

コンピュータグラフィックス論

－アニメーション(2)－

2017年5月25日

高山 健志

物理シミュレーションに基づく 変形アニメーション

- 物理法則を忠実にモデル化する古典的なシミュレーション方法
 - バネ質点モデル
 - 有限要素法
- 物理法則から逸脱するが、見た目がリアルで計算コストの低い方法
 - Shape Matching (Position-Based Dynamics)

簡単な例：単一バネ質点 (1D)

- 質点の質量 m , 位置 x , バネの係数 k , 自然長 l , 重力 g

運動方程式：

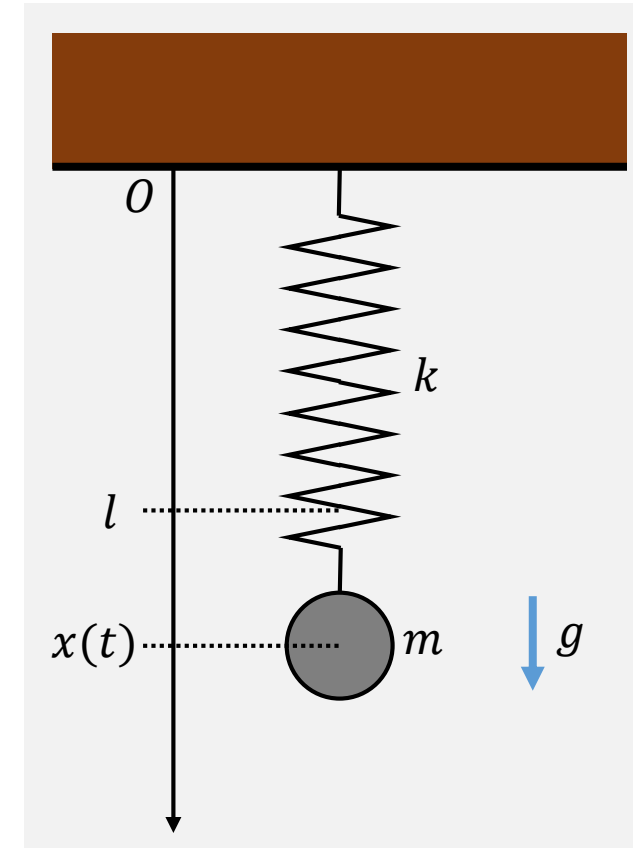
$$m \frac{d^2 x}{dt^2} = -k(x - l) + g$$
$$= f_{\text{int}}(x) + f_{\text{ext}}$$

- 外力 f_{ext} : 重力、床との衝突、ユーザ操作
- 内力 $f_{\text{int}}(x)$: 系が安定状態に戻ろうとする力
 - バネの内部エネルギー (ポテンシャル)

$$E(x) = \frac{k}{2} (x - l)^2$$

- 内力はポテンシャルの勾配の反対

$$f_{\text{int}}(x) = -\frac{dE}{dx} = -k(x - l)$$



3D 空間上のバネ質点系

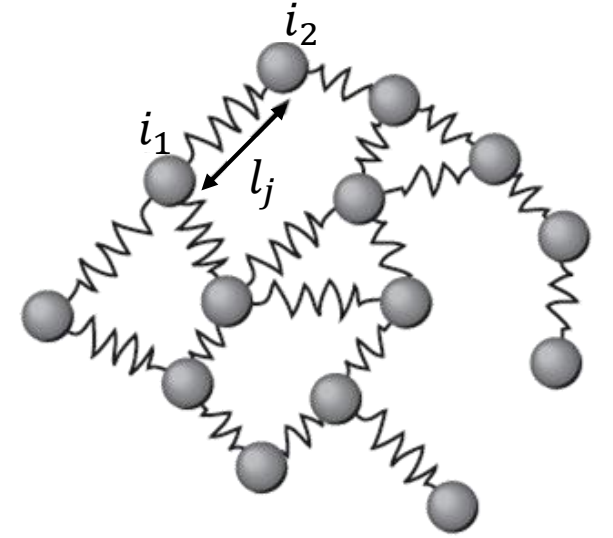
- N 個の質点： i 番目の質点の質量 m_i , 位置 $x_i \in \mathbb{R}^3$
- M 本のバネ： j 番目のバネ $e_j = (i_1, i_2)$
 - バネ係数 k_j , 自然長 l_j
- 状態 $\mathbf{x} = (x_1, \dots, x_N) \in \mathbb{R}^{3N}$ における系のポテンシャル：

$$E(\mathbf{x}) = \sum_{e_j=(i_1, i_2)} \frac{k_j}{2} (\|x_{i_1} - x_{i_2}\| - l_j)^2$$

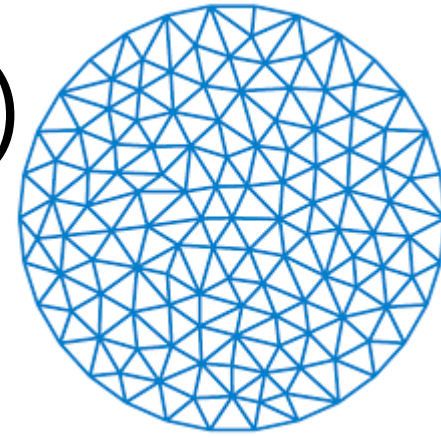
- 運動方程式：

$$\mathbf{M} \frac{d^2 \mathbf{x}}{dt^2} = -\nabla E(\mathbf{x}) + \mathbf{f}_{\text{ext}}$$

- $\mathbf{M} \in \mathbb{R}^{3N \times 3N}$: m_i を成分とする対角行列



連続な弾性体モデル (有限要素法, FEM)



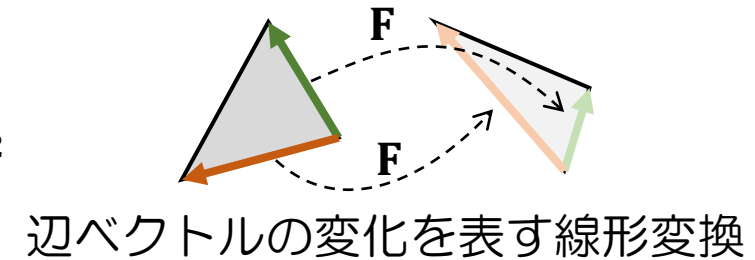
- N 個の頂点 : i 番目の頂点の位置 $x_i \in \mathbb{R}^2$
- M 個の三角形 : j 番目の三角形 $t_j = (i_1, i_2, i_3)$

- 変形前の状態 : $\mathbf{X} = (X_1, \dots, X_N) \in \mathbb{R}^{2N}$
- 変形後の状態 : $\mathbf{x} = (x_1, \dots, x_N) \in \mathbb{R}^{2N}$
- 変形勾配行列 :

$$\mathbf{F}_j(\mathbf{x}) = \begin{pmatrix} x_{i_1} - x_{i_1} & x_{i_1} - x_{i_1} \\ x_{i_2} - x_{i_1} & x_{i_3} - x_{i_1} \end{pmatrix} \begin{pmatrix} X_{i_1} - X_{i_1} & X_{i_1} - X_{i_1} \\ X_{i_2} - X_{i_1} & X_{i_3} - X_{i_1} \end{pmatrix}^{-1} \in \mathbb{R}^{2 \times 2}$$

- 系のポテンシャル :

$$E(\mathbf{x}) = \sum_{t_j=(i_1, i_2, i_3)} \frac{A_j}{2} \left\| \mathbf{F}_j(\mathbf{x})^T \mathbf{F}_j(\mathbf{x}) - \mathbf{I} \right\|_{\mathcal{F}}^2$$



Green's strain energy

- 運動方程式 :

$$\mathbf{M} \frac{d^2 \mathbf{x}}{dt^2} = -\nabla E(\mathbf{x}) + \mathbf{f}_{\text{ext}}$$

- $\mathbf{M} \in \mathbb{R}^{2N \times 2N}$: 各頂点のボロノイ領域の面積を成分とする対角行列

ダイナミックな変形の計算

- 位置 $\mathbf{x}(t)$ と速度 $\mathbf{v}(t) := \frac{d\mathbf{x}}{dt}$ の初期条件

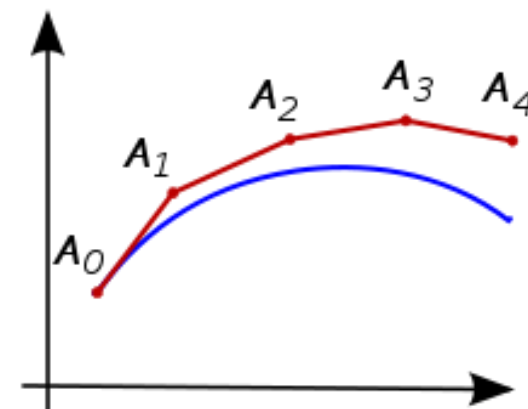
$$\mathbf{x}(0) = \mathbf{x}_0, \quad \mathbf{v}(0) = \mathbf{v}_0$$

が与えられたとき、 $\mathbf{x}(t)$, $\mathbf{v}(t)$ を求める問題 (initial value problem)

- 問題が簡単な場合：単一バネ質点

$$m \frac{d^2 x}{dt^2} = -k(x - l) + g$$

- 解析解が求まる (sine curve)
- 一般の問題には解析解が存在しない
 - ➔ 時刻 t における状態 $(\mathbf{x}_n, \mathbf{v}_n)$ から、時刻 $t + h$ における状態 $(\mathbf{x}_{n+1}, \mathbf{v}_{n+1})$ を計算する (time integration)
 - h : 時間幅



最も単純な方法：explicit Euler

加速度を差分法で離散化：

$$\mathbf{M} \frac{\mathbf{v}_{n+1} - \mathbf{v}_n}{h} = \mathbf{f}_{\text{int}}(\mathbf{x}_n) + \mathbf{f}_{\text{ext}}$$

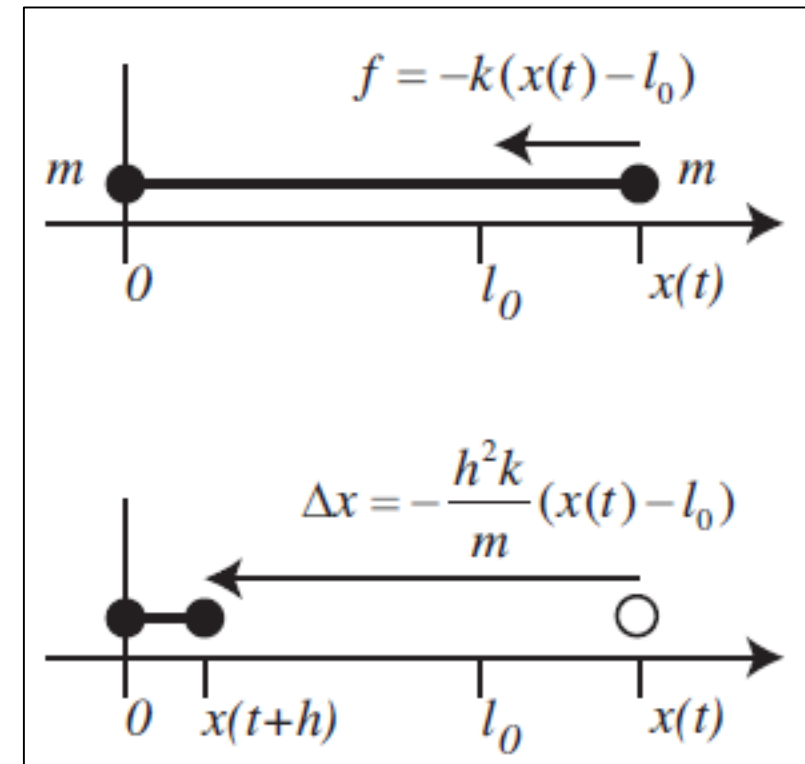
速度の更新

$$\mathbf{v}_{n+1} \leftarrow \mathbf{v}_n + h \mathbf{M}^{-1} (\mathbf{f}_{\text{int}}(\mathbf{x}_n) + \mathbf{f}_{\text{ext}})$$

位置の更新

$$\mathbf{x}_{n+1} \leftarrow \mathbf{x}_n + h \mathbf{v}_{n+1}$$

- 利点：計算が簡単
- 欠点：overshooting
 - 時間幅を大きくすると、簡単に元の振幅よりも遠い地点に到達する
→ 時間経過とともにエネルギーが発散



使うべき手法：implicit Euler

Find $(\mathbf{x}_{n+1}, \mathbf{v}_{n+1})$ such that:

$$\begin{cases} \mathbf{v}_{n+1} = \mathbf{v}_n + h \mathbf{M}^{-1} (\mathbf{f}_{\text{int}}(\mathbf{x}_{n+1}) + \mathbf{f}_{\text{ext}}) \\ \mathbf{x}_{n+1} = \mathbf{x}_n + h \mathbf{v}_{n+1} \end{cases}$$

- 未知の移動先 \mathbf{x}_{n+1} における内力を使って \mathbf{v}_{n+1} を表す
- 利点：overshoot を回避できる
- 難点：計算コストが高い (方程式を解く)

implicit Euler の中身

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h \mathbf{v}_{n+1}$$

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

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

$$= \mathbf{x}_n + h \mathbf{v}_n + h^2 \mathbf{M}^{-1} (-\nabla E(\mathbf{x}_{n+1}) + \mathbf{f}_{\text{ext}})$$

未知数 \mathbf{x}_{n+1} を \mathbf{y} とおく

$$h^2 \nabla E(\mathbf{y}) + \mathbf{M} \mathbf{y} - \mathbf{M}(\mathbf{x}_n + h \mathbf{v}_n) - h^2 \mathbf{f}_{\text{ext}} = \mathbf{0}$$
$$\mathbf{F}(\mathbf{y})$$

- 関数 $\mathbf{F}: \mathbb{R}^{3N} \mapsto \mathbb{R}^{3N}$ のルートを求める問題に帰着 \rightarrow Newton 法

$$\mathbf{y}^{(i+1)} \leftarrow \mathbf{y}^{(i)} - \left(\frac{d\mathbf{F}}{d\mathbf{y}} \right)^{-1} \mathbf{F}(\mathbf{y}^{(i)})$$
$$= \mathbf{y}^{(i)} - \left(h^2 \mathcal{H}_E(\mathbf{y}^{(i)}) + \mathbf{M} \right)^{-1} \mathbf{F}(\mathbf{y}^{(i)})$$

ポテンシャル関数 E の2階微分 (ヘッセ行列)

- 大規模線形方程式の係数行列が、反復毎に変わる \rightarrow 計算が大変！

バネ質量モデル vs 連続体モデル (FEM)

- 微小要素の変形量の合計としてポテンシャルを定義する点は共通
 - いずれも implicit Euler が必要
- 2D/3D 領域を密に満たす物体を近似する場合、バネ質量は不正確
 - 網目状の材質などに対しては有効
 - <https://www.youtube.com/watch?v=N520KFOxaDg>
- FEM は一般に計算コストが高い
 - 領域のメッシュ分割
 - 複雑なポテンシャル関数
 - 多様な (非線形) 材質を扱える
 - <http://graphics.cs.cmu.edu/projects/Bargteil-2007-AFE/Stuff/examples-360.mov>

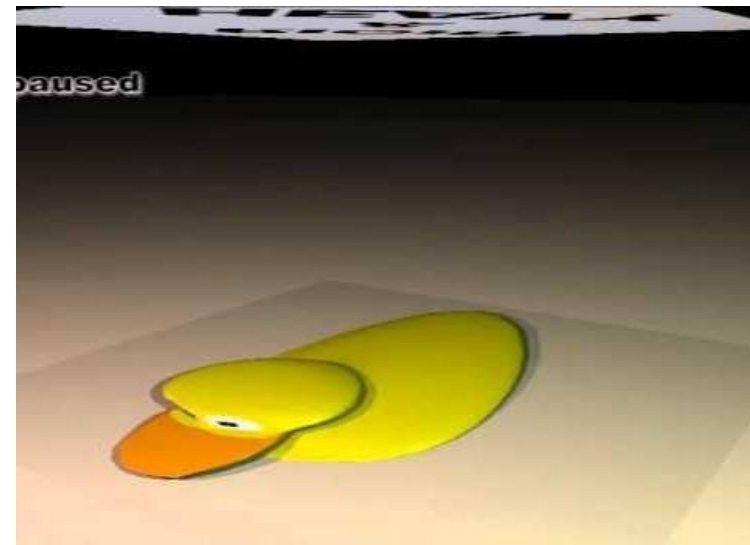
	バネ質量	FEM
物理的正確さ	△	○
計算・実装コスト	○	△

物理シミュレーションに基づく 変形アニメーション

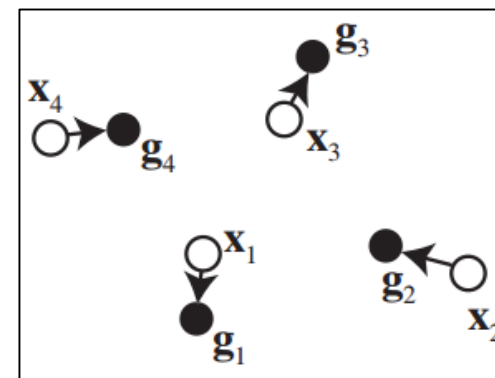
- 物理法則を忠実にモデル化する古典的なシミュレーション方法
 - バネ質点モデル
 - 有限要素法
- 物理法則から逸脱するが、見た目がリアルで計算コストの低い方法
 - Shape Matching (Position-Based Dynamics)

CG に特化した物理アニメーション計算

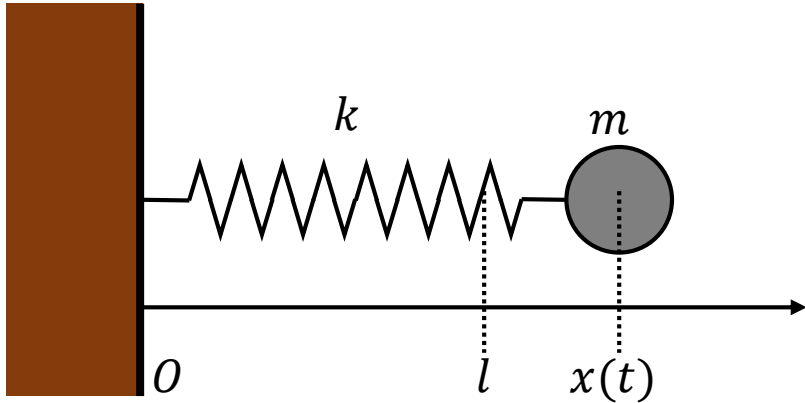
- 最初の論文
 - Meshless deformations based on shape matching [Müller et al., SIGGRAPH 2005]
 - Position Based Dynamics [Müller et al., VRIPhys 2006]
- 基本アイディア
ポテンシャルがゼロになる点 (goal position) を求め、そこに向けて質点 (パーティクル) を引っ張る
- 系全体のエネルギーが必ず減少する (発散しない)
- 計算が簡単 → ゲーム等に最適
- 物理的に意味のあるエネルギーに基づく計算 (FEM) ではない
 - CG 用途なら問題無し



<https://www.youtube.com/watch?v=CClwiC37kks>



比較：単一のバネ質点の場合



Explicit Euler

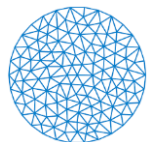
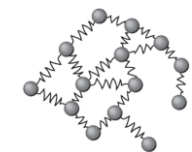
$$\begin{aligned}v_{n+1} &\leftarrow v_n + \frac{h k}{m} (l - x_n) \\x_{n+1} &\leftarrow x_n + h v_{n+1}\end{aligned}$$

Shape Matching

$$\begin{aligned}v_{n+1} &\leftarrow v_n + \frac{\alpha}{h} (l - x_n) \\x_{n+1} &\leftarrow x_n + h v_{n+1}\end{aligned}$$

- $0 \leq \alpha \leq 1$ は Shape Matching に特有の「硬さ」を表すパラメタ
 - $\alpha = 0$ の場合 \rightarrow 速度が変化しない (バネが無限大に柔らかい)
 - $\alpha = 1$ の場合 \rightarrow バネが無限大に硬いのと同じ (?)
 - \rightarrow どんな場合でも系のエネルギーが発散しない ☺
- 係数の単位に注目： α/h の単位は(時間) $^{-1}$ $\rightarrow \alpha$ は物理量ではない！
 - SMが物理ベースではなく幾何ベースと呼ばれるゆえん

比較：複雑な形状の変形の場合



Explicit Euler

$$\mathbf{v}_{n+1} \leftarrow \mathbf{v}_n - h \mathbf{M}^{-1} \nabla E(\mathbf{x}_n)$$

$$\mathbf{x}_{n+1} \leftarrow \mathbf{x}_n + h \mathbf{v}_{n+1}$$

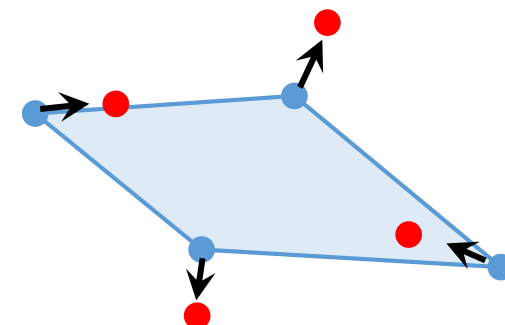
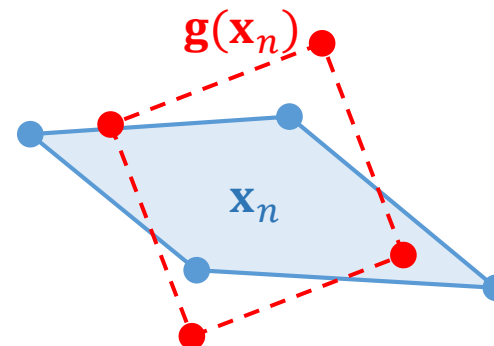
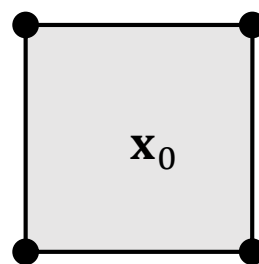
Shape Matching

$$\mathbf{v}_{n+1} \leftarrow \mathbf{v}_n + \frac{\alpha}{h} (\mathbf{g}(\mathbf{x}_n) - \mathbf{x}_n)$$

$$\mathbf{x}_{n+1} \leftarrow \mathbf{x}_n + h \mathbf{v}_{n+1}$$

- Goal position \mathbf{g}

- 変形前の初期形状を現在の
変形形状に最も良くマッチ
するように剛体変換したもの
 - (モーメント行列の特異値分解)

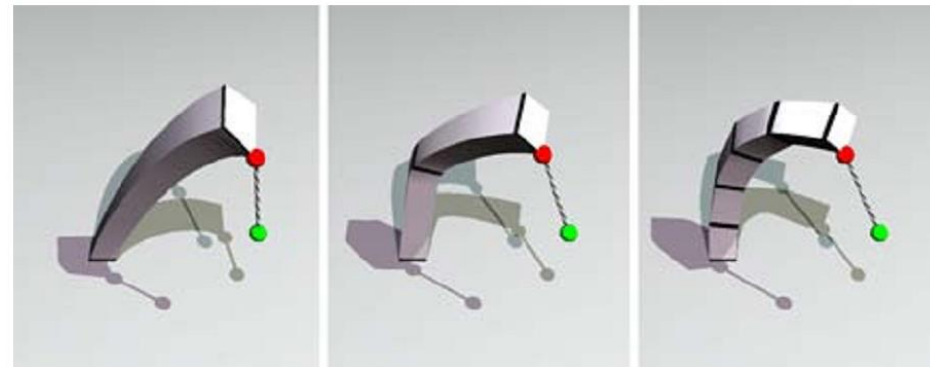


- 特長：計算が簡単・速い！

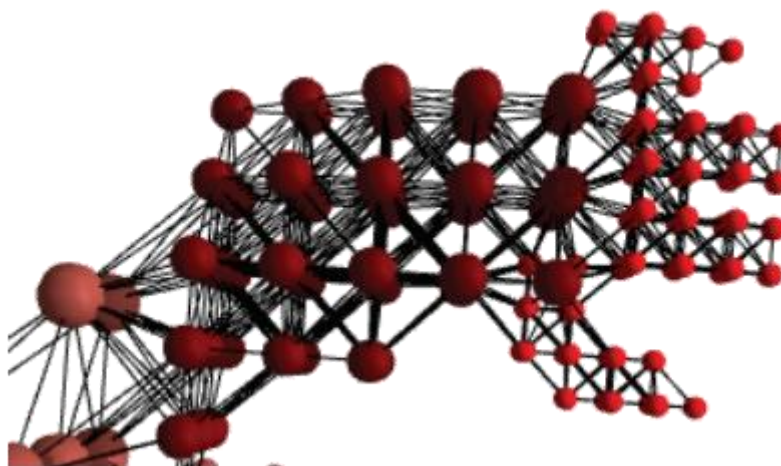
- 頂点同士の接続情報 (バネ、メッシュなど) が不要 (meshless)
- 大規模方程式を解かなくて良い

局所領域ごとの Shape Matching

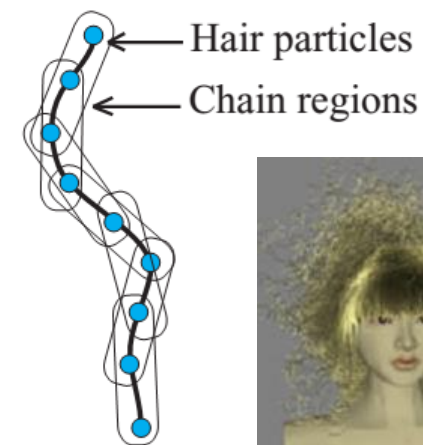
- より複雑な変形を実現
- 高速化などの拡張



ボクセル格子による局所領域



Octree による局所領域



一次元的構造を使った
髪の毛のアニメーション

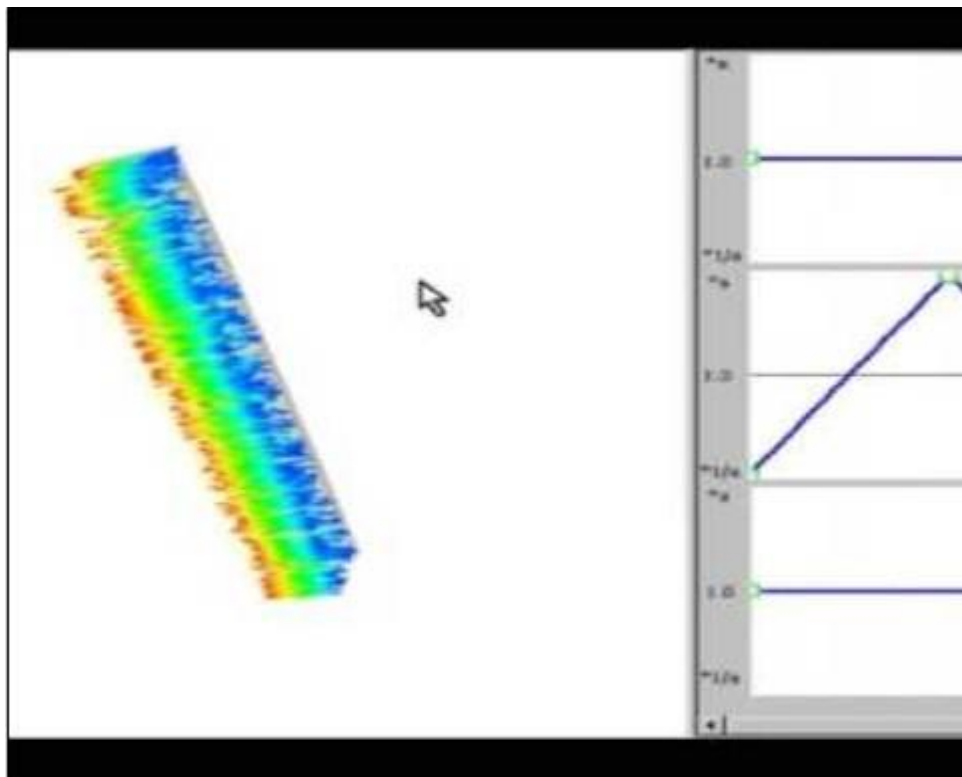
FastLSM; fast lattice shape matching for robust real-time deformation [Rivers SIGGRAPH07]

Fast adaptive shape matching deformations [Steinemann SCA08]

Chain Shape Matching for Simulating Complex Hairstyles [Rungjiratananon CGF10]

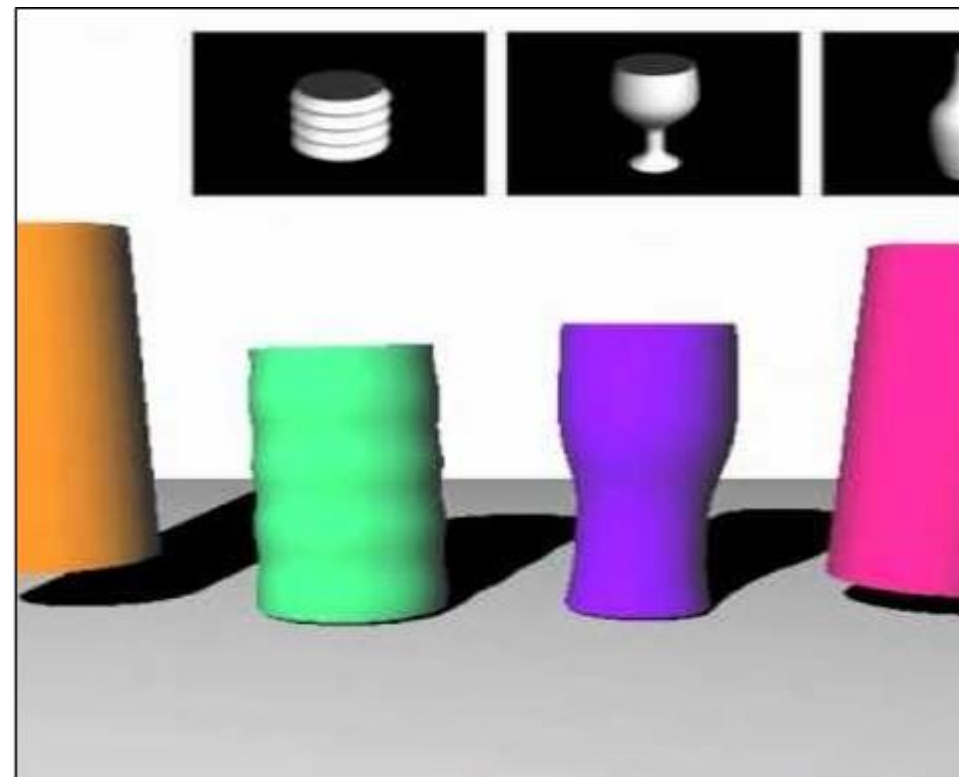
応用：局所領域の rest shape を変化させる

自律的に動く柔軟物体



<https://www.youtube.com/watch?v=0AWtQbVBi3s>

変形の仕方を例示によって
制御できる弾性体



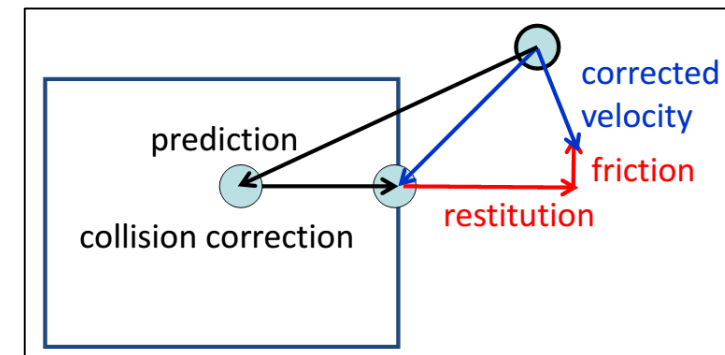
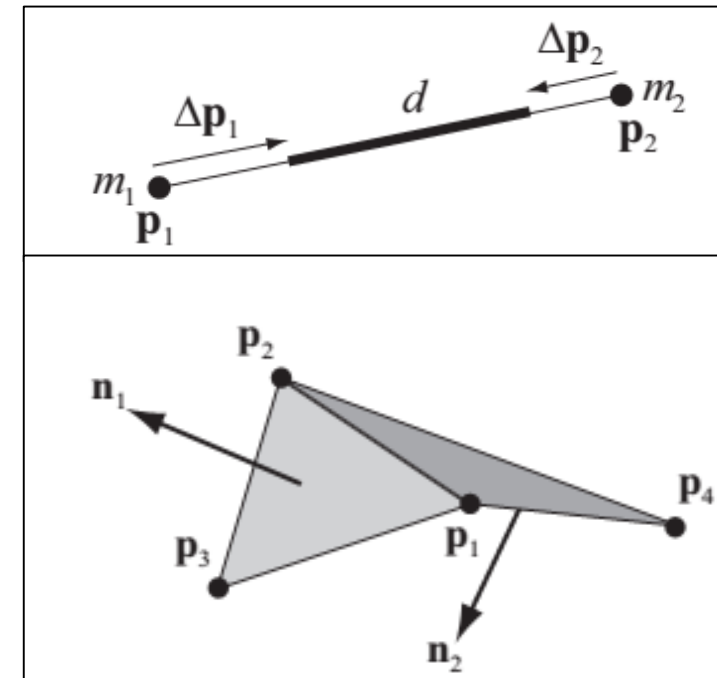
<https://www.youtube.com/watch?v=45QjojWiOEc>

Position-Based Dynamics (PBD)

- Shape Matching法を含めた、幾何ベースのアニメーション計算の枠組み
- 入力：初期位置 \mathbf{x}_0 , 初期速度 \mathbf{v}_0
- フレーム毎の処理：

\mathbf{p}	$= \mathbf{x}_n + h \mathbf{v}_n$	prediction
\mathbf{x}_{n+1}	$= \text{modify}(\mathbf{p})$	position correction
\mathbf{u}	$= (\mathbf{x}_{n+1} - \mathbf{x}_n)/h$	velocity update
\mathbf{v}_{n+1}	$= \text{modify}(\mathbf{u})$	velocity correction

(衝突と摩擦の扱い等、全体的にちゃんと理解できていない)



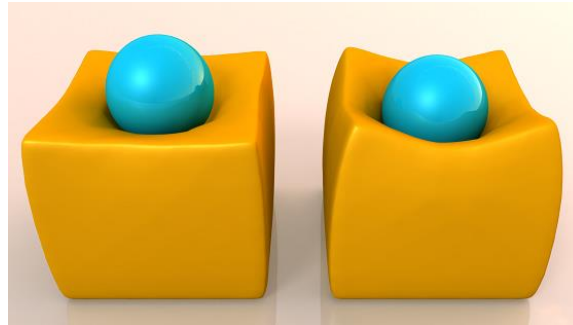
PBD 利用できる Shape Matching 以外の 様々な幾何制約



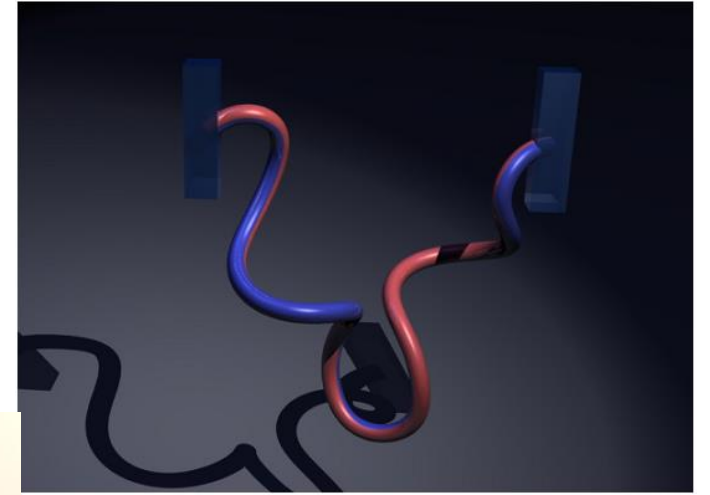
体積制約



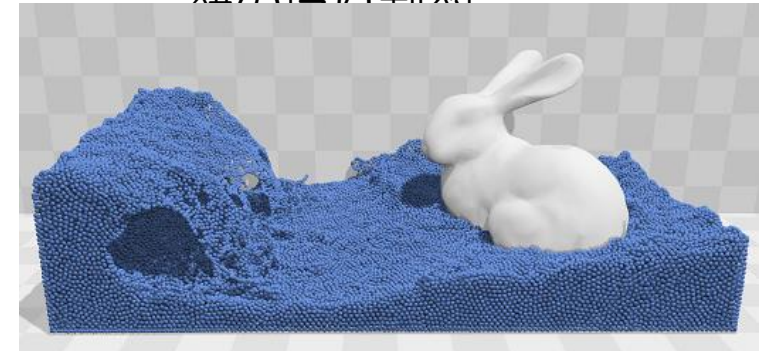
布の伸び幅制約



連続体の歪み制約



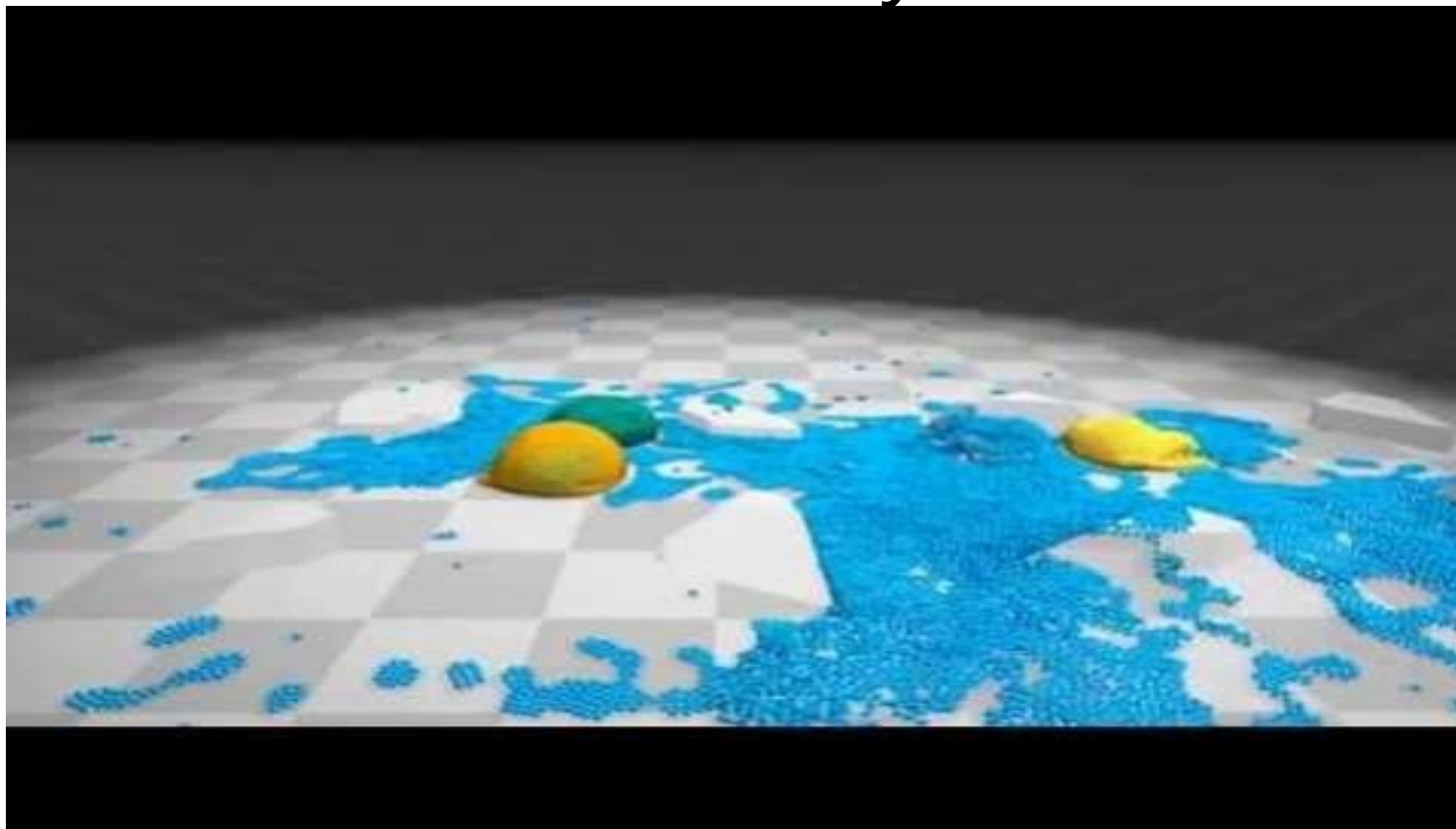
紐の結合制約



粒子の密度制約

Robust Real-Time Deformation of Incompressible Surface Meshes [Dziol SCA11]
Long Range Attachments - A Method to Simulate Inextensible Clothing in Computer Games [Kim SCA12]
Position Based Fluids [Macklin SIGGRAPH13]
Position-based Elastic Rods [Umetani SCA14]
Position-Based Simulation of Continuous Materials [Bender Comput&Graph14]

PBD の集大成：FLEX in PhysX

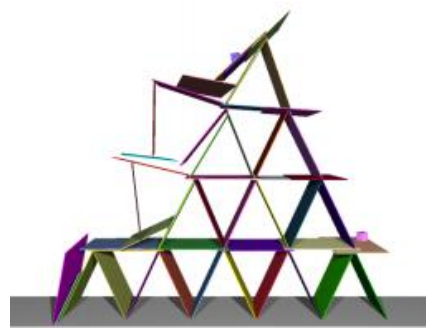


- NVIDIA が SDK を公開！

<https://www.youtube.com/watch?v=z6dAahLUbZg>

衝突判定

- また別のやっかいな問題
- PBD でよく使う方法
 - ボクセル格子のどのセルにパーティクルが存在するか記録
 - 近傍セルのパーティクル同士でのみ衝突判定
- 最近発表された PBD 用の手法



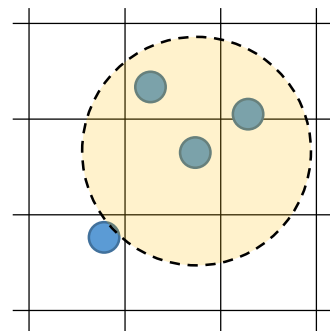
[Kaufman08]



[Harmon09]



[Zheng12]



[Muller15]

Collision detection for deformable objects [Teschner CGF05]

Staggered Projections for Frictional Contact in Multibody Systems [Kaufman SIGGRAPHAsia08]

Asynchronous Contact Mechanics [Harmon SIGGRAPH09]

Energy-based Self-Collision Culling for Arbitrary Mesh Deformations [Zheng SIGGRAPH12]

Air Meshes for Robust Collision Handling [Muller SIGGRAPH15]

参考情報

- サーベイ・チュートリアル等
 - A Survey on Position-Based Simulation Methods in Computer Graphics [Bender CGF14]
 - http://www.csee.umbc.edu/csee/research/vangogh/I3D2015/matthias_muller_slides.pdf
 - Position-Based Simulation Methods in Computer Graphics [Bender EG15Tutorial]
- ライブラリ、実装例等
 - <https://code.google.com/p/opencloth/>
 - <http://shapeop.org/>
 - <http://matthias-mueller-fischer.ch/demos/matching2dSource.zip>
 - <https://bitbucket.org/yukikoyama>
 - <https://developer.nvidia.com/physx-flex>
 - <https://github.com/janbender/PositionBasedDynamics>