Physics Simulation

课程内容规划（可能有些调整）：

主题	标题	预计课时
入门	引言	2h
	颜色	2h
	显示	2h
	画图	2h
	反走样	2h
	曲线	2h
图像处理	图像处理	2h
	图像编辑	2h
几何	几何表示	2h
	几何变换	2h
	几何处理	2h
	隐式几何	2h
	几何重建	2h
渲染	光照和着色	2h
	渲染管线	2h
	全局渲染	2h
	纹理映射	2h
	高级渲染	2h

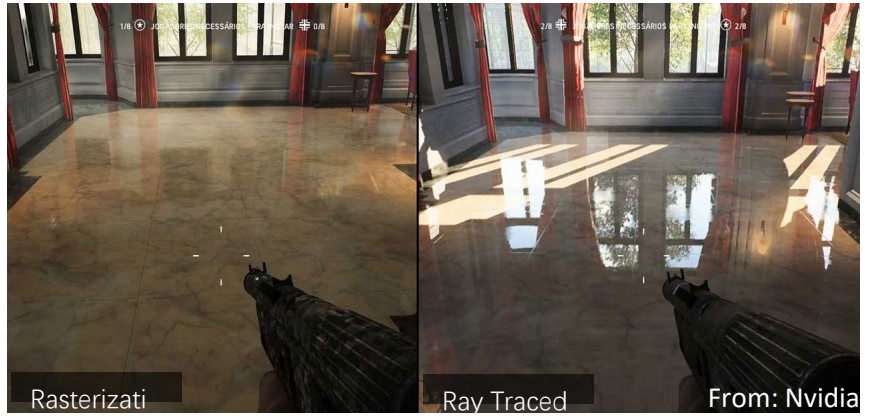
主题	标题	预计课时
仿真	物理模拟	2h
	动画原理	2h
可视化	可视化基础	2h
	科学数据可视化	2h
	信息数据可视化	2h
	高级可视化	2h
	交互式可视分析	2h
交互	经典交互技术	2h
	交互设计与评估	2h
	交互输入	2h
	三维空间交互	2h
	智能交互	2h
	虚拟/增强现实	2h
	自然交互	2h

Animation/Simulation

Graphics

Geometry

Animation/Simulation



Rendering

Manual

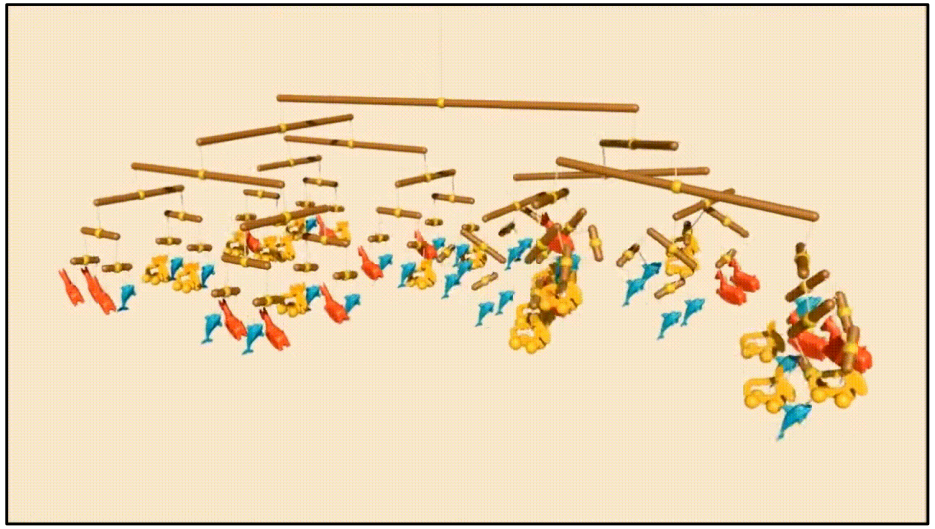
Procedural

Physically-based

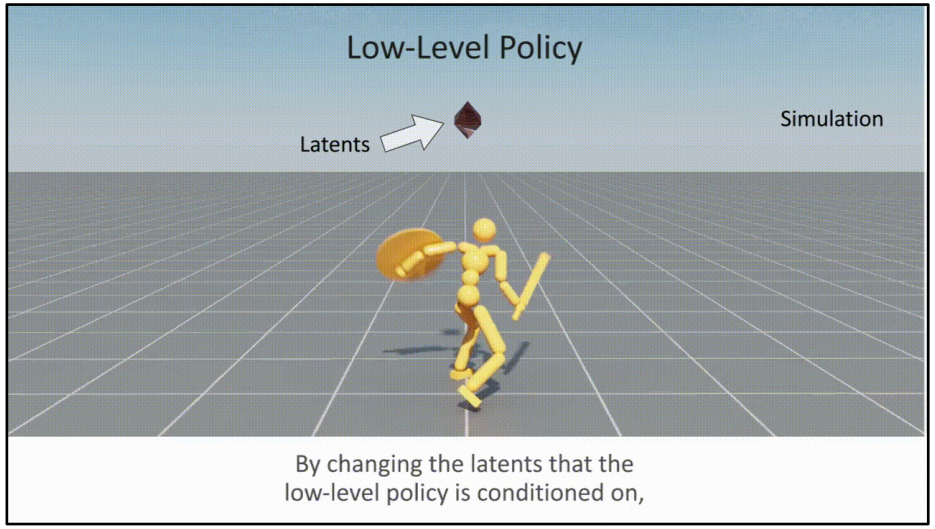
Data-Driven
(Motion Capture)

Simulating Everything

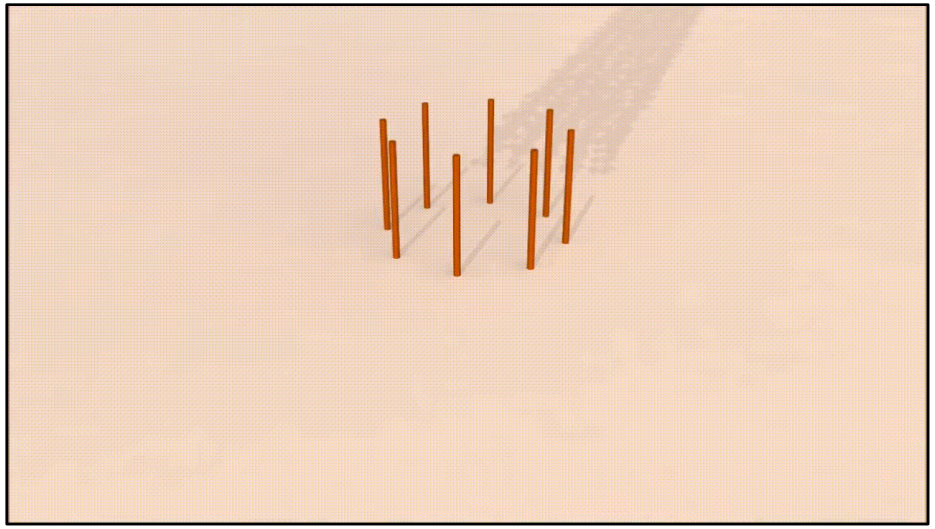
Rigid Body



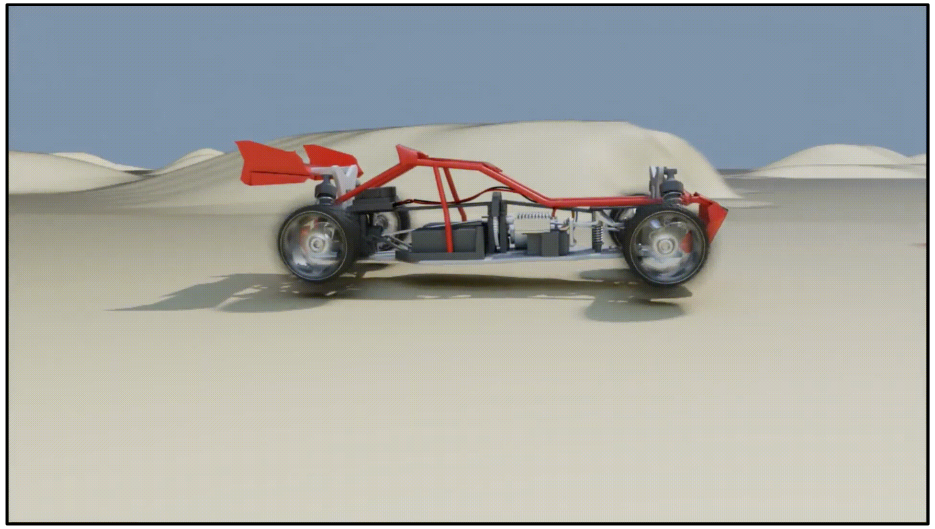
Deul et al. CAVW2014



Peng et al. SIGGRAPH2022

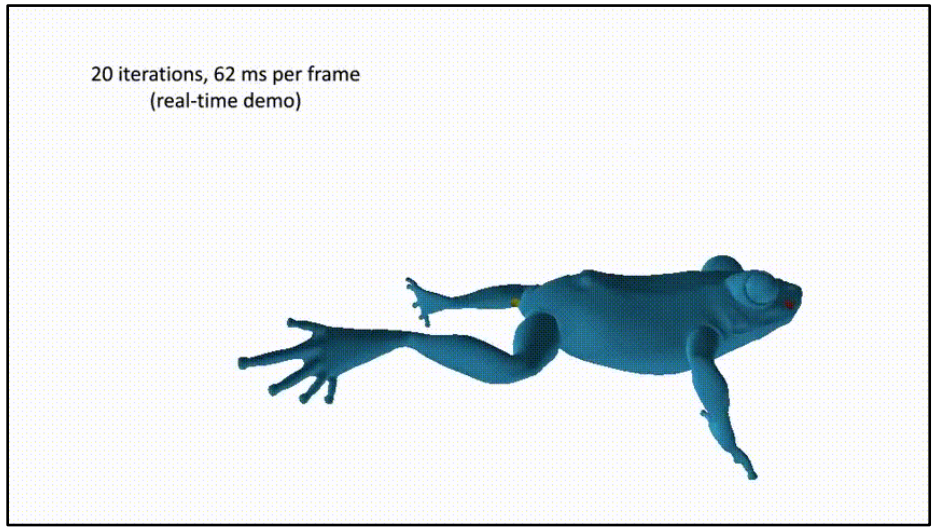


Deul et al. CAVW2014

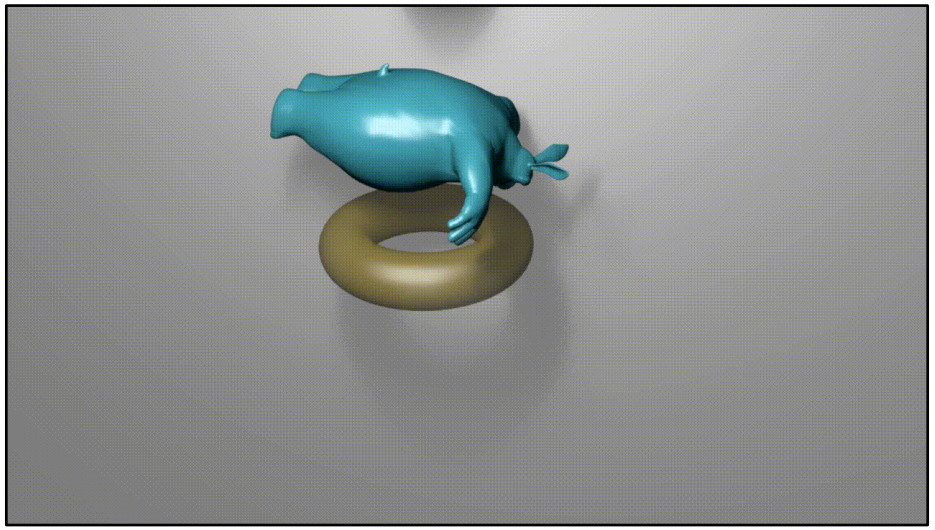


Muller et al. SCA2020

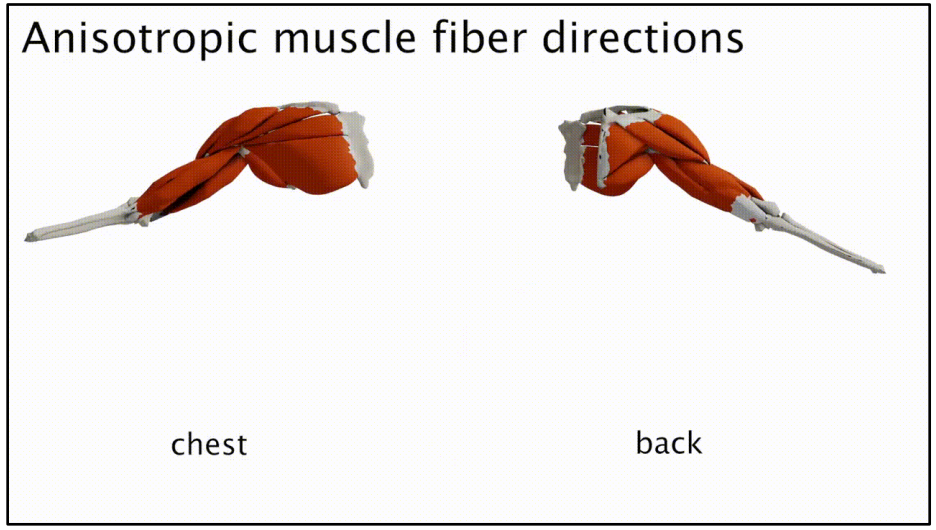
Deformable Object



Liu et al. SIGGRAPH Asia2013



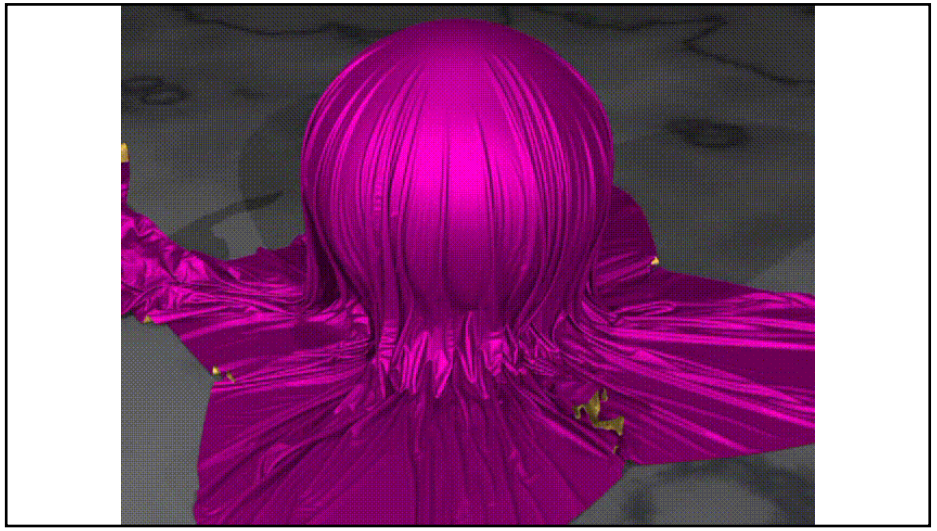
Liu et al. SIGGRAPH2017



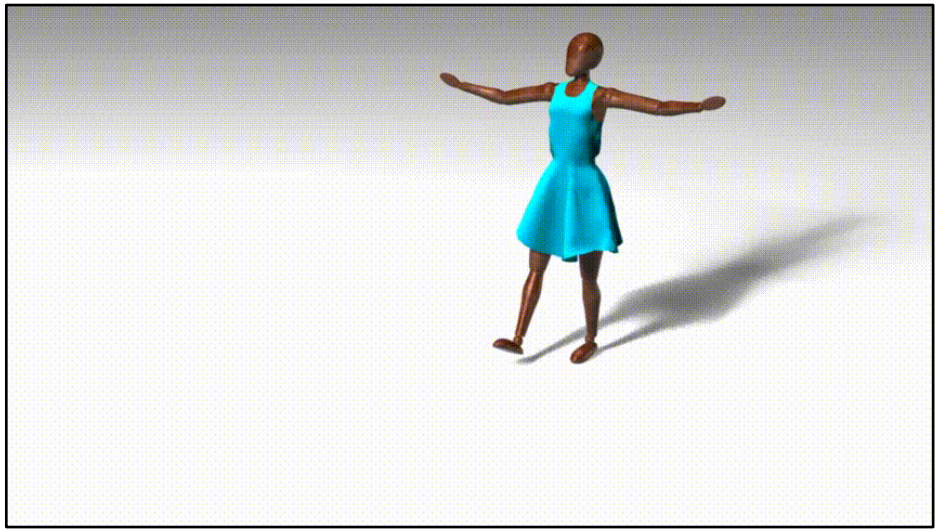
Modi et al. CGF2020

Liu et al. SIGGRAPH2013

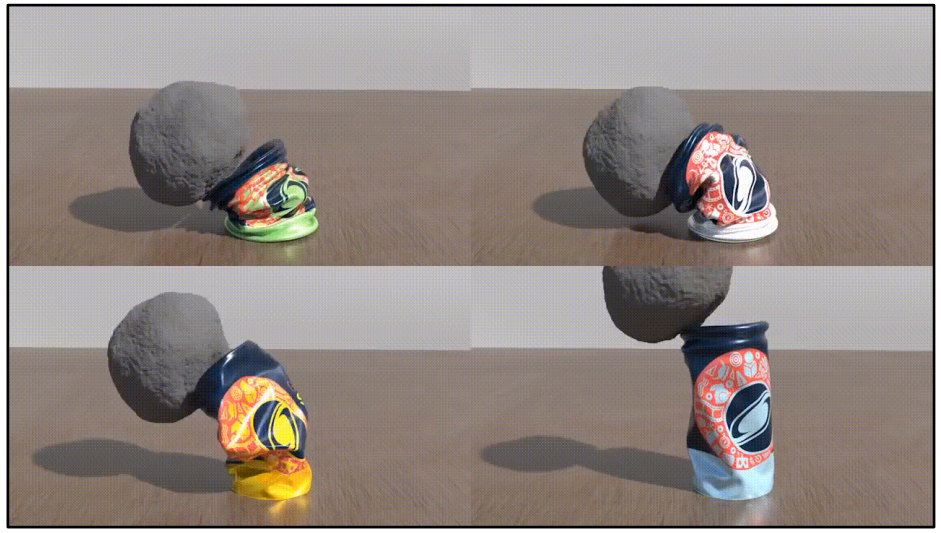
Thin Shell



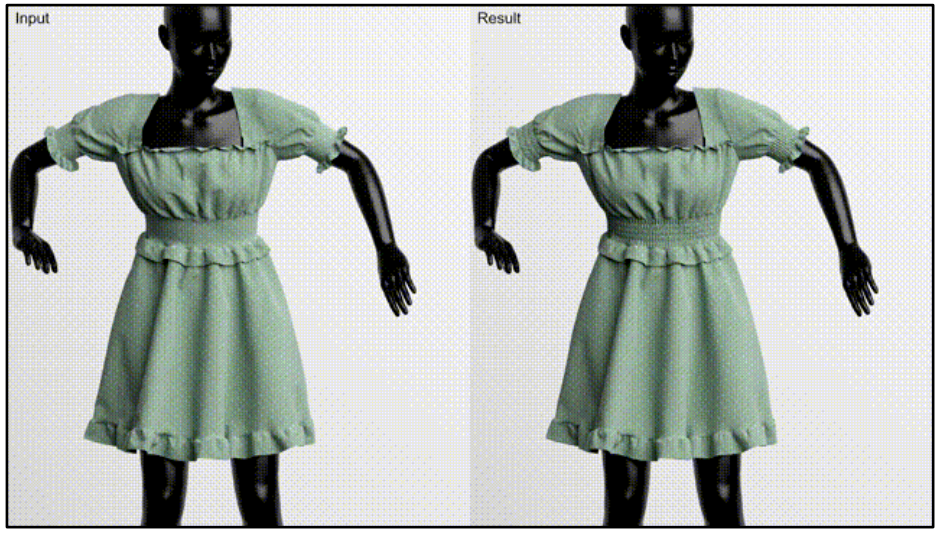
Selle et al. TVCG2015



Liu et al. SIGGRAPH Asia2013

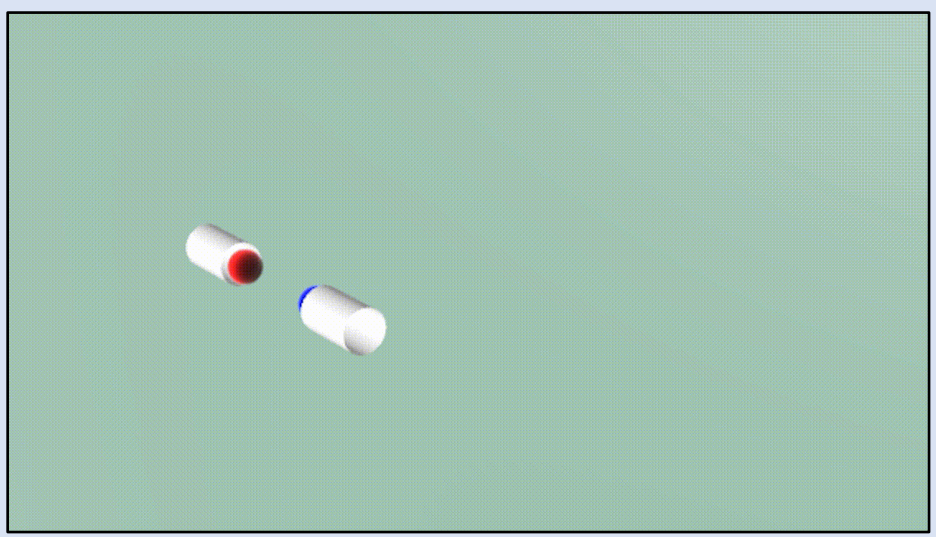


Guo et al. SIGGRAPH2018

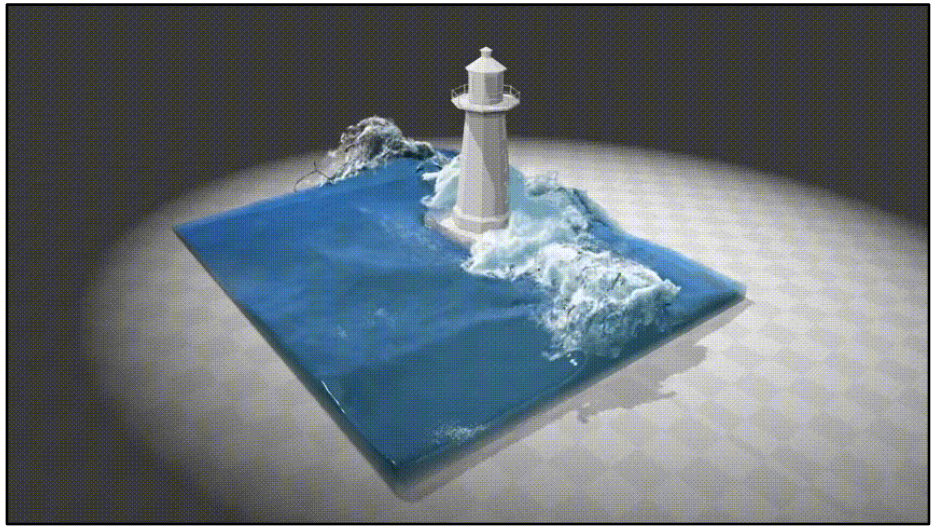


Huamin Wang SIGGRAPH2021

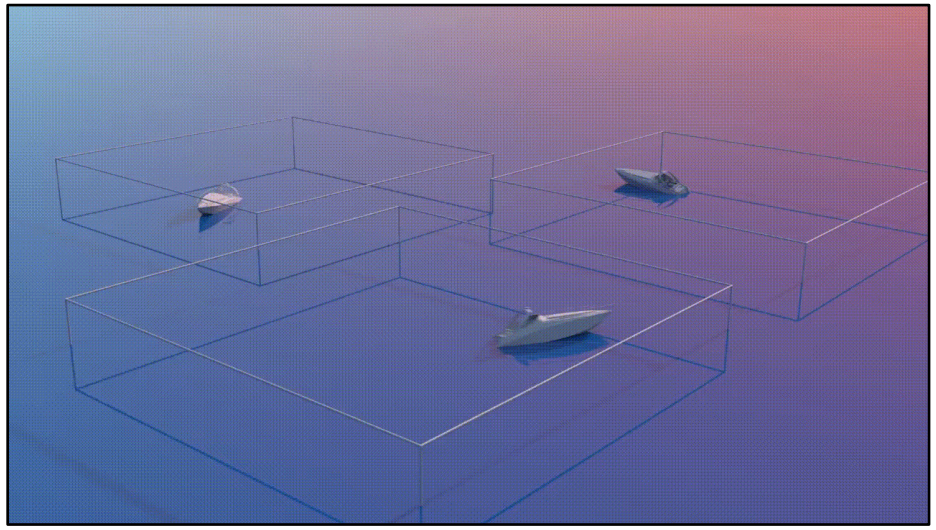
Fluid



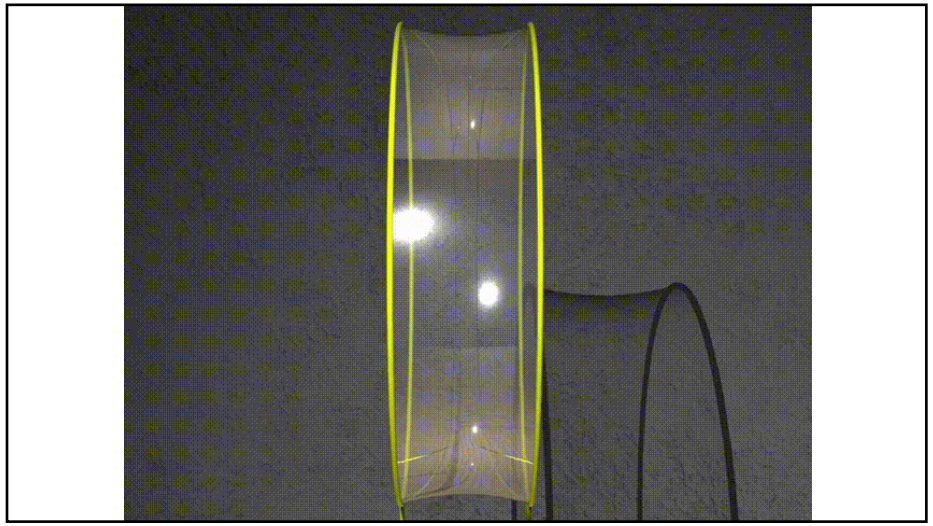
Qu et al. SIGGRAPH2019



Macklin et al. SIGGRAPH2013



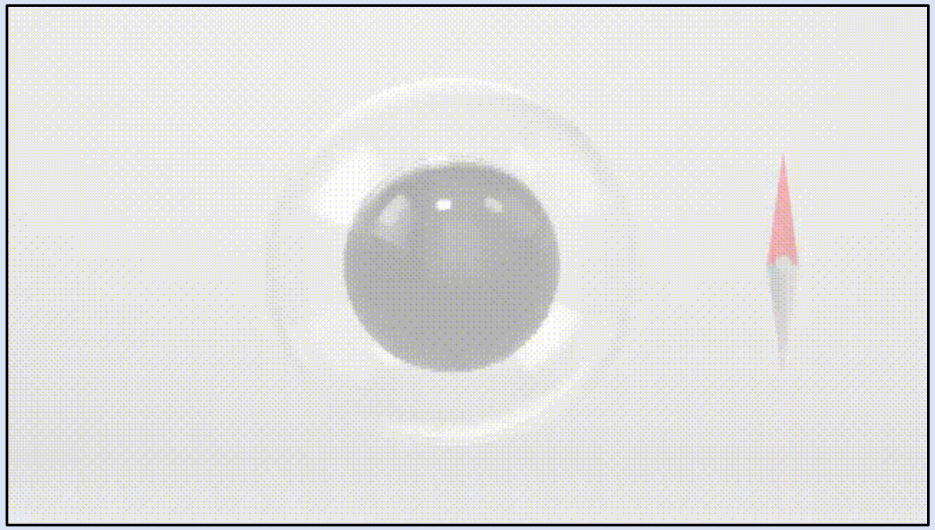
Huang et al. SIGGRAPH Asia2021



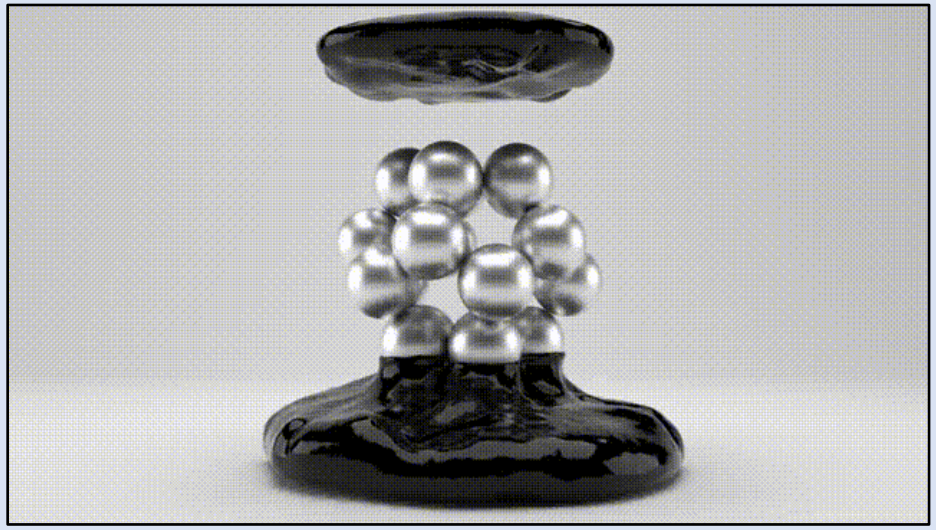
Zhu et al. SIGGRAPH2014

Anything else…

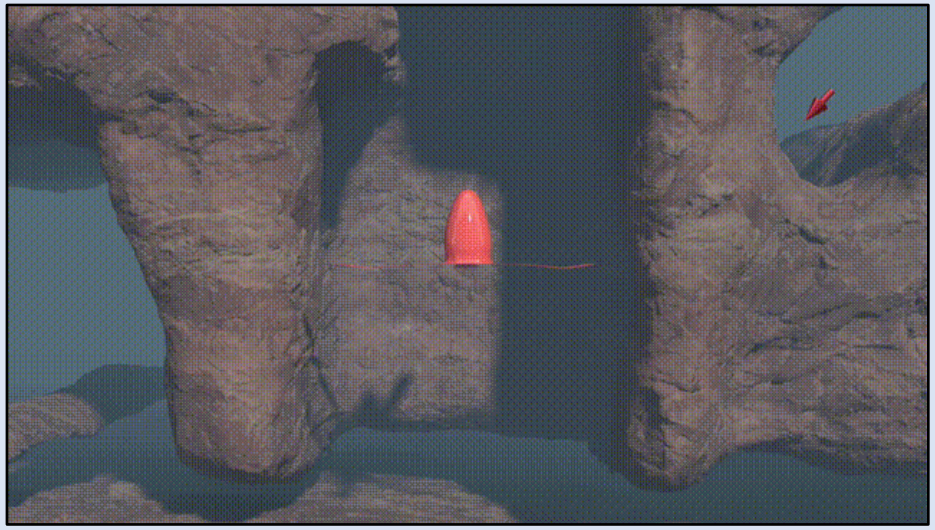
Ruan et al. SIGGRAPH2021



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
- http://www.physicsbasedanimation.com/
- SIGGRAPH(Asia) papers, courses…

Some Basics…

Physics Model

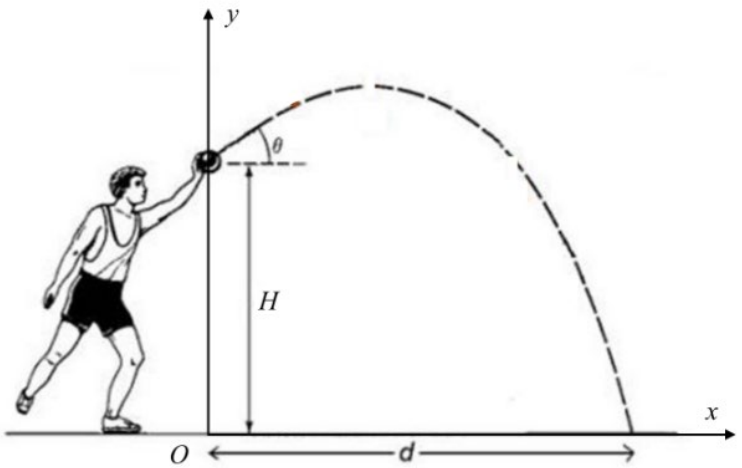
> The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are thus completely known, and the difficulty is only that the exact application of these laws leads to equations much too complicated to be soluble.
>
> Paul M. Dirac, 1929

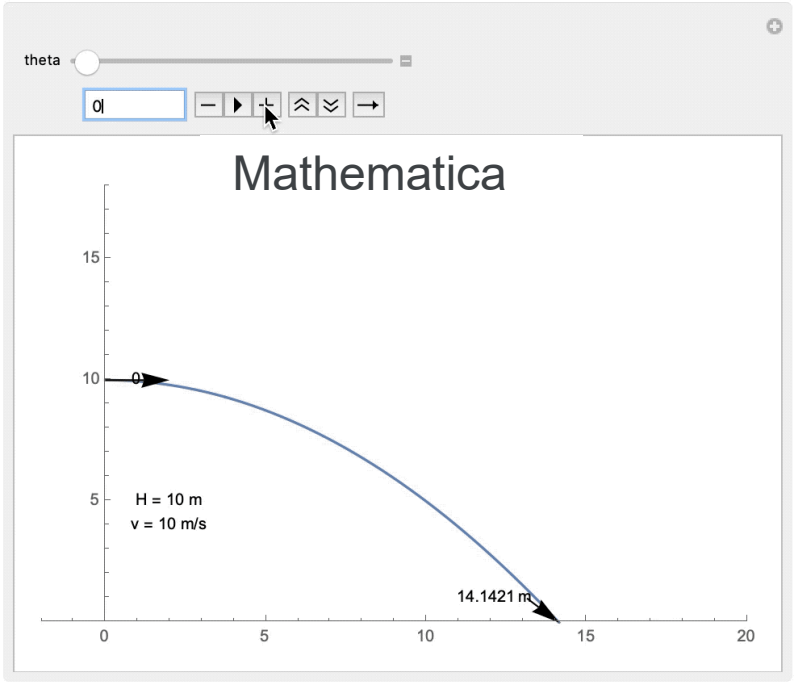
$$F = ma$$

- 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

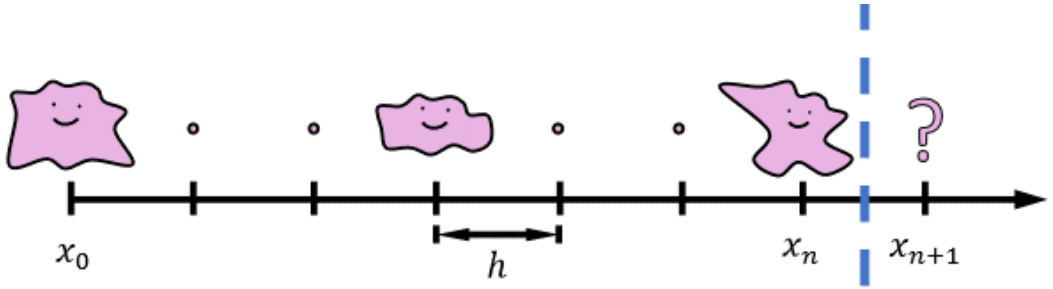
$$F = ma$$

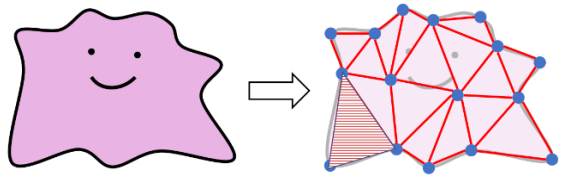
- For a particle:
  - $a = f(t),\ v = \int f(t)\,dt + v_0$. Easy to solve!
- A piece of material consists of **infinite number** of particles
  - When **analytical solutions** don't exist, we look for **numerical solutions**
  - We need **discretization, calculus, differential equations, tensor analysis,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

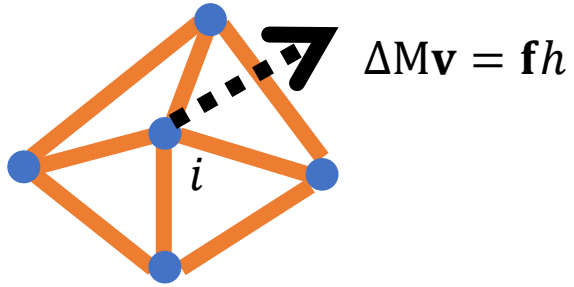
Three Building Blocks of Numerical Simulation

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





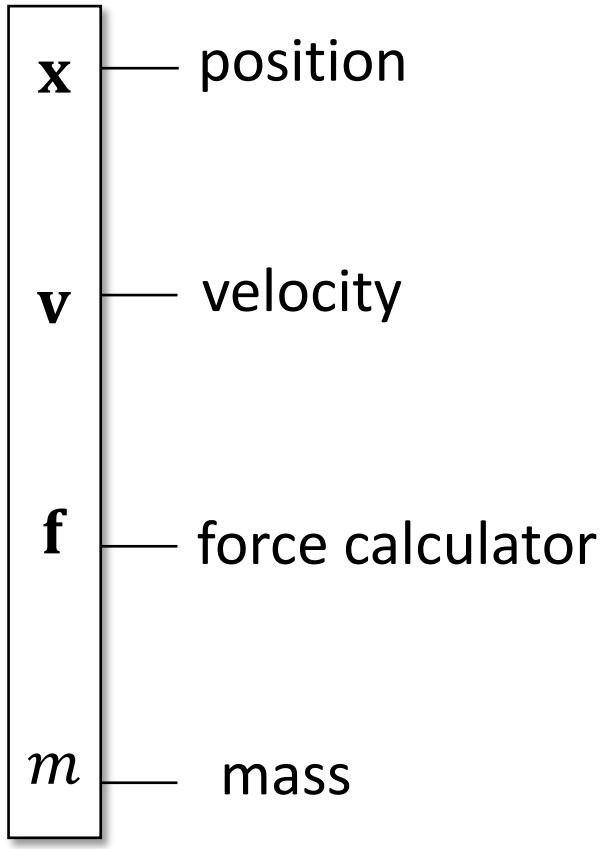


Questions:

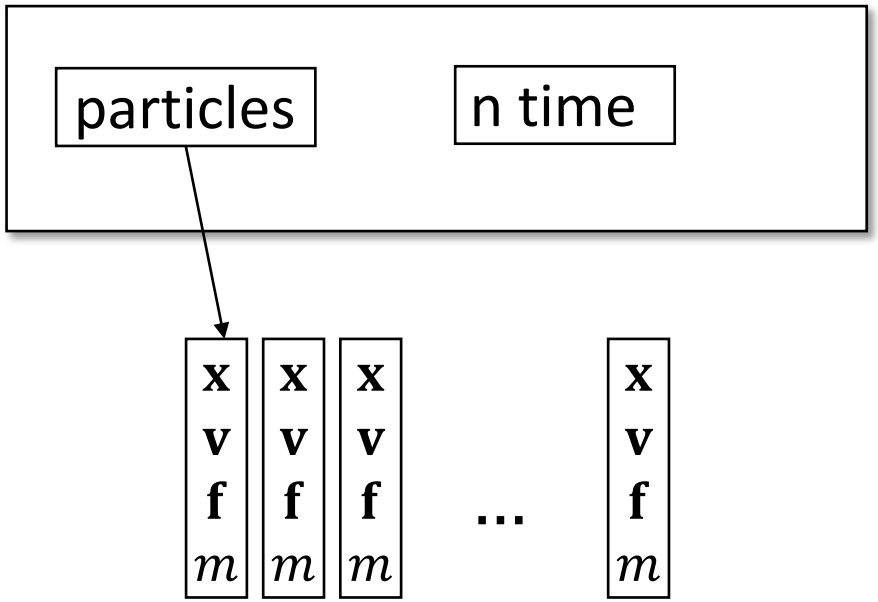
1. What is the **spatial representation**?
2. How to solve dynamics (**discretize momentum equations**) based on the representation?
3. How to perform kinematics via **time integration** (for a single time step)?

Your First Simulator: Spring-Mass System

Spatial Discretization: 1. Particle System



Particle Structure

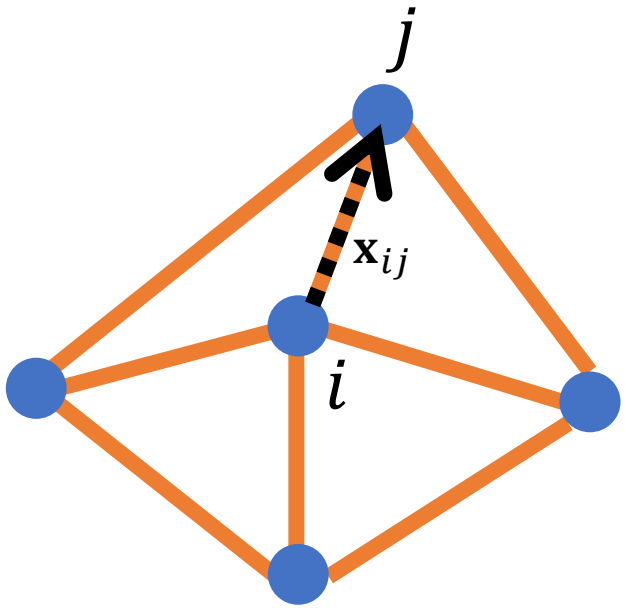


Particle System

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

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

integrate

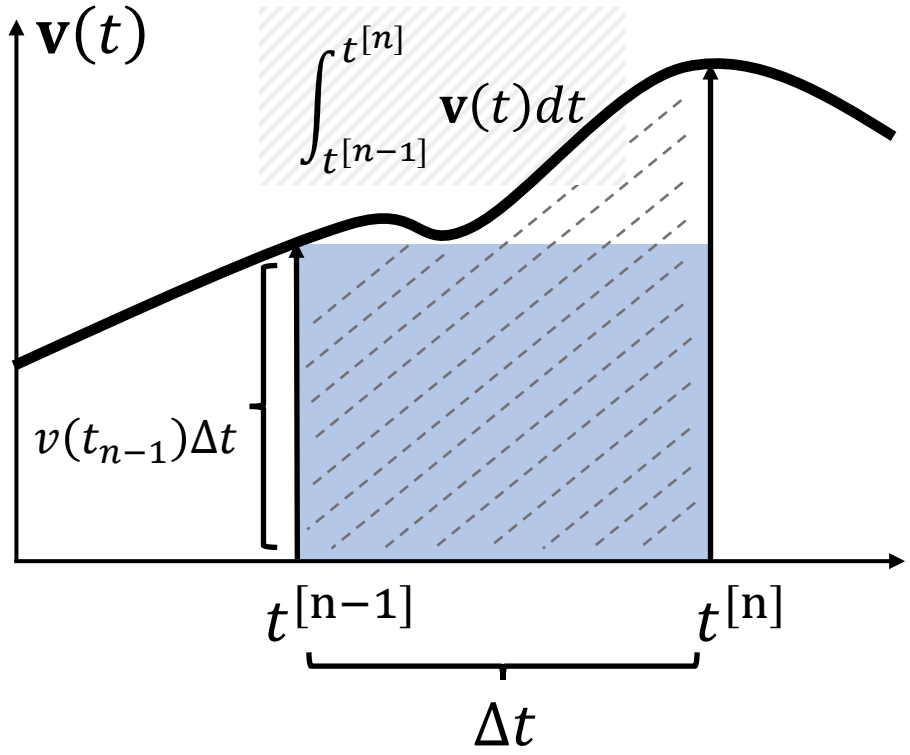
$$\mathbf{x}_\mathrm{p}(t_n) - \mathbf{x}_p(t_{n-1}) = \int_{t_{n-1}}^{t_n} \mathbf{v}_\mathrm{p}(t)\, dt$$

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

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

Temporal Discretization: Explicit Euler

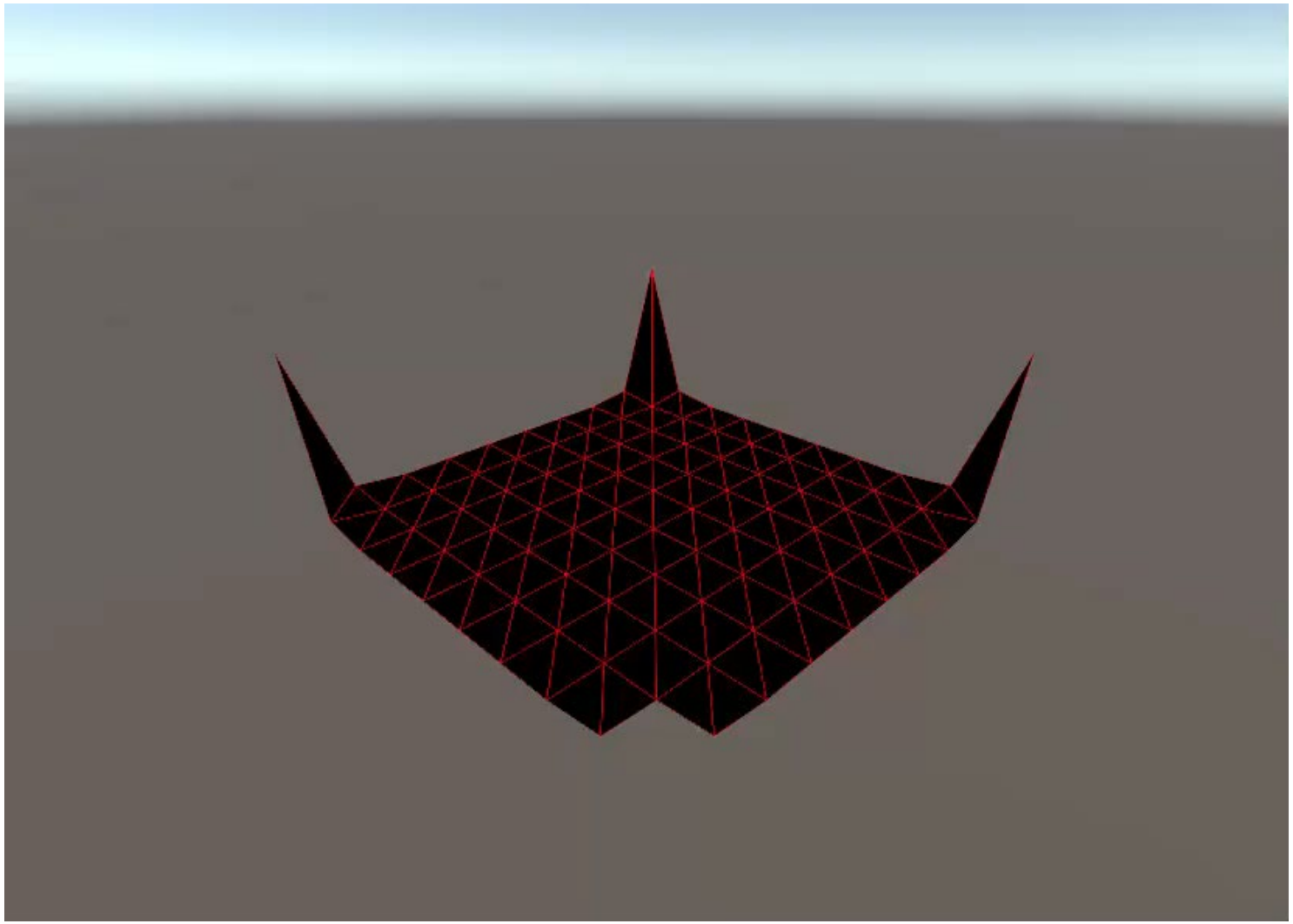
- We don't want integrations
- $\mathbf{x}(t_n) - \mathbf{x}(t_{n-1}) = \int_{t_{n-1}}^{t_n} \mathbf{v}\,(t)\,dt \approx \mathbf{v}(t_{n-1})\Delta t$
- $\mathbf{v}(t_n) - \mathbf{v}(t_{n-1}) = \frac{1}{m}\int_{t_{n-1}}^{t_n} \mathbf{f}(t)\,dt \approx \frac{1}{m}\mathbf{f}(t_{n-1})\Delta \mathrm{t}$
-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



Explicit Euler Simulator

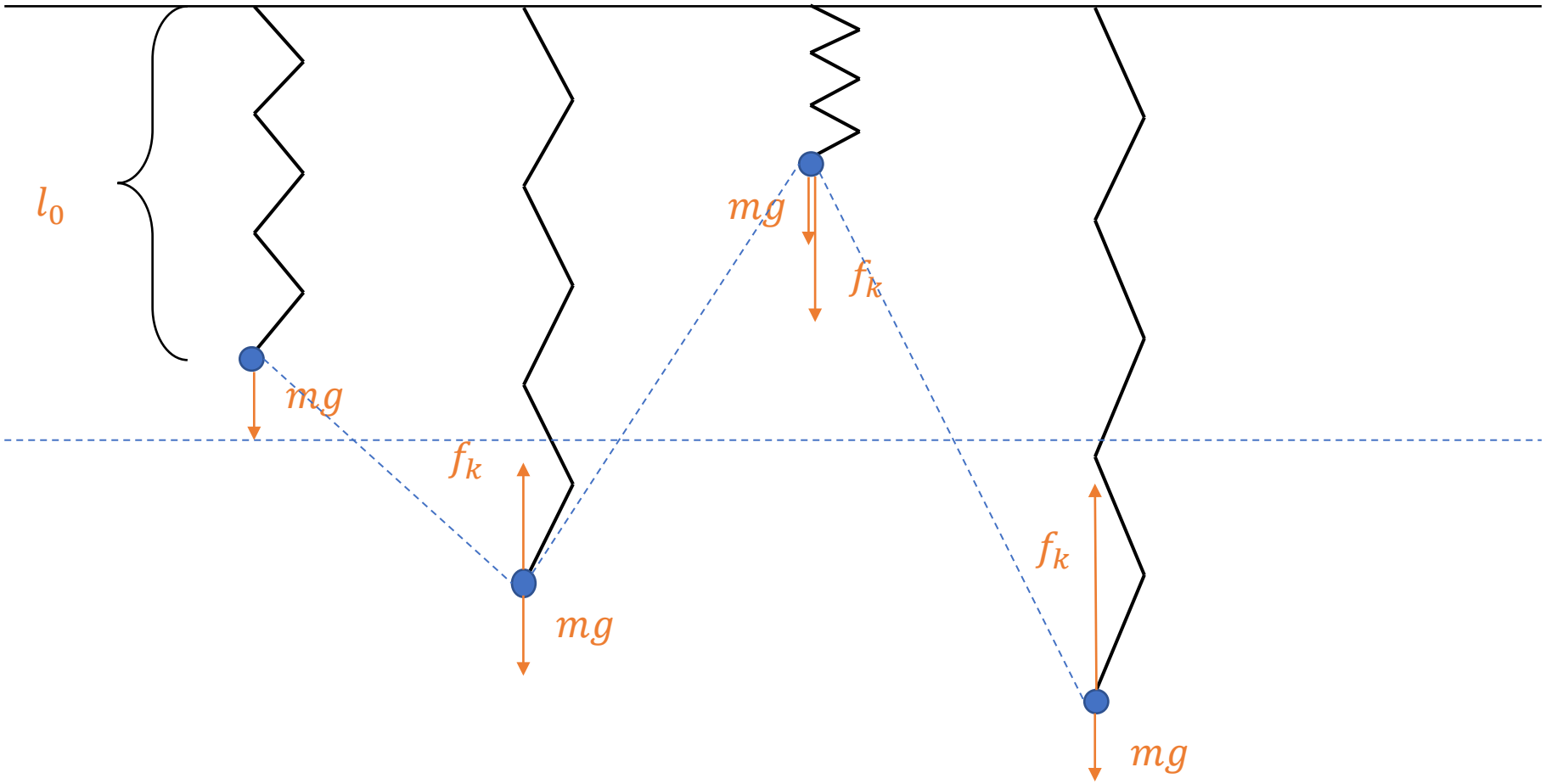
- At each step $t_n \rightarrow t_{n+1}$:
  - For each particle:
    - Compute $\mathbf{f}(t_n) = \Sigma \mathbf{f}_{ij}$ using current particle positions: $\mathbf{f}_{ij} = k\left(\left\|\mathbf{x}_{ij}\right\| - l_0\right)\frac{\mathbf{x_{ij}}}{\|\mathbf{x_{ij}}\|}$
    - Update its position $\mathbf{x}(t_{n+1})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

$$f(x+h) = f(x) + f'(x)h + \frac{f''(x)}{2}h^2 + \dots$$

$O(h^{n+1})$ *Truncation Error*

n^{th} *Order Approximation*

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

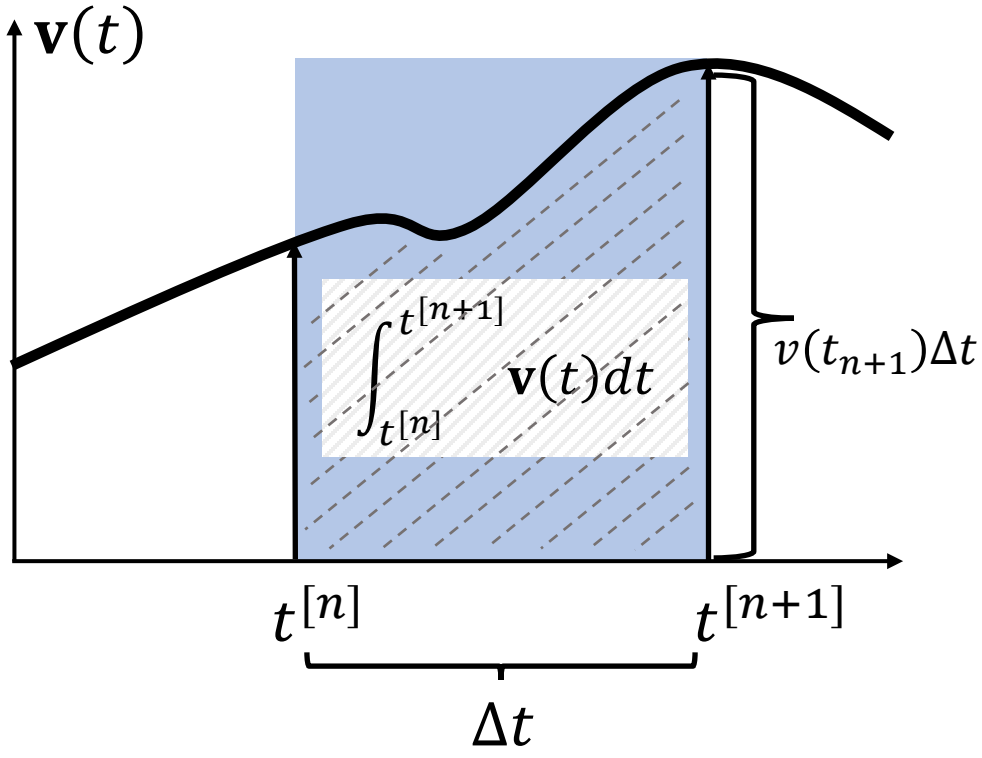
    Do *errors* accumulate? Is there a upper bound? (Error Propagation, Energy Preservation)
  - Convergence (Consistency):

    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

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

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

independent of $\mathbf{x}_{n+1}$ function of $\mathbf{x}_{n+1}$

Implicit Euler

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

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

This is because:

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

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

Applicable to every system
not just a mass-spring system

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

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

Implicit Euler

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

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

- Implicit Euler = energy minimization:

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

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

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

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

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

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

Numerical Solver

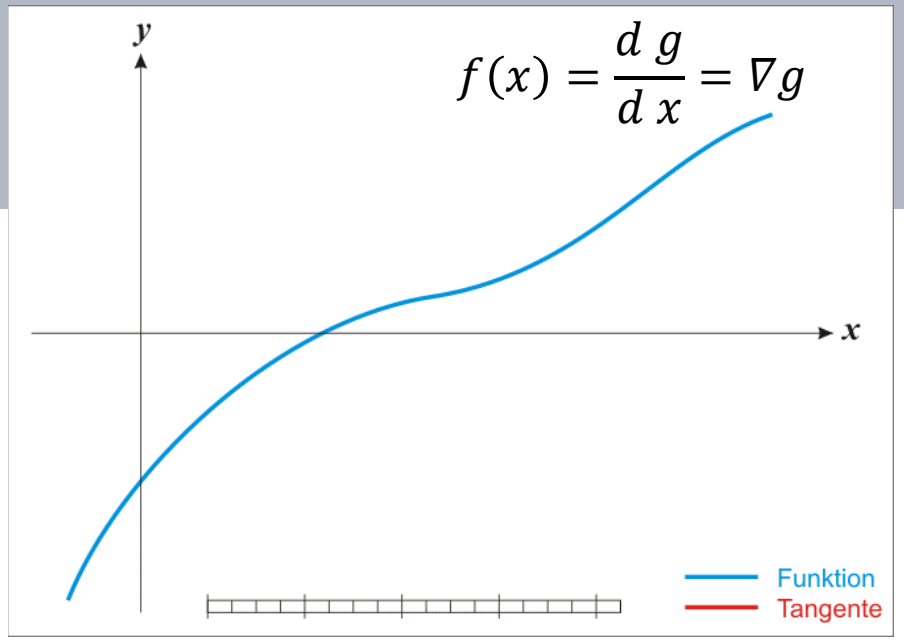
$$\mathbf{x}_{n+1} = argmin_{\mathbf{x}}\ g$$

- 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:

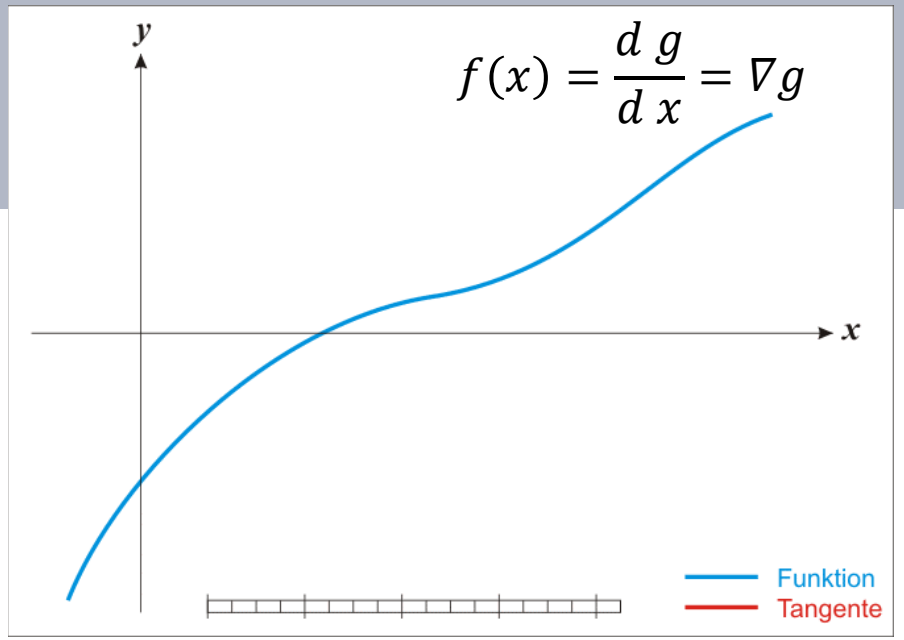
  Start from a guess $\mathbf{x}_1$,

  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

$\mathbf{x}^{(1)} := \mathbf{y}$;

$g(\mathbf{x}^{(1)}) := \texttt{evalObjective}(\mathbf{x}^{(1)})$

for $k = 1, \ldots,$ numIterations **do**

  $\nabla g(\mathbf{x}^{(k)}) := \texttt{evalGradient}(\mathbf{x}^{(k)})$

  $\nabla^2 g(\mathbf{x}^{(k)}) := \texttt{evalHessian}(\mathbf{x}^{(k)})$

  $\delta\mathbf{x}^{(k)} := -\nabla^2 g(\mathbf{x}^{(k)})^{-1}\nabla g(\mathbf{x}^{(k)})$

  $\alpha := 1/\beta$

  repeat

    $\alpha := \beta\alpha$

    $\mathbf{x}^{(k+1)} := \mathbf{x}^{(k)} + \alpha\delta\mathbf{x}^{(k)}$

    $g(\mathbf{x}^{(k+1)}) := \texttt{evalObjective}(\mathbf{x}^{(k+1)})$

  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:

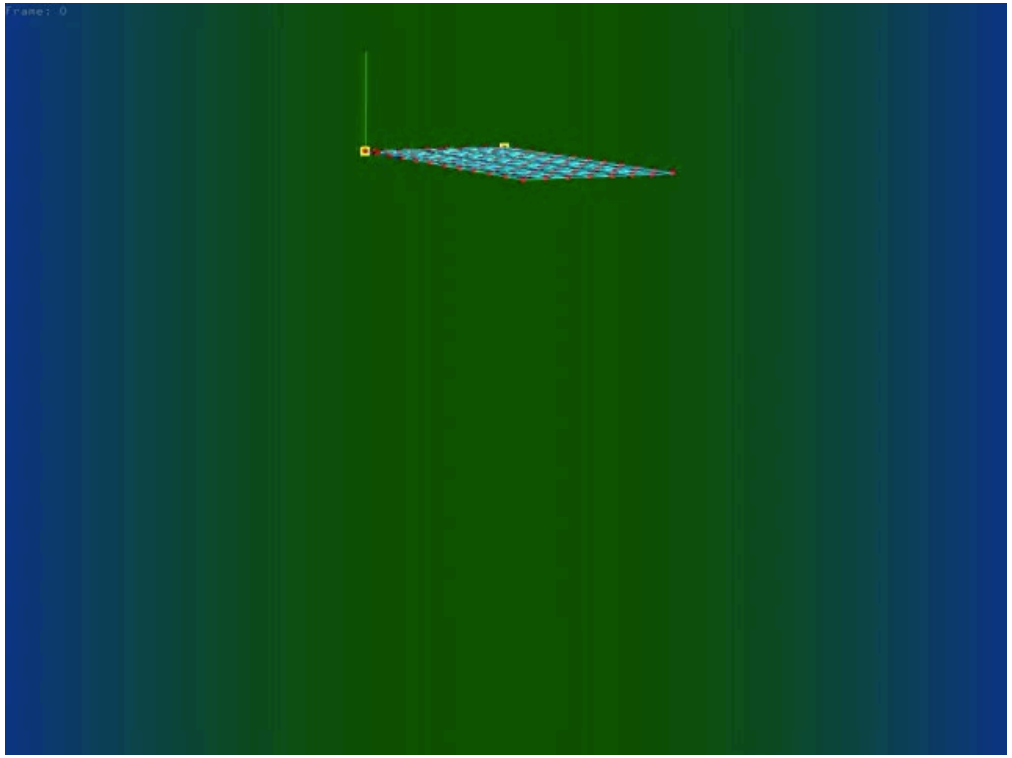
- Newton's methods
- Quasi-Newton Methods
- BFGS…

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

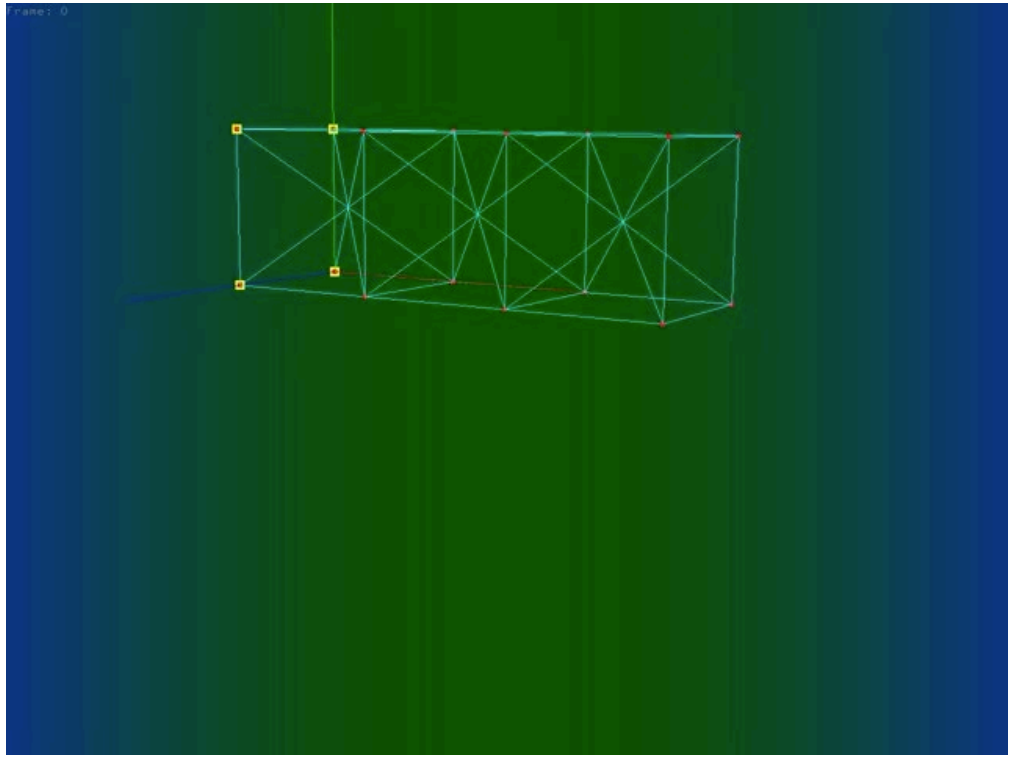
- Jacobi/Gauss-Seidel Iteration
- Conjugate Gradient
- Multigrid…

No one general optimal solver for all problems

Implicit Euler Results



cloth



rod

This is Spring-Mass System

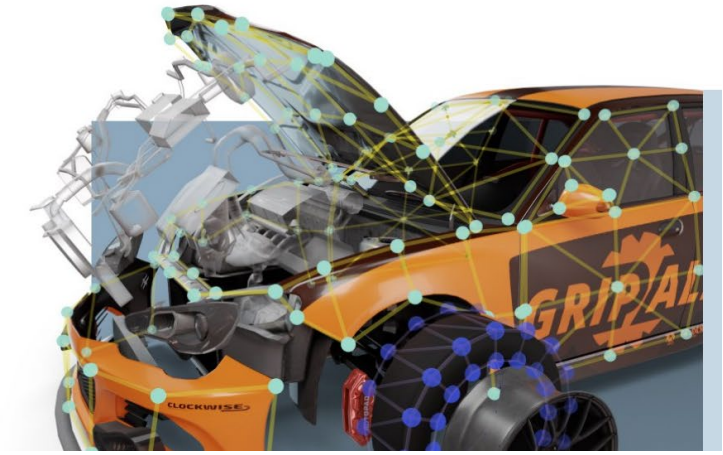
Huamin Wang SIGGRAPH2021

GPU-Based Simulation of Cloth Wrinkles at Submillimeter Levels

This is Also Spring Mass

Soft-body physics

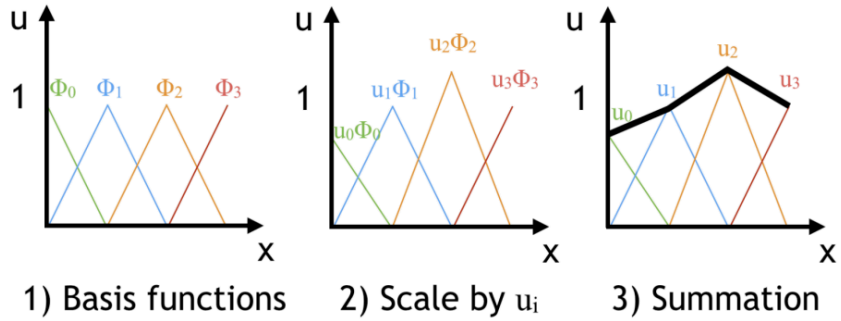
The BeamNG physics engine is at the core of the most detailed and authentic vehicle simulation you've ever seen in a game. Every component of a vehicle is simulated in real-time using nodes (mass points) and beams (springs). Crashes feel visceral, as the game uses an incredibly accurate damage model.



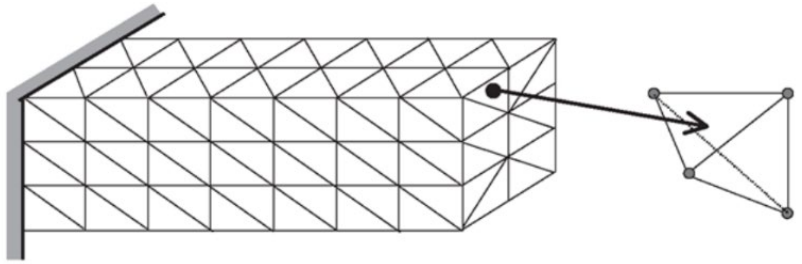
A More physical Way

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, ……)



1) Basis functions 2) Scale by u_i 3) Summation



What We've Covered Today

- The three building blocks of simulator: spatial discretization, temporal discretization, numerical solver
- Spring Mass System: explicit Euler, implicit Euler (and a simple intro to FEM)

Appendix: 1st-Order Derivatives

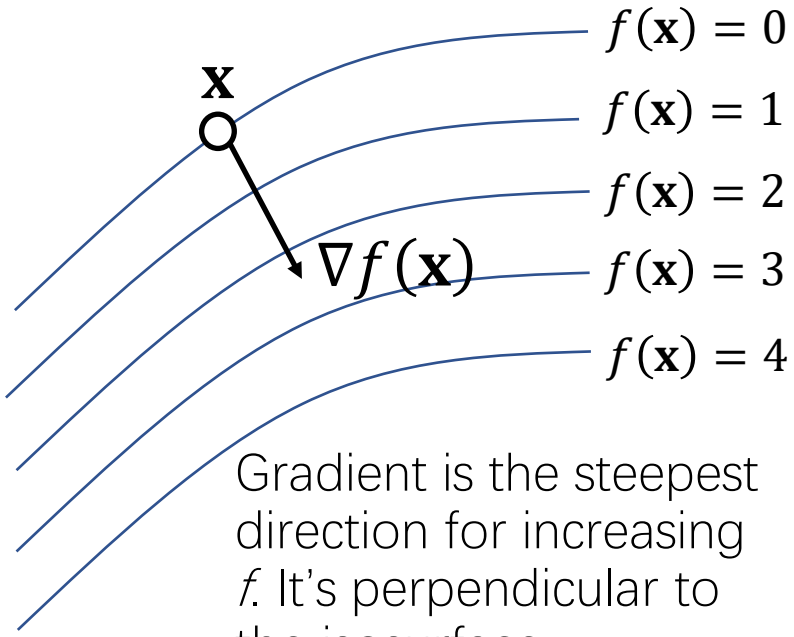
If $f(\mathbf{x}) \in \mathbf{R}$, then

$$df = \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial z}dz = \begin{bmatrix}\frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial z}\end{bmatrix}\begin{bmatrix}dx\\ dy\\ dz\end{bmatrix}.$$

$$\frac{\partial f}{\partial \mathbf{x}} = \begin{bmatrix}\frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial z}\end{bmatrix} \quad \text{or}$$

Gradient

$$\nabla f(\mathbf{x}) = \begin{bmatrix}\frac{\partial f}{\partial x}\\ \frac{\partial f}{\partial y}\\ \frac{\partial f}{\partial z}\end{bmatrix}$$



Gradient is the steepest direction for increasing *f*. It's perpendicular to the isosurface.

If $\mathbf{f}(\mathbf{x}) = \begin{bmatrix}f(\mathbf{x})\\ g(\mathbf{x})\\ h(\mathbf{x})\end{bmatrix} \in \mathbf{R}^3$, then:

Jacobian

$$\mathbf{J}(\mathbf{x}) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix}\frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial z}\\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} & \frac{\partial g}{\partial z}\\ \frac{\partial h}{\partial x} & \frac{\partial h}{\partial y} & \frac{\partial h}{\partial z}\end{bmatrix}$$

Appendix : 1st&2nd -Order Derivatives

If $f(\mathbf{x}) \in \mathbf{R}$, then:

Gradient

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix}$$

Hessian

$$\mathbf{H} = \mathbf{J}(\nabla f(\mathbf{x})) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial x \partial z} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} & \frac{\partial^2 f}{\partial y \partial z} \\ \frac{\partial^2 f}{\partial x \partial z} & \frac{\partial^2 f}{\partial y \partial z} & \frac{\partial^2 f}{\partial z^2} \end{bmatrix}$$

$$\frac{\partial \|\mathbf{x}\|}{\partial \mathbf{x}} = \frac{\partial (\mathbf{x}^{\mathrm{T}}\mathbf{x})^{1/2}}{\partial \mathbf{x}} = \frac{1}{2}(\mathbf{x}^{\mathrm{T}}\mathbf{x})^{-1/2}\frac{\partial (\mathbf{x}^{\mathrm{T}}\mathbf{x})}{\partial \mathbf{x}} = \frac{1}{2\|\mathbf{x}\|} 2\mathbf{x}^{\mathrm{T}} = \frac{\mathbf{x}^{\mathrm{T}}}{\|\mathbf{x}\|}$$

Example: A Spring

Rest length L, $\mathbf{x}_{01} = \mathbf{x}_0 - \mathbf{x}_1$

Energy: $$E(\mathbf{x}) = \frac{k}{2}(\|\mathbf{x}_{01}\| - L)^2$$

Force: $$\mathbf{f}(\mathbf{x}) = -\nabla E(\mathbf{x}) = \begin{bmatrix} -\nabla_0 E(\mathbf{x}) \\ -\nabla_1 E(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \mathbf{f}_e \\ -\mathbf{f}_e \end{bmatrix}$$

$$\mathbf{f}_e = -k(\|\mathbf{x}_{01}\| - L)\frac{\mathbf{x}_{01}}{\|\mathbf{x}_{01}\|}$$

Tangent stiffness: $$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 E}{\partial \mathbf{x}_0^2} & \frac{\partial^2 E}{\partial \mathbf{x}_0 \partial \mathbf{x}_1} \\ \frac{\partial^2 E}{\partial \mathbf{x}_0 \partial \mathbf{x}_1} & \frac{\partial^2 E}{\partial \mathbf{x}_1^2} \end{bmatrix} = \begin{bmatrix} \mathbf{H}_e & -\mathbf{H}_e \\ -\mathbf{H}_e & \mathbf{H}_e \end{bmatrix}$$

$$\mathbf{H}_e = k\frac{\mathbf{x}_{01}\mathbf{x}_{01}^{\mathrm{T}}}{\|\mathbf{x}_{01}\|^2} + k\left(1 - \frac{L}{\|\mathbf{x}_{01}\|}\right)\left(\mathbf{I} - \frac{\mathbf{x}_{01}\mathbf{x}_{01}^{\mathrm{T}}}{\|\mathbf{x}_{01}\|^2}\right)$$

Appendix: Spring-Mass System

Specifically to simulation, we have: $g(\mathbf{x}) = \frac{1}{2h^2}\|\mathbf{x} - \boldsymbol{y}\|_{\mathbf{M}}^2 + E(\mathbf{x})$

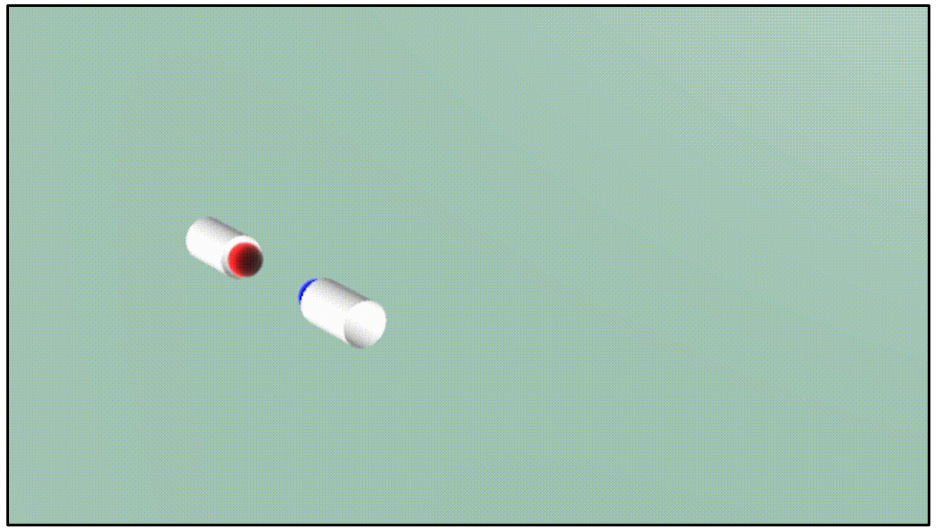
$$\nabla g\left(\mathbf{x}^{(k)}\right) = \frac{1}{h^2}\mathbf{M} - \left(\mathbf{x}^{(k)} - \boldsymbol{y}\right)\mathbf{f}\left(\mathbf{x}^{(k)}\right) \qquad \frac{\partial^2 g\left(\mathbf{x}^{(k)}\right)}{\partial \mathbf{x}^2} = \frac{1}{h^2}\mathbf{M} + \mathbf{H}\left(\mathbf{x}^{(k)}\right)$$

$$\mathbf{H}(\mathbf{x}) = \sum_{e=\{i,j\}} \begin{bmatrix} \frac{\partial^2 E}{\partial \mathbf{x}_i^2} & \frac{\partial^2 E}{\partial \mathbf{x}_i \partial \mathbf{x}_j} \\ \frac{\partial^2 E}{\partial \mathbf{x}_i \partial \mathbf{x}_j} & \frac{\partial^2 E}{\partial \mathbf{x}_j^2} \end{bmatrix} = \sum_{e=\{i,j\}} \begin{bmatrix} \mathbf{H}_e & -\mathbf{H}_e \\ -\mathbf{H}_e & \mathbf{H}_e \end{bmatrix} \begin{bmatrix} & \end{bmatrix}$$

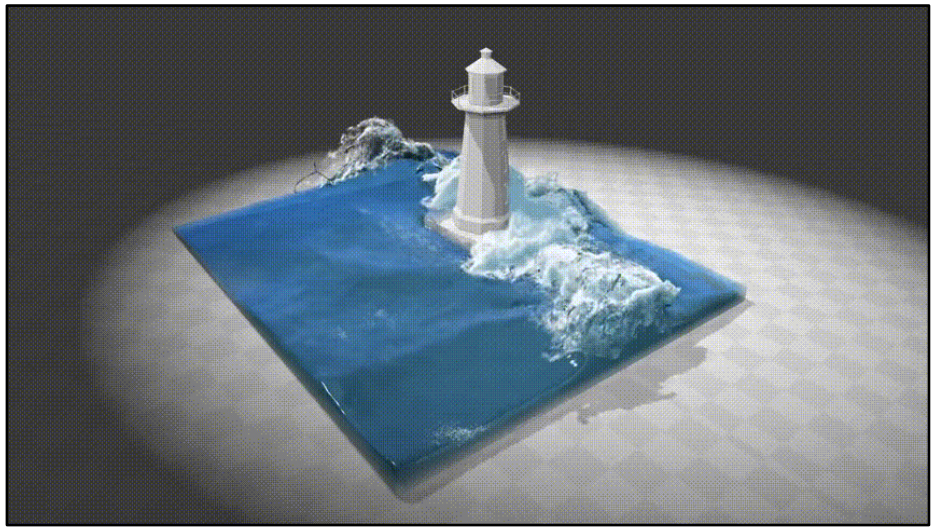
$$\mathbf{H}_e = k\frac{\mathbf{x}_{ij}\mathbf{x}_{ij}^{\mathrm{T}}}{\|\mathbf{x}_{ij}\|^2} + k\left(1 - \frac{L}{\|\mathbf{x}_{ij}\|}\right)\left(\mathbf{I} - \frac{\mathbf{x}_{ij}\mathbf{x}_{ij}^{\mathrm{T}}}{\|\mathbf{x}_{ij}\|^2}\right)$$

Fluid Simulation

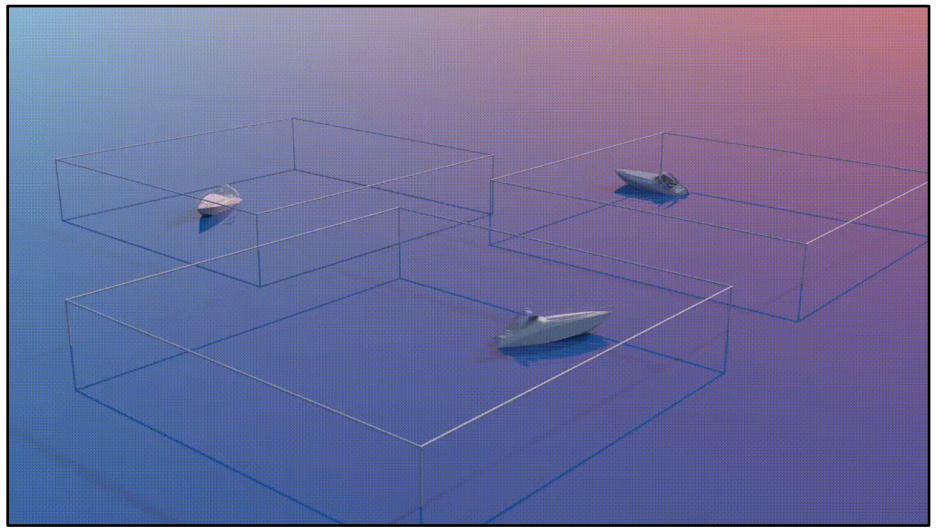
Fluid



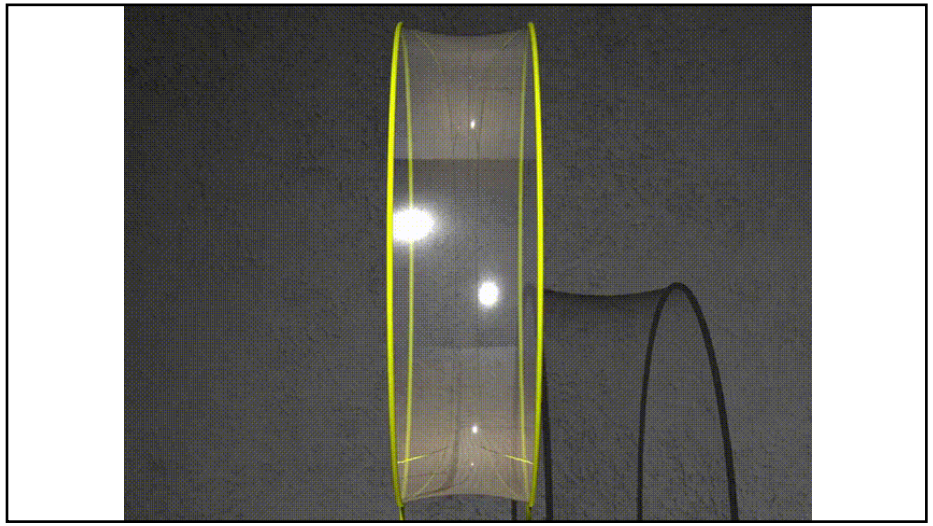
Qu et al. SIGGRAPH 2019



Macklin et al. SIGGRAPH 2013



Huang et al. SIGGRAPH Asia 2021



Zhu et al. SIGGRAPH 2014

Physics model

- What's the dominant equation for fluids? $m\boldsymbol{a} = \boldsymbol{F}$
- Navier-Stokes Equation:

$$\rho \frac{D\vec{v}}{Dt} = -\nabla p + \rho\vec{g} + \underline{\mu\nabla^2\vec{v}}$$

Viscosity, usually safely dropped

- The left-hand side is in **Lagrangian** perspective.

  The right-hand side is in **Eulerian** perspective.

Two viewpoints

Lagrangian

Eulerian

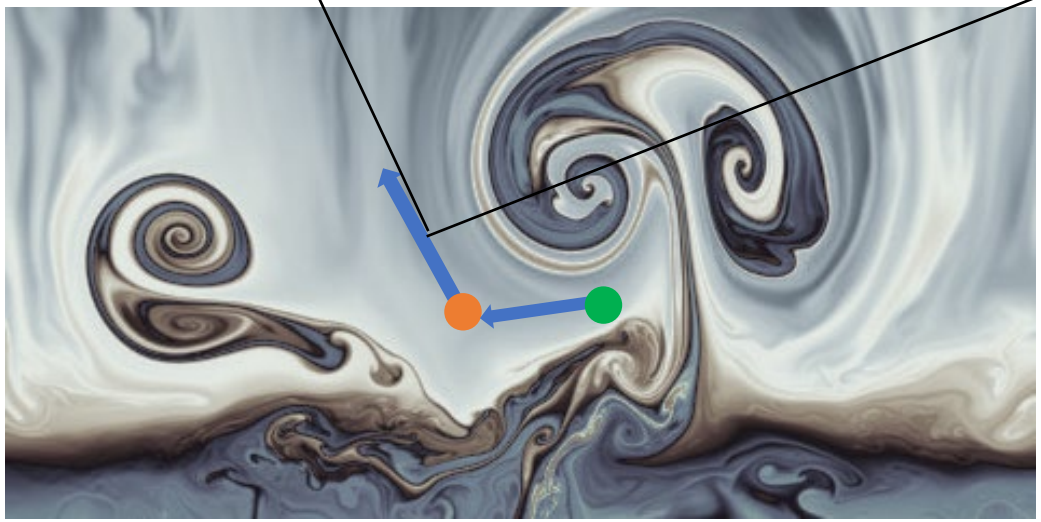
Difference on Computing Accelerations

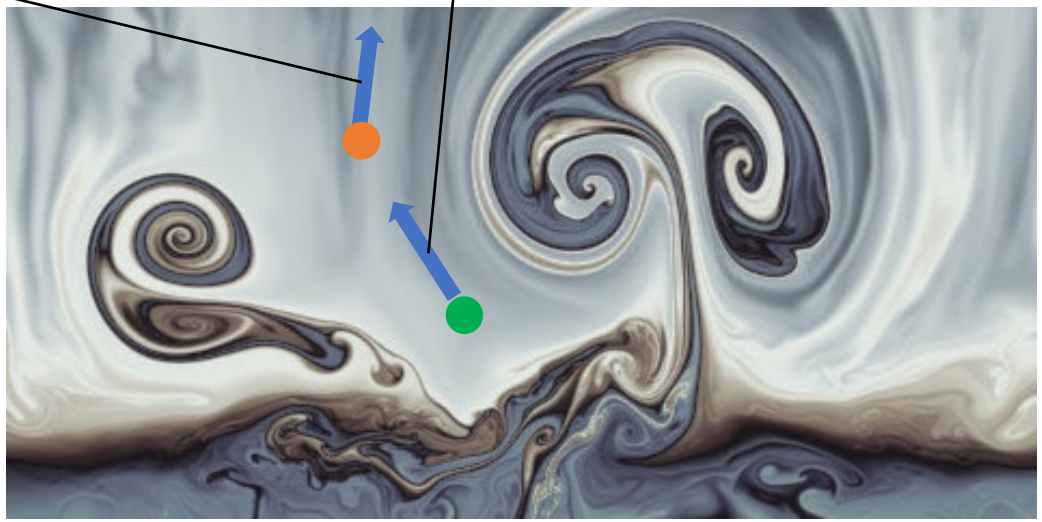
$\frac{D\boldsymbol{v}}{Dt}$

Lagrangian Derivative

$\frac{\partial \boldsymbol{v}}{\partial t}$

Eulerian Derivative





t_n

t_{n+1}

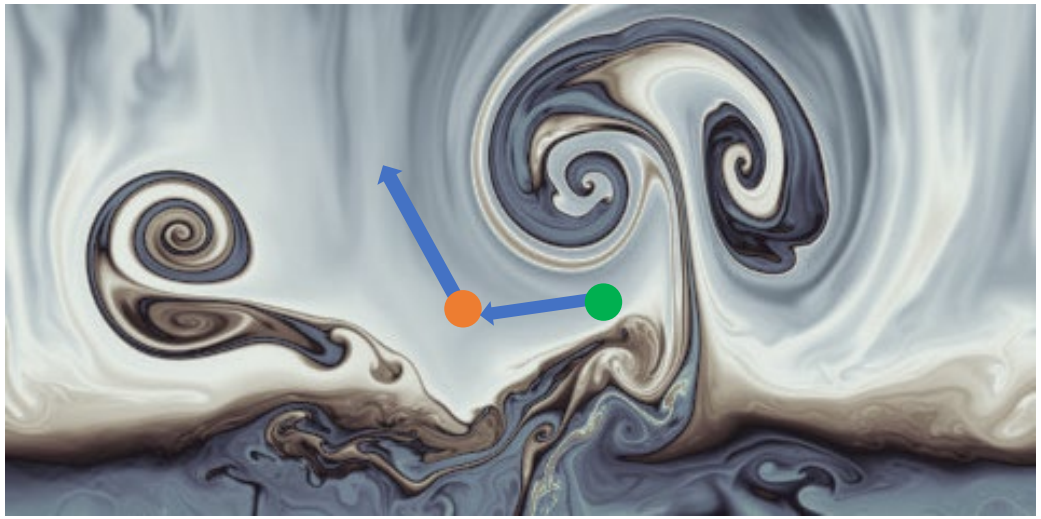
Difference on Computing Accelerations

$$\frac{D\boldsymbol{v}}{Dt} = \left(\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}} \mathrm{D}\boldsymbol{x} + \frac{\partial \boldsymbol{v}}{\partial t} Dt\right)/Dt = \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}} \frac{D\boldsymbol{x}}{Dt} + \frac{\partial \boldsymbol{v}}{\partial t} = \frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{v} \cdot \nabla \boldsymbol{v}$$

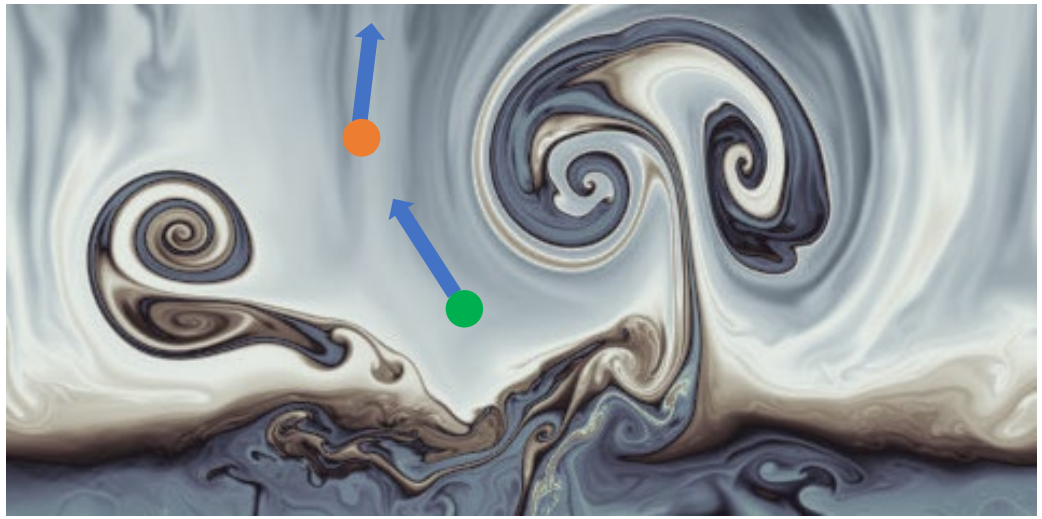
Lagrangian Derivative

Eulerian Derivative

$$\frac{D\boldsymbol{q}}{Dt} = \frac{\partial \boldsymbol{q}}{\partial t} + \boldsymbol{v} \cdot \nabla \boldsymbol{q}$$



t_n



t_{n+1}

Physics model

- What's the dominant equation for fluids?
- Navier-Stokes Equation:

$$\underbrace{\rho \frac{D\vec{v}}{Dt}}_{ma} = \underbrace{-\nabla p + \rho \vec{g} + \mu \nabla^2 \vec{v}}_{F}$$

- The left-hand side is in **Lagrangian** perspective.

  The right-hand side is in **Eulerian** perspective.

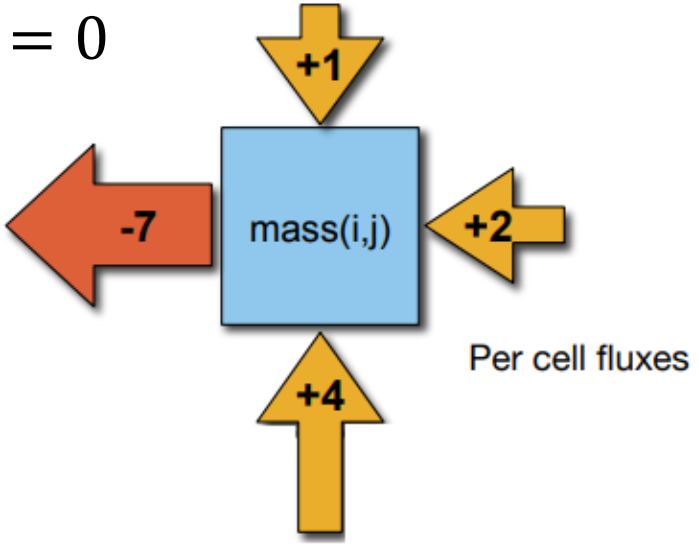
- We can simulate it with **Lagrangian** particle system or **Eulerian** grid system. Eulerian and Lagrangian are **equivalent.**

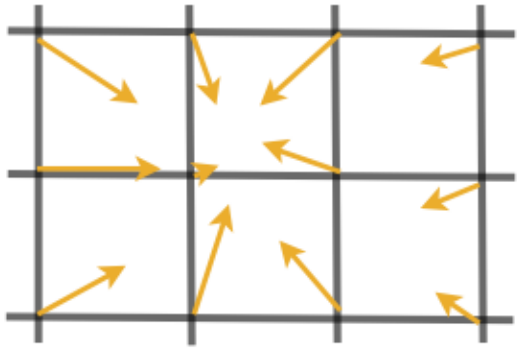
Incompressibility

- NS equation alone is not enough, mass conservation should be maintained:

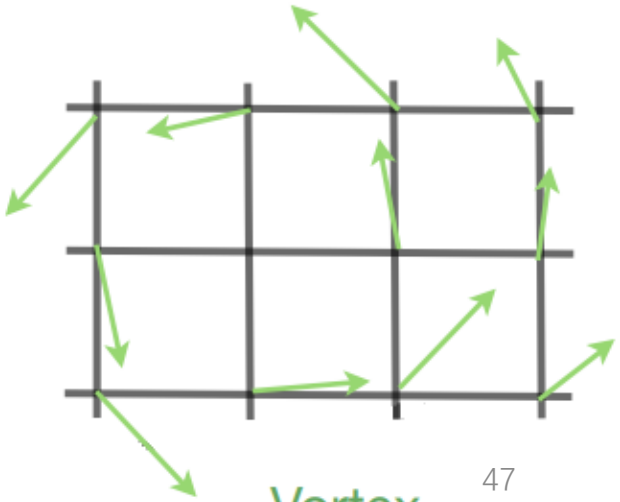
$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho v)$$

- For most fluid (even for smokes), we can assume $\rho = constant$
- Thus, for most fluid (even for smokes), we can assume the fluid is incompressible, which leads to $\nabla \cdot \boldsymbol{v} = 0$





Sink



Vortex

Revisit NS Equation

- $\rho \frac{D\vec{v}}{Dt} = -\nabla p + \rho \vec{g} + \mu \nabla^2 \vec{v}$
- We need pressure to compute forces to update velocity field
- The advance of the velocity field should follow incompressibility constraints
- $\Longrightarrow$The **outcomes** of pressure is incompressibility

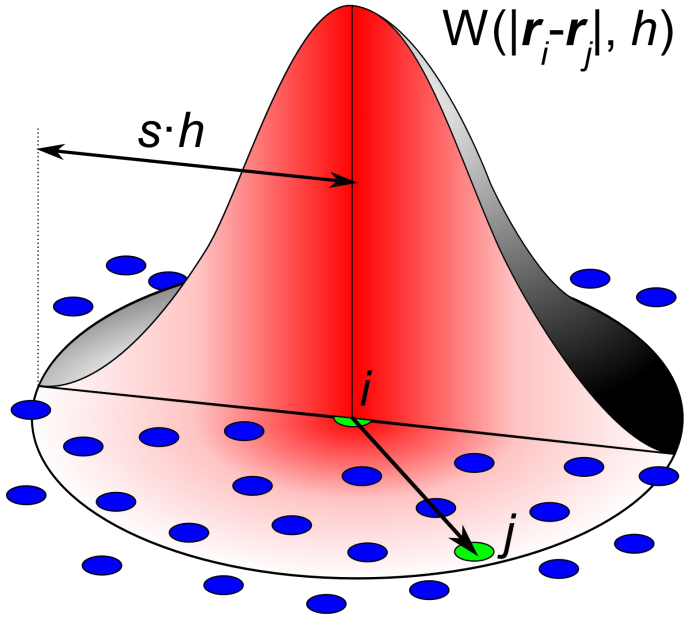
First Attempt: Particle System

- Store the mass, velocity, position on each particle
- What is the density field?

Kernel Functions

- Weight(kernel) function:
  - Manually designed with some condition
  - Larger weight for neighbor
  - $\int w(x)dx = 1$
  - Typically, with a clamped radius
- An example:

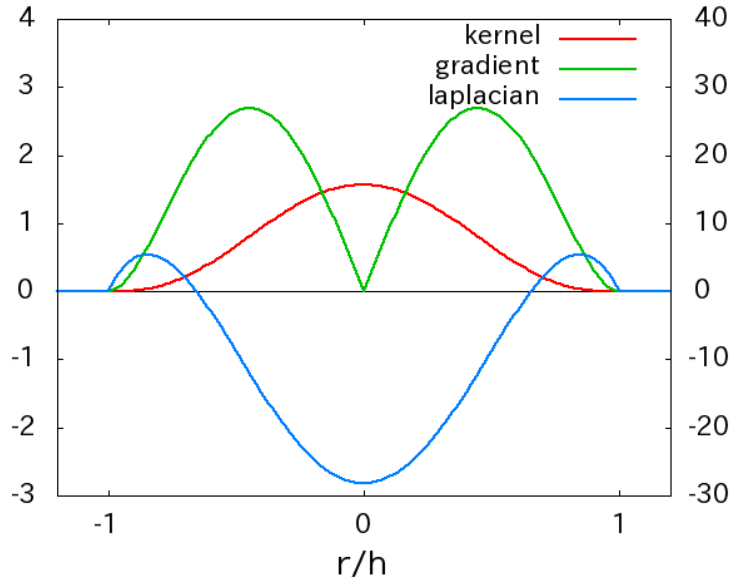
$$W_{\text{poly6}}(r) = \frac{315}{64\pi d^9} \begin{cases} (d^2 - r^2)^3 & 0 \le r \le d \\ 0 & \text{otherwise,} \end{cases}$$



First Attempt: Particle System

- Store the mass, velocity, position on each particle
- What is the density field?

$$\rho(x) = \sum m_i w(|x - x_i|)$$



- What is the pressure field?

$$p(x) = \sum p_i \frac{m_i}{\rho_i} w(|x - x_i|), \nabla p(x) = \sum p_i \frac{m_i}{\rho_i} \nabla w(|x - x_i|)$$

- What is p_i?

$$p_i = k(\rho_i - \rho_0)^\gamma$$

$$pV^\gamma = const$$

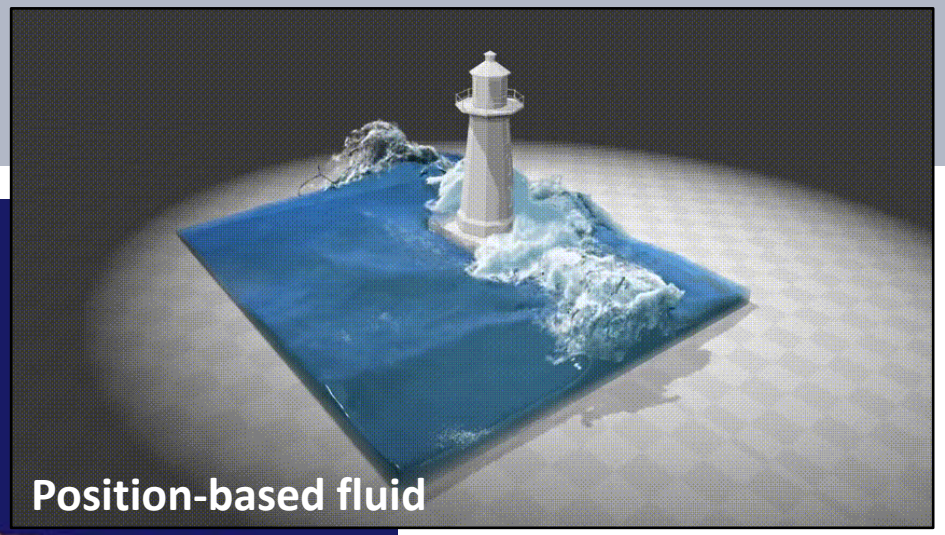
Smoothed Particle Hydrodynamics

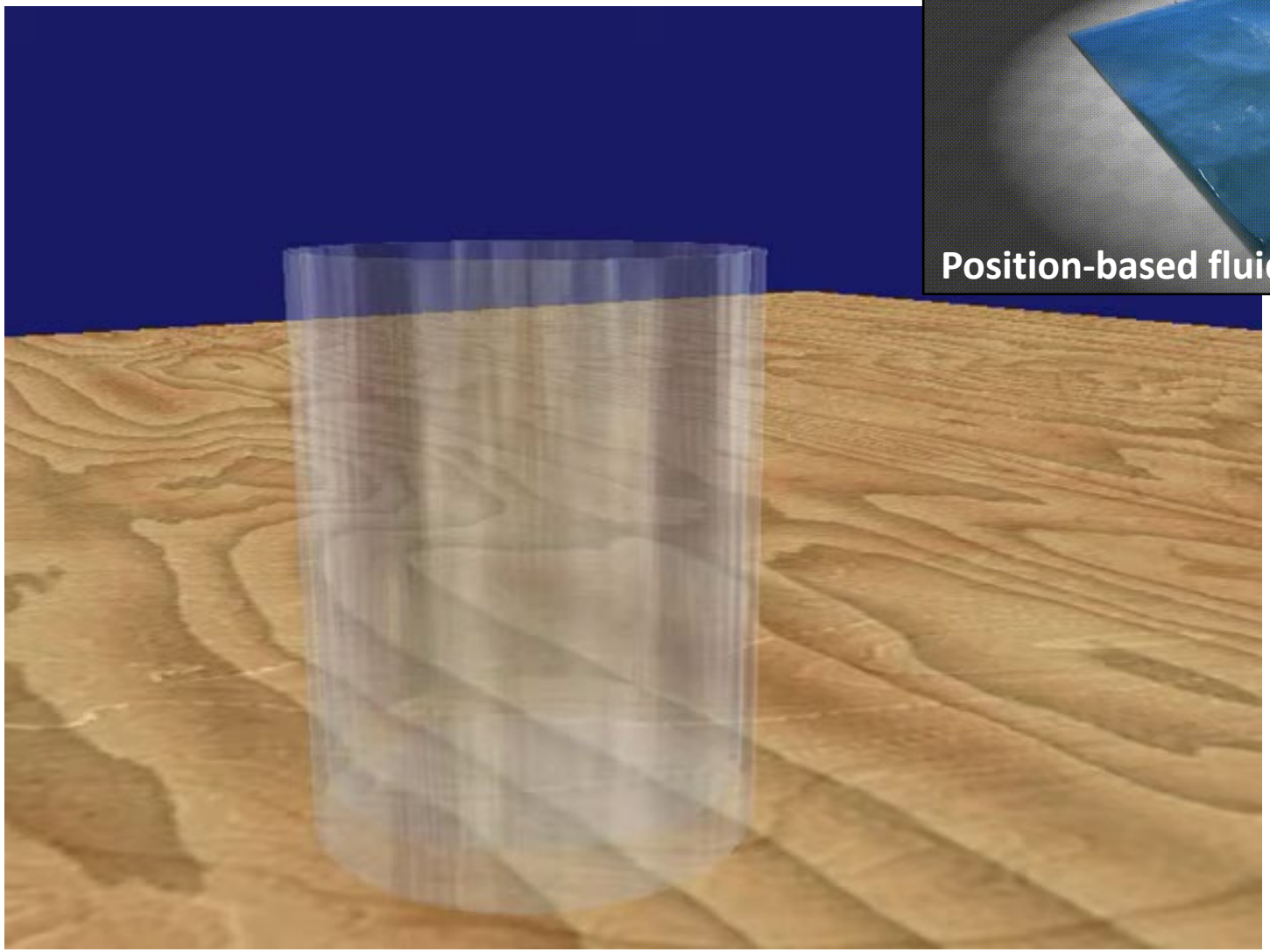
- In each step:
  - Update positions and velocities with gravity
  - Compute density at each particle: $\rho(x_i) = \sum_j m_j w(|x_j - x_i|)$
  - Compute pressure at each particle: $p_i = k(\rho_i - \rho_0)^\gamma$
  - Compute pressure force at each particle: $f_i = -\frac{m_i}{\rho_i} \sum_j p_j \frac{m_j}{\rho_j} \nabla_i w(|x_j - x_i|)$
  - Update positions and velocities with pressure force

Momentum Conserved Way:

$$f_i = -m_i \sum_j m_j \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2} \right) \nabla_i w(|x_j - x_i|)$$

SPH Results

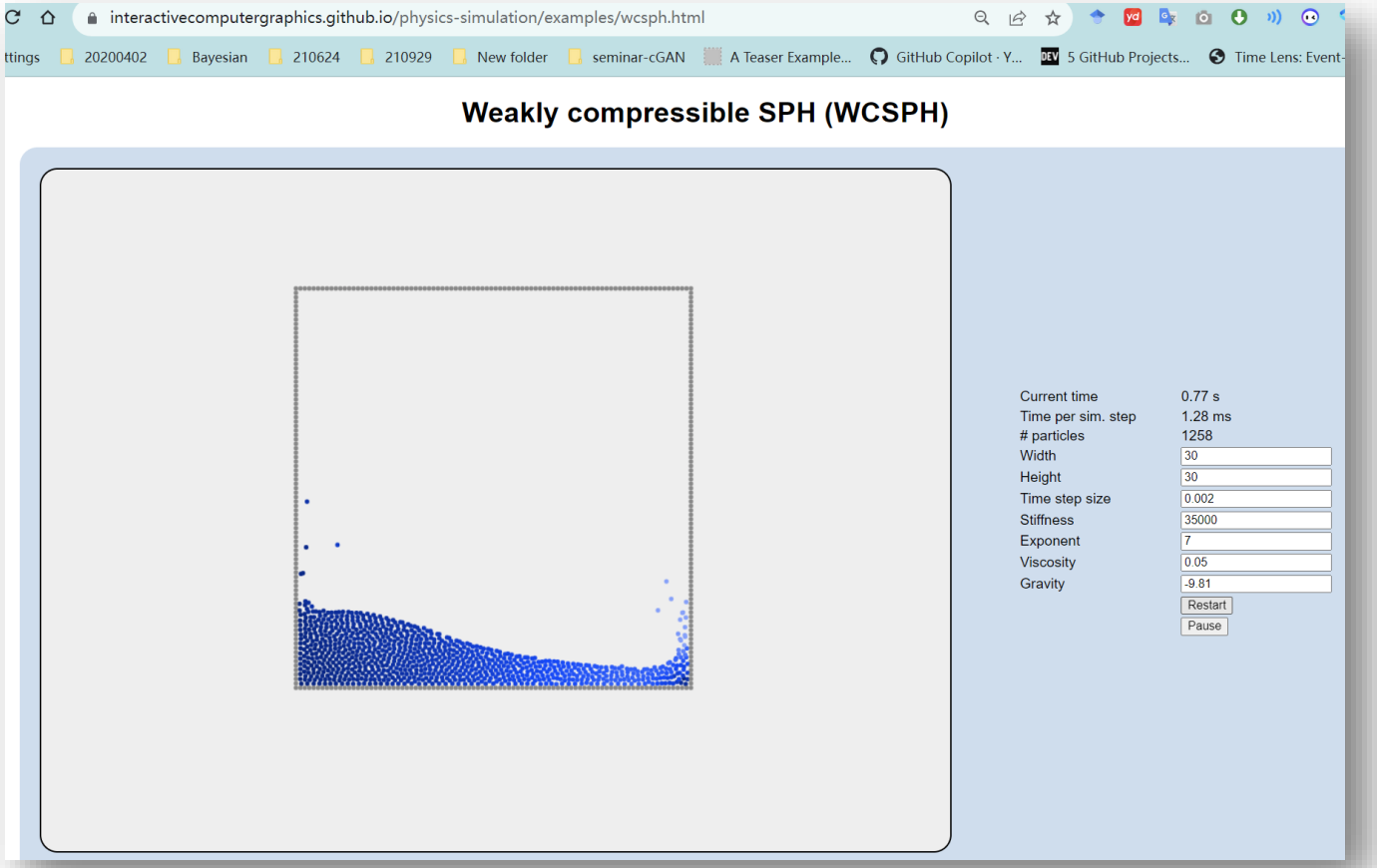


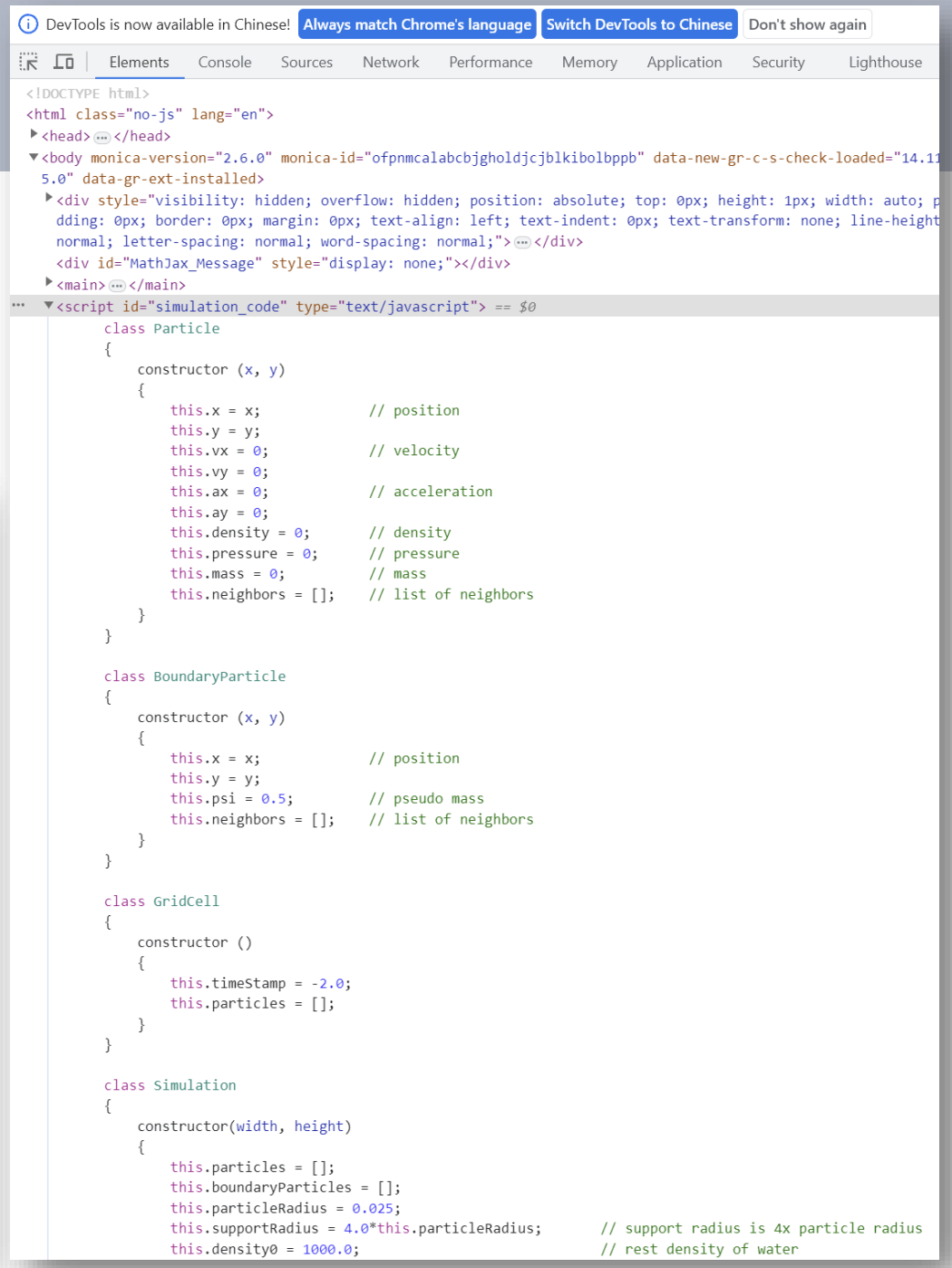


Muller et al. SCA2003

SPH Resources

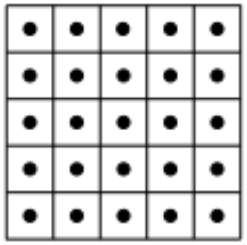
- https://interactivecomputergraphics.github.io/physics-simulation/
- https://interactivecomputergraphics.github.io/physics-simulation/examples/wcsph.html

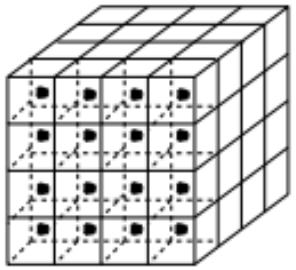


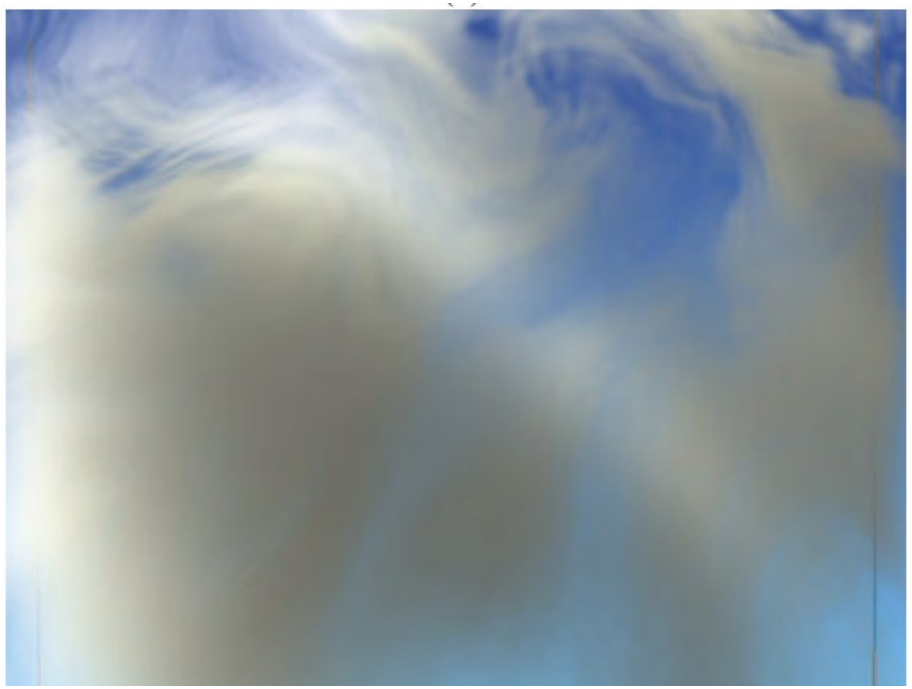


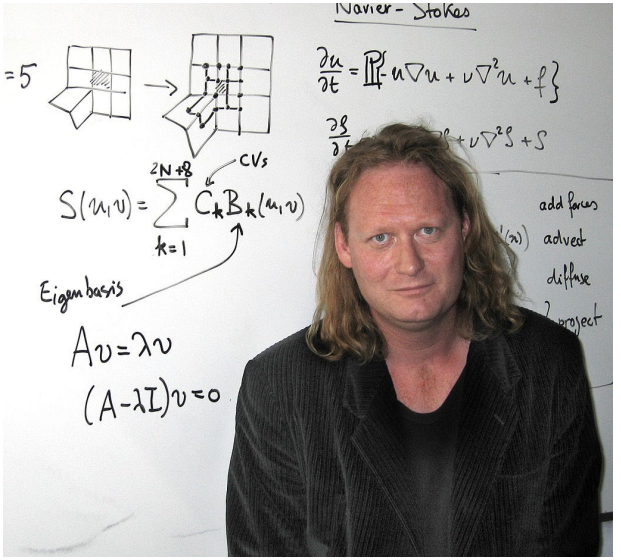
A More Physical Way

Discretize the space into **Grid**, solve NS equation on it





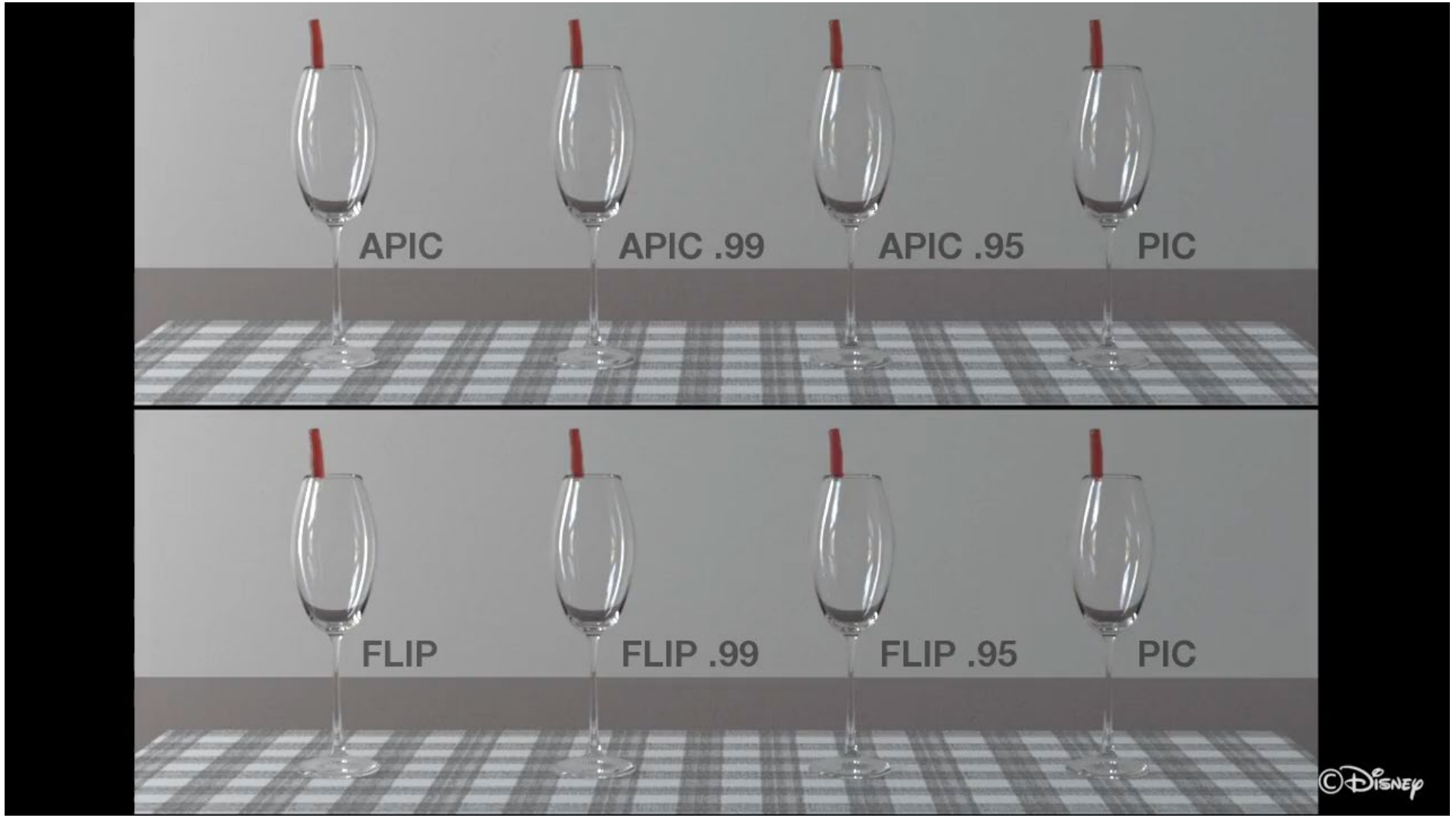




Jos Stam, Stable Fluids, 1999

Online Demo: https://aadebdeb.github.io/WebGL_StableFluids/

Mixed Lagrangian-Eulerian Fluids



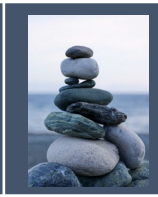
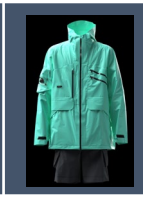
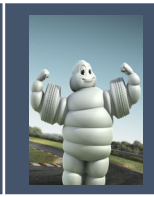
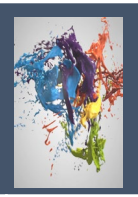
Jiang et al. SIGGRAPH2015

Summary

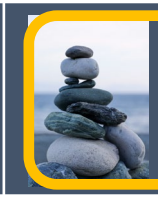
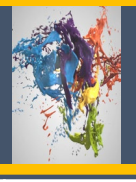
What We've Covered

- The three building blocks of simulator: spatial discretization, temporal discretization, numerical solver
- Spring Mass System: explicit Euler, implicit Euler
- The two viewpoint of fluids: Lagrangian, Eulerian
- SPH method for fluids

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	Coupling

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

Hybrid Methods

Much More…

- Forward：

  New Phenomena

  - Coupling and Interaction
  - The Growth of Plants
  - Animal Development

  New Representation

  - Bubble
  - Monte Carlo-based Simulation

  Acceleration

  - Subspace-Physics
  - Multigrid Solver

  Assets Generation

  - Landscape Generation

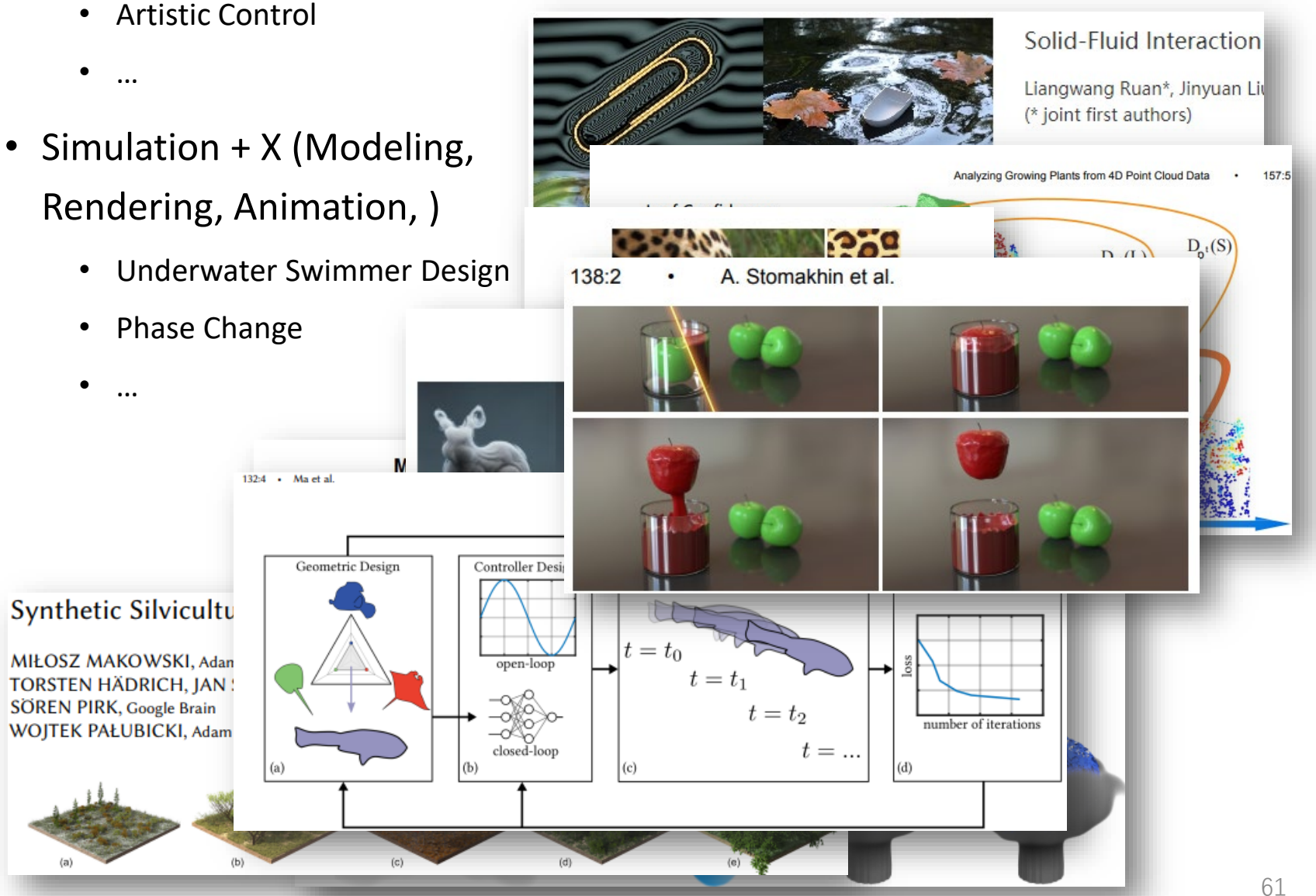
  …

- Inverse:

  - Elasticity in 3D Printing
  - Artistic Control
  - …

- Simulation + X (Modeling, Rendering, Animation,)

  - Underwater Swimmer Design
  - Phase Change
  - …



Thanks