

Position Based Dynamics Combo

物理シミュレーション関連の最新論文実装

株式会社ポリフォニー・デジタル
中川展男

2017年8月31日

はじめに

- 本セッションのスライド、論文の日本語訳、ソースコードは全て github で公開しております
 - <https://github.com/nobuo-nakagawa/>
- 本スライド内で引用した論文の図や動画の著作権は全て論文著者に帰属します
- 本セッションはニコニコ生放送でライブ配信されます
 - <http://live.nicovideo.jp/gate/lv304038400>

本セッションの目的

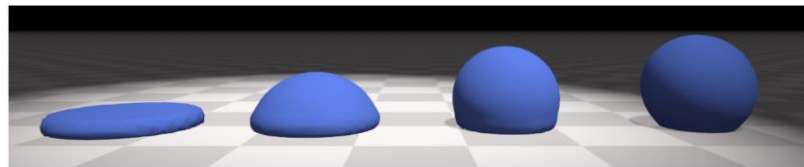
- Position Based Dynamics(PBD)
 - 基礎から最新研究まで理解し追実装できるように
- Combo?
 - Yakitori Combo 等、日本語で言うと“盛り合わせ”
 - 2014年～2016年の手法の発展を盛り合わせで学ぶ



[MNTM14]



[MNTM15]



[MMN16]

なぜ学ぶのか

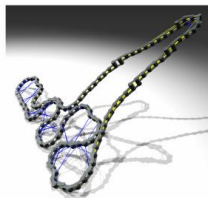
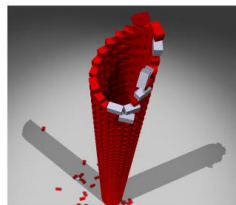
- Position Based Dynamics(PBD)

- ビデオゲーム・映像業界向けの高速で堅牢な手法

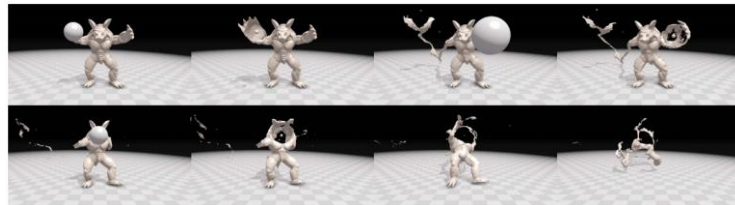
- NVIDIA PhysX
 - Houdini
 - Obi Unified particle physics for Unity

- 毎年多くの論文発表が行われているトピック

- Long Range Constraints for Rigid Body Simulations(2017)
 - Real-time simulation of large elasto-plastic deformation with shape matching(2016)

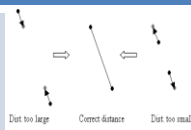
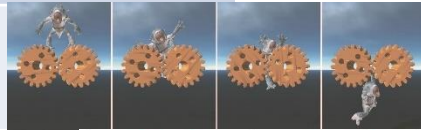
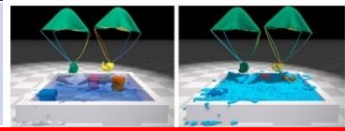


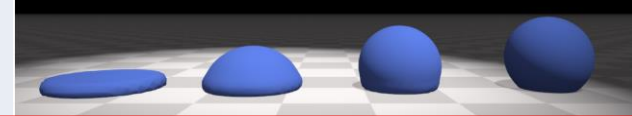


[MNMS17]



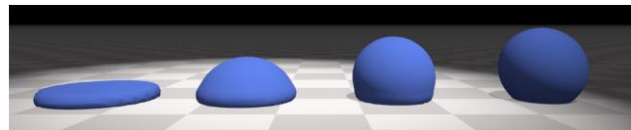
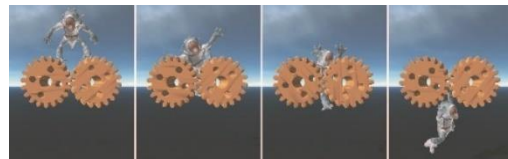
[NMM16]

関連論文の発展の歴史

発表年	論文タイトル	
2001	Advanced character physicx	
2007	Position based dynamics	
2014	Unified Particle Physics for Real-Time Applications	
2014	Strain based dynamics	
2015	Air meshes for robust collision handling	
2016	XPBD: position-based simulation of compliant constrained dynamics	

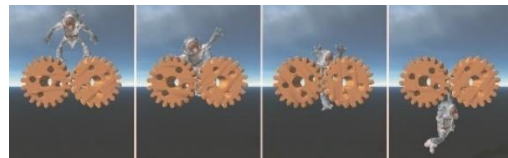
内容

- Position Based Dynamics 入門
- 論文紹介
 - Strain Based Dynamics
 - Air Meshes for Robust Collision Handling
 - XPBD Position-Based Simulation of Compliant Constrained Dynamics
- 考察、質疑応答



内容

- Position Based Dynamics 入門



- 論文紹介

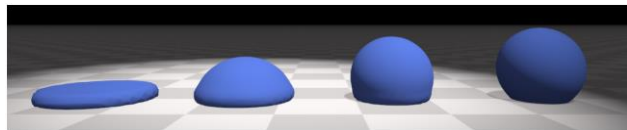
- Strain Based Dynamics



- Air Meshes for Robust Collision Handling

- XPBD Position-Based Simulation of Compliant Constrained Dynamics

- 考察、質疑応答



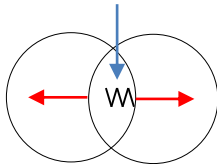
Position Based Dynamics 入門

- 物理シミュレーションの位置更新手法

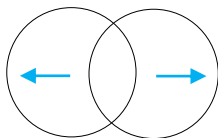
- 従来手法: 力(Force)ベース手法

- バネ定数 k が小さいと柔らかい動き
 - バネ定数 k が大きいと硬い動き(押し出しすぎることも)

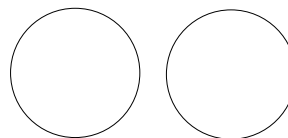
ペナルティ法: 仮想的なバネを考え侵入量に応じて押し出す



侵入量に応じて力発生



力が速度を更新



速度が位置を修正

フックの法則: $F_{\text{spring}} = kd$
 k : バネ定数, d : 侵入距離

運動方程式: $F=ma$
 m : 質量, a : 加速度
速度: $v = v_0 + at$
 v_0 : 初速度, t : 時間

位置: $x = x_0 + vt$
 x_0 : 初期位置

衝突

→ 力を求める
→ 速度を求める
→ 位置が求まる

Position Based Dynamics 入門

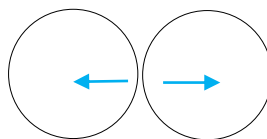
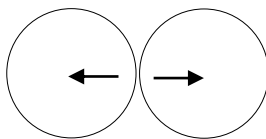
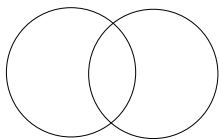
• 物理シミュレーションの位置更新手法

– Position Based Dynamics(PBD)

- 位置を直接修正するから Position Based Dynamics
- 必要なだけ押し出す→押し出しすぎることはなくなる
- 無条件で安定、幾何的な拘束条件なので物理的ではない(?)

衝突

→修正位置を求める
→速度は間接的に求まる



押し出しとか(?)
コリジョンとか(?)
→非侵入拘束

侵入を検出するだけ 侵入解決するように位置の修正 速度の更新

最初は何も考えずに動かすが、この値ではまだ位置を更新しない。
予測位置: $x_{predict} = x_0 + vt$

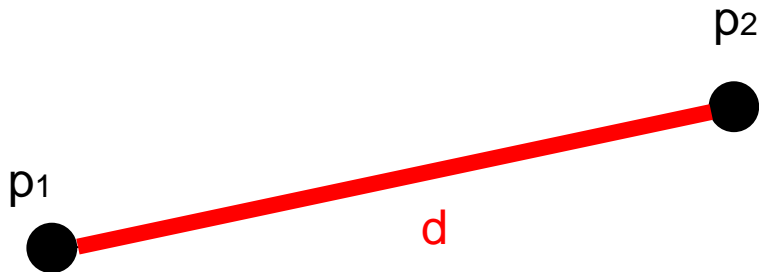
拘束(constraint)を考え
 $x_{predict}$ を修正し
修正位置: $x_{correct}$ を求める
位置: $x \leftarrow x_{correct}$

修正速度: $v_{correct} = (x - x_0) / t$
速度: $v \leftarrow v_{correct}$

Position Based Dynamics 入門

- **Position Based Dynamics(PBD)** [MHHR07]
 - 拘束(**C**onstraint)
 - 伸び(stretch) by Jakobsen [Jak01]

$$C_{stretch}(\mathbf{p}_1, \mathbf{p}_2) = |\mathbf{p}_1 - \mathbf{p}_2| - d$$



p1, p2: パーティクル位置

d: 定常状態(伸びたり縮んだりしない)の長さ

Position Based Dynamics 入門

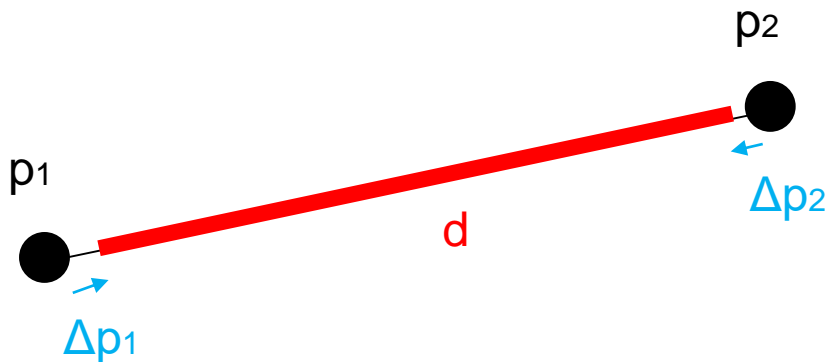
- シンプルな伸び(stretch)拘束(Constraint)

$$C_{stretch}(\mathbf{p}_1, \mathbf{p}_2) = |\mathbf{p}_1 - \mathbf{p}_2| - d$$

$\mathbf{p}_1, \mathbf{p}_2$: パーティクル位置

d : 定常状態の長さ

$\Delta \mathbf{p}_1, \Delta \mathbf{p}_2$: パーティクル位置修正値



Position Based Dynamics 入門

- シンプルな伸び(stretch)拘束(Constraint)

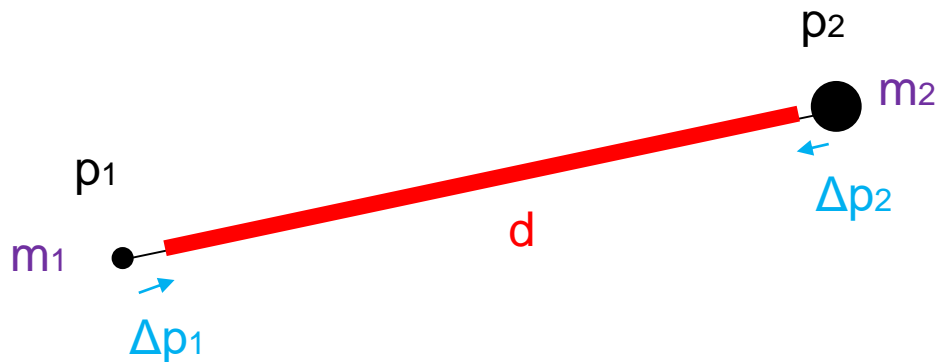
$$C_{stretch}(\mathbf{p}_1, \mathbf{p}_2) = |\mathbf{p}_1 - \mathbf{p}_2| - d$$

$\mathbf{p}_1, \mathbf{p}_2$: パーティクル位置

d : 定常状態の長さ

$\Delta \mathbf{p}_1, \Delta \mathbf{p}_2$: パーティクル位置修正値

m_1, m_2 : パーティクル質量



$$\Delta \mathbf{p}_1 = -\frac{w_1}{w_1 + w_2} (|\mathbf{p}_1 - \mathbf{p}_2| - d) \frac{\mathbf{p}_1 - \mathbf{p}_2}{|\mathbf{p}_1 - \mathbf{p}_2|}$$

$$w_i = \frac{1}{m_i}$$

$$\Delta \mathbf{p}_2 = +\frac{w_2}{w_1 + w_2} (|\mathbf{p}_1 - \mathbf{p}_2| - d) \frac{\mathbf{p}_1 - \mathbf{p}_2}{|\mathbf{p}_1 - \mathbf{p}_2|}$$

Position Based Dynamics 入門

- **Position Based Dynamics(PBD)** [MHHR07]

- 拘束(Constraint)

- 伸び(stretch)拘束 by Jakobsen [Jak01]

- PBD として一般化 [MHHR07]

- $C(\mathbf{p}) = 0 \Leftrightarrow$ 拘束が満たされた(ひもが伸び縮みしない)

- $C(\mathbf{p} + \Delta\mathbf{p}) \approx C(\mathbf{p}) + \nabla_{\mathbf{p}} C(\mathbf{p}) \cdot \Delta\mathbf{p} = 0$ (1次テイラー展開)

ちよつと動いた \nearrow

元の位置

勾配 \cdot ちよつと動いた

$$w_i = 1/m_i$$

s: scaling factor

$$\Delta\mathbf{p}_i = -s \nabla_{\mathbf{p}_i} C(\mathbf{p}_1, \dots, \mathbf{p}_n), \quad (6)$$

$$s = \frac{C(\mathbf{p}_1, \dots, \mathbf{p}_n)}{\sum_j w_j |\nabla_{\mathbf{p}_j} C(\mathbf{p}_1, \dots, \mathbf{p}_n)|^2} \quad (8)$$

$$s = \frac{C(\mathbf{p}_1, \dots, \mathbf{p}_n)}{\sum_j |\nabla_{\mathbf{p}_j} C(\mathbf{p}_1, \dots, \mathbf{p}_n)|^2} \quad (7)$$

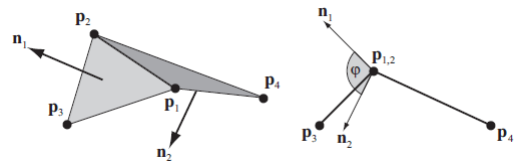
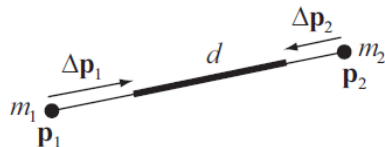
$$\Delta\mathbf{p}_i = -s w_i \nabla_{\mathbf{p}_i} C(\mathbf{p}_1, \dots, \mathbf{p}_n). \quad (9)$$

Position Based Dynamics 入門

- **Position Based Dynamics(PBD)** [MHHR07]
 - 様々な拘束(**C**onstraint)
 - 伸び(stretch) by Jakobsen [Jak01]
 - 曲げ(bend)

$$C_{stretch}(\mathbf{p}_1, \mathbf{p}_2) = |\mathbf{p}_1 - \mathbf{p}_2| - d \quad \rightarrow \text{Jakobsen}$$

$$C_{bend}(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4) = \arccos \left(\frac{(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)}{|\mathbf{p}_2 - \mathbf{p}_1| |\mathbf{p}_3 - \mathbf{p}_1|} \cdot \frac{(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_4 - \mathbf{p}_1)}{|\mathbf{p}_2 - \mathbf{p}_1| |\mathbf{p}_4 - \mathbf{p}_1|} \right) - \varphi_0$$



[MHHR07]

Position Based Dynamics 入門

- **Position Based Dynamics(PBD)** [MHHR07]

- 処理の流れ

1. 前ステップの位置 x_n から予測位置 x_{predict} を求める

2. 修正位置 x_0 を x_{predict} で初期化

3. **for(i<イテレーション回数)**

- 3.1 Δx を求める

- 3.2 $x_{i+1} = x_i + \Delta x$ で更新する

4. 位置 $x_{n+1} = x_i$ が確定する

5. 速度 $v_{n+1} = (x_{n+1} - x_n) / \Delta t$ で間接的に求まる

Δx が十分に小さくなくてもイテレーションを打ち切る

Position Based Dynamics 入門

- 物理シミュレーションの位置更新手法

- Position Based Dynamics(PBD)

- 位置を直接修正するから Position Based Dynamics
 - 必要だけ押し出す→押し出しすぎることはなくなる
 - 無条件で安定、幾何的な拘束条件なので物理的ではない(?)



侵入を検出するだけ



侵入解決するように位置の修正



速度の更新

最初は何も考えずに動かすが、この値ではまた位置を更新しない。
予測位置 $x_{\text{predict}} = x_i + v_i$

拘束(constraint)を考え x_{connect} を求め修正位置 x_{correct} を求める
位置 $x \leftarrow x_{\text{correct}}$

修正速度 $v_{\text{correct}} = (x - x_i) / t$
速度 $v \leftarrow v_{\text{correct}}$

衝突
→修正位置を求める
→速度は間接的に求まる

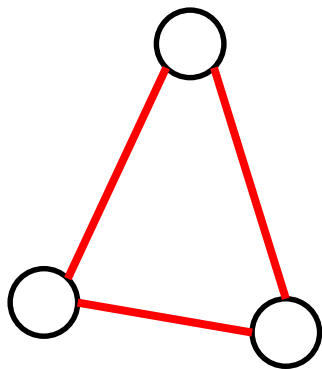
押し出しとか(?)
コリジョンとか(?)
→非侵入拘束

Position Based Dynamics 入門

- 伸び(stretch)拘束(**C**onstraint) 3本の三角形

$$C_{stretch}(\mathbf{p}_1, \mathbf{p}_2) = |\mathbf{p}_1 - \mathbf{p}_2| - d$$

p1, p2: パーティクル位置
d: 赤い線の長さ

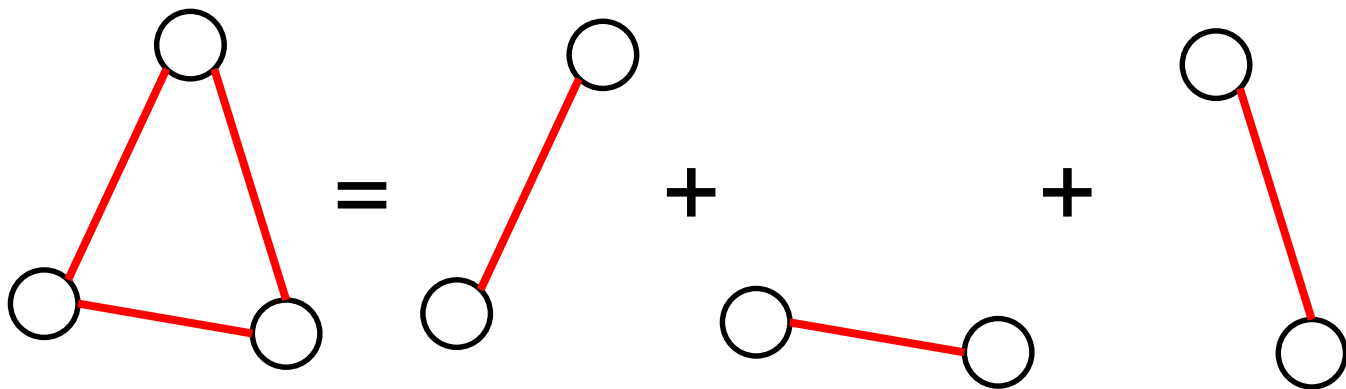


Position Based Dynamics 入門

- 伸び(stretch)拘束(**C**onstraint) 3本の三角形

$$C_{stretch}(\mathbf{p}_1, \mathbf{p}_2) = |\mathbf{p}_1 - \mathbf{p}_2| - d$$

$\mathbf{p}_1, \mathbf{p}_2$: パーティクル位置
 d : 赤い線の長さ



Position Based Dynamics 入門

- シンプルな伸び(stretch)拘束(Constraint)

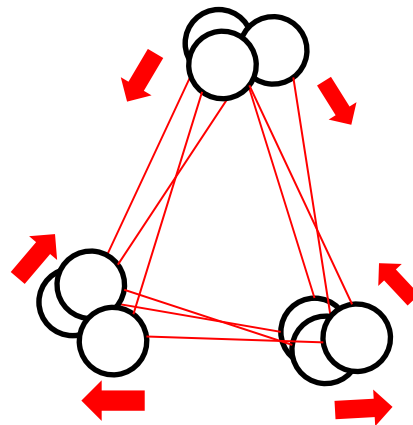
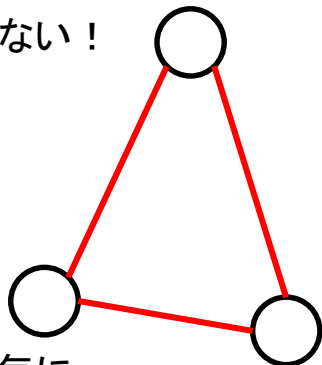
$$C_{stretch}(\mathbf{p}_1, \mathbf{p}_2) = |\mathbf{p}_1 - \mathbf{p}_2| - d$$

1つの拘束を満たすと
他の2つの拘束を満たさない！

では、どうすれば？

順番に拘束を解く処理を
繰り返していきましょう

繰り返し(イテレーション)毎に
3つの拘束全てを徐々に満たすようになっていきます
これを「繰り返し法」と呼びます



Position Based Dynamics 入門

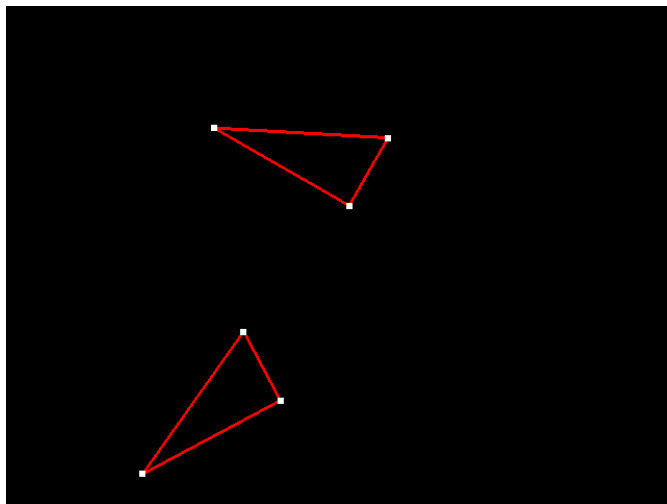
- **Position Based Dynamics(PBD)** [MHHR07]
 - 修正位置: x_{correct} の求め方
 - ガウス・ザイデル法
 - 全てのパーティクルを拘束に従って順に繰り返し動かす
 - 処理順で結果が変わるが一般に収束が早い。CPU向け。
 - ヤコビ法
 - 全てのパーティクルを拘束に従ってまとめて動かす
 - GPU 並列化向き

Position Based Dynamics 入門

- デモ
 - 伸び(stretch)拘束
 - 非侵入拘束(三角形同士の押し出し、壁の押し出し)

白い点には、重力と拘束条件だけを適用することで回転表現ができる。

このデモでは、三角形の摩擦表現も実装されている



Position Based Dynamics 入門

- まとめ

- 従来手法(Force Based Dynamics)

- 力を求める→速度を求める→位置を求める

- Position Based Dynamics(PBD) 手法

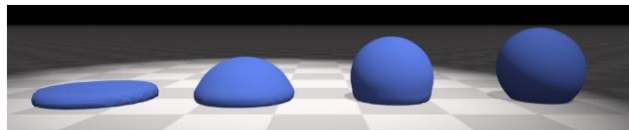
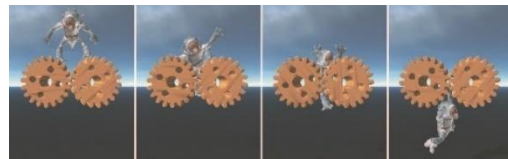
- 予測位置→位置の修正→位置、速度の確定
 - 高速、堅牢、シンプルな実装
 - 陰解法の近似方法と見ることもできる

- 実装

- Jan Bender <https://github.com/InteractiveComputerGraphics/PositionBasedDynamics>

内容

- Position Based Dynamics 入門
- 論文紹介
 - Strain Based Dynamics
 - Air Meshes for Robust Collision Handling
 - XPBD Position-Based Simulation of Compliant Constrained Dynamics
- 考察、質疑応答



Strain Based Dynamics

- 解説の流れ
 - 動画
 - 基本概念
 - 実装
 - 応用例
 - 長所、短所

Strain Based Dynamics

- 解説の流れ
 - 動画
 - 基本概念
 - 実装
 - 応用例
 - 長所、短所

Strain Based Dynamics

- 解説の流れ
 - 動画
 - 基本概念
 - 実装
 - 応用例
 - 長所、短所

Strain Based Dynamics

- 拘束条件

- 四面体(tetrahedral)拘束

- 伸び(stretch)拘束は2点間の距離だけだったが
 - 4点まとめた拘束のイメージ
 - グリーン(Green)の歪み(strain)テンソル

- 三角形(triangle)拘束

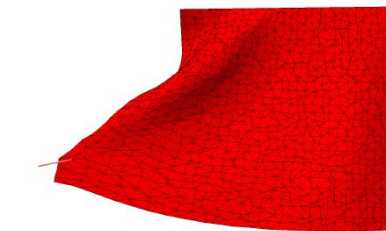
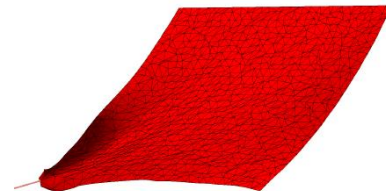
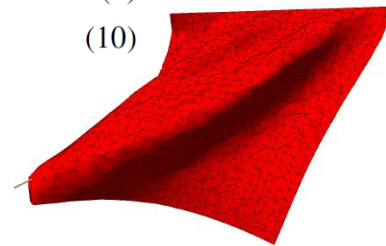
- 同様に3点まとめた拘束

- 異方性

- x方向, y方向 で歪みを変える

$$C(p0, p1, p2, p3) = S_{ii} - s_i^2 \quad (9)$$

$$C(p0, p1, p2, p3) = S_{ij} \quad i < j, \quad (10)$$



物体の基準(初期)状態の
単位長さあたりに物体内の
物質点がどれだけ変位するか

Strain Based Dynamics

- 拘束条件

- 体積保存(area)拘束

- 面積保存(volume)拘束

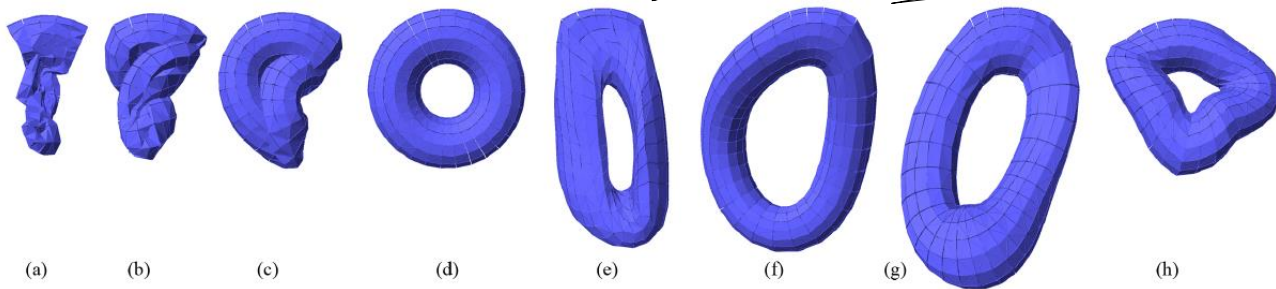
- $C_{\text{volume}}(p_0, p_1, p_2, p_3) = \det(F) - 1$

- $C_{\text{area}}(p_0, p_1, p_2) = \det(F) - 1$

F: 変形勾配, $\det(F)$: 体積変化率

体積保存優先

体積保存弱め、トーラス主軸方向の剛性値強め



Strain Based Dynamics

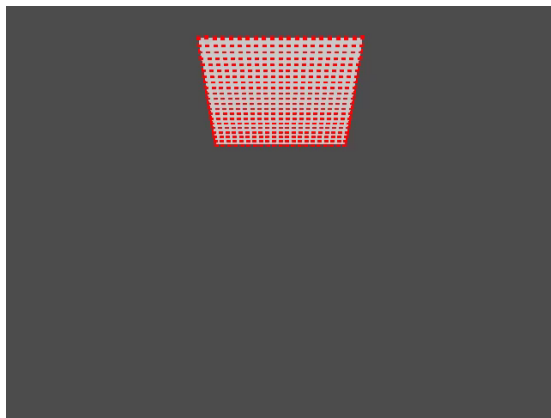
- マテリアル座標系
 - 2D(triangle)の場合は、三角形の頂点のテクスチャ座標(u, v) は伸びが考慮されていない
 - グローバル座標から三角形の座標への射影を考える
- 3D(tetrahedral)の場合
 - 重心座標のようなものを考える
- メッシュが不均一でも問題なくなる

Strain Based Dynamics

- 解説の流れ
 - 動画
 - 基本概念
 - 実装
 - 応用例
 - 長所、短所

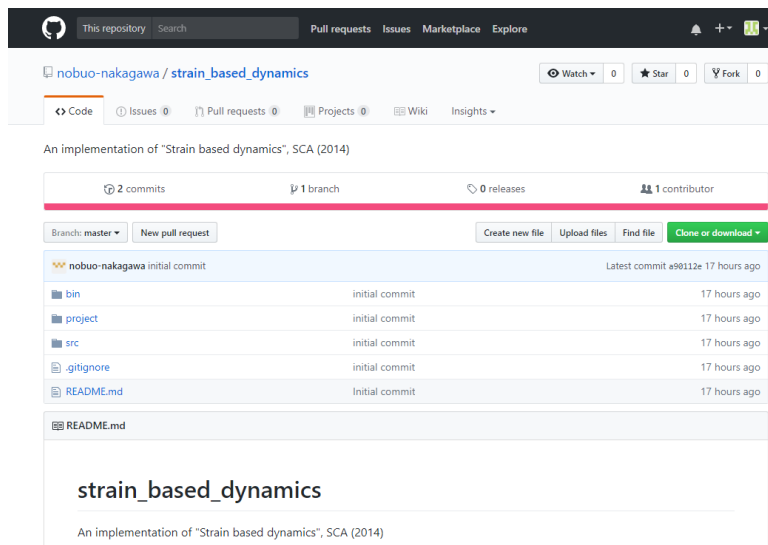
Strain Based Dynamics

- 実装
 - C++, OpenGL, glut, glm で実装(win, osx)
- デモ



Strain Based Dynamics

- Pull Request 大歓迎です
 - Game Engine Gems 3 [Muhammad 16] を参考
 - https://github.com/nobuo-nakagawa/strain_based_dynamics



Strain Based Dynamics

- 解説の流れ
 - 動画
 - 基本概念
 - 実装
 - 応用例
 - 長所、短所

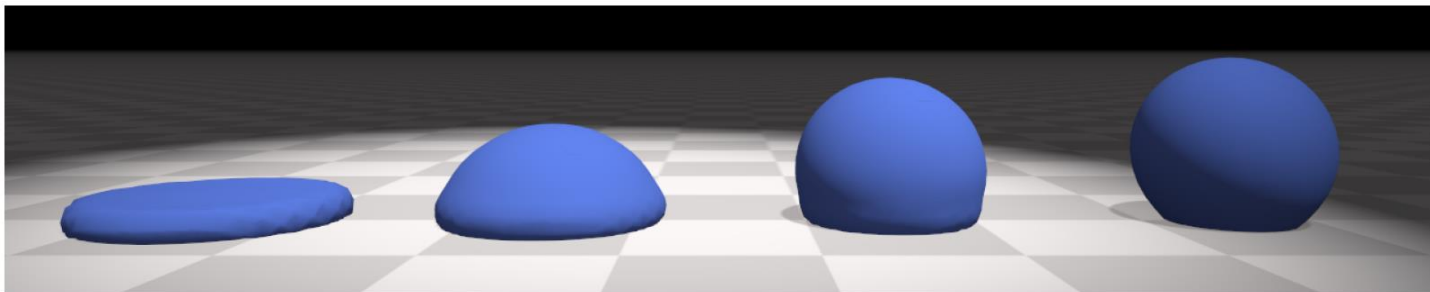
Strain Based Dynamics

- 応用例ではなく、同時期研究
 - Projective Dynamics: Fusing Constraint Projections for Fast Simulation [SSTLM14]
 - global / local solve 物理的に正確な方向



Strain Based Dynamics

- 応用例
 - XPBD: position-based simulation of compliant constrained dynamics
 - 後ほどお話します



Strain Based Dynamics

- 解説の流れ
 - 動画
 - 基本概念
 - 実装
 - 応用例
 - 長所、短所

Strain Based Dynamics

- 長所

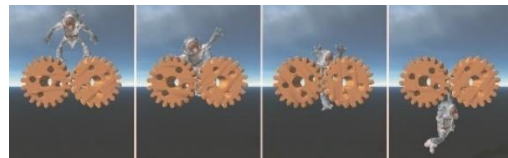
- 歪み(strain)ベースの拘束条件により高品質なクロスや弾性体シミュレーションが可能
- 異方性の歪みがきちんと扱える
- 体積保存性や曲がりやすさなどを細かく制御可能

- 短所

- FEM ほどの正確さはない
- 計算コスト増加

內容

- Position Based Dynamics 入門



- 論文紹介

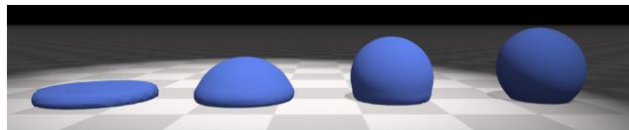
- Strain Based Dynamics



- Air Meshes for Robust Collision Handling

- XPBD Position-Based Simulation of Compliant Constrained Dynamics

- 考察、質疑応答



Air Meshes for Robust Collision Handling

- 解説の流れ
 - 動画
 - 基本概念
 - 実装方法
 - 応用例
 - 長所、短所

Air meshes for robust collision handling

- 解説の流れ
 - 動画
 - 基本概念
 - 実装方法
 - 応用例
 - 長所、短所

Air meshes for robust collision handling

- 解説の流れ
 - 動画
 - 基本概念
 - 実装方法
 - 応用例
 - 長所、短所

Air meshes for robust collision handling

- アルゴリズム

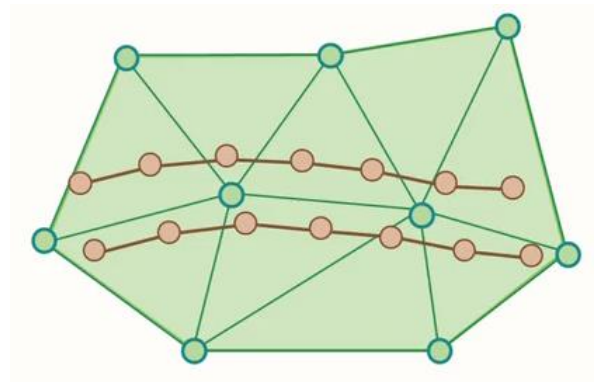
- 1 シミュレーション前

- 1.1 オブジェクトの周りの Air mesh 生成

- 2 シミュレーション中

- 2.1 Air mesh が反転しないような Unilateral Volume Constraint を適用

- 2.2 大変形の際はメッシュ最適化



[MNTM15]

Air meshes for robust collision handling

- アルゴリズム

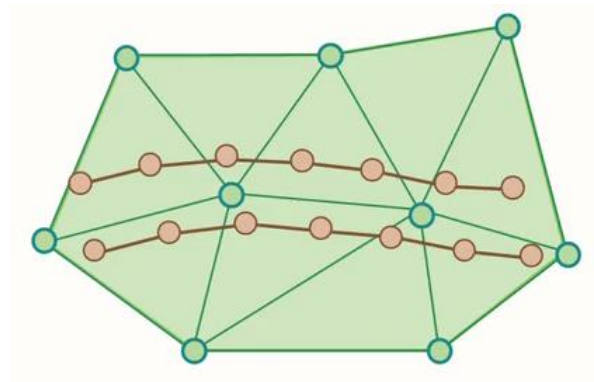
- 1 シミュレーション前

- 1.1 オブジェクトの周りの Air mesh 生成

- 2 シミュレーション中

- 2.1 Air mesh が反転しないような Unilateral Volume Constraint を適用

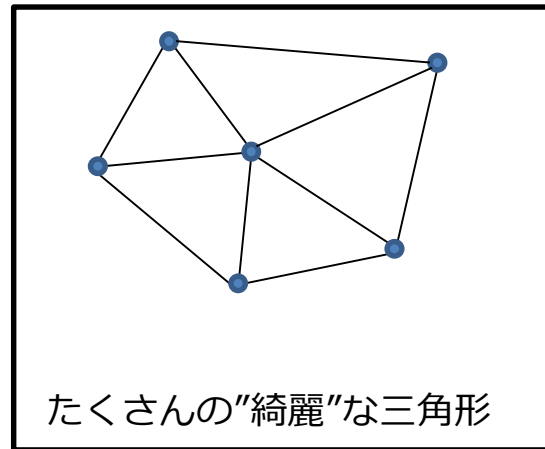
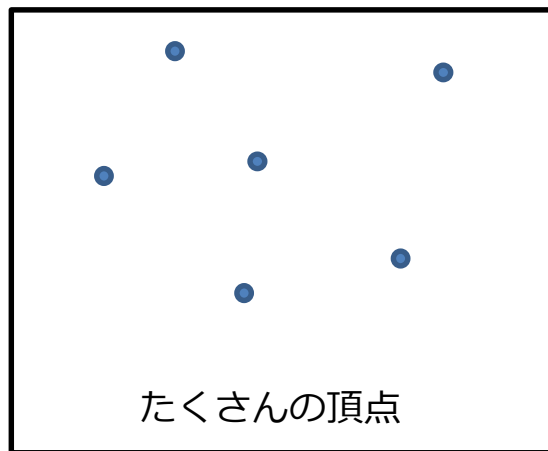
- 2.2 大変形の際はメッシュ最適化



[MNTM15]

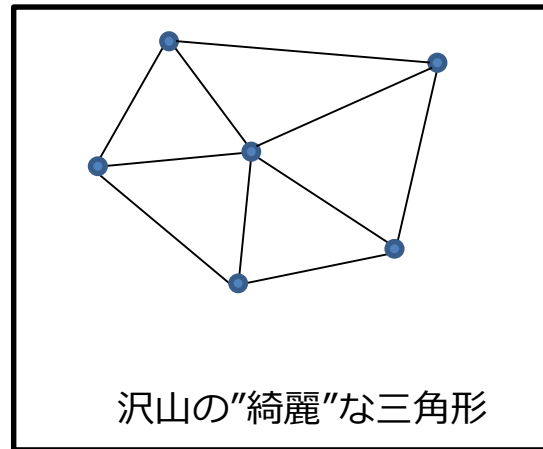
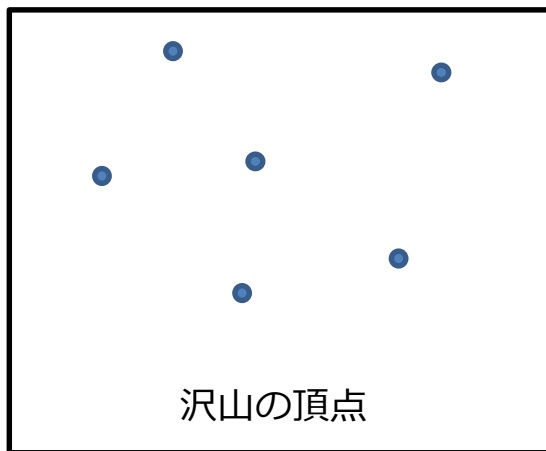
Air meshes for robust collision handling

- オブジェクトの周りの Air mesh 生成
 - ドロネー三角形分割(Delaunay triangulation)



Air meshes for robust collision handling

- ドロネー三角形分割(Delaunay triangulation)
 - 沢山の頂点 → 沢山の重ならない綺麗な三角形
 - 綺麗とは、すべての可能な三角形分割の中で最小の内角が最大となるもの



Air meshes for robust collision handling

- ドロネー三角形分割(Delaunay triangulation)
 - 論文内では、TetGen [Hang 15] が紹介されている
 - 私の追実装では、Paul Bourke さんの手法[Paul 89]
 - C, C++, C# 他非常に多くの言語の実装例が紹介されている
 - <http://paulbourke.net/papers/triangulate/>
 - 既存の信頼できるライブラリを用いる

Air meshes for robust collision handling

- アルゴリズム

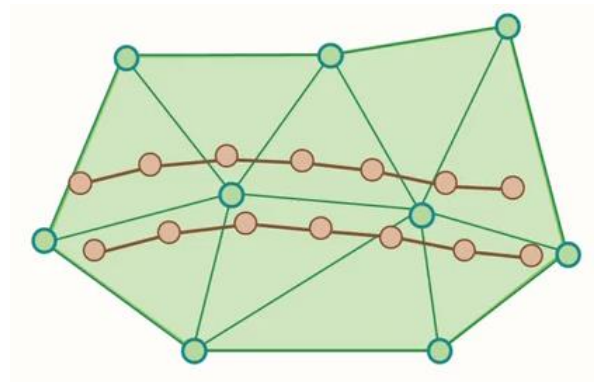
- 1 シミュレーション前

- 1.1 オブジェクトの周りの Air mesh 生成

- 2 シミュレーション中

- 2.1 Air mesh が反転しないような Unilateral Volume Constraint を適用

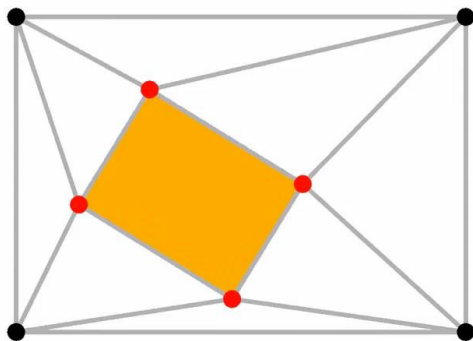
- 2.2 大変形の際はメッシュ最適化



[MNTM15]

Air meshes for robust collision handling

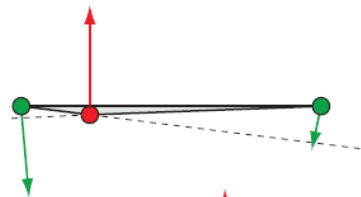
- Unilateral Area / Volume Constraint
 - 符号付き面積(2D)/体積(3D)が負にならない拘束



[MNTM15]

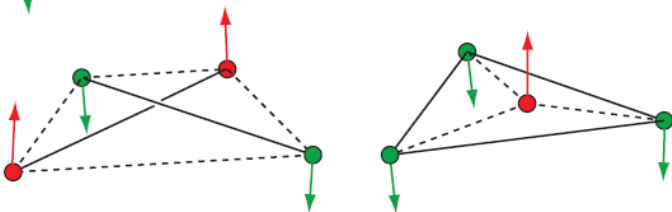
Air meshes for robust collision handling

- 拘束条件(constraint)
 - 符号付き面積(2D)体積(3D)が正を維持する拘束
 - 単一方向(unilateral)負になったときのみ働く



$$C_{\text{air}} = |(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)| \geq 0 \quad (1)$$

$$C_{\text{air}} = \det[\mathbf{p}_2 - \mathbf{p}_1, \mathbf{p}_3 - \mathbf{p}_1, \mathbf{p}_4 - \mathbf{p}_1] \geq 0 \quad (2)$$



[MNTM15]

Air meshes for robust collision handling

- アルゴリズム

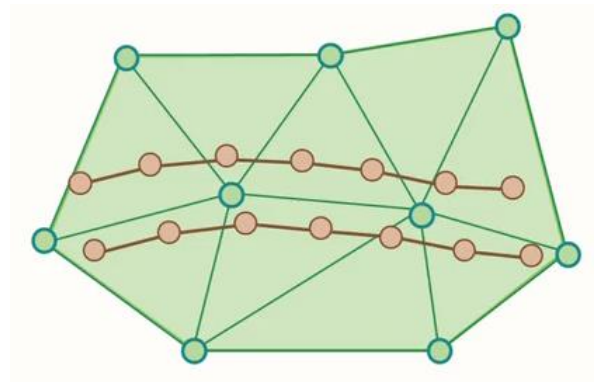
- 1 シミュレーション前

- 1.1 オブジェクトの周りの Air mesh 生成

- 2 シミュレーション中

- 2.1 Air mesh が反転しないような Unilateral Volume Constraint を適用

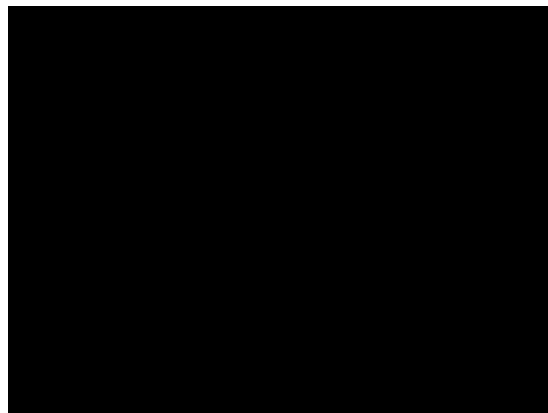
- 2.2 大変形の際はメッシュ最適化



[MNTM15]

Air meshes for robust collision handling

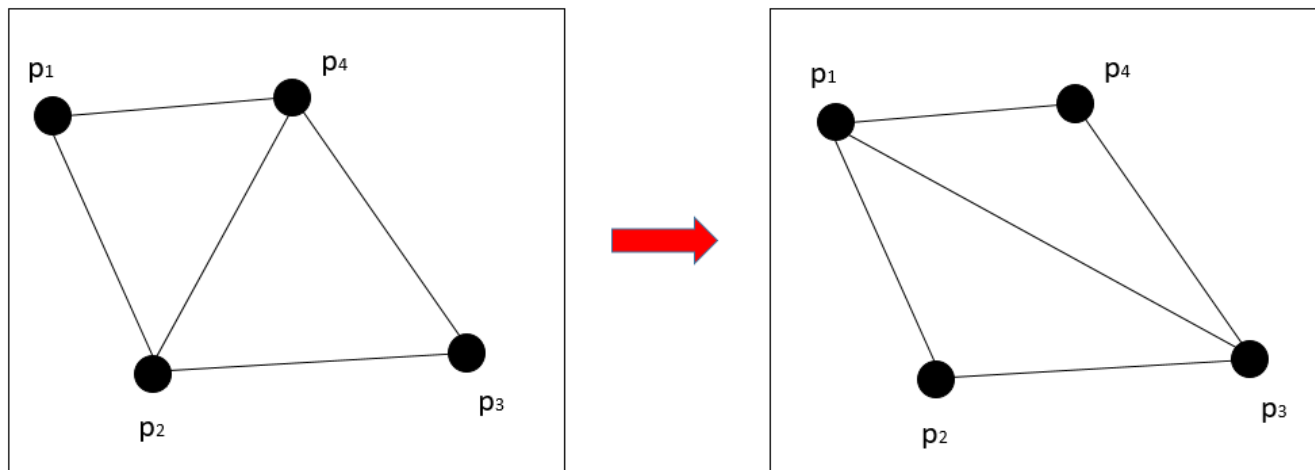
- メッシュ最適化
 - 回転が大きいときロックする
 - つぶれないように辺を入れ替える(edge flip)



[MNTM15]

Air meshes for robust collision handling

- メッシュ最適化
 - 回転が大きいときロックする
 - つぶれないように辺を入れ替える(edge flip)



Air meshes for robust collision handling

- メッシュ最適化

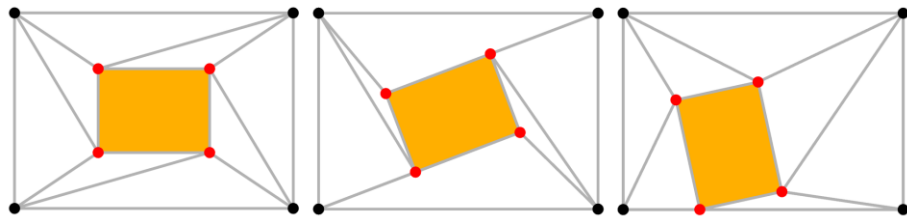
- 辺を入れ替える条件

- (3): 三角形の品質
 - (4): 四面体の品質

$$q_{\text{triangle}} = \frac{4}{\sqrt{3}} \frac{A}{l_1^2 + l_2^2 + l_3^2} \quad (3)$$

- 品質が下がると再分割

$$q_{\text{tetrehedron}} = \frac{12}{\sqrt{2}} \frac{V}{l_{\text{rms}}^3}, \quad l_{\text{rms}} = \sqrt{\frac{l_1^2 + l_2^2 + l_3^2 + l_4^2 + l_5^2 + l_6^2}{6}}, \quad (4)$$



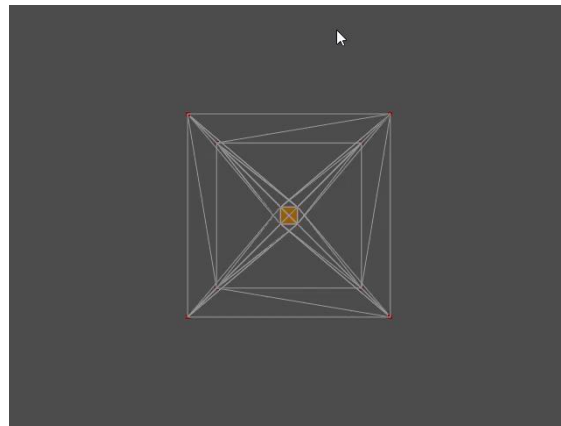
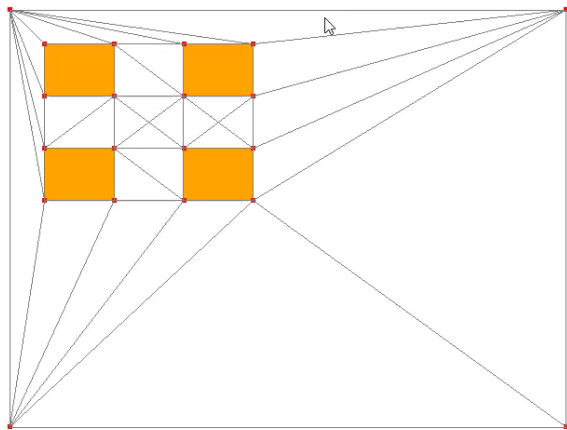
[MNTM15]

Air Meshed for Robust Collision Handling

- 解説の流れ
 - 動画
 - 基本概念
 - 実装方法
 - 応用例
 - 長所、短所

Air Meshed for Robust Collision Handling

- 実装
 - C++, OpenGL, glut, glm で実装(win, osx)
- デモ(2D版, 3D版)



Air Meshed for Robust Collision Handling

- Pull Request 大歓迎です

- https://github.com/nobuo-nakagawa/air_mesh_2d

The screenshot shows the GitHub repository page for `nobuo-nakagawa / air_mesh_2d`. At the top, there are buttons for Watch (0), Star (0), and Fork (0). Below this is a navigation bar with links for Code, Issues (0), Pull requests (0), Projects (0), Wiki, and Insights. The repository description is "An implementation of 'Air Meshes for Robust Collision Handling', SIGGRAPH (2015)". Below the description, there are statistics: 2 commits, 1 branch, 0 releases, and 1 contributor. A pink bar separates the repository header from the file list. Below the bar, there are buttons for "Branch: master", "New pull request", "Create new file", "Upload files", "Find file", and "Clone or download". The file list shows the following files and their commit history:

File	Commit History	Time
bin	first commit	17 hours ago
project	first commit	17 hours ago
src	first commit	17 hours ago
.gitignore	first commit	17 hours ago
README.md	Initial commit	18 hours ago

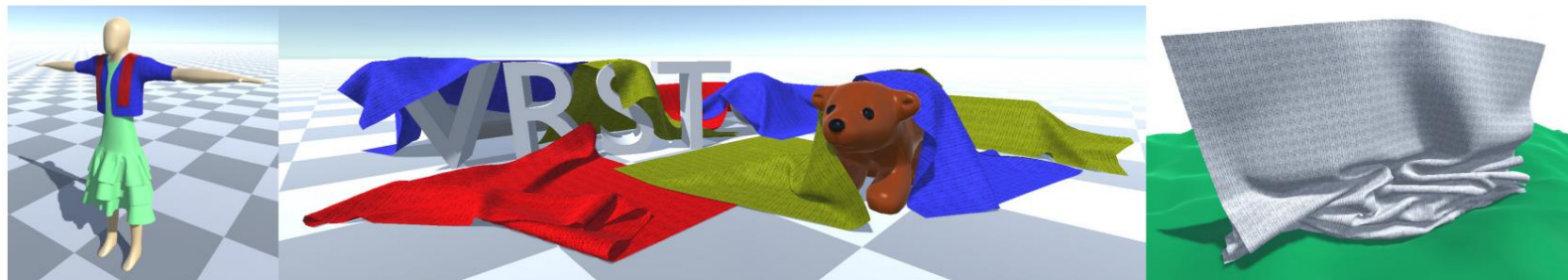
Below the file list, there is a section for the README.md file, which contains the title `air_mesh_2d` and the description "An implementation of 'Air Meshes for Robust Collision Handling', SIGGRAPH (2015)".

Air Meshed for Robust Collision Handling

- 解説の流れ
 - 動画
 - 基本概念
 - 実装方法
 - 応用例
 - 長所、短所

Air Meshed for Robust Collision Handling

- 応用例
 - GPU ray-traced collision detection for cloth simulation [FVB15]



Air Meshed for Robust Collision Handling

- 解説の流れ
 - 動画
 - 基本概念
 - 実装方法
 - 応用例
 - 長所、短所

Air Meshed for Robust Collision Handling

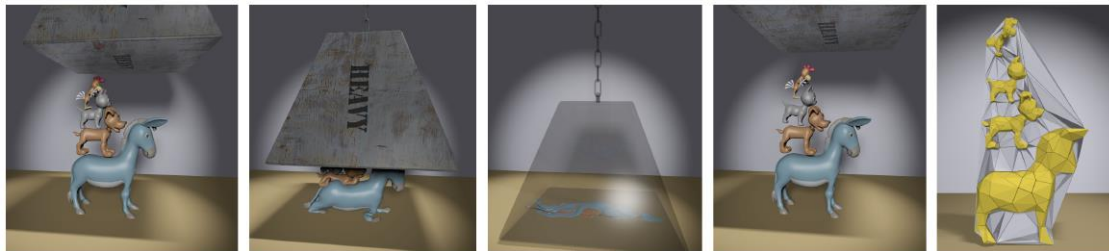
- 長所

- 複数レイヤーが重なり合うクロスシミュレーションなどで重なりを考慮して堅牢に衝突応答が可能
- 完全に潰れるケース(体積0)も扱える

- 短所

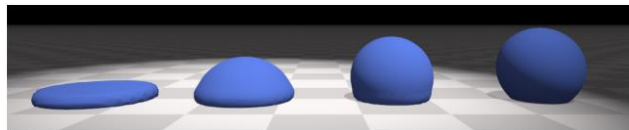
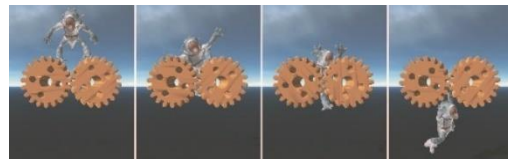
- Air Mesh 対象領域が広い場合は、計算コストが大きくなりすぎるのでうまく絞り込む必要がある

[MNTM15]



内容

- Position Based Dynamics 入門
- 論文紹介
 - Strain Based Dynamics
 - Air Meshes for Robust Collision Handling
 - XPBD Position-Based Simulation of Compliant Constrained Dynamics
- 考察、質疑応答



XPBD Position-Based Simulation of Compliant Constrained Dynamics

- 解説の流れ
 - 動画
 - 基本概念
 - 実装
 - 応用例
 - 長所、短所

XPBD Position-Based Simulation of Compliant Constrained Dynamics

- 解説の流れ
 - 動画
 - 基本概念
 - 実装
 - 応用例
 - 長所、短所

XPBD Position-Based Simulation of Compliant Constrained Dynamics

- 解説の流れ
 - 動画
 - 基本概念
 - 実装
 - 応用例
 - 長所、短所

XPBD Position-Based Simulation of Compliant Constrained Dynamics

- XPBD
 - e**X**tended + **P**osition **B**ased **D**ynamics
 - PBD の課題を解決
 - イテレーション回数、タイムステップによって剛性値 (stiffness) が変化してしまっていた課題を解決
 - 弾性エネルギーに基づいており力の推定が可能
 - 現実のコンプライアンス値がそのまま使える
 - local 解法
 - Projective Dynamics 等の global 解法の事前計算不要

ヤング率の逆数

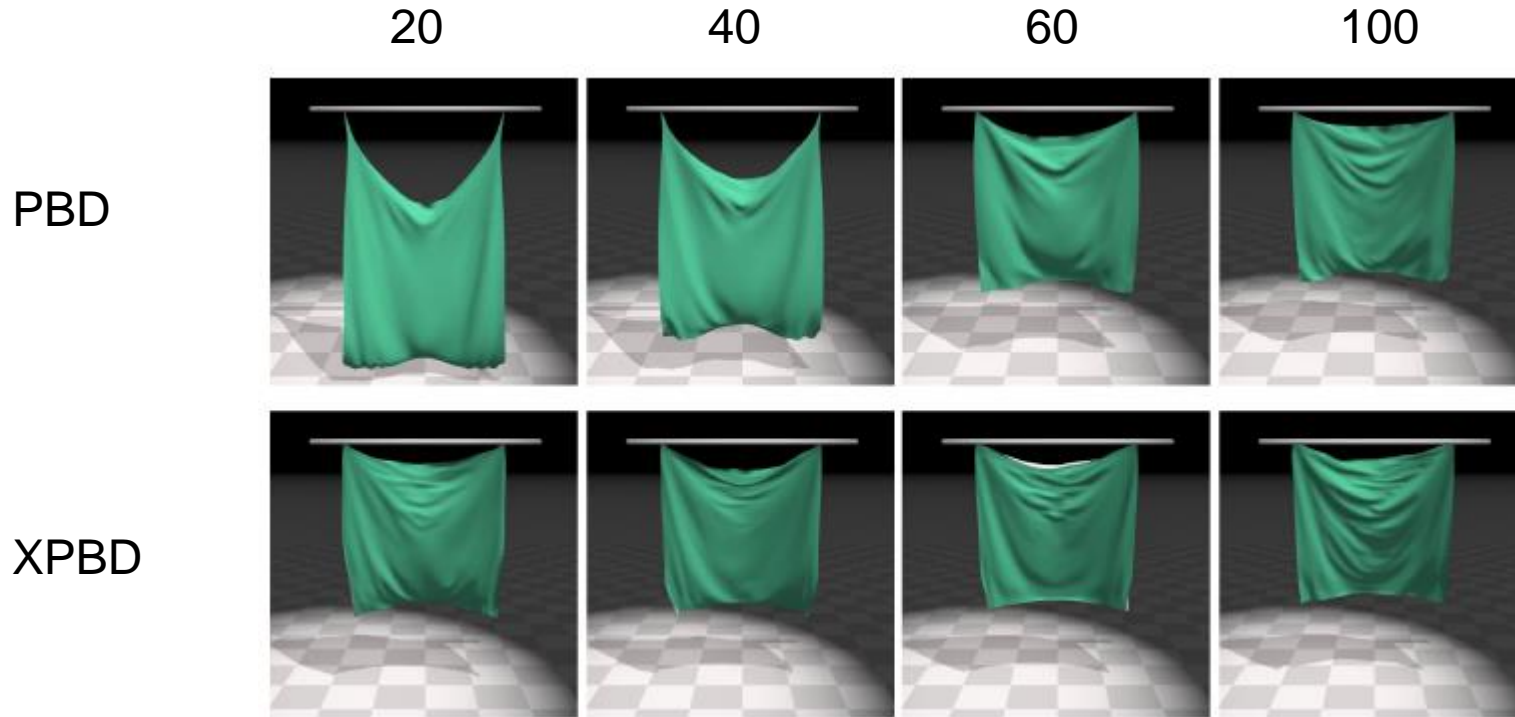
XPBD Position-Based Simulation of Compliant Constrained Dynamics

- コンプライアンス値

- <http://blog.mmacklin.com/>

MATERIAL	STIFFNESS (N/M ²)	COMPLIANCE (M ² /N)
Concrete	25.0 x 10 ⁹	0.04 x 10 ⁻⁹
Wood	6.0 x 10 ⁹	0.16 x 10 ⁻⁹
Leather	1.0 x 10 ⁸	1.0 x 10 ⁻⁸
Tendon	5.0 x 10 ⁷	0.2 x 10 ⁻⁷
Rubber	1.0 x 10 ⁶	1.0 x 10 ⁻⁶
Muscle	5.0 x 10 ³	0.2 x 10 ⁻³
Fat	1.0 x 10 ³	1.0 x 10 ⁻³

XPBD Position-Based Simulation of Compliant Constrained Dynamics



[MMN16]

XPBD Position-Based Simulation of Compliant Constrained Dynamics

1. 前ステップの位置 \mathbf{x}_n から予測位置 $\mathbf{x}_{\text{predict}}$ を求める
2. 修正位置 \mathbf{x}_0 を $\mathbf{x}_{\text{predict}}$ で初期化
3. ラグランジュ乗数 λ_0 を 0 で初期化する
4. for(i<イテレーション回数)
 - 4.1 $\Delta\lambda$ を求める(式18)
 - 4.2 $\Delta\mathbf{x}$ を求める(式17)
 - 4.3 $\lambda_{i+1} = \lambda_i + \Delta\lambda$ で更新する
 - 4.4 $\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta\mathbf{x}$ で更新する
4. 位置 $\mathbf{x}_{n+1} = \mathbf{x}_i$ が確定する
5. 速度 $\mathbf{v}_{n+1} = 1/\Delta t(\mathbf{x}_{n+1} - \mathbf{x}_n)$ で間接的に求まる

$\Delta\mathbf{x}$ が十分に小さくなくても
イテレーションを打ち切る

$$\Delta\lambda_j = \frac{-C_j(\mathbf{x}_i) - \tilde{\alpha}_j\lambda_{ij}}{\nabla C_j \mathbf{M}^{-1} \nabla C_j^T + \tilde{\alpha}_j}. \quad (18)$$

$$\Delta\mathbf{x} = \mathbf{M}^{-1} \nabla C(\mathbf{x}_i)^T \Delta\lambda. \quad (17)$$

XPBD Position-Based Simulation of Compliant Constrained Dynamics

1. 前ステップの位置 x_n から予測位置 x^* を求める
2. 修正位置 x_0 を x^* で初期化
3. 乗数 λ_0 を 0 で初期化する
4. for(i<イテレーション回数)
 - 4.1 $\Delta\lambda$ を求める(式18)
 - 4.2 Δx を求める(式17)
 - 4.3 $\lambda_{i+1} = \lambda_i + \Delta\lambda$ で更新する
 - 4.4 $x_{i+1} = x_i + \Delta x$ で更新する
4. 位置 $x_{n+1} = x_i$ が確定する
5. 速度 $v_{n+1} = 1/\Delta t(x_{n+1} - x_n)$ で間接的に求まる

Δx が十分に小さくなくても
イテレーションを打ち切る

$$\Delta\lambda_j = \frac{-C_j(x_i) - \tilde{\alpha}_j \lambda_{ij}}{\nabla C_j M^{-1} \nabla C_j^T + \tilde{\alpha}_j}. \quad (18)$$

$$\Delta x = M^{-1} \nabla C(x_i)^T \Delta\lambda. \quad (17)$$

XPBD Position-Based Simulation of Compliant Constrained Dynamics

- PBD の “scaling factor” s を XPBD のラグランジュ乗数 λ と捉えると α (コンプライアンス値) が 0 の場合同じ値に！

PBD

$$s_j = \frac{-C_j(\mathbf{x}_i)}{\nabla C_j \mathbf{M}^{-1} \nabla C_j^T}$$

XPBD

$$\Delta \lambda_j = \frac{-C_j(\mathbf{x}_i) - \tilde{\alpha}_j \lambda_{ij}}{\nabla C_j \mathbf{M}^{-1} \nabla C_j^T + \tilde{\alpha}_j}$$

[MMN16]

XPBD Position-Based Simulation of Compliant Constrained Dynamics

- 解説の流れ
 - 動画
 - 基本概念
 - 実装
 - 応用例
 - 長所、短所

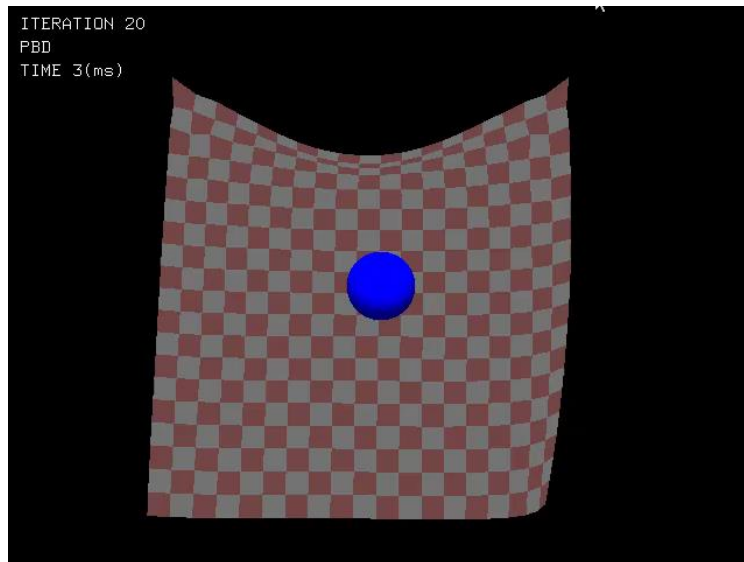
XPBD Position-Based Simulation of Compliant Constrained Dynamics

- 実装

- C++, OpenGL, glut, glm で実装(win, osx)

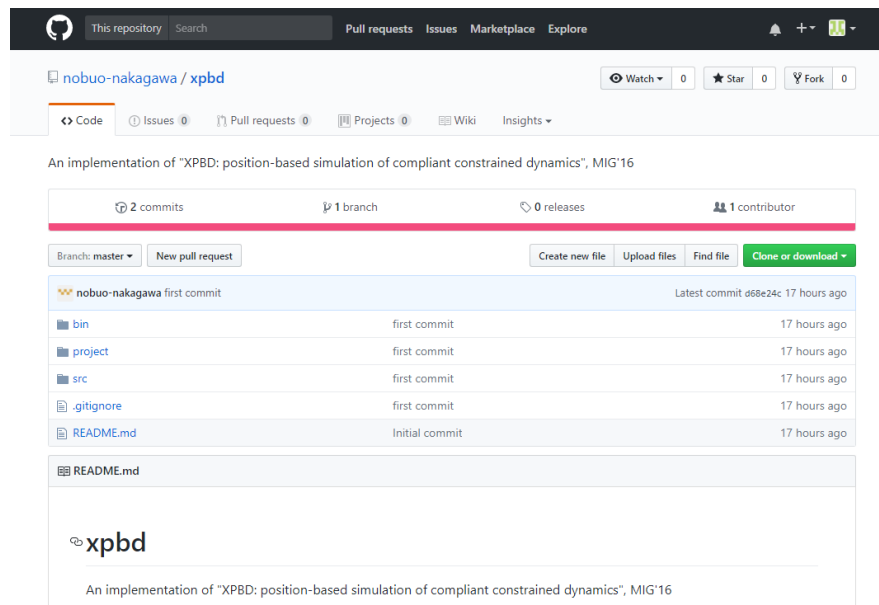
- デモ

- Concrete
 - Wood
 - Leather
 - Tendon
 - Rubber
 - Musscle
 - Fat



XPBD Position-Based Simulation of Compliant Constrained Dynamics

- Pull Request 大歓迎です
 - <https://github.com/nobuo-nakagawa/xpbd>



XPBD Position-Based Simulation of Compliant Constrained Dynamics

- 解説の流れ
 - 動画
 - 基本概念
 - 実装方法
 - 応用例
 - 長所、短所

XPBD Position-Based Simulation of Compliant Constrained Dynamics

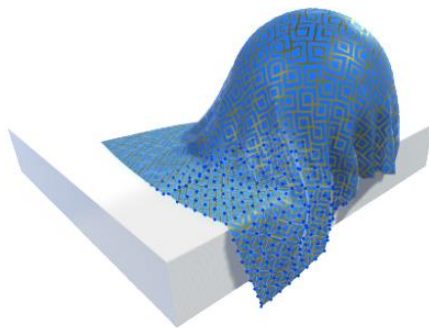
- 応用例
 - Obi Unified particle physics for Unity

Obi

Unified particle physics for Unity

Obi is the first CPU-based realtime unified physics framework:

- Unified framework for character and interactive cloth, fluids, ropes...
- Advanced editor tools
- High performance multithreaded solver
- Two-way interaction with rigidbodies, supports all collider types
- From low budget, simple effects to extremely complex behavior



And last, we have updated our core constraint solver to **XPBD**. XPBD is a recent technique that improves upon position-based dynamics, decoupling constraint stiffness from iteration count and time-step size. It also reduces artificial damping. This means **more accurate simulation, and easier to tune constraint parameters**. In regular PBD, you set a material stiffness value but the actual result will vary wildly depending on the amount of solver iterations and your physics timestep. This makes tuning material parameters a nightmare when deploying on multiple platforms with different performance budgets. XPBD is guaranteed to converge to the same result **given the same stiffness parameter and enough iterations**, which makes your life so much easier.

As an added bonus, XPBD also provides accurate force estimates for constraints, which come in handy for things like tearable cloth. Take a look at this video by the developers of the technique for a more in-depth comparison of PBD, XPBD and a reference Newton solver:

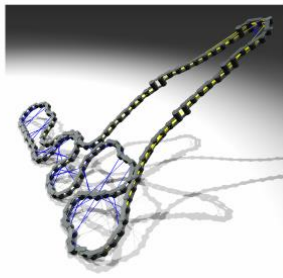
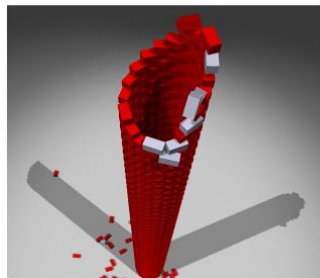
Blog に XPBD の記載がある。引用元

<http://blog.virtualmethodstudio.com/2017/01/obi-cloth-3-0-for-unity-whats-new/>

XPBD Position-Based Simulation of Compliant Constrained Dynamics

- 応用例

- Long range constraints for rigid body simulations
 - 筆者らの最新の論文
- 今後 PBD から XPBD への乗り換えが進む(?)



[MNMS17]

XPBD Position-Based Simulation of Compliant Constrained Dynamics

- 解説の流れ
 - 動画
 - 基本概念
 - 実装方法
 - 応用例
 - 長所、短所

XPBD Position-Based Simulation of Compliant Constrained Dynamics

- 長所

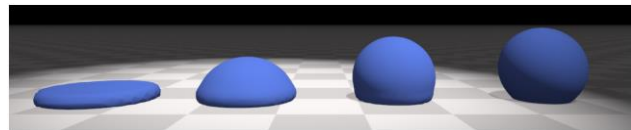
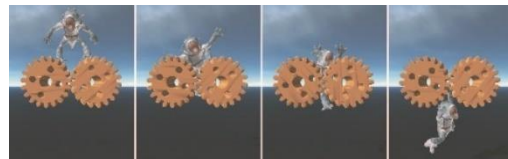
- イテレーション回数、タイムステップ依存なし
- PBD に比べて計算コストも同等
- 現実のコンプライアンス値を扱える

- 短所

- 収束が PBD に比べて早まらないことがある
- 近似手法なので Projective Dynamics[SSTLM14] 等の手法ほど正確ではない(ほぼ問題にはならない)

內容

- Position Based Dynamics 入門
- 論文紹介
 - Strain Based Dynamics
 - Air Meshes for Robust Collision Handling
 - XPBD Position-Based Simulation of Compliant Constrained Dynamics
- 考察、質疑応答



内容

- Position Based Dynamics

- より正確により堅牢に

- 現実のコンプライアンス値が使えるように

- XPBD への移行が進むか(?)

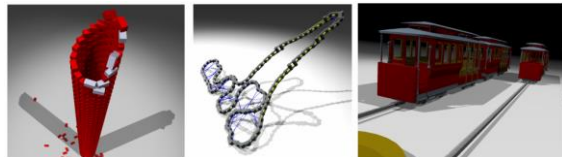
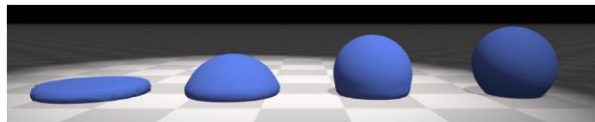
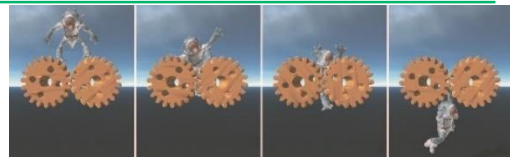
- 手法の改善が進んでいる

- Long Range Constraints for Rigid Body Simulations

- Adaptive mesh(未解決) [JMM17]

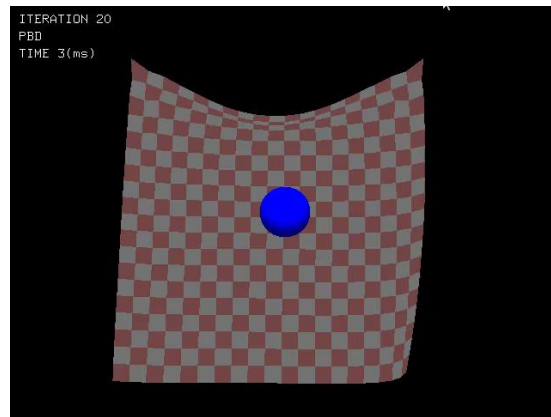
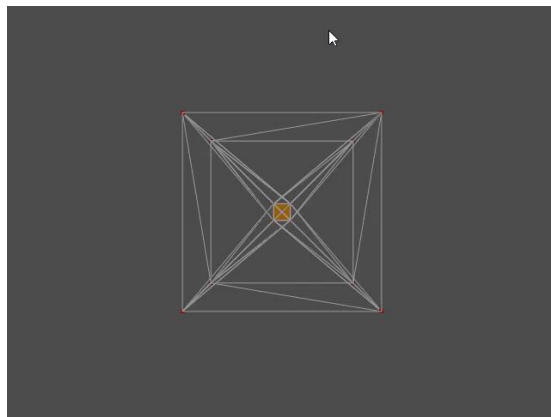
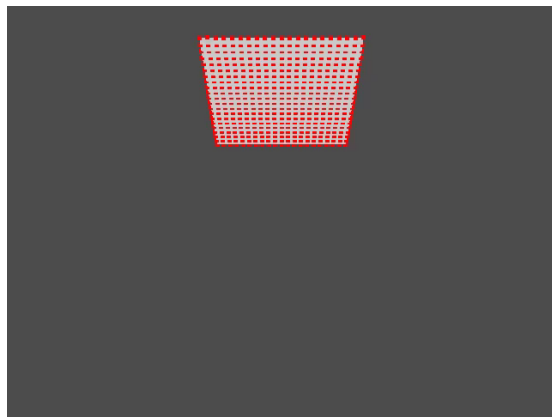
- クロス、チェーン、弾性体…

- 今後も期待できる手法!



ご清聴ありがとうございました

- 何か質問はございますか
 - 中川展男
 - nobuo_nakagawa@outlook.com



参考文献(1/3)

- [MNTM14] Matthias Muller, Nuttapong Chentanez, Tae-Yong Kim, Miles Macklin: Strain based dynamics. SCA '14 Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation Pages 149-157
- [MNTM15] Matthias Muller, Nuttapong Chentanez, Tae-Yong Kim, Miles Macklin: Air meshes for robust collision handling. ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2015, Volume 34 Issue 4, August 2015
- [MMN16] Miles Macklin, Matthias Muller, Nuttapong Chentanez: XPBD: position-based simulation of compliant constrained dynamics. MIG '16 Proceedings of the 9th International Conference on Motion in Games, Pages 49-54
- [MNMS17] Matthias Muller, Nuttapong Chentanez, Miles Macklin, Stefan Jeschke: Long range constraints for rigid body simulations. SCA '17 Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation, Article No. 14
- [NMM16] Nuttapong Chentanez, Matthias Muller, Miles Macklin: Real-time simulation of large elastoplastic deformation with shape matching. SCA '16 Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Pages 159-167
- [Jak01] Thomas Jakobsen: Advanced character physics. In Proceedings, Game Developer's Conference San Jose, March 20-24, 2001.

参考文献(2/3)

- [MHHR07] Matthias Muller, Bruno Heidelberger, Marcus Hennix, John Ratcliff: Position based dynamics. Journal of Visual Communication and Image Representation. Volume 18 Issue 2, April, 2007, Pages 109-118
- [MMNT14] Miles Macklin, Matthias Muller, Nuttapong Chentanez, Tae-Yong Kim: Unified particle physics for real-time applications. ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2014, Volume 33 Issue 4, July 2014, Article No. 153
- [Muhammad 16] Muhammad Mobeen Movania: Simulating Soft Bodies Using Strain Based Dynamics. Game Engine Gems, Volume 3, ISBN-13: 978-1-4987-5565-8, A K Peters / CRC Press c 2016. p.159-p.181
- [FVB15] Francois Lehericey, Valerie Gouranton, Bruno Arnaldi: GPU ray-traced collision detection for cloth simulation. VRST '15 Proceedings of the 21st ACM Symposium on Virtual Reality Software and Technology. Pages 47-50
- [JMM17] Jan Bender, Matthias Muller, Miles Macklin: A Survey on Position Based Dynamics, 2017. Eurographics 2017. In Tutorial Proceedings of Eurographics, 2017. Eurographics Association
- [Hang 15] Hang Si: TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator. ACM Transactions on Mathematical Software (TOMS), Volume 41 Issue 2, January 2015, Article No. 11

参考文献(3/3)

- [Paul 89] Paul Bourke: Triangulate Efficient Triangulation Algorithm Suitable for Terrain Modelling or An Algorithm for Interpolating Irregularly-Spaced Data with Applications in Terrain Modelling. Pan Pacific Computer Conference, Beijing, China. January 1989.
- [SSTLM14] Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, Mark Pauly: Projective dynamics: fusing constraint projections for fast simulation. ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2014, Volume 33 Issue 4, July 2014, Article No. 154.