コンピュータグラフィクス論

– アニメーション(2) –

2015年5月21日

高山 健志

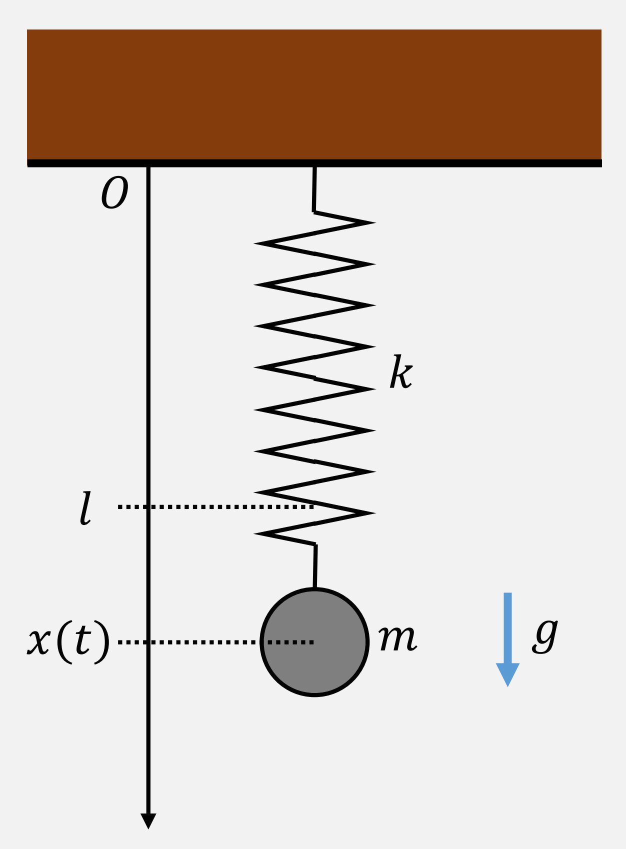
物理ベースの変形アニメーション

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：

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



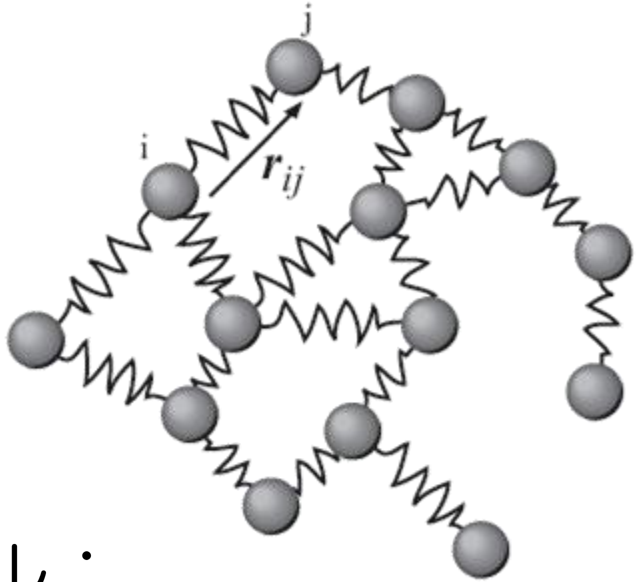
- 外力 f_{ext}　：重力、床との衝突、ユーザ操作
- 内力 $f_{\mathrm{int}}(x)$　：系が安定状態に戻ろうとする力
  - バネの内部エネルギー (ポテンシャル)

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

  - 内力はポテンシャルの勾配の反対

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

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 番目のバネの係数 k_j, 自然長 l_j

- 状態 $\mathbf{x} = (x_1, \ldots, x_N) \in \mathbb{R}^{3N}$における系のポテンシャル：

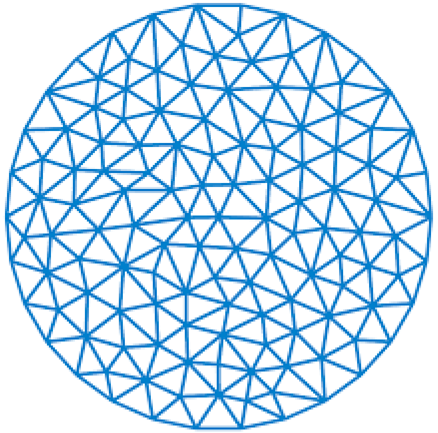
$$E(\mathbf{x}) = \sum_{e_j=(i_1,i_2)} \frac{k_j}{2}\left(\|x_{i_1} - x_{i_2}\| - l_j\right)^2$$

- 運動方程式：

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

  - $\mathbf{M} \in \mathbb{R}^{3N\times 3N}$ ：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)



領域を三角形メッシュに分割

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

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

辺ベクトルの変化
を表す線形変換

- 系のポテンシャル：

t_j の面積

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

Green's strain energy

- 運動方程式：

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

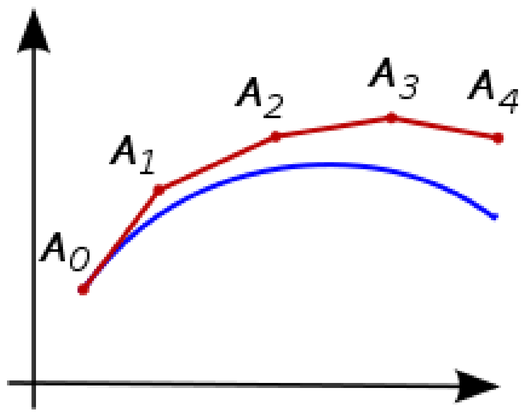
  - $\mathbf{M} \in \mathbb{R}^{2N\times 2N}$ ：各頂点のボロノイ領域の面積を成分とする対角行列

ダイナミックな変形の計算

- 位置 $\mathbf{x}(t)$ と速度 $\frac{d\mathbf{x}}{dt} = \mathbf{v}(t)$ の初期条件
$$\mathbf{x}(0) = \mathbf{x}_0, \qquad \mathbf{v}(0) = \mathbf{v}_0$$
を与えられたとき、$\mathbf{x}(t)$, $\mathbf{v}(t)$ を求める問題 (initial value problem)

- 問題が簡単な場合：単一バネ質点
$$m\frac{d^2x}{dt^2} = -k(x-l)+g$$
  - 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 $(\mathbf{x}_{n+1}, \mathbf{v}_{n+1})$ を求める (time integration)
    - h：時間幅



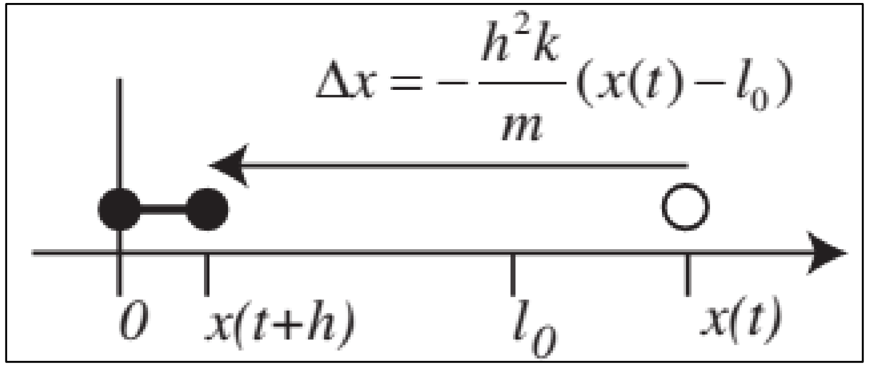
最も簡単な方法：explicit Euler

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

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

- 現在位置 $\mathbf{x}_n$ から計算した速度 $\mathbf{v}_{n+1}$ で移動先 $\mathbf{x}_{n+1}$ を求める
  ➔ 計算が簡単

- 問題点：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

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

- 未知の移動先 $\mathbf{x}_{n+1}$ における内力を使って $\mathbf{v}_{n+1}$ を定義
  ➔ 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

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

$$\begin{aligned}\mathbf{x}_{n+1} &= \mathbf{x}_n \ + \ h\,\mathbf{v}_{n+1} \\ &= \mathbf{x}_n \ + \ h\,\mathbf{v}_n + h^2\mathbf{M}^{-1}\left(-\nabla E(\mathbf{x}_{n+1}) + \mathbf{f}_{\text{ext}}\right)\end{aligned}$$

- 未知数を区別するために $\mathbf{x}_{n+1}$ を $\mathbf{y}$ とおくと、

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

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

Newton 法による implicit Euler の計算

- ヤコビ行列 $\boldsymbol{\mathcal{J}}(\mathbf{y}) = \frac{d\mathbf{F}}{d\mathbf{y}} = h^2 \boldsymbol{\mathcal{H}}_E(\mathbf{y}) + \mathbf{M} \in \mathbb{R}^{3N\times 3N}$ を使って、反復計算

$$\mathbf{y}^{(i+1)} = \mathbf{y}^{(i)} + \boldsymbol{\mathcal{J}}\left(\mathbf{y}^{(i)}\right)^{-1} \mathbf{F}(\mathbf{y}^{(i)})$$

  - $\boldsymbol{\mathcal{H}}_E$：ポテンシャル関数 $E(\mathbf{x})$ のヘシアン行列 (2階微分)

- 大規模線形方程式の係数行列が、反復毎に変わる ➔ 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
  - 紐などに対しては有効

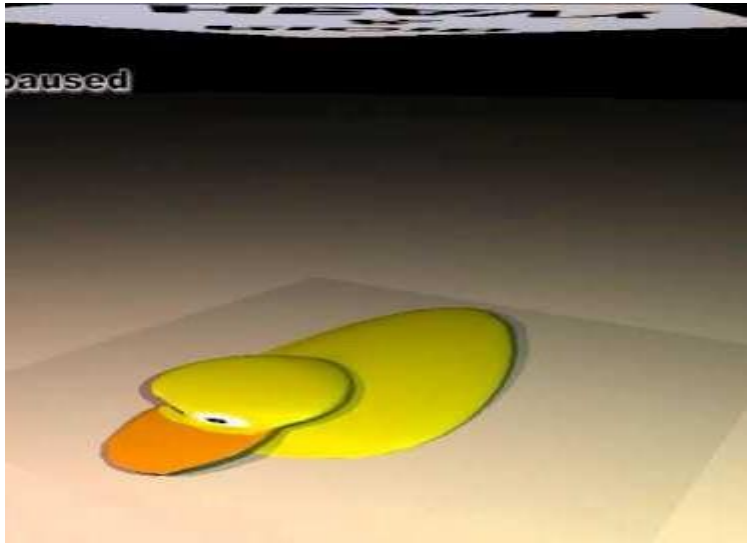
- FEM は一般に計算コストが高い
  - 領域のメッシュ分割
  - 複雑なポテンシャル関数
    - 多様な (非線形) 材質を扱える

	バネ質量	FEM
物理的正確さ	△	○
計算・実装コスト	○	△

Position-**B**ased **D**ynamics

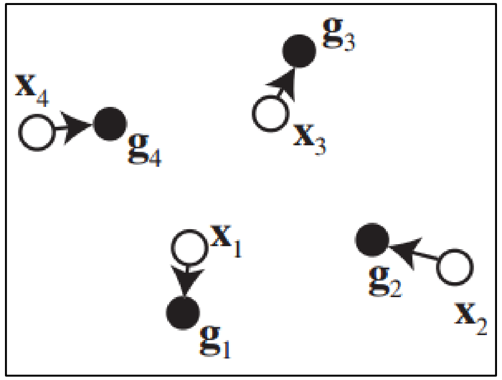
PBD : CG に特化した物理アニメーション計算

- PBD の最初の論文
  - 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



https://www.youtube.com/watch?v=CCIwiC37kks

- 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

単一のバネ質点の場合

goal position

$$v_{n+1} = (1-\alpha)\left(v_n + h\frac{f_{\text{ext}}}{m}\right) + \alpha\frac{l - x_n}{h}$$

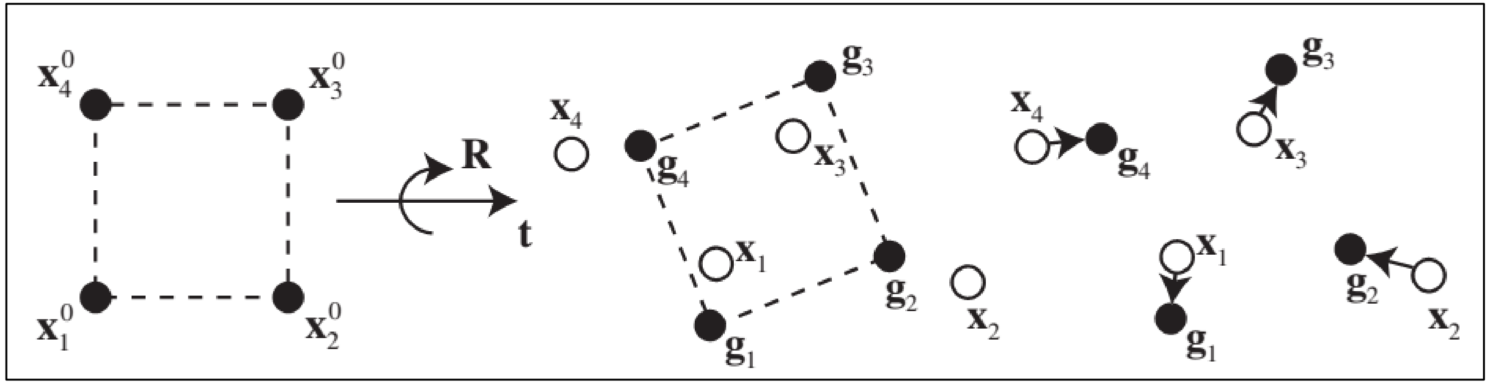
$$x_{n+1} = x_n + h\,v_{n+1}$$

論文の式 (9) は間違い (？)

- 問題：一般の場合に、 goal position をどう求めるか？

Shape Matching による変形計算

- 基本アイディア：
  - 初期の形状を、現在の形状に最も近づけるような剛体変換を求め、それを goal position とする



- 平行移動 $\mathbf{t}$
  - $\mathbf{t} = \mathbf{c} - \mathbf{C}$ (初期の重心と現在の重心の差)
- 回転 $\mathbf{R}$
  - モーメント行列

$$\mathbf{A} = \sum_i m_i\,(\mathbf{x}_i - \mathbf{c})(\mathbf{X}_i - \mathbf{C})^\intercal$$

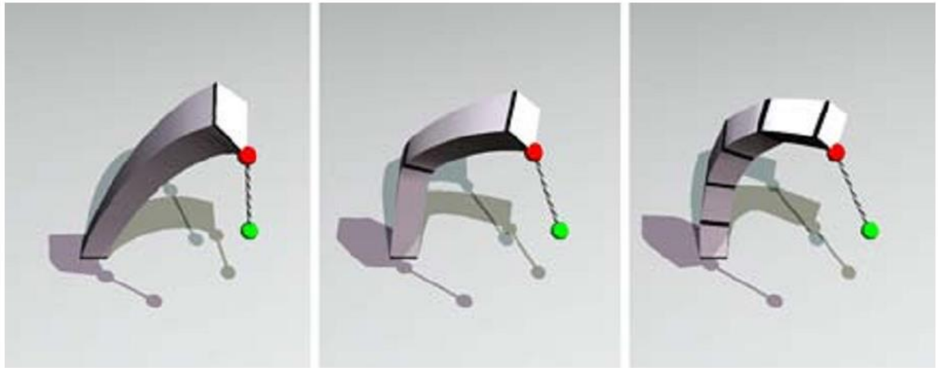
から、回転成分を取り出す

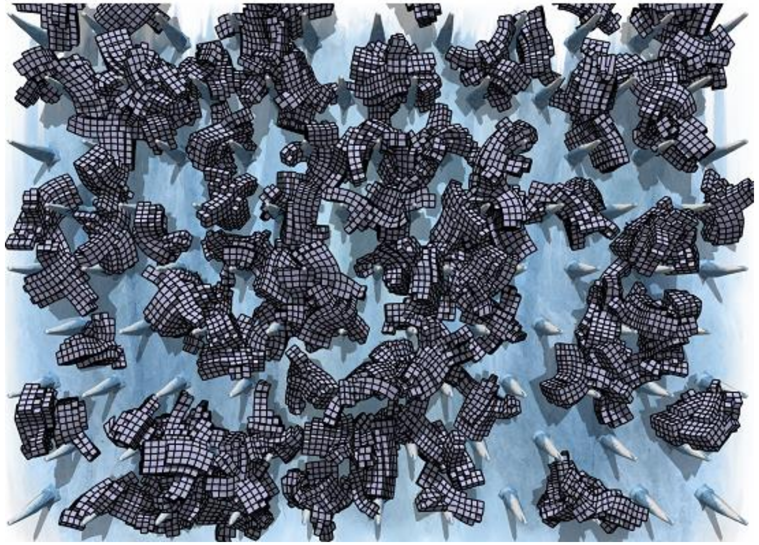
$$\mathbf{R} = \mathbf{A}\,(\mathbf{A}^\intercal\mathbf{A})^{-1/2}$$

- $\mathbf{A}^\intercal\mathbf{A}$ を対角化

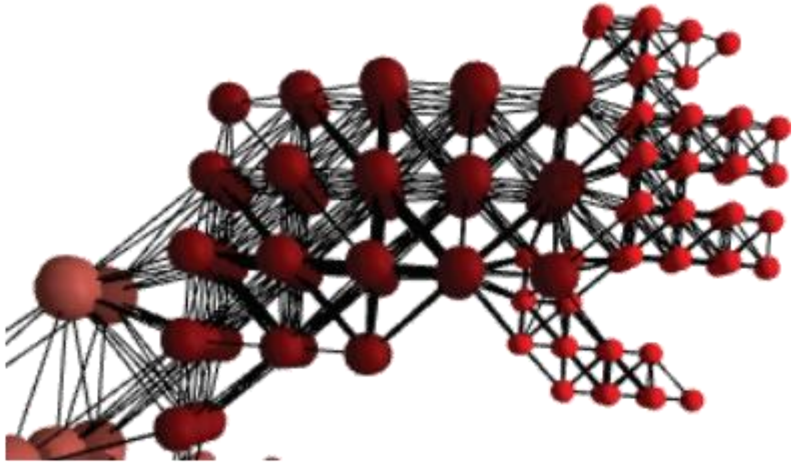
局所領域ごとの Shape Matching

- より複雑な変形を実現
- 高速化などの拡張

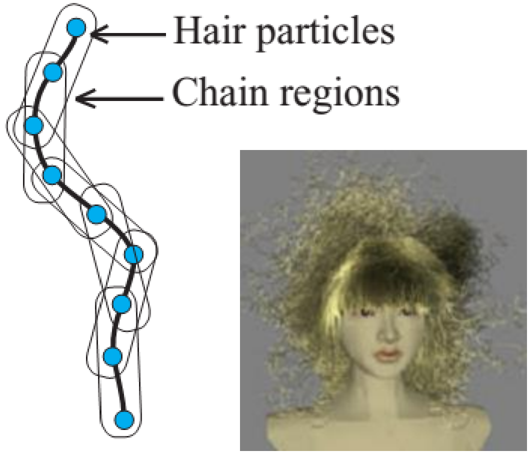




ボクセル格子による局所領域



Octree による局所領域

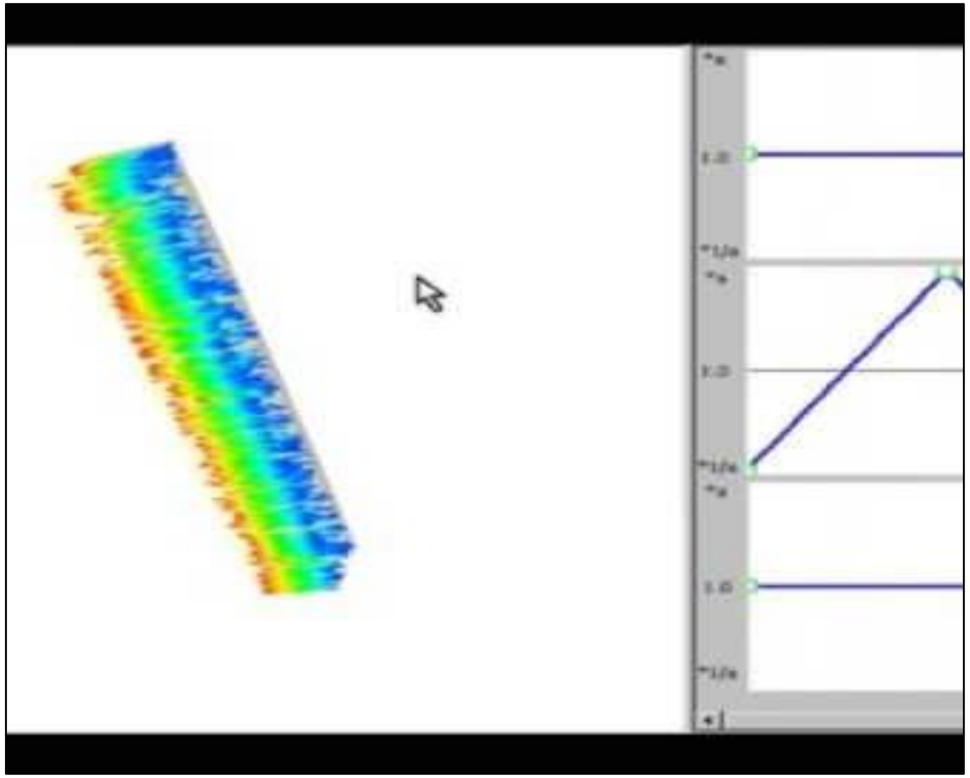


一次元的構造を使った
髪の毛のアニメーション

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]

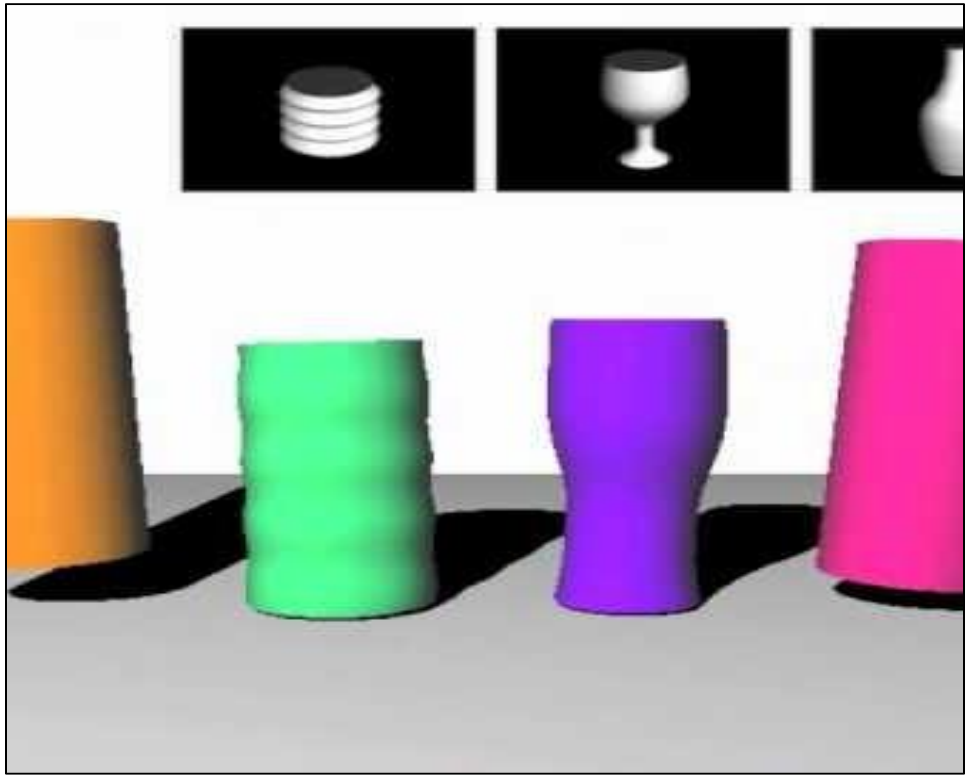
応用：局所領域の rest shape を変化させる

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

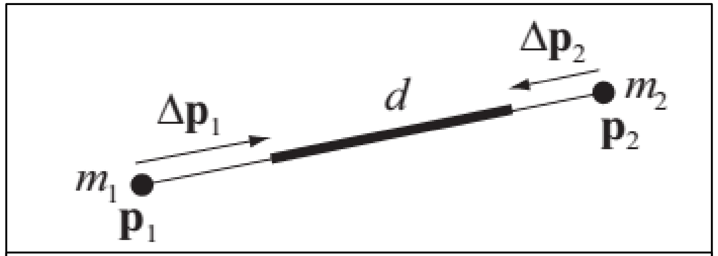
ProcDef; local-to-global deformation for skeleton-free character animation [Ijiri PG09]
Real-Time Example-Based Elastic Deformation [Koyama SCA12]

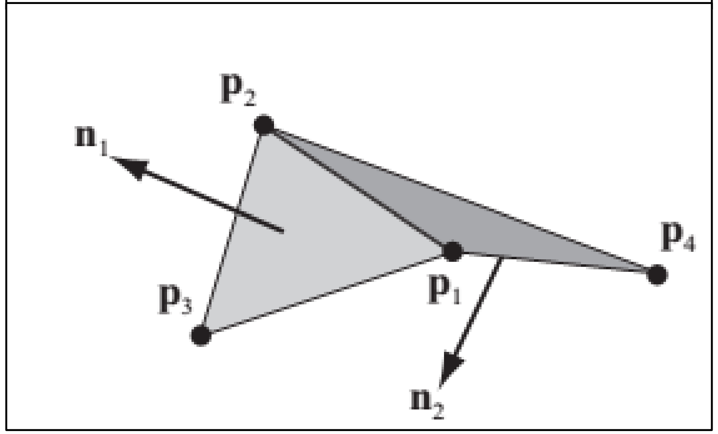
PBD 計算の流れ

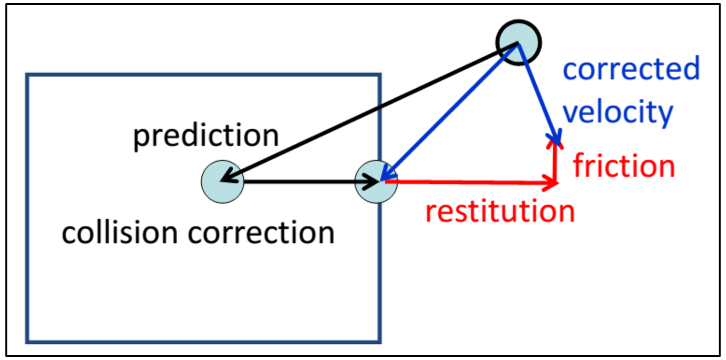
- 入力：初期位置 $\mathbf{x}_0$, 初期速度 $\mathbf{v}_0$

- フレーム毎の処理：

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







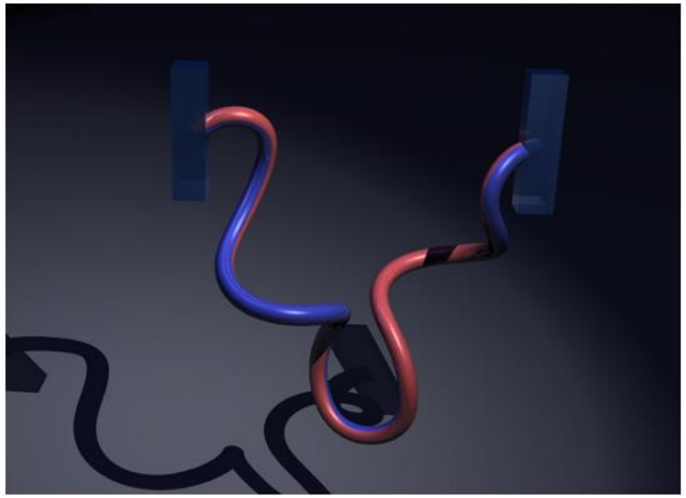
(衝突と摩擦の扱いについて、あまり理解できていない)

http://www.csee.umbc.edu/csee/research/vangogh/I3D2015/matthias_muller_slides.pdf

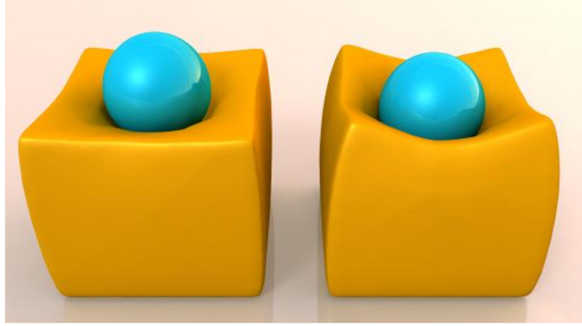
PBD に導入できる様々な制約

体積制約

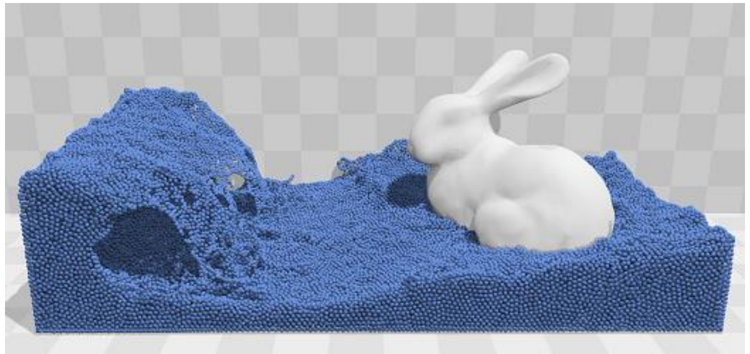
布の伸び幅制約



紐の捻り制約



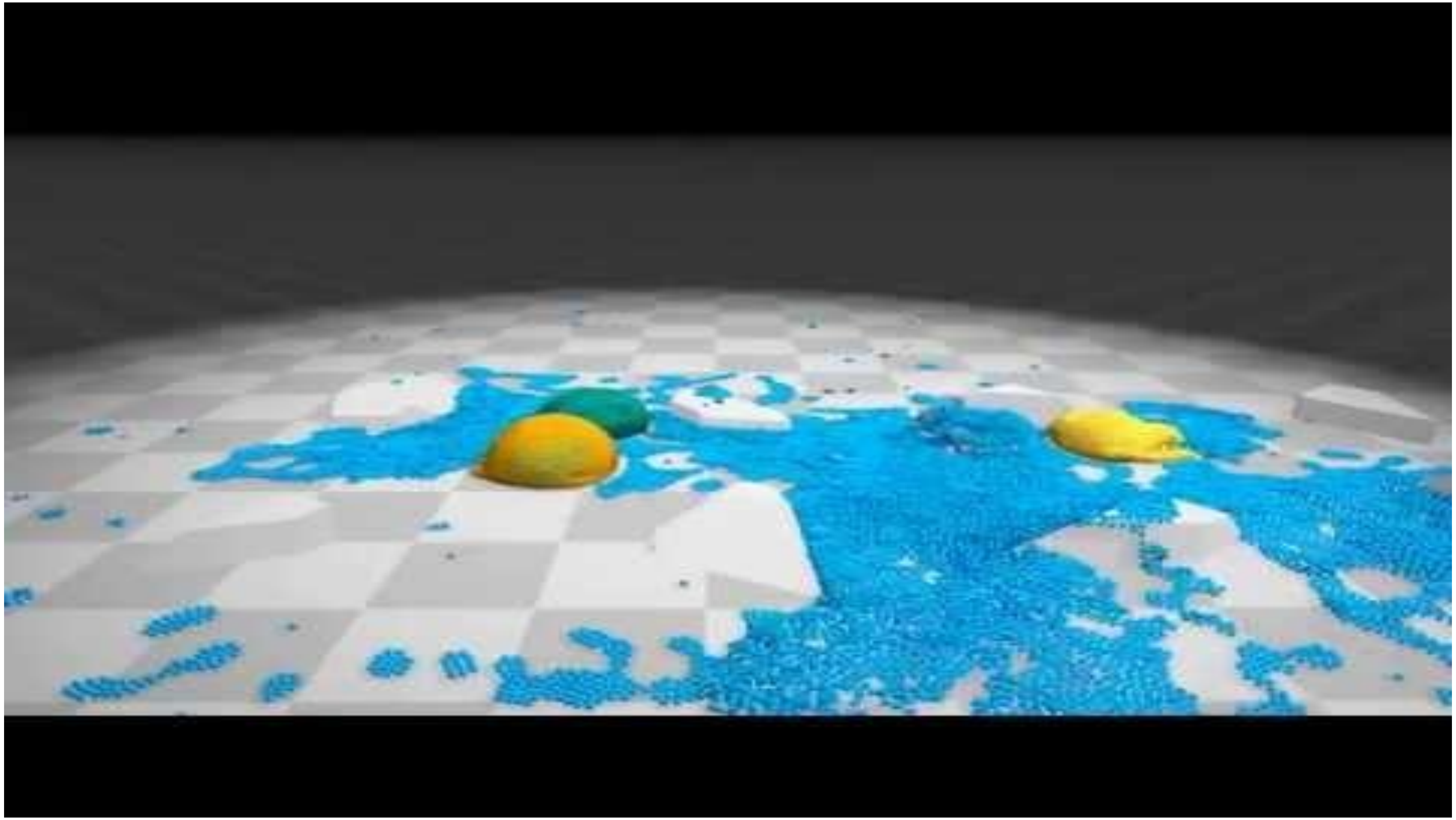
連続体の歪み制約



粒子の密度制約

Robust Real-Time Deformation of Incompressible Surface Meshes [Diziol 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

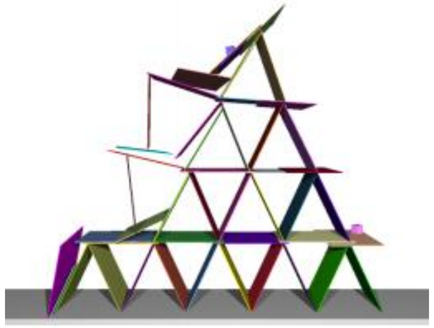


https://www.youtube.com/watch?v=z6dAahLUbZg

- NVIDIA が SDKを公開！

Unified Particle Physics for Real-Time Applications [Macklin SIGGRAPH14]

衝突判定

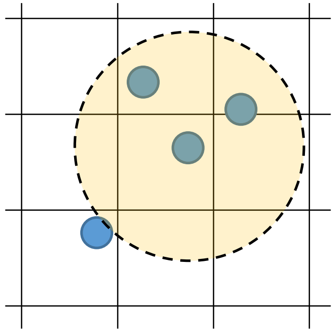


[Kaufman08]

[Harmon09]

[Zheng12]

- 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
- 今年発表される PBD 用の手法



[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