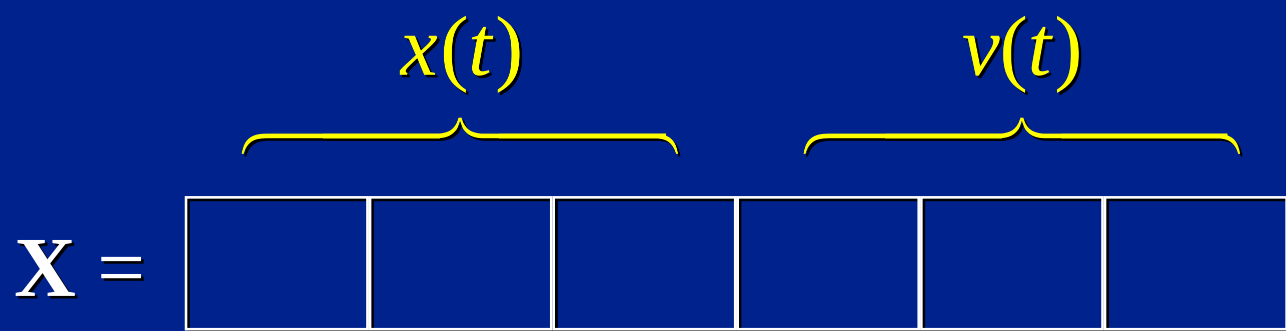
# Rigid Body Dynamics

*David Baraff*

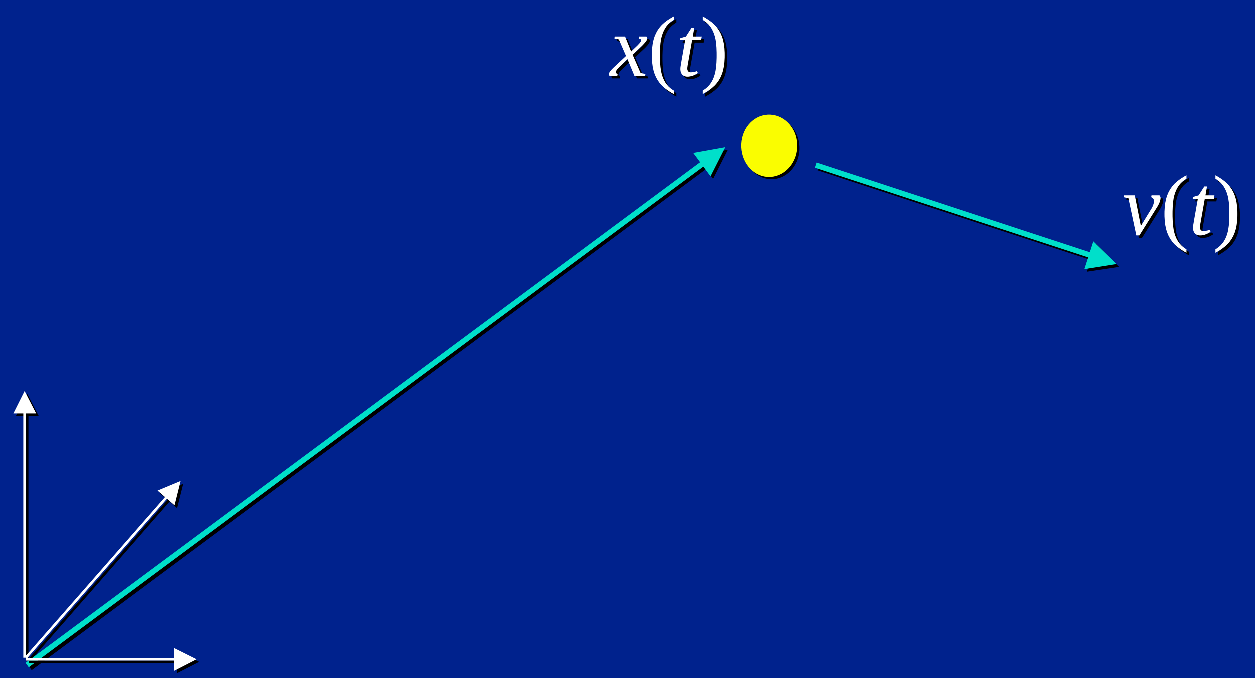
PIXAR
ANIMATION STUDIOS

# Particle State

$$\mathbf{X}(t) = \begin{pmatrix} x(t) \\ v(t) \end{pmatrix}$$



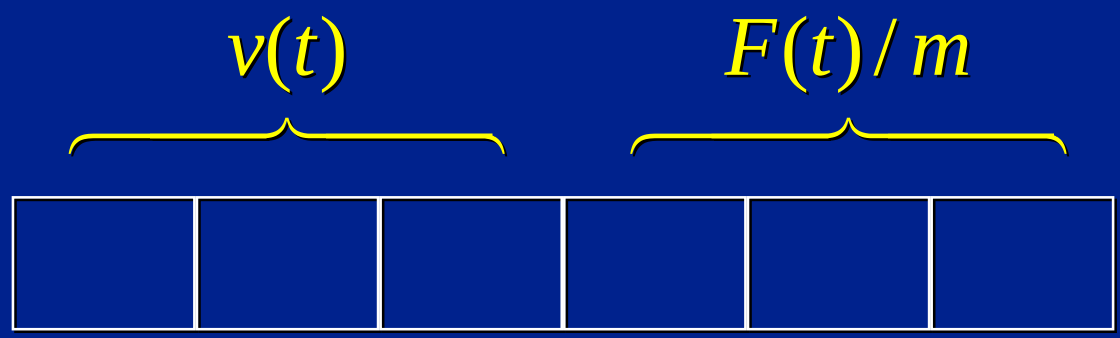
# Particle Motion



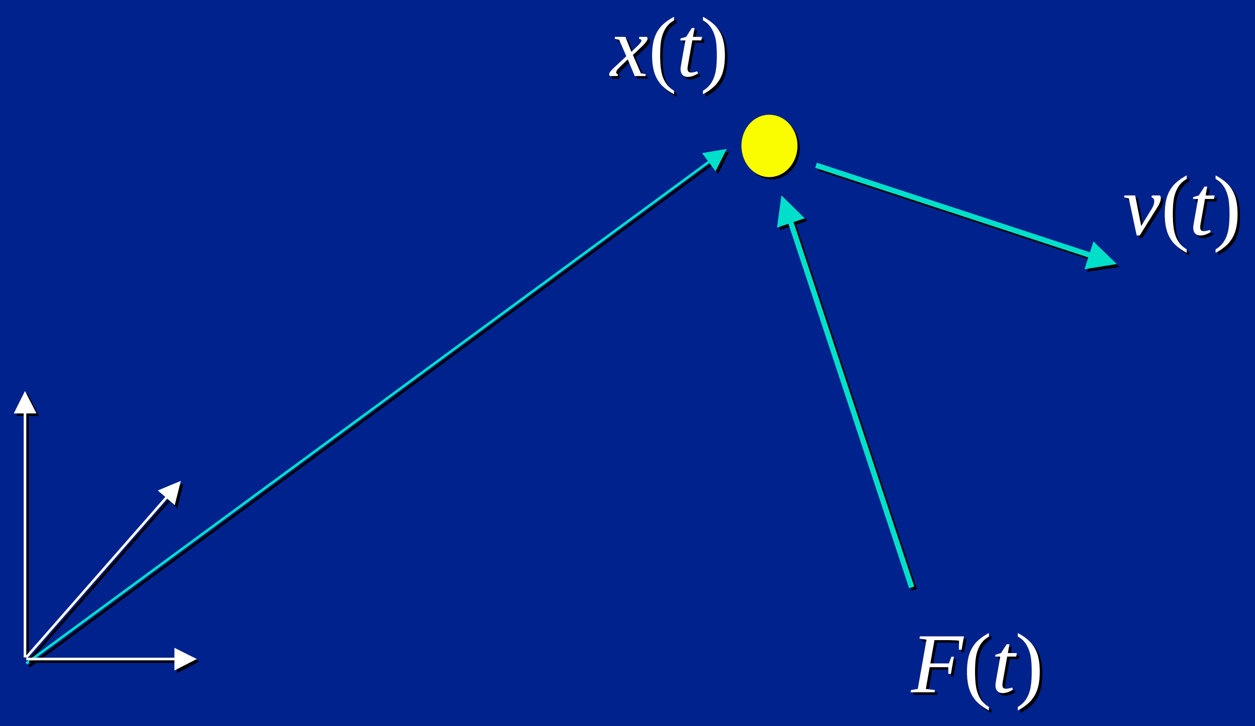
# State Derivative

$$\frac{d}{dt}\mathbf{X}(t) = \frac{d}{dt}\begin{pmatrix} x(t) \\ v(t) \end{pmatrix} = \begin{pmatrix} v(t) \\ F(t)/m \end{pmatrix}$$

$$\frac{d}{dt}\mathbf{X} =$$



# Particle Dynamics



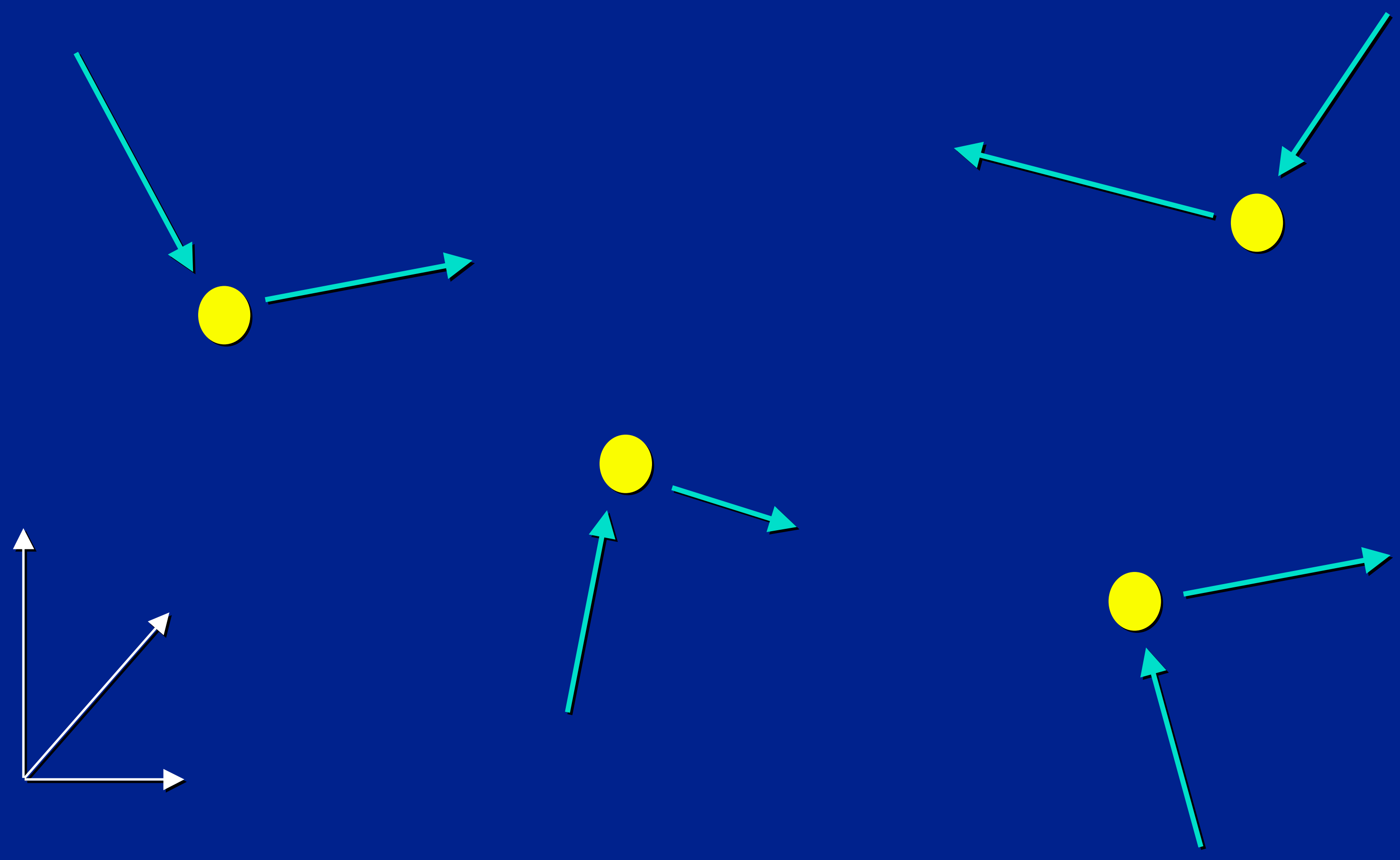
# State Derivative

$$\frac{d}{dt}\mathbf{X} = \frac{d}{dt}\begin{pmatrix} x_1(t) \\ v_1(t) \\ \vdots \\ x_n(t) \\ v_n(t) \end{pmatrix} = \begin{pmatrix} v_1(t) \\ F_1(t)/m_1 \\ \vdots \\ v_n(t) \\ F_n(t)/m_n \end{pmatrix}$$

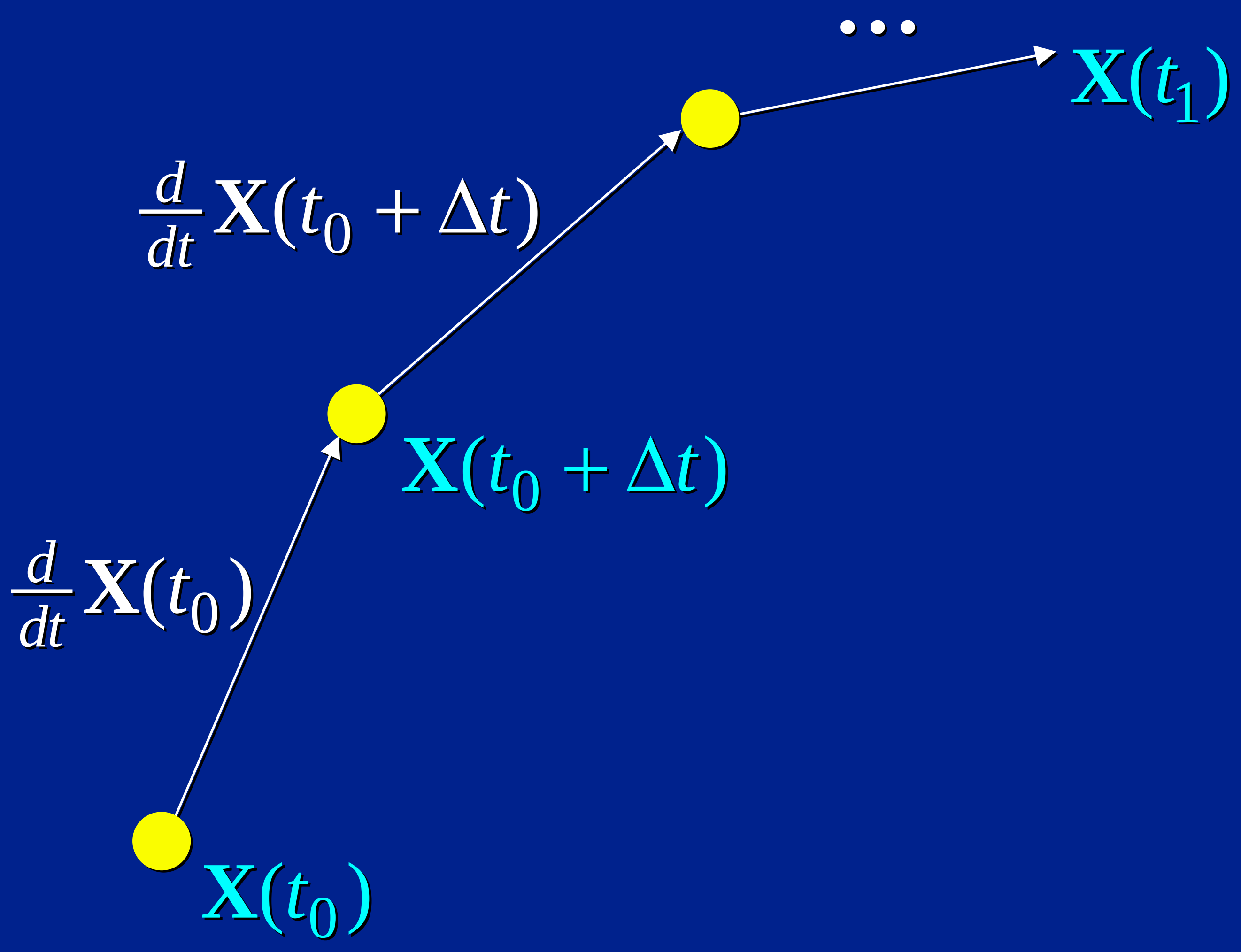
$$\frac{d}{dt}\mathbf{X} =$$

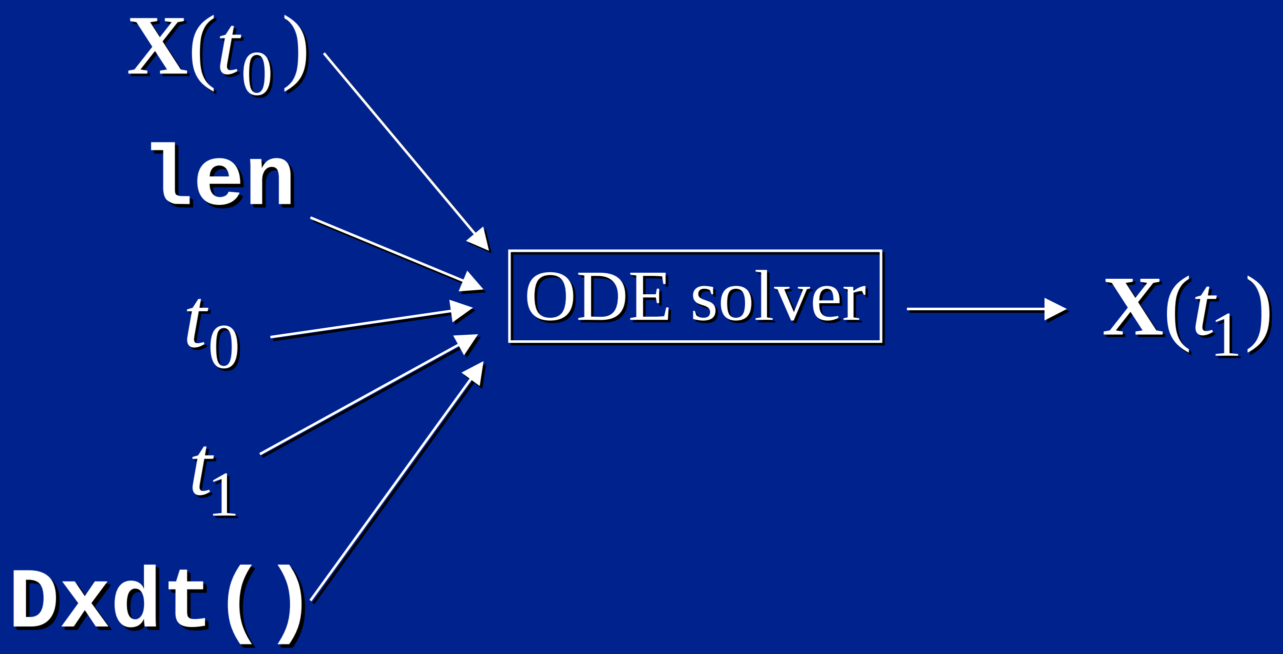
| | | | | |
|---|---|---|---|---|
| | | ••• 6$n$ elements ••• | | |

# Multiple Particles



# ODE solution



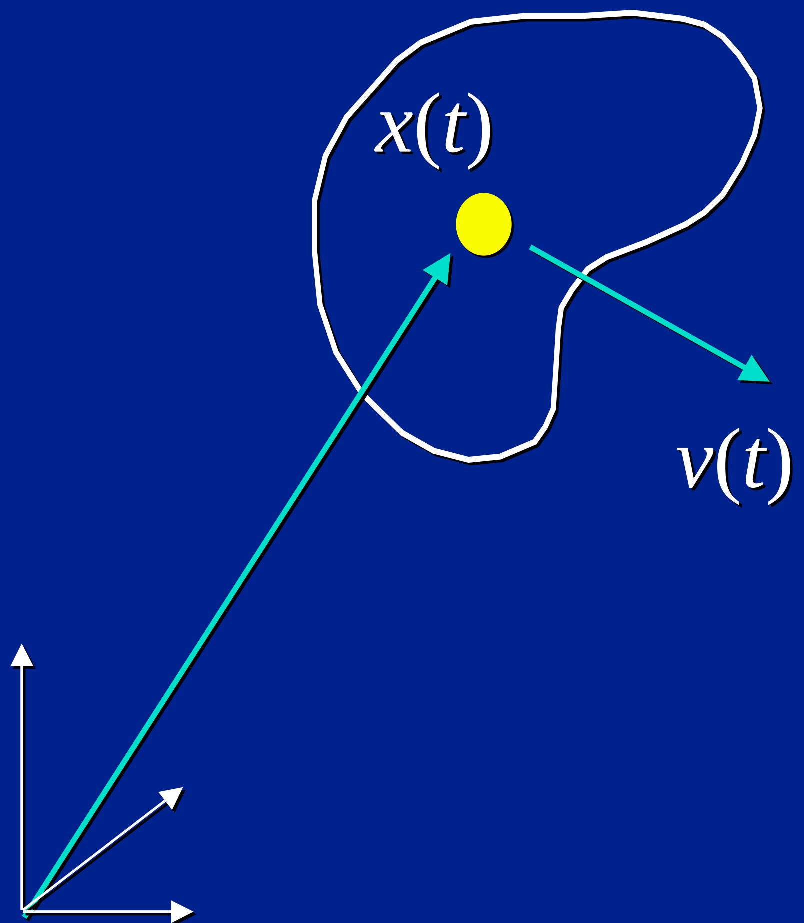



```
void Dxdt(double t, double x[],
                    double xdot[])
```

# Dxdt()

$$\mathbf{X}(t) = \begin{bmatrix} x_1(t) \\ v_1(t) \\ \vdots \\ x_n(t) \\ v_n(t) \end{bmatrix} \qquad \frac{d}{dt}\mathbf{X}(t) = \begin{bmatrix} v_1(t) \\ F_1(t)/m_1 \\ \vdots \\ v_n(t) \\ F_n(t)/m_n \end{bmatrix}$$

# Rigid Body State

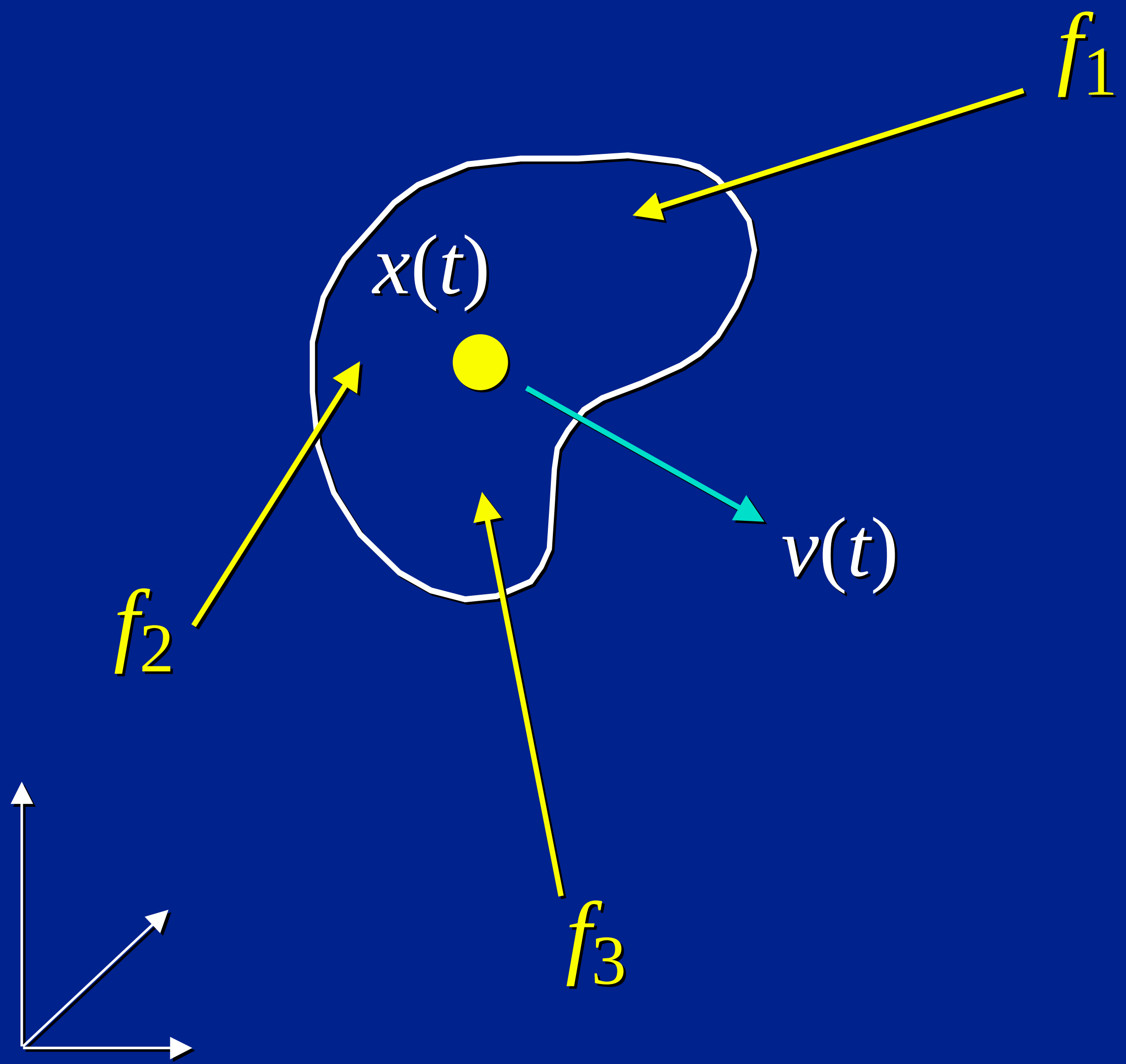



$$\mathbf{X}(t) = \begin{pmatrix} x(t) \\ ? \\ v(t) \\ ? \end{pmatrix}$$

# Rigid Body Equation of Motion

$$\frac{d}{dt}\mathbf{X}(t) = \frac{d}{dt}\begin{pmatrix} x(t) \\ ? \\ Mv(t) \\ ? \end{pmatrix} = \begin{pmatrix} v(t) \\ ? \\ F(t) \\ ? \end{pmatrix}$$

# Net Force




$$F(t) = \sum f_i$$

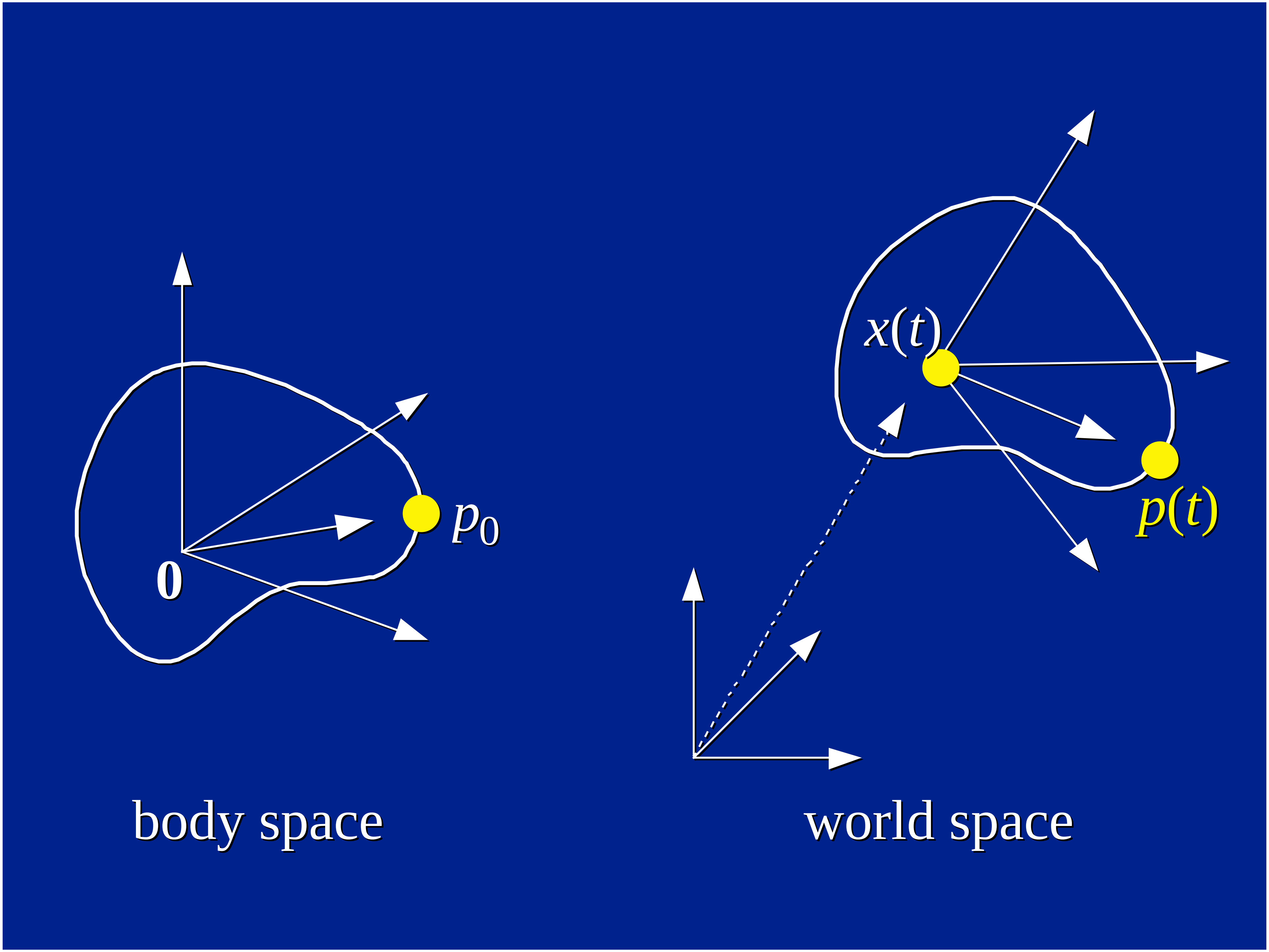
# Orientation

We represent orientation as a rotation matrix[†] $\mathbf{R}(t)$. Points are transformed from body-space to world-space as:

$$p(t) = \mathbf{R}(t) p_0 + x(t)$$

---

[†] He's lying. Actually, we use quaternions.

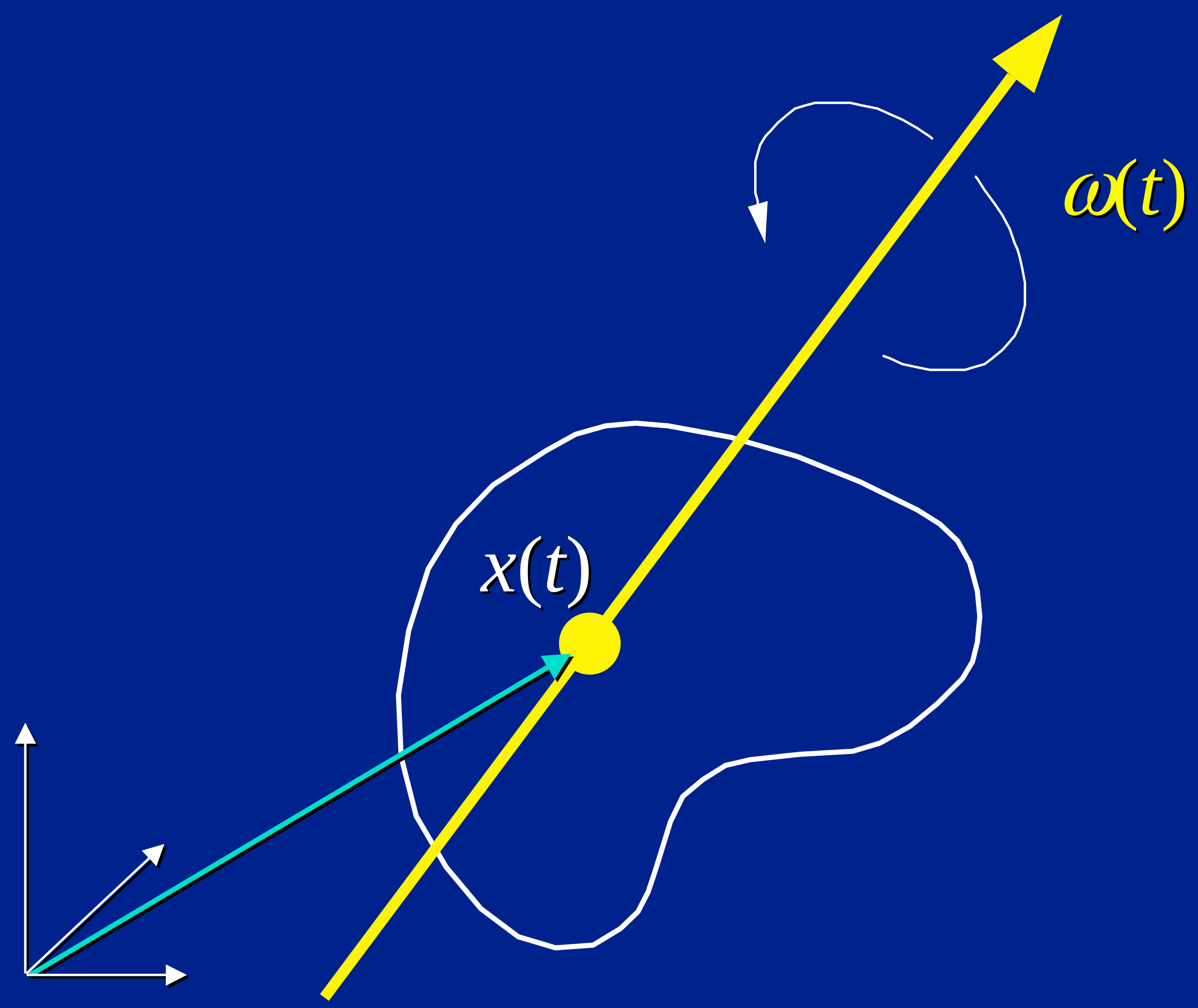

$p_0$
0
body space
$x(t)$
$p(t)$
world space

# Angular Velocity

We represent angular velocity as a vector $\omega(t)$, which encodes both the axis of the spin and the speed of the spin.

How are $\mathbf{R}(t)$ and $\omega(t)$ related?

# Angular Velocity Definition



# Angular Velocity

- $\dot{\mathbf{R}}(t)$ and $\omega(t)$ are related by:

$$\frac{d}{dt}\mathbf{R}(t) = \begin{pmatrix} 0 & -\omega_z(t) & \omega_y(t) \\ \omega_z(t) & 0 & -\omega_x(t) \\ -\omega_y(t) & \omega_x(t) & 0 \end{pmatrix} \mathbf{R}(t)$$

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

# Rigid Body Equation of Motion

$$\frac{d}{dt}\mathbf{X}(t) = \frac{d}{dt}\begin{pmatrix} x(t) \\ \mathbf{R}(t) \\ Mv(t) \\ \langle \omega(t) \rangle \end{pmatrix} = \begin{pmatrix} v(t) \\ \omega(t)^{*}\mathbf{R}(t) \\ F(t) \\ \textbf{?} \end{pmatrix}$$

Need to relate $\dot{\omega}(t)$ and mass distribution to $F(t)$.

# Inertia Tensor

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

diagonal terms

$$I_{xx} = M\int_V (y^2 + z^2)\, dV$$

off-diagonal terms

$$I_{xy} = -M\int_V xy\, dV$$

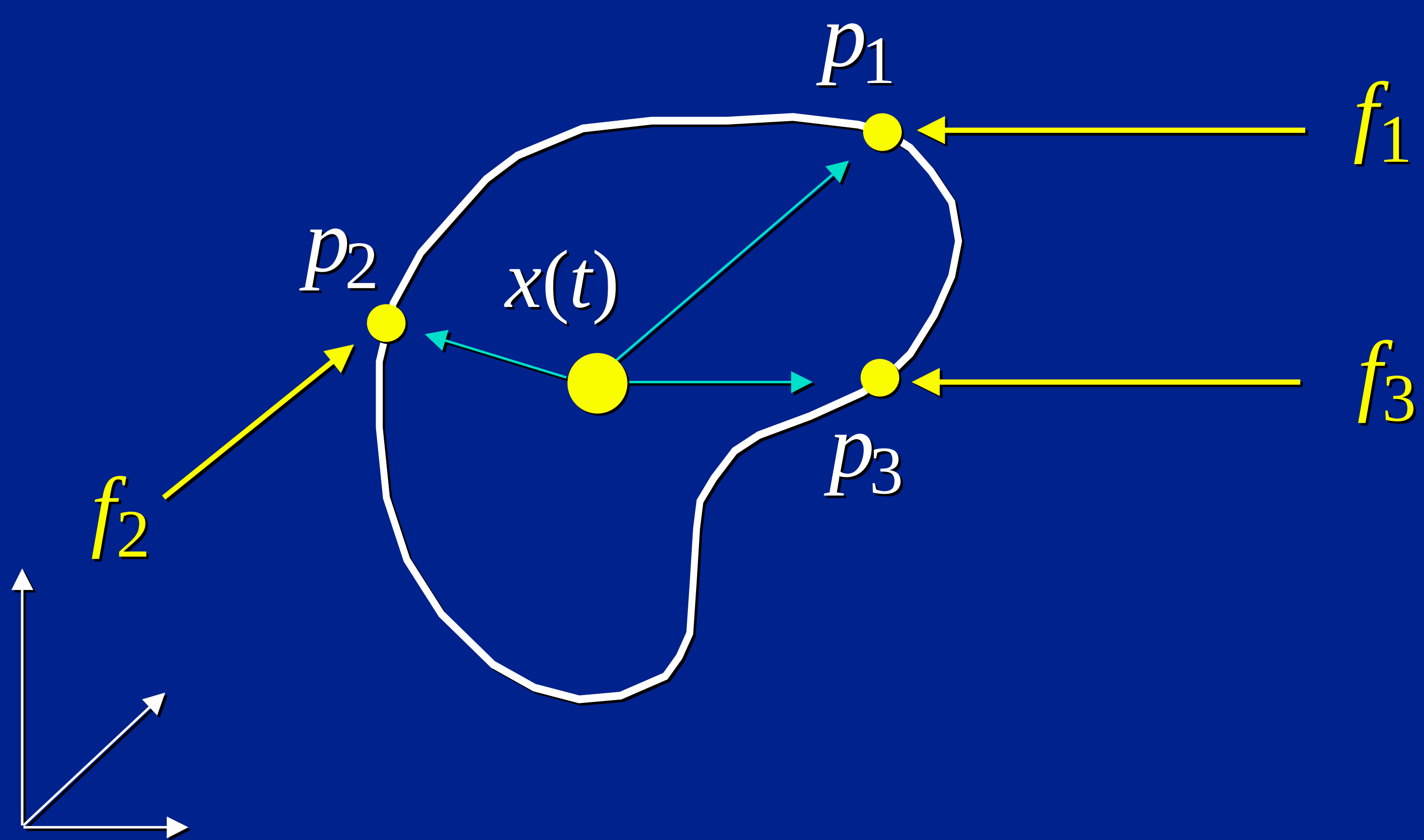
# Rigid Body Equation of Motion

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

$P(t)$ – linear momentum

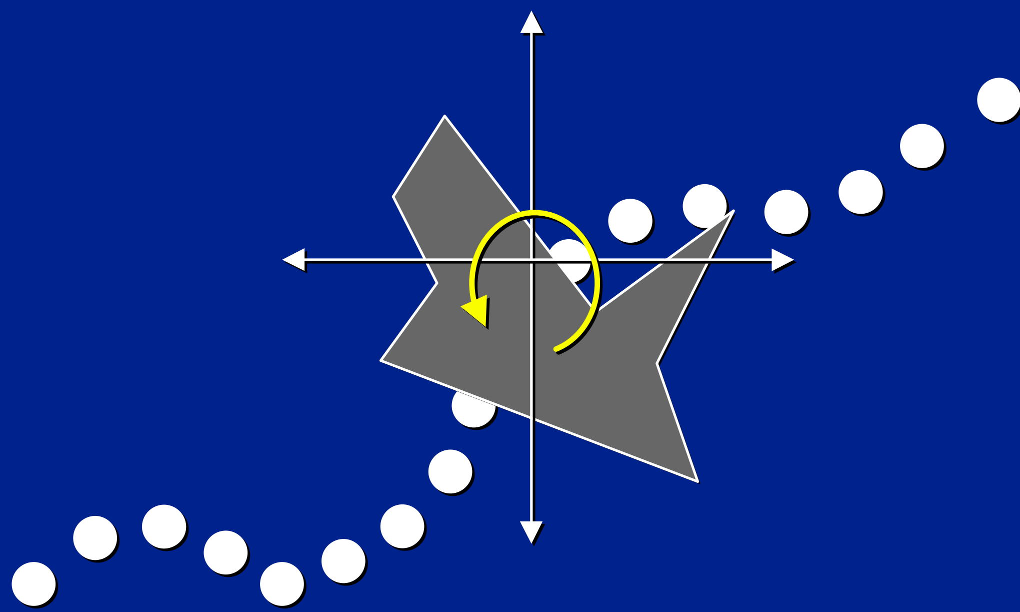
$L(t)$ – angular momentum

# Net Torque



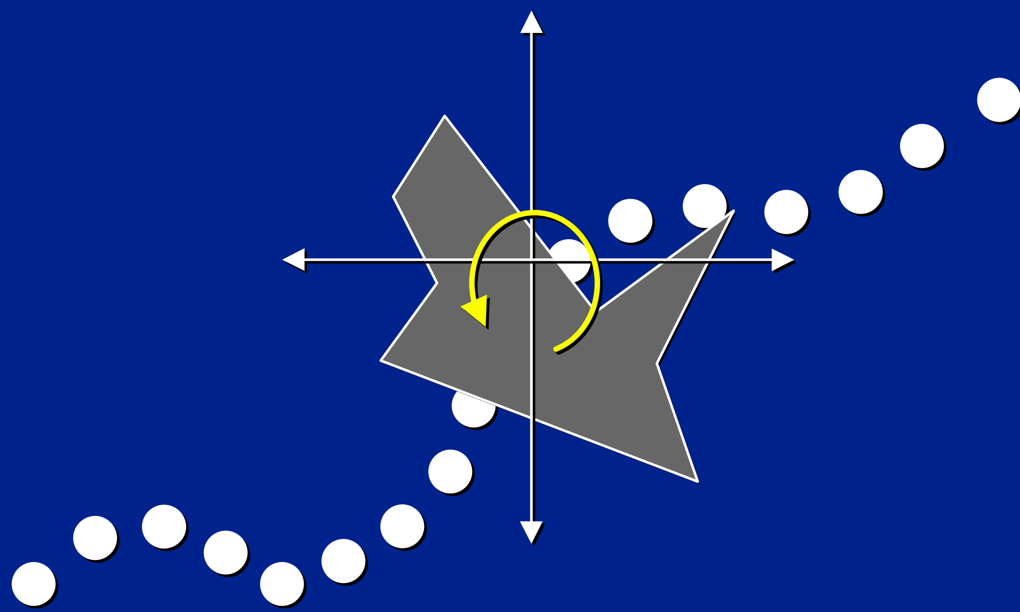

$$\tau(t) = \sum \left( p_i - x(t) \right) \times f_i$$

# Inertia Tensors Vary in World Space...



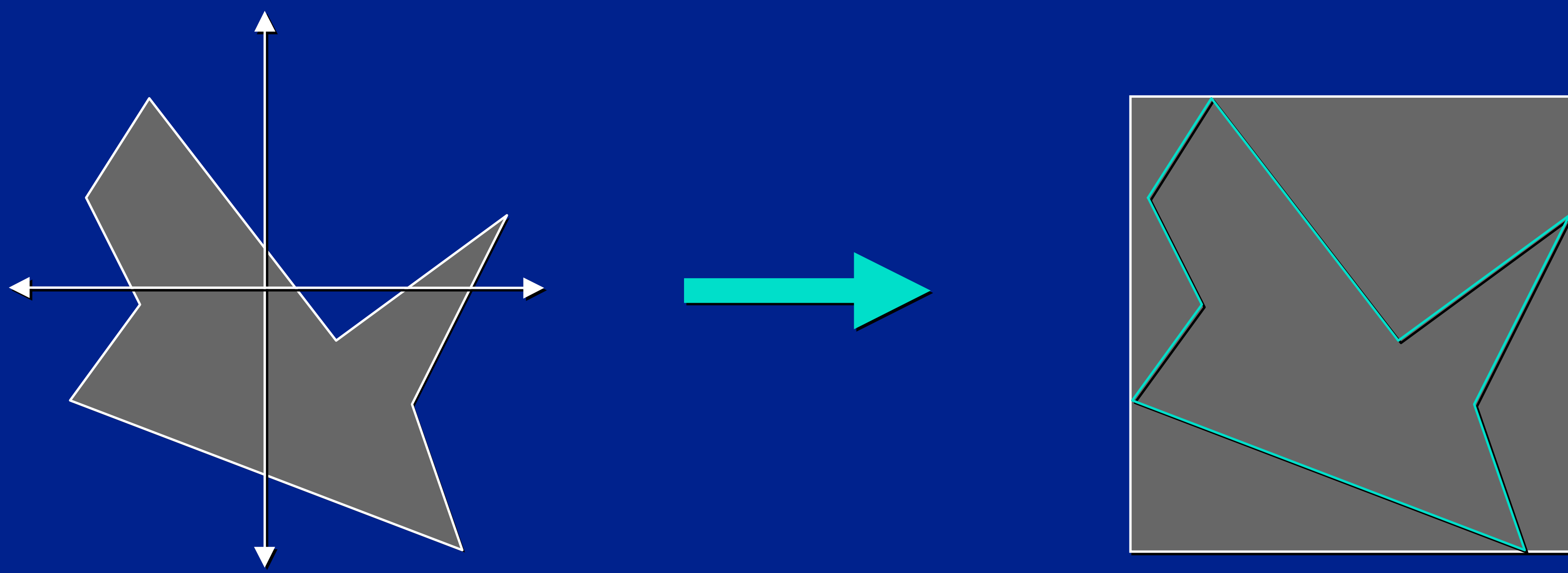
$$I_{xx} = M \int_V (y^2 + z^2)\, dV \qquad I_{xy} = -M \int_V xy\, dV$$

# ... but are Constant in Body Space



$$\mathbf{I}(t) = \mathbf{R}(t)\mathbf{I}_{\text{body}}\mathbf{R}(t)^T$$

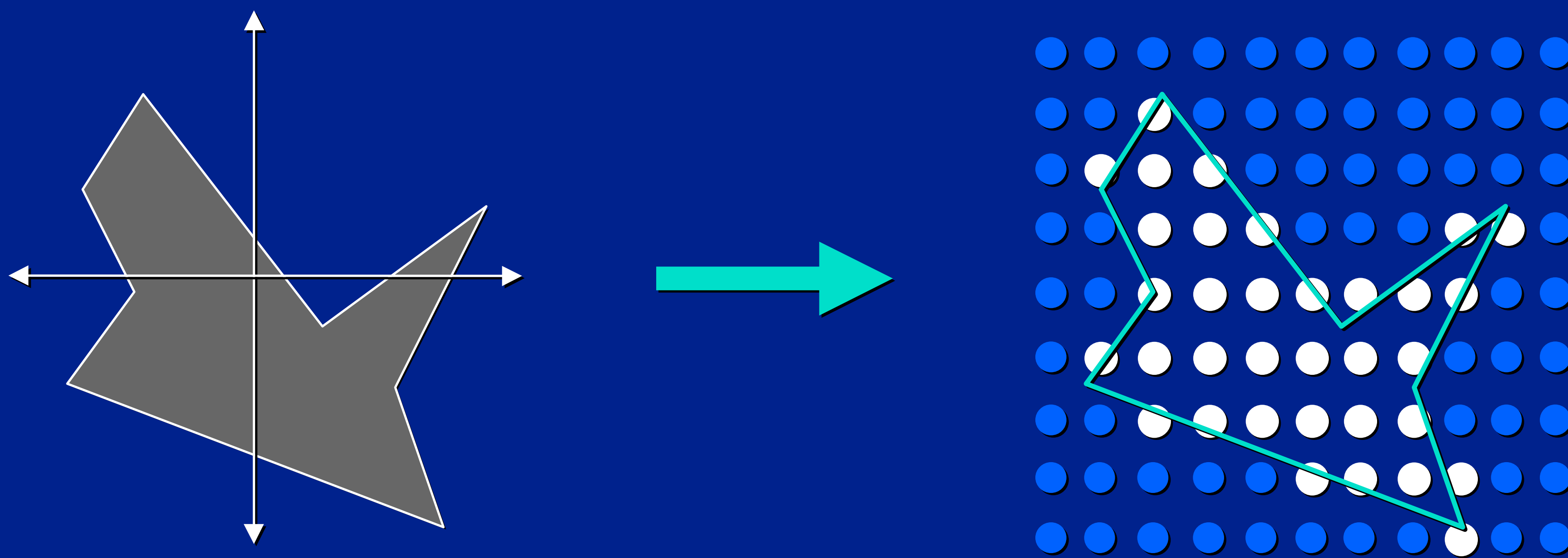
# Approximating $I_{body}$: Bounding Boxes



Pros: Simple.

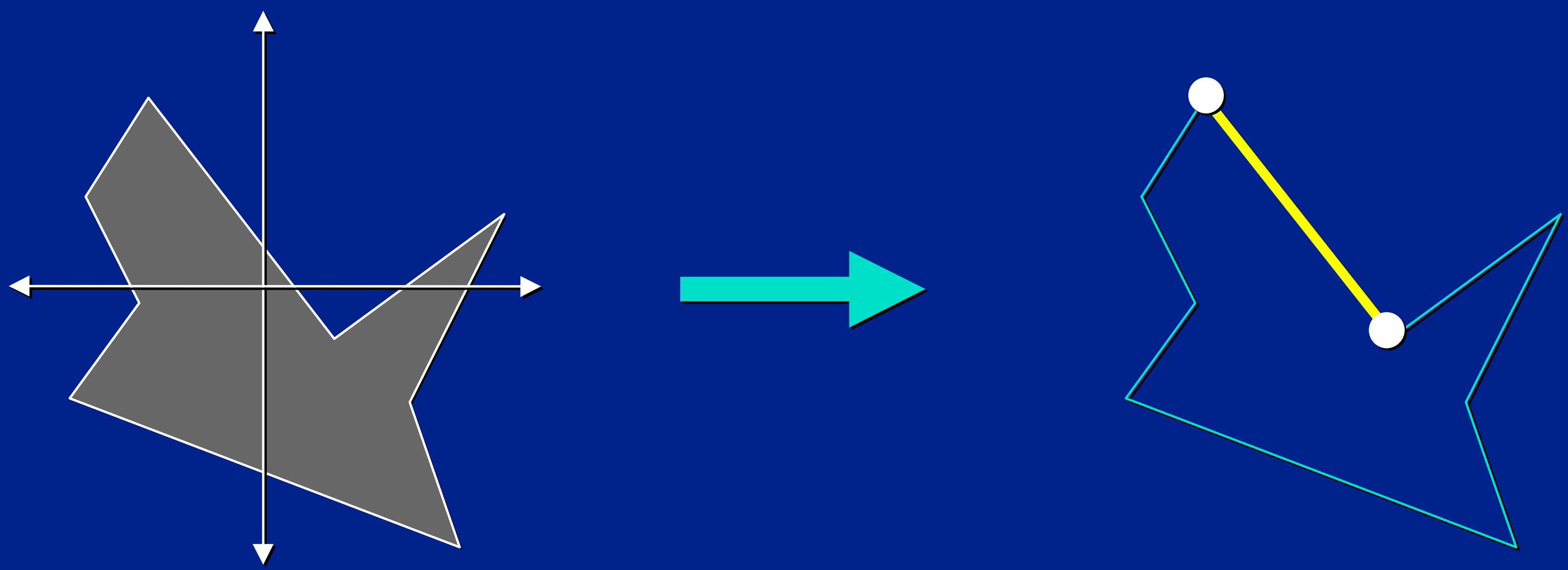
Cons: Bounding box may not be a good fit.
Inaccurate.

# Approximating $I_{body}$: Point Samping



Pros: Simple, fairly accurate, no B-rep needed.
Cons: Expensive, requires volume test.

# Computing $I_{body}$: Green's Theorem (2x!)



Pros: Simple, exact, no volumes needed.
Cons: Requires boundary representation.
Code: http://www.acm.org/jgt/papers/Mirtich96

# What's in the Course Notes

1. Implementation of `Dxdt()` for rigid bodies (bookkeeping, data structures, computations)
2. Quaternions—derivations and code
3. Miscellaneous formulas and examples
4. Derivations for force and torque equations, center of mass, inertia tensor, rotation equations, velocity/acceleration of points