Physically based simulation - Models
When physically based simulation is needed

- Accurate dynamics
- Tedious to model by hand or procedurally
  - Multiple interacting elements: ex. Multiple collisions: rigid bodies, hairs, etc.
  - Complex animated geometry: Cloths, fluids
General methodology

1. Description of the system
   *Describe system by some parameters (positions, speed, orientation, etc).*
   - State of the system is known at time $t = 0$ - *Initial value problem in time*
   - State of the system may be constrained in space - *Boundary value problem in space*

2. Evolution
   *Link the evolution of the system to forces or constraints using dynamic principles and conservation laws*
   \[ \Rightarrow \text{Differential equation} \]

3. Numerical Solution
   *Solve the differential equation using numerical iterative approaches.*

   **Note:** Fundamentally different that direct approach controlling the trajectories at key-frames
   - The system is set at an initial step
   - We let the numerical solution build the space-time trajectory for us
   (+) Allows to model complex behavior
   (-) Lack of control on the result
Fundamental models

1- **Particles**
2- Rigid bodies
3- Continuum models
Physically-based particle system

1. Description
   Particle is fully described by: Position $p$, Velocity $v$, Mass $m$
   
   Fundamental quantities: position and linear momentum $P = m \, v$
   Linear Momentum preserved in isolated system

2. Evolution
   - Fundamental principle of dynamics
     Force applied on particle $F(p, v, t)$
     
     \[
     \begin{aligned}
     p'(t) &= v(t) \\
     P'(t) &= m \, v'(t) = F(p, v, t)
     \end{aligned}
     \]
   - Conservation of energy (ex. kinetic energy $(1/2 \, m \, v^2)$+potential energy = const, etc.)
   - Lagragian, or Hamiltonian (reduced coordinates)
Physically-based particle system

3. Numerical Solution

ODE (Ordinary Differential Equation) formulated as an Initial Value Problem

\[
\begin{align*}
  p'(t) &= v(t) \\
  m v'(t) &= F(p, v, t)
\end{align*}
\]

ex. \( \begin{align*}
  p'(t) &= v(t) \\
  m v'(t) &= F(p, v, t) \\
  v(0) &= v_0, \ p(0) = p_0
\end{align*} \)

- Discretize in time \( t^k = k \cdot h \), \( h = \Delta t = \) time step.
  
  \[ \Rightarrow \] Build a discrete numerical solution \( p^k = p(t^k), v^k = v(t^k) \).

- We can consider initially the following iterative scheme

\[
\begin{align*}
  v^{k+1} &= v^k + h \ F(p^k, v^k, t^k) \\
  p^{k+1} &= p^k + h \ v^{k+1}
\end{align*}
\]

Simple to implement, reasonably OK for simple examples (more details later).
Physically-based particle system

**Pro**

(-) Simple to implement, and control.
(-) Efficient to compute, scalable
(-) Highly adaptable from simple particle to rigid and deformable models

**Cons**

(-) Limited accuracy - highly simplified model from physical point of view.

⇒ Dominant model in CG production for general purpose deformable model animation.

*Common use: Lots of sparsely interacting particles*
Rigid spheres
System modeling

Particles modeling the center of hard spheres.
- Spheres can collide with surrounding obstacles
- Spheres can collide with each others

- **System**: N particles with position $p_i$, velocity $v_i$, mass $m_i$, modeling a sphere of radius $r_i$.
  - Initial conditions $p_i(0) = p^0_i$, $v_i(0) = v^0_i$

- **Forces**: Single gravity forces $F_i = m_i \ g$. Collisions handled by *impulses*.

- **Temporal evolution**: Fundamental principle of dynamics $v_i(t) = p'_i(t)$, $v'_i(t) = g$

- **Numerical solution**

\[
\begin{align*}
  v^{k+1} &= v^k + h \ g \\
  p^{k+1} &= p^k + h \ v^{k+1}
\end{align*}
\]
Collision with a plane

Plane \( \mathcal{P} \): parameterized using a point \( a \) and its normal \( n \).
\[
\{ p \in \mathbb{R}^3 \in \mathcal{P} \Rightarrow (p - a) \cdot n = 0 \}
\]
- Sphere above plane: \((p_i - a) \cdot n > r_i\)
- Sphere in collision: \((p_i - a) \cdot n \leq r_i\)

- Collision detection algorithm

```c
for(int i=0; i<N; ++i)
{
    float detection = dot(p[i]-a, n);
    if (detection <= r[i])
    {
        // ... collision response
    }
}
```

What should we do when a collision is detected
Collision response with plane

Suppose exact contact: \((p_i - a) \cdot n = r_i\)
Collision response = **Update velocity**

\[
\text{Split } v = v_{\parallel} + v_{\perp} \\
- v_{\perp} = (v \cdot n) n \\
- v_{\parallel} = v - (v \cdot n)n
\]

**New velocity**

\[
v^{\text{new}} = \alpha v_{\parallel} - \beta v_{\perp}
\]

- \(\alpha \in [0, 1]\) Restitution coefficient in \(\parallel\) direction (friction)
- \(\beta \in [0, 1]\) Restitution coefficient in \(\perp\) direction (impact)
Result: Collision response

Applying collision response on speed only
Result: Collision response - issue with discrete time

We assumed contact b/w sphere and plane
But: Exact contact never happens in discrete time

- When collision is detected → already inside the wall
- Weight is still acting

doesn't bounce high enough
Collision response with plane: position

In real case (discrete time) no exact contact, but penetration \((p_i - a) \cdot n_i < r_i\)
⇒ Need to compute collision response at contact point.

Three possibilities
(1) Correct position in projecting on the constraint
   (+) Simple to implement
   (-) Physically incorrect position
(2) Approximate the correct position
(3) Go backward in time to find exact instant of collision

Continuous Collision Detection
(+) Physically correct
(-) Computationally heavy (binary search, etc.)
Result: Projecting position on plane

\[ p_i^{\text{new}} = p_i + d \mathbf{n} \]
\[ d = r_i - (p_i - a) \cdot n_i : \text{distance of penetration} \]
Collision between spheres

Given 2 spheres \((p_1, v_1, r_1, m_1), (p_2, v_2, r_2, m_2)\).
Collision when \(\|p_1 - p_2\| \leq r_1 + r_2\)

What happen with their velocities?
\[ v_1 \rightarrow v_{1\text{ new}}, v_2 \rightarrow v_{2\text{ new}} \]
Notion of impulse

An impulse $J$ is the integrated force over time $J = \int_{t_1}^{t_2} F(t) \, dt$

→ results in a sudden change of speed (/momentum) in a discrete case

For a particle with constant mass

$$\int_{t_1}^{t_2} F(t) \, dt = \int_{t_1}^{t_2} m a(t) \, dt$$

$$\Rightarrow J = m (v(t_2) - v(t_1))$$

For an impact $v \rightarrow v^{\text{new}}$

$$v^{\text{new}} = v + J / m$$
Two spheres in collision

Impulse orthogonal to the separating plane between the two surfaces

\[ J = j \, u, \quad u = (p_1 - p_2)/\|p_1 - p_2\| \]

The system with the two spheres is preserving its linear momentum

\[ \Rightarrow \text{Respective impulses } j \text{ are equals in magnitude, and opposed in direction} \]

\[ m_1 v_1 + m_2 v_2 = m_1 v_1^{\text{new}} + m_2 v_2^{\text{new}} \Rightarrow m_1 (v_1^{\text{new}} - v_1) = -m_2 (v_2^{\text{new}} - v_2) \Rightarrow J_1 = -J_2 \]

Assume collision of "hard spheres" = "Elastic collision"

\[ \Rightarrow j = 2 \frac{m_1 \, m_2}{m_1 + m_2} \left( v_2 - v_1 \right) \cdot u \]

\[ \Rightarrow 0 = 2 j \, v_1 \cdot u + \frac{j^2}{m_1} - 2 j \, v_2 \cdot u + \frac{j^2}{m_2} \]

\[ \Rightarrow j = \frac{2}{1/m_1 + 1/m_2} (v_2 - v_1) \cdot u \]
Two spheres in collision

\[ v_{1\text{new}} = v_1 + \frac{j}{m_1} u = v_1 + 2 \frac{m_2}{m_1 + m_2} ((v_2 - v_1) \cdot u) u \]
\[ v_{2\text{new}} = v_2 - \frac{j}{m_2} u = v_2 - 2 \frac{m_1}{m_1 + m_2} ((v_2 - v_1) \cdot u) u \]

Rem. If \( m_1 = m_2 \): Switch their \( \perp \) speeds
\[ v_{1\text{new}} = v_1 + ((v_2 - v_1) \cdot u) u = v_{1\parallel} + v_{2\perp} \]
\[ v_{2\text{new}} = v_2 - ((v_2 - v_1) \cdot u) u = v_{2\parallel} + v_{1\perp} \]

Can use restitution coefficient and attenuation \((\alpha, \beta) \in [0, 1]\)
\[ v_{1\text{new}} = \alpha v_{1\parallel} + \beta v_{2\perp} \]
\[ v_{2\text{new}} = \alpha v_{2\parallel} + \beta v_{1\perp} \]
Summary

1. Detect collision \( \|p_1 - p_2\| \leq r_1 + r_2 \)

2-a. If collision (relative speed > \( \epsilon \))
   Elastic collision (/bouncing) \( v_{1/2} = \alpha v_{1/2} \pm \beta J / m_{1/2} \)

2-b. If static contact (relative speed \( \leq \epsilon \))
   Friction \( v_{1/2} = \mu v_{1/2} \), \( \mu \in [0, 1] \)
   Avoids jittering

3. Correct position (project on contact surface)
   \( p = p \pm d/2u \)
   \( d = r_1 + r_2 - \|p_1 - p_2\|: \) Collision depth
Multiple collisions

Pairwise collisions $\Rightarrow$ no global collision free state
- Correcting one collision may induce new collisions.
- Order of correction does matter

Reducing time-step help, Iterating over multiple pass help

But correct solution in all cases is complex $\Rightarrow$ global approach
- Precompute contact graph
  - explicit shock propagation management
- Global constraint-based method
  - Impulse: $n_i \cdot (v_i - v_j) \geq 0$
  - Momentum preservation: $m_i v_i - m_j v_j = 0$
  - Energy preservation/dissipation
  $\Rightarrow$ Linear Complementarity Program, Gauss Seidel, etc.

[ Realistic Animation of Rigid Bodies. J. Hahn. SIGGRAPH 1988. ]
[ Reflections on Simultaneous Impact. B. Smith et al. SIGGRAPH 2012 ]
Elastic models

Spring structure
Numerical solution of ODE
Cloth simulation
**Material model**

**Elasticity**: Shape goes back toward its original rest position when external forces are removed.
- Purely elastic models don't lose energy when deformed (potential ↔ kinetic)

**Plasticity**: Opposite of elasticity. Plastic material don't come back to their original shape (/change their rest position during deformation).
- Ductile material - can allow large amount of plastic deformation without breaking (plastic)
- Brittle - Opposite (glass, ceramics)

**Viscosity**: Resistance to flow (usually for fluid, ex. honey)

In reality
- *Elasto-plastic materials*: Allow elastic behavior for small deformation, and plastic at larger one.
- *Visco-elastic materials*: Elastic properties with delay.
Modeling elastic shapes with particles

Spring mass systems
- Particles (position, velocity, mass): samples on shape
- Springs: link closed-by particles in the reference shape

1D curve structure
2D surface structure
3D volume structure
Spring structure

How to model spring connectivity?

- **Structural springs**: 1-ring neighbors springs ($\sim$ mesh edges)
  
  (+) Limit elongation/contraction, (-) Allows shearing, and bending

  $\Rightarrow$ Add extra springs connectivity

- **Shearing springs**: Diagonal links

- **Bending springs**: 2-ring neighborhood
Example: 1D spring (/Harmonic oscillator)

- Force $F(t) = -k (p(t) - l^0)$, $k$ spring stiffness, $l^0$ rest length
- Second order differential equation: $m p''(t) + k p(t) = kl^0$

- First order system
  $$\begin{pmatrix} p' \\ v' \end{pmatrix}(t) = \begin{pmatrix} v(t) \\ -k/m (p(t) - l^0) \end{pmatrix}$$

- Linear system
  $$\begin{pmatrix} p' \\ v' \end{pmatrix}(t) = \begin{pmatrix} 0 & 1 \\ -k/m & 0 \end{pmatrix} \begin{pmatrix} p \\ v \end{pmatrix}(t) + \begin{pmatrix} 0 \\ k/m l^0 \end{pmatrix}$$

- Exact solution known: $p(t) = A \sin(\omega t + \phi) + l^0$
  $$\omega = \sqrt{k/m}, \quad A^2 = (p^0 - l^0)^2 + \left(\frac{v_0}{\omega}\right)^2, \quad \tan(\phi) = \frac{p^0 - l^0}{v_0/\omega}$$
Example: 3D mass spring

- Force $F(t) = m \mathbf{g} - k \left( \| \mathbf{p}(t) \| - l^0 \right) \frac{\mathbf{p}(t)}{\| \mathbf{p}(t) \|}$, $k$ spring stiffness, $l^0$ rest length
- Second order differential equation: $m \mathbf{p}''(t) = m \mathbf{g} - k \left( \| \mathbf{p}(t) \| - l^0 \right) \frac{\mathbf{p}(t)}{\| \mathbf{p}(t) \|}$
- First order system

$$\begin{pmatrix} p' \\ v' \\ u'(t) \end{pmatrix} = \begin{pmatrix} \mathbf{v}(t) \\ g - k/m (\| \mathbf{p}(t) \| - l^0) \mathbf{p}(t) \\ \mathcal{F}(u,t) \end{pmatrix}$$

- Not linear
- No simple explicit solution
Numerical integration of ODE

General formulation: \( u'(t) = \mathcal{F}(u, t), u(t) = (p(t), v(t)) \).

**Explicit Euler**

\[ u^{k+1} = u^k + \Delta t \mathcal{F}(u^k, t^k) \]

(+) Easy to implement
(-) Worst scheme in all cases (divergence, low accuracy)

**Explicit Runge-Kutta**

\[ u^{k+1} = u^k + \Delta t \sum_j \alpha_j k_j \]

(+ ) Good accuracy
(+ ) Efficient to apply
(+/- ) Stability OK for non-stiff problem, diverge on stiff problem
(-) Artificial damping for constant energy system

**Implicit methods**

\[ u^{k+1} = u^k + \Delta t \mathcal{F}(u^{k+1}, t^{k+1}) \]

(+ ) Good to deal with **stiff problem** - very stable
(-) Add numerical damping (converge even if solution oscillates)
(-) Hard/computationally costly to apply on non-linear problem

**Symplectic integrator**

\[ v^{k+1} = v^k + \Delta t \frac{F^k}{m} \]
\[ p^{k+1} = p^k + \Delta t v^{k+1} \]

(+ ) Handle well constant energy system, preserves energy (Hamiltonian systems)
(+ ) Simple and efficient to implement
(-) Less accurate than RK
(-) Diverge on stiff problem
Cloth Simulation
Mass-spring cloth simulation

- Particles are sampled on a $N \times N$ grid.
  - Each particle has a mass $m$ ($m_{\text{cloth}} = N^2 \, m$)
- Set structural, shearing and bending springs.
Forces

- On each particle: gravity + drag + spring forces

\[ F_i(p, v, t) = m_i g - \mu v_i(t) + \sum_{j \in V_i} K_{ij} \left( \|p_j(t) - p_i(t)\| - L_{ij}^0 \right) \frac{p_j(t) - p_i(t)}{\|p_j(t) - p_i(t)\|} \]

- \( V_i \): neighborhood of particle \( i \)
- \( L_{ij}^0 \): rest length of spring \( ij \)

Associated ODE \( \forall i, \left\{ \begin{array}{l} p_i'(t) = v_i(t) \\ v_i'(t) = F_i(p, v, t)/m_i \end{array} \right. \)

Q. How can we model the effect of the wind?
Note on Mass-Spring numerical solution

- Non-linear ODE

- Large $K_{ij}$: good length preservation, but stiff ODE
  $\Rightarrow$ divergence of explicit schemes.

- Avoid explicit Euler (divergence)

- Semi-implicit Euler/Verlet works fine for low $K_{ij}$
  
  Semi-implicit Euler + PBD allows simple integration + stable stiff springs
  [Muller et al. PBD, Inextensible clothing in Computer Games]

- RK4 more accurate (but higher complexity than Verlet)

- Implicit Euler: requires linearization, but very stable
Collisions

- Simple approach: Handled as collision between particles and shapes
  (+) Simple and efficient
  (-) Collision may still appear within a triangle
  ⇒ Exhaustive approach: edges + faces
**Limitation of mass spring model and continuous model**

- Does mass-spring system converge toward a unique solution when sampling increase?
  
  ⇒ No :(

  Depends on the connectivity → bad for physical accuracy

**Corollary**

- Mass-springs work well for grid-mesh structure (draping)
- Less for arbitrary triangular meshes

1st improvement: Change toward energy formulation for bending springs (limits locking effect)

\[
F = \frac{\partial E}{\partial p}
\]

\[
E = \frac{1}{2} K L \kappa^2, \kappa: \text{curvature}
\]

[Cho et al, Stable but Responsive Cloth, ACM SIGGRAPH 2002]
Triangle as continuous elements

- Defining Bending Energy between triangles

\[ W_B(x) = \sum_{edges \ e} (\theta_e - \theta^0_e) \left\| e^0 \right\| \frac{1}{h^0_e} \]

[E. Grinspun et al., Discrete Shells, SCA 2003]
(or expressed using forces in [R. Bridson et al., SCA 2003])

- Going toward full FEM numerical resolution

- B. Thomaszewski et al. [SCA 2006], [VRIPHYS 2008], [EG 2009].
Clothing

- Stitch 2D patterns together to generate full cloth
- Cloths are developable material (preserve length w/r their 2D patterns)

[Thalman et al. 2002]  [Umetani et al., 2011]
Detecting self collision

Handled as moving point in collision with moving triangle

**Inputs**
- Triangle $P_1(t)P_2(t)P_3(t)$, a point $P(t)$
- Each position $P_k(t) = P_k(0) + tv_{P_k}$

**Computing intersection**

**Necessary condition**
- Find $t_i \in [0, h]$ such that $P(t_i)$ is in triangle plane
  
  \[
  (P(t_i) - P_1(t_i)) \times n(t_i) = 0
  \]

  $n(t_i)$: normal of the triangle at time $t_i$

**Sufficient condition**
- Check $P(t_i)$ is inside the triangle

  \[
  P(t_i) = \alpha P_1(t_i) + \beta P_2(t_i) + \gamma P_3(t_i)
  \]

  \[
  (\alpha, \beta, \gamma) \in [0, 1]^3, \alpha + \beta + \gamma = 1
  \]


[R. Bridson et al. Robust Treatment of Collisions, Contact and Friction for Cloth Animation. ACM SIGGRAPH 2002]