CS 287 Lecture 21 (Fall 2019)
Physics Simulation

Pieter Abbeel
UC Berkeley EECS

A lightning tour of physics simulation

- Newton’s Laws – Rigid Body Motion
- Lagrangian Formulation
- Continuous Time → Discrete Time
- Contact / Collisions

Want to learn more?

- **Featherstone book**: Rigid Body Dynamics Algorithms
- **Mujoco**
  - book: http://www.mujoco.org/book/computation.html
  - mujoco paper: https://homes.cs.washington.edu/~todorov/papers/TodorovIROS12.pdf
- **Bullet**
  - simulation: https://docs.google.com/presentation/d/1-UqEzGEHdskq8blwNWqdgnmUDwZDPjlZUvg437z7XCM/edit#slide=id.ga4b37291a_0_0
  - Constraint solving: https://docs.google.com/presentation/d/1wGUJ4neOhw5i4pQRfSGtZPE3CIm7MfmqfTp5aJKuFYM/edit#slide=id.ga4b37291a_0_0
- **constraints / collisions**: https://www.toptal.com/game/video-game-physics-part-iii-constrained-rigid-body-simulation

Newton

- Point mass: $F = ma$

- Rigid body:

$$F = ma$$

$$\tau = I\dot{\omega} + \omega \times (I\omega)$$

Lagrangian Dynamics -- Motivation

- Newton
  - Generally applicable
  - But can become a bit cumbersome in multi-body systems with constraints/internal forces
- Lagrangian dynamics method eliminates the internal forces from the outset and expresses dynamics w.r.t. the degrees of freedom of the system

Lagrangian Dynamics

- r_i: generalized coordinates
- T: total kinetic energy
- U: total potential energy
- Q_i : generalized forces $\quad Q_i = \sum_j F_j \frac{dr_i}{dq_j}$
- Lagrangian L = T – U

→ Lagrangian dynamic equations: $$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = Q_i$$

[Nice reference: Goldstein, Poole and Satko, “Classical Mechanics”]

Lagrangian Dynamics: Point Mass Example

Consider a point mass m with coordinates (x, y, z) close to earth and with external forces F_x, F_y, F_z.

$$T = \frac{1}{2}m(\dot{x}^2 + \dot{y}^2 + \dot{z}^2)$$

$$U = mgz$$

Lagrangian dynamic equations:

$$\begin{aligned}
F_x &= \frac{d}{dt}\frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} = m\ddot{x} \\
F_y &= \frac{d}{dt}\frac{\partial L}{\partial \dot{y}} - \frac{\partial L}{\partial y} = m\ddot{y} \\
F_z &= \frac{d}{dt}\frac{\partial L}{\partial \dot{z}} - \frac{\partial L}{\partial z} = m\ddot{z} - mg
\end{aligned}$$

Lagrangian Dynamics: Simple Double Pendulum

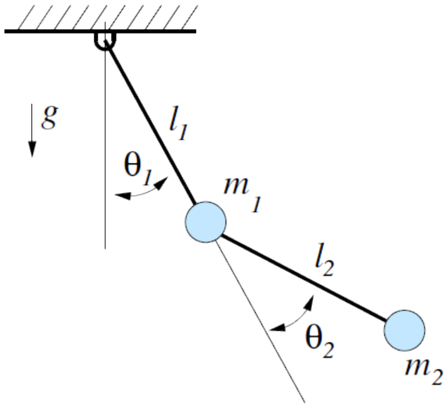


FIGURE A.1 Simple Double Pendulum

$$q_1 = \theta_1, q_2 = \theta_2, s_i = \sin\theta_i, c_i = \cos\theta_i, s_{1+2} = \sin(\theta_1 + \theta_2)$$

$$\mathbf{x}_1 = \begin{bmatrix} l_1 s_1 \\ -l_1 c_1 \end{bmatrix}, \quad \mathbf{x}_2 = \mathbf{x}_1 + \begin{bmatrix} l_2 s_{1+2} \\ -l_2 c_{1+2} \end{bmatrix}$$

$$\dot{\mathbf{x}}_1 = \begin{bmatrix} l_1 \dot{q}_1 c_1 \\ l_1 \dot{q}_1 s_1 \end{bmatrix}, \quad \dot{\mathbf{x}}_2 = \dot{\mathbf{x}}_1 + \begin{bmatrix} l_2(\dot{q}_1 + \dot{q}_2) c_{1+2} \\ l_2(\dot{q}_1 + \dot{q}_2) s_{1+2} \end{bmatrix}$$

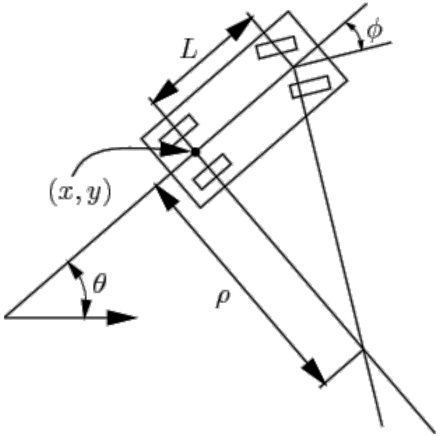
$$\begin{aligned} T =& \frac{1}{2}\dot{\mathbf{x}}_1^T m_1 \dot{\mathbf{x}}_1 + \frac{1}{2}\dot{\mathbf{x}}_2^T m_2 \dot{\mathbf{x}}_2 \\ =& \frac{1}{2}(m_1 + m_2) l_1^2 \dot{q}_1^2 + \frac{1}{2} m_2 l_2^2 (\dot{q}_1 + \dot{q}_2)^2 + m_2 l_1 l_2 \dot{q}_1 (\dot{q}_1 + \dot{q}_2) c_2 \\ U =& m_1 g y_1 + m_2 g y_2 = -(m_1 + m_2) g l_1 c_1 - m_2 g l_2 c_{1+2} \end{aligned}$$

$$\begin{aligned} (m_1 + m_2) l_1^2 \ddot{q}_1 + m_2 l_2^2 (\ddot{q}_1 + \ddot{q}_2) + m_2 l_1 l_2 (2\ddot{q}_1 + \ddot{q}_2) c_2 \\ - m_2 l_1 l_2 (2\dot{q}_1 + \dot{q}_2)\dot{q}_2 s_2 + (m_1 + m_2) l_1 g s_1 + m_2 g l_2 s_{1+2} = \tau_1 \end{aligned}$$

$$m_2 l_2^2 (\ddot{q}_1 + \ddot{q}_2) + m_2 l_1 l_2 \ddot{q}_1 c_2 + m_2 l_1 l_2 \dot{q}_1^2 s_2 + m_2 g l_2 s_{1+2} = \tau_2$$

[From: Tedrake Appendix A]

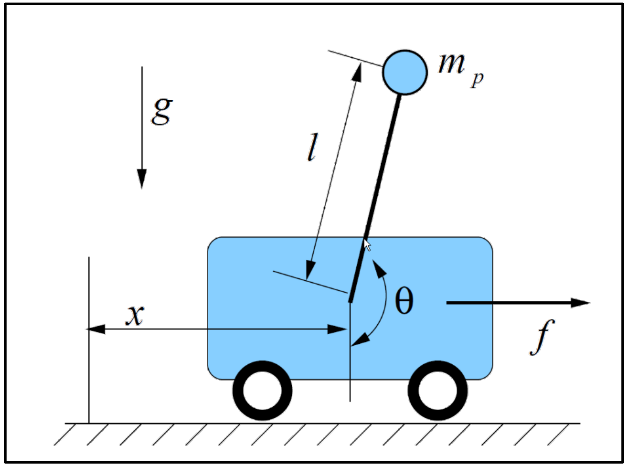
Car



$$\dot{x} = u_s \cos\theta$$
$$\dot{y} = u_s \sin\theta$$
$$\dot{\theta} = \frac{u_s}{L}\tan u_\phi.$$

- Standard (kinematic) car models: (Lavalle, Planning Algorithms, 2006, Chapter 13)
  - Tricycle: $u_S \in [-1, 1], u_\phi \in [-\pi/2, \pi/2]$
  - Simple Car: $u_S \in [-1, 1], u_\phi \in [-\phi_{\max}, \phi_{\max}], \phi_{\max} < \pi/2$
  - Reeds-Shepp Car: $u_S \in \{-1, 0, 1\}, u_\phi \in [-\phi_{\max}, \phi_{\max}], \phi_{\max} < \pi/2$
  - Dubins Car: $u_S \in \{0, 1\}, u_\phi \in [-\phi_{\max}, \phi_{\max}], \phi_{\max} < \pi/2$

Cart-pole

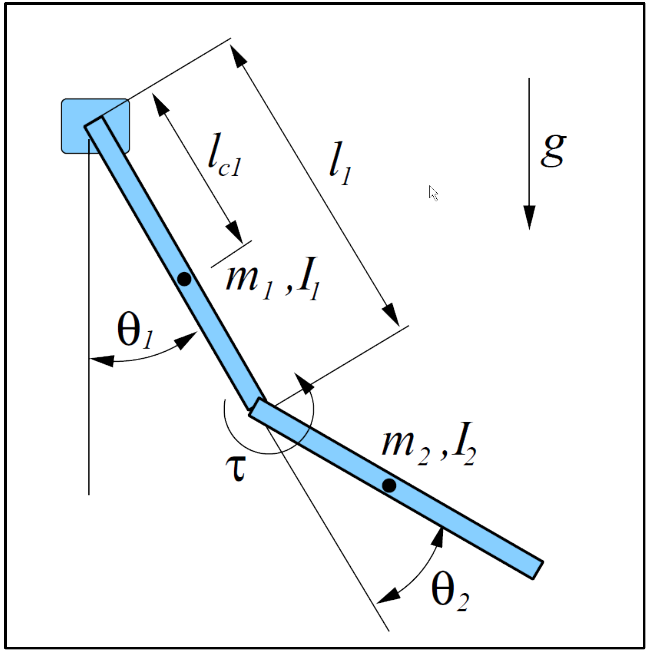


$$H(q)\ddot{q} + C(q,\dot{q}) + G(q) = B(q)u$$

$$
\begin{aligned}
H(q) &= \begin{bmatrix} m_c + m_p & m_p l \cos\theta \\ m_p l \cos\theta & m_p l^2 \end{bmatrix} \\
C(q,\dot{q}) &= \begin{bmatrix} 0 & -m_p l \dot{\theta} \sin\theta \\ 0 & 0 \end{bmatrix} \\
G(q) &= \begin{bmatrix} 0 \\ m_p g l \sin\theta \end{bmatrix} \\
B &= \begin{bmatrix} 1 \\ 0 \end{bmatrix}
\end{aligned}
$$

[See also Section 3.3 in Tedrake notes.]

Acrobot



$$H(q)\ddot{q} + C(q,\dot{q}) + G(q) = B(q)u$$

$$H(q) = \begin{bmatrix} I_1 + I_2 + m_2 l_1^2 + 2 m_2 l_1 l_{c_2} c_2 & I_2 + m_2 L_1 l_{c2} c_2 \\ I_2 + m_2 l_1 l_{c2} c_2 & I_2 \end{bmatrix}$$

$$C(q,\dot{q}) = \begin{bmatrix} -2 m_2 l_1 l_{c2} s_2 \dot{q}_2 & -m_2 l_1 l_{c2} s_2 \dot{q}_2 \\ m_2 l_1 l_{c2} s_2 \dot{q}_1 & 0 \end{bmatrix}$$

$$G(q) = \begin{bmatrix} (m_1 l_{c1} + m_2 l_1) g s_1 + m_2 g l_2 s_{1+2} \\ m_2 g l_2 s_{1+2} \end{bmatrix}$$

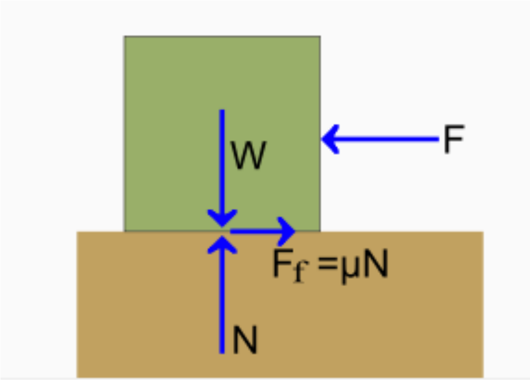
$$B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

[See also Section 3.2 in Tedrake notes.]

Friction & Drag

- Friction:
  - Static friction coefficient mu

  > Dynamic friction coefficient mu

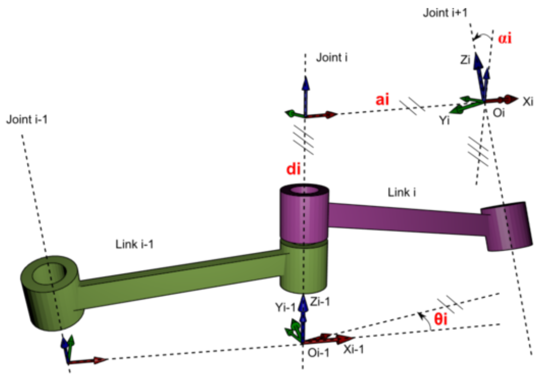


- Drag:

$$F_D = C_D A \frac{\rho V^2}{2}$$

Robot Specification?

- Denavit Hartenberg Parameterization



- In implementation: URDF Files

A lightning tour of physics simulation

- Newton’s Laws – Rigid Body Motion
- Lagrangian Formulation
- ***Continuous Time → Discrete Time***
- Contact / Collisions

Forward Euler (Explicit)

$$\dot{y} = f(t, y)$$

$$y_{n+1} = y_n + hf(t_n, y_n)$$

Backward Euler (Implicit)

$$\dot{y} = f(t, y)$$

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1})$$

Symplectic Euler (aka Semi-Implicit Euler)

$$\frac{dx}{dt} = f(t, v)$$

$$\frac{dv}{dt} = g(t, x).$$

$$v_{n+1} = v_n + g(t_n, x_n)\,\Delta t$$

$$x_{n+1} = x_n + f(t_n, v_{n+1})\,\Delta t$$

https://en.wikipedia.org/wiki/Semi-implicit_Euler_method

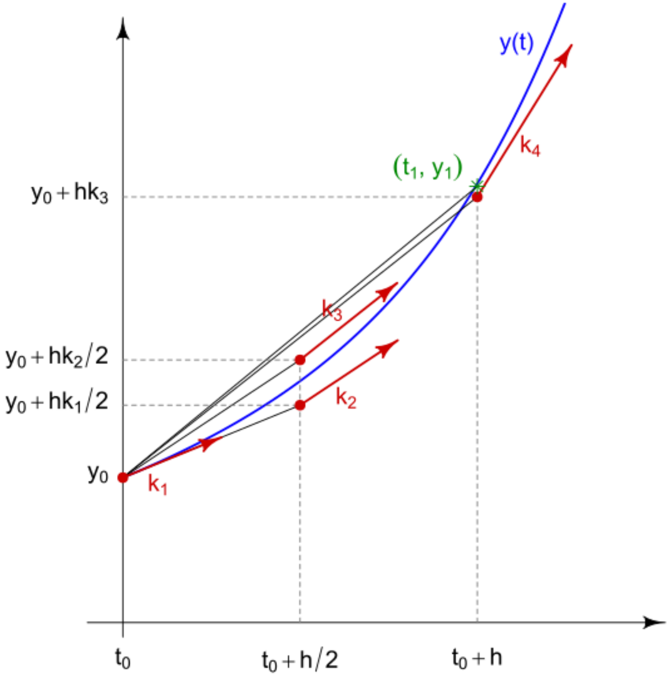
Runge-Kutta

Now pick a step-size $h > 0$ and define

$$y_{n+1} = y_n + \tfrac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right),$$
$$t_{n+1} = t_n + h$$

for $n = 0, 1, 2, 3, \ldots$, using[2]

$$k_1 = h\ f(t_n, y_n),$$
$$k_2 = h\ f\left(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right),$$
$$k_3 = h\ f\left(t_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right),$$
$$k_4 = h\ f\left(t_n + h, y_n + k_3\right).$$

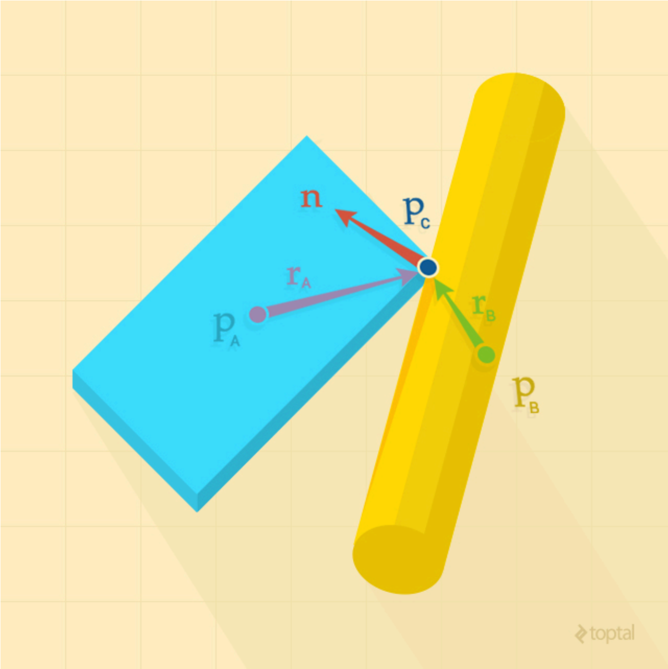


A lightning tour of physics simulation

- Newton's Laws – Rigid Body Motion
- Lagrangian Formulation
- Continuous Time → Discrete Time
- ***Contact / Collisions***

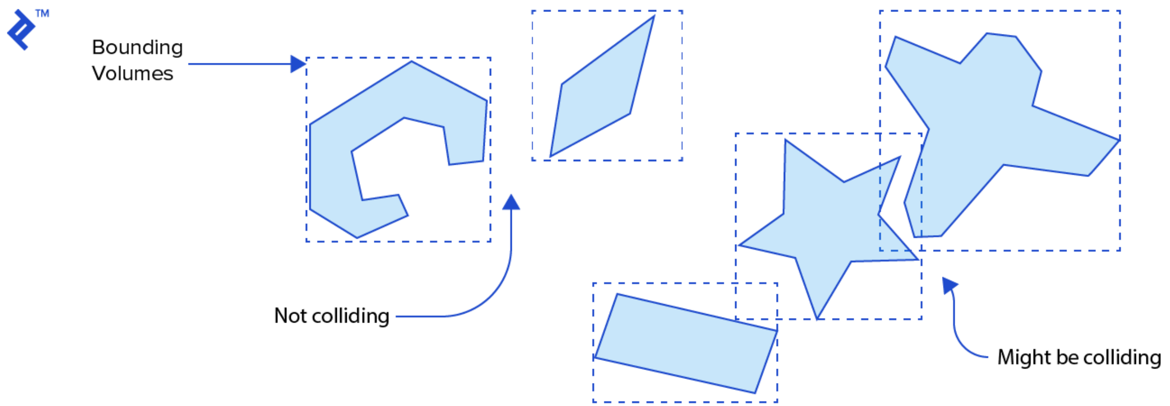
Collision Checking

- Broad phase
- Narrow phase



Broad Phase Collision Checking

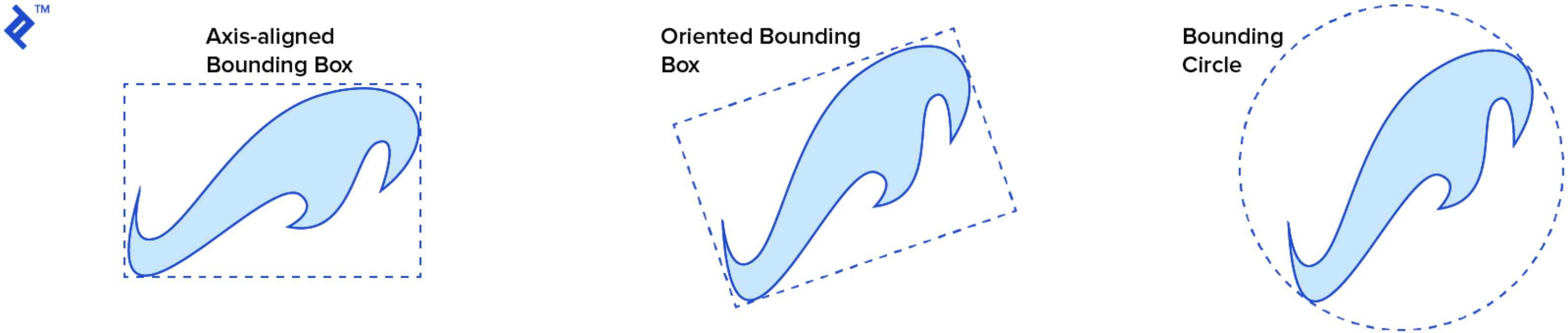
- Quadtrees/spatial
- Conservative checks



https://www.toptal.com/game/video-game-physics-part-ii-collision-detection-for-solid-objects

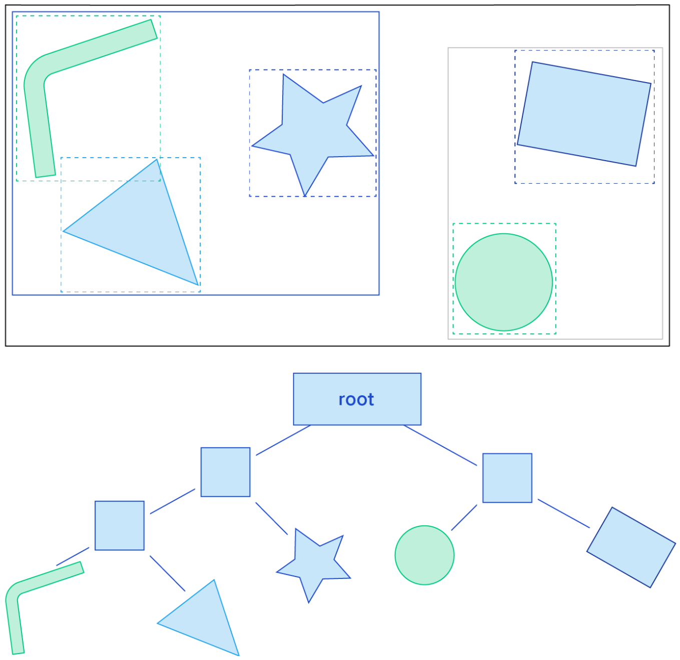
Broad Phase Collision Checking

- Quadtrees/spatial
- Conservative checks



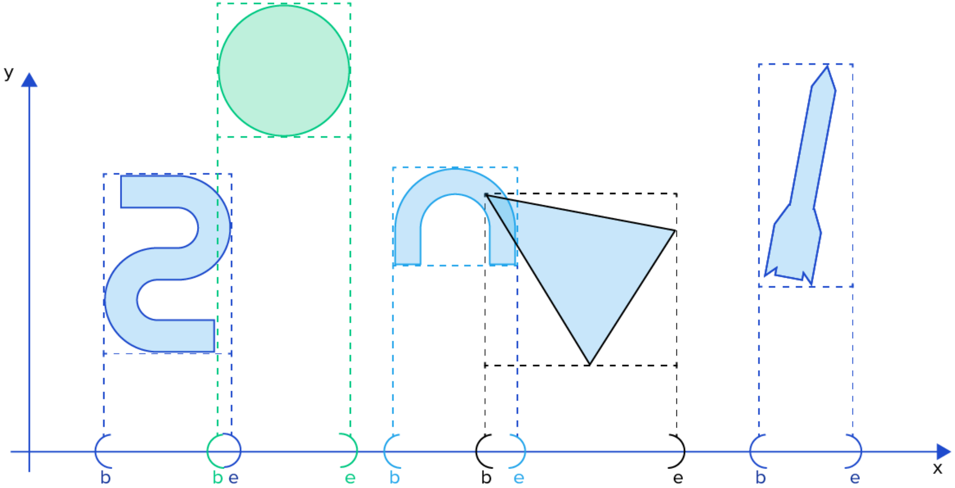
Broad Phase Collision Checking

- Quadtrees/spatial
- Conservative checks



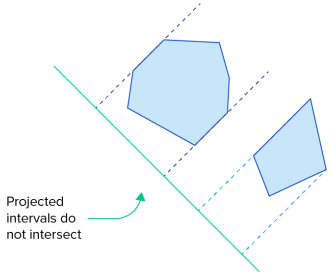
Broad Phase Collision Checking

- Quadtrees/spatial
- Conservative checks

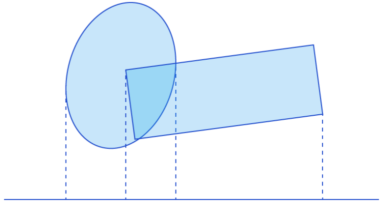


Narrow Phase Collision Checking

- Convex-Convex ---- separating axis theorem



Challenge: find a sperarating axis



Narrow Phase Collision Checking

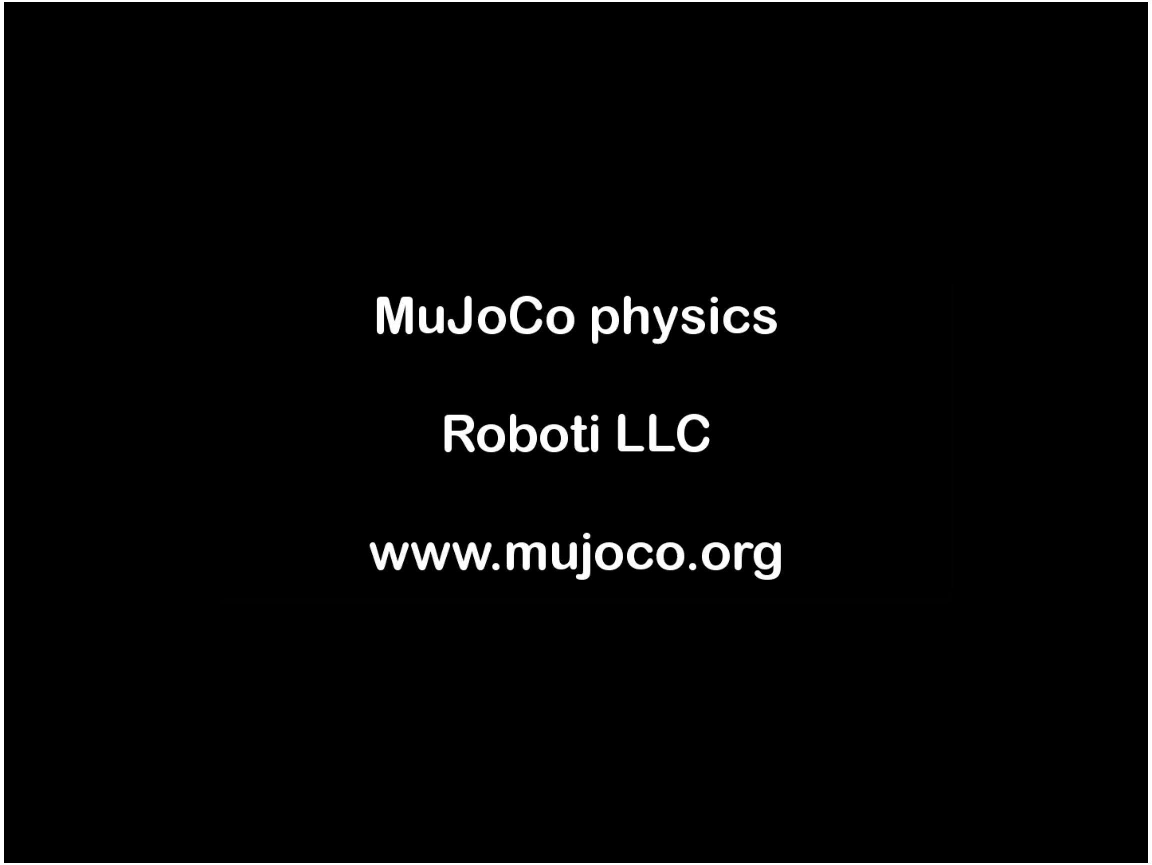
- Gilbert-Johnson-Keerthi (GJK) Algorithm
- Expanding Polytopes Algorithm (EPA)

Contact

- Impulse formulation

$$\int_t F(t)dt = m\Delta v$$

Mujoco



Bullet