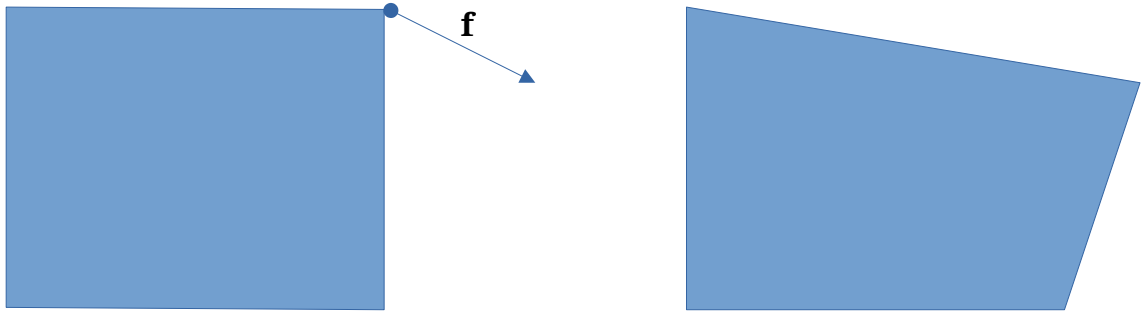
Human Riyahi

Flexible and deformable Objects

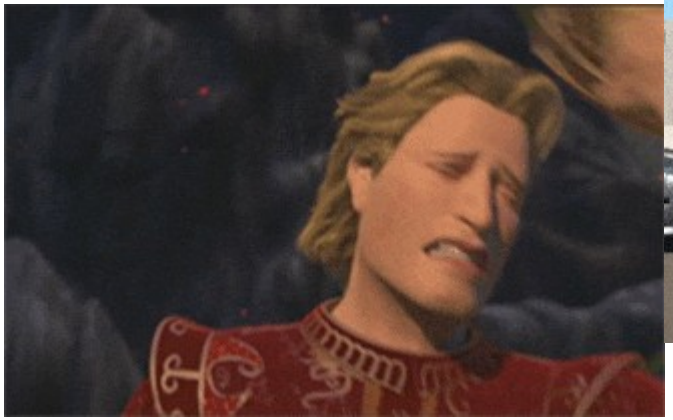
Unterschied zu Rigid Bodies

- Abstände zwischen Punkten können sich ändern

- Für jeden Punkt 2-3 Freiheitsgrade



Unzählige Anwendungen



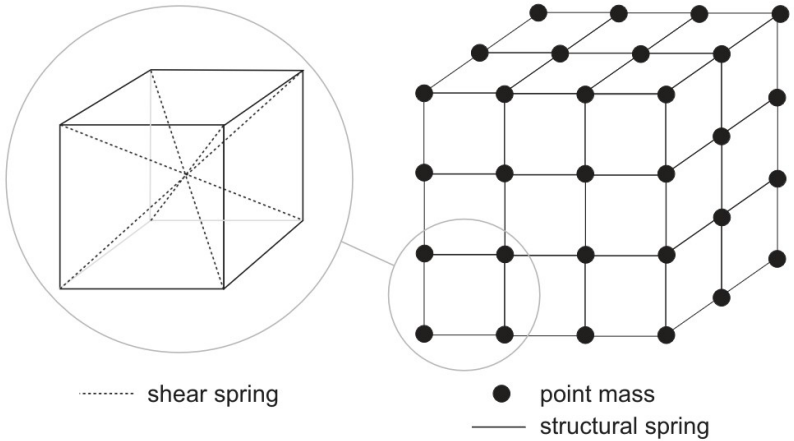
Gliederung

- Partikelbasierte Methoden
  - Mass Spring System
  - PBD/XPBD
- Darunter auch
  - Formerhaltung
  - Kollisionen in Soft Bodies
- Demos

Mass Spring System

Das Modell - Mass Spring System

- diskretes Modell
- Massenpunkte verbunden mit massenlosen Federn
- Massenpunkte frei beweglich

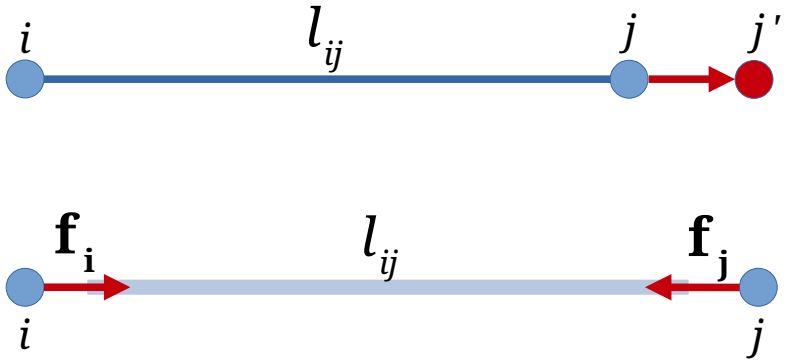


Die Springs - Mass Spring System

- Springs haben Ruhelänge l_{ij}
- Verbundene Punkte streben sie an:
  - Mit Hookesches Gesetz
    - Steifigkeit k_s
    - Vektor i – j x_{ij}

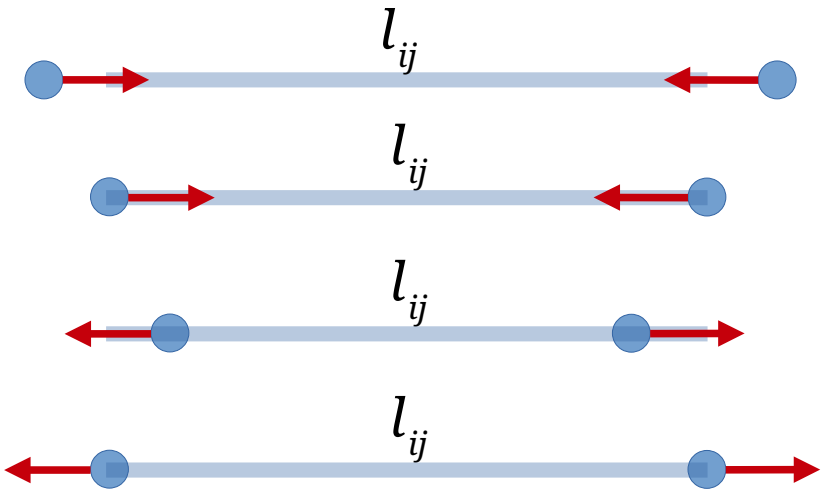
$$\mathbf{f_i} = k_s(|x_{ij}| - l_{ij})\frac{x_{ij}}{|x_{ij}|}$$

Die Springs - Mass Spring System



$$\mathbf{f}_i = k_s(|x_{ij}| - l_{ij})\frac{x_{ij}}{|x_{ij}|}$$

Die Springs schwingen für immer...



1) Maximale **elastische** Energie
2) Maximale **kinetische** Energie
3) Maximale **elastische** Energie
4) Maximale **kinetische** Energie

Springs abdämpfen

- „Damper“ : Viskoelastische Springs
  - Viskuöse Kraft :
    - Dämpfung k_d
    - Geschwindigkeiten v_i, v_j

$$\mathbf{f_i} = k_d(\mathbf{v}_j - \mathbf{v}_i)$$

Springs abdämpfen

- „Damper“ : Viskoelastische Springs
  - Viskuöse Kraft :
    - Dämpfung k_d
    - Geschwindigkeiten v_i, v_j
- Dämpft leider auch Rotationen

$$\mathbf{f_i} = k_d(v_j - v_i)$$

Springs abdämpfen

- „Damper“ : Viskoelastische Springs
  - Viskuöse Kraft :
    - Dämpfung k_d
    - Geschwindigkeiten v_i, v_j
- Nur Kräfte entlang der Feder

$$\mathbf{f_i} = k_d \left(\frac{\mathbf{v}_{ij}^T \mathbf{x}_{ij}}{\mathbf{x}_{ij}^T \mathbf{x}_{ij}} \right) \mathbf{x}_{ij}$$

$$v_{ij} = v_j - v_i$$

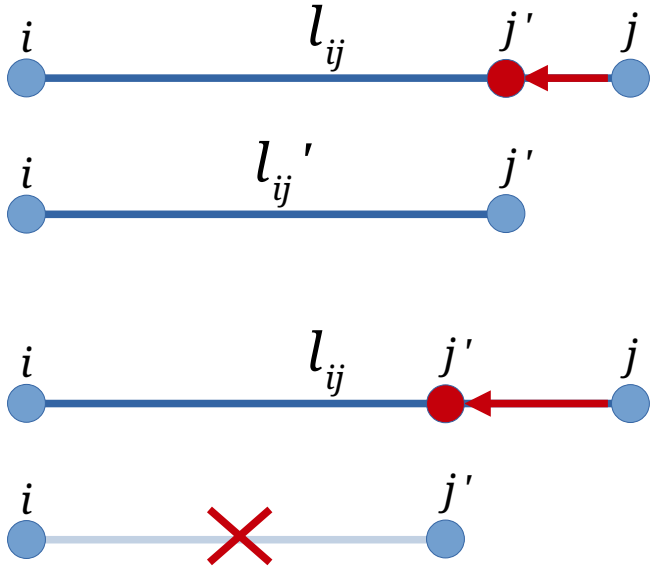
Springs deformieren oder zerstören

- Inelastische Verformung modellieren
- Mit Kraftgrenzen umsetzen
  - Plastische Verformung

$$|\Delta l| > \Delta l_{max} \rightarrow l_{ij} := |x_{ij}|$$

  - Bruch

$$|\Delta l| > \Delta l_B \rightarrow \textbf{lösen}$$



Simulation – Mass Spring System

- Beschleunigkeiten berechnen
  - Newtons zweites Gesetz: $\mathbf{f} = ma$
  - Mit Diagonalmatrix **M**: $\mathbf{f}(x, v) = \mathrm{M}a$
- ODE wird auch numerisch gelöst

Simulation – Mass Spring System

- Integrationsverfahren
  - Explicit / Forward Euler → ist **instabil**

$$v(t+\Delta t)=v(t)+\Delta t*a(t)$$
$$x(t+\Delta t)=x(t)+\Delta t*v(t)$$

Simulation – Mass Spring System

- Integrationsverfahren
  - Semiimplicit / F.B. / Sympletic Euler → bedingt stabil: $\Delta t < 2 * \sqrt{\frac{m}{k}}$

$$v(t+\Delta t) = v(t) + \Delta t * a(t)$$

$$x(t+\Delta t) = x(t) + \Delta t * v(t+\Delta t)$$

Simulation – Mass Spring System

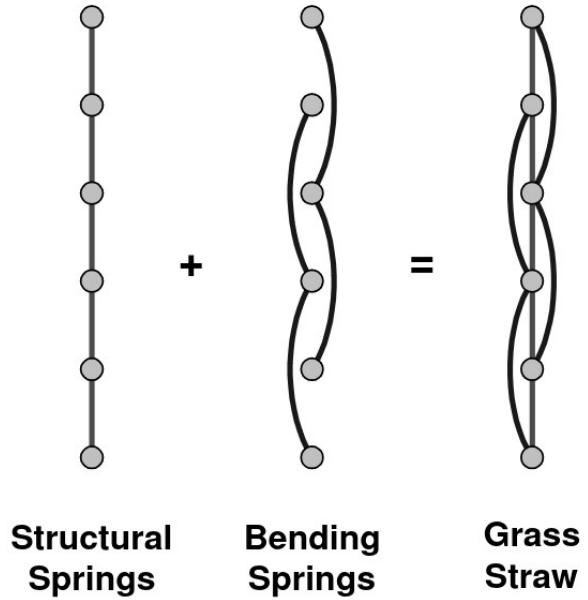
- Integrationsverfahren
  - Implicit / Backward Euler → bedingungslos stabil, verliert aber Energie

$$v(t+\Delta t)=v(t)+\Delta t * a(t+\Delta t)$$

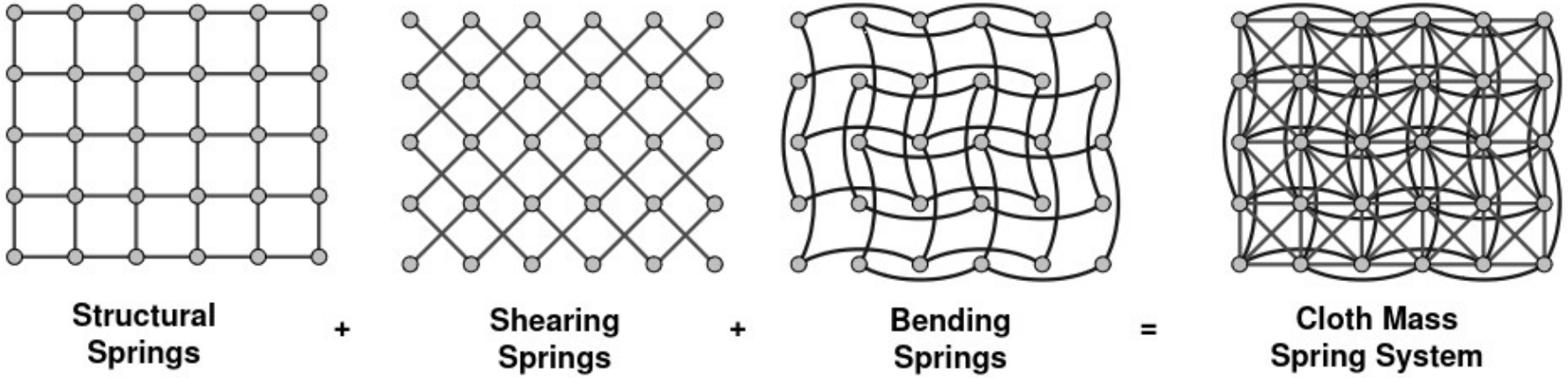
$$x(t+\Delta t)=x(t)+\Delta t * v(t+\Delta t)$$

Mass Spring System in 1D

- Für
  - Seile
  - Gras
  - Haare

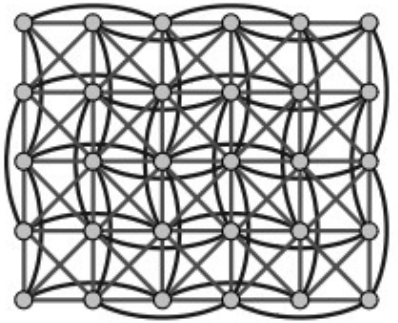


Mass Spring System in 2D – Cloths / Surface Mesh



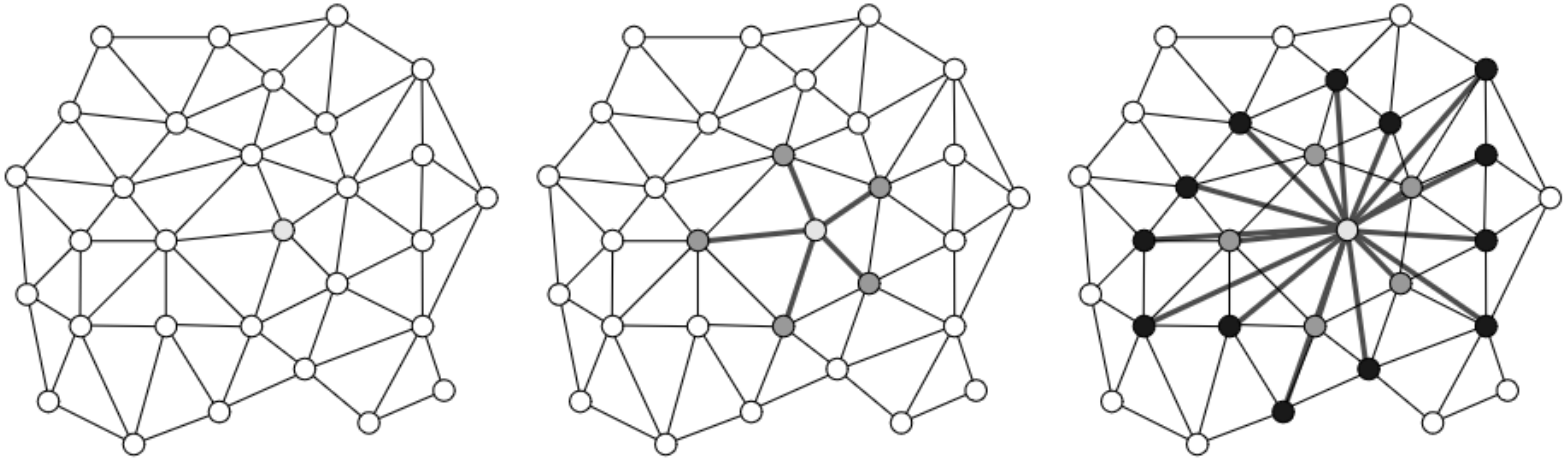
Mass Spring System in 2D - Cloths

- Modell passt zu anistropen Cloths 👍
  - Steifigkeitswerte anpassen für verschiedene Arten von Cloth (Baumwolle, Polyester, Seide, ...)
- Modell nur für rechteckige Cloths geeignet 👎



Cloth Mass Spring System

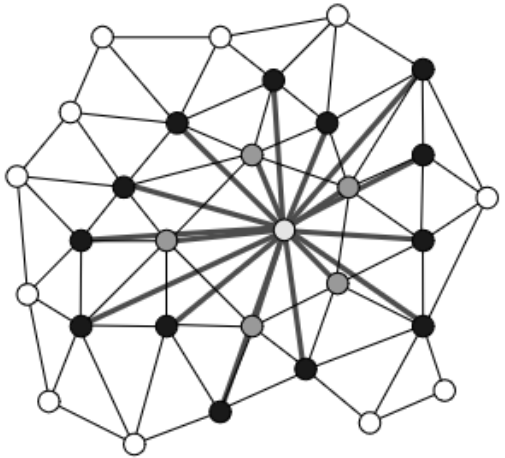
Mass Spring System in 2D - Generalisiert



(a) 0 neighborhood (b) 1 neighborhood (c) 2 neighborhood

Mass Spring System in 2D - Generalisiert

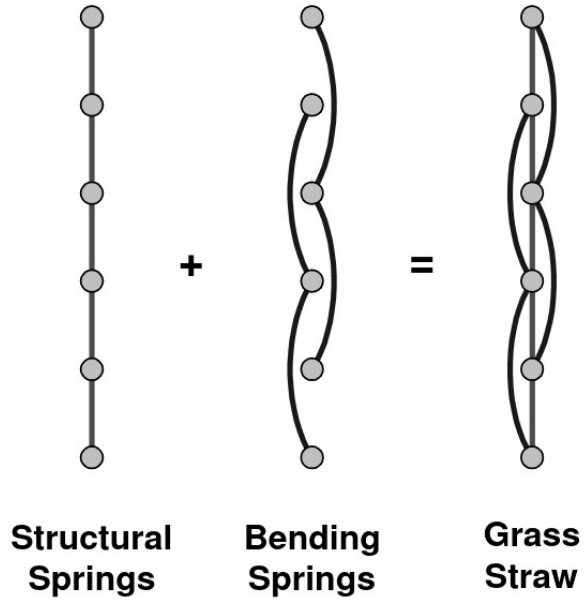
- Generalisiert Modell von eben
- Nachbarschaftsgröße größer → Cloth rigider/starrer
  - Wir müssen die Federkonstanten nicht erhöhen 👍



(c) 2 neighborhood

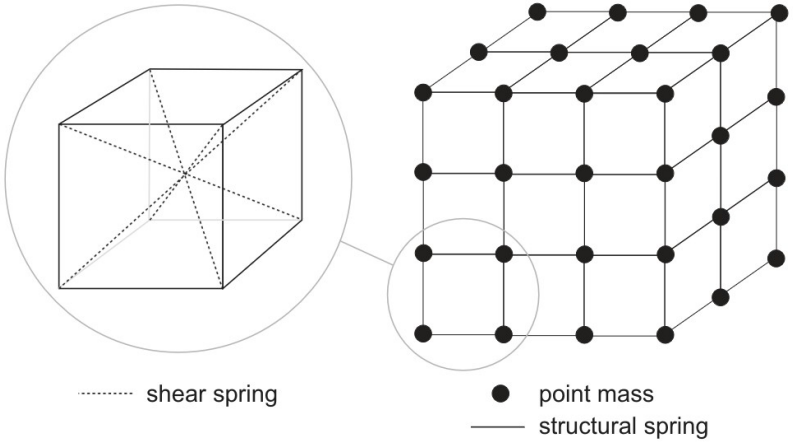
Mass Spring System in 1D – II

- (**X>2**)-Neighborhood auch auf 1D-Objekte übertragbar



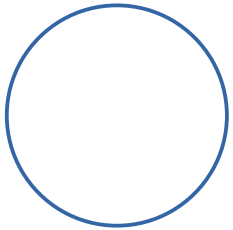
Mass Spring System in 3D

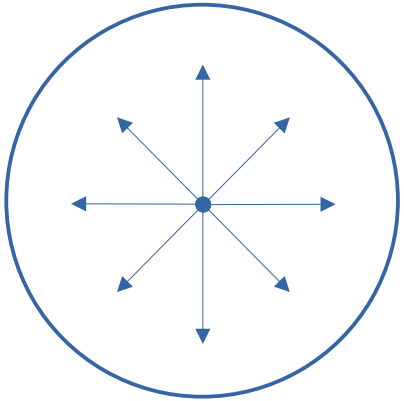
- Gitter aus Würfeln
  - Körper vorher **voxelisieren**
  - Surface Mesh mit Voxel Mesh koppeln
- Hat üblichen Springs sowie **Spatial diagonals** um Volumen zu erhalter



Andere Methode zur Volumenerhaltung...

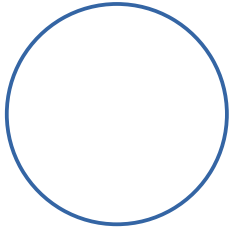
- Das Pressure Model
  - Idee: „Windkraft“ kommt von „innen“

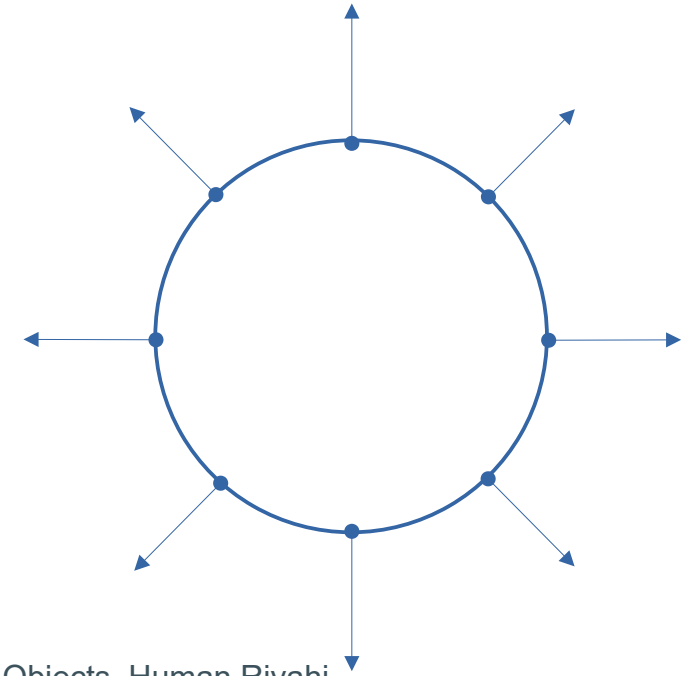




Das Pressure Model

- Kräfte Richtung Normale





Das Pressure Model

- Algorithmus bestimmt wie stark diese Kräfte sind
- Arbeitet mit
  - **Allgemeine Gasgleichung**
  - Volumenapproximation mit Bounding Boxes

- **Allg. Gasgl.** $PV = nRT$
- Druck: $P = V^{-1} \times nRT$
- Druckvektor: $\vec{P} = P \times \hat{n}$
- Druckkraft: $\vec{F}_p = \vec{P} \times A$

V: Volumen
n: Stoffmenge:
R: Gaskonstante

T: Temperatur
n_hat: Normale
A: Fläche

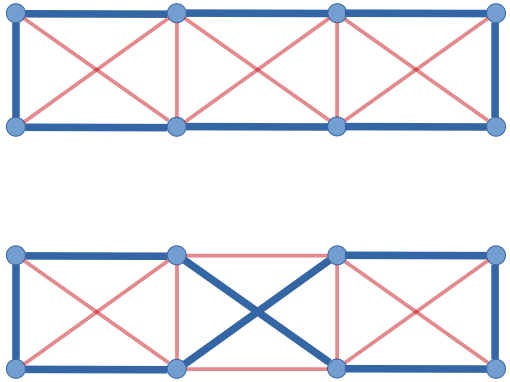
Das Pressure Model

1. Volumen des Körpers berechnen → V
2. Für alle Faces
    1. Flächeninhalt berechnen → A
    2. Druck berechnen → $P = V^{-1} \times nRT$
    3. Druckkraft berechnen → $\vec{P} = P \times \hat{n}$ → $\vec{F}_p = \vec{P} \times A$
    4. Für alle Partikel vom Face
        1. Druckkraft akkumulieren

Formerhaltung

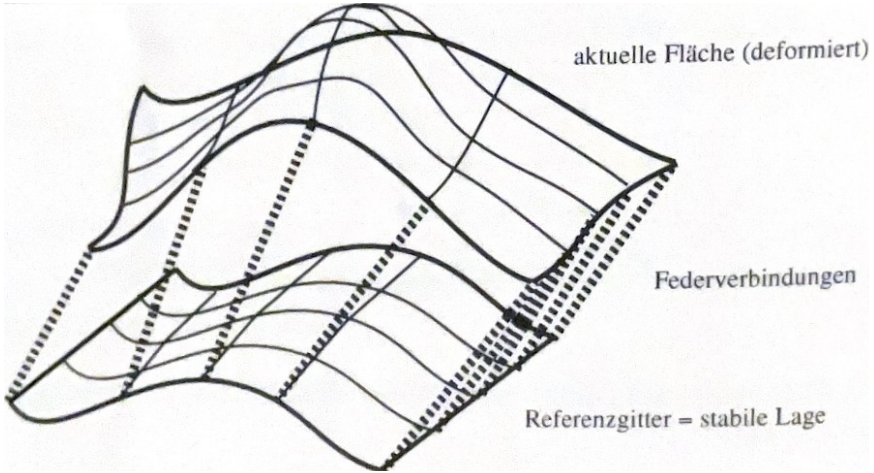
Formerhaltung momentan nicht garantiert

- Haben nur Abstandserhaltung zwischen Punkten
- Volumenerhaltung nicht immer geeignet
- → Falsche Formen in denen die Abstandsdifferenzen lokal minimiert sind



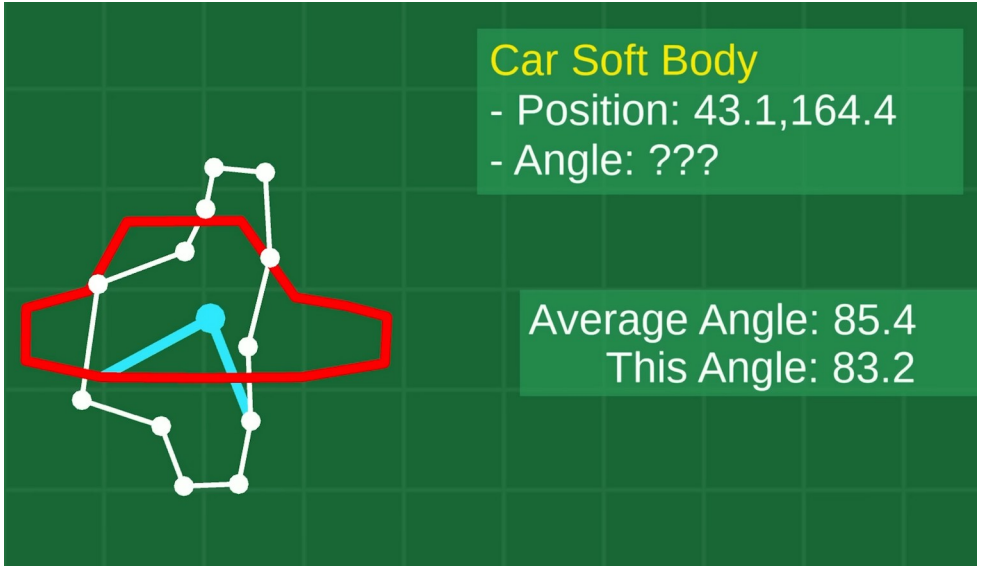
Unsichtbare Referenzgitter

- Punkte sind über Springs oder Distance Constraints mit Referenzgitter verbunden
- Auf **Mass Spring** und **XPBD** anwendbar



Unsichtbare Referenzgitter

- Orientierung/Winkel des Referenzgitter anpassen
- Dafür aktueller Winkel des Objekts nötig



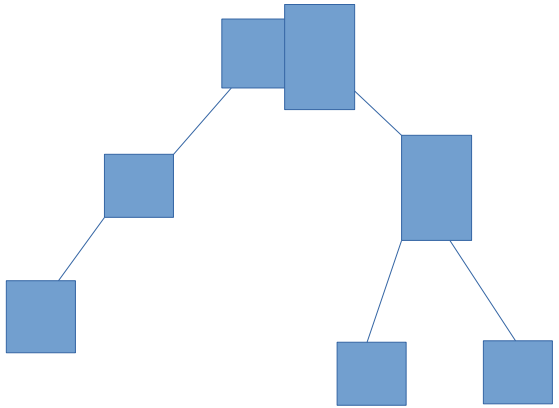
Kollisionen bei Soft Bodies

Kollisionen bei Soft Bodies sind teuer!

- Alle Kontaktpunkte müssen betrachtet werden
- Selbstkollision möglich
- Geometrie kann sich ändern
- Durchdringung möglich, da Stoff dünn → CCD nötig

Boundary Volume Hierarchies

- Broad Phase CD
- Geometrie hierarchisch organisieren
  - Einer Szene oder eines Objekts
- Spart Kollisionstests
  - log2(n) vs n^2



BVH bei Soft Bodies

- Geometrie kann sich stets ändern, Selbstkollision möglich
- Eigene BVH nötig
- Effiziente BVH Updates nötig
- BVH gegen sich selbst testen
  - Benachbarte Primitive ausschließen

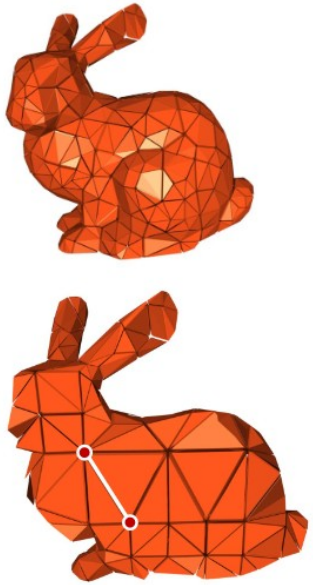
Position Based Dynamics

Position Based Dynamics

- Position-basiert anstatt Kraftbasiert
- Positionen korrigieren um **Constraints** C zu erfüllen
- Keine Stabilitätsprobleme bei steifen Systemen

PBD - Soft Bodies modellieren

- 3D - Mesh aus Tetraedern mit
  - 1 **Distant Constraint** pro Edge
  - 1 **Volume Constraint** pro Tetraeder

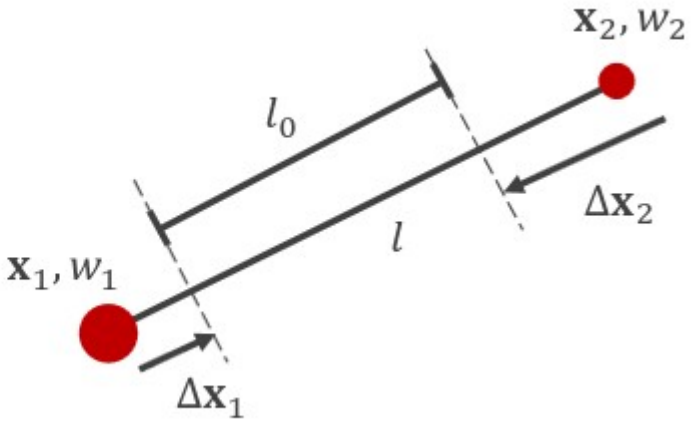


Distance Constraints

$$C = l - l_0$$

$$\nabla C_1 = \frac{\mathbf{x}_2 - \mathbf{x}_1}{|\mathbf{x}_2 - \mathbf{x}_1|}$$

$$\nabla C_2 = -\frac{\mathbf{x}_2 - \mathbf{x}_1}{|\mathbf{x}_2 - \mathbf{x}_1|}$$

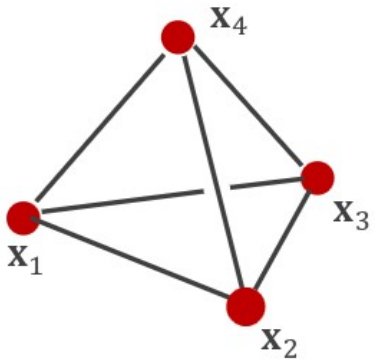


Volume Conservation Constraints

$$C = 6(V - V_{\text{rest}})$$

$$V = \frac{1}{6}\big((\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1)\big) \cdot (\mathbf{x}_4 - \mathbf{x}_1)$$

$$\nabla_1 C = (\mathbf{x}_4 - \mathbf{x}_2) \times (\mathbf{x}_3 - \mathbf{x}_2)$$
$$\nabla_2 C = (\mathbf{x}_3 - \mathbf{x}_1) \times (\mathbf{x}_4 - \mathbf{x}_1)$$
$$\nabla_3 C = (\mathbf{x}_4 - \mathbf{x}_1) \times (\mathbf{x}_2 - \mathbf{x}_1)$$
$$\nabla_4 C = (\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1)$$



PBD - Simulation

- Vorläufig Positionen berechnen
- **Positionen korrigieren anhand Constraints**
  - näherungsweise
- Geschwindigkeit updaten

while simulating
  for all particles i
    $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t\, \mathbf{g}$
    $\mathbf{p}_i \leftarrow \mathbf{x}_i$
    $\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta t\, \mathbf{v}_i$

  for all constraints C
    solve$(C, \Delta t)$

  for all particles i
    $\mathbf{v}_i \leftarrow (\mathbf{x}_i - \mathbf{p}_i)/\Delta t$

PBD - Simulation

- **Positionen korrigieren anhand Constraints**
  - Positionsupdate

solve($C, \Delta t$):

for all particles i of C
compute $\Delta \mathbf{x}_i$
$\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta \mathbf{x}_i$

PBD - Simulation

- **Positionen korrigieren anhand Constraints**
  - Mit N Iterationen

while simulating
  for all particles i
    $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t\, \mathbf{g}$
    $\mathbf{p}_i \leftarrow \mathbf{x}_i$
    $\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta t\, \mathbf{v}_i$
  for n iterations
    for all constraints C
      solve$(C, \Delta t)$
  for all particles i
    $\mathbf{v}_i \leftarrow (\mathbf{x}_i - \mathbf{p}_i)/\Delta t$

PBD - Simulation

- **Positionen korrigieren anhand Constraints**
  - Mit Substeps
  - Und nur einer Iteration
  - **Ist effektiver**

$\Delta t_s \leftarrow \Delta t / n$
while simulating
  for n substeps
    for all particles i
      $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t_s \mathbf{g}$
      $\mathbf{p}_i \leftarrow \mathbf{x}_i$
      $\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta t_s \mathbf{v}_i$
    for all constraints C
      solve(C, Δt_s)
    for all particles i
      $\mathbf{v}_i \leftarrow (\mathbf{x}_i - \mathbf{p}_i) / \Delta t_s$

PBD – Constraint lösen

- Gradienten vorberechnen $\nabla C1, \nabla C2, \ldots, \nabla Cm$
- Skalar λ vorberechnen

$$\lambda = \frac{-C}{w_1|\nabla C_1|^2 + w_2|\nabla C_2|^2 + \cdots + w_n|\nabla C_n|^2 + \frac{\alpha}{\Delta t^2}}$$

- Update für alle $\mathbf{x}i$ berechnen

$$\Delta \mathbf{x}_i = \lambda w_i \nabla C_i$$

PBD – Constraint lösen

- Skalar λ vorberechnen

$$\lambda = \frac{-C}{w_1|\nabla C_1|^2 + w_2|\nabla C_2|^2 + \cdots + w_n|\nabla C_n|^2 + \frac{\alpha}{\Delta t^2}}$$

→ **XPBD** (E**x**tended **PBD**) fügt Term hinzu

- Erlaubt Zeitschritt-Invarianz für Soft Constraints (mit Steifigkeit **k**)
- Compliance $\alpha = 1/k$ erlaubt unendliche Steifigkeit ($\alpha = 0$)

PBD – Constraint lösen – Beispiel Distance Constraint

$$C = l - l_0$$

$$\nabla C_1 = \frac{\mathbf{x}_2 - \mathbf{x}_1}{|\mathbf{x}_2 - \mathbf{x}_1|}$$

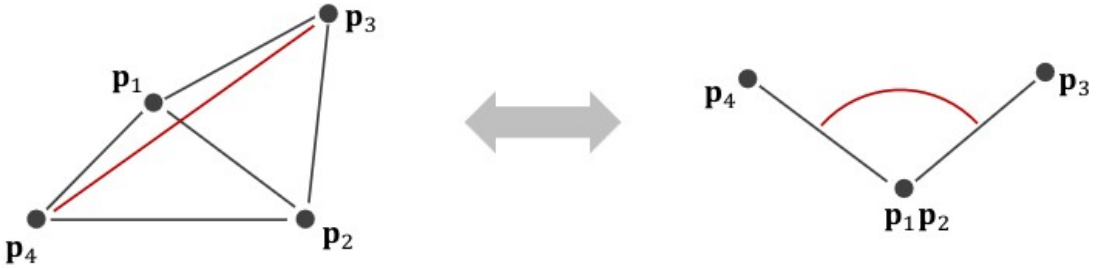
$$\nabla C_2 = -\frac{\mathbf{x}_2 - \mathbf{x}_1}{|\mathbf{x}_2 - \mathbf{x}_1|}$$

$$\Delta \mathbf{x}_1 = \frac{w_1}{w_1 + w_2}(l - l_0)\frac{\mathbf{x}_2 - \mathbf{x}_1}{|\mathbf{x}_2 - \mathbf{x}_1|}$$

$$\Delta \mathbf{x}_2 = -\frac{w_2}{w_1 + w_2}(l - l_0)\frac{\mathbf{x}_2 - \mathbf{x}_1}{|\mathbf{x}_2 - \mathbf{x}_1|}$$

Cloths simulieren

- Distance Constraints mit Compliance $\alpha \to 0$
- Bending Constraint
  - 2-Distance Constraint
  - Angle Constraint



Kollisionen mit PBD

- Collision Constraints
  - in jedem Physikstep neu generiert
    - Betrachte Strahl **x_i** → **p_i**
    - Durchdringt er einen Körper?
      - Eintrittspunkt **q_c** und Normale **n_c** berechnen
      - Füge Soft Constraint **C(p) = (p-q_c) * n_c** hinzu

Kollisionen mit PBD

- Collision Constraints
  - Ist der Strahl in einem Körper?
    - Nächsten Oberflächenpunkt **q_s** zu **p_i** finden
    - Füge Soft Constraint **C(p) = (p-q_s) * n_s** hinzu

Kollisionen mit PBD

- Methode nur für Kollision mit statischen Objekten korrekt
  - Fügen dem Kollisionspartner kein Impuls hinzu
- PBD-Paper schlägt weitere Collision Constraints vor, u.a. für Self Collision

Collision Response

- Geschwindigkeiten manipulieren
  - Abdämpfen
  - In Richtung der Eintritts-Normale reflektie

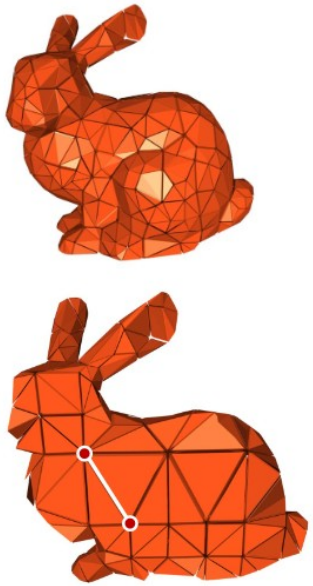
while simulating
  for all particles i
    $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t\, \mathbf{g}$
    $\mathbf{p}_i \leftarrow \mathbf{x}_i$
    $\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta t\, \mathbf{v}_i$

  for all constraints C
    solve$(C, \Delta t)$

  for all particles i
    $\mathbf{v}_i \leftarrow (\mathbf{x}_i - \mathbf{p}_i)/\Delta t$

PBD - Fazit

- Simples Verfahren 👍
- Bedingungslos **stabil**
- XPBD kann beliebig steife Objekte simulieren



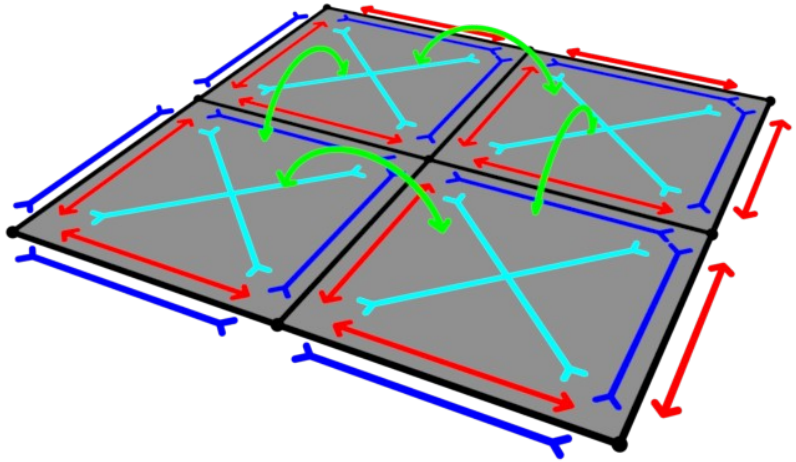
Ein paar Praxisbeispiele...

Blender – Open Source 3D-Modelling Software

- Hat Softbody Dynamics
  - Nutzt für Cloth **Mass Spring**
  - Unterstützt allerlei Kollisionen

Blender – Open Source 3D-Modelling Software

- Cloth hat 4 Spring Arten
  - Tension Springs
  - Compression Springs
  - Shear Springs
  - Angular Bending Springs
- Habe eine Demo mit einer Flagge erstellt

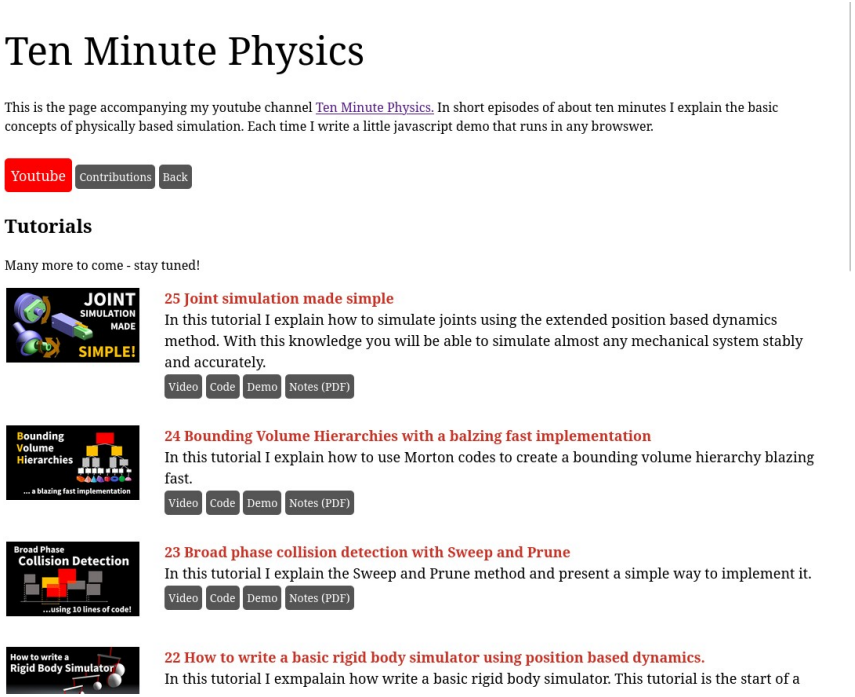


BeamNG – Autosimulator

- Hat Softbody Dynamics
  - Nutzt **Mass Spring**
    - **Nodes** und „**Beams**“
    - **2000hz** Physik-updates
  - Unterstützt
    - allerlei Kollisionen
    - Platizität und Zerstörung
  - Gute Dokuseite

10-Minute-Physics

- Von Matthias Müller, Co-Autor von PBD ***und*** XPBD Paper
- u.a. Demos und Erklärvideos für Soft Bodies mit **XPBD**



Ten Minute Physics

This is the page accompanying my youtube channel Ten Minute Physics. In short episodes of about ten minutes I explain the basic concepts of physically based simulation. Each time I write a little javascript demo that runs in any browswer.

Youtube | Contributions | Back

Tutorials

Many more to come - stay tuned!

25 Joint simulation made simple
In this tutorial I explain how to simulate joints using the extended position based dynamics method. With this knowledge you will be able to simulate almost any mechanical system stably and accurately.
Video | Code | Demo | Notes (PDF)

24 Bounding Volume Hierarchies with a balzing fast implementation
In this tutorial I explain how to use Morton codes to create a bounding volume hierarchy blazing fast.
Video | Code | Demo | Notes (PDF)

23 Broad phase collision detection with Sweep and Prune
In this tutorial I explain the Sweep and Prune method and present a simple way to implement it.
Video | Code | Demo | Notes (PDF)

22 How to write a basic rigid body simulator using position based dynamics.
In this tutorial I exmpalain how write a basic rigid body simulator. This tutorial is the start of a

Godot: Open Source Game Engine

- Hat Softbody Dynamics in 3D
  - **Jolt** ist **XPBD**-based
  - Keine Kollisionen zw. Softbodies…
  - Kein Shapematching...
- → habe dazu 2 „Spiele“ erstellt

Vor den Demos ein kurzes Fazit...

Kleines Fazit

- Verfahren behandelt, die für Echtzeitsimulation relevant sind
- Komplizierte, realistischere, Verfahren nicht behandelt
  - Finite Element Methode (FEM)
  - Energy Minimization
  - ...

Methode aus dem Pixar Buch

- Physikalisch korrekt die Verformung berechnen
  - Deformationsgradient **F**
- Daraus „Deformierungs“-**Energie** bestimmen
- Kräfte bestimmen, die **Energie** verringern

Danke für eure Aufmerksamkeit!

Quellen für die jeweiligen Abschnitte

Quellen - Mass Spring System

1. Physically Based Deformable Models in C.G., Nealen et al. 2006.
    1. Folien: Modell - Damper
    2. Folien: Beschleunigkeitsberechnung + Time Integration
    3. Explicit Euler ist instabil

2. Lebendige virtuelle Welten, F. Dai. 1996
    1. Kraftgrenzen für Springs

3. Meshless Deformations Based on Shape Matching, Müller et. All.
    1. Sympletic Euler ist bedingt stabil

4. Physics Based Animation, Erleben. 2005
    1. Folie MSS in 1D (I) – Folie MSS in 3D

5. Pressure Model of Soft Body Sim. Matyka et. al. 2003
    1. Pressure Model

Quellen - Formerhaltung

1. Jelly Car Youtube Video: https://youtu.be/3OmkehAJoyo?si=D-m6oiD-2ASxxLoC
    1. Formerhaltung nicht garantiert
    2. Orientierung der Referenzgitter

2. Lebendige virtuelle Welten, F. Dai. 1996
    1. Referenzgitter

Quellen - Kollisionen

1. Collision Detection for Deformable Objects, Teshner et al 2005

Quellen - PBD

1. Position Based Dynamics, Müller et. al. 2007
    1. Verständnis von PBD

2. XPBD: Position-Based Simulation of Compliant Constrained Dynamics, Macklin et. al. 2016
    1. Verständnis von XPBD

3. 10-Minute-Physics https://matthias-research.github.io/pages/tenMinutePhysics/index.html
    1. Für die Folien

Quellen - Demo

1. Blender Dokuseite: https://docs.blender.org/manual/en/4.0/physics/cloth/introduction.html
    1. Die 4 Springarten für Cloth
2. BeamNG Dokuseite: https://www.beamng.com/game/about/physics/
    1. Vorgezeigt
3. BeamNG.tech Technical Paper, Maul et. al.
    1. 2000 hz Physik Update Rate
4. 10-Minute-Physics: https://matthias-research.github.io/pages/tenMinutePhysics/index.html
    1. vorgezeigt
5. Godot Source Code: https://github.com/godotengine/godot/blob/master/thirdparty/jolt_physics/Jolt/Physics/SoftBody/SoftBodyMotionProperties.h
    1. Jolt nutzt XPBD

Quellen - Fazit

1. Dynamic Deformables…, Theodore Kim, David Eberle. 2022

    1. Das Pixar Buch aus der Folie nach dem Fazit