Introduction To Computer Animation

DIWEN XU, University of Washington, USA

These notes are primarily based on the notes by Lingqi Yan.

1 Keyframe Animation

- Lead animators create *keyframes*. Assistants or tools generate in-between (*tweening*).
- Treat each frame as a vector of parameters. Interpolate each parameter.
- Linear interpolation is often insufficient. Use splines for smooth and controllable motion.

2 Physically Based Animation

2.1 Newton's Law and Time Integration

$$\mathbf{F} = m\,\mathbf{a}, \quad \mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \Delta t\,\mathbf{v}_t + \tfrac{1}{2}(\Delta t)^2\mathbf{a}_t.$$

2.2 Mass–Spring Modeling

2.2.1 *Nonzero rest length l.*

$$\mathbf{f}_{a\to b} = k_s \frac{\mathbf{b}-\mathbf{a}}{\|\mathbf{b}-\mathbf{a}\|}\left(\|\mathbf{b}-\mathbf{a}\| - l\right).$$

2.2.2 *Damping.*

$$\mathbf{f}_a = -k_d \frac{\mathbf{b}-\mathbf{a}}{\|\mathbf{b}-\mathbf{a}\|}\left((\dot{\mathbf{b}}-\dot{\mathbf{a}})\cdot\tfrac{\mathbf{b}-\mathbf{a}}{\|\mathbf{b}-\mathbf{a}\|}\right).$$

2.2.3 *Structures from Springs.*

- Pure grids may not resist shear. Add diagonals to resist shear, but can be anisotropic.
- To resist out-of-plane bending, include out-of-plane couplings or use FEM.

2.3 Particle Systems

- Forces: gravity/electromagnetism, springs/propulsion. Damping: friction/air drag/viscosity. Collisions: walls/objects.
- Per frame: emit particles, compute forces, update positions and velocities, cull dead particles, render.

3 Kinematics

3.1 Forward Kinematics (FK)

- Articulated skeleton with joints: pin (1D rotation), ball (2D rotation), prismatic (translation).
- Given joint angles over time, compute end-effector positions.
- Pros: direct control, straightforward to implement. Cons: may violate physics, time-consuming.

3.2 Inverse Kinematics (IK)

- Given desired end-effector position, solve joint parameters.

(1) choose an initial configuration,
(2) define an error,
(3) compute gradient/Jacobian,
(4) optimize.

- Challenges: multiple solutions, disconnected solution sets, no solution for some targets.

4 Rigging

4.1 Rigging

Higher-level controls for rapid/intuitive posing and deformation. Expensive to build. Requires art+tech skill. Tailored per character.

4.2 Blend Shapes

Interpolate directly between surface shapes via linear combinations of vertex positions. Drive weights over time often with splines.

4.3 Motion Capture (MoCap)

- Data-driven capture of real performances.
- Pros: fast, realistic. Cons: costly/complex setups, edits needed to meet artistic intent.
- Modalities: optical (multiple IR cameras with retroreflective markers), magnetic (tethered), mechanical (direct joint sensing). Occlusion is a challenge for optical.
- Facial animation: beware the *uncanny valley*. Performance capture drives facial rigs.

5 Single Particle Simulation

5.1 Explicit Euler Method

$$\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \Delta t\,\dot{\mathbf{x}}_t, \quad \dot{\mathbf{x}}_{t+\Delta t} = \dot{\mathbf{x}}_t + \Delta t\,\ddot{\mathbf{x}}_t$$

Pros: simple, widely used. Cons: inaccurate, often unstable.
First-order accuracy ($O(h^2)$ per step). Consider $f(x) = -kx$.

- True solution: exponential decay to zero.
- Euler update: $x_{t+\Delta t} = (1 - k\Delta t)x_t$.
- If $\Delta t > 1/k$, oscillations occur. if $\Delta t > 2/k$, solution diverges.

5.2 Implicit Euler Method

$$\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \Delta t\,\dot{\mathbf{x}}_{t+\Delta t}, \quad \dot{\mathbf{x}}_{t+\Delta t} = \dot{\mathbf{x}}_t + \Delta t\,\ddot{\mathbf{x}}_{t+\Delta t}$$

Pros: far more stable. Cons: computationally expensive.
First-order accuracy ($O(h^2)$ per step).

5.3 Modified Euler Method

$$\dot{x}_{t+\Delta t} = \dot{x}_t + \Delta t\,\ddot{x}_t, \quad x_{t+\Delta t} = x_t + \frac{\Delta t}{2}\left(\dot{x}_t + \dot{x}_{t+\Delta t}\right)$$

Second-order accuracy ($O(h^3)$ per step).

5.4 Midpoint Method

$$\mathbf{x}_{\text{mid}} = \mathbf{x}_t + \frac{\Delta t}{2}\,\dot{\mathbf{x}}_t, \quad \mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \Delta t\,\dot{\mathbf{x}}_{\text{mid}}$$

Second-order accuracy ($O(h^3)$ per step).

5.5 Adaptive Step Size

(1) Compute x_T with one Euler step of size T.
(2) Compute $x_{T/2}$ with two half steps.
(3) Compute error $||x_T - x_{T/2}||$.

Author's Contact Information: Diwen Xu, rwbyaloupeep@gmail.com, University of Washington, Seattle, Washington, USA.

(4) If error > threshold, reduce Δt and repeat.

5.6 Position-Based / Verlet Integration

- Constrain positions of particles after each step.
- Compute velocities based on updated positions.
- Fast and stable, but dissipates energy, not physically accurate.

6 Fluid Simulation

6.1 Position-Based Fluid (PBF)

- Treat fluids as collections of small rigid-body particles.
- Enforce incompressibility via density constraints.
- Compute gradients of density w.r.t. particle positions.
- Update via gradient descent to restore target density.

6.2 Material Point Method (MPM)

Hybrid of Eulerian and Lagrangian

- Eulerian: fixed grid, track properties per cell.
- Lagrangian: track individual particles.
- Particles carry material properties, and transfer data to grid.
- Grid performs numerical updates, and interpolates data back.

Acknowledgments

To my parents and teachers, whose guidance and support have shaped who I am today. And to my beloved Sunny Sun, your companionship and encouragement enable me to go further on my journey.