DYNAMICS FOR ANIMATION

Rémi Ronfard, Animation, M2R MOSIG
Summary of physics-based animation

- Motivation
- Newton’s Laws
- Point-mass models
- Rigid and articulated bodies
- Ragdoll physics
From kinematics to dynamics

- Compute positions, velocities, angles and angular velocities as a function of time
- Integration of Newton’s equations of motion

\[
\begin{bmatrix}
    p(t) \\
    v(t) \\
    q(t) \\
    \omega(t)
\end{bmatrix}
\rightarrow
\begin{bmatrix}
    p(t + \Delta t) \\
    v(t + \Delta t) \\
    q(t + \Delta t) \\
    \omega(t + \Delta t)
\end{bmatrix}
\]
Motivation: Real-time animation

- Blender Game Engine: Sensors, Controllers and Actuators
- Bullet Physics
- Ragdoll physics
Important concepts in dynamics

- **Forces**
  - contact force
  - torque
  - field force (gravity)
  - envir. force (bouyancy)

- **Center of mass and moment of inertia**
  - center of mass
  - high moment of inertia
  - low moment of inertia
Newton’s Law and Euler integration

- **Euler method**, named after Leonhard Euler, is a first-order numerical procedure for solving ordinary differential equations (ODEs) with a given initial value.

- It is the most basic kind of explicit method for numerical integration of ordinary differential equations.

- **Start at** x₀

- **Compute** acceleration from \( f = ma \) (Newton)

- **Update** velocity and position

\[
x' = x + v \cdot \Delta t \\
v' = v + a \cdot \Delta t,
\]
Midpoint and second order methods

- Compute an Euler step
  \[ \Delta x = hf(t, x) \]
- Evaluate \( y \) at the midpoint
  \[ f_{\text{mid}} = f(t + \frac{h}{2}, x + \frac{\Delta x}{2}) \]
- Use the midpoint value
  \[ x(t + h) = x(t) + hf_{\text{mid}} \]

- Newton's equation is second-order
- Reduce it to first-order using the phase space

\[
\begin{align*}
X &= (x, x)^T \\
\cdot & \quad \cdot \\
X &= (x, x)^T
\end{align*}
\]
Verlet integration

- Verlet integration was used by Carl Størmer to compute the trajectories of particles moving in a magnetic field (hence it is also called Störmer's method) and was popularized in molecular dynamics by French physicist Loup Verlet in 1967.
- It is frequently used to calculate trajectories of particles in molecular dynamics simulations and video games.
- At first it may seem natural to simply calculate trajectories using Euler integration. However, this kind of integration suffers from many problems.
- Stability of the technique depends fairly heavily upon either a uniform update rate, or the ability to accurately identify positions at a small time delta into the past.
Verlet integration

- Remember previous position $x^*$
- Current position is $x$
- Update $x$ and $x^*$
  \[ x' = 2x - x^* + a \cdot \Delta t^2 \]
  \[ x^* = x \]

- Advantage: velocity cannot go wrong!
- Applications to particle systems, mass-spring models, rigid and soft bodies
Mass spring models (1)

Properties of a spring:

- $k_s$: stiffness of the spring,
- $l_0$: initial spring length,
- $L$: current spring length.

Spring follow a linear force-deformation: Hooke’s law

$$F = k_s(L - l_0)$$ (1)
Mass-spring models (2)

A mass-spring model (MSM) is composed by a set of particles:
- Mass,
- Position,
- Speed.

And a set of springs that link them together in pairs. The springs exerts two elastic forces on each particle of the pair $i$ and $j$:

$$
F_i = -F_j = k_s \frac{x_j - x_i}{|x_j - x_i|} (|x_j - x_i| - l_0)
$$

The forces are proportional to the elongation of the spring. The value $k_s$, the rigidity of the spring, determines the behaviour of the model:
- High values $\rightarrow$ Rigid body (and also numerical instabilities!).
- Low values $\rightarrow$ Elastic body.
Mass spring models (3)

In addition to elastic forces, damping can also be added to the springs:

\[ F_i = -F_j = k_d (v_j - v_i) \frac{x_j - x_i}{|x_j - x_i|} \]  \hspace{1cm} (3)

This force is proportional to the velocity difference of the particles projected along the line of the spring.

Damping is used to:

- Simulate the internal energy loss that happens in deformable bodies.
- Avoid continuous oscillation of the springs.
- Increase the stability of the system.
Mass-spring models (4)

In a generalized spring we define a constraint \( C \):
- This constraint depends on mass positions \( C(x_1, x_2, \ldots x_n) \),
- Iff the constraint is met then \( C(x_1, x_2, \ldots x_n) = 0 \),
- We define a potential energy such as:

\[
E(x_1, x_2, \ldots x_n) = \frac{1}{2} k C(x_1, x_2, \ldots x_n)^2
\]  

(4)

- \( E = 0 \) iff the constraint is met,
- \( E > 0 \) otherwise.
Mass spring models (5)

Force at point $j$ is based on the energy $E$:

$$
F_j = -\frac{\partial}{\partial x_j} E (x_1, x_2, \ldots x_n) = kC (x_1, x_2, \ldots x_n) \frac{\partial}{\partial x_j} C (x_1, x_2, \ldots x_n)
$$

The sum of all this forces for a constraint $C$ is 0:

- Linear and angular momentum are preserved,
- Constraint forces are internal forces.
Mass spring models (6)

Preserve distance between two points:

\[ C(x_1, x_2) = |x_1, x_2| - L = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} - L \]

\[ \frac{\partial C_d}{\partial x_1} = \begin{pmatrix} \frac{\partial C_d}{\partial x_1} \\ \frac{\partial C_d}{\partial y_1} \\ \frac{\partial C_d}{\partial z_1} \end{pmatrix} = \frac{1}{|x_1 - x_2|} \begin{pmatrix} x_1 - x_2 \\ y_1 - y_2 \\ z_1 - z_2 \end{pmatrix} = \frac{x_1 - x_2}{|x_1 - x_2|} \]

And the force:

\[ F_d = k_d (|x_1 - x_2| - L) \frac{x_1 - x_2}{|x_1 - x_2|} \]

This is a spring of stiffness \( k_d \)!
Collision and contact

- **Realistic forces include collision and contact**
- **Springs at contact point**
  - Test all pairs of possibly colliding objects
  - Reduce search using space partitioning (bounding boxes, BSP-trees,...)
  - Example: particle-plane
- **Collision detection**
- **Collision backtracking**
  - Return to collision time (interpolation, binary search,...)
  - Compute new velocity
  - Numerous collisions reduce time steps!
Cartoon physics vs. Real world physics

- Any body suspended in space will remain in space until made aware of its situation.
- Any body in motion will tend to remain in motion until solid matter intervenes suddenly.
- Any body passing through solid matter will leave a perforation conforming to its perimeter.
- The time required for an object to fall twenty stories is greater than or equal to the time it takes for whoever knocked it off the ledge to spiral down twenty flights to attempt to capture it unbroken.
- Certain bodies can pass through solid walls painted to resemble tunnel entrances; others cannot.
Rigid and articulated bodies

- **Forward dynamics**
  - from joint (muscle) forces to joint trajectories (positions, velocities, accelerations)
  - Useful for animation and simulation

- **Inverse dynamics**
  - from joint trajectories (positions, velocities, accelerations) to joint (muscle) forces
  - Useful for motion capture and control
Physical animation of the human body

- Dynamics much more complex than kinematics
- 320 pairs = 640 "voluntary" skeletal muscles in the human body move the various parts of the body.
Motivation: useful motor skills

Typical virtual human motor skills include:

- Playing a stored motion sequence; this may have been synthesized by a procedure, captured from a live person, or manually scripted.
- Posture changes and balance adjustments.
- Reaching (and other arm gestures).
- Grasping (and other hand gestures).
- Locomoting (stepping, walking, running, climbing).
- Looking (and other eye and head gestures).
- Facial expressions.
- Physical force- or torque-induced movements (jumping, falling, swinging).
- Blending one movement into another.
Rigid body motion
**Rigid body motion**

\[ R(t) = [x' \ y' \ z'] \]

Figure 2: Physical interpretation of the orientation matrix \( R(t) \). At time \( t \), the columns of \( R(t) \) are the world-space directions that the body-space \( x, y, \) and \( z \) axes transform to.
Rigid body motion

Figure 3: Linear velocity $v(t)$ and angular velocity $\omega(t)$ of a rigid body.
Rigid body motion

Figure 4: The rate of change of a rotating vector. As the tip of $r(t)$ spins about the $\omega(t)$ axis, it traces out a circle of diameter $|b|$. The speed of the tip of $r(t)$ is $|\omega(t)||b|$. 
Rigid body motion

Figure 5: The velocity of the $i$th point of a rigid body in world space. The velocity of $p(t)$ can be decomposed into a linear term $v(t)$ and an angular term $\omega(t) \times (r_i(t) - x(t))$.
Rigid body motion

Figure 6: The torque $\tau_i(t)$ due to a force $F_i(t)$ acting at $r_i(t)$ on a rigid body.
Rigid body motion

Figure 10: A rectangular block acted on by a force through its center of mass.
Figure 11: A block acted on by a force, off-center of the center of mass.
Rigid body motion

Figure 12: The path the force acts over is longer than in figure 10. As a result, the force does more work, imparting a larger kinetic energy to the block.
Moments of inertia

- Kinetic energy of a solid rotating around axis \( N \) with angular velocity \( \omega \)

\[
T = \sum_{i=1}^{N} \frac{1}{2} m_i v_i^2 = \sum_{i=1}^{N} \frac{1}{2} m_i (\omega r_i)^2 = \frac{1}{2} \omega^2 \left( \sum_{i=1}^{N} m_i r_i^2 \right).
\]

- Moment of inertia

\[
I = \sum_{i=1}^{N} m_i r_i^2.
\]
Internal and external forces

- In most animation problems
  - External forces are known or easy to compute
  - Internal forces are unknown and hard to compute
- Lagrange Method
  - Equation of the constraints -> generalized forces
  - Reduced variables
    - Use constraints to reduce the number of variables
Forward dynamics: Newton-Euler

- Combine Newton and Euler equations of motion
  - Linear acceleration
  - Angular acceleration
- As a result of direct forces on the joint
- Other forces
- And inertia matrix

\[ f_i + f_i^x = I_i a_i + v_i \times I_i v_i \]
Featherstone's algorithm

- **Featherstone's algorithm** is a forward dynamic algorithm for multiple bodies.
- Useful for computing the effects of forces applied to a structure of joints and links (“armature”).
- Featherstone's algorithm uses a reduced coordinate representation.
- This is in contrast to the more popular Lagrange multiplier method, which uses maximal coordinates.
  - Brian Mirtich's PhD Thesis has a very clear and detailed description of the algorithm.
  - Baraff's paper "Linear-time dynamics using Lagrange multipliers" has a discussion and comparison of both algorithms.
Constrained Dynamics in 4 Easy Steps

1. Start out with the matrix version of Newton's \( \mathbf{f} = \mathbf{ma} \), with explicit known external forces, \( \mathbf{f}_e \), and unknown constraint forces, \( \mathbf{f}_c \).

2. Form the constraint equations for the state vector \( \mathbf{q} \), and differentiate them twice.

3. Symbolically solve for the accelerations (\( \mathbf{M} \) is trivially invertible because these are augmented coordinates) in terms of the workless constraint forces, using the Jacobian transpose and \( \lambda \) as the unknown Lagrange multipliers.

4. Form an \( \mathbf{Ax} = \mathbf{b} \) matrix equation for the unknown \( \lambda \) multipliers, solve for them, then substitute them back into (3) to find the accelerations.

Source: Chris Hecker
Inverse dynamics: articulated body algorithm

- Each body in an articulated chain has acceleration.
- Forces can be computed as
  \[ f = I^A a + p^A \]
  \[ a = \Phi f + b \]
- With
  \[ I^A = \Phi^{-1} \]
  \[ p^A = -I^A b \]
- Used to decompose and adapt motion.
The future: Physics-based 3D character animation

Real-time Physics-based 3D Biped Character Animation Using an Inverted Pendulum Model
Proportional Derivative Control

- A PD control scheme emulates the effect of a linear spring and damper acting between the current state and the desired state.
  - i.e. imagine “puppet” with a set of elastic bands pulling her to where she should be.
- Formally: \( q'' = (q_{\text{des}} - q) \cdot ks + (q'_{\text{des}} - q') \cdot kd \)
- By applying this accelerations at each time step, you will get closer and closer to your target
- As \( ks \rightarrow \infty \), it will be instantaneous
- There are a number of implementation issues associated with PD controllers
  - See SIGGRAPH course notes by David Baraff
Jacobians for velocity (left) and gravity (right)

\[ F_V \]

\[ J_{COM} = \frac{\partial P_{COM}}{\partial \theta} \]

\[ \tau_V = J_{COM}^T F_V \]

\[ J_i = \frac{\partial P_{COM(i)}}{\partial \theta} \]

\[ \tau_i = J_i^T F_i \]
Inverted pendulum model

- Compute motor forces that keep the human body in balance
- Source: Real-time Physics-Based 3D Biped Character Animation using an Inverted Pendulum Model, Tsai, 2010.
Ragdoll physics =« lifelike death »

- In computer physics engines, ragdoll physics is a type of procedural animation that is often used as a replacement for traditional static death animations.

- The term ragdoll comes from the problem that the articulated systems, due to the limits of the solvers used, tend to have little or zero joint/skeletal muscle stiffness, leading to a character collapsing much like a toy rag doll, often into comically improbable or compromising positions.
Ragdoll physics

- Dead bodies -> all physics, no muscles
- Ragdolls have been implemented using Featherstone's algorithm and spring-damper contacts (Method and system for generating realistic collisions in graphical simulation).
- An alternative approach uses constraint solvers and idealized contacts (Physically Based Modeling Principles and Practice. Proc. SIGGRAPH '97.)
Procedural animation of ragdolls

- Physics for multi-layered physical models in non-playing characters (bones / muscle / nervous systems) and deformable scenic elements from "simulated materials" in vehicles, etc. (*Medal of Honor European Assault*)

- Alternative to pre-made animation: each game is unique, whilst still deterministic. Rather than detract from gameplay through overstimulation, the "natural" qualities of movement provide for a more immersive experience, and extended replayability.
Other rag doll techniques

While the constrained-rigid-body approach to ragdolls is the most common, other "pseudo-ragdoll" techniques have been used:

- **Verlet integration** (cours 5) was used by Thomas Jakobsen in *Hitman: Codename 47*. Described in GDC 2001: Advanced Character Physics.

- Blended ragdoll was used by *Halo 2*, *Call of Duty 4*, *Left 4 Dead*, *Medal of Honor: Airborne* and *Uncharted: Drake’s Fortune*.

- It works by playing a pre-made animation, but constraining the output of that animation to what a physical system would allow.
Blended Ragdoll Example: Running

- Inverted pendulum model
- Adaptation of motion capture movements
Blended Ragdoll (continued)

- Captured motion is divided into small segments
Blended Ragdoll (continued)

- Compute inverted pendulum model per segment

Figure 5: A minimum-error point cloud matching algorithm is used for pendulum-state estimation.
Adapt segment to the chosen trajectory and context
Conclusion
Ressources

- Physically-based modeling at PIXAR!

- *Game Physics Engine Development, by Ian Millington, Morgan Kaufmann*; 2 edition (August 6, 2010)
  - [http://animationphysics.wordpress.com/](http://animationphysics.wordpress.com/)

- Chris Hecker’s notes
  - [http://chrishecker.com/Rigid_body_dynamics](http://chrishecker.com/Rigid_body_dynamics)
Highly recommended: SIGGRAPH 2001 PIXAR Course on Physically-based modeling

- Differential Equations (13 slides)
- Particle Systems (26 slides)
- Hair and Clothes (18 slides)
- Rigid Body Dynamics (28 slides)
- Constrained Dynamics (32 slides)
- Collision and Contact (39 slides)
Bullet Collision Detection & Physics Library by Erwin Coumans

- Rigid body and soft body simulation with discrete and continuous collision detection
- Collision shapes include: sphere, box, cylinder, cone, convex hull using GJK, non-convex and triangle mesh
- A rich set of rigid body and soft body constraints with constraint limits and motors
- **Soft body** support: cloth, rope and deformable objects