# Animation & Simulation

He Wang (王鹤)

# Physically-based Animation

- Forces used to keep geometric relationships
  - We only care geometry at the end of day, not physics
  - Modelling the underlying governing laws
    - Cloth, we only *see* wrinkles, need to model forces that generate them
  - Dynamic control
    - The forces needed to achieve target goals

# Physically-based Animation

- Basic physics

$$\boldsymbol{f} = m\boldsymbol{a} \qquad \boldsymbol{a} = \frac{\boldsymbol{f}}{m}$$

$$\boldsymbol{v}' = \boldsymbol{v} + \boldsymbol{a}\Delta t \qquad \boldsymbol{p}' = \boldsymbol{p} + \frac{1}{2}(\boldsymbol{v} + \boldsymbol{v}')\Delta t$$

What scheme here?

# Physically-based Animation

- Basic physics
  - A spring
    - Virtual spring as forces between object, particles, lumped mass points

$$f_s = -k_s(L_c - L_r)\left(\frac{p_2 - p_1}{\|p_2 - p_1\|}\right)$$

spring force ($f_s$)  rest length ($L_r$)  current length ($L_c$)

constant of proportionality ($k_s$), also called the *spring constant*.

Stiffness

# Physically-based Animation

- Basic physics
  - A spring
    - Virtual spring as forces between object, particles, lumped mass points

$$f_s = -k_s(L_c - L_r)\left(\frac{p_2 - p_1}{\|p_2 - p_1\|}\right)$$ Linear

Non-linear: $k_s$ changes at certain lengths, or in general a function of length

# Physically-based Animation

- Basic physics
  - A spring
    - Virtual spring as forces between object, particles, lumped mass points
    - Damper

$$f_d = -k_d\left(\dot{p}_2 - \dot{p}_1\right) \cdot \left(\frac{p_2 - p_1}{\|p_2 - p_1\|}\right) \left(\frac{p_2 - p_1}{\|p_2 - p_1\|}\right)$$

# Physically-based Animation

- Basic physics
  - A spring
    - Virtual spring as forces between object, particles, lumped mass points
    - Damper
    - Viscosity

$$f_v = -k_v v$$

    - Momentum

$$\sum m_i \mathrm{v}_i = \mathrm{c} \qquad \tau = I\alpha$$

# Physically-based Animation

- Basic physics
  - Spring-damp pair

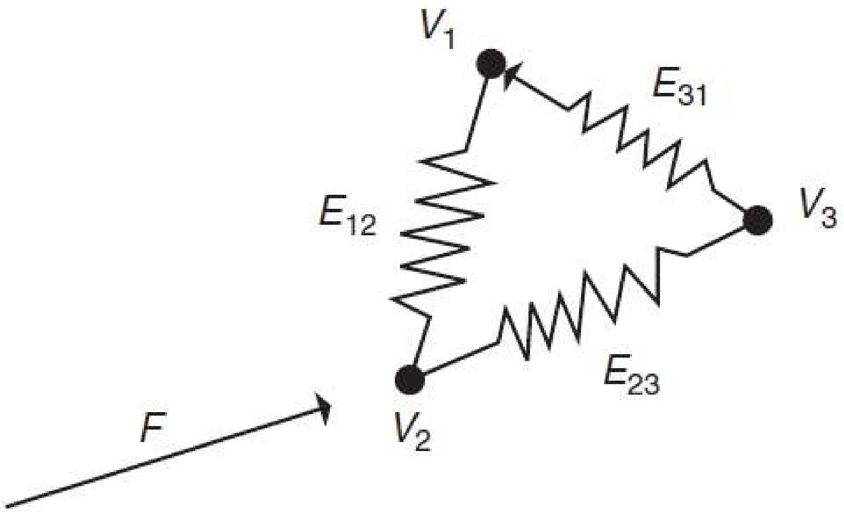
$$f_s = -k_s(L_c - L_r)\left(\frac{p_2 - p_1}{\|p_2 - p_1\|}\right) \qquad f_d = -k_d\left(\dot{p}_2 - \dot{p}_1\right)\cdot\left(\frac{p_2 - p_1}{\|p_2 - p_1\|}\right)\left(\frac{p_2 - p_1}{\|p_2 - p_1\|}\right)$$

Linear Spring damper

$$\mathrm{f} = \left(k_s(\mathrm{L}_c - \mathrm{L}_r) - k_d\left(\dot{\mathrm{p}}_2 - \dot{\mathrm{p}}_1\right)\cdot\left(\frac{\mathrm{p}_2 - \mathrm{p}_1}{\|\mathrm{p}_2 - \mathrm{p}_1\|}\right)\right)\left(\frac{\mathrm{p}_2 - \mathrm{p}_1}{\|\mathrm{p}_2 - \mathrm{p}_1\|}\right)$$

# Physically-based Animation

- Spring animation
  - Flexible objects
    - Mass-spring-damper model




$$f = \left( k_s(L_c - L_r) - k_d\left(\dot{p}_2 - \dot{p}_1\right) \cdot \left( \frac{p_2 - p_1}{\| p_2 - p_1 \|} \right) \right) \left( \frac{p_2 - p_1}{\| p_2 - p_1 \|} \right)$$

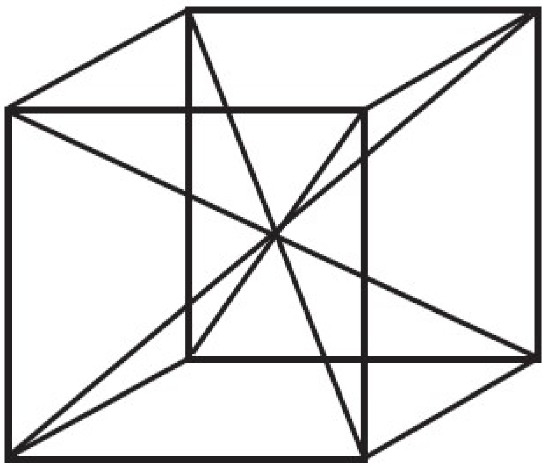
Write the force equations for three vertices
Assuming:

1. Initial velocities are zeros
2. Initial positions are (x1, y1) (x2, y2)(x3, y3)
3. Force F applied onto V2
4. Initial rest spring lengths are E11, E23 and E31
5. Masses are m1, m2 and m3
6. Ks and kd are known for all springs

Compute the positions at t + delta t

# Physically-based Animation

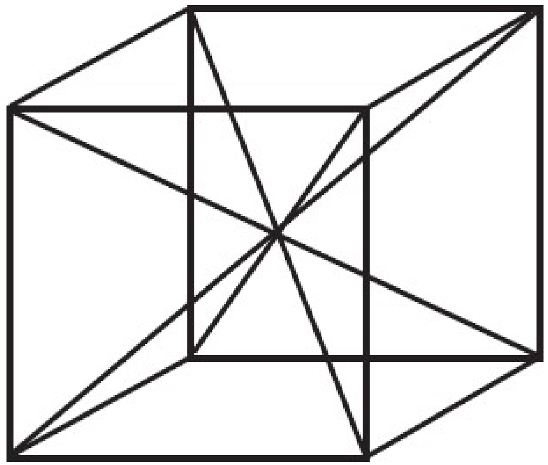
- Spring animation
  - Flexible objects
    - Mass-spring-damper model
    - Not stable if only edges are modelled: a cube could turn inside out
      - Add more springs



# Physically-based Animation

- Spring animation
  - Flexible objects
    - Mass-spring-damper model
    - Not stable if only edges are modelled: a cube could turn inside out
      - Add more springs
      - Add angular springs to maintain right angles

$$\hat{\tau} = k_s(\theta(t) - \theta_r) - k_d\,\dot{\theta}(t)$$



# Physically-based Animation

- Spring animation
  - Flexible objects
    - Mass-spring-damper model
    - Not stable if only edges are modelled: a cube could turn inside out
      - Add more springs
      - Add angular springs to maintain right angles
  - Too many parameters to tune Ks, Kd, Kv for each spring
    - Hard to tune
    - Numerically explode
    - Why? Forces are assumed constant during the time step
    - Clipping values might help, but introducing slaggishness

# Physically-based Animation

- Spring animation
  - Virtual Springs
    - Proportional derivative control (PD)

$$u(t) = K_{\mathrm{p}} e(t) + K_{\mathrm{d}} \frac{de(t)}{dt}$$

$$\tau = k_s(\theta(t) - \theta_d(t)) - k_d\big(\dot{\theta}(t) - \dot{\theta}_d(t)\big)$$

    - Proportional integral control (PI)

$$u(t) = K_{\mathrm{p}} e(t) + K_{\mathrm{i}} \int_0^t e(t')\, dt'$$

    - Proportional integral derivative control (PID)

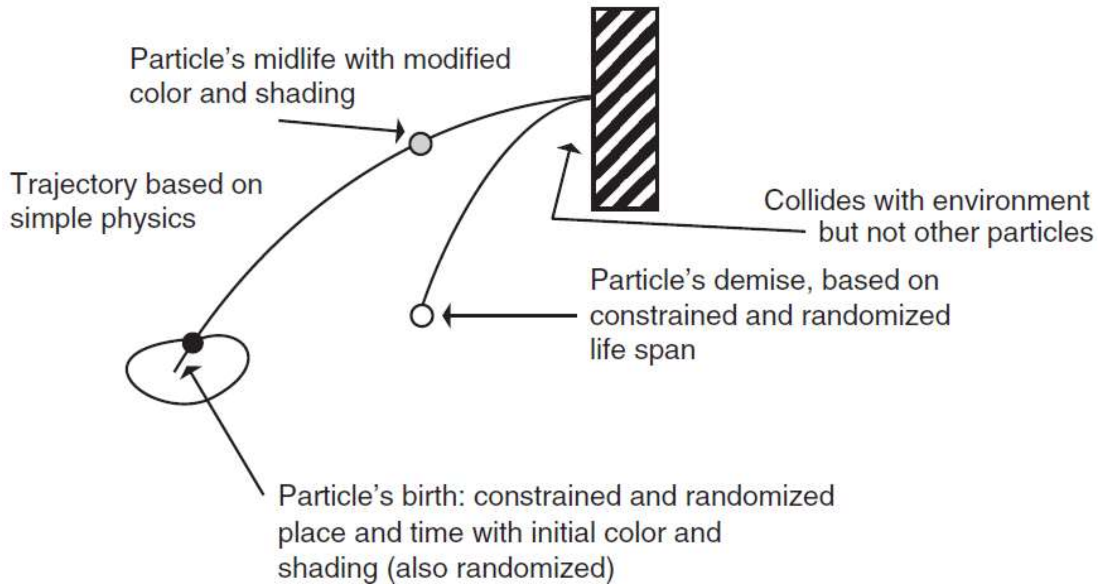
$$u(t) = K_{\mathrm{p}} e(t) + K_{\mathrm{i}} \int_0^t e(t')\, dt' + K_{\mathrm{d}} \frac{de(t)}{dt}$$

# Physically-based Animation

- Particle Systems
  - A large number of small particles, common simplifications:
    - Volume of individual particles are (largely) ignored
    - Masses are assumed to be lumped at points
    - Not colliding with other particles
    - Not casting shadows
    - Not reflecting lights

# Physically-based Animation

- Particle Systems
  - A large number of small particles, common simplifications:
  - Life spans
    - Particles born in this time step
    - Attributes assigned
    - Terminating dead particles
    - Animating remaining particles
    - Render particles

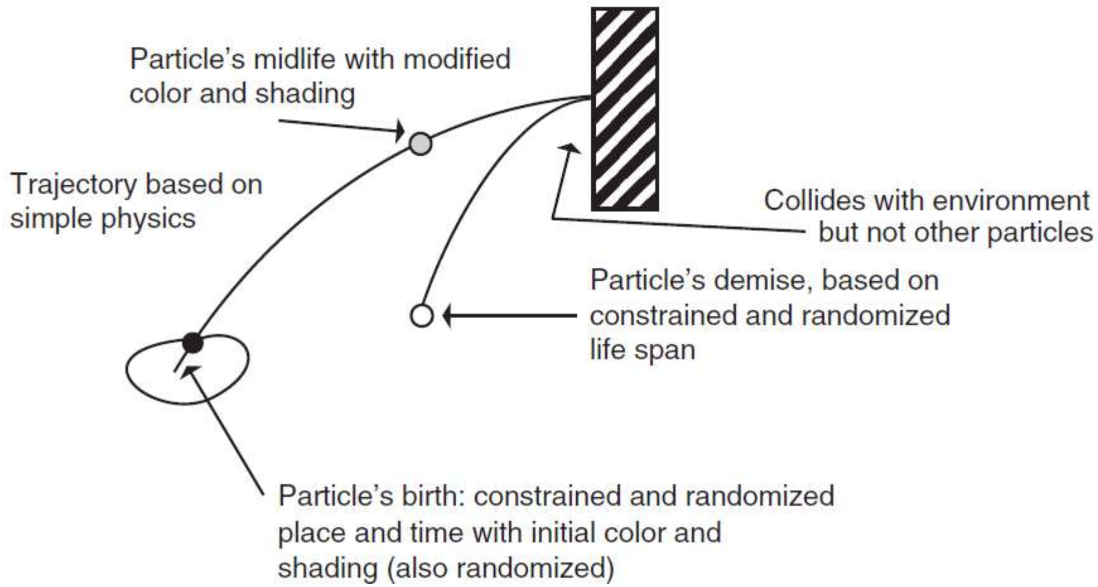


# Physically-based Animation

- Particle Systems
  - A large number of small particles, common simplifications:
  - Life spans
    - Particles born in this time step

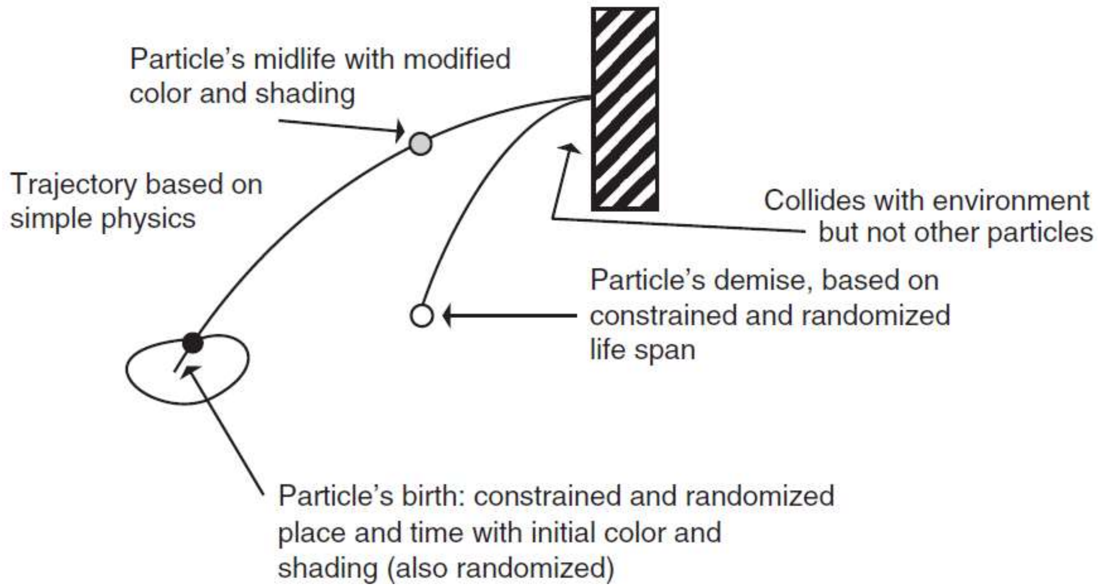
$$\# \; of \; particles = n + Rand() * r$$

$$\# \; of \; particles = n(A) + Rand() * r$$



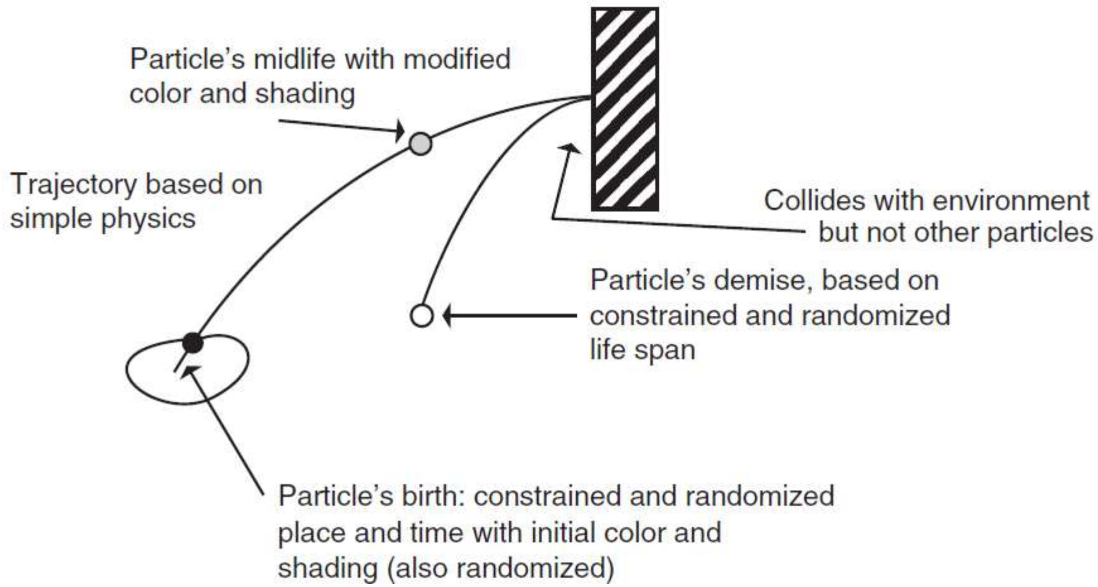
# Physically-based Animation

- Particle Systems
  - A large number of small particles, common simplifications:
  - Life spans
    - Particles born in this time step
    - Attributes assigned
      - Pos, vel, shape,
        color, transparency, lifetime



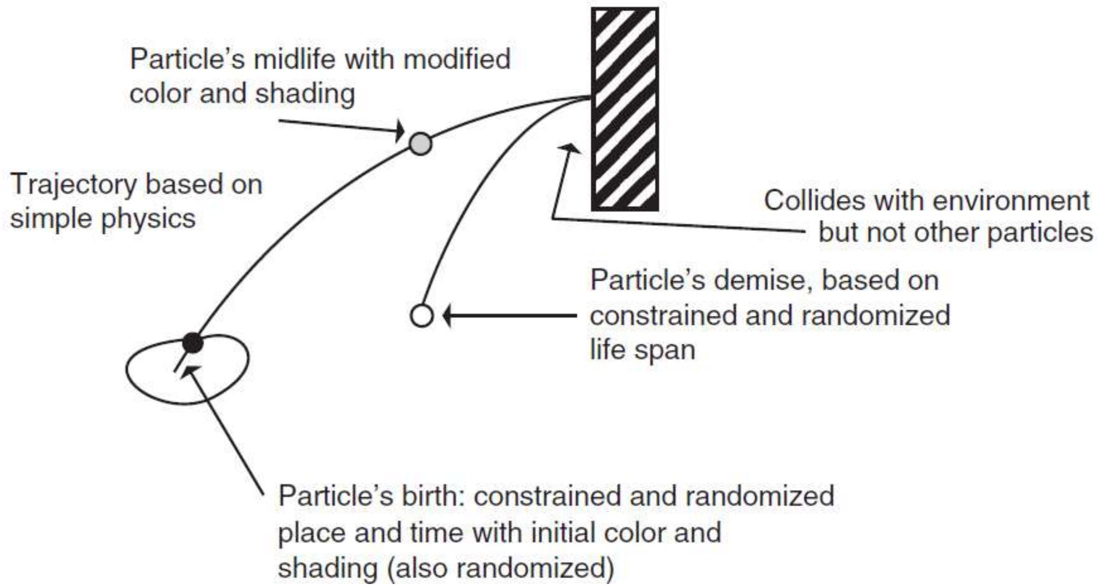
# Physically-based Animation

- Particle Systems
  - A large number of small particles, common simplifications:
  - Life spans
    - Particles born in this time step
    - Attributes assigned
    - Terminating dead particles
      - Check particle clocks, remove dead ones



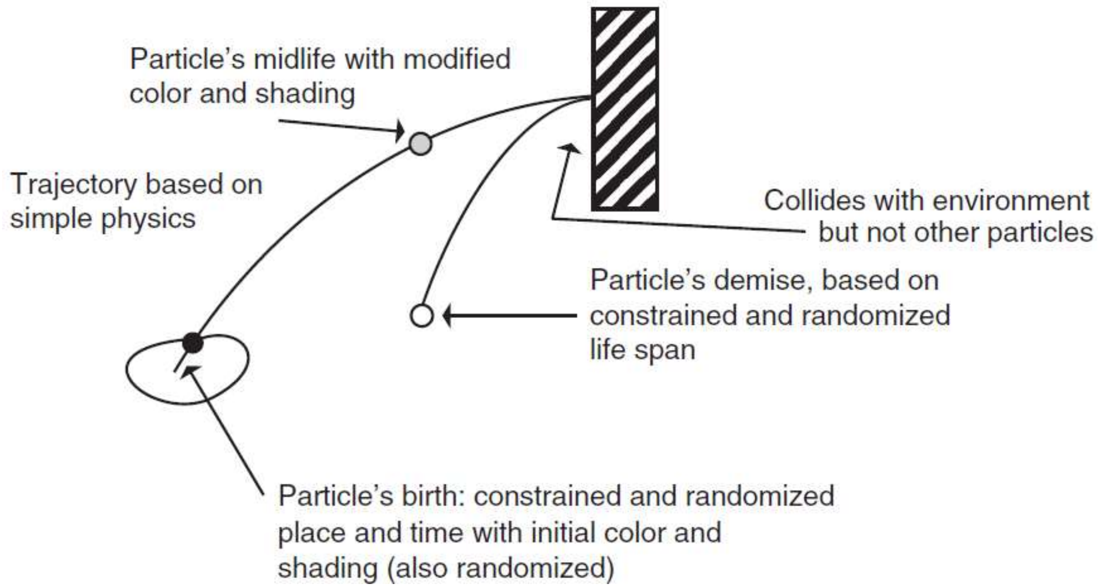
# Physically-based Animation

- Particle Systems
  - A large number of small particles, common simplifications:
  - Life spans
    - Particles born in this time step
    - Attributes assigned
    - Terminating dead particles
    - Animating remaining particles
      - Simulation
        (gravity, wind, force field, etc.)



# Physically-based Animation

- Particle Systems
  - A large number of small particles, common simplifications:
  - Life spans
    - Particles born in this time step
    - Attributes assigned
    - Terminating dead particles
    - Animating remaining particles
    - Render particles
      - Render as light sources



# Physically-based Animation

- Rigid body simulation
  - Cloth (listed under ‘rigid body simulation’?)
    - Direct modelling of folds, pure geometry
      - Only works for very special situations
    - Physically based modelling
      - Stretch, bend, skew
      - Springs or energy functions

# Physically-based Animation

- Rigid body simulation
  - Cloth
    - Physically based modelling
      - Stretch

Rest length Current length

$$F_s = \left( \frac{k_s |v1 - v2| - |v1^* - v2^*|}{|v1^* - v2^*|} \right) \frac{v1 - v2}{|v1 - v2|}$$

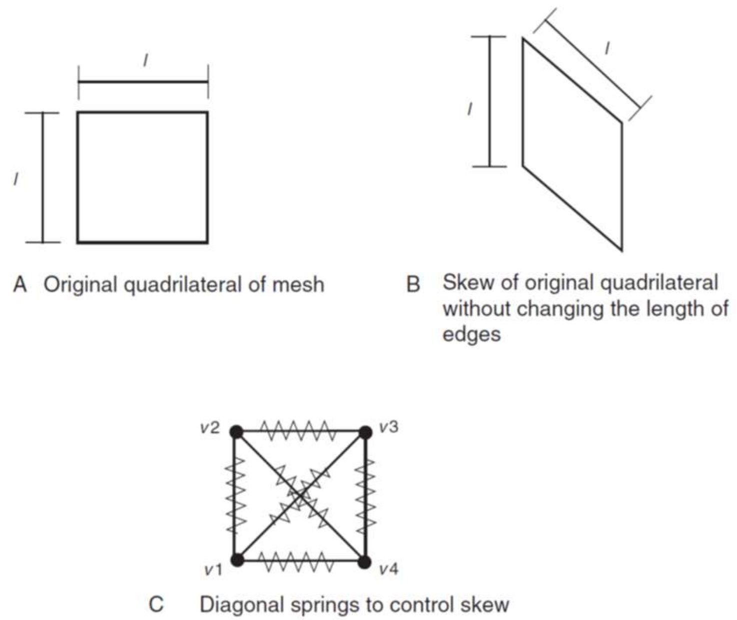
$$E_s = k_s \frac{1}{2} \left( \frac{|v1 - v2| - |v1^* - v2^*|}{|v1^* - v2^*|} \right)^2$$

# Physically-based Animation

- Rigid body simulation
  - Cloth
    - Physically based modelling
      - Stretch
      - In-plane skew

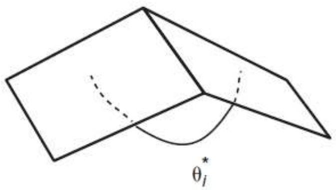
$$S(v1\,,\,v2) = \left(\frac{1}{2}\right)\left(\frac{|v1 - v2| - |v1^* - v2^*|}{|v1^* - v2^*|}\right)^2$$

$$E_{\mathrm{w}} = k_{\mathrm{w}} \cdot S(v1, v3)\; S(v2, v4)$$

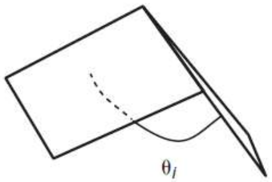


# Physically-based Animation

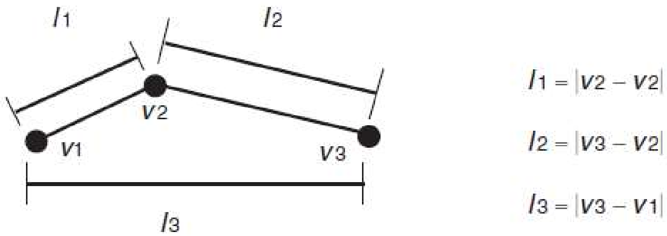
- Rigid body simulation
  - Cloth
    - Physically based modelling
      - Stretch
      - In-plane skew
      - Out-plane skew (restricting dihedral angle or control vertices separation)




Original dihedral angle




Bending along the edge that
changes dihedral angle




$$F_b = k_b(\theta_i - \theta_i{}^*)$$

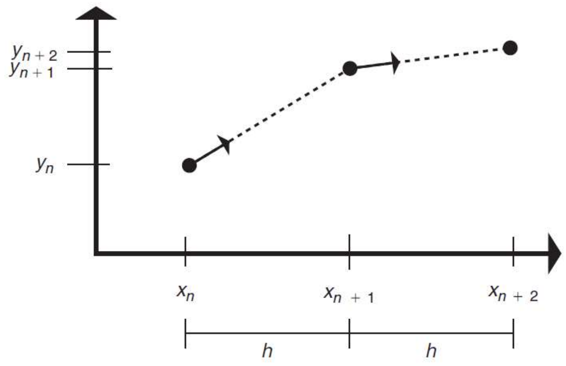
$$F_{\mathrm{b}} = k_{\mathrm{b}}\left(\frac{l3}{(l1 + l2)} - \frac{l3^*}{(l1^* + l2^*)}\right)$$

# Physically-based Animation

- Rigid body simulation
  - Cloth
    - Physically based modelling
      - Mass-spring-damper model, compute forces and update states of mass points
      - Integration

$$y_{n+1} = y_n + hf^{'}(x_n, y_n)$$

explicit Euler (fast but ?)

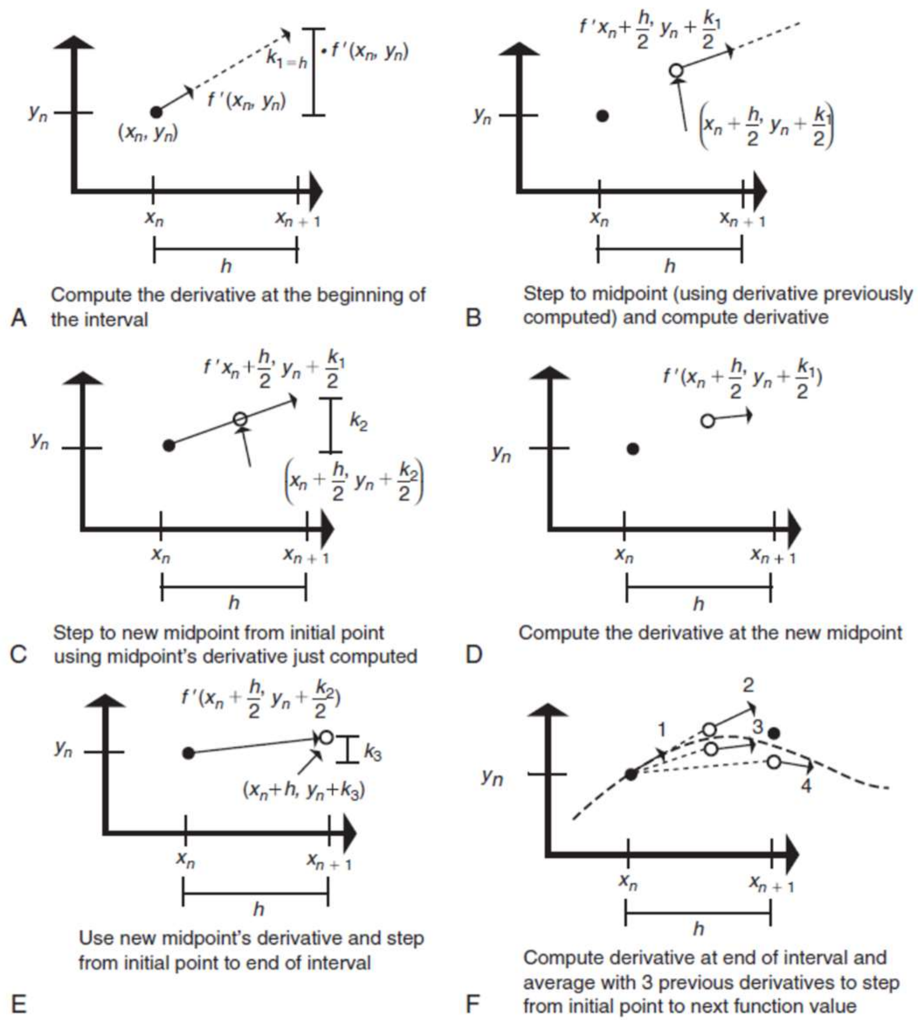


# Physically-based Animatio

- Rigid body simulation
  - Cloth
    - Physically based modelling
      - Mass-spring-damper model, compute forc
      - Integration

Fourth-order Runge-Kutta

$$k_1 = hf'(x_n, y_n)$$
$$k_2 = hf'\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right)$$
$$k_3 = hf'\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right)$$
$$k_4 = hf'(x_n + h, y_n + k_3)$$
$$y_{n+1} = y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5)$$



# Physically-based Animation

- Rigid body simulation
  - Cloth
    - Physically based modelling
      - Mass-spring-damper model, compute forces and update states of mass points
      - Integration

Implicit Euler

$$\boxed{y_{n+1}} = y_n + hf'(x_{n+1}, \boxed{y_{n+1}})$$

Unknowns on both sides, need to solve an equation for every step

[Baraff et al, Large Steps in Cloth Simulation. 1998]

# Physically-based Animation

- Rigid body simulation
  - Cloth
    - Physically based modelling
      - Mass-spring-damper model, compute forces and update states of mass points
      - Integration

Semi-implicit Euler

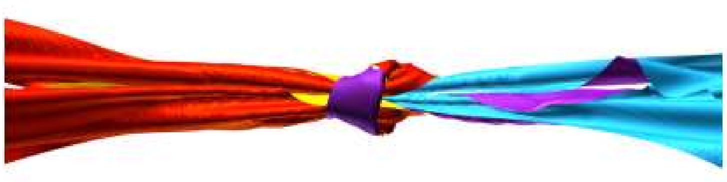
$$y_{n+1} = y_n + hf^{'}\left(x_{n+1}, y_n + hf^{'}\left(x_n, y_n\right)\right)$$

# Physically-based Animation

- Rigid body simulation
  - Cloth
    - Physically based modelling
      - Mass-spring-damper model, compute forces and update states of mass points
      - Integration
      - Control stretching
        - Spring model can lead to unrealistic stretching:super-elasticity
        - Stiffer springs, smaller time steps, limiting initial stretching, modelling non-linear effects
          - Biphasic springs (allow initial stretching but becomes stiff exceeding threshold)
          - Velocity damping (controlling vertex displacement)

# Physically-based Animation

- Rigid body simulation
  - Cloth
    - Physically based modelling
      - Mass-spring-damper model, compute forces and update states of mass points
      - Integration
      - Control stretching
      - Collision detection (self and environment)
        - Hierarchical bounding volumes to accelerate the detection



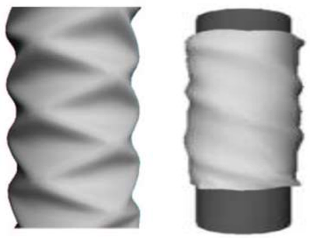
Harmon et al, Asynchronous Contact Mechanics, 2011

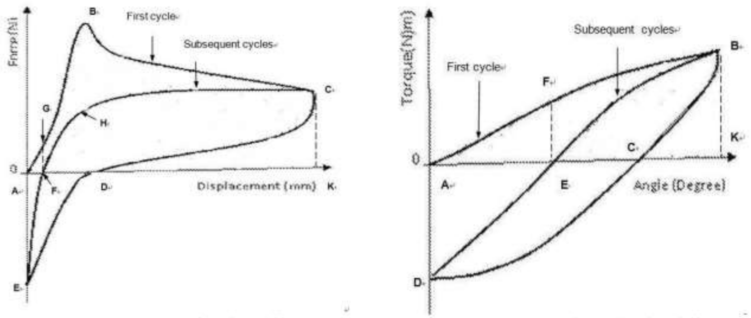
# Physically-based Animation

- Rigid body simulation
  - Cloth
    - Physically based modelling
      - Mass-spring-damper model, compute forces and update states of mass points
      - Integration
      - Control stretching
      - Collision detection (self and environment)
      - Collision response
        - Damped inelastic collision
        - Restraining the positions/velocities of the colliding vertices
        - Open research question

# Physically-based Animation

- Rigid body simulation
  - Cloth
    - Physically based modelling
      - Mass-spring-damper model, compute forces and update states of mass points
      - Integration
      - Control stretching
      - Collision detection (self and environment)
      - Collision response
      - Folds, wrinkles, buckling
        - More difficult, still under research






(a) Compression buckling (b) twist buckling