d}\mathbf{v}(t)}{\mathrm{d}t}$$

- For a rigid body:
  - Position ***x***
  - Orientation ***r***
  - Linear velocity ***v***
  - Angular velocity ***w***
- Angular dynamics:

$$\mathbf{q}(t) = \sum_i \mathbf{x}_i \times \mathbf{f}_i$$

$$\mathbf{L}(t+h) = \mathbf{L}(t) + h\mathbf{q}$$

$$\mathbf{w}(t+h) = \mathbf{I}^{-1}\ \mathbf{L}(t+h)$$

Simulation Algorithm 3D

Pre-compute:

$$M \leftarrow \sum_i m_i$$

$$\mathbf{x}'_{cm} \leftarrow \sum_i \mathbf{x}'_i m_i / M$$

$$\mathbf{x}_i \leftarrow \mathbf{x}'_i - \mathbf{x}'_{cm}$$

$$\mathbf{I}^{-1} \leftarrow \sum_i m_i \ \ldots$$

Initialize:

$$\mathbf{x}_{cm}, \mathbf{v}_{cm}, \mathbf{r}, \mathbf{L}$$

$$\mathbf{I}^{-1} \leftarrow \mathrm{Rot}_{\mathbf{r}}\ \mathbf{I}_0^{-1}\ \mathrm{Rot}_{\mathbf{r}}^T$$

$$\mathbf{w} \leftarrow \mathbf{I}^{-1}\mathbf{L}$$

External forces

$$\mathbf{F} \leftarrow \sum_i \mathbf{f}_i$$

$$\mathbf{q} \leftarrow \sum_i \mathbf{x}_i \times \mathbf{f}_i$$

Euler step

$$\mathbf{x}_{cm} \leftarrow \mathbf{x}_{cm} + h\mathbf{v}_{cm}$$

$$\mathbf{v}_{cm} \leftarrow \mathbf{v}_{cm} + h\mathbf{F}/M$$

$$\text{“}\,\mathbf{r} \leftarrow \mathbf{r} + h\mathbf{w}\,\text{“}$$

Depends on representation!

$$\mathbf{L} \leftarrow \mathbf{L} + h\mathbf{q}$$

$$\mathbf{I}^{-1} \leftarrow \mathrm{Rot}_{\mathbf{r}}\ \mathbf{I}_0^{-1}\ \mathrm{Rot}_{\mathbf{r}}^T$$

$$\mathbf{w} \leftarrow \mathbf{I}^{-1}\ \mathbf{L}$$

World position

$$\mathbf{x}_i^{world} \leftarrow \mathbf{x}_{cm} + \mathrm{Rot}_r \mathbf{x}_i$$

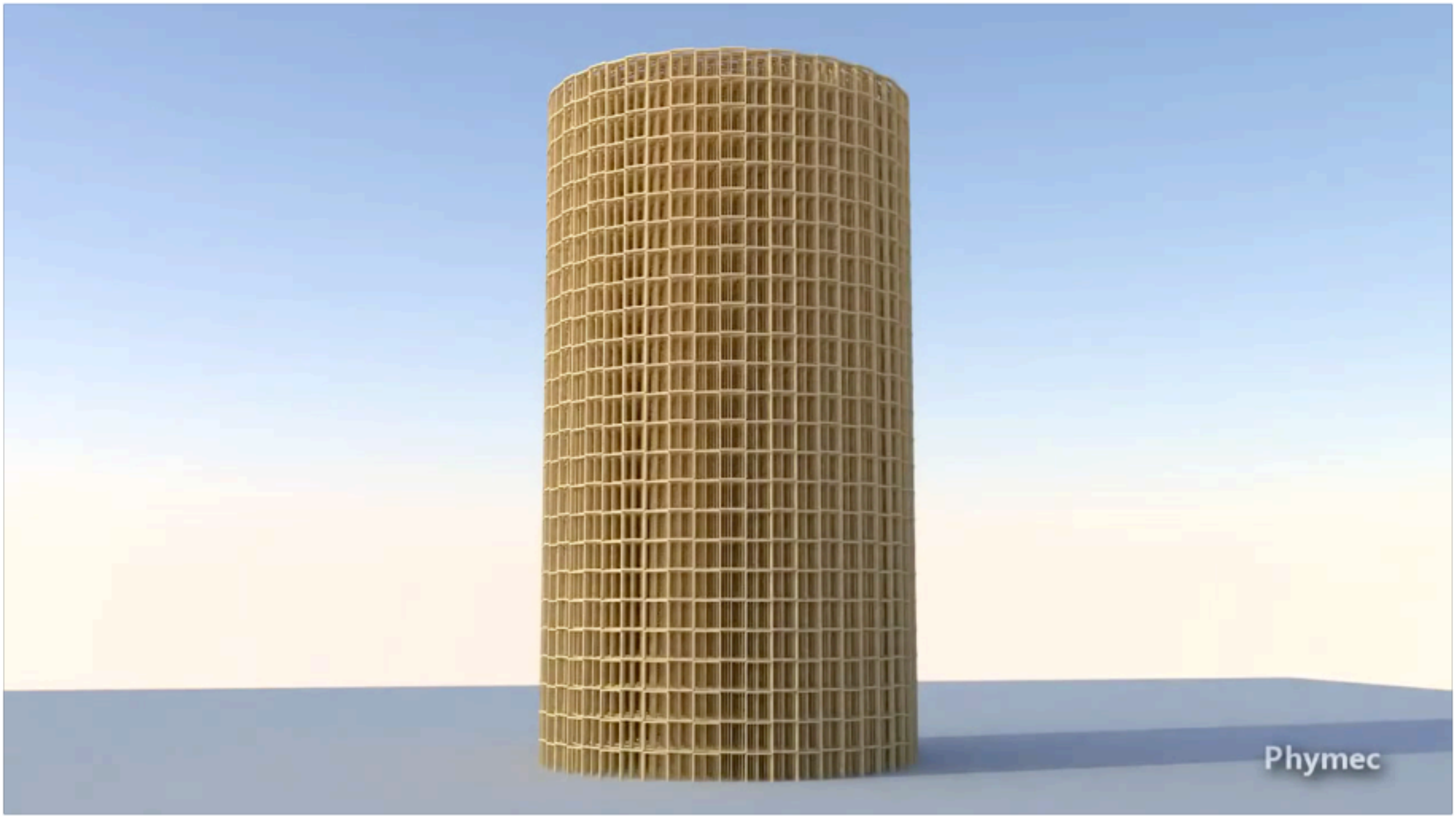
$$\mathbf{v}_i^{world} \leftarrow \mathbf{v}_{cm} + \mathbf{w} \times \mathbf{x}_i$$

Integrating the Orientation

- Example: Quaternion
  - General question - what is time derivative of orientation given as quaternion?
  - It turns out: $$\frac{\mathrm{d}\mathbf{r}}{\mathrm{d}t} = \frac{1}{2}\begin{pmatrix} 0 \\ \mathbf{w} \end{pmatrix}\mathbf{r};\ \ \mathbf{r} = (s, xi, yj, zk)$$
  - Thus, integrate with:

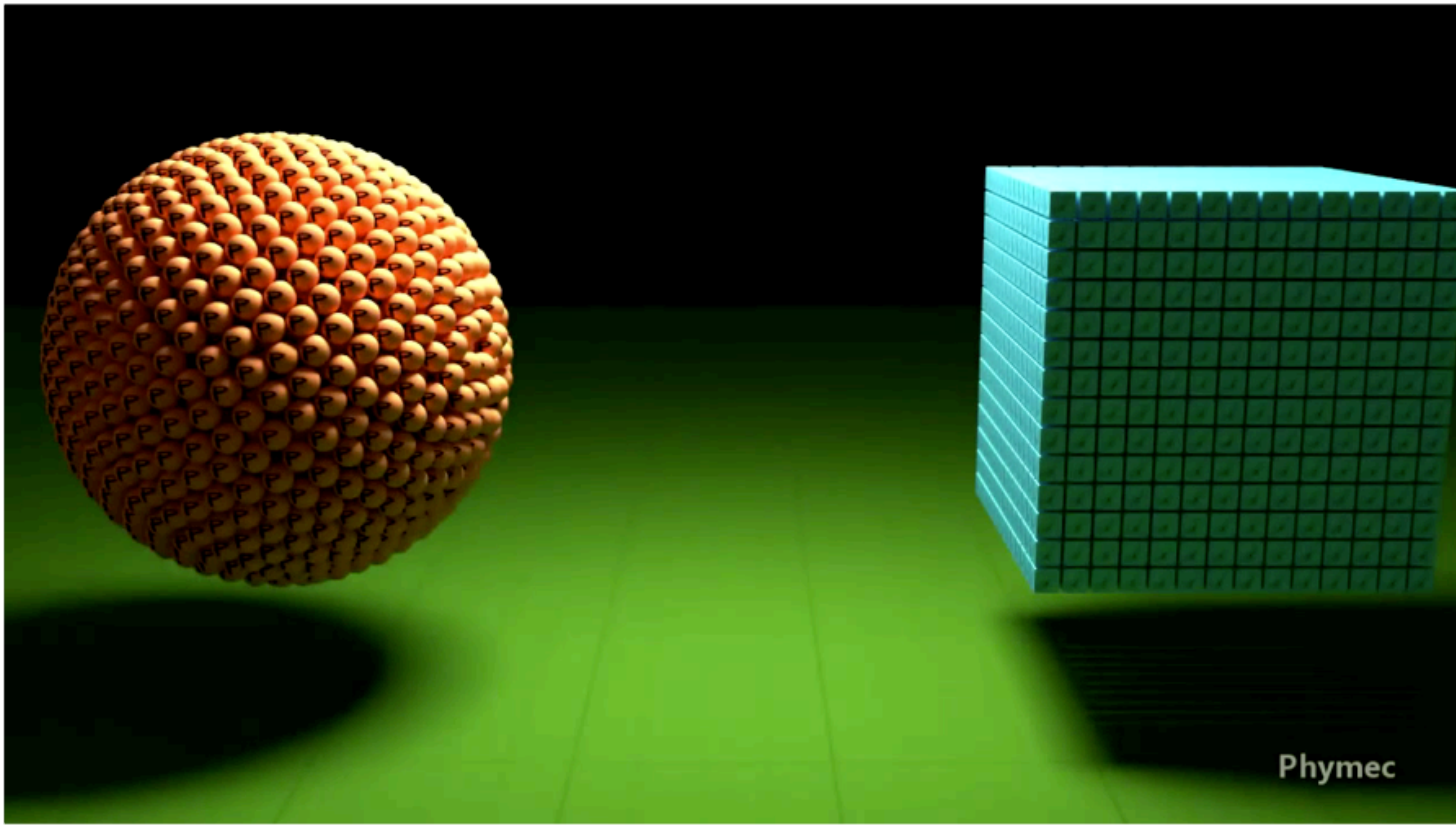
$$\mathbf{r}' = \mathbf{r} + h/2\begin{pmatrix} 0 \\ \mathbf{w} \end{pmatrix}\mathbf{r}$$

How well does this work?



Bullet Physics Engine / Blender. Video by Phymec

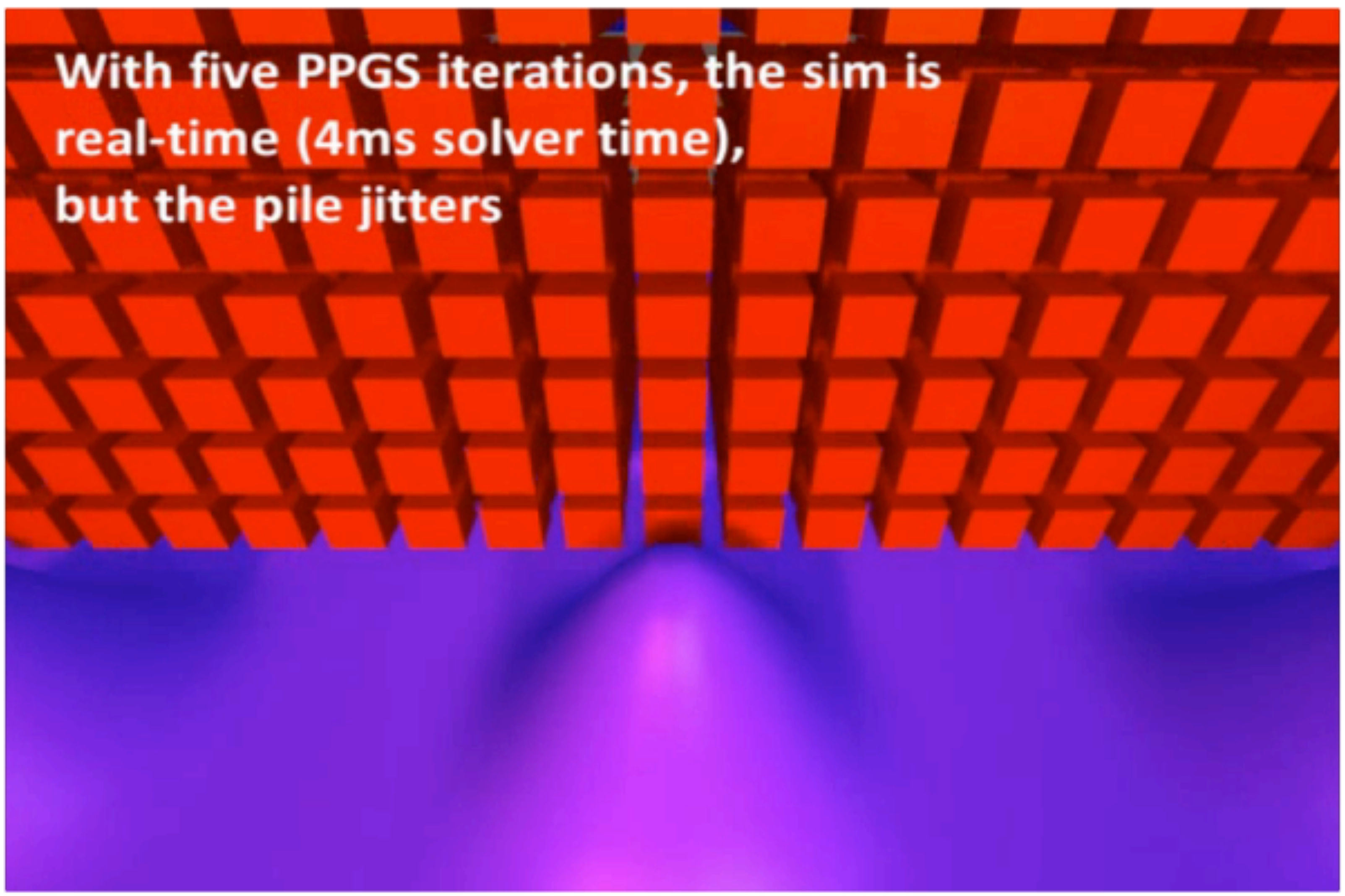
Rigidity



Bullet Physics Engine / Blender. Video by Phymec

How well does this work?

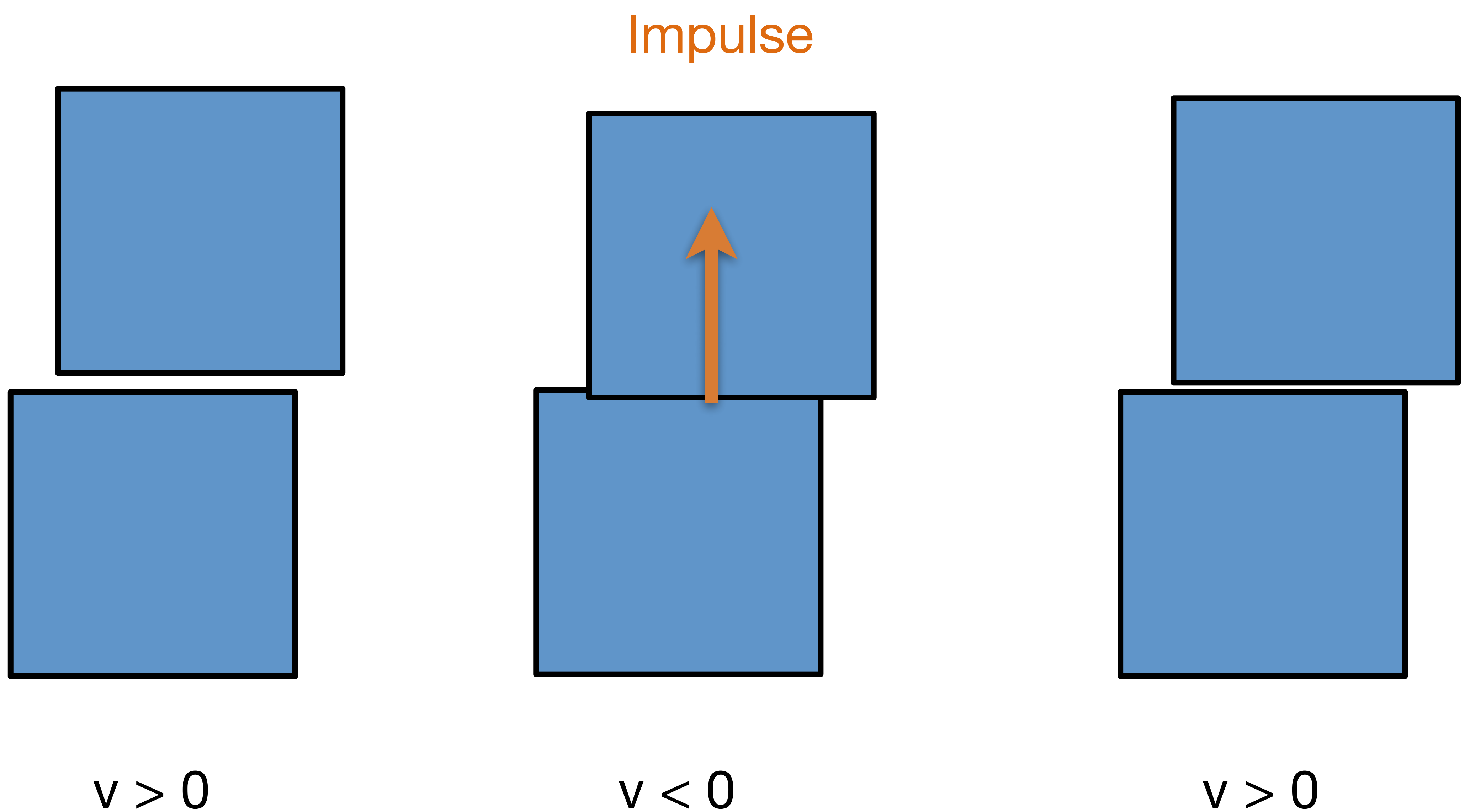
- Collision handling is problematic!
  - Stacking / Resting contact is hard



Tonge et al. 2012
Mass splitting for jitter-free RB simulations

How well does this work?

- Resting contact



References

- David Baraff's SIGGRAPH course
  http://www.cs.cmu.edu/~baraff/sigcourse
- David Eberly: Game Physics (book)
  www.geometrictools.com
- Chris Hecker: Rigid Body Dynamics
  chrishecker.com/Rigid_Body_Dynamics