COL865: Special Topics in Computer Applications

Physics-Based Animation

8 — Rigid bodies

Announcements

Assignment 1 fully posted

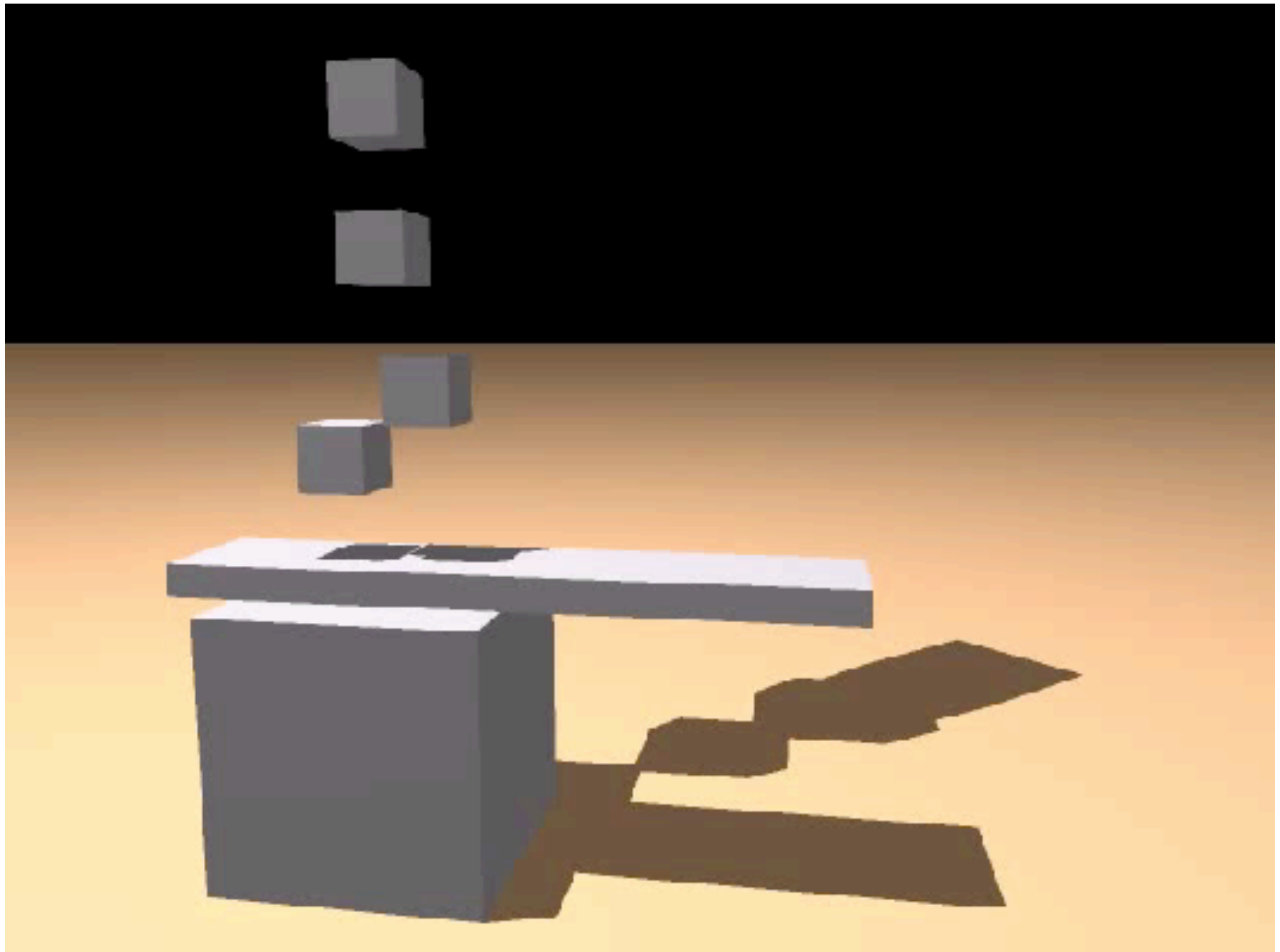
- Starter graphics code updated with smooth shading, functions for drawing spheres and boxes

I'm on travel next week

- **Monday**: Guest lecture by Prof. Prem Kalra on collision detection
  - Attendance will be taken
- **Thursday**: Paper discussions
  - Provot [1997], Weinstein et al. [2006]

Rigid bodies

Objects that don't deform at all



[Guendelman et al. 2006]

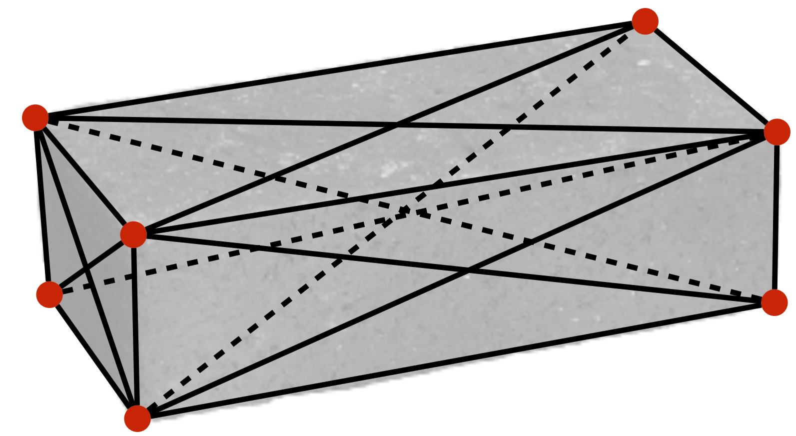
Reading

- Witkin and Baraff, *Physically Based Modeling*, Ch. “Rigid Body Simulation, Part I: Unconstrained Rigid Body Dynamics”
- **Optional:** Bender et al., “Interactive Simulation of Rigid Body Dynamics in Computer Graphics” (2012), Ch. 2, 4.1

What is a rigid body?

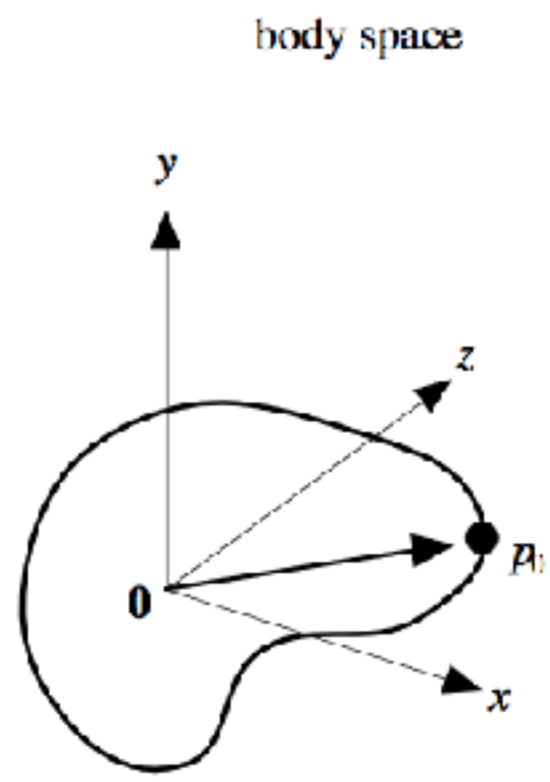
Imagine a particle system with distance constraints
$||\mathbf{x}_i - \mathbf{x}_j||$ = const between ***all*** pairs of particles

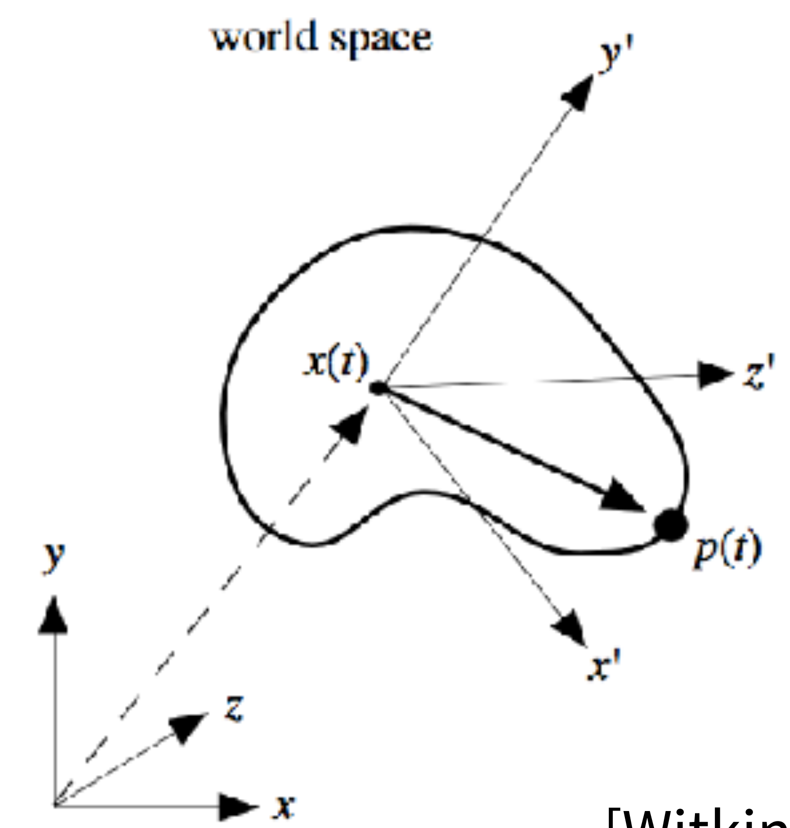
What are the possible motions?



Reduced coordinates strategy:
Store only translation **x** and rotation **R**
(relative to some reference pose)

Rigid body representation





[Witkin and Baraff]

Reference pose is arbitrary

- Usually choose origin at ***center of mass***,
  orientation along principal axes (will explain later)

How many DOFs? 3 translation + 3 rotation (why?)

Rigid body representation

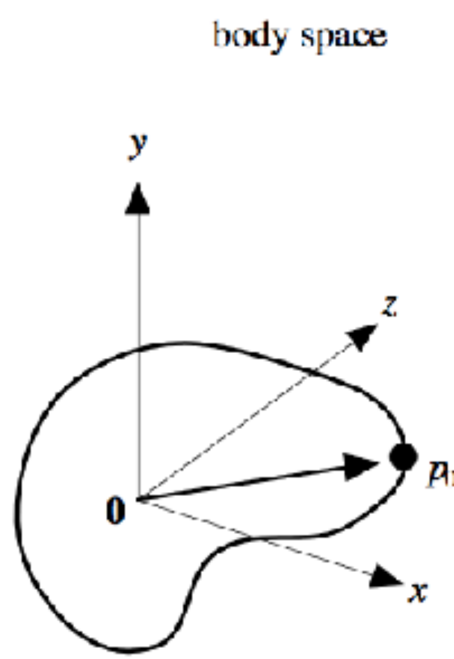
Position data: center of mass $\mathbf{x} \in \mathbb{R}^3$, rotation $\mathbf{R} \in \mathbb{R}^{3\times 3}$

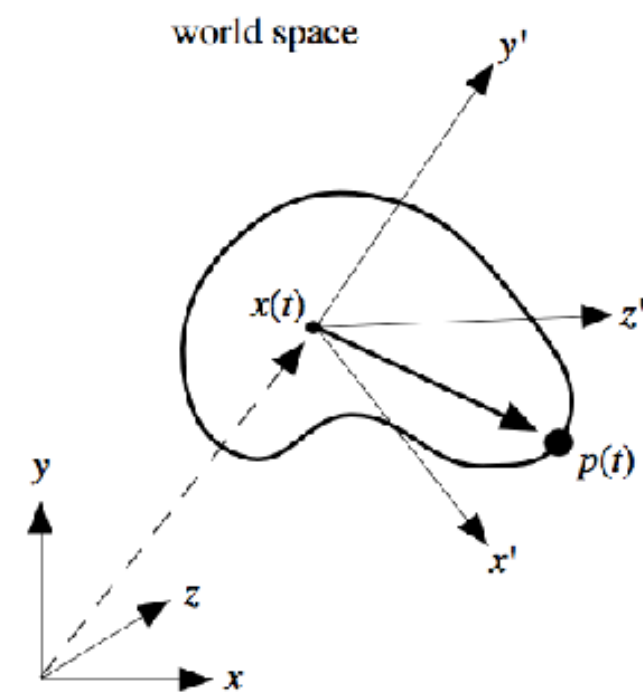
$$\mathbf{q} = (\mathbf{x}, \mathbf{R})$$

Time derivatives: $\dot{\mathbf{q}} = (\mathbf{v}, \dot{\mathbf{R}})$

For any point $\mathbf{p}_0$ in body space, world-space position & velocity is

$$\mathbf{p} = \mathbf{x} + \mathbf{R}\,\mathbf{p}_0$$
$$\dot{\mathbf{p}} = \mathbf{v} + \dot{\mathbf{R}}\,\mathbf{p}_0$$





Rigid body kinematics

$$d\mathbf{x}/dt = \mathbf{v}$$
$$d\mathbf{R}/dt = \dot{\mathbf{R}}$$

$\dot{\mathbf{R}}$ cannot be ***any*** 3×3 matrix

$\mathbf{R}\,\mathbf{R}^T = \mathbf{I}$

$\Rightarrow\ \mathbf{R}\,\dot{\mathbf{R}}^T + \dot{\mathbf{R}}\,\mathbf{R}^T = \mathbf{0}$

$\Rightarrow\ \dot{\mathbf{R}}\,\mathbf{R}^T$ is an antisymmetric matrix

$$\dot{\mathbf{R}}\mathbf{R}^T = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} = [\boldsymbol{\omega}]_\times$$

So $\dot{\mathbf{R}} = [\boldsymbol{\omega}]_\times \mathbf{R}$ for some ***angular velocity*** vector $\boldsymbol{\omega}$!

Rigid body kinematics

Position data: $\mathbf{q} = (\mathbf{x}, \mathbf{R})$
Velocity data: $\mathbf{u} = (\mathbf{v}, \boldsymbol{\omega})$

$$\dot{\mathbf{x}} = \mathbf{v}, \quad \dot{\mathbf{R}} = [\boldsymbol{\omega}]_\times \mathbf{R}$$

Velocity of point: $\dot{\mathbf{p}} = \mathbf{v} + \dot{\mathbf{R}}\, \mathbf{p}_0 = \mathbf{v} + \boldsymbol{\omega} \times (\mathbf{p} - \mathbf{x})$

Problem: After time stepping (e.g. $\mathbf{R}^{n+1} = \mathbf{R}^n + [\boldsymbol{\omega}^n]_\times \mathbf{R}^n \Delta t$),
$\mathbf{R}$ may not remain orthogonal

- Project to orthogonal matrix after every time step

Quaternions

Unit quaternions are another representation of 3D rotation

$$\boldsymbol{q} = s + i\,x + j\,y + k\,z$$
$$\text{where } s^2 + x^2 + y^2 + z^2 = 1$$

- Projection is much easier: $\boldsymbol{q}/\|\boldsymbol{q}\|$
- Velocity equation: $\dot{\boldsymbol{q}} = \tfrac{1}{2}\,\boldsymbol{\omega}\,\boldsymbol{q}$

(See Witkin & Baraff for details)

General rigid body kinematics

Position data is either $\mathbf{q} = (\mathbf{x}, \mathbf{R})$ or $\mathbf{q} = (\mathbf{x}, \boldsymbol{q})$

Velocity data is $\mathbf{u} = (\mathbf{v}, \boldsymbol{\omega})$

- Can define a matrix $\mathbf{H}$ such that $\dot{\mathbf{q}} = \mathbf{H}\,\mathbf{u}$.
  e.g. for quaternions:

$$\mathbf{H} = \begin{bmatrix} \mathbf{1}_{3\times3} & & & \\ & -\frac{1}{2}q_x & -\frac{1}{2}q_y & -\frac{1}{2}q_z \\ & \frac{1}{2}q_s & \frac{1}{2}q_z & -\frac{1}{2}q_y \\ & -\frac{1}{2}q_z & \frac{1}{2}q_s & \frac{1}{2}q_x \\ & \frac{1}{2}q_y & -\frac{1}{2}q_x & \frac{1}{2}q_s \end{bmatrix}$$