Physics Simulation

Mengyu Chu 楚梦渝

https://rachelcmy.github.io/
mchu@pku.edu.cn
http://vcl.pku.edu.cn/index.html

VISUAL COMPUTING AND LEARNING LAB

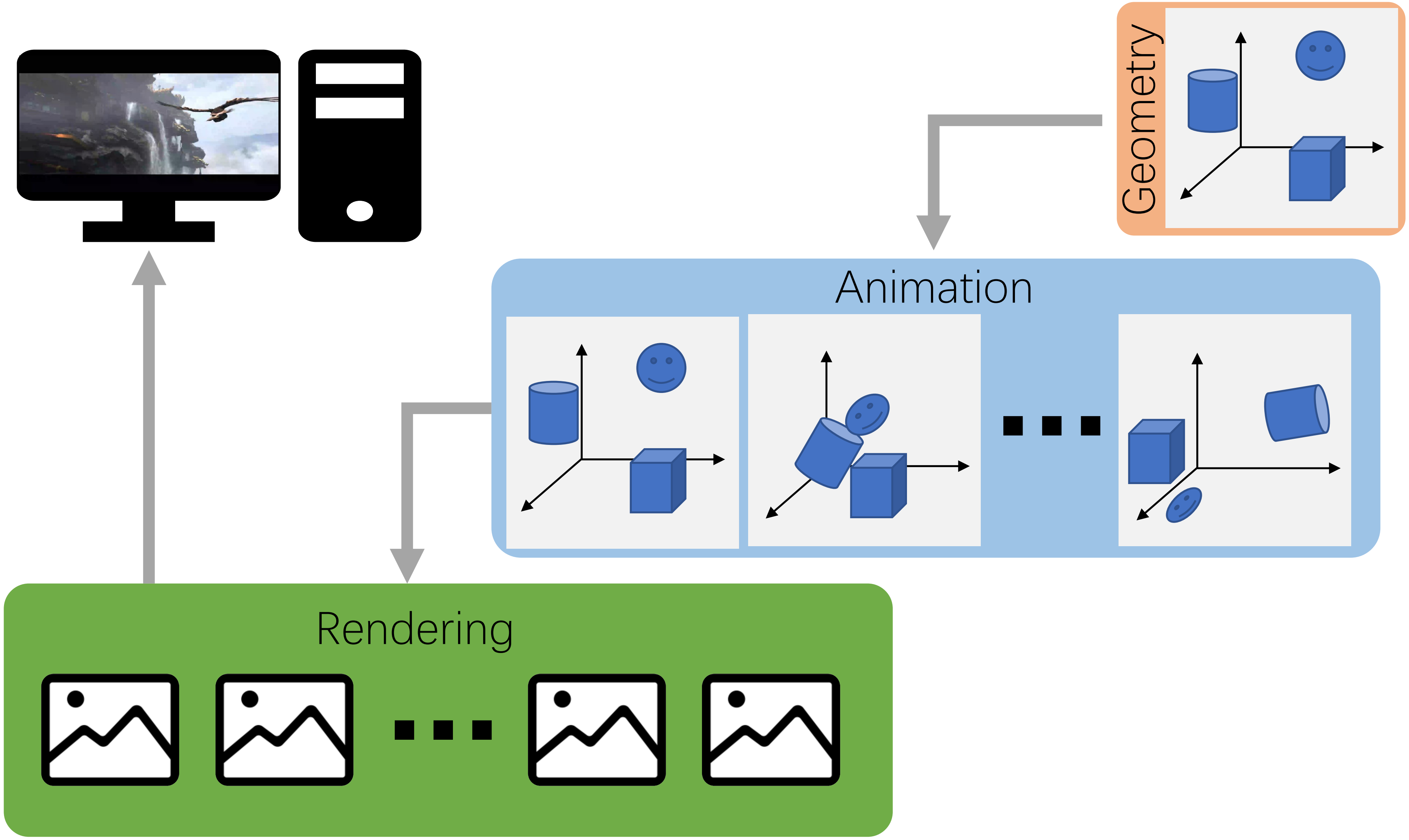
Computer Graphics

Modeling

Simulation,
Animation

Rendering

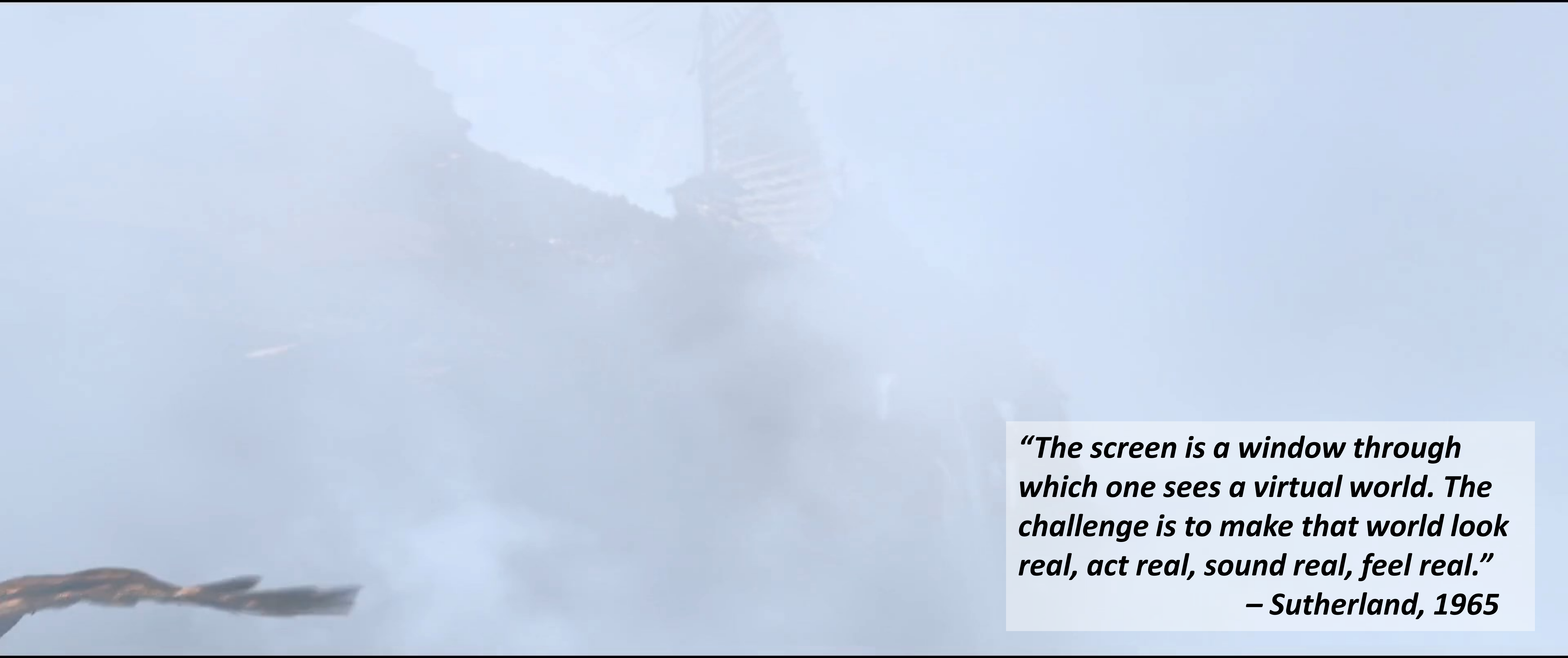
Computer Graphics



Modeling

Simulation, Animation

Rendering



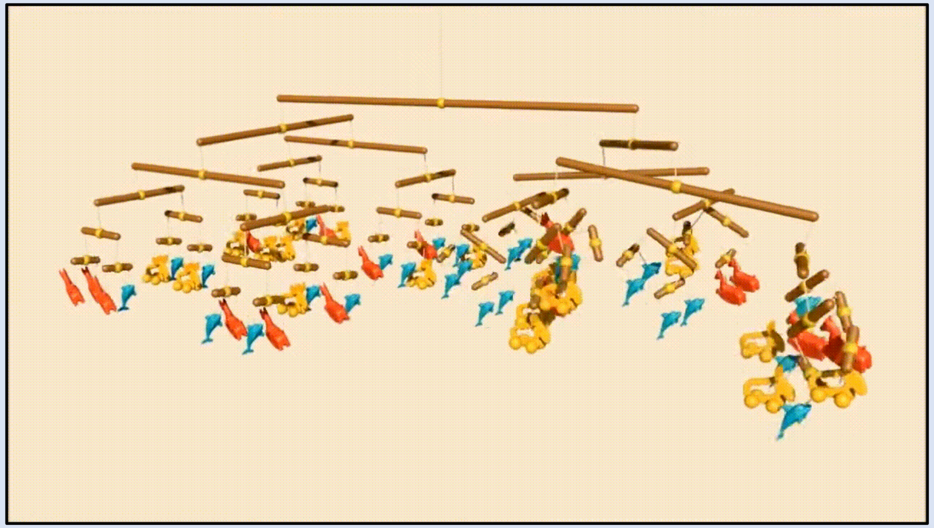
"The screen is a window through which one sees a virtual world. The challenge is to make that world look real, act real, sound real, feel real."

– Sutherland, 1965

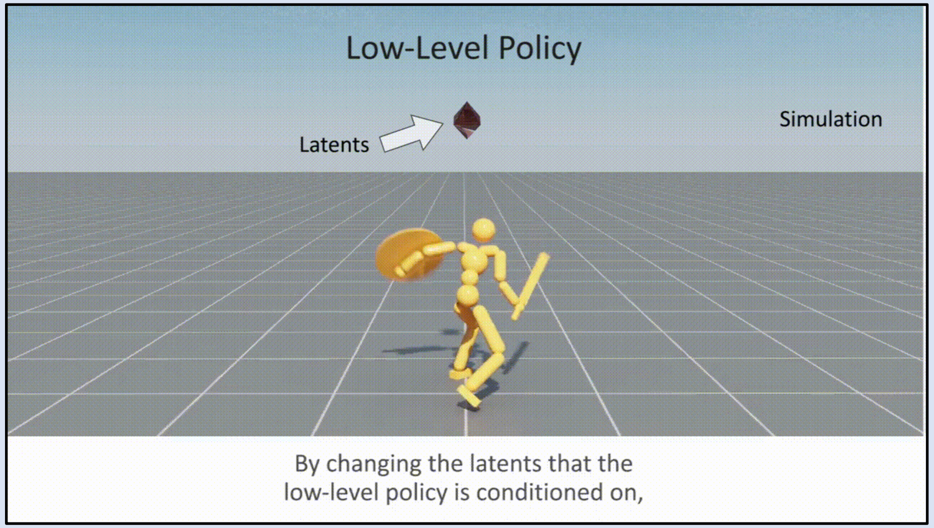
From: Computer Animation Festival Trailer, SIGGRAPH 2014

Simulating *a virtual world*

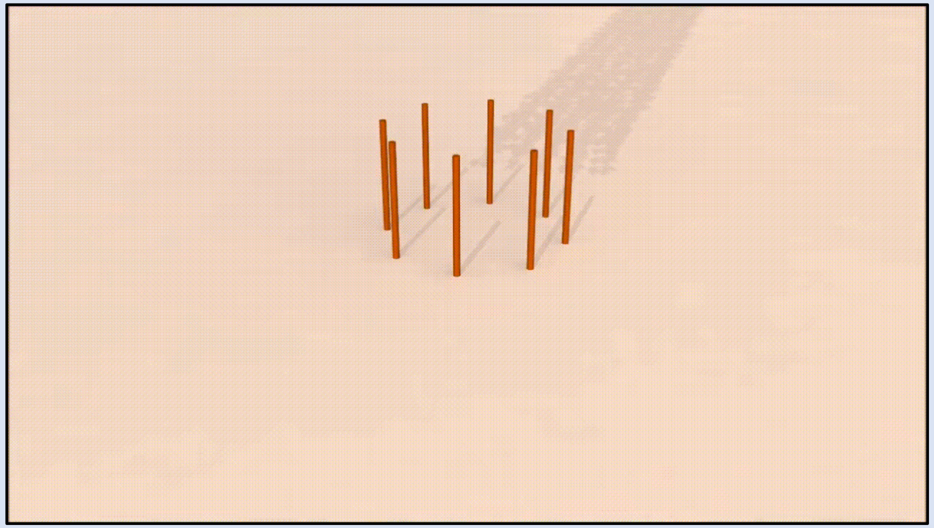
Rigid Body



Deul et al. CAVW2014



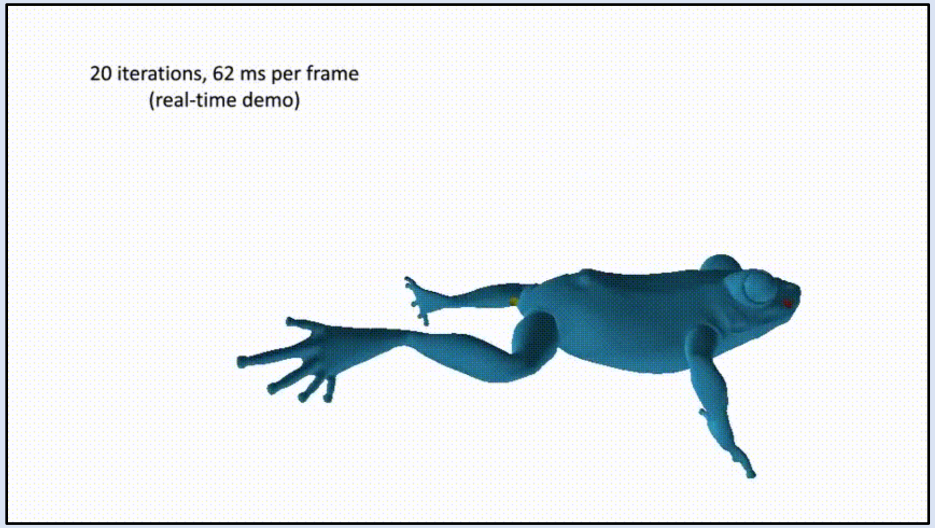
Peng et al. SIGGRAPH2022



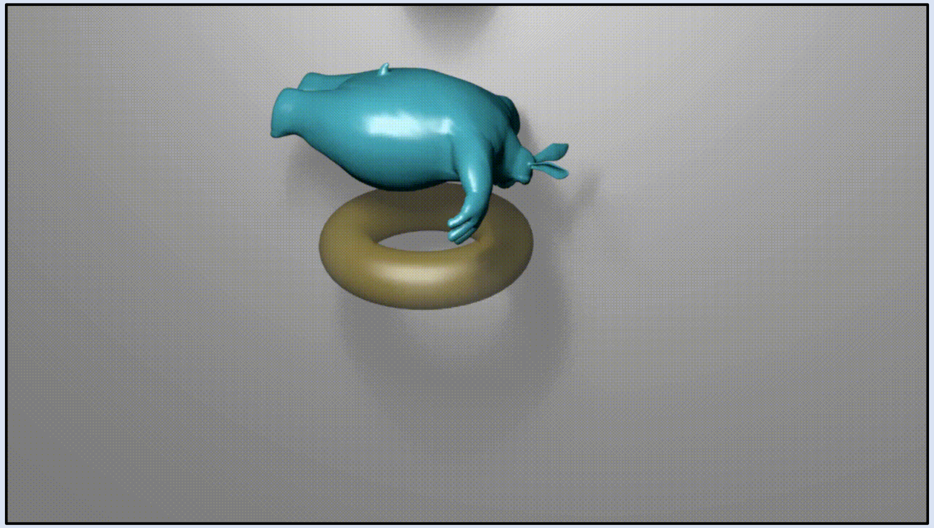
Deul et al. CAVW2014

Muller et al. SCA2020

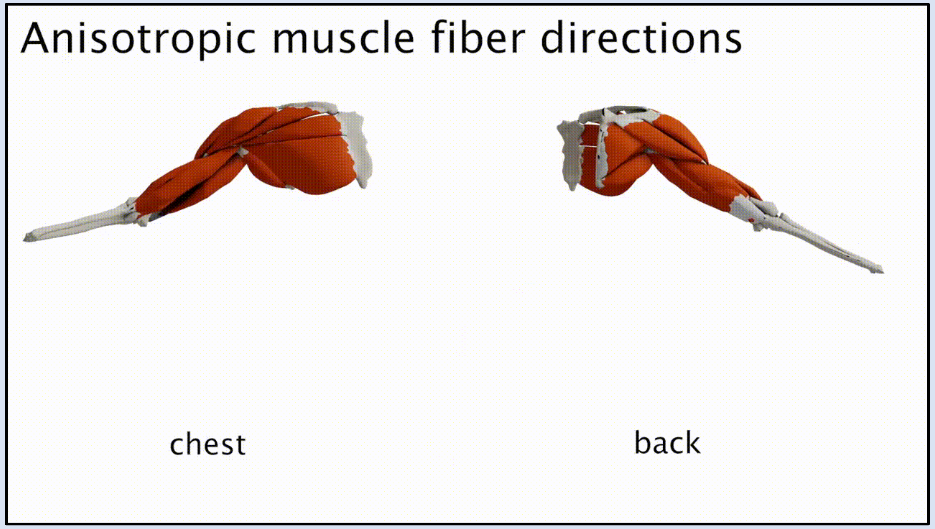
Deformable Object



Liu et al. SIGGRAPH Asia2013



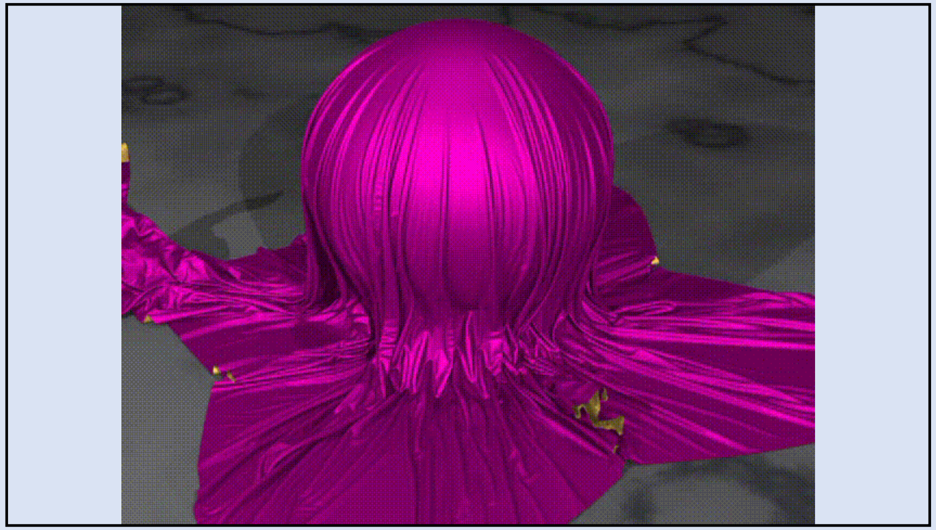
Liu et al. SIGGRAPH2017



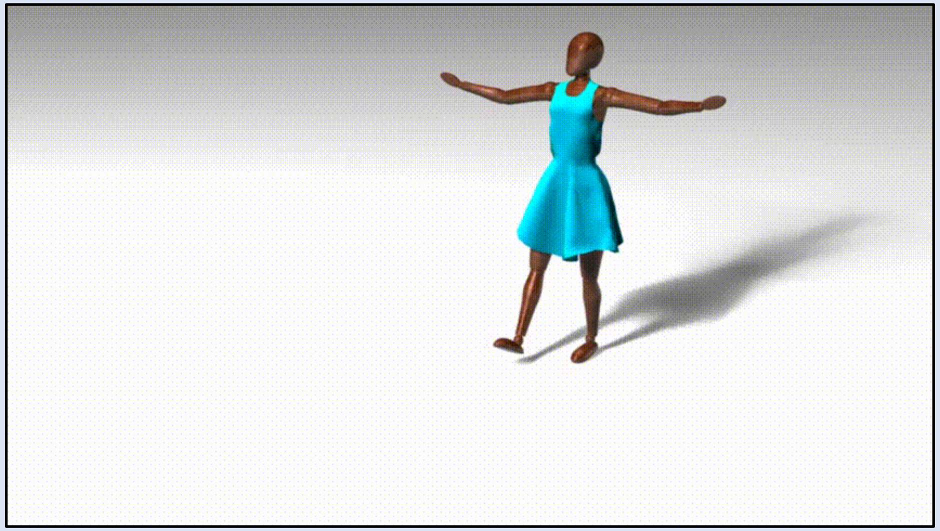
Modi et al. CGF2020

Liu et al. SIGGRAPH2013

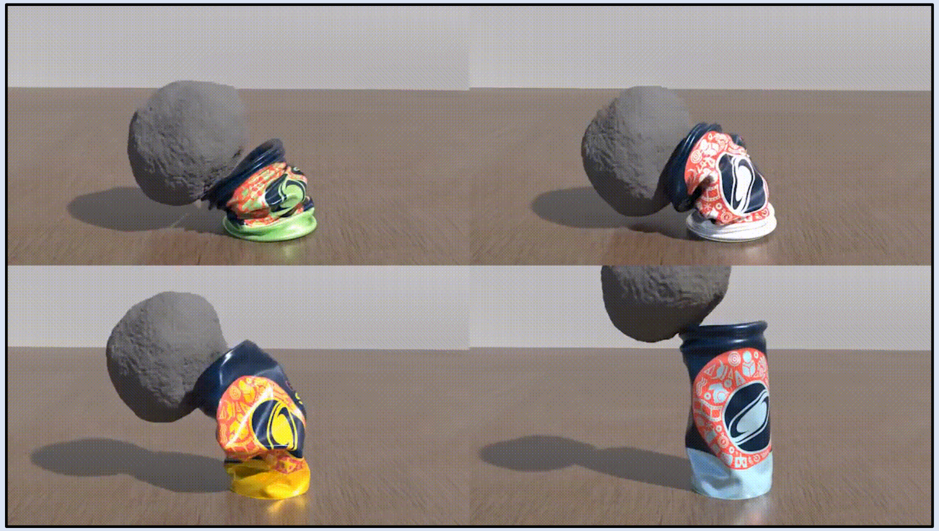
Thin Shell



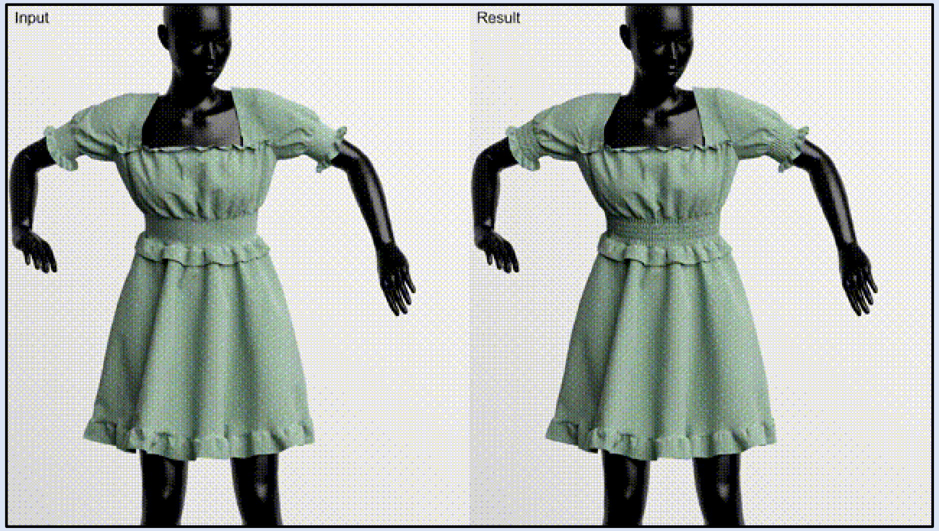
Selle et al. TVCG2015



Liu et al. SIGGRAPH Asia2013

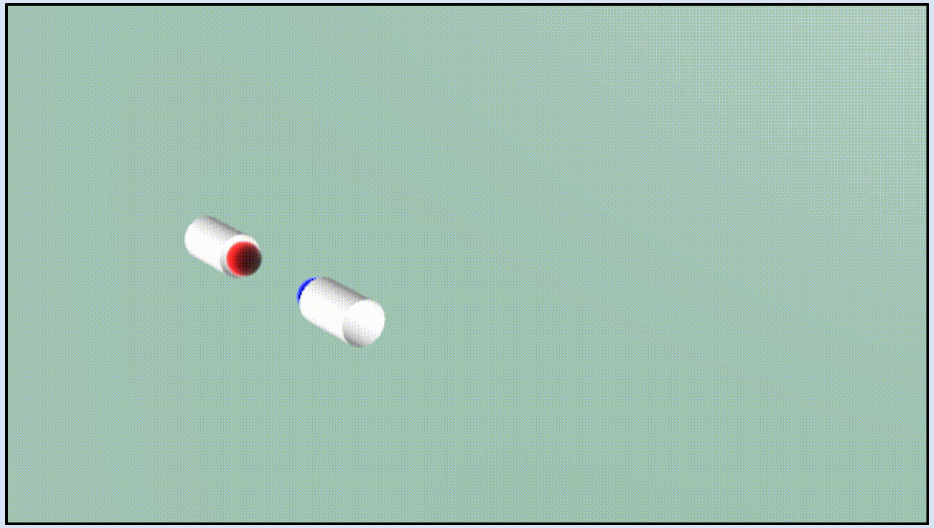


Guo et al. SIGGRAPH2018

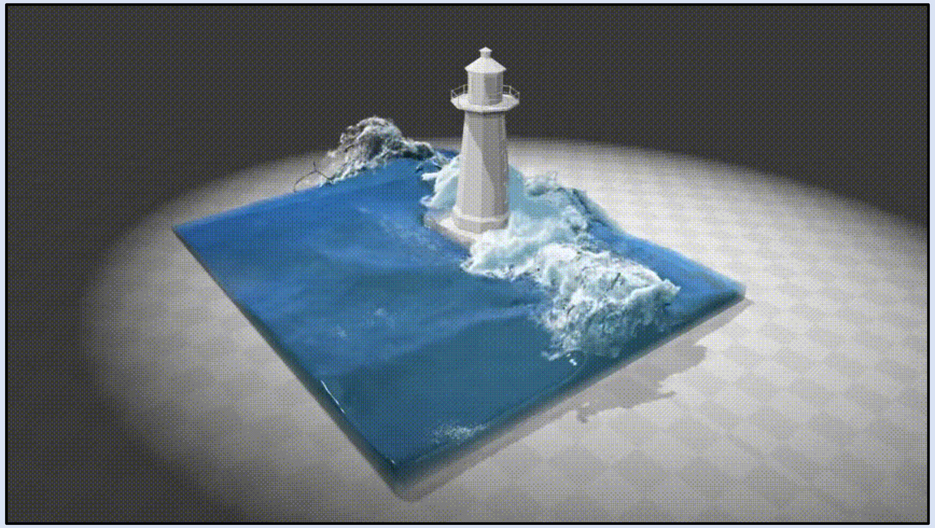


Huamin Wang SIGGRAPH2021

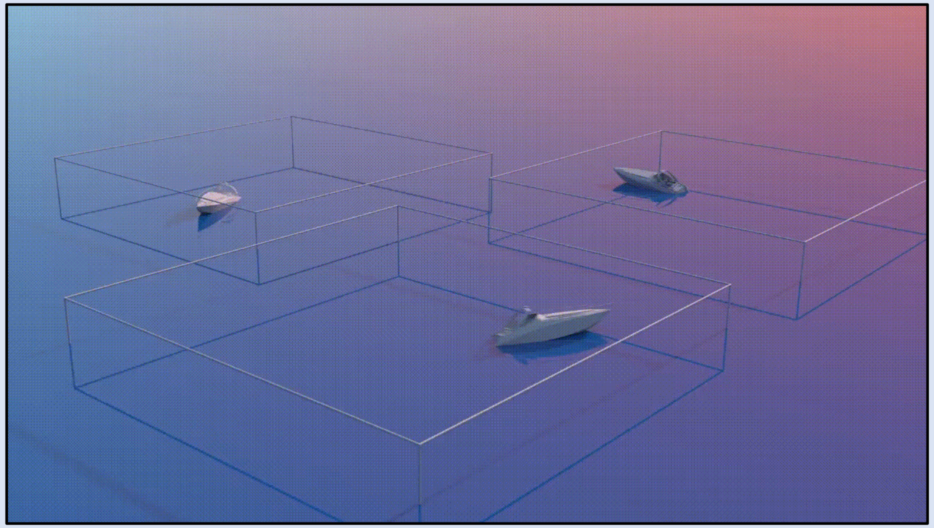
Fluid



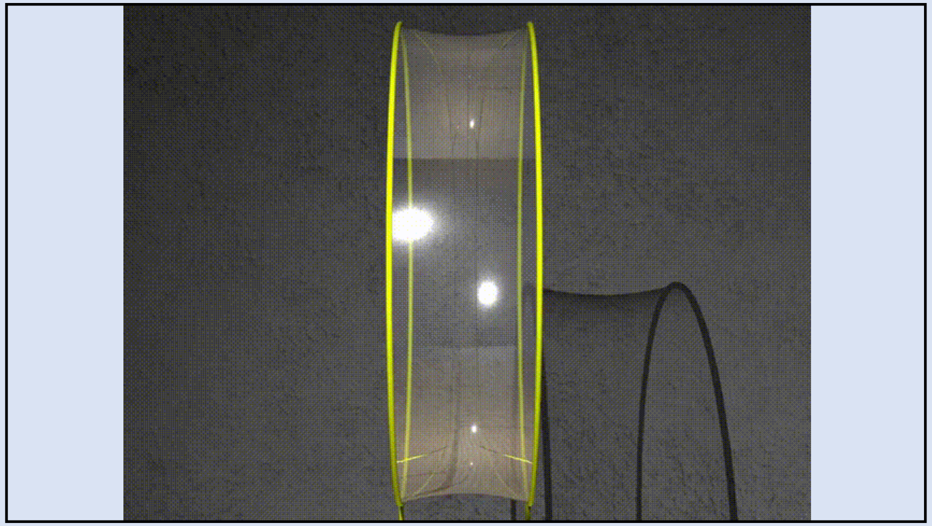
Qu et al. SIGGRAPH2019



Macklin et al. SIGGRAPH2013



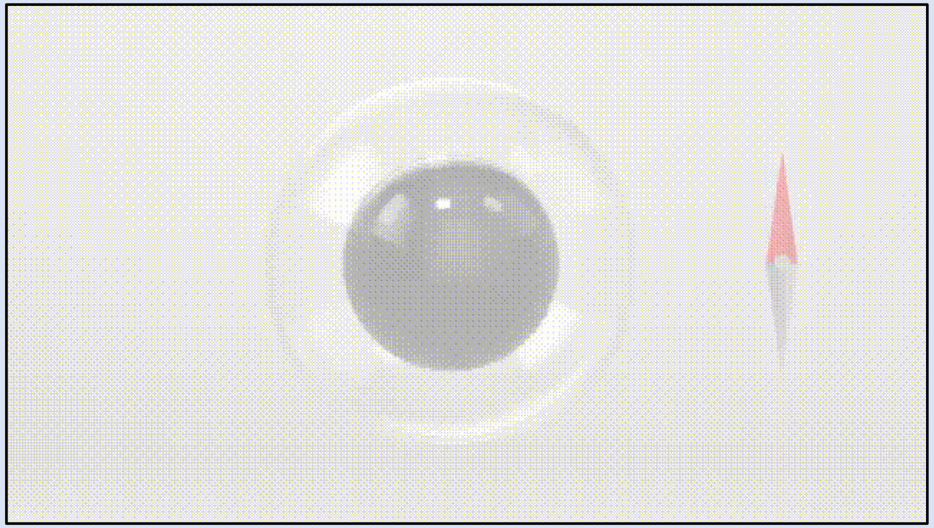
Huang et al. SIGGRAPH Asia2021



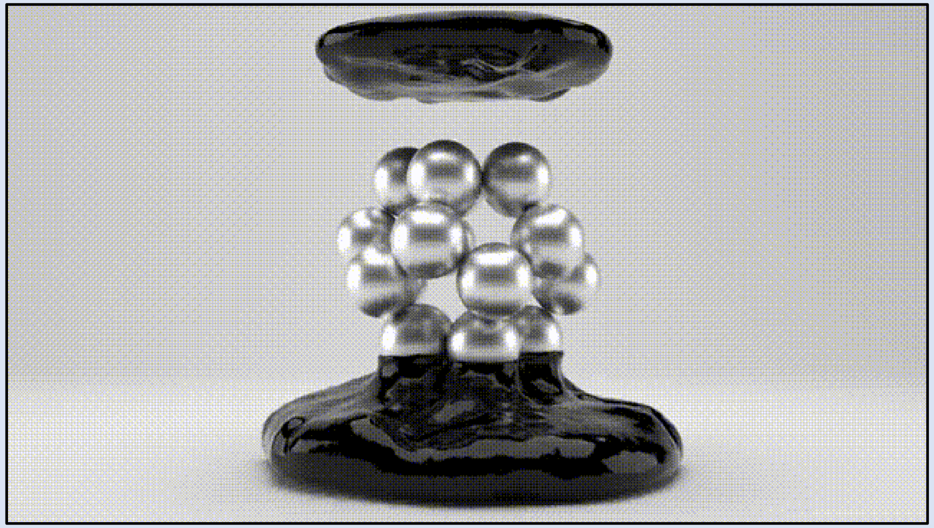
Zhu et al. SIGGRAPH2014

Anything else…

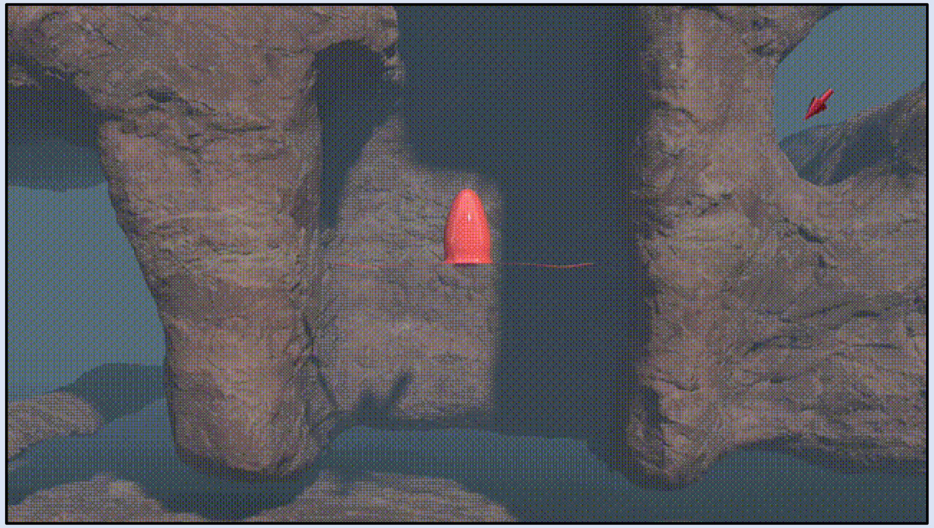
Ruan et al. SIGGRAPH2021



Ni et al. SIGGRAPH2020



Sun et al. SIGGRAPH Asia2021



Chen et al. SIGGRAPH2022

Online Resources

- GAMES103, GAMES201 on Bilibili
- Physics-based Animation by David Levin
- Ten Minute Physics by Matthias Müller
- http://www.physicsbasedanimation.com/
- SIGGRAPH(Asia) papers, courses…

Some Basics…

Physics Model

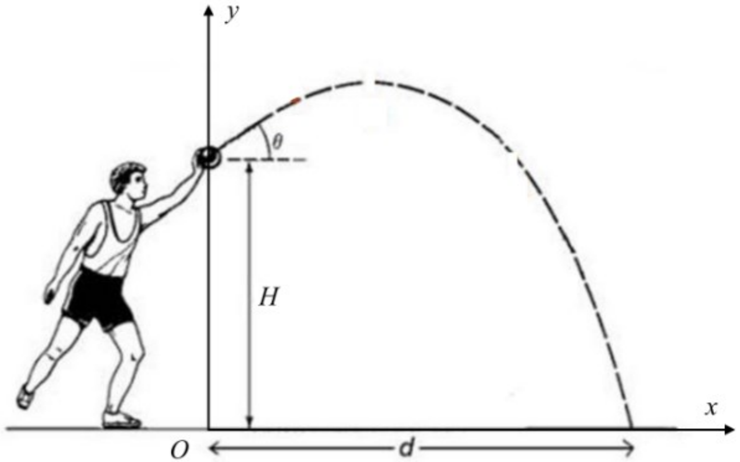
> The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are thus completely known, and the difficulty is only that the exact application of these laws leads to equations much too complicated to be soluble.
>
> Paul M. Dirac, 1929

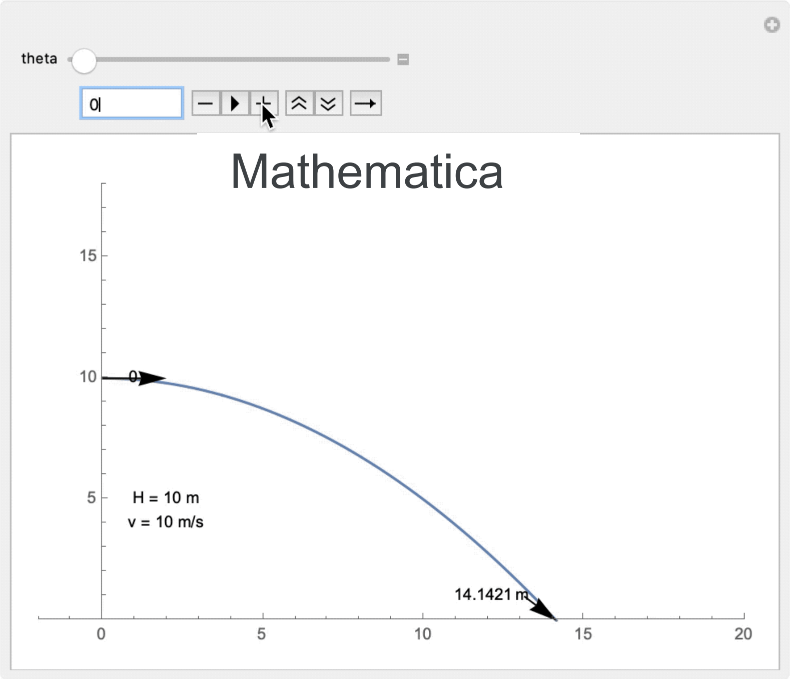
$$F = ma$$

- For a particle:

$$x = v_0 cos\theta \cdot t$$

$$y = H + v_0 sin\theta \cdot t + \frac{1}{2}(-g) \cdot t^2$$





Physics Model

> The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are thus completely known, and the difficulty is only that the exact application of these laws leads to equations much too complicated to be soluble.
>
> Paul M. Dirac, 1929

$$F = ma$$

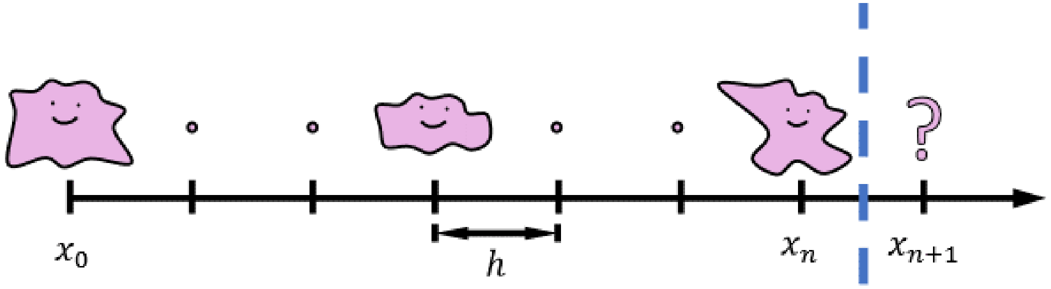
- For a particle:
  - $a = f(t),\ v = \int f(t)\,dt + v_0$. Easy to solve!
- A piece of material consists of **infinite number** of particles
  - When **analytical solutions** don't exist, we look for **numerical solutions**
  - We need **discretization, calculus, differential equations, tensor analysis, field theories…**

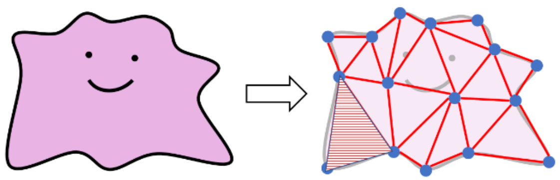
Three Building Blocks of Numerical Simulation

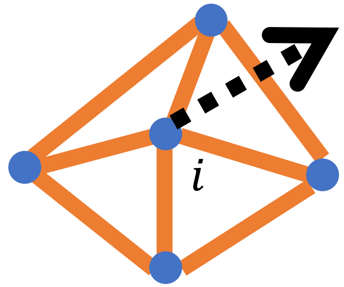
Physics Simulation = Kinematics $\frac{d\mathbf{x}_p(t)}{dt} = \mathbf{v}(\mathbf{x}_p, t)$ & Dynamics $\frac{d\mathbf{v}_p(t)}{dt} = \frac{\mathbf{f}(\mathbf{x}_p, t)}{m_p}$

Approx–imation →

Temporal Discretization + Spatial Discretization + Numerical Solver







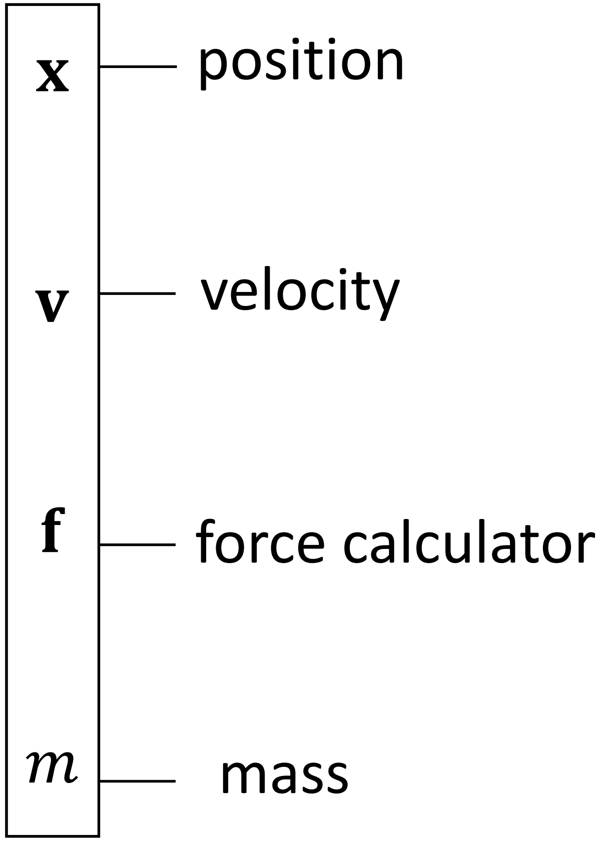
$\Delta \mathbf{M}\mathbf{v} = \mathbf{f}h$

Questions:

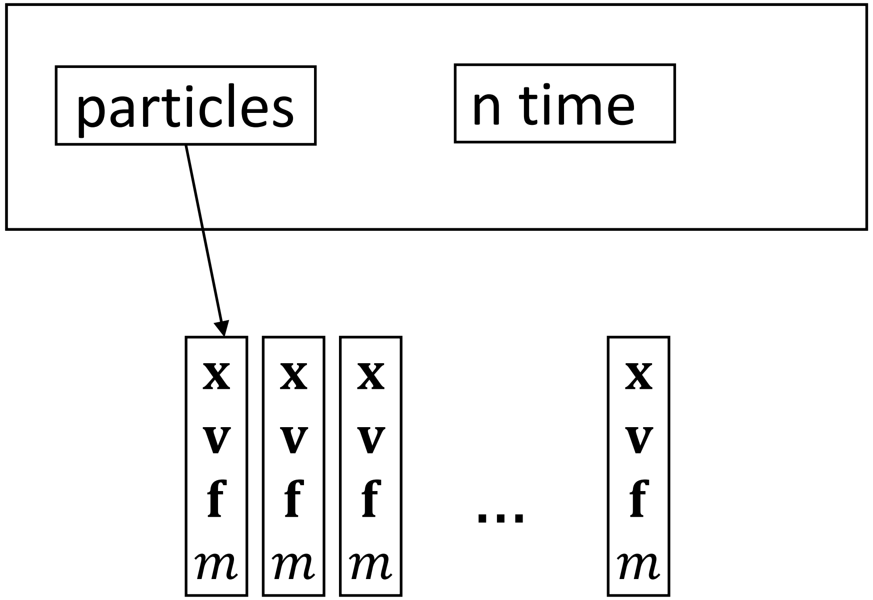
1. What is the **spatial representation**?
2. How to solve dynamics (**discretize momentum equations**) based on the representation?
3. How to perform kinematics via **time integration** (for a single time step)?

Your First Simulator: Spring-Mass System

Spatial Discretization: 1. Particle System



Particle Structure



Particle System

Spatial Discretization: 2. Spring Energy

- Use **springs** to mimic elasticity energy
- For one spring:

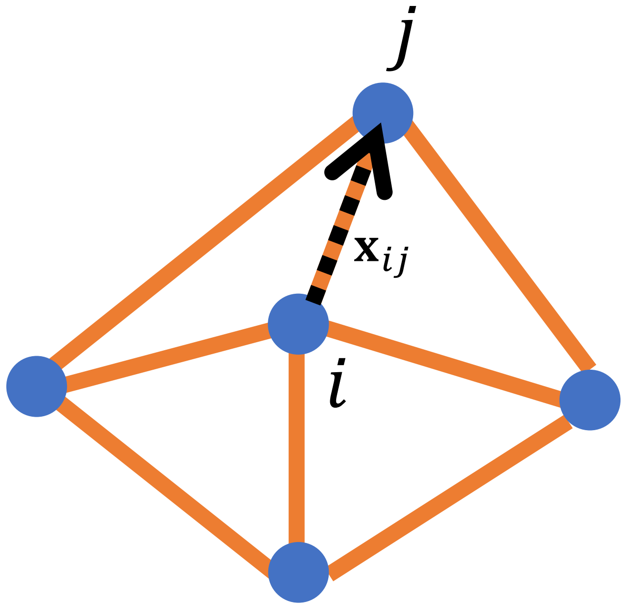
$$\mathbf{x}_{ij} = \mathbf{x}_j - \mathbf{x}_i, \qquad E_{ij} = \frac{1}{2} k (\|\mathbf{x}_{ij}\| - l_0)^2$$

- Force from particle j to particle i:

$$\mathbf{f}_{ij} = k\left(\|\mathbf{x}_{ij}\| - l_0\right) \frac{\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|} = -\mathbf{f}_{ji}$$

- The total force on particle i:

$$\mathbf{f}_i = \sum_{j \in N(i)} \mathbf{f}_{ij} + \mathbf{f}_i^{ext}$$



Temporal Discretization

$$\frac{d\mathbf{x}_p(t)}{dt} = \mathbf{v}(\mathbf{x}_p, t)$$

$$\frac{d\mathbf{v}_p(t)}{dt} = \frac{\mathbf{f}(\mathbf{x}_p, t)}{m_p}$$

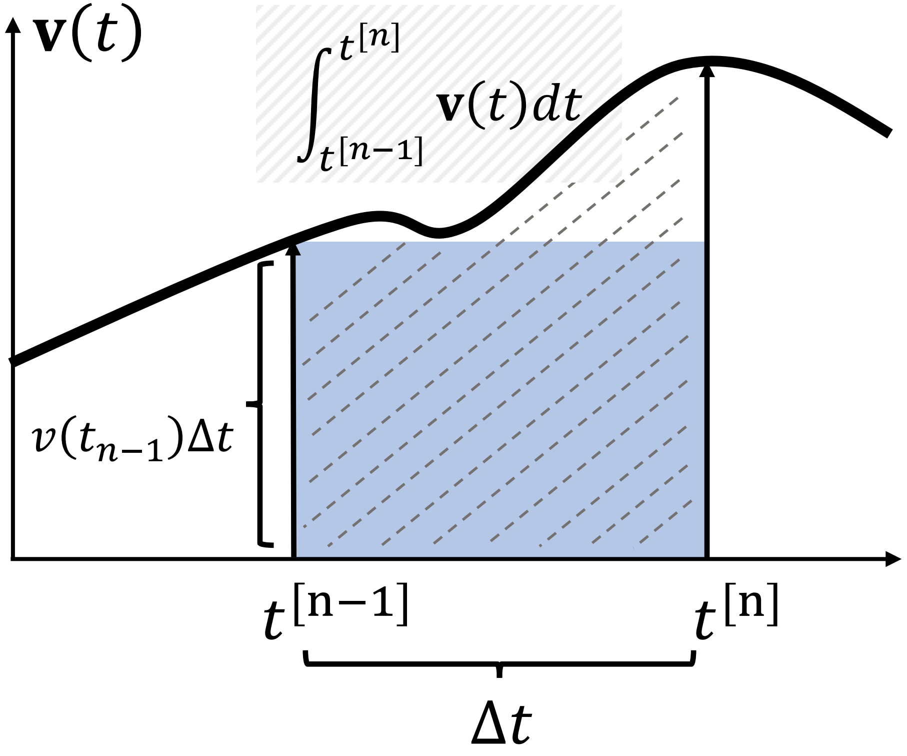
integrate

$$\mathbf{x}_{\mathrm{p}}(t_n) - \mathbf{x}_p(t_{n-1}) = \int_{t_{n-1}}^{t_n} \mathbf{v}_{\mathrm{p}}(t)\, dt$$

$$\mathbf{v}_p(t_n) - \mathbf{v}_{\mathrm{p}}(t_{n-1}) = \frac{1}{m} \int_{t_{n-1}}^{t_n} f(\mathbf{x}_p, t)\, dt$$

Temporal Discretization: Explicit Euler

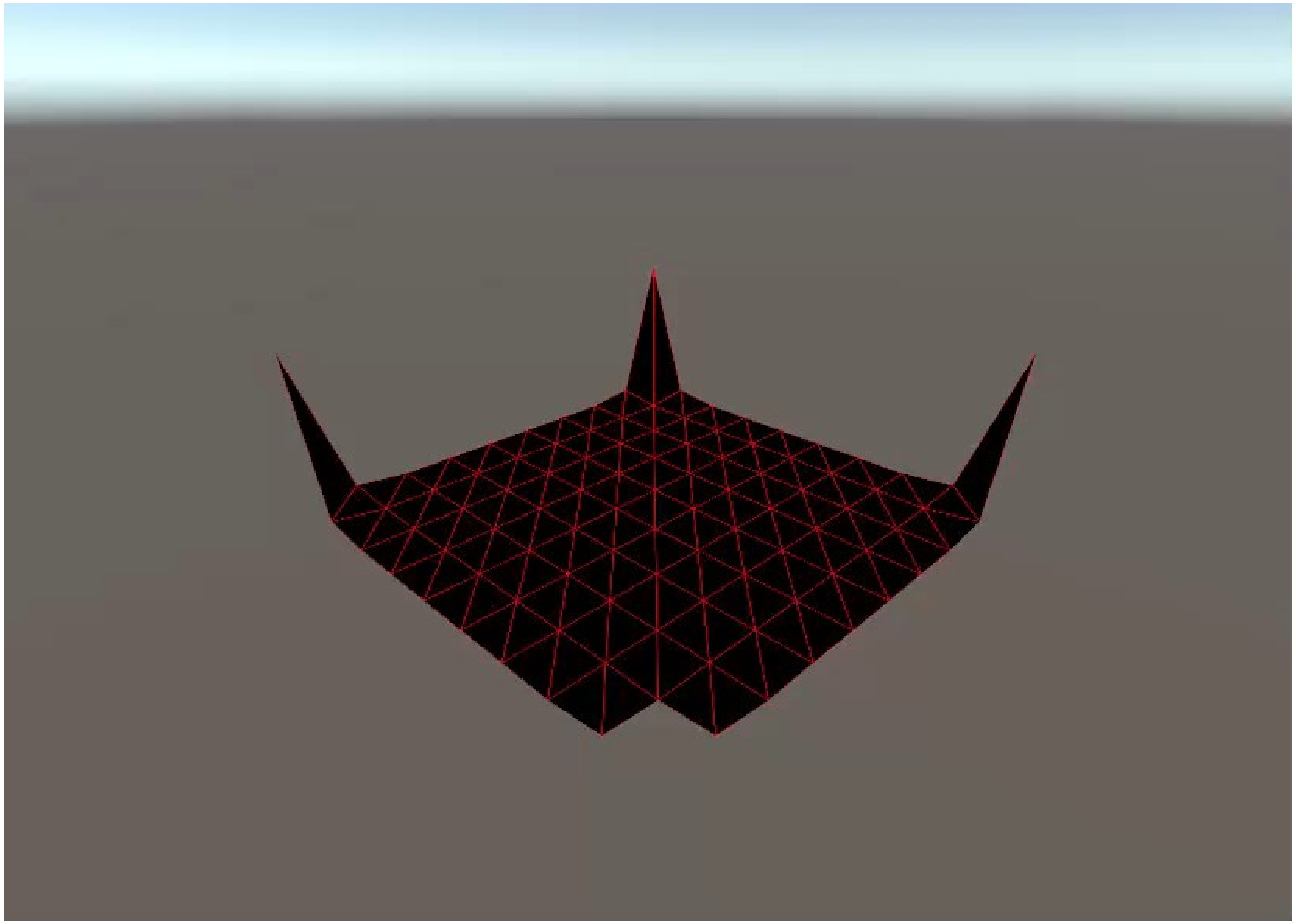
- We don't want integrations
- $\mathbf{x}(t_n) - \mathbf{x}(t_{n-1}) = \int_{t_{n-1}}^{t_n} \mathbf{v}\,(t)\,dt \approx \mathbf{v}(t_{n-1})\Delta t$
- $\mathbf{v}(t_n) - \mathbf{v}(t_{n-1}) = \frac{1}{m}\int_{t_{n-1}}^{t_n} \mathbf{f}(t)\,dt \approx \frac{1}{m}\mathbf{f}(t_{n-1})\Delta \mathrm{t}$
- We know this is very inaccurate…



Explicit Euler Simulator

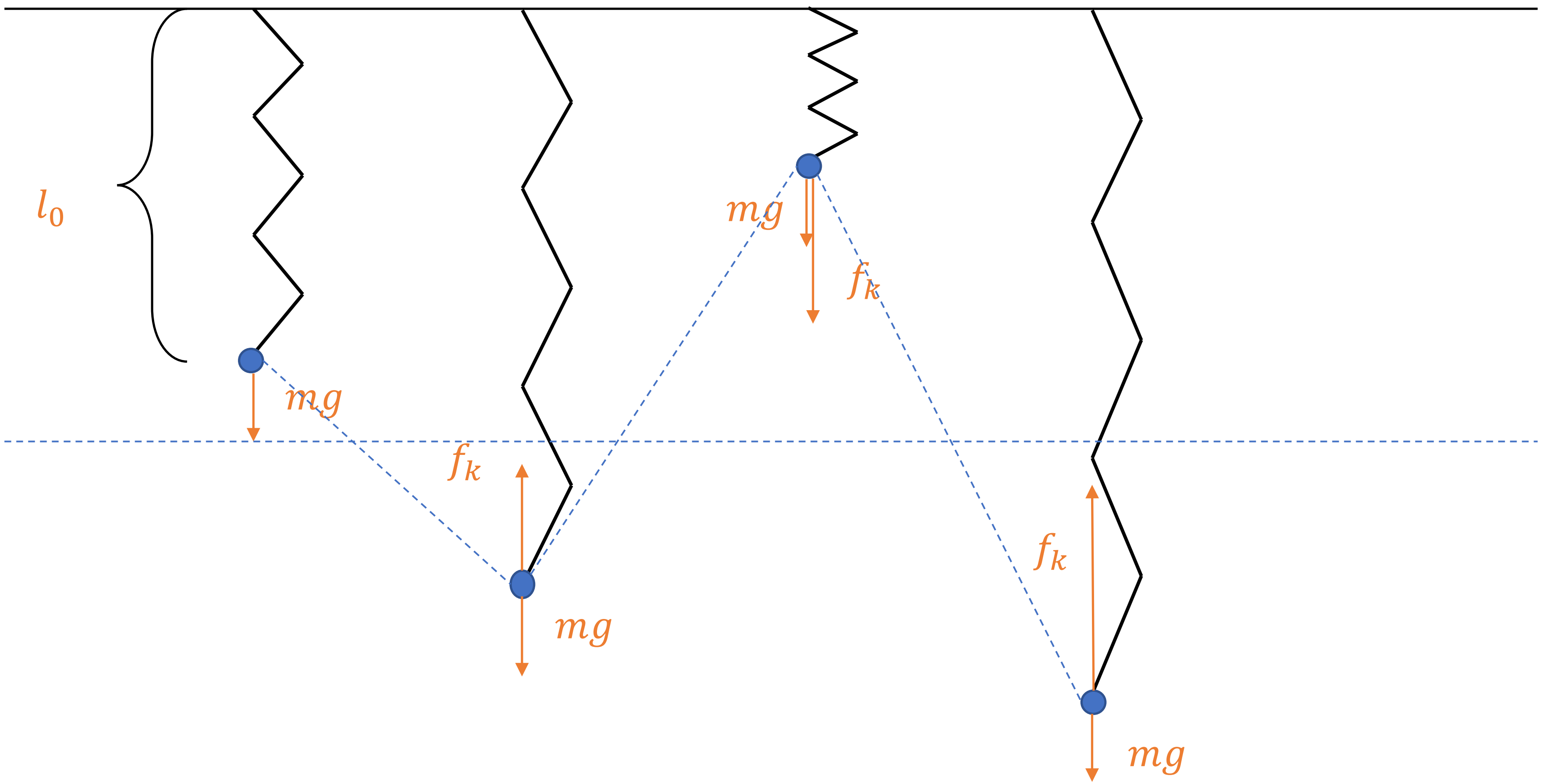
- At each step $t_n \rightarrow t_{n+1}$:
  - For each particle:
    - Compute $\mathbf{f}(t_n) = \Sigma\mathbf{f}_{ij}$ using current particle positions: $\mathbf{f}_{ij} = k\left(\|\mathbf{x}_{ij}\| - l_0\right)\frac{\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|}$
    - Update its position $\mathbf{x}(t_{n+1}) = \mathbf{x}(t_n) + \mathbf{v}(t_n)\Delta t$
    - Update its velocity $\mathbf{v}(t_{n+1}) = \mathbf{v}(t_n) + \frac{1}{m}\mathbf{f}(t_n)\Delta \mathrm{t}$

Here is the result



Why?

- Suppose Δt is large → energy increase!



$Explode$!

$force\ balance$:
$mg = f_k$

How to Evaluate Discretization & Integration?

- Discretization → *Truncation Errors*:

$$f(x+h) = \underbrace{f(x) + f'(x)h + \frac{f''(x)}{2}h^2}_{n^{th}\ Order\ Approximation} + \underbrace{\ldots}_{O(h^{n+1})\ Truncation\ Error}$$

- Integration → *Error could Accumulate*:
  - ➢ Stability:

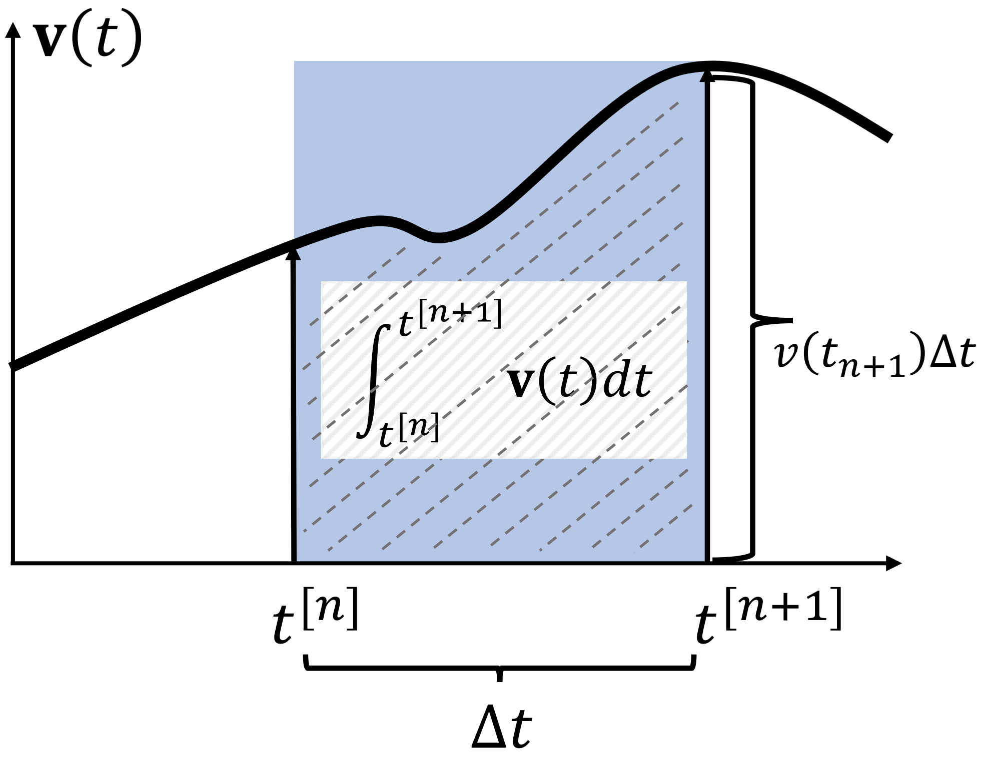
    Do *errors* accumulate? Is there a upper bound? (Error Propagation, Energy Preservation)
  - ➢ Convergence (Consistency):

    If $h \to 0$, will $Error \to 0$?
  - ➢ Accuracy and Convergence (Speed):

    n^{th} *Order Accurate* with $O(h^{n+1})$ error

To Improve Stability: Implicit Euler

- Explicit Euler:
  - $\mathbf{x}(t_{n+1}) = \mathbf{x}(t_n) + \mathbf{v}(t_n)\Delta t$
  - $\mathbf{v}(t_{n+1}) = \mathbf{v}(t_n) + \frac{1}{m}\mathbf{f}(t_n)\Delta \mathrm{t}$
- Implicit Euler:
  - $\mathbf{x}(t_{n+1}) = \mathbf{x}(t_n) + \mathbf{v}(t_{n+1})\Delta t$
  - $\mathbf{v}(t_{n+1}) = \mathbf{v}(t_n) + \frac{1}{m}\mathbf{f}(t_{n+1})\Delta \mathrm{t}$
- Later we'll see why implicit Euler is stable



Implicit Euler

- Target: solve equations for $\mathbf{x}_{n+1}$ and $\mathbf{v}_{n+1}$
- For convenience, denote $h = \Delta t$, $\mathbf{x}, \mathbf{v}, \mathbf{f} \in R^{3n}$ are collections of all particles

Implicit Euler

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\,\mathbf{v}_{n+1}$$
$$\mathbf{v}_{n+1} = \mathbf{v}_n + h\,\mathrm{M}^{-1}\,\mathbf{f}(\mathbf{x}_{n+1})$$

Substitute $\mathbf{v}_{n+1}$ into $\mathbf{x}_{n+1}$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\mathbf{v}_n + h^2\,\mathrm{M}^{-1}\,\mathbf{f}(\mathbf{x}_{n+1})$$

Divide $f(x_{n+1})$ into 2 parts

$$\begin{aligned}\mathbf{x}_{n+1} &= (\mathbf{x}_n + h\mathbf{v}_n + h^2\,\mathrm{M}^{-1}\,\mathbf{f}_{ext}) + h^2\,\mathrm{M}^{-1}\,\mathbf{f}_{int}(\mathbf{x}_{n+1}) \\ &= \underbrace{(\mathbf{x}_n + h(\mathbf{v}_n + h\,\mathrm{M}^{-1}\,\mathbf{f}_{ext}))}_{\text{independent of } \mathbf{x}_{n+1}} + \underbrace{h^2\,\mathrm{M}^{-1}\,\mathbf{f}_{int}(\mathbf{x}_{n+1})}_{\text{function of } \mathbf{x}_{n+1}}\end{aligned}$$

denote the first term as $\mathbf{y}$

Implicit Euler

- $\mathbf{x}_{n+1} = \mathbf{y} + h^2 \mathrm{M}^{-1} \mathbf{f}_{int}(\mathbf{x}_{n+1})$

$$= \quad \mathbf{x}_{n+1} = argmin_{\mathbf{x}}\, g(\mathbf{x}), \text{ for } g(\mathbf{x}) = \frac{1}{2h^2} |\mathbf{x} - \mathbf{y}|_{\mathrm{M}}^2 + E(\mathbf{x})$$

This is because:

$$\nabla g(\mathbf{x}) = \frac{1}{h^2} \mathrm{M}(\mathbf{x} - \mathbf{y}) - \mathbf{f}_{int}(\boldsymbol{x}) = 0$$

$$\Downarrow$$

$$\mathbf{x} - \mathbf{y} - h^2 \mathrm{M}^{-1} \mathbf{f}_{int}(x) = 0$$

Applicable to every system
not just a mass-spring system

$$\|\mathbf{x}\|_{\mathrm{M}}^2 = \mathbf{x}^{\mathrm{T}} \mathrm{M} \mathbf{x}$$

$$\mathbf{f}_{int}(\mathbf{x}_{n+1}) = \frac{dE(\mathbf{x})}{d\mathbf{x}}$$

Implicit Euler

- $\mathbf{x}_{n+1} = \mathbf{y} + h^2 \mathrm{M}^{-1} \mathbf{f}_{\mathbf{int}}(\mathbf{x}_{\mathbf{n+1}})$

$$\mathbf{x}_{n+1} = argmin_{\mathbf{x}}\, g(\mathbf{x}),\ \text{for } g(\mathbf{x}) = \frac{1}{2h^2} |\mathbf{x}-\mathbf{y}|_{\mathrm{M}}^2 + E(\mathbf{x})$$

- Implicit Euler = energy minimization:

$$argmin_{\mathbf{x}} \underbrace{\frac{1}{2h^2} |\mathbf{x}-\mathbf{y}|_{\mathrm{M}}^2}_{\text{inertia}} + \underbrace{E(\mathbf{x})}_{\text{elasticity}}$$

$$\|\mathbf{x}\|_{\mathrm{M}}^2 = \mathbf{x}^{\mathrm{T}} \mathrm{M} \mathbf{x}$$

$$\mathbf{f}_{int}(\mathbf{x}_{n+1}) = \frac{dE(\mathbf{x})}{d\mathbf{x}}$$

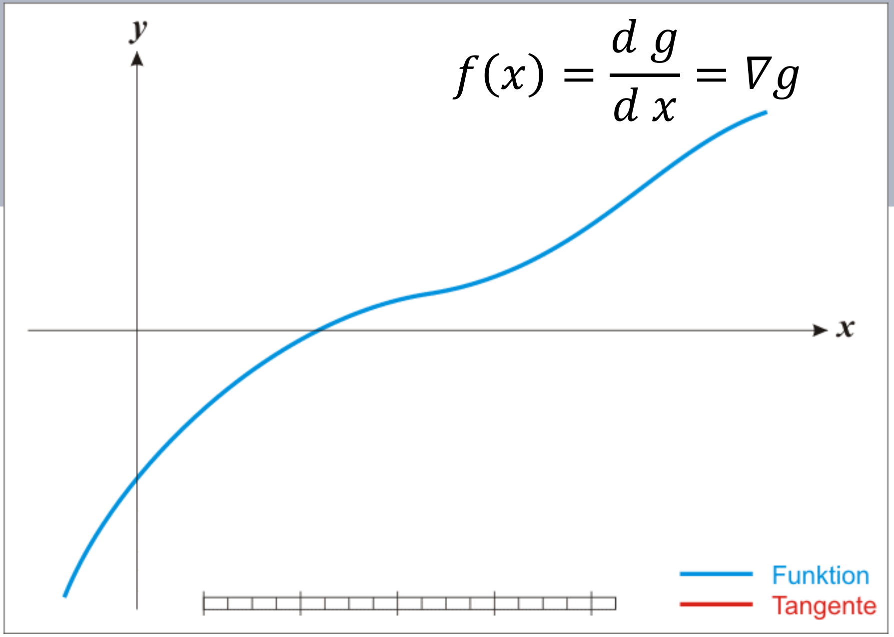
$$E_{ij} = \frac{1}{2} k(\|\mathbf{x}_{\mathbf{ij}}\| - l_0)^2$$

$$\mathbf{f}_{ij} = k(\|\mathbf{x}_{\mathbf{ij}}\| - l_0) \frac{\mathbf{x}_{\mathbf{ij}}}{\|\mathbf{x}_{\mathbf{ij}}\|}$$

- **Stable** under **any** timestep size

Numerical Solver

$$\mathbf{x}_{n+1} = argmin_{\mathbf{x}}\ g$$



- Newton's method:

  Start from a guess $\mathbf{x}_1$,

  update with

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathrm{H}(g)^{-1} \nabla g$$

Numerical Solver

$$\mathbf{x}_{n+1} = argmin_{\mathbf{x}}\ g$$

- Newton's method:

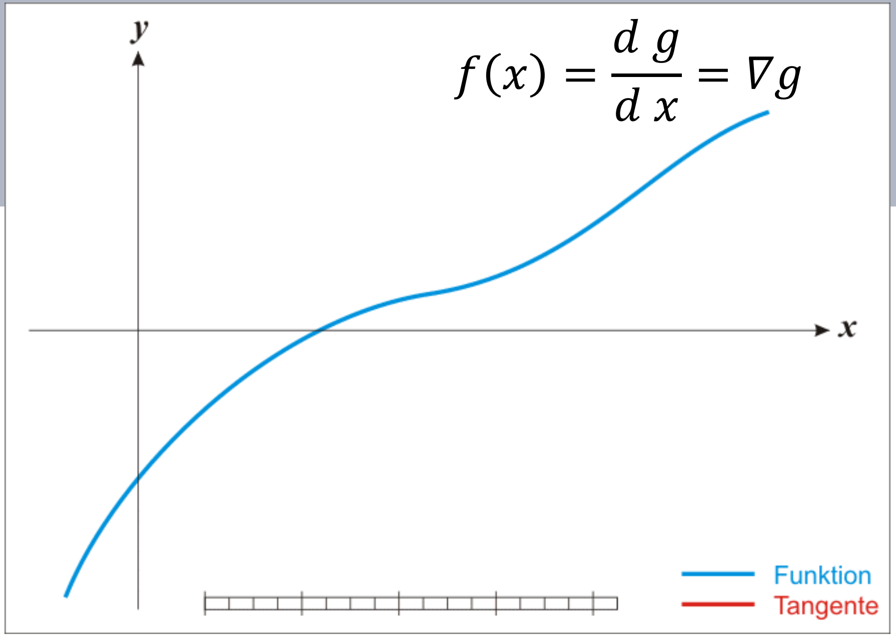
  Start from a guess $\mathbf{x}_1$,

  update with

  $$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathrm{H}(g)^{-1}\nabla g,$$

  until converge.

  - Compute Hessian matrix $\mathrm{H}(g) \in R^{3n\times 3n}$ at every step
  - Solve Matrix equation at every step (**main bottleneck**)
  - Line search: prevent overshoot



Algorithm 2: Newton Solver with Backtracking Line Search

$\mathbf{x}^{(1)} := \mathbf{y}$;
$g(\mathbf{x}^{(1)}) :=$ `evalObjective`$(\mathbf{x}^{(1)})$
for $k = 1, \ldots, \mathrm{numIterations}$ **do**
  $\nabla g(\mathbf{x}^{(k)}) :=$ `evalGradient`$(\mathbf{x}^{(k)})$
  $\nabla^2 g(\mathbf{x}^{(k)}) :=$ `evalHessian`$(\mathbf{x}^{(k)})$
  $\delta\mathbf{x}^{(k)} := -\nabla^2 g(\mathbf{x}^{(k)})^{-1}\nabla g(\mathbf{x}^{(k)})$
  $\alpha := 1/\beta$
  repeat
    $\alpha := \beta\alpha$
    $\mathbf{x}^{(k+1)} := \mathbf{x}^{(k)} + \alpha\delta\mathbf{x}^{(k)}$
    $g(\mathbf{x}^{(k+1)}) :=$ `evalObjective`$(\mathbf{x}^{(k+1)})$
  until $g(\mathbf{x}^{(k+1)}) \leq g(\mathbf{x}^{(k)}) + \gamma\alpha\,(\nabla g(\mathbf{x}^{(k)}))^{\mathsf{T}}\delta\mathbf{x}^{(k)}$;
end

Numerical Solver

To solve nonlinear equation systems:

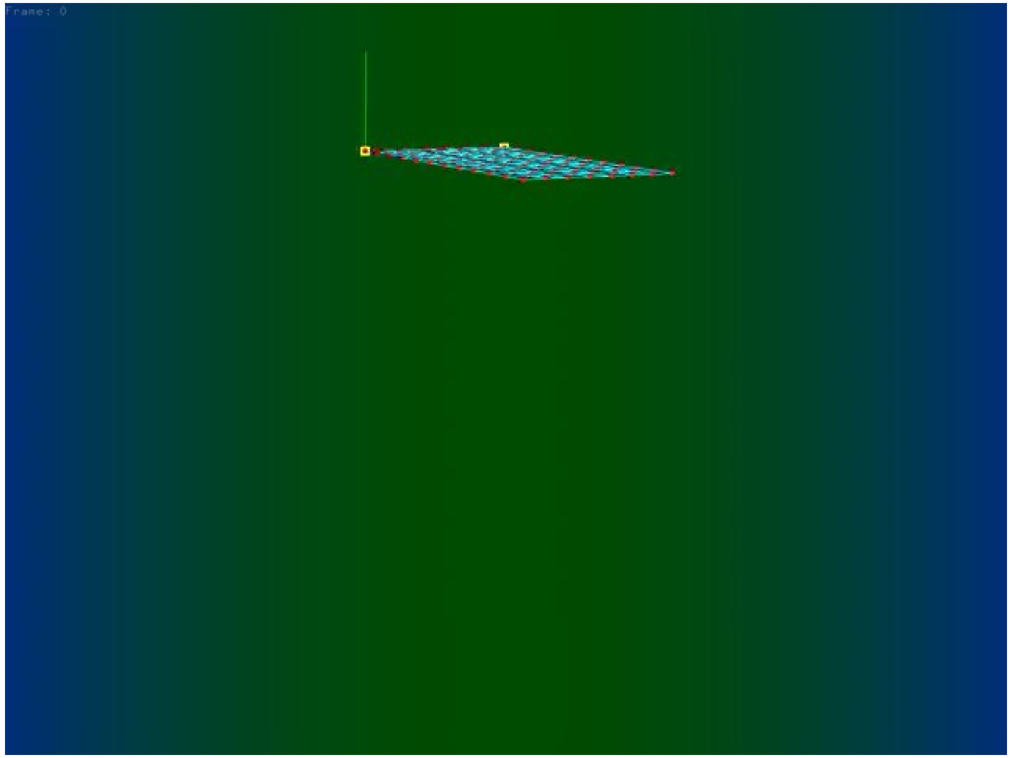
- Newton's methods
- Quasi-Newton Methods
- BFGS…

To solve matrix equations (linear or quadratic equation systems):

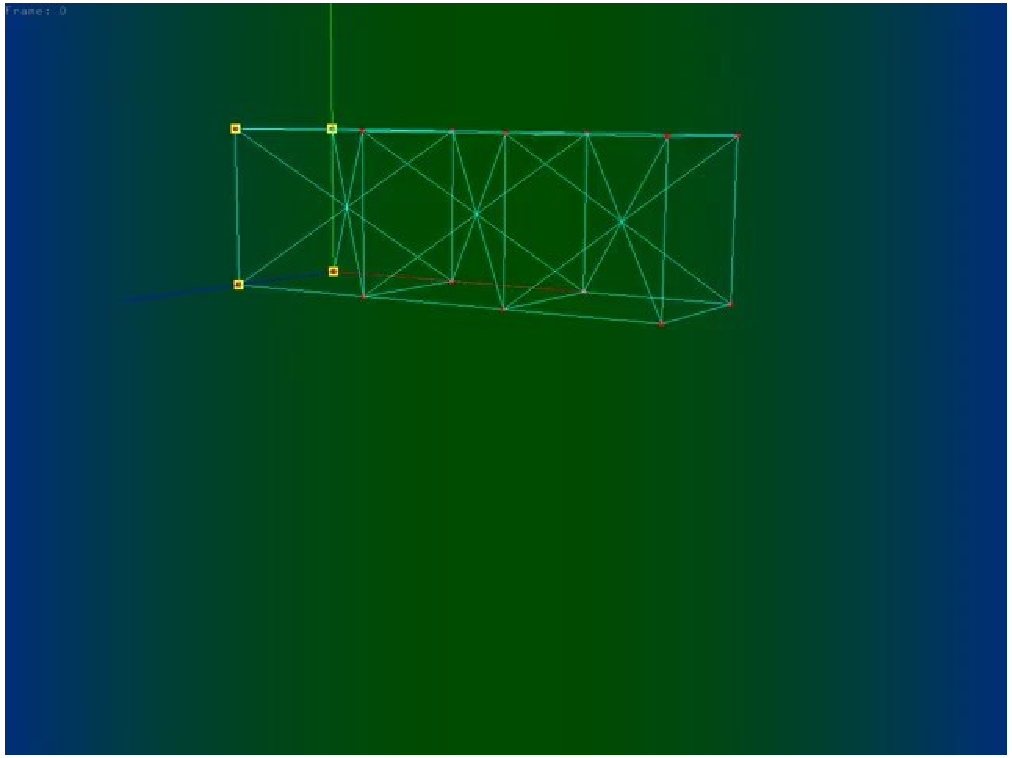
- Jacobi/Gauss-Seidel Iteration
- Conjugate Gradient
- Multigrid…

No one general optimal solver for all problems

Implicit Euler Results



cloth



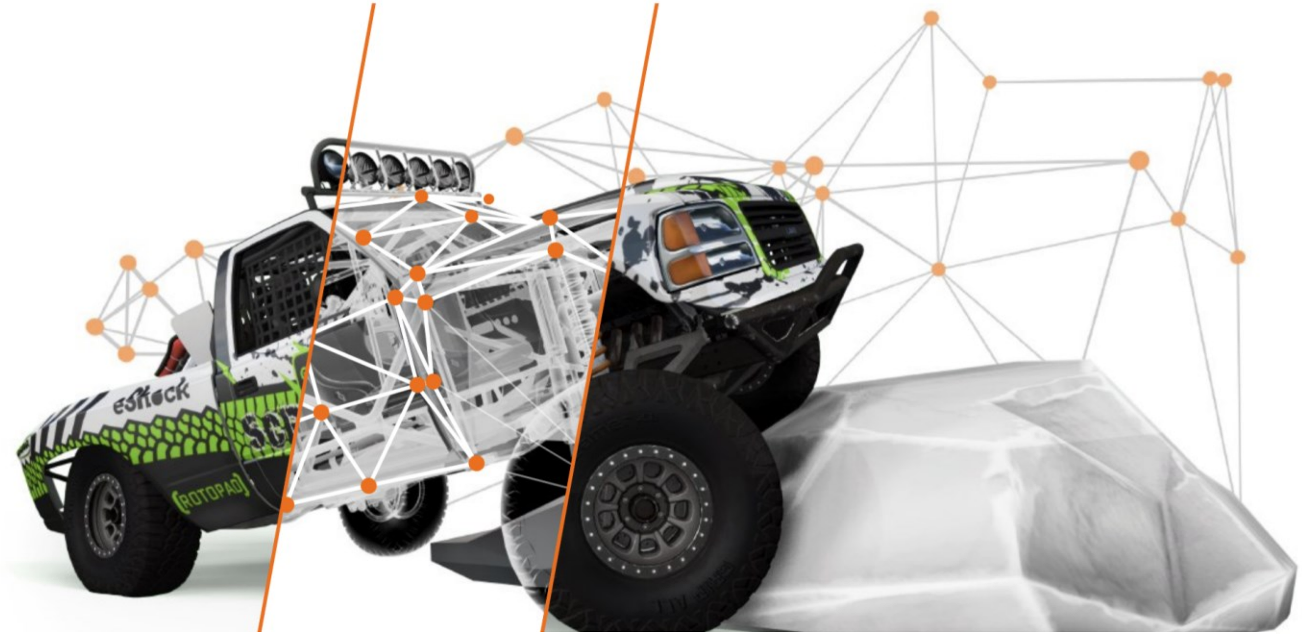
rod

This is Spring-Mass System

Huamin Wang SIGGRAPH2021
GPU-Based Simulation of Cloth Wrinkles at Submillimeter Levels

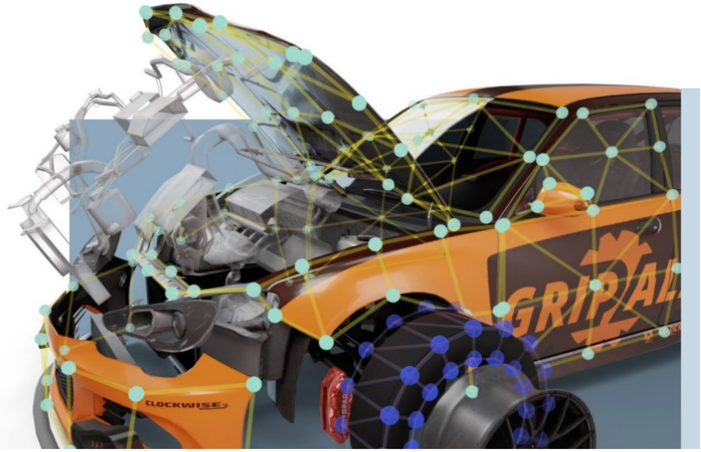
This is Also Spring Mass

BeamNG.drive | News | Community | Support | Documentation | Press | Search



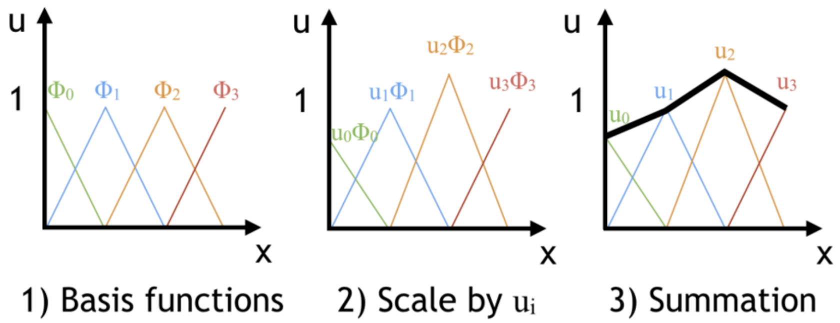
Soft-body physics

The BeamNG physics engine is at the core of the most detailed and authentic vehicle simulation you've ever seen in a game. Every component of a vehicle is simulated in real-time using nodes (mass points) and beams (springs). Crashes feel visceral, as the game uses an incredibly accurate damage model.

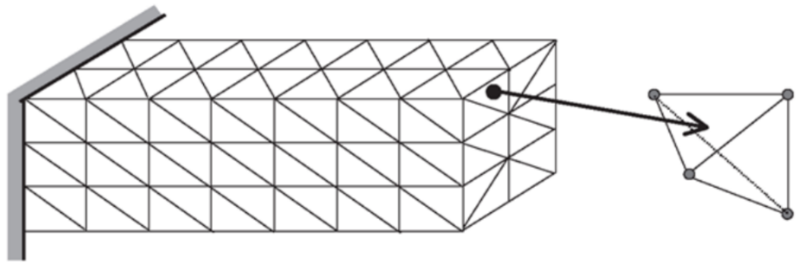


A More physical Way

Finite Element Method

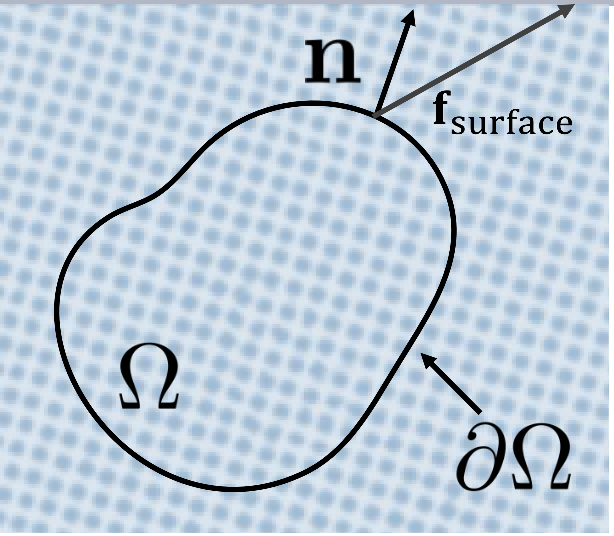


- Based on continuum mechanics:
  - Energy from deformation: $\Psi(F) = \mu F : F + \frac{\lambda}{2}\mathrm{tr}^2(F)$
  - Force from stress tensor: $\rho\ddot{x} = \nabla \cdot \sigma + f_{ext}$
- Spatial discretization

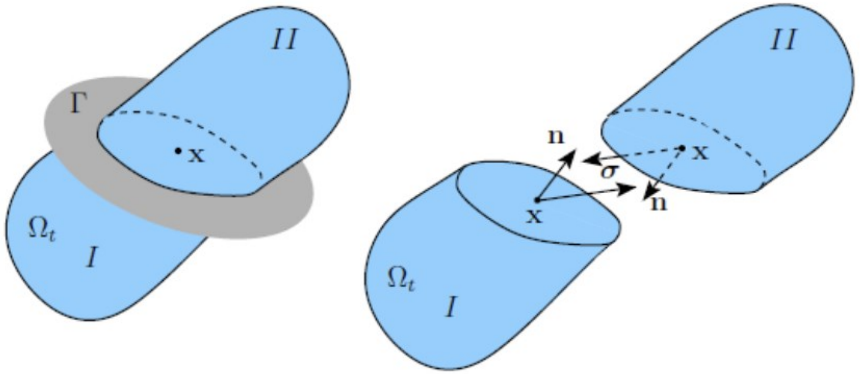


  - 1d: intervals;
  - 2d: triangles, squares, ……;
  - 3d: tetrahedra, cubes, ……
  - discretization of elastic energy: (e.g. StVK, Neo-Hookean, ……)

Appendix: Linear momentum balance for continuum mechanics



(a volume element): $\int_\Omega \rho \frac{d\mathbf{v}}{dt} \mathrm{d}V = \oint_{\partial\Omega} \mathbf{f}_{\text{surface}}\, \mathrm{d}S + \int_\Omega \mathbf{f}_{\text{body}} \mathrm{d}V$, For e.g., $\mathbf{f}_{\text{body}} = \rho\mathbf{g}$

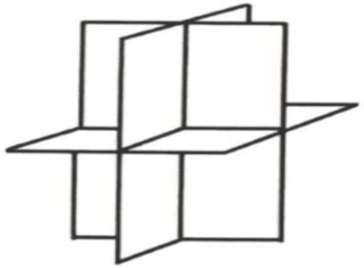


Traction **t**, (N/m^2, internal force per unit area/length)

$$\mathbf{t} = \boldsymbol{\sigma}\mathbf{n},$$

σ: Cauchy Stress Tensor,

$$\oint_{\partial\Omega} \boldsymbol{\sigma}\mathbf{n}\, \mathrm{d}S = \int_\Omega \nabla \cdot \boldsymbol{\sigma}\, \mathrm{d}V$$



$$\mathbf{n}_{yz} = \boldsymbol{e}_x \;\&\; \mathbf{f}_{yz} = \boldsymbol{\sigma}_x$$
$$\mathbf{n}_{zx} = \boldsymbol{e}_y \;\&\; \mathbf{f}_{zx} = \boldsymbol{\sigma}_y$$
$$\mathbf{n}_{xy} = \boldsymbol{e}_z \;\&\; \mathbf{f}_{xy} = \boldsymbol{\sigma}_z$$

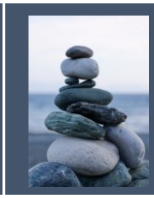
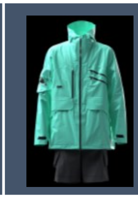
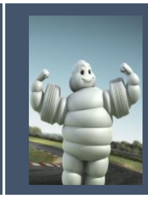
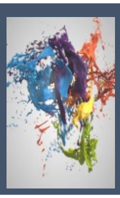
$$\mathbf{n} = n_x\boldsymbol{e}_x + n_y\boldsymbol{e}_y + n_z\boldsymbol{e}_z;$$

$$\mathbf{t} = n_x\boldsymbol{\sigma}_x + n_x\boldsymbol{\sigma}_y + n_x\boldsymbol{\sigma}_z$$
$$= [\boldsymbol{\sigma}_x \quad \boldsymbol{\sigma}_y \quad \boldsymbol{\sigma}_z]\mathbf{n} = \boldsymbol{\sigma}\mathbf{n}$$

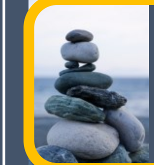
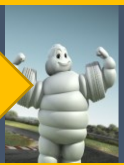
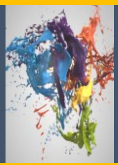
Cauchy's equation of motion $\rho \frac{d\mathbf{v}}{dt} = \nabla \cdot \boldsymbol{\sigma} + \mathbf{f}_{body}$

The equation of motion of an infinitesimal volume of any continuum.

Physics-Based Simulation Topics

Contents	 Rigid Bodies		 Cloth and Hair		 Soft Bodies		 Fluids		
Effects	Contacts	Fracture	Cloth	Hair	Elastic	plastic	Smoke	Drops and Waves	Splashes
Mesh	✓	✓	✓	✓	✓	✓		✓ (real-time)	?
Particle		☆ (meshless)					✓ (real-time)		✓
Grid			☆ (contact)	☆ (contact)			✓	✓	✓

Physics-Based Simulation Topics

Contents	 Rigid Bodies		 Cloth and Hair		 Soft Bodies			 Fluids	
Effects	Contacts	Fracture	Cloth	Hair	Elastic	plastic	Smoke	Drops and Waves	Splashes
Mesh	✓	✓	✓	✓	✓	✓		✓ (real-time)	?
Particle		☆ (meshless)					✓ (real-time)		✓
Grid			☆ (contact)	☆ (contact)			✓	✓	✓

Coupling

Hybrid Methods

Much More…

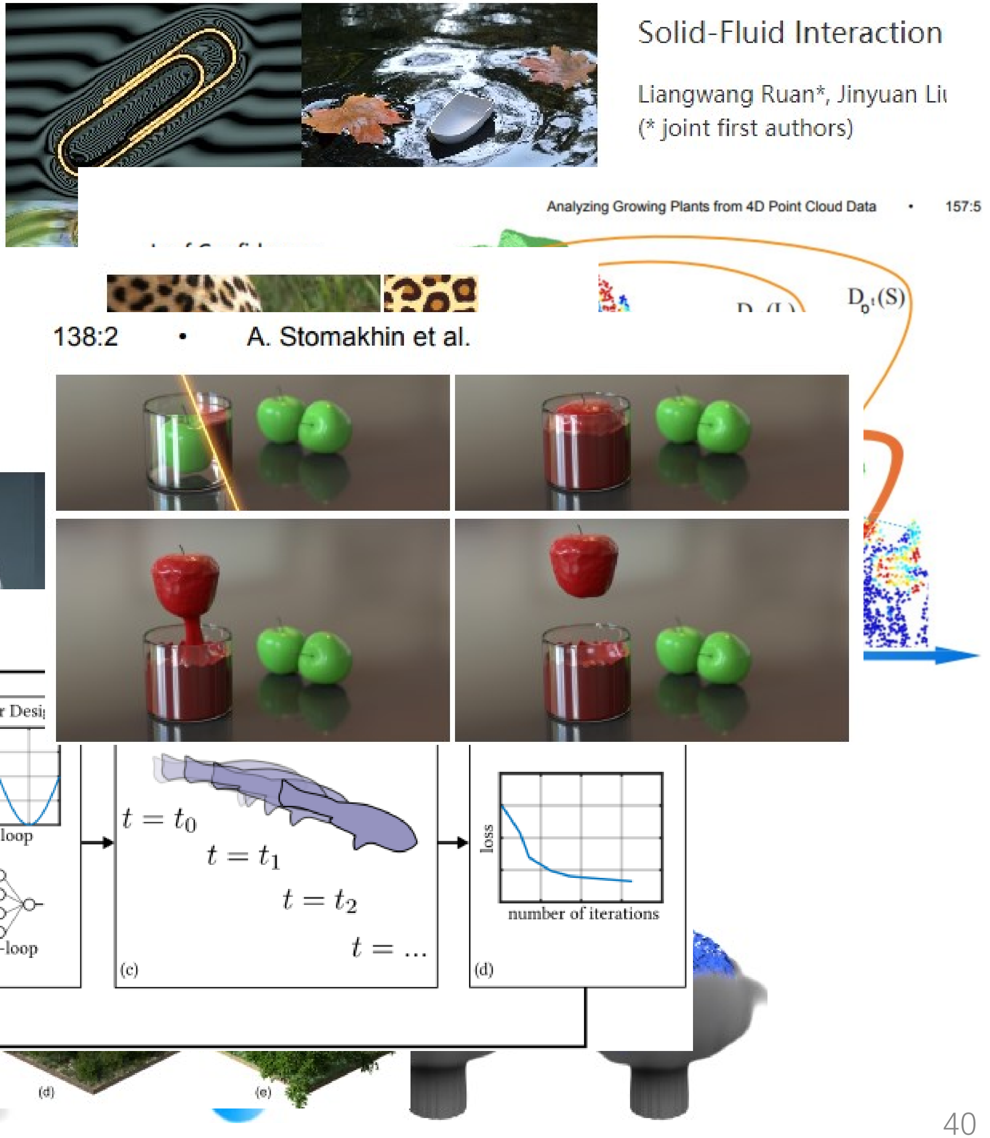
- Forward：
  New Phenomena
  - Coupling and Interaction
  - The Growth of Plants
  - Animal Development

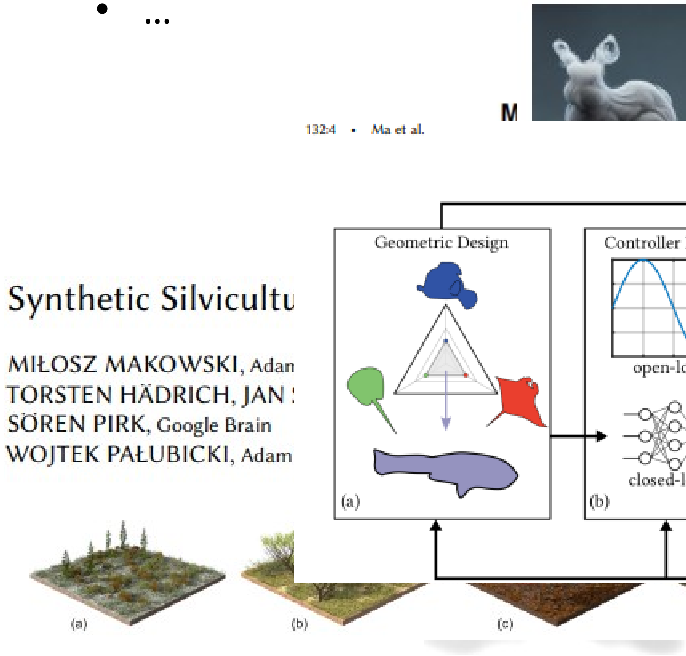
  Acceleration
  - Subspace-Physics
  - Multigrid Solver

  New Representation
  - Bubble
  - Monte Carlo-based Simulation

  Assets Generation
  - Landscape Generation

  …
- Inverse:
  - Elasticity in 3D Printing
  - Artistic Control
  - …
- Simulation + X (Modeling, Rendering, Animation,)
  - Underwater Swimmer Design
  - Phase Change
  - …





Summary

What We've Covered Today

- The building blocks of simulator:
  - spatial representation,
  - (linear/nonlinear)numerical solve,
  - temporal integration.
- Spring Mass System:
  - explicit Euler,
  - implicit Euler
- Interesting Simulation Topics

Online Resources

Physics-based Simulation:

Physical Laws (Elastic Potential) + *Geometric Constraints* (Volume Limiting, Collision and Contact Constraints)

- Force-Based Models:
  - Mass-Spring System: Tension, Shearing, Bending Springs 【what we learnt today!】
  - FEM: Continuum Mechanics, Hyperelastic Models 【 GAMES103-P7 】
  - Projective Dynamics 【 Talk from Tiantian Liu 】
  - Fluids: SPH 【 htmlCG_course】 APIC 【Games201】
  - Hybrid: MPM 【SIGGRAPH MPM Course】 …
- Constraints:
  - Position-Base Methods: 【 Ten Minute Physics 】
    - Kinematics w.o. Constraints + Constraint Projection
    - *No Dynamics!!*
  - Penalty-Force-Based Methods: 【 Again, GAMES103-P7 】
    - Mass-Spring / FEM + Strain/Area/Volume Limiting
    - IPC 【 GAMES103-P9 IPC】

Thanks!

Questions?

Mengyu Chu

mchu@pku.edu.cn

http://vcl.pku.edu.cn/index.html

VISUAL COMPUTING AND LEARNING LAB