

Physically-Based Animation

We usually want realistic looking motion

- as part of our overall goal of photorealism

Our methods can in fact achieve fair degrees of realism

- key-framing
 - if our animator makes good enough key frames
- procedural/behavioral generation
 - if we write good enough scripts
- motion capture
 - by definition realistic, but we need to instrument some human

Let's consider an alternative approach to realism

- we know how objects move in the real world (as per Newton)
- given suitable physical parameters of models, we can simulate their physical motion

Dynamics

Direct physical simulation (e.g., with Newton's Laws)

- specify positions, masses, forces, ...
- apply relevant laws to compute new positions, velocities, ...
 - we can express the relevant laws as differential equations

$$F = ma \rightarrow \frac{d^2 \mathbf{x}}{dt^2} = \frac{F}{m} \text{ or } \ddot{\mathbf{x}} = \frac{F}{m}$$

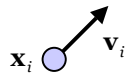
- and in general, we must solve them numerically

Keep in mind that we don't always want to mimic nature

- exact replication of reality isn't always artistically interesting
- so we can always invent our own physical laws
 - in particular, phantom forces are particularly easy to add

Particle Dynamics

Set of particles modeled as point masses in motion



\mathbf{x}_i : position of particle i
 \mathbf{v}_i : velocity vector of particle i
 m_i : mass of particle i

Can write Newton's second law as differential equation

$$\mathbf{f}_i(t) = m_i \mathbf{a}(t)$$

so

$$\ddot{\mathbf{x}}_i(t) = \frac{\mathbf{f}_i(t)}{m_i}$$

$$\text{velocity } \mathbf{v}_i = \frac{d\mathbf{x}_i}{dt} = \dot{\mathbf{x}}_i$$

$$\text{acceleration } \mathbf{a}_i = \frac{d\mathbf{v}_i}{dt} = \frac{d^2 \mathbf{x}_i}{dt^2} = \ddot{\mathbf{x}}_i$$

\mathbf{f}_i : sum of all forces acting on particle

The Force of Gravity

First, we have to arbitrarily select a "down" direction

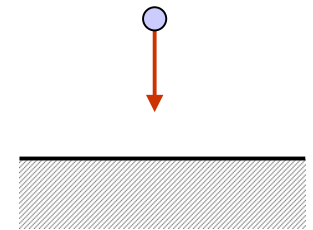
- here we'll assume that the y axis points up

Force due to gravity is simply

$$\mathbf{f}_i = \begin{pmatrix} 0 \\ -m\mathbf{g} \\ 0 \end{pmatrix}$$

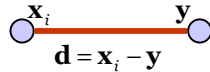
g : gravitational constant

$\approx 9.78 \text{ m/sec}^2$ on Earth



Connecting Particles with Springs

Suppose two particles are connected by a spring



- spring is characterized by
 - **rest length**: natural length of spring when unattached
 - **spring constant**: function of material; essentially “stiffness”

Force of spring acting on particle i is given by Hooke’s Law

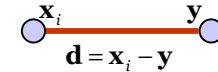
$$\mathbf{f}_i = -k_s (\|\mathbf{d}\| - s) \frac{\mathbf{d}}{\|\mathbf{d}\|} \quad k_s : \text{spring constant}$$

$$s : \text{rest length}$$

Try XSpringies [<http://www.cs.rutgers.edu/~decarlo/software.htm>]
or JSpringies [<http://www.jcraft.com/jspringies/>]

Damped Springs

No spring oscillates forever



$$\mathbf{f}_i = -k_s (\|\mathbf{d}\| - s) \frac{\mathbf{d}}{\|\mathbf{d}\|} - k_d \frac{\mathbf{d} \cdot \dot{\mathbf{d}}}{\|\mathbf{d}\|} \frac{\mathbf{d}}{\|\mathbf{d}\|} \quad \dot{\mathbf{d}} = \dot{\mathbf{x}}_i - \dot{\mathbf{y}}$$

- k_s : spring constant
- k_d : damping factor
- s : rest length

Attraction/Repulsion

Often want to simulate attractive/repulsive particles

- e.g., charged particles repel each other
- e.g., gravitational forces between planets

Inverse-square repulsion

$$\mathbf{f}_i = -k_r \frac{\mathbf{d}}{\|\mathbf{d}\|^3}$$

Gravitational Attraction

$$\mathbf{f}_i = \frac{m_1 m_2}{1/G} \frac{\mathbf{d}}{\|\mathbf{d}\|^3}$$

Numerical Solution for Particle System

We have some state vector

$$\mathbf{u}(t) = \langle \mathbf{x}_1 \quad \mathbf{v}_1 \quad \dots \quad \mathbf{x}_n \quad \mathbf{v}_n \rangle$$

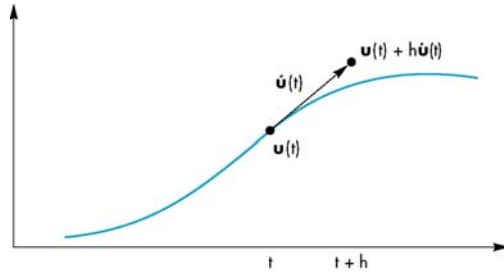
that we can differentiate

$$\begin{aligned} \dot{\mathbf{u}}(t) &= \langle \dot{\mathbf{x}}_1 \quad \dot{\mathbf{v}}_1 \quad \dots \quad \dot{\mathbf{x}}_n \quad \dot{\mathbf{v}}_n \rangle \\ &= \langle \mathbf{v}_1 \quad \mathbf{a}_1 \quad \dots \quad \mathbf{v}_n \quad \mathbf{a}_n \rangle \\ &= \left\langle \mathbf{v}_1 \quad \frac{\mathbf{f}_1}{m_1} \quad \dots \quad \mathbf{v}_n \quad \frac{\mathbf{f}_n}{m_n} \right\rangle \end{aligned}$$

Numerical Solution for Particle System

We can apply a local linear approximation

$$\mathbf{u}(t+h) \approx \mathbf{u}(t) + h\dot{\mathbf{u}}(t)$$



And we do this iteratively

- this is usually referred to as [Euler's method](#) or Euler integration

Practical Application of Dynamics

For real applications, things get significantly more complex

- most objects are not point mass particles
- need to track collisions and other events
- and we must do all this quickly and accurately

Euler's method is not numerically stable

- larger time steps rapidly lead to errors
- which accumulate because of incremental computation
- either we need to take tiny time steps,
- or use a more advanced solution method

Keep in mind that simulation is not always what we want

- we can directly simulate the movement of a golf ball
- but how do we make sure that it lands in the hole?