Implizite Integration

- Bisherige Schemata sind nur *conditionally stable*
  - D.h.: Nur stabil für Δt in einem bestimmten Bereich
  - Dieser Bereich hängt von der Steifigkeit der Federn ab
- Ziel: *unconditionally stable*
- Eine Möglichkeit: implizite Euler-Integration

explizit

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \Delta t \mathbf{v}_i^t$$

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \Delta t \frac{1}{m_i} \mathbf{f}(\mathbf{x}^t)$$

implizit

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \Delta t \mathbf{v}_i^{t+1}$$

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \Delta t \frac{1}{m_i} \mathbf{f}(\mathbf{x}^{t+1})$$

- Wir haben jetzt ein System von nicht-linearen, algebraischen Gleichungen, mit $\mathbf{x}^{t+1}$ und $\mathbf{v}^{t+1}$ als Unbekannte auf beiden Seiten → implizite Integration

Lösungsmethode

- Schreibe das gesamte Feder-Masse-System mit Vektoren:

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{pmatrix} = \begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ \vdots \\ x_{n3} \end{pmatrix} \quad \mathbf{v} = \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_n \end{pmatrix} = \begin{pmatrix} v_{11} \\ v_{12} \\ v_{13} \\ v_{21} \\ v_{22} \\ \vdots \\ v_{n3} \end{pmatrix} \quad \mathbf{f}(\mathbf{x}) = \begin{pmatrix} \mathbf{f}_1(\mathbf{x}) \\ \vdots \\ \mathbf{f}_n(\mathbf{x}) \end{pmatrix}$$

$$\mathbf{f}_i = \begin{pmatrix} f_{i1}(\mathbf{x}) \\ f_{i2}(\mathbf{x}) \\ f_{i3}(\mathbf{x}) \end{pmatrix} \quad M_{3n\times 3n} = \begin{pmatrix} m_1 &&&&&&&& \\ & m_1 &&&&&&& \\ && m_1 &&&&&& \\ &&& m_2 &&&&& \\ &&&& m_2 &&&& \\ &&&&& \ddots &&& \\ &&&&&& m_n && \\ &&&&&&& m_n & \\ &&&&&&&& m_n \end{pmatrix}$$

- Schreibe die vielen impliziten Gleichungen als ein großes Gleichungssystem um :

$$M\mathbf{v}^{t+1} = M\mathbf{v}^t + \Delta t\mathbf{f}(\mathbf{x}^{t+1}) \quad (1)$$

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \Delta t\,\mathbf{v}^{t+1} \quad (2)$$

- Einsetzen von (2) in (1) ergibt:

$$M\mathbf{v}^{t+1} = M\mathbf{v}^t + \Delta t\,\mathbf{f}(\,\mathbf{x}^t + \Delta t\mathbf{v}^{t+1}\,)$$

- Die Taylor-Reihe für **f** ist:

$$\mathbf{f}(\mathbf{x}^t + \Delta t\,\mathbf{v}^{t+1}) = \mathbf{f}(\,\mathbf{x}^t) + \frac{\partial}{\partial \mathbf{x}}\mathbf{f}(\,\mathbf{x}^t) \cdot (\Delta t\,\mathbf{v}^{t+1}) + O((\,\Delta t\,\mathbf{v}^{t+1})^2)$$

- Einsetzen:

$$\begin{aligned} M\,\mathbf{v}^{t+1} &= M\,\mathbf{v}^t + \Delta t \left(\mathbf{f}(\mathbf{x}^t) + \underbrace{\frac{\partial}{\partial x}\mathbf{f}(x^t)}_{K} \cdot \left(\Delta t\,\mathbf{v}^{t+1}\right) \right) \\ &= M\,\mathbf{v}^t + \Delta t\,\mathbf{f}(\mathbf{x}^t) + \Delta t^2\,K\,\mathbf{v}^{t+1} \end{aligned}$$

- K ist die Jacobi-Matrix (= Ableitung):

$$K = \begin{pmatrix} \frac{\partial}{\partial x_{11}} f_{11} & \frac{\partial}{\partial x_{12}} f_{11} & \cdots & \frac{\partial}{\partial x_{n3}} f_{11} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial x_{11}} f_{n3} & \cdots & \cdots & \frac{\partial}{\partial x_{n3}} f_{n3} \end{pmatrix}$$

  - Heißt tangent stiffness matrix
    - (Die normale Steifigkeitsmatrix wird im Gleichgewichtszustand ausgewertet; hier aber an einer beliebigen, aktuellen Position des Systems; daher der Name "*tangent* ...")

- Terme umordnen:

$$\left(M - \Delta t^{\,2} K\right) \mathbf{v}^{t+1} = M\,\mathbf{v}^{t} + \Delta t\,\mathbf{f}(\mathbf{x}^{t})$$

- Das ist von der Form:

$$A\,\mathbf{v}^{t+1} = \mathbf{b}$$

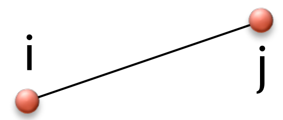
$$\text{mit} \quad A \in \mathbb{R}^{3n\,x\,3n}, \quad b \in \mathbb{R}^{3n}$$

- Löse dieses LGS mittels einer iterativen Methode
  - Denn: A ändert sich in jedem Frame

Berechnung der Steifigkeitsmatrix

- Die Anatomie der Matrix K :
  - Eine Feder (i,j) addiert folgende vier 3x3 Untermatrizen zu K :

$$\begin{array}{cc} & \\ \begin{array}{r} 3i \rightarrow \\[2em] 3j \rightarrow \end{array} & \begin{pmatrix} \boxed{K_{ii}} & \boxed{K_{ij}} \\[1em] \boxed{K_{ji}} & \boxed{K_{jj}} \end{pmatrix} \\ & \begin{array}{cc} \uparrow & \uparrow \\ 3i & 3j \end{array} \end{array}$$



  - Die Matrix K_{ij} entsteht durch die Ableitung von $\mathbf{f}_i = (f_{i1}, f_{i2}, f_{i3})$ nach $\mathbf{x}_j = (x_{j1}, x_{j2}, x_{j3})$:

$$K_{ij} = \begin{pmatrix} \frac{\partial}{\partial x_{j1}} f_{i1} & \frac{\partial}{\partial x_{j2}} f_{i1} & \frac{\partial}{\partial x_{j3}} f_{i1} \\ \vdots & & \vdots \\ \frac{\partial}{\partial x_{j1}} f_{i3} & \cdots & \frac{\partial}{\partial x_{j3}} f_{i3} \end{pmatrix}$$

  - Betrachte im folgenden nur f^S (Federkraft)

- Zunächst K_{ii}:

$$
\begin{aligned}
K_{ii} &= \frac{\partial}{\partial \mathbf{x}_i} f_i(\mathbf{x}_i, \mathbf{x}_j) \\
&= k_s \frac{\partial}{\partial \mathbf{x}_i} \left((\mathbf{x}_j - \mathbf{x}_i) - l_0 \frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|} \right) \\
&= k_s \left(-I - l_0 \frac{-I \cdot \|\mathbf{x}_j - \mathbf{x}_i\| - (\mathbf{x}_j - \mathbf{x}_i) \cdot 2 \frac{(\mathbf{x}_j - \mathbf{x}_i)^\top}{\|\mathbf{x}_j - \mathbf{x}_i\|}}{\|\mathbf{x}_j - \mathbf{x}_i\|^2} \right) \\
&= k_s \left(-I + l_0 \frac{1}{\|\mathbf{x}_j - \mathbf{x}_i\|} I + \frac{2 l_0}{\|\mathbf{x}_j - \mathbf{x}_i\|^3} (\mathbf{x}_j - \mathbf{x}_i)(\mathbf{x}_j - \mathbf{x}_i)^\mathsf{T} \right)
\end{aligned}
$$

- Aus einigen Symmetrien folgt:
  - $$K_{ij} = \frac{\partial}{\partial \mathbf{x}_j} f_i(\mathbf{x}_i, \mathbf{x}_j) = -K_{ii}$$
  - $$K_{jj} = \frac{\partial}{\partial x_j} f_j(\mathbf{x}_i, \mathbf{x}_j) = \frac{\partial}{\partial \mathbf{x}_j} (-\mathbf{f}_i(\mathbf{x}_i, \mathbf{x}_j)) = K_{ii}$$
  - $$K_{ji} = K_{ij}$$

- Zur Erinnerung:
  - $$\left(\frac{f}{g}\right)' = \frac{f'g - fg'}{g^2}$$
  - $$\frac{\partial}{\partial \mathbf{x}} \|\mathbf{x}\| = \frac{\partial}{\partial \mathbf{x}} \left(\sqrt{x_1^2 + x_2^2 + x_3^2} \right) = 2 \frac{\mathbf{x}^\mathsf{T}}{\|\mathbf{x}\|}$$

Lösungsverfahren

- Setze $K = 0$
- Für jede Feder (i,j) berechne K_{ii}, K_{ij}, K_{ji}, K_{jj} und akkumuliere zu K an den richtigen Stellen
- Berechne $\mathbf{b} = M\,\mathbf{v}^t + \Delta t\, f(\mathbf{x}^t)$
- Löse das LGS $\rightarrow \mathbf{v}^{t+1}$
- Berechne $\mathbf{x}^{t+1} = \mathbf{x}^t + \Delta t\,\mathbf{v}^{t+1}$

Vor- und Nachteile

- Explizite Integration:

  + Sehr einfach zu programmieren

  - kleine Schrittweite nötig

  - Steife Federn funktionieren nicht gut

  - Kräfte werden nur um eine Feder pro Schritt propagiert

- Implizite Integration:

  + Unconditionally stable

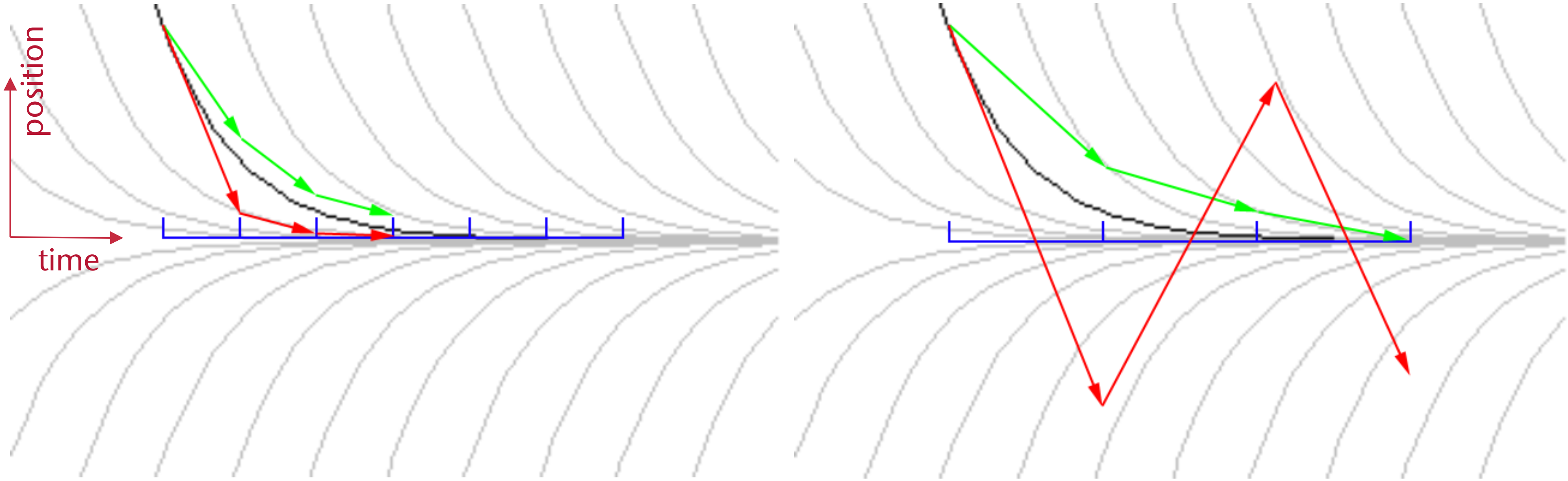
  + Steife Federn werden besser handhabbar

  + Globaler Solver → Kräfte werden schon bei einem Simulationsschritt durch das ganze System propagiert

  - Große Schrittweiten nötig, da ein Schritt sehr teuer (und in Wahrheit schon aus vielen Einzelschritten besteht)

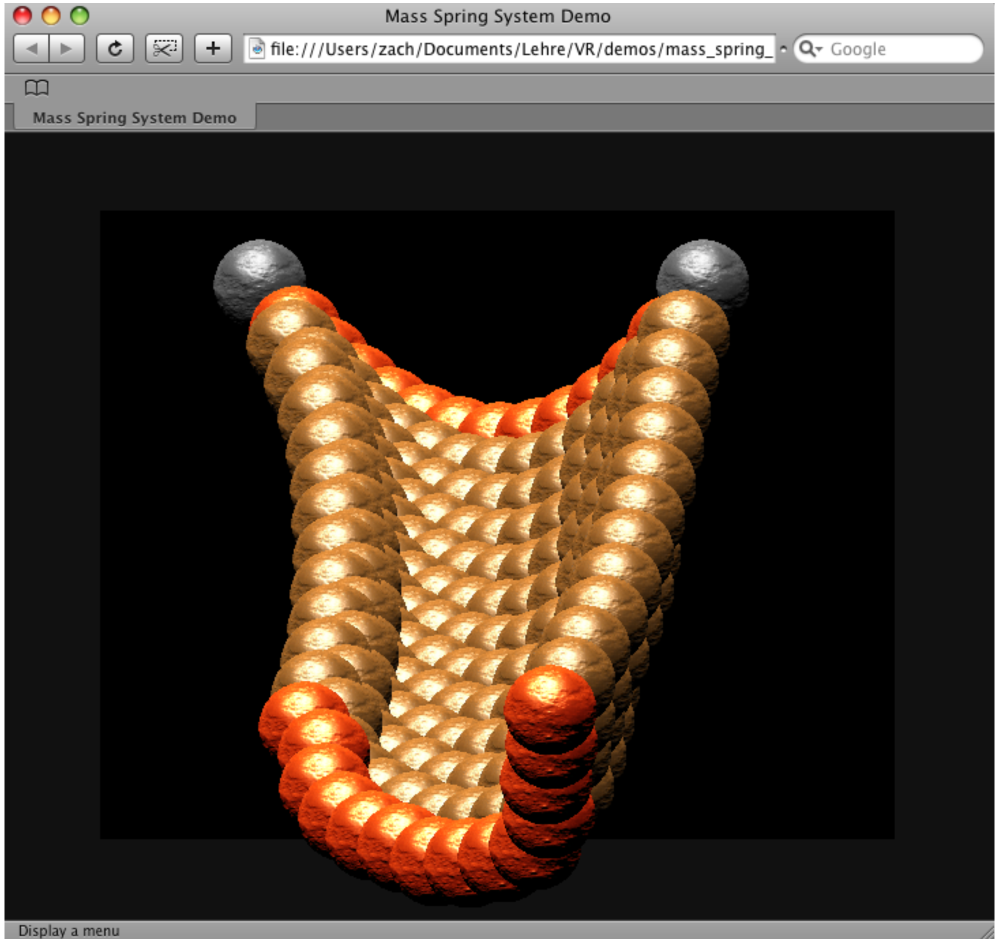
  - Unerwünschte Dämpfung durch das Integrationsverfahren selbst

- Visualisierung: $\dot{x}(t) = -x(t)$



- Umgangssprachliche Beschreibung:
  - Explizit springt blind vorwärts, basierend auf der aktuellen Information
  - Implizit springt "rückwärts" und sucht dabei eine "nächste" Position genau so, daß der Rückwärtssprung bei der aktuellen Position landet

Demo

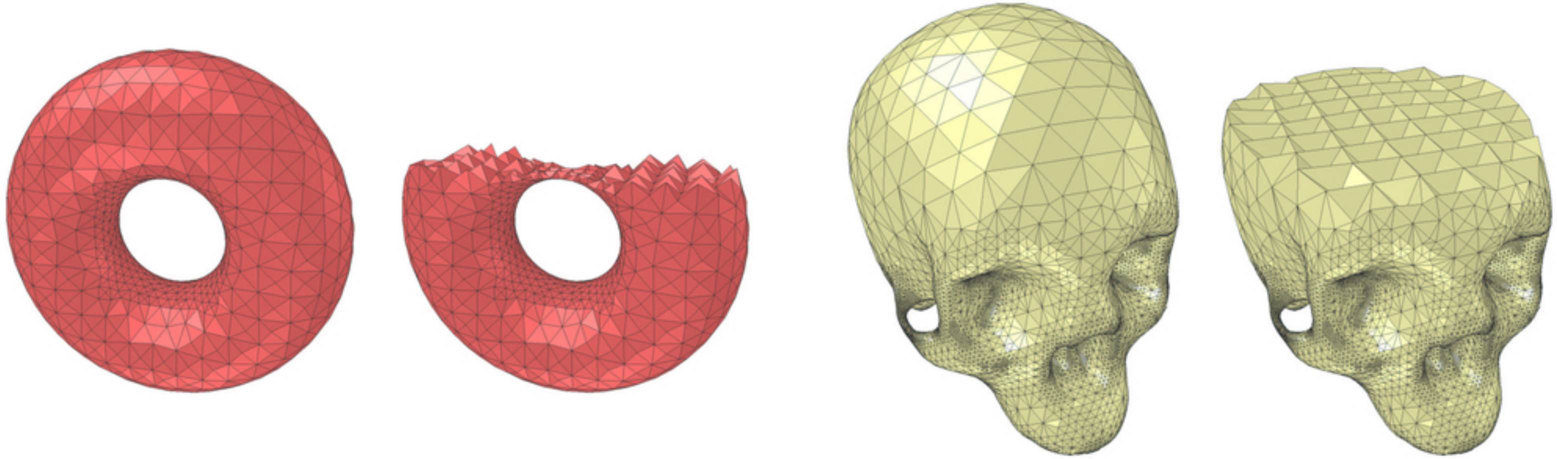


http://www.dhteumeuleu.com/dhtml/v-grid.html

Mesh-Erzeugung für Objekte mit Volumen

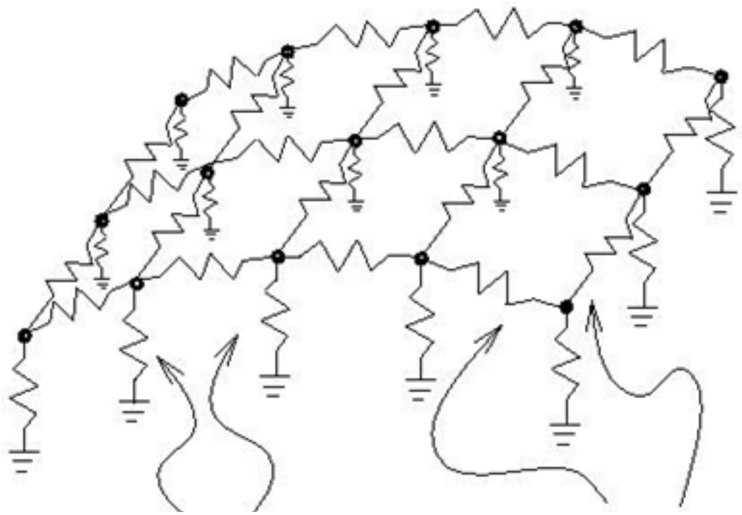
- Wie erzeugt man ein Feder-Masse-System aus einem 3D-Modell?
- Direkt 3D-Geometrie in Masse-Feder-System übersetzen liefert keine guten Resultate:
  - Geometrie oft zu hoch aufgelöst
  - Degenerierte Polygone
- Bessere (sehr einfache) Idee:
  - Erzeuge ein Tetraeder-Mesh (irgendwie) aus der Geometrie
  - Jeder Vertex wird ein Massepunkt, jede Kante eine Feder
  - Verteile die Masse der Tetraeder (= Dichte × Volumen) gleichmäßig auf die Massepunkte

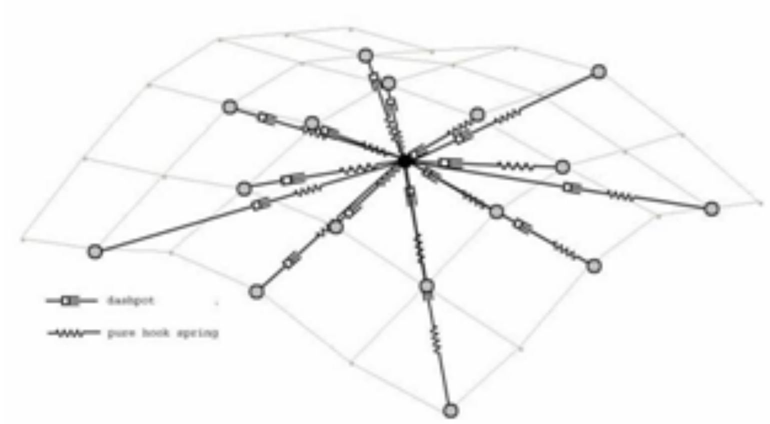
- Erzeugung des Tetraeder-Meshes (einfache Methode):
  - Verteile eine Menge von Punkten gleichmäßig (evtl. zufällig) im Inneren der Geometrie (sog. "Steiner-Punkte")
  - Dito in einer Schicht über der Oberfläche



- Verankerung des Oberflächen-Meshes im Tetraeder-Mesh:
  - Stelle jeden Vertex des Oberflächen-Meshes als baryzentrische Kombination der Tetraeder-Vertices dar, in dem er enthalten ist

- Bei 2-mannigfaltigen Feder-Masse-Systemen:
  - Verankere evtl. die Federn an einer Ruhelage
  - Führe evtl. "Querverstrebungen" ein



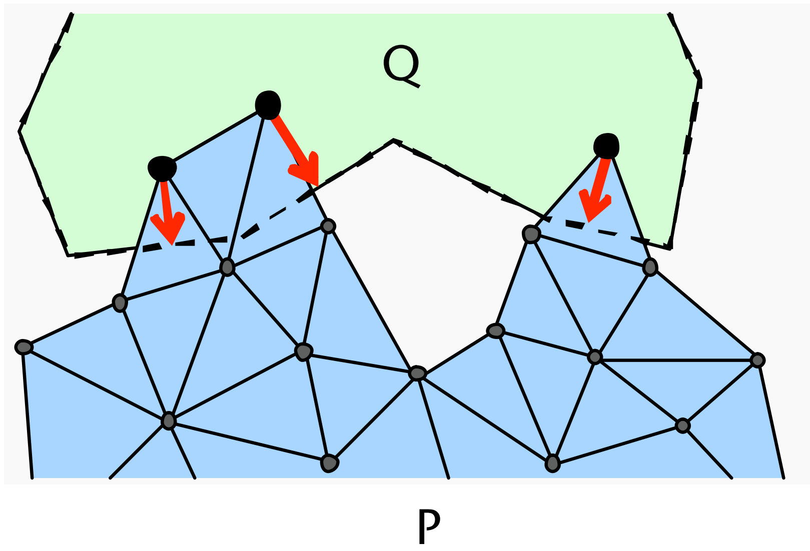


Kollisionserkennung

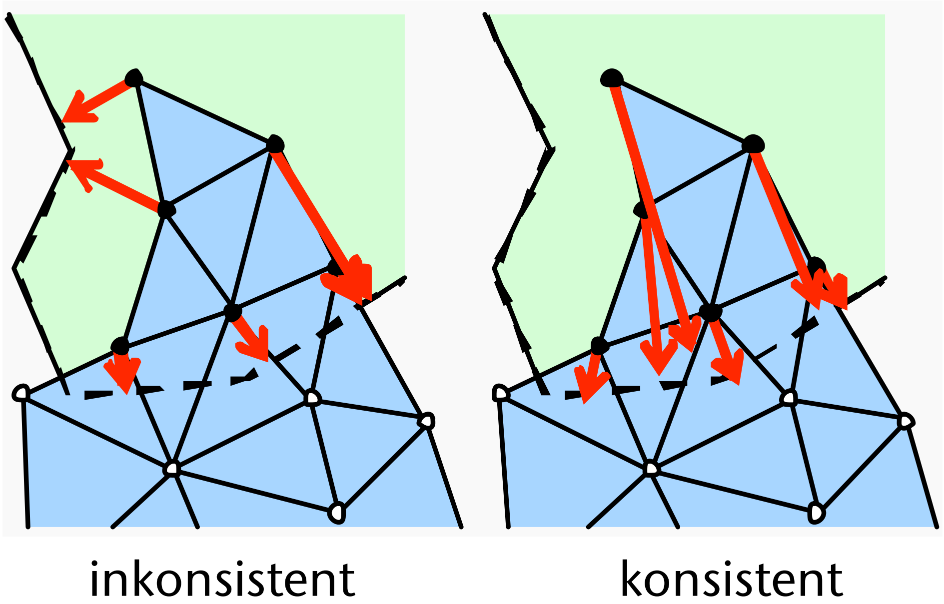
- Sortiere die Tetraeder in ein 3D Gitter (Hash-Tabelle!) ein
- Bei Kollision in der Hash-Tabelle:
  - Führe exakten Schnitttest zwischen 2 Tetraedern durch

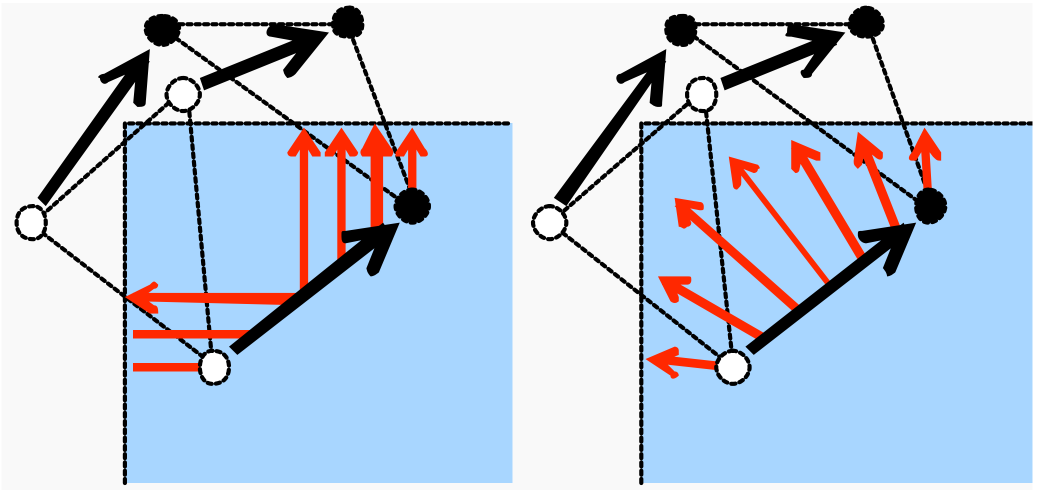
Kollisionsantwort (collision response)

- Aufgabenstellung: gegeben zwei kollidierende Objekte P und Q (Tetraeder-Meshes) — welches ist die Rückstellkraft (penalty force)?
- Naiver Ansatz:
  - Berechne für jeden eingedrungenen Massepunkt von P die kleinste Distanz zur Oberfläche von Q → Kraft (Betrag + Richtung)
  - Problem:
    - unplausible Kräfte (implausibel?)
    - "Durchtunneln" (s. a. Force-Feedback-Kapitel)



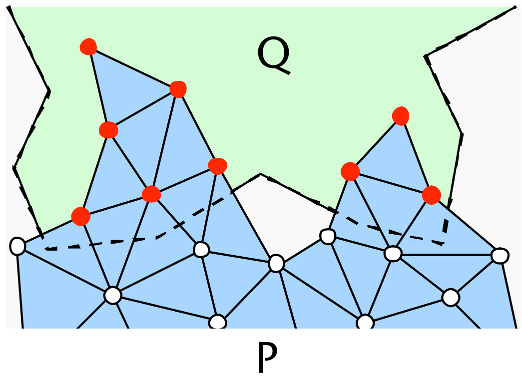
- Beispiele:

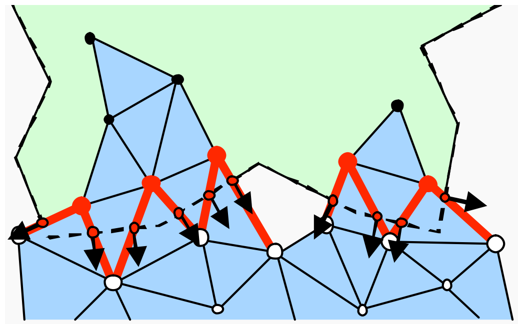




Konsistente Penalty Forces

1. Phase: identifiziere alle eindringenden Punkte von P
2. Phase: bestimme alle schneidenden Kanten von P
   - Berechne zu jeder solchen Schnittkante den exakten Schnittpunkt $\mathbf{x}_i$
   - Berechne zu jedem Schnittpunkt eine Normale $\mathbf{n}_i$
     - Z.B. mittels baryzentrischer Interpolation der Vertexnormalen von Q

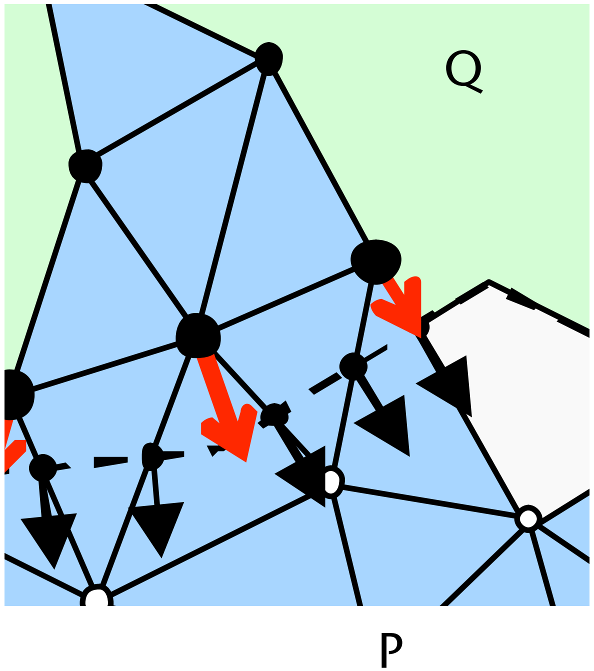




3. Phase: berechne ungefähre Kraft für "Randpunkte"

- Randpunkt = eingedrungener Punkt **p** inzident zu einer Schnittkante
- Beobachtung: ein Randpunkt kann zu mehreren Schnittkanten inzident sein
- Eindringtiefe = gewichtete Summe

$$d(\mathbf{p}) = \frac{\sum_{i=1}^{k} \omega(\mathbf{x}_i, \mathbf{p})\,(\mathbf{x}_i - \mathbf{p})\cdot\mathbf{n}_i}{\sum_{i=1}^{k} \omega(\mathbf{x}_i, \mathbf{p})}$$



wobei $d(\mathbf{p})$ = approx. Eindringtiefe von Massepunkt **p**, $\mathbf{x}_i$ = Schnittpunkt der Schnittkante inzident zu **p**, $\mathbf{n}_i$ = Normale zur Oberfläche von Q im Schnittpunkt $\mathbf{x}_i$,

und $\omega(\mathbf{x}_i, \mathbf{p}) = \frac{1}{\|\mathbf{x}_i - \mathbf{p}\|}$

- Richtung der Kraft für Randpunkte:

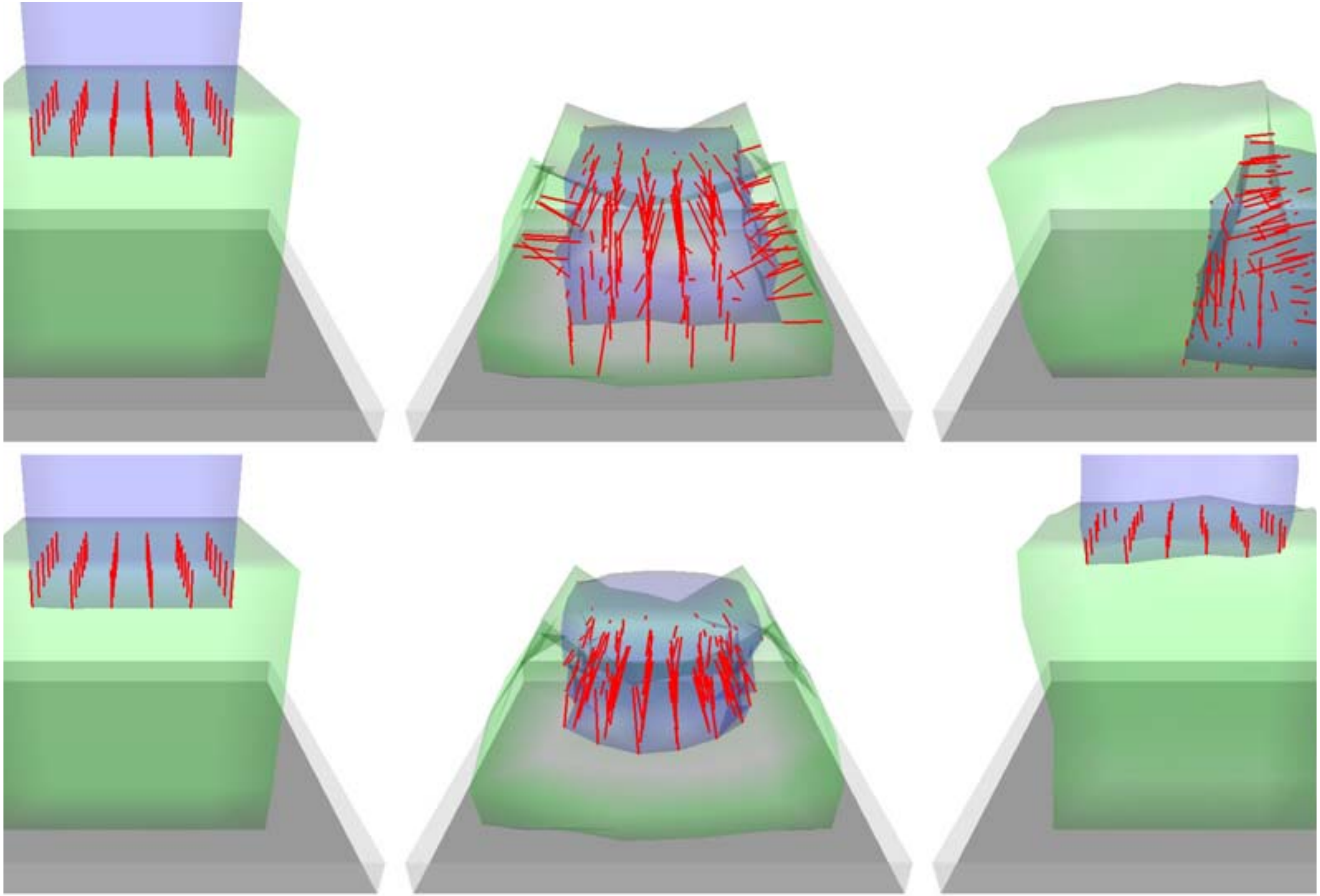
$$\hat{\mathbf{r}}(\mathbf{p}) = \frac{\sum_{i=1}^{k} \omega(\mathbf{x}_i, \mathbf{p})\, \mathbf{n}_i}{\sum_{i=1}^{k} \omega(\mathbf{x}_i, \mathbf{p})} \qquad \mathbf{r}(\mathbf{p}) = \hat{\mathbf{r}}(\mathbf{p})^0$$

4. Phase: Propagation mittels breadth-first traversal durch das Tetraeder-Mesh

$$d(\mathbf{p}) = \frac{\sum_{i=1}^{k} \omega(\mathbf{p}_i, \mathbf{p}) \big((\mathbf{p}_i - \mathbf{p}) \cdot \mathbf{r}_i + d(\mathbf{p}_i) \big)}{\sum_{i=1}^{k} \omega(\mathbf{x}_i, \mathbf{p})}$$

wobei $\mathbf{p}_i$ = schon besuchter eindringender Punkt von P, $\mathbf{p}$ = noch nicht besuchter Punkt, $\mathbf{r}_i$ = Richtung der geschätzten penalty force in Punkt $\mathbf{p}_i$.

Viualisierung



Video

http://cg.informatik.uni-freiburg.de