






# Virtuelle Realität Feder-Masse-Systeme



G. Zachmann  
Clausthal University, Germany  
[cg.in.tu-clausthal.de](http://cg.in.tu-clausthal.de)

## Die Newton'schen Gesetze

1. Gesetz (Trägheitsgesetz):
 

Ein kräftefreier Körper bewegt sich geradlinig mit konstanter Geschwindigkeit weiter.

  - Ein Körper in Ruhe ist also nur ein Spezialfall hiervor.
2. Gesetz (Aktionsprinzip):
 

Wenn eine Kraft  $F$  auf einen Körper mit Masse  $m$  wirkt, beschleunigt sie diesen gemäß

$$F = m \cdot a$$

  - Mit anderen Worten: Kraft und Beschleunigung sind proportional zueinander; die Proportionalitätskonstante ist  $m$ . Insbesondere haben beide dieselbe Richtung.

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 2

3. Gesetz (Reaktionsprinzip):

Wenn die Kraft  $F$ , die auf einen Körper wirkt, ihren Ursprung in einem anderen Körper hat, so wirkt auf diesen die entgegengesetzte Kraft  $-F$ .

- In der Schule lernt man : "Kraft = Gegenkraft"

4. Gesetz (Superpositionsprinzip):

Wirken auf einen Punkt (oder einen starren Körper) mehrere Kräfte  $F_1, \dots, F_n$ , so addieren sich diese vektoriell zu einer resultierenden Kraft  $F$  auf:

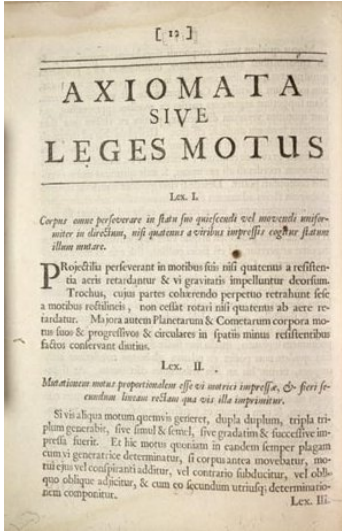
$$F = F_1 + \dots + F_n .$$

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 3

Historischer Exkurs

- Newton's Gesetze in der Originalschrift  
*Principia Mathematica* (1687):

- Lex I. Corpus omne perseverare in statu suo quiescendi vel movendi uniformiter in directum, nisi quatenus illud a viribus impressis cogitur statum suum mutare.*
- Lex II. Mutationem motus proportionalem esse vi motrici impressae, et fieri secundum lineam rectam qua vis illa imprimitur.*



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 4

- Definition:
 

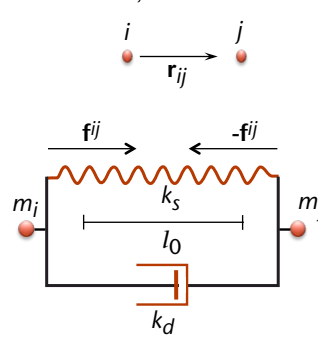
Ein **Feder-Masse-System** ist ein System, bestehend aus:

  1. einer Menge von Punktmassen  $m_i$  mit Positionen  $\mathbf{x}_i$  und Geschwindigkeit  $\mathbf{v}_i, i = 1 \dots N$ ;
  2. einer Menge von Federn  $s_{ij} = (i, j, k_s, k_d)$ , die die Massen  $i$  und  $j$  verbindet, mit der Ruhelänge  $l_0$ , Federkonstante (= Steifigkeit)  $k_s$  und den Dämpfungskoeffizienten  $k_d$
- Vorteile:
  - Sehr einfach zu programmieren
  - Ideal zum Studium verschiedener numerischer Lösungsverfahren
  - *Ubiquitous in games* (cloths, capes, ...)
- Nachteile:
  - Einige Parameter (Federkonstanten z.B.) sind nicht offensichtlich
  - Keine volumetrischen Effekte (z.B. Volumenerhaltung)

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 5

### Eine einzelne Feder (mit Dämpfer)

- Gegeben: Massen  $m_i$  und  $m_j$  mit Positionen  $\mathbf{x}_i, \mathbf{x}_j$
- Definiere  $\mathbf{r}_{ij} = \frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|}$
- Kräfte zwischen den beiden Massen :
  1. Von der Feder:
 
$$\mathbf{f}_s^{ij} = k_s \mathbf{r}_{ij} (\|\mathbf{x}_j - \mathbf{x}_i\| - l_0)$$
 wirkt auf Masse  $m_i$  in Richtung  $m_j$
  2. Durch den Dämpfer:  $\mathbf{f}_d^{ij} = k_d ((\mathbf{v}_j - \mathbf{v}_i) \cdot \mathbf{r}_{ij}) \mathbf{r}_{ij}$
  3. Zusammen :  $\mathbf{f}^{ij} = \mathbf{f}_s^{ij} + \mathbf{f}_d^{ij}$
  4. Kraft auf  $m_j$ :  $\mathbf{f}^{ji} = -\mathbf{f}^{ij}$
- Aus (4) → der Impuls bleibt erhalten!



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 6

▪ Alternative Federkraft:

$$\mathbf{f}_s^{ij} = k_s r_{ij} \frac{|\mathbf{x}_j - \mathbf{x}_i| - l_0}{l_0}$$

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 7

## Simulation einer einzelnen Feder

- Aus Newton'schen Gesetzen folgt:  $\ddot{\mathbf{x}} = \frac{1}{m} \mathbf{f}$
- DGL der Ordnung 2 umwandeln in DGLs erster Ordnung:
 
$$\dot{\mathbf{v}}(t) = \frac{1}{m} \mathbf{f}(t)$$

$$\dot{\mathbf{x}}(t) = \mathbf{v}(t)$$
- Anfangswerte:  $\mathbf{v}(t_0) = \mathbf{v}_0$ ,  $\mathbf{x}(t_0) = \mathbf{x}_0$
- "Simulation" = "Integration von DGLs über die Zeit"
- Taylor-Expansion liefert:
 
$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t \dot{\mathbf{x}}(t) + O(\Delta t^2)$$
- Analog:  $\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \Delta t \dot{\mathbf{v}}(t)$

→ Dieses Integrationsverfahren heißt **explizite Euler-Integration**

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 8

## Der Algorithmus

```

forall particles i :
  initialize  $x_i, v_i, m_i$ 

loop:
  forall particles i :
     $f_i \leftarrow f^g + f_i^{coll} + \sum_{j, (i,j) \in S} f(x_i, v_i, x_j, v_j)$ 

    forall particles i :
       $v_i + = \Delta t \cdot \frac{f_i}{m_i}$ 
       $x_i + = \Delta t \cdot v_i$ 

  render system every n-th time

```

$f^g$  = Gravitationskraft  
 $f^{coll}$  = Rückstoßkraft aus Kollision (z.B. mit Hindernis)

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 9

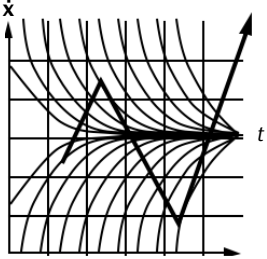
- Weitere Integrations-Schemata:
  - Midpoint
  - ...
- Vorteil: einfach zu implementieren, schnelle Ausführung pro Zeitschritt
- Nachteil: nur für sehr kleine Schrittweiten stabil
  - Typ.weise  $\Delta t \approx 10^{-4} \dots 10^{-3}$  sec!
  - Bei großen Schrittweiten wird zusätzliche Energie im System erzeugt, und schließlich explodiert es ☹
  - Beispiel: Overshooting bei einfacher Feder
- Weiterer Nachteil: Fehler akkumulieren sich schnell (bei Euler)

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 10

### Beispiel für die Instabilität des Euler-Integrationsverfahrens

- Betrachte die DGL
 
$$\dot{x}(t) = -kx(t)$$
- Die exakte Lösung:
 
$$x(t) = x_0 e^{-kt}$$
- Das Euler-Verfahren liefert:
 
$$x^{t+1} = x^t + \Delta t(-kx^t)$$
- Falls  $\Delta t > \frac{1}{k}$  :
 
$$x^{t+1} = x^t \underbrace{(1 - k\Delta t)}_{<0}$$

$\Rightarrow x^t$  oszilliert um 0,  
geht aber (hoffentlich) gegen 0
- Falls  $\Delta t > \frac{2}{k} \Rightarrow x^t \rightarrow \infty !$

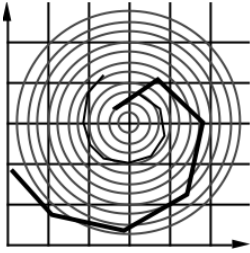


The figure shows a phase portrait in the  $(x, \dot{x})$  plane. The horizontal axis is  $x$  and the vertical axis is  $\dot{x}$ . The exact solution is a smooth curve that spirals inward towards the origin. The Euler method solution is a jagged line that oscillates around the origin and eventually diverges to infinity, illustrating instability.

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 11

### Veranschaulichung der Fehlerakkumulation


- Betrachte die DGL:
 
$$\dot{\mathbf{x}}(t) = \begin{pmatrix} -x_1 \\ x_2 \end{pmatrix}$$
- Exakte Lösung:
 
$$\mathbf{x}(t) = \begin{pmatrix} r \cos(t + \phi) \\ r \sin(t + \phi) \end{pmatrix}$$
- Euler läuft in Spiralen nach außen, egal wie klein die Schrittweite gewählt wird!



The figure shows a phase portrait in the  $(x_1, x_2)$  plane. The horizontal axis is  $x_1$  and the vertical axis is  $x_2$ . The exact solution is a smooth circle centered at the origin. The Euler method solution is a jagged line that spirals outward from the origin, illustrating error accumulation.

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 12

Veranschaulichung von DGLs

- Die allgemeine Form einer DGL (hier):
 
$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t)$$
- Visualisiere  $\mathbf{f}$  als Vektorfeld:
 
- Achtung: dieses Vektorfeld kann sich mit  $t$  ändern!
- Lösung eines Anfangswertproblems = Pfad durch dieses Feld

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 13

- Runge-Kutta 2-ter Ordnung:
  - Idee: approximiere  $\mathbf{f}(\mathbf{x}(t), t)$  durch quadratische Funktion, die an der Stelle  $\mathbf{x}(t)$ ,  $\mathbf{x}(t + \frac{1}{2}\Delta t)$  und  $\mathbf{x}(t + \Delta t)$  definiert ist
  - Der Integrator:
 
$$\mathbf{a}_1 = \mathbf{v}^t \qquad \mathbf{a}_2 = \frac{1}{m} \mathbf{f}(\mathbf{x}^t, \mathbf{v}^t)$$

$$\mathbf{b}_1 = \mathbf{v}^t + \frac{1}{2} \Delta t \mathbf{a}_2 \qquad \mathbf{b}_2 = \frac{1}{m} \mathbf{f}\left(\mathbf{x}^t + \frac{1}{2} \Delta t \mathbf{a}_1, \mathbf{v}^t + \frac{1}{2} \Delta t \mathbf{a}_2\right)$$

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \Delta t \mathbf{b}_1 \qquad \mathbf{v}^{t+1} = \mathbf{v}^t + \Delta t \mathbf{b}_2$$
- Runge-Kutta 4-ter Ordnung:
  - Der Standard-Integrator unter den expliziten Integrations-Schemata
  - Benötigt 4 Funktionsauswertungen (Kräfte berechnen) pro Zeitschritt
  - Konvergenzordnung:  $e(\Delta t) = O(\Delta t^4)$

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 14

## Verlet-Integration

- Alternative Methode zur Steigerung der Konvergenzordnung: verwende Werte aus der **Vergangenheit**
- Verlet: verwendet  $\mathbf{x}(t - \Delta t)$
- Herleitung:
  - Taylor-Reihe in beide Zeitrichtungen entwickeln:

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t \dot{\mathbf{x}}(t) + \frac{1}{2} \Delta t^2 \ddot{\mathbf{x}}(t) + \frac{1}{6} \Delta t^3 \dddot{\mathbf{x}}(t) + O(\Delta t^4)$$

$$\mathbf{x}(t - \Delta t) = \mathbf{x}(t) - \Delta t \dot{\mathbf{x}}(t) + \frac{1}{2} \Delta t^2 \ddot{\mathbf{x}}(t) - \frac{1}{6} \Delta t^3 \dddot{\mathbf{x}}(t) + O(\Delta t^4)$$

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 15

- Addieren:
 
$$\mathbf{x}(t + \Delta t) + \mathbf{x}(t - \Delta t) = 2\mathbf{x}(t) + \Delta t^2 \ddot{\mathbf{x}}(t) + O(\Delta t^4)$$

$$\mathbf{x}(t + \Delta t) = 2\mathbf{x}(t) - \mathbf{x}(t - \Delta t) + \Delta t^2 \ddot{\mathbf{x}}(t) + O(\Delta t^4)$$
- Initialisierung:
 
$$\mathbf{x}(\Delta t) = \mathbf{x}(0) + \Delta t \mathbf{v}(0) + \frac{1}{2} \Delta t^2 \left( \frac{1}{m} \mathbf{f}(\mathbf{x}(0), \mathbf{v}(0)) \right)$$

Bemerkung: Geschwindigkeit taucht nicht mehr (explizit) auf

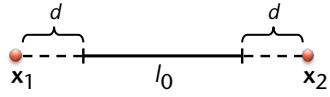
G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 16



## Constraints

- Großer Vorteil gegenüber Euler & Runge-Kutta:  
man kann sehr leicht Constraints behandeln
- Definition: **Constraint** = Einschränkung der Position eines oder mehrerer Massenpunkte
- Beispiele:
  1. Ein Massenpunkt darf nicht in ein Hindernis eindringen
  2. Der Abstand zwischen zwei Massenpunkten soll konstant sein, oder  $\leq$  bestimmter Abstand



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 17

- Verfahren anhand eines Beispiels:
  - Constraint:
 
$$\|\mathbf{x}_1 - \mathbf{x}_2\| \stackrel{!}{=} l_0$$
    1. Führe einen Verlet-Integrationsschritt durch  $\rightarrow \tilde{\mathbf{x}}^{t+1}$
    2. Enforce Constraint:
 

$$\mathbf{x}_1^{t+1} = \tilde{\mathbf{x}}_1^{t+1} + \frac{1}{2} \mathbf{r}_{12} \cdot (\|\tilde{\mathbf{x}}_2^{t+1} - \tilde{\mathbf{x}}_1^{t+1}\| - l_0)$$



$$\mathbf{x}_2^{t+1} = \tilde{\mathbf{x}}_2^{t+1} - \underbrace{\frac{1}{2} \mathbf{r}_{12} \cdot (\|\tilde{\mathbf{x}}_2^{t+1} - \tilde{\mathbf{x}}_1^{t+1}\| - l_0)}_d$$

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 18

- Problem, falls mehrere Constraints denselben Massepunkt einschränken sollen
  - Verwende Constraint-Algorithmen

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 19

## Implizite Integration

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

explizit	implizit
$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \Delta t \mathbf{v}_i^t$	$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \Delta t \mathbf{v}_i^{t+1}$
$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \Delta t \frac{1}{m} \mathbf{f}(\mathbf{x}^t)$	$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \Delta t \frac{1}{m} \mathbf{f}(\mathbf{x}^{t+1})$
- 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**

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 20




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

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 23

- Terme umordnen:
 
$$(M - \Delta t^2 K) \mathbf{v}^{t+1} = M \mathbf{v}^t + \Delta t \mathbf{f}(\mathbf{x}^t)$$
- Das ist von der Form:
 
$$\begin{aligned}
 A \mathbf{v}^{t+1} &= \mathbf{b} \\
 \text{mit } A &\in \mathbb{R}^{3n \times 3n}, \quad \mathbf{b} \in \mathbb{R}^{3n}
 \end{aligned}$$
- Löse dieses LGS mittels einer iterativen Methode
  - Denn:  $A$  ändert sich in jedem Frame

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 24

## Berechnung der Steifigkeitsmatrix

- Die Anatomie der Matrix  $K$  :
  - Eine Feder  $(i, j)$  addiert folgende vier  $3 \times 3$  Untermatrizen zu  $K$  :
 
$$\begin{array}{c}
 3i \rightarrow \\
 3j \rightarrow
 \end{array}
 \left(
 \begin{array}{cc}
 \boxed{K_{ii}} & \boxed{K_{ij}} \\
 \boxed{K_{ji}} & \boxed{K_{jj}}
 \end{array}
 \right)$$

- Die Matrix  $K_{ij}$  entsteht durch die Ableitung von  $f_i$  nach  $\mathbf{x}_j = (x_{j1}, x_{j2}, x_{j3})$ :
 
$$K_{ij} = \begin{pmatrix} \frac{\partial}{\partial x_{j1}} f_{i1} & \frac{\partial}{\partial x_{j2}} f_{i1} & \frac{\partial}{\partial x_{j3}} f_{i1} \\ \vdots & \vdots & \vdots \\ \frac{\partial}{\partial x_{j1}} f_{i3} & \cdots & \frac{\partial}{\partial x_{j3}} f_{i3} \end{pmatrix}$$
- Betrachte im folgenden nur  $f^s$  (Federkraft)

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 25

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

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 26

■ Aus einigen Symmetrien folgt:

- $K_{ij} = \frac{\partial}{\partial \mathbf{x}_j} f_i(\mathbf{x}_i, \mathbf{x}_j) = -K_{ji}$
- $K_{jj} = \frac{\partial}{\partial \mathbf{x}_j} f_j(\mathbf{x}_i, \mathbf{x}_j) = \frac{\partial}{\partial \mathbf{x}_j} (-f_i(\mathbf{x}_i, \mathbf{x}_j)) = K_{ii}$
- $K_{ji} = K_{ij}$

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 27

■ Zur Erinnerung:

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

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 28

## Lösungsverfahren

- Setze  $K = 0$
- Für jede Feder  $(i, j)$  berechne  $K_{ij}$ ,  $K_{ji}$ ,  $K_{ji}$ ,  $K_{ji}$  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  $\mathbf{x}^{t+1} = \mathbf{x}^t + \Delta t \mathbf{v}^{t+1}$

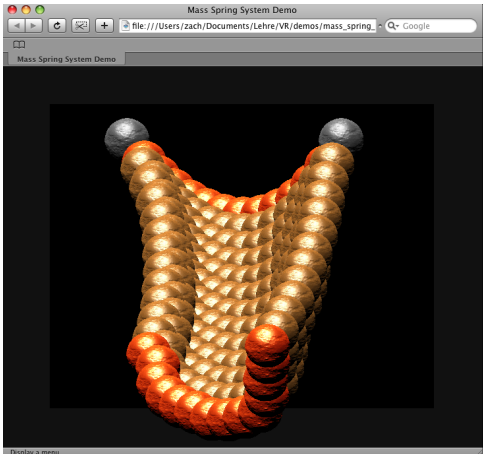
G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 29

## 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  $\rightarrow$  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

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 30

## Demo



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

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 31

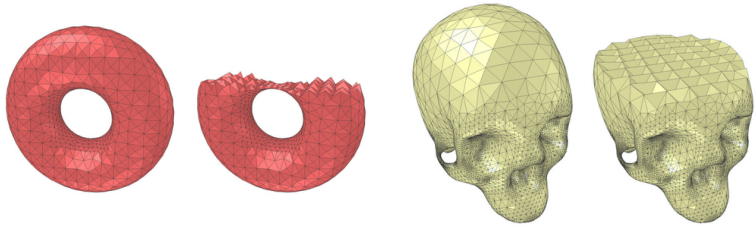
## Mesh-Erzeugung für Objekte mit Volumen

- 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  $\times$  Volumen) gleichmäßig auf die Massepunkte

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 32



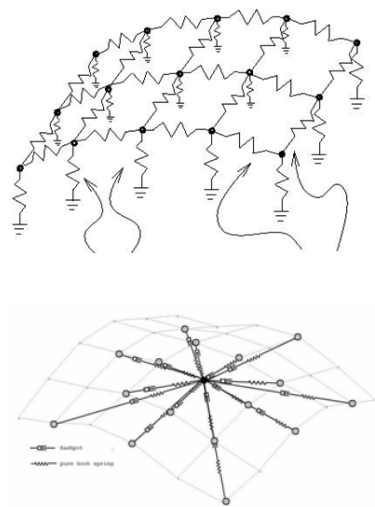
- Erzeugung des Tetraeder-Meshes (einfache Methode):
  - Verteile eine Menge von Punkten gleichmäßig (evtl. zufällig) im Inneren der Geometrie (sog. "Steiner-Punkte")
  - Dito in einer Schicht über der Oberfläche



- 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

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 33

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



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 34

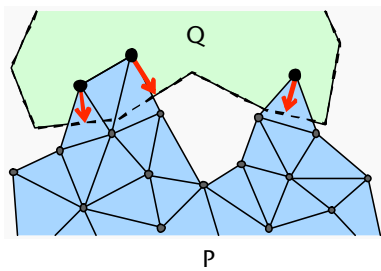
## Kollisionserkennung

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

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 35

## 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)



The diagram shows two overlapping tetrahedral meshes. The bottom mesh is labeled 'P' and is colored blue. The top mesh is labeled 'Q' and is colored green. Several vertices of mesh P are shown with red arrows pointing towards the surface of mesh Q, indicating the naive approach of calculating the distance from each vertex to the surface of the other object.

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 36

Beispiele:

inkonsistent      konsistent

G. Zachmann    Virtuelle Realität und Simulation - WS 09/10      Feder-Masse-Systeme 37

### 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  $x_j$
  - Berechne zu jedem Schnittpunkt eine Normale  $\mathbf{n}_j$ 
    - Z.B. mittels baryzentrischer Interpolation der Vertexnormalen von Q

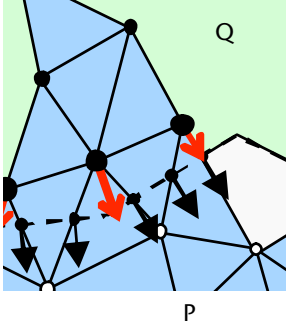
G. Zachmann    Virtuelle Realität und Simulation - WS 09/10      Feder-Masse-Systeme 38

3. Phase: berechne ungefähre Kraft für "Randpunkte"

- Randpunkt = eingedrungener Punkt  $\mathbf{p}$  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  $\mathbf{p}$ ,  $\mathbf{x}_i$  = Schnittpunkt der Schnittkante inzident zu  $\mathbf{p}$ ,  $\mathbf{n}_i$  = Normale zur Oberfläche von Q im Schnittpunkt  $\mathbf{x}_i$ ,  
und  $\omega(\mathbf{x}_i, \mathbf{p}) = \frac{1}{\|\mathbf{x}_i - \mathbf{p}\|}$



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 39

- Richtung der Kraft für Randpunkte:

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

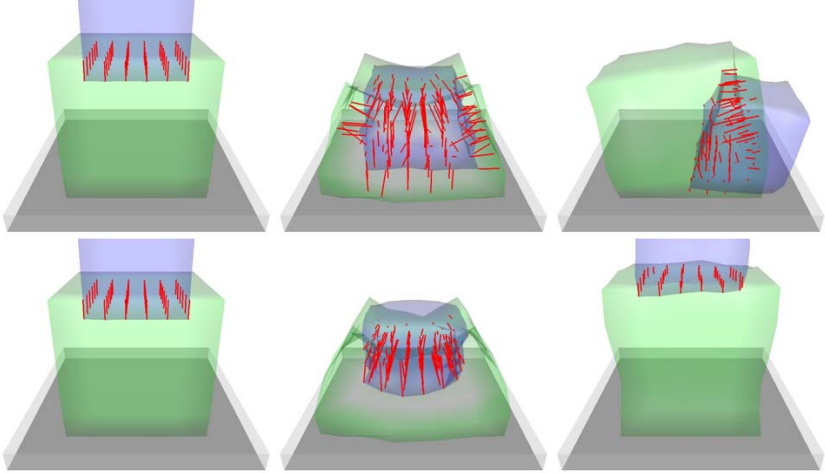
4. Phase: Propagation mittels breadth-first traversal durch das Tetraeder-Mesh

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

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  $\mathbf{p}_i$ .

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 40

### Visualisierung



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 41

### Video

---

Consistent Penetration  
Depth Estimation  
for Deformable  
Collision Response

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

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Feder-Masse-Systeme 42