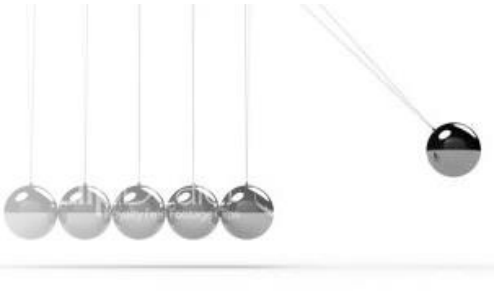


Advanced Game Physics

Stoffsimulation

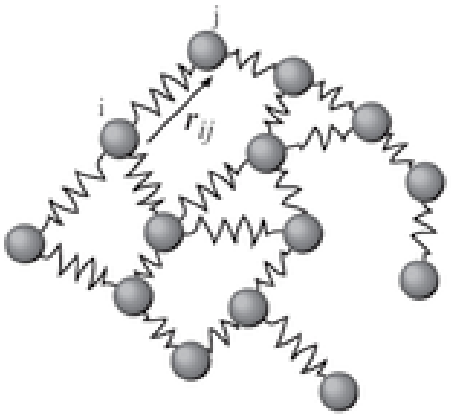
Prof-Dr. Günther Greiner,
Matteo Colaianni M.Sc., Benjamin Keinert M.Sc.
Darius Rückert B.Sc.

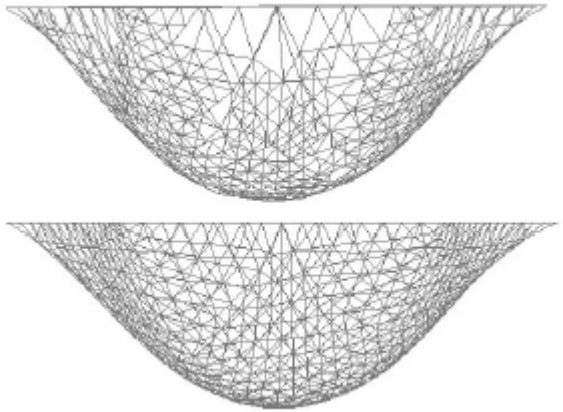
Lehrstuhl für Graphische Datenverarbeitung

Wintersemester 2015/16

Einführung

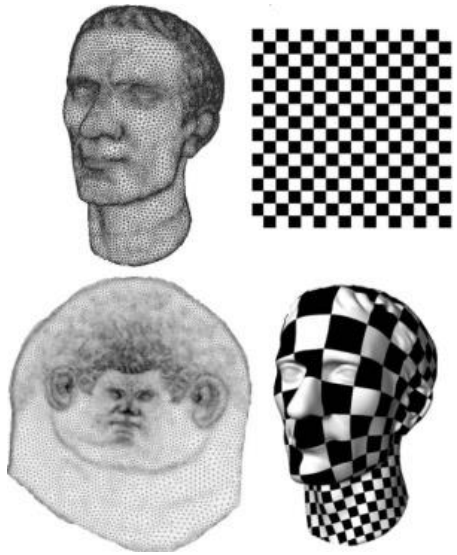
- Masse-Feder-Systeme
- Masse-Feder-Netzwerke
- Position-Based-Dynamics



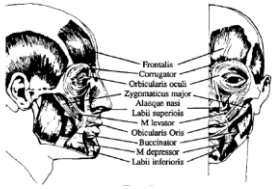


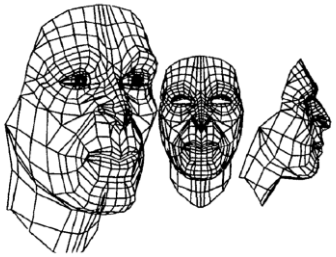
Einführung

- Masse-Feder-Systeme (Mass Spring Systems, kurz: MSS) zur darstellung deformierbarer Geometrie
- Die meisten Cloth-Simulationen gehen auf MSS zurück
- Anwendung finden sie auch in anderen Gebieten (z.B. Parameterisierung)

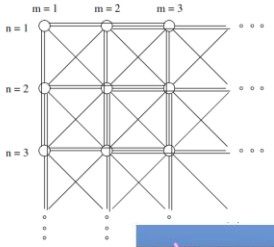


Entwicklung Stoffsimulation

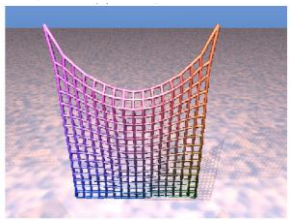




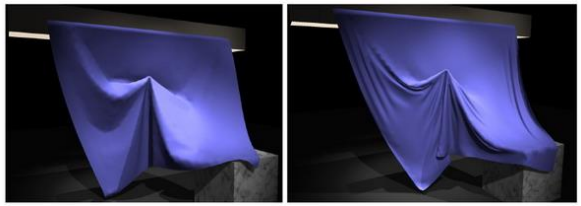
[Keith Waters87]



[Provot96]



[Mueller07]

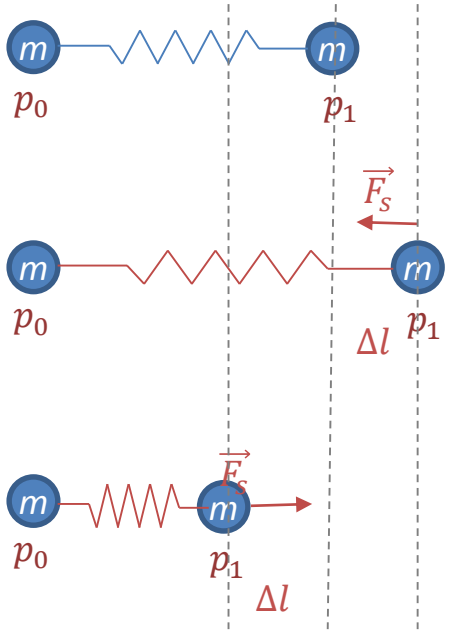


[Tyson13]

Masse-Feder System

- Aufbau
  - Ein Federsystem besteht aus Massepartikeln der Masse m
  - … und sie verbindende Federn mit Federhärte k und Ruhelänge l_0

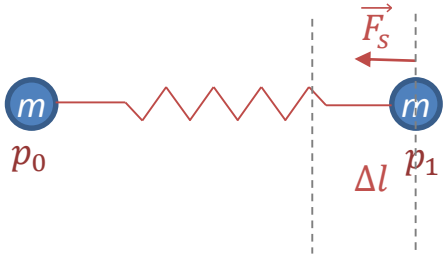
- Dynamik
  - Wird eine Feder aus ihrer Ruhelänge gebracht, entsteht eine Federkraft $\overrightarrow{F_s}$
  - Diese ist so gerichtet, daß sie der die deformation verursachenden Kraft entgegen wirkt.



Masse-Feder System

- Gesetz von Hooke
  - Die Kraft, die durch eine gestreckte/gestauchte Feder verursacht wird
    - Ist linear zur Deformation
    - Wirkt in die Richtung der Feder gegen die deformierende Kraft

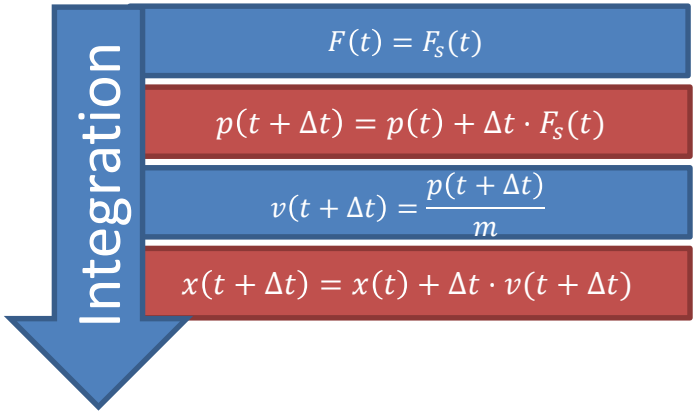
$$\overrightarrow{F_s} = -\frac{(\overrightarrow{p_1} - \overrightarrow{p_0})}{\|\overrightarrow{p_1} - \overrightarrow{p_0}\|} k \cdot \Delta l$$



Masse-Feder System

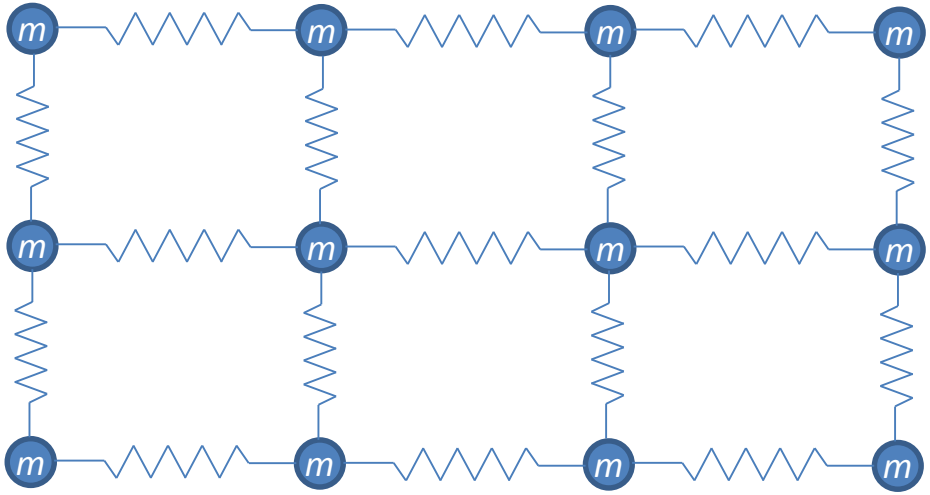
- Simulation durch lösen der Differenzialgleichung

$$v(t) = \dot{x}(t)$$
$$a(t) = \ddot{x}(t)$$



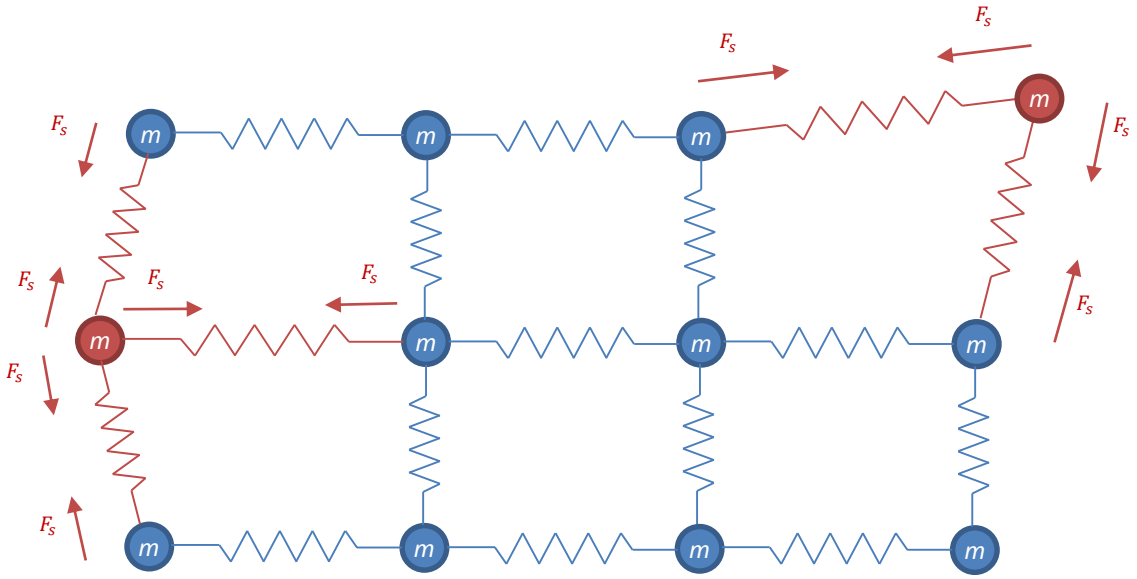
Masse-Feder-Netzwerke

- Aufbau eines Masse-Feder-Netzwerkes (MFN)



Masse-Feder-Netzwerke

- Dynamik eines deformierten MFN



Masse-Feder-Netzwerke

- Simulation

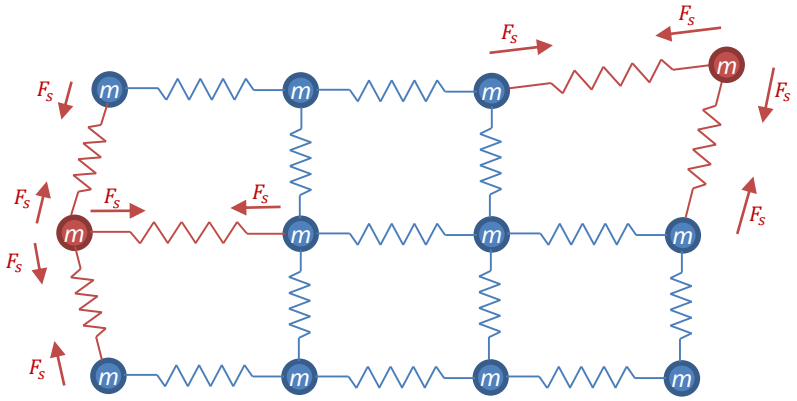
Integration

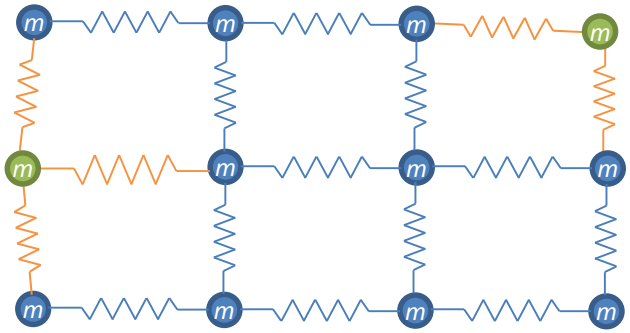
$$F(t) = F_{ext} + \sum_{j \in C} F_j(t)$$

$$p(t + \Delta t) = p(t) + \Delta t \cdot F(t)$$

$$v(t + \Delta t) = \frac{p(t + \Delta t)}{m}$$

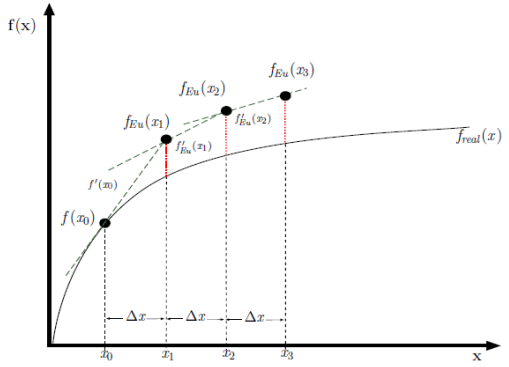
$$x(t + \Delta t) = x(t) + \Delta t \cdot v(t + \Delta t)$$



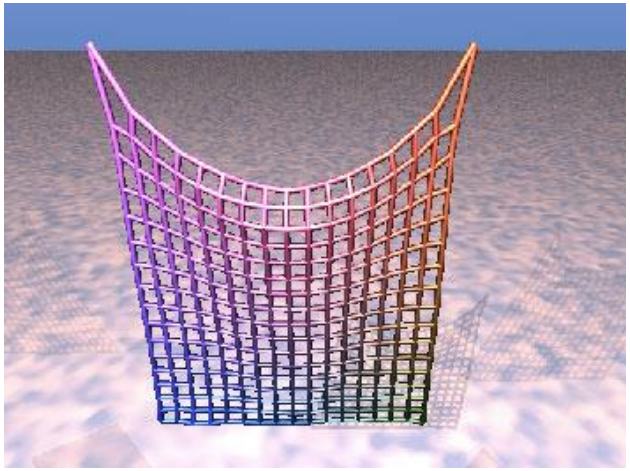


Masse-Feder-Netzwerke

- Fehler der Integration
  - Die DLG wird explizit gelöst
  - Der Fehler hängt vom Zeitschritt Δt ab
  - Die Simulation wird unstabil für hohe Zeitschritte



- "Superelastizität"
  - Das lineare Verhalten der Federn führt zu gummiartigen Resultaten
  - Erhöhen der Federhärte nicht unbegrenzt möglich (vergl. [Provot96])



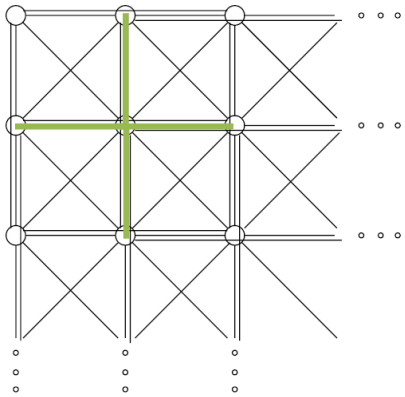
$$T_0 \approx \pi\sqrt{\frac{m}{k}} \Rightarrow \mathrm{k_{krit}} \approx m\frac{\Delta t^2}{\pi^2}$$

Stoff-Simulation mit MFN

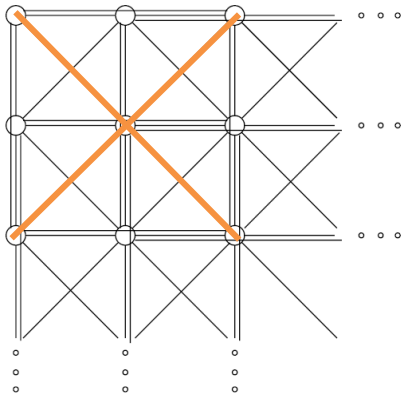
- Datenstruktur
  - Auf einem Gitter angeordnete Massepartikel; es gilt

$$m_{gesamt} = \sum_i m_i$$

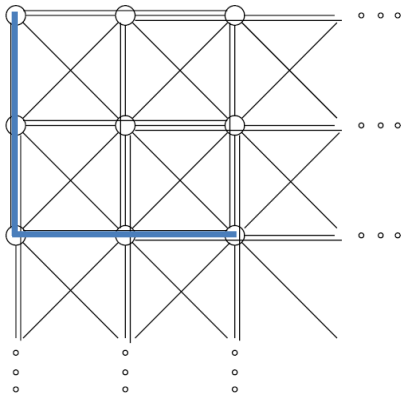
  - Federn, durch welche die Stoffbewegung beschränkt wird



Strukturfedern



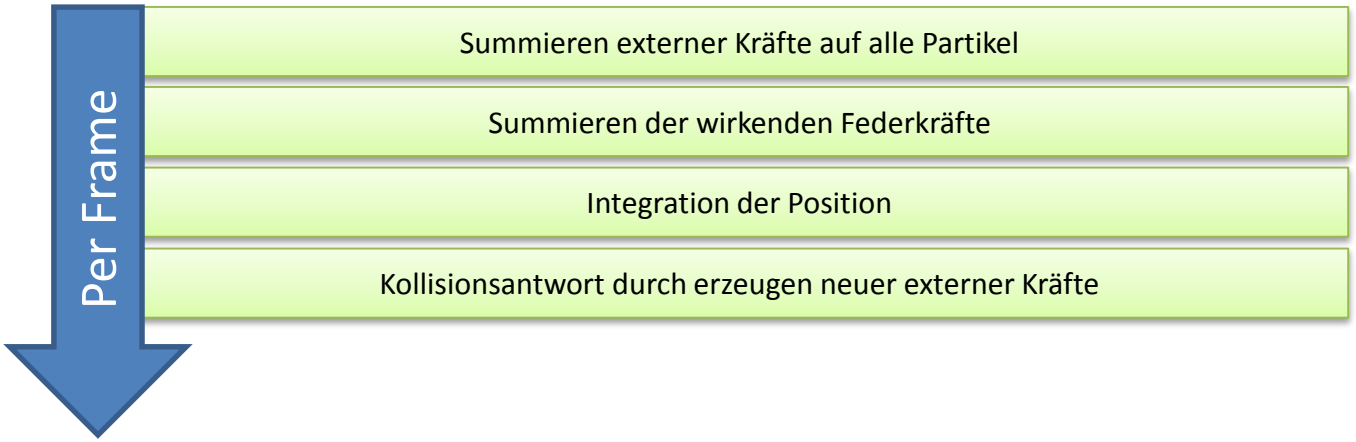
Scherfedern



Biegefedern

Stoff-Simulation mit MFN

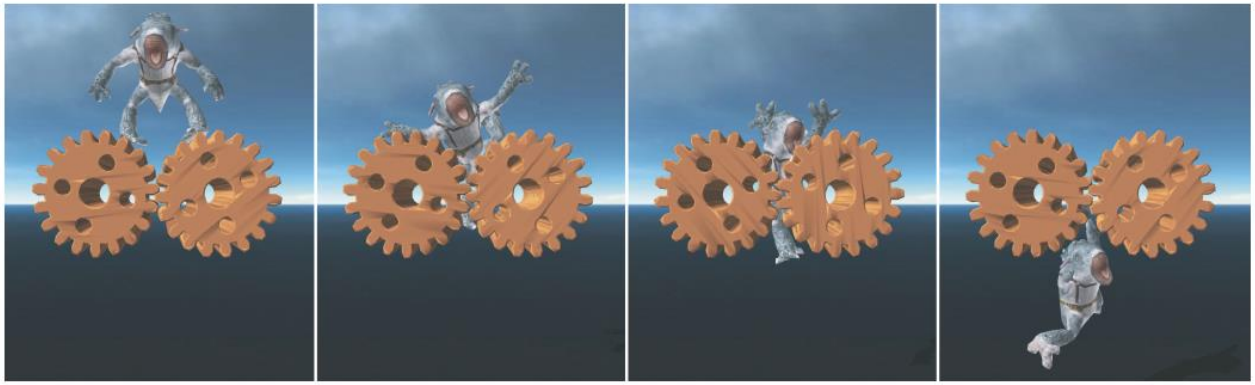
- Ablauf



Position-Based-Dynamics

- Probleme mit MFN
  - Stark unrobust für grosse Zeitschritte (numerische Stabilität!)
  - Nur begrenzt konfigurierbar, da Federhärten begrenzt sind (Eigenperiode!)

- Ansatz
  - Statt eine DGL 2. Ordnung zu lösen, wird ein System gelöst, dass direkt für die Positionen der Partikel löst → Position Based Dynamics



[Mueller07]

Position-Based-Dynamics

- Analog zu Masse-Feder-Netzwerken
  - Ein Massepartikel bestehen aus
    - Position: x_i
    - Geschätzte Position: p_i
    - Inverse Masse: w_i
    - Geschwindigkeit: v_i

Position-Based-Dynamics

- Analog zu Masse-Feder-Netzwerken
  - Federn werden durch *Constraints* (engl. Bedingung) ersetzt
  - Ein *Constraint* c_j besteht aus
    - Kardinalität: n_j
    - Kostenfunktion: C_j: $\mathbb{R}^{3n_j} \rightarrow \mathbb{R}$
    - Gewicht (stiffness): k_j
    - Typ: *Gleichung* oder *Ungleichung*
  - Ziel ist es nun, alle Constraints bestmöglich zu erfüllen, d.h. die Kosten zu minimieren.

Position-Based-Dynamics

- Algorithmus

(1) **forall** vertices i
(2) initialize $\mathbf{x}_i = \mathbf{x}_i^0, \mathbf{v}_i = \mathbf{v}_i^0, w_i = 1/m_i$
(3) **endfor**
(4) **loop**
(5) **forall** vertices i **do** $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t w_i \mathbf{f}_{ext}(\mathbf{x}_i)$
(6) dampVelocities$(\mathbf{v}_1, \ldots, \mathbf{v}_N)$
(7) **forall** vertices i **do** $\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$
(8) **forall** vertices i **do** generateCollisionConstraints$(\mathbf{x}_i \rightarrow \mathbf{p}_i)$
(9) **loop** solverIterations **times**
(10) projectConstraints$(C_1, \ldots, C_{M+M_{coll}}, \mathbf{p}_1, \ldots, \mathbf{p}_N)$
(11) **endloop**
(12) **forall** vertices i
(13) $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i)/\Delta t$
(14) $\mathbf{x}_i \leftarrow \mathbf{p}_i$
(15) **endfor**
(16) velocityUpdate$(\mathbf{v}_1, \ldots, \mathbf{v}_N)$
(17) **endloop**

Position-Based-Dynamics

- Algorithmus

(1) **forall** vertices i
(2) initialize $\mathbf{x}_i = \mathbf{x}_i^0, \mathbf{v}_i = \mathbf{v}_i^0, w_i = 1/m_i$
(3) **endfor**
(4) **loop**
(5) **forall** vertices i **do** $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t w_i \mathbf{f}_{ext}(\mathbf{x}_i)$
(6) dampVelocities$(\mathbf{v}_1, \ldots, \mathbf{v}_N)$
(7) **forall** vertices i **do** $\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$
(8) **forall** vertices i **do** generateCollisionConstraints$(\mathbf{x}_i \rightarrow \mathbf{p}_i)$
(9) **loop** solverIterations **times**
(10) projectConstraints$(C_1, \ldots, C_{M+M_{coll}}, \mathbf{p}_1, \ldots, \mathbf{p}_N)$
(11) **endloop**
(12) **forall** vertices i
(13) $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i)/\Delta t$
(14) $\mathbf{x}_i \leftarrow \mathbf{p}_i$
(15) **endfor**
(16) velocityUpdate$(\mathbf{v}_1, \ldots, \mathbf{v}_N)$
(17) **endloop**

Geschwindigkeiten integrieren

Position-Based-Dynamics

- Algorithmus

```
(1)  forall vertices i
(2)     initialize x_i = x_i^0, v_i = v_i^0, w_i = 1/m_i
(3)  endfor
(4)  loop
(5)     forall vertices i do v_i ← v_i + Δt w_i f_ext(x_i)
(6)     dampVelocities(v_1, ..., v_N)
(7)     forall vertices i do p_i ← x_i + Δt v_i
(8)     forall vertices i do generateCollisionConstraints(x_i → p_i)
(9)     loop solverIterations times
(10)       projectConstraints(C_1, ..., C_{M+M_coll}, p_1, ..., p_N)
(11)    endloop
(12)    forall vertices i
(13)       v_i ← (p_i − x_i)/Δt
(14)       x_i ← p_i
(15)    endfor
(16)    velocityUpdate(v_1, ..., v_N)
(17) endloop
```

Positionen schätzen

Position-Based-Dynamics

- Algorithmus

(1) **forall** vertices i
(2) initialize $\mathbf{x}_i = \mathbf{x}_i^0, \mathbf{v}_i = \mathbf{v}_i^0, w_i = 1/m_i$
(3) **endfor**
(4) **loop**
(5) **forall** vertices i **do** $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t w_i \mathbf{f}_{ext}(\mathbf{x}_i)$
(6) dampVelocities$(\mathbf{v}_1, \ldots, \mathbf{v}_N)$
(7) **forall** vertices i **do** $\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$
(8) **forall** vertices i **do** generateCollisionConstraints$(\mathbf{x}_i \rightarrow \mathbf{p}_i)$
(9) **loop** solverIterations **times**
(10) projectConstraints$(C_1, \ldots, C_{M+M_{coll}}, \mathbf{p}_1, \ldots, \mathbf{p}_N)$
(11) **endloop**
(12) **forall** vertices i
(13) $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i)/\Delta t$
(14) $\mathbf{x}_i \leftarrow \mathbf{p}_i$
(15) **endfor**
(16) velocityUpdate$(\mathbf{v}_1, \ldots, \mathbf{v}_N)$
(17) **endloop**

Positionsschätzungen
Aktualisieren
→ Constraints lösen

Position-Based-Dynamics

- Algorithmus

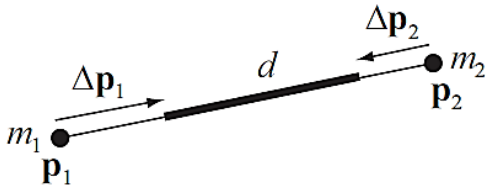
(1) **forall** vertices i
(2) initialize $\mathbf{x}_i = \mathbf{x}_i^0, \mathbf{v}_i = \mathbf{v}_i^0, w_i = 1/m_i$
(3) **endfor**
(4) **loop**
(5) **forall** vertices i **do** $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t w_i \mathbf{f}_{ext}(\mathbf{x}_i)$
(6) dampVelocities$(\mathbf{v}_1, \ldots, \mathbf{v}_N)$
(7) **forall** vertices i **do** $\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$
(8) **forall** vertices i **do** generateCollisionConstraints$(\mathbf{x}_i \rightarrow \mathbf{p}_i)$
(9) **loop** solverIterations **times**
(10) projectConstraints$(C_1, \ldots, C_{M+M_{coll}}, \mathbf{p}_1, \ldots, \mathbf{p}_N)$
(11) **endloop**
(12) **forall** vertices i
(13) $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i)/\Delta t$
(14) $\mathbf{x}_i \leftarrow \mathbf{p}_i$
(15) **endfor**
(16) velocityUpdate$(\mathbf{v}_1, \ldots, \mathbf{v}_N)$
(17) **endloop**

Geschw. update
und Positionen setzen

Position-Based-Dynamics

- Distanz-Constraint (distance)
  - Modellierung einer Feder (lineare „Kosten“ abhängig vom Abstand)

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



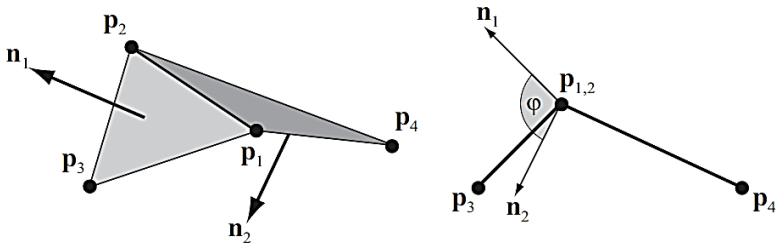
  - Projektion des Constraint:

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

Position-Based-Dynamics

- Biege-Constraint (bending)
  - Modellierung der Festigkeit des Materials

$$C(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4) = \arccos(\mathbf{n}_1 \cdot \mathbf{n}_2) - \varphi_0$$



  - Projektion des Constraint (*Siehe [Müller07], Appendix A*):

$$\mathbf{q}_3 = \frac{\mathbf{p}_2 \times \mathbf{n}_2 + (\mathbf{n}_1 \times \mathbf{p}_2)d}{|\mathbf{p}_2 \times \mathbf{p}_3|}$$

$$\mathbf{q}_4 = \frac{\mathbf{p}_2 \times \mathbf{n}_1 + (\mathbf{n}_2 \times \mathbf{p}_2)d}{|\mathbf{p}_2 \times \mathbf{p}_4|}$$

$$\mathbf{q}_2 = -\frac{\mathbf{p}_3 \times \mathbf{n}_2 + (\mathbf{n}_1 \times \mathbf{p}_3)d}{|\mathbf{p}_2 \times \mathbf{p}_3|} - \frac{\mathbf{p}_4 \times \mathbf{n}_1 + (\mathbf{n}_2 \times \mathbf{p}_4)d}{|\mathbf{p}_2 \times \mathbf{p}_4|}$$

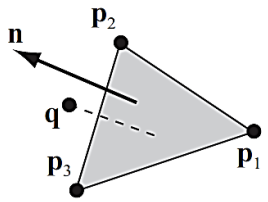
$$\mathbf{q}_1 = -\mathbf{q}_2 - \mathbf{q}_3 - \mathbf{q}_4$$

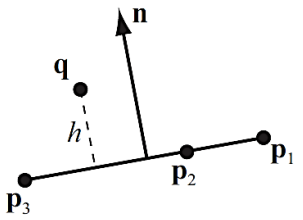
$$\Delta\mathbf{p}_i = -\frac{w_i\sqrt{1-d^2}(\arccos(d) - \varphi_0)}{\sum_j w_j |\mathbf{q}_j|^2}\,\mathbf{q}_i$$

Position-Based-Dynamics

- Kollisions-Konstraint

$$C(\mathbf{p}) = (\mathbf{p} - \mathbf{q}_s) \cdot \mathbf{n}_s + \mathrm{h}$$





  - Projektion des Constraint:
    - Siehe Kollisionsantwort aus den Vergangenen Vorlesungen
    - Die Geschwindigkeit muss gesondert beachtet werden