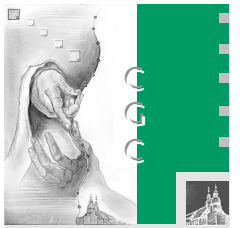
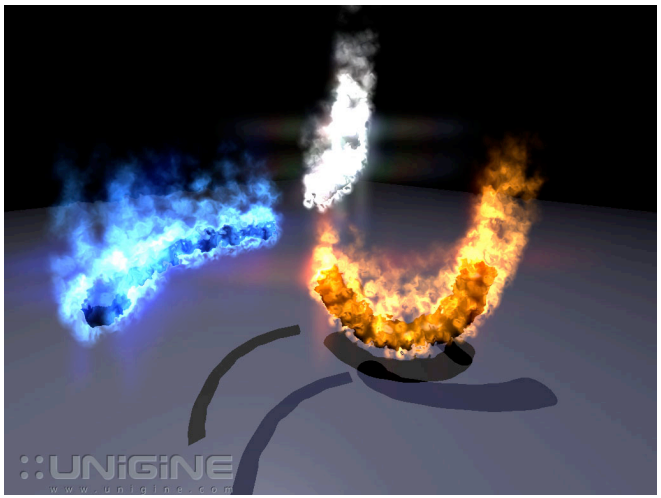


Virtuelle Realität
Partikelsysteme

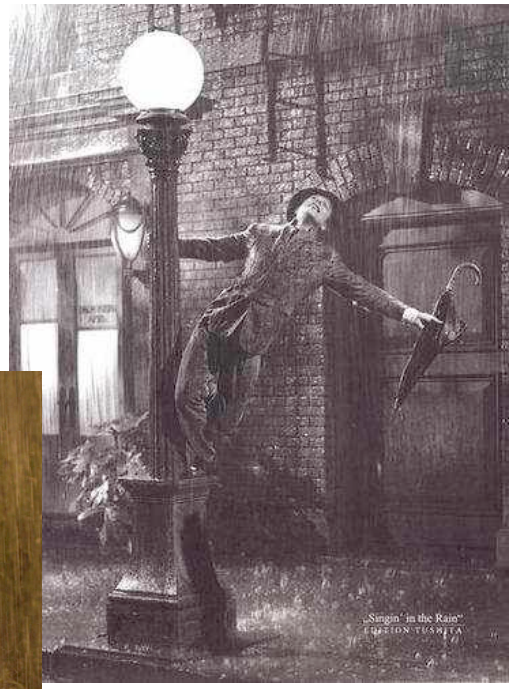


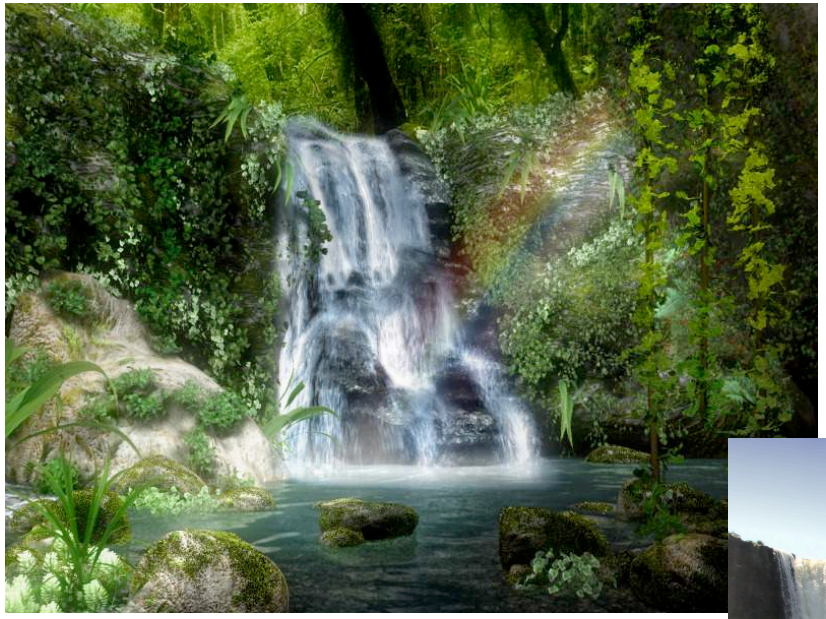
G. Zachmann

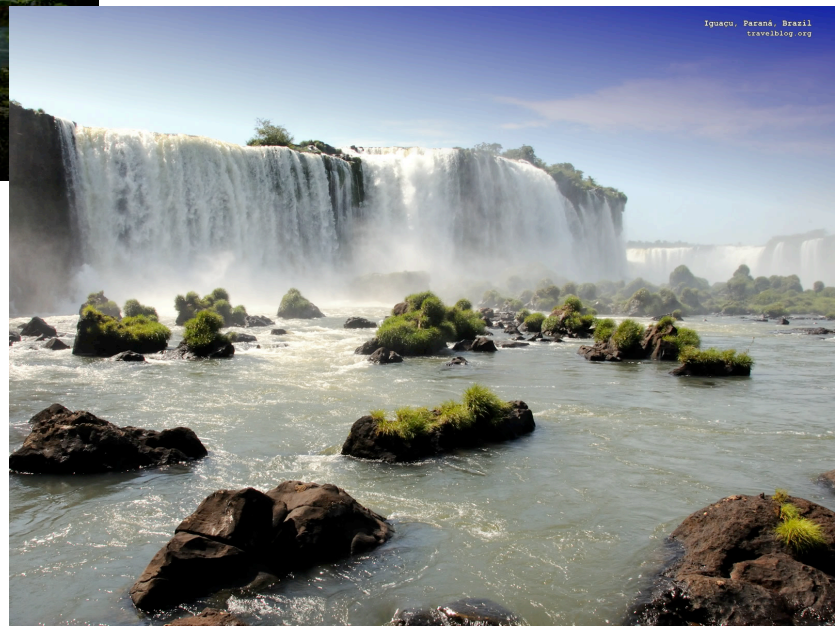
Clausthal University, Germany

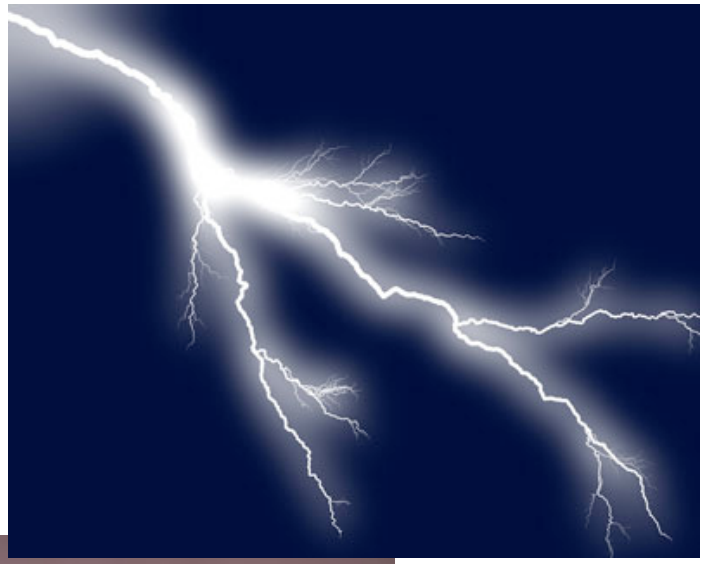
cg.in.tu-clausthal.de

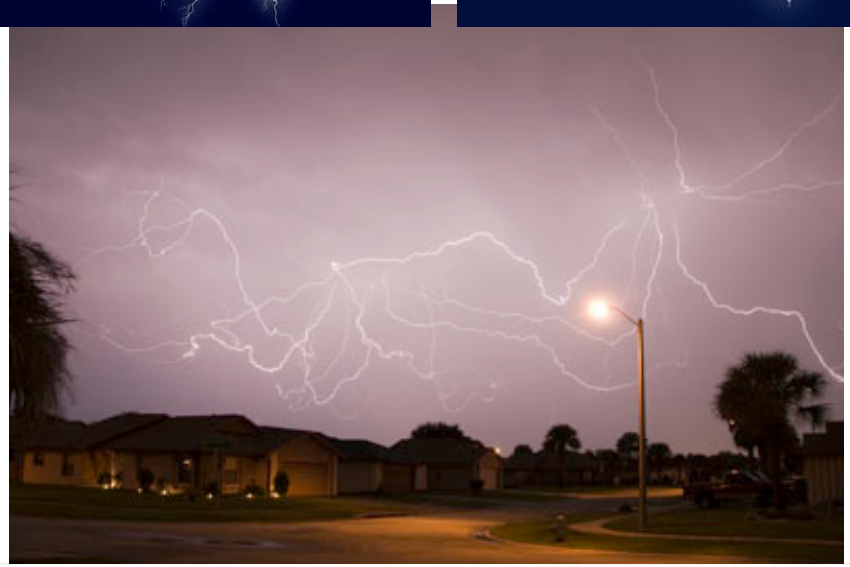
Modellierung/Simulation/Rendering natürlicher Phänomene






Iguaçu, Paraná, Brazil
travelblog.org



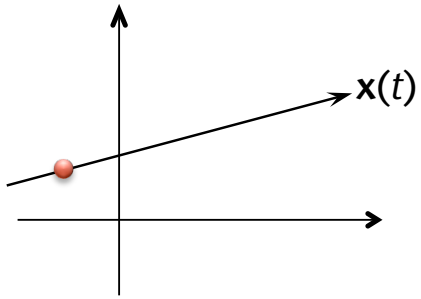


Dynamik eines Massenpunktes

- Definition Partikel:

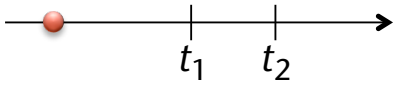
  Ein Partikel ist ein ideeller Punkt mit einer Masse m und einer Geschwindigkeit $\mathbf{v}$.

  →Die Orientierung ist irrelevant



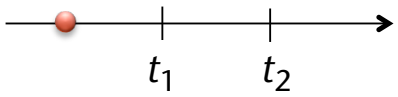
- Bahn eines Partikels: $\mathbf{x}(t)$
- Geschwindigkeit:

$$\mathbf{v} = \frac{\text{Weg}}{\text{Zeit}} = \frac{\mathbf{x}(t_2) - \mathbf{x}(t_1)}{t_2 - t_1}$$



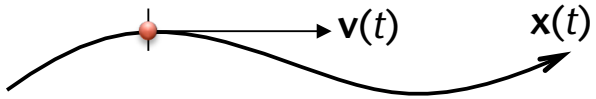
  - Einheit: m/s
  - Beachte: Geschwindigkeit = Vektor — Ort des Partikels = Punkt!

- Momentangeschwindigkeit:



$$\mathbf{v}(t_1) = \lim_{t_2 \to t_1} \frac{x(t_2) - x(t_1)}{t_2 - t_1}$$

$$= \frac{d}{dt} x(t_1) = \dot{\mathbf{x}}(t_1)$$



- Beispiele:
  - Punkt bewegt sich auf Kreisbahn: $|\dot{\mathbf{x}}|$ ist konstant
  - Punkt beschleunigt auf Gerade: $\frac{\dot{\mathbf{x}}}{|\dot{\mathbf{x}}|}$ ist konstant
- Beschleunigung :

$$\mathbf{a}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{v}(t) = \dot{\mathbf{v}}(t) = \frac{\mathbf{F}(t)}{m}$$

Newtons 2. Gesetz

Euler-Integration

- Gegeben: ein Partikel der Masse m; eine Kraft $\mathbf{F}(t)$, die auf das Partikel über die Zeit wirkt
- Gesucht: die Bahn $\mathbf{x}(t)$ des Partikels
- Analytischer Ansatz:

$$\mathbf{v}(t) = \mathbf{v}_0 + \int_{t_0}^{t} \mathbf{a}(t)\,\mathrm{d}t$$

$$\mathbf{x}_0(t) = \mathbf{x}_0 + \int_{t_0}^{t} \mathbf{v}(t)\,\mathrm{d}t$$

- Diskretisieren und Linearisieren:

$$v^{t+1} = v^t + a^t \cdot \Delta t$$

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

oder

$$x^{t+1} = x^t + \frac{v^t + v^{t+1}}{2}\Delta t \qquad \text{(approx. midpoint method)}$$

Der Phasenraum

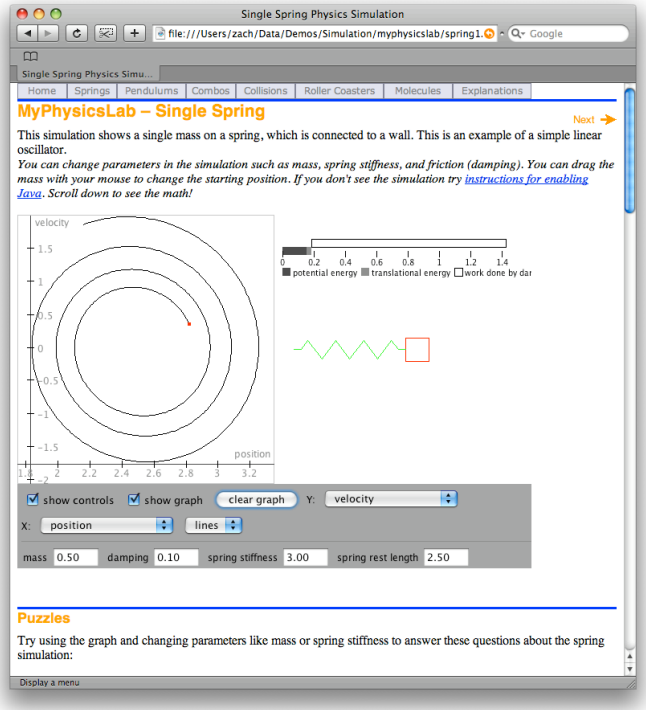
- Der (physikalische) momentane Zustand eines Partikels ist vollständig beschrieben durch

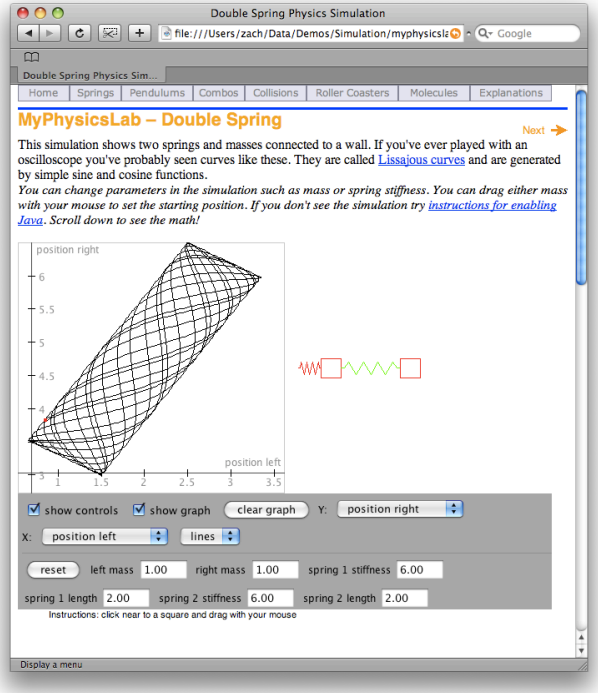
$$\mathbf{q} = (\mathbf{x}, \mathbf{v}) = (x_1, x_2, x_3, v_1, v_2, v_3)$$
$$= (x_1, x_2, x_3, \dot{x}_1, \dot{x}_2, \dot{x}_3) \in \mathbb{R}^6$$

- Der Raum aller möglischer Zustände heißt Phasenraum (*phase space*)
- Die Dimension ist $6n$, n = Anzahl Partikel
- Bewegungsgleichungen im Phasenraum:

$$\dot{\mathbf{q}} = (\dot{x}_1, \dot{x}_2, \dot{x}_3, \dot{v}_1, \dot{v}_2, \dot{v}_3) = \left(v_1, v_2, v_3, \frac{\mathbf{F}_1}{m}, \frac{\mathbf{F}_2}{m}, \frac{\mathbf{F}_3}{m} \right)$$

- Beispiel für ein Partikel, das sich nur auf der x-Achse bewegen kann und durch eine Feder in einer Ruhelage gehalten wird:





www.myphysicslab.com

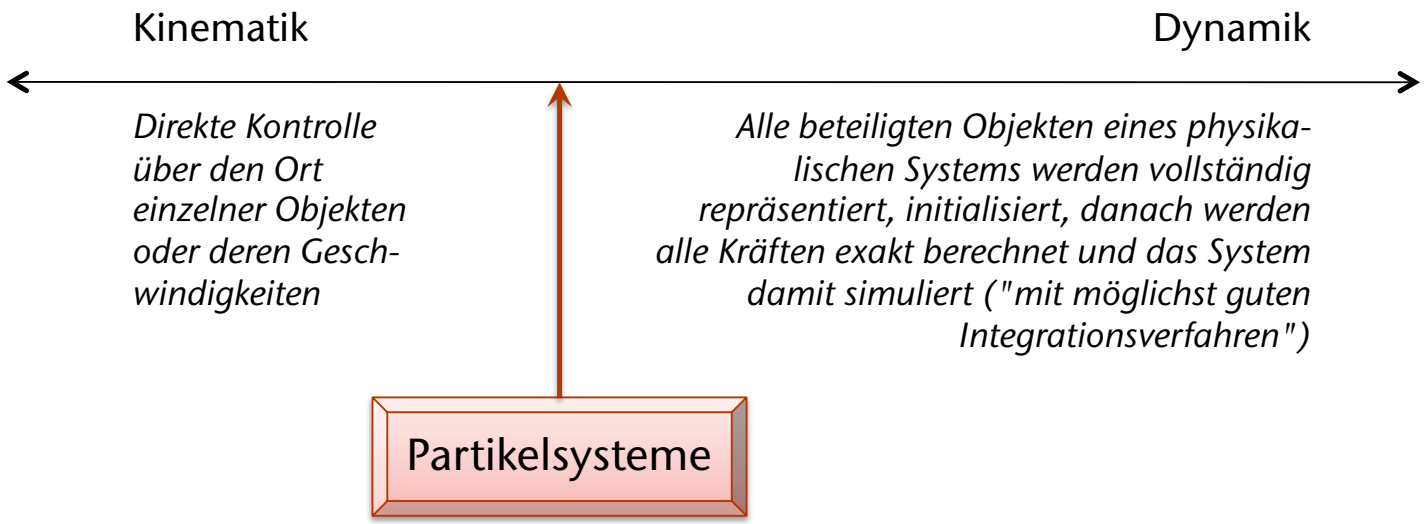
Kinematik versus Dynamik

- Begriffe:

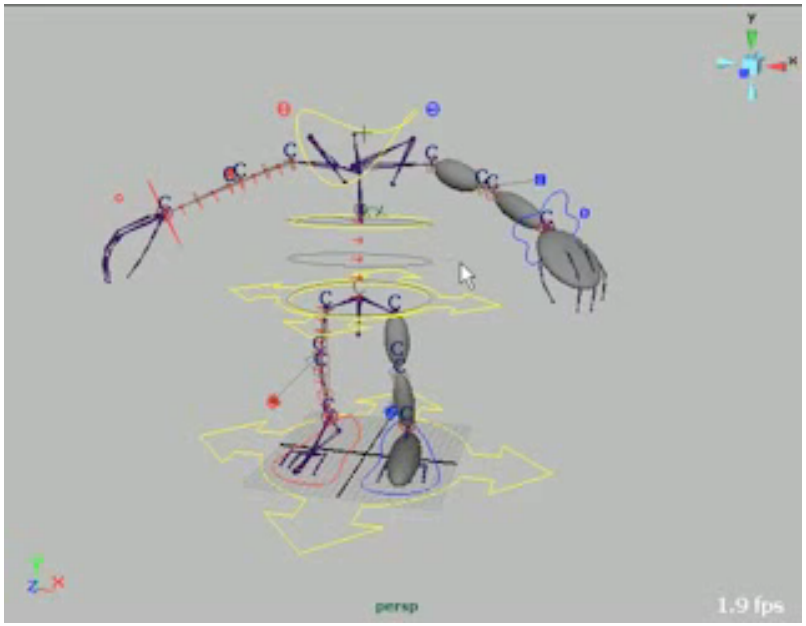
  Kinematik = Bewegung von Körpern ohne Simulation von Kräften

  Dynamik = Simulation / Berechnung von Kräften und die daraus resultierende Bewegungen der Objekte

- In der Computergraphik bewegt man sich in einem Kontinuum:



- Beispiel für reine Kinematik: inverse Kinematik



Partikelsysteme

- Definition: Ein Partikelsystem besteht aus
  1. Einer Menge von Partikeln; jedes Partikel i hat (mindestens) folgende Attribute:
     - Masse m_i
     - Position $\mathbf{x}_i$
     - Geschwindigkeit $\mathbf{v}_i$
     - Alter a_i
     - Kräfteakkumulator $\mathbf{F}_i$
     - und eventuell weitere, wie z.B. Farbe, Transparez, Optische Größe, Lebensdauer, Typ/Art ... einer Menge
  2. Einer Menge Partikelquellen; jeder ist beschrieben durch
     - Form der Partikelquelle
     - Stochastische Prozesse, die die initialen Attribute der Partikel festlegen (Geschwindigkeit, Richtung, etc.)
     - Stochastische Prozesse, die die Anzahl der erzeugten Partikel pro Frame festlegen
  3. Weitere (globale) Parameter, z.B.
     - TTL (time to live) = max. Lebensdauer eines Partikels
     - Globale Kräfte (z.B. Gravitation, Wind, ...)
     - die Algorithmen, die die Partikel bewegen und rendern

- Stochastischer Prozeß =
  - Im einfachsten Fall Mittelwert + Varianz; Prozeß liefert zufälligen Wert gemäß Gleichverteilung
  - Etwas komplizierter: Mittelwert und Varianz sind Funktionen der Zeit

- Form der Partikelquelle:
  - Ist intutive Art, den stochastischen Prozess für die initiale Position von Partikeln zu beschreiben
  - Häufig: Kreisscheibe, Würfel, Kegel, etc.

Das Execution Model

- Der Ablauf eines Partikelsystems:

```
loop forever:
  rendere alle Partikel
  Δt := Rendering-Zeit
  kille alle Partikel mit Alter > Lebensdauer
  erzeuge neue Partikel an der Quelle
  lösche alle Kräfteakkumulatoren
  berechne alle Kräfte auf jedes Partikel (akkumuliere diese)
  aktualisiere Geschwindigkeit (ein Eulerschritt mit Δt)
  modifiziere gegebenfalls Geschwindigkeiten (*)
  aktualisiere Positionen (ein weiterer Eulerschritt)
  modifiziere eventuell Positionen (z.B. wg. Constraints)
  sortiere Partikel nach Tiefe (für Alpha-Rendering)
```

Bemerkungen

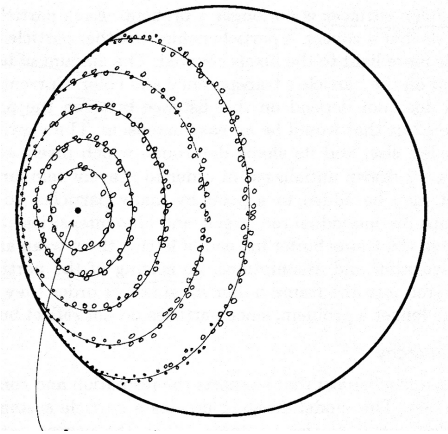
- Hier gibt es viel Raum für Optimierungen, z. B.
  - Gravitationskraft gleich beim Löschen des F-Akkumulators setzen
  - Nicht bei jedem Partikel händisch das Alter inkrementieren sondern Zeit t_{gen} der Entstehung speichern, dann nur noch $t_{current} - t_{gen} >$ TTL testen
    - Wird später bei paralleler Implementierung wichtig
- Zu (*) im Algorithmus:
  - Ist "un-physikalisch", erlaubt aber bessere kinematische Kontrolle durch den Programmierer / Animateur
  - Ist auch bei Kollisionen nötig
- Der Rest ist Intuition und Kreativität ...
- Oft speichert man eine kleine Historie der Positionen der Partikel, um einen einfachen "motation blur"-Effekt zu erhalten
- Partikel können auch auf Grund anderer Bedingungen gekillt werden, z.B. Entfernung von der Quelle, Eintritt in einer bestimmter Region, etc.
- Achtung, für eine effiziente Implementierung kann eine "Struct-of-Array"-Datenstruktur besser sein! (SoA statt AoS)

Beispiel eines Partikelsystems

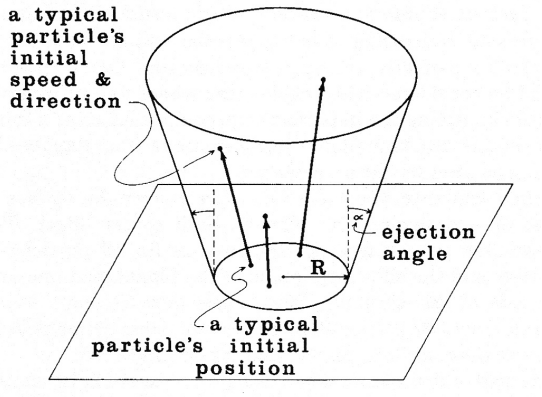
- Ausschnitt aus "Wrath of Khan":

(William Reeves, 1984)

- Partikelquelle = Kreis auf der Kugel um den *impact point*, der sich vergrößert

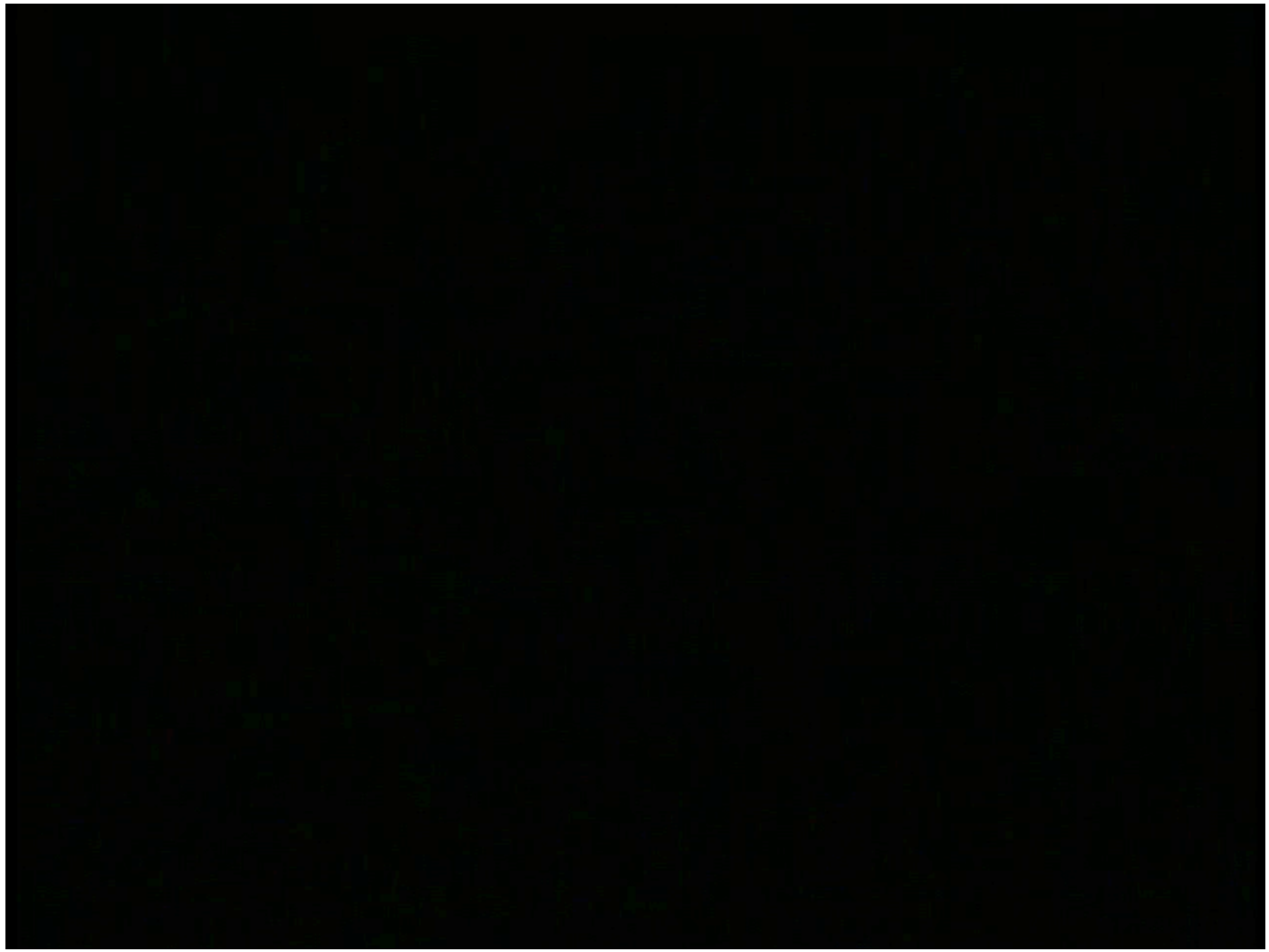


- Stochastische Prozesse für Partikel-generierung:
  - Kegelstumpf senkrecht zu Kugeloberfläche
  - Varianz für Lebendauer



- Farbe = *f*(*Alter*)

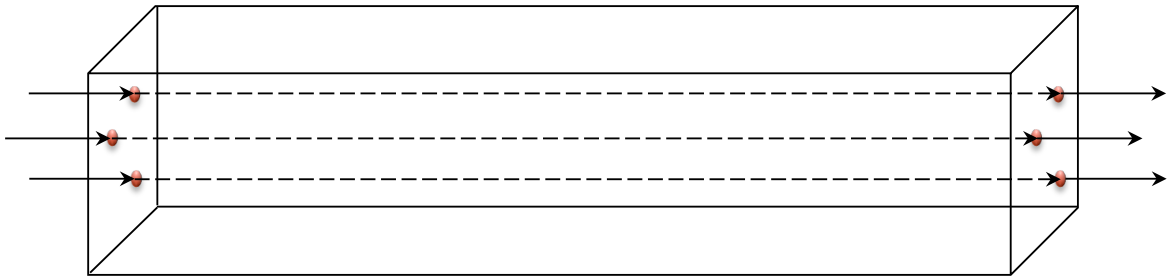
Exkurs: die Panspermie-Hypothese



Karl Sims, 1990

Operation auf Partikeln

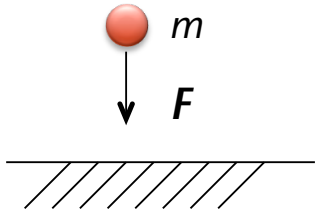
- Positionsoperationen:
  - Eher selten
  - z.B. “Tunneln”



Physikalische Effekte

- Schwerkraft:

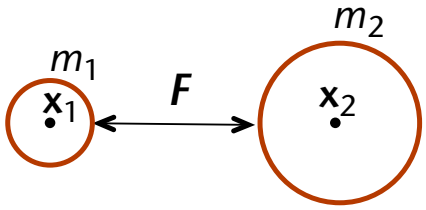
$$\mathbf{F} = m \cdot \mathbf{g} \quad , \qquad g = 9.81 \frac{\mathrm{m}}{\mathrm{s}^2}$$



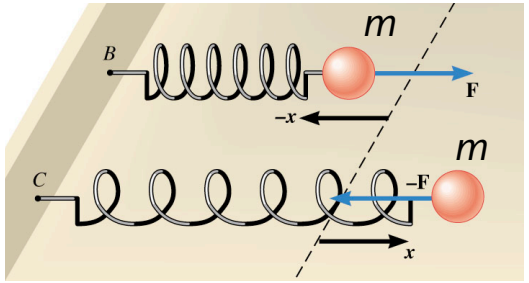
- Gravitation:

$$\mathbf{F} = G \cdot \frac{m_1 m_2}{r^2} \cdot \frac{\mathbf{x}_1 - \mathbf{x}_2}{|\mathbf{x}_1 - \mathbf{x}_2|}$$

$$G = 6,67 \cdot 10^{-11}$$



- Federkraft: später



- Viskose Hemmung/Dämpfung (viscous drag):

$$\mathbf{F} = -b\,\mathbf{v}$$

in einem ruhenden Fluid/Gas;

oder auch

$$\mathbf{F} = 6\pi\eta r(\mathbf{v} - \mathbf{v}_{fl})$$

bei Fluid mit Geschwindigkeit $\mathbf{v}_{fl}$, Partikel mit Radius r, Viskosität η;

oder auch

$$\mathbf{F} = -\frac{1}{2}c\rho A\mathbf{v}^2$$

bei hoher Geschwindigkeit; ρ = Dichte, A = Querschnittfläche des Körpers, c = Viskositätskonstante

- Elektromagnetische Kraft (Lorentz-Kraft):

$$\mathbf{F} = q \cdot \mathbf{v} \times \mathbf{B}$$

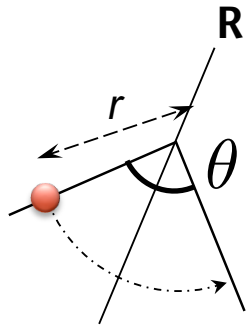
wobei q die Ladung des Partikels , $\mathbf{v}$ dessen Geschwindigkeit, und **B** das magnetische Feld ist.

https://elearning.mat.univie.ac.at/physikwiki/index.php/LV002:LV-Uebersicht/Videos/Lorentzkraft_1

Nicht-physikalische Effekte

- Strudel (*vortex*): rotiere Position eines Partikels um Achse **R** und Winkel

$$\theta = a \cdot f(r)$$

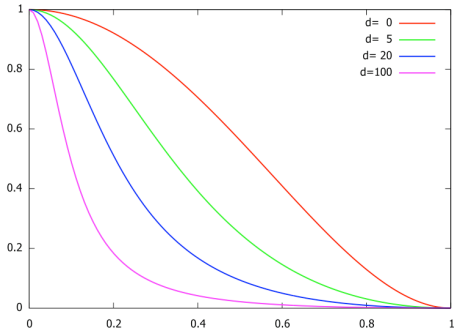


wobei a = "Stärke" des Vortex,
r = Abstand Partikel – Achse, und

$$f(r) = \frac{1}{r^{\alpha}}$$

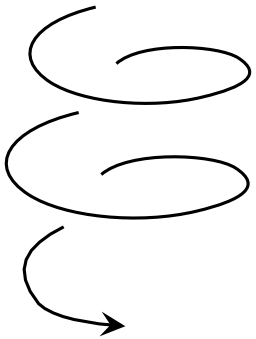
oder

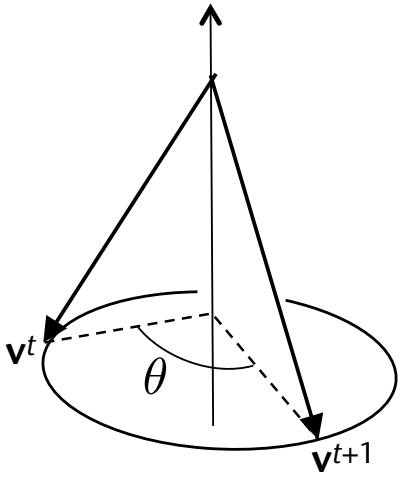
$$f(r) = \begin{cases} \frac{r^4 - 2r^2 + 1}{1 + dr^2} & , \; r \leq 1 \\ 0 & , \; r > 1 \end{cases}$$



- Erweiterungen:
  - Masse des Partikels einbeziehen
  - B-Spline als Achse des Vortex (für Tornados z.B.)
  - Achse des Vortex animieren

- Winkel:
  - Oftmals bewegt sich jedes einzelne Partikel auf einer spiralförmigen Bahn (z.B. in Feuer, oder Schneeflocken)
  - Idee:
    Rotiere $\mathbf{v}$ um eine Achse mit Winkel
    $$\theta = \sigma \cdot \Delta t$$
  - σ kann wieder leicht zufällig variieren, ebenso die Achse
  - Die Achse und σ können über die Zeit animiert werden

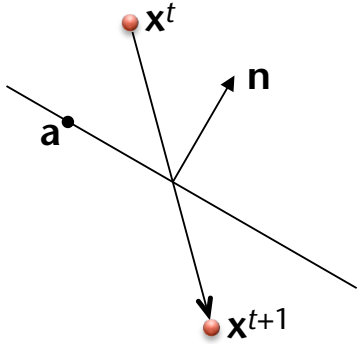




Kollisionen

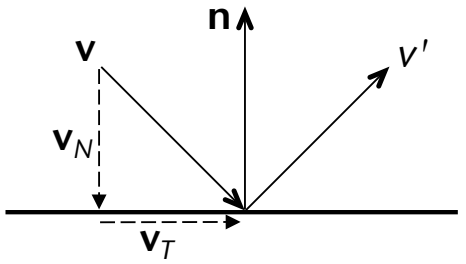
- Die wichtigste Form von geometrischen Constraints
- Zunächst: Kollision mit einer Ebene
- Test:

$$(\mathbf{x}^t - a)\,\mathbf{n} > 0 \;\wedge\; \left(\mathbf{x}^{t+1} - \mathbf{a}\right)\mathbf{n} < 0$$



- Koll.-Behandlung: reflektiere $\mathbf{v}$

$$\mathbf{v}_N = (\mathbf{v}\cdot\mathbf{n})\,\mathbf{n}$$
$$\mathbf{v}_T = \mathbf{v} - \mathbf{v}_N$$
$$\mathbf{v}' = \mathbf{v}_T - \mathbf{v}_N = \mathbf{v} - 2(\mathbf{v}\cdot\mathbf{n})\,\mathbf{n}$$



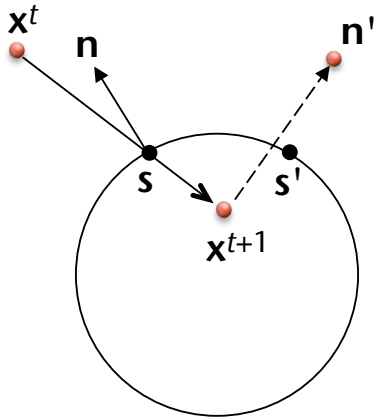
- Erweiterung um Reibung und elastischer/inelastischer Stoß:

$$\mathbf{v}' = (1-\mu)\,\mathbf{v}_T - \varepsilon\mathbf{v}_N$$

mit μ = Reibungskoeffizient (friction parameter) und

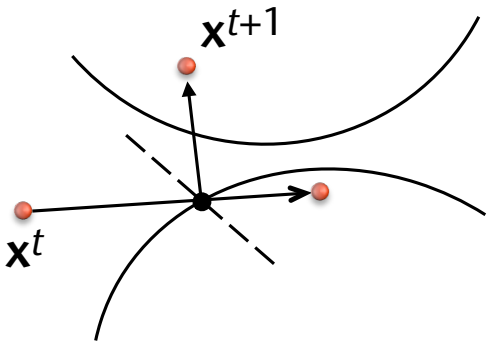
ε = Federung / Elastizität (resilience)

- Fazit: Kollisionserkennung für Partikel = "Punkt-in-Geometrie-Test" bzw. Schnitttest zwischen Geradensegment und Geometrie
- Analog für Kugeln:
  - Exakten Schnittpunkt **s** und Normale **n** bestimmen
  - Dann weiter wie eben

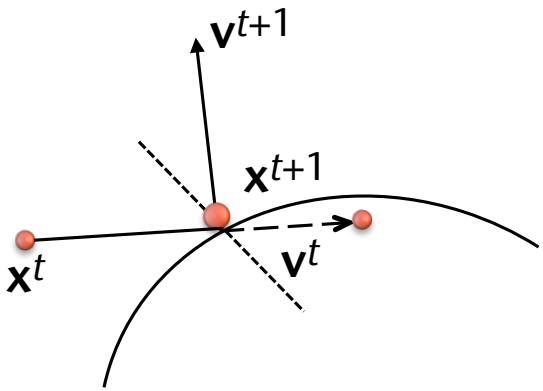


- Für Polyeder und implizite Flächen: siehe CG1
- Für Höhenfeld (Terrain): siehe CG2 (Raytracing)

- Achtung: stelle konsistenten Zustand nach der Kollisionsbehandlung her!
  - Problem: "Doppelkollisionen" an engen Stellen
  - Beispiel:



  - Korrekte Behandlung:



  - Gibt noch weitere Möglichkeiten

Hierarchische Partikelsysteme

- Idee:
  - Ein Partikel ist seinerseits wieder ein Partikelsystem
  - Transformation des Vater-”Partikels” wirkt sich auf dessen Kind-Partikel aus (analog zu Scenengraph)

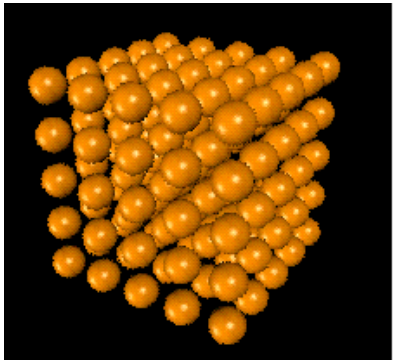
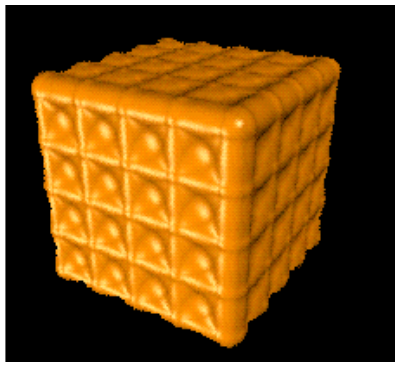
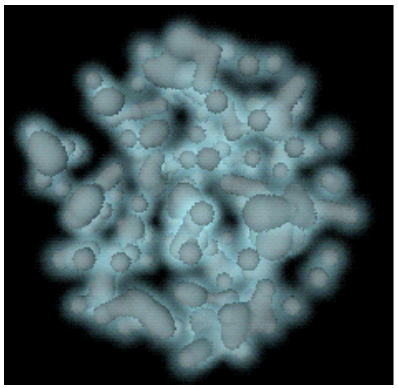
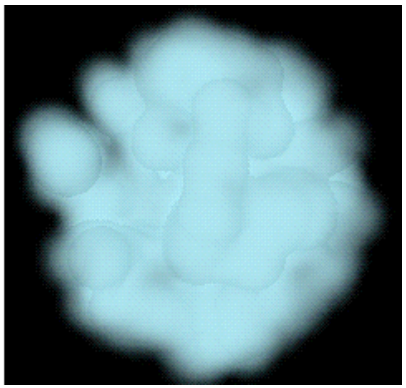
- Second-Order-Partikelsysteme:
  - Auch alle Kräfte werden durch Partikel repräsentiert
  - Diese können wechselwirken, werden an Partikelquellen geboren, sterben, etc.

Rendering

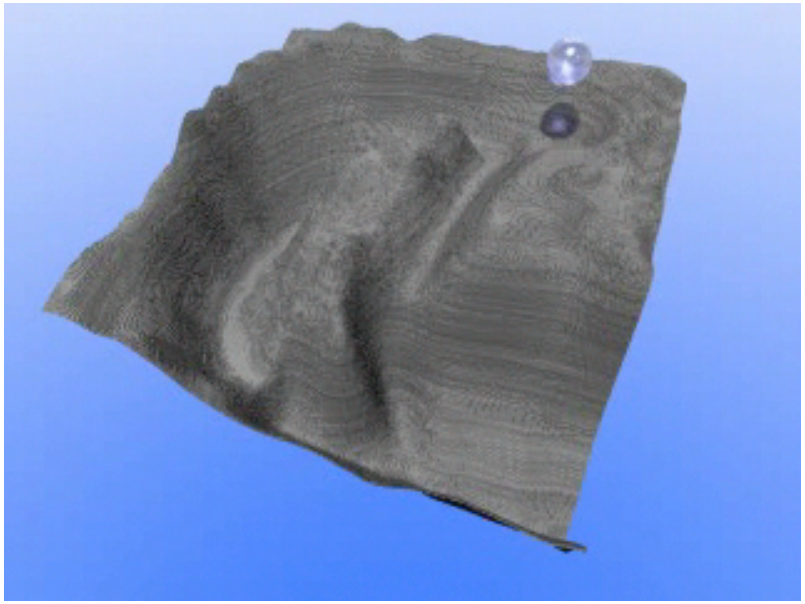
- Es gibt kein Standardverfahren
- Häufig:
  - Partikel als kleine Kreisscheibe (Splat, Sprite, Billboard) rendern
  - Meist mit Transparenz, die zum Rand abfällt
  - Benötigt Alpha-Blending!
- Alternative:
  - Farbe aller Partikel im Framebuffer akkumulieren (z.B. für Feuer)
  - Benötigt ca. 10 Partikel/Pixel

Rendering von "blobby objects"

- Betrachte Partikel als Metaballs
  - Zur Erinnerung (aus CG 1): Metaballs = spheres that blend together to form (implicit) surfaces
  - Rendering mittels Ray-Casting
  - Entweder: Nullstelle der impliziten Fläche suchen
  - Oder: "Dichte" entlang des gesamten Strahls aufsummieren und als Opazität oder Leuchtdichte interpretieren

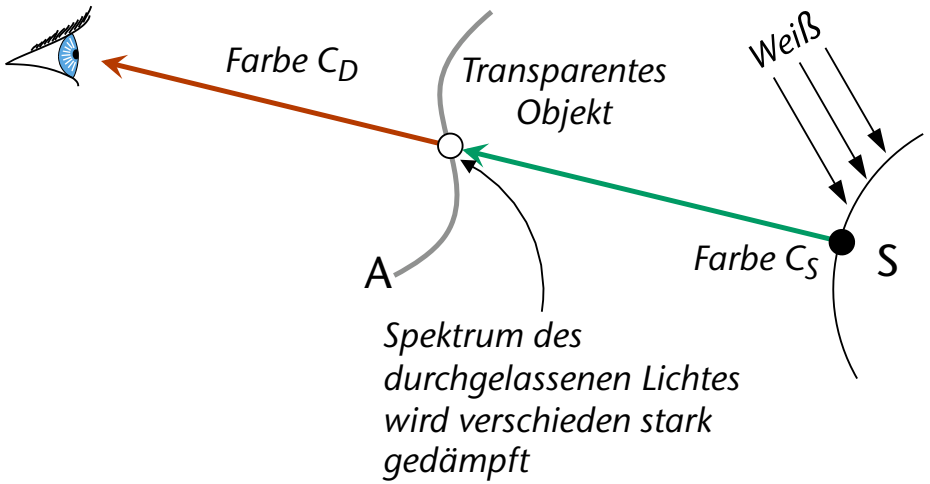





Beispiel



Rendering von transparenten Objekten

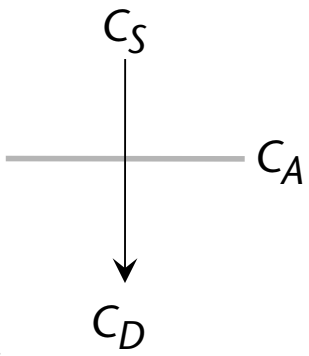
- Transparenz ≈ Licht wird von einem Material teilweise durchgelassen, wobei verschiedene Wellenlängen verschieden stark gedämpft werden



  - Extremfall: Farbfilter

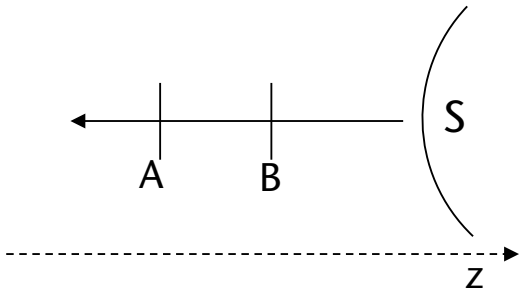
- Approximation: Alpha-Blending
  - $\alpha \in [0, 1]$ = Transparenz / Opacity
    - $\alpha = 0$ → komplett durchsichtig,
      $\alpha = 1$ → komplett opak (opaque)
  - Objekt A bekommt eine transparente "Farbe" C_A
  - Resultat:

$$C_D = \alpha C_A + (1 - \alpha) C_S$$



  - α wird als 4-te Komponente in Farbvektoren gespeichert
- Beim Rendern führt die Graphikkarte folgende Operationen aus:
  - Color aus Framebuffer lesen → C_S
  - Formel auswerten → C_D
  - C_D in Framebuffer schreiben

- Achtung bei mehreren transparenten Objekten hintereinander!
  - Erst *A*, dann *B* → *B* wird durch z-Test gekillt
  - Naive Idee: Z-Buffer abschalten
  - Erst *A* dann *B* (ohne z-Test) ergibt:



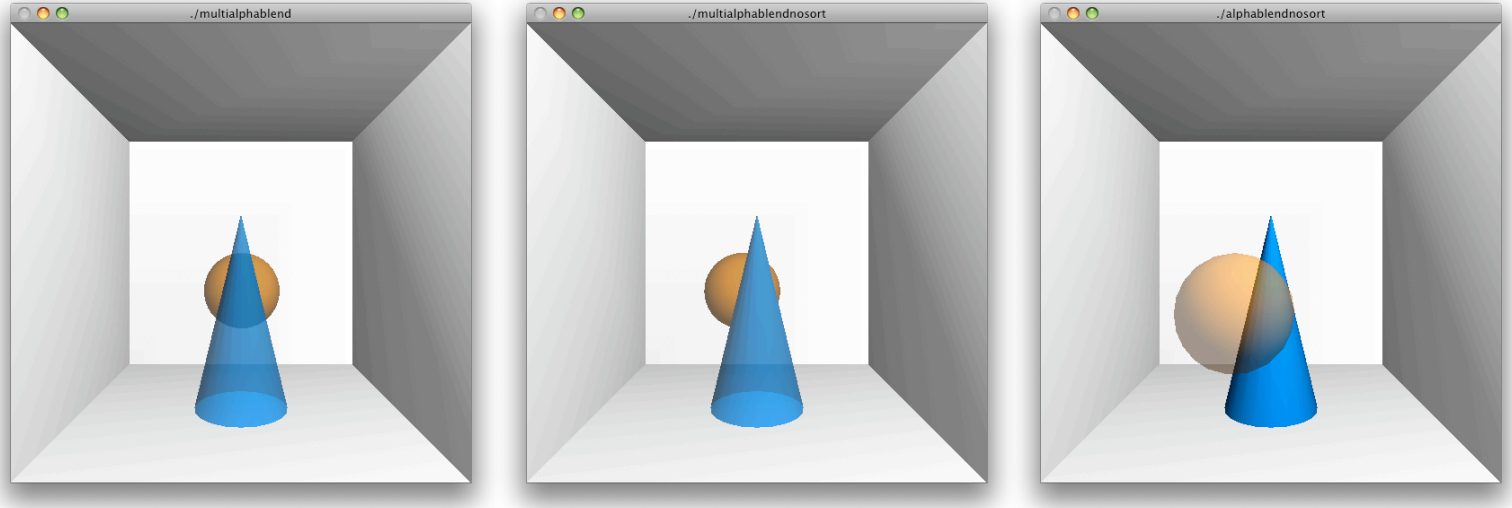
$$C'_D = \alpha_A C_A + (1 - \alpha_A)\, C_S$$
$$C_D = \alpha_B C_B + (1 - \alpha_B)\, C'_D$$
$$= \alpha_B C_B + (1 - \alpha_B)\, \alpha_A C_A + (1 - \alpha_B)\,(1 - \alpha_A)\, C_S$$

  - Erst *B*, dann *A* (ohne z-Test) ergibt:

$$C'_D = \alpha_B C_B + (1 - \alpha_B)\, C_S$$
$$C_D = (1 - \alpha_A)\, \alpha_B C_B + \alpha_A C_A + (1 - \alpha_B)\,(1 - \alpha_A)\, C_S$$

- Fazit: man muss die Polygone/Partikel von hinten nach vorne rendern, selbst dann, wenn der Z-Buffer abgeschaltet wird!

- Beispiele:




```
% cd VR/demos/alphablending; ./multialphablend; ./multialphablendnosort; ./alphanosortblend
```

- In Open GL:
  - Einschalten mit:

```
glEnable( GL_BLEND );
```

  - Blending-Funktion festlegen:

```
glBlendFrame( Glenum s, Glenum d );
```

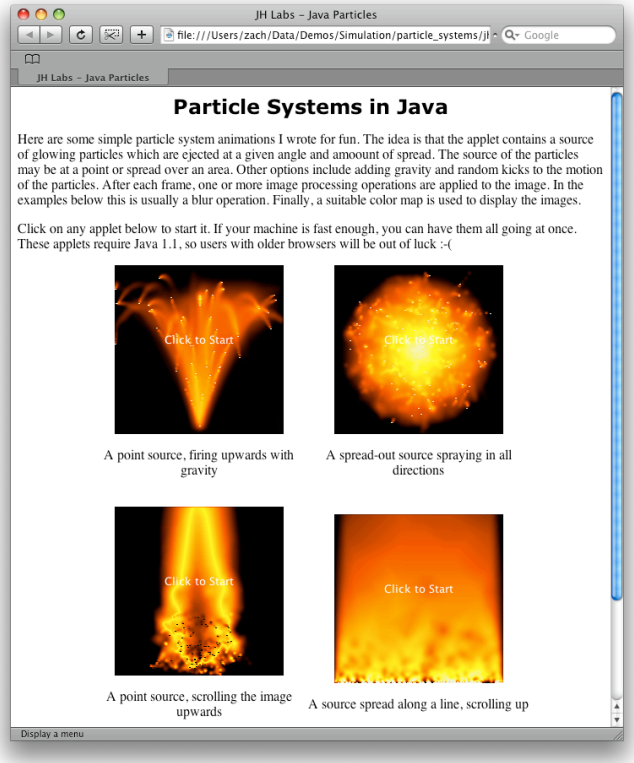
GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA →

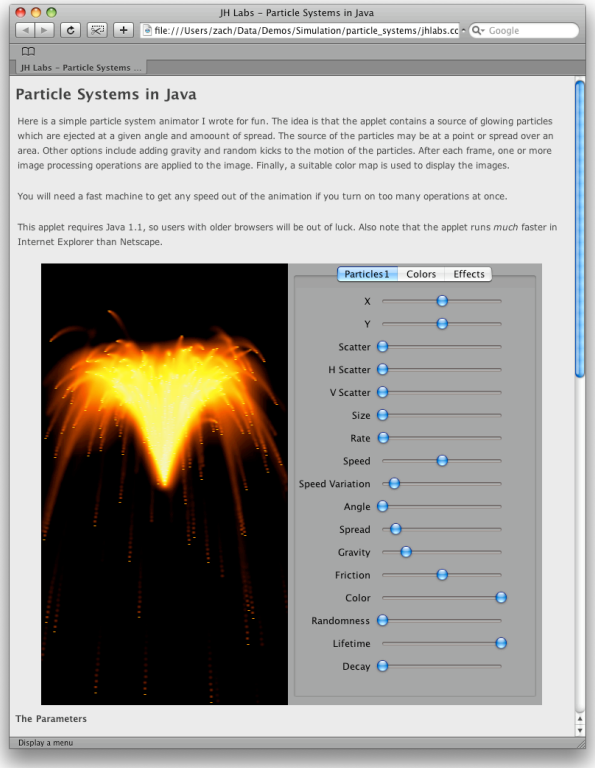
$$C'_D = \alpha C_A + (1 - \alpha)\, C_B$$

wobei C_D die Farbe aus dem Framebuffer ist;

  - Es gibt noch viele andere Varianten, z.B. kann man damit die Farben auch einfach aufakkumulieren (**GL_ONE, GL_ONE**)

Partikel-Systeme-Demos





http://www.jhlabs.com/java/particles.html

Flammen

[2002]

- Ziel:
  1. Glaubhaft aussehende Flammen
  2. Möglichst volle Kontrolle über die Flammen
- Das Modell:
  1. Einzelne Flammen (-elemente) durch parametrische Kurven modellieren → “spine” der Flamme
  2. Kontrollpunkte als Partikel simulieren
  3. Zylindrisches Profil um den Spine ergibt Oberfläche der Flamme (wo Oxidation = Verbrennen stattfindet)
  4. Der Raum in der Nähe der Oberfläche wird mit Partikeln gesampelt
  5. Rendern der Partikel (entweder volumetrisch, oder mit Alpha-Blending)
- Kontrollelemente:
  - Länge der Spines
  - Lebensdauer der Partikel
  - Intensität (=Anzahl Partikel), Quellen, Richtung, Wind, etc
  - Farbe und Größe der Partikel

- Generierung des *Spines*:
  - Spine-Partikel P im ersten Frame generieren
  - Dieses aufsteigen lassen (Auftrieb) und durch Wind bewegen:

$$\mathbf{v}_P^{t+1} = \mathbf{v}_P^t + w\left(\mathbf{x}_P, t\right) + b\left(T_P\right) + d\left(T_P\right)$$

wobei

w = Windfeld

b = Auftrieb

d = Diffusion = Rauschen;

T_P = Temperatur des Partikels = Alter

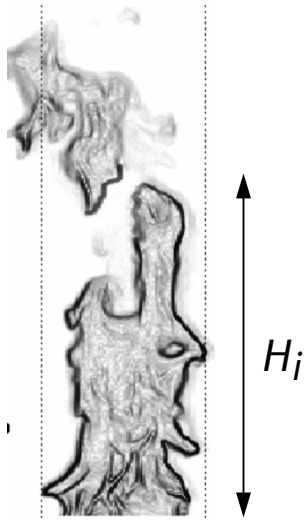
(Vereinfachung hier: Partikel haben keine Masse)

  - In Folge-Frames weitere solche Partikel generieren, bis Max.-Anzahl für ein Spine erreicht
  - Spine-Partikel durch B-Spline verbinden

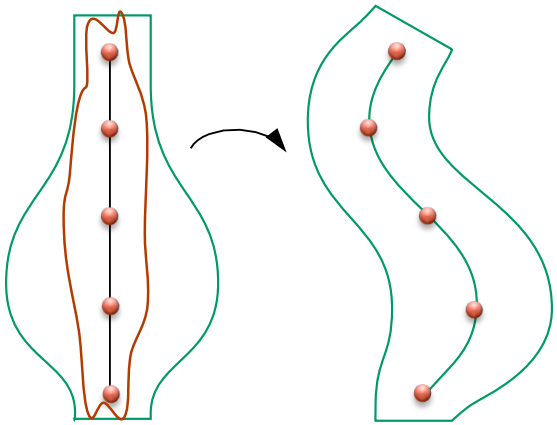
- Aufbrechen von Flammen-Elementen:
  - Das obere Stück des Spines wird zu einem zufälligen Zeitpunkt abgetrennt, wenn Höhe > H_i
  - Lebensdauer nach der Abtrennung:

$$A \cdot \alpha^3 \ , \quad \alpha \in [0,1] \text{ zufällig}$$

$$A = 0.1 \ldots 2 \text{ sec}$$



- Das Profil der Flamme:
  - Rotationssymmetrisch um den Spine herum



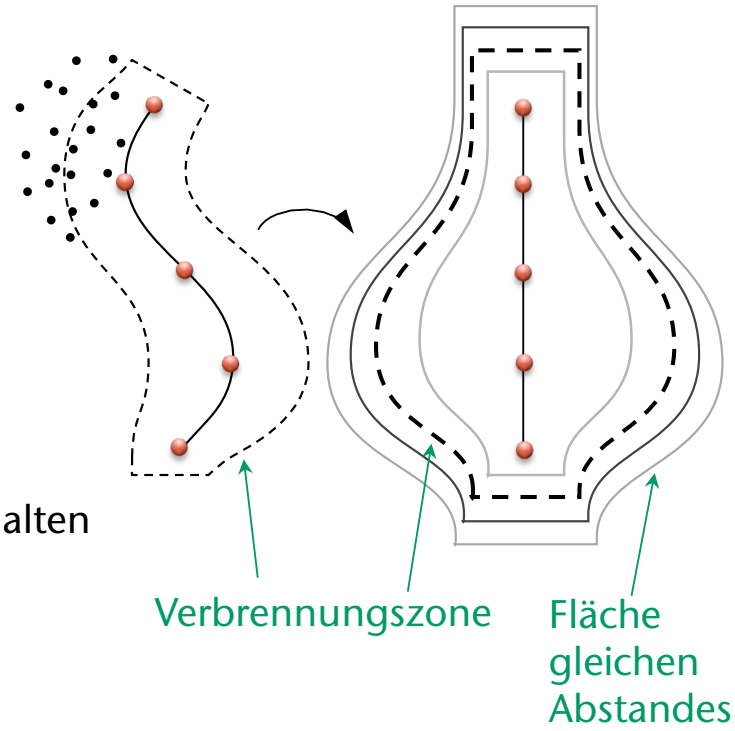
- Rendering:
  - Raum um die Flamme durch eine große Menge von Punkten sampeln gemäß der Dichtefunktion

$$D(\mathbf{x}) = \frac{1}{1 + ||\mathbf{x} - \mathbf{x}'||}$$

wobei **x**' der Punkt auf der (deformierten) Profilfläche ist, der zu **x** am nächsten ist:

- Zufälliges **x** erzeugen
- Transformieren in Modellraum
- **x**' bestimmen
- *D* auswerten
- Falls $D(\mathbf{x})$ > Zufallszahl → Sample **x** behalten

  - Lege Referenzfoto als Textur auf die Profilfläche → Basisfarbe für **x**



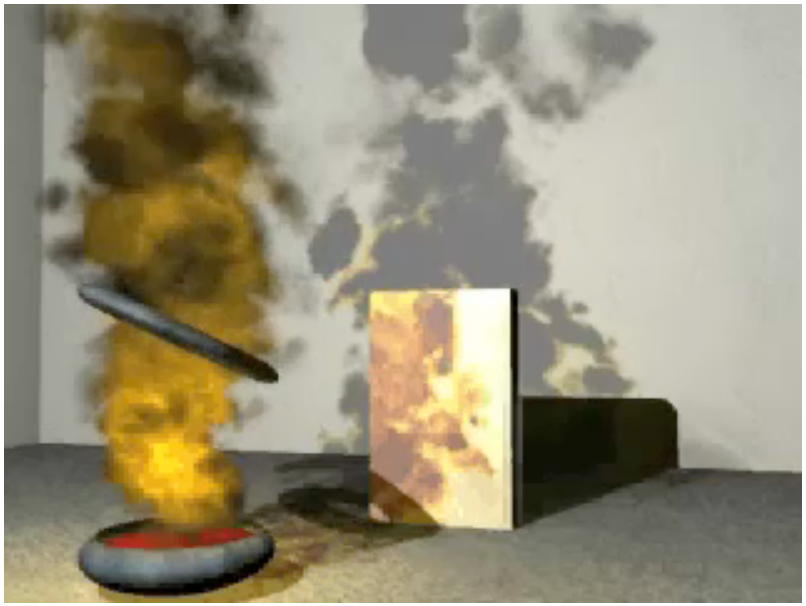
- Helligkeit eines Samples an Position **x**:

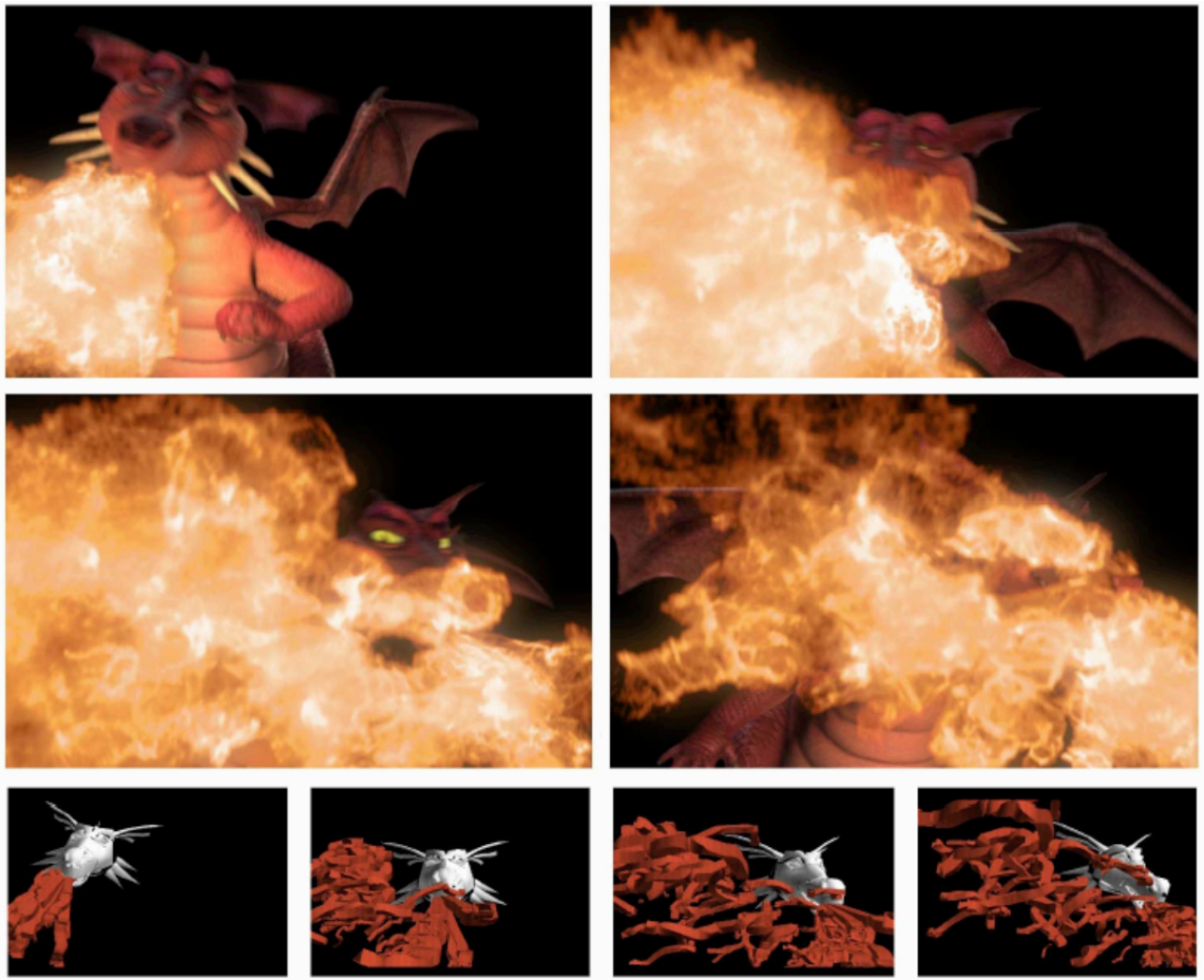
$$E(\mathbf{x}) = k\frac{D(\mathbf{x})}{n}$$

  wobei k = Faktor zur Kontrolle, n = Anzahl Samples

- Anzahl: ca. 10 Samples pro Pixel, ca 10,000 Samples pro Flamme
- Samples im Inneren von anderen Objekten werden verworfen
- Rauch: Samples > "Rauchhöhe" werden grau/schwarz gerendert

Results





Arnauld Lamorlette and Nick Foster, PDI/DreamWorks

Exkurs: Procedural Modeling of Plants with Particles

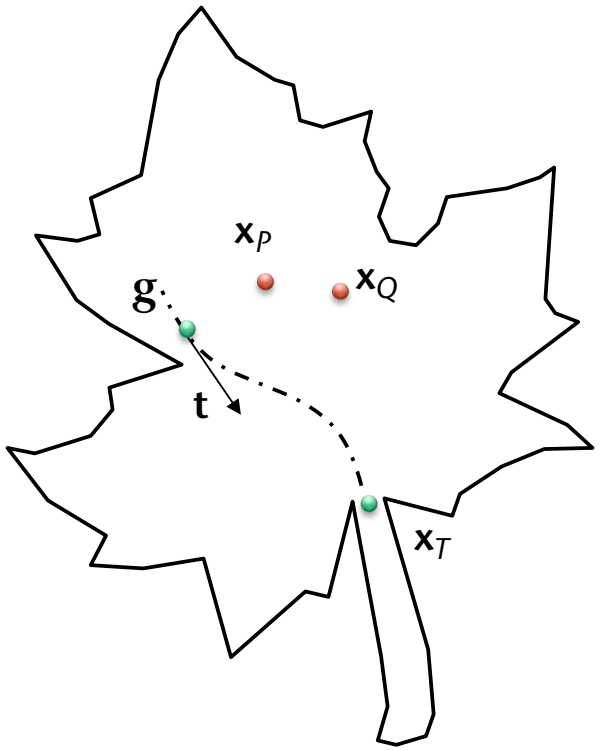
- Idee: verwende Partikel, um den Transport von Flüssigkeit in einem Blatt zu simulieren
  - Bahnen der Partikel ergeben die Adern
- Axiome:
  1. Die Natur versucht, die Länge der Bahnen zu minimieren
     → Partikel versuchen, sich auf gemeinsamen Bahnen zu bewegen
  2. Es geht keine Flüssigkeit verloren oder kommt hinzu
     → Wenn 2 Partikel eine gemeinsame Bahn verfolgen, muss die Ader dort doppelt so dick sein
  3. Alle Bahnen gehen vom Blattstiel aus

- Übersicht des Algorithmus:

```
plaziere Partikel zufällig auf der Oberfläche des Blattes
loop bis kein Partikel übrig:
    bewege jedes Partikel in Richtung seines nächsten
       Nachbarn oder der nächsten schon existierenden Bahn,
       und in Richtung des Blattstiels
    falls Partikel bei Blattstiel angekommen ist:
       lösche dieses Partikel
    falls zwei Partikel einander "nahe genug" sind:
       verschmelze beide zu einem Partikel
```

Zur Bewegung der Partikel

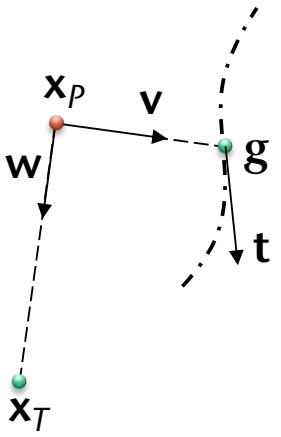
- Seien
  - $\mathbf{x}_P$ = aktuelle Position des Partikels P,
  - $\mathbf{x}_T$ = Position des Ziels (Blattstiel),
  - $\mathbf{g}$ = nächster Punkt zu $\mathbf{x}_P$ auf einer Bahn,
  - $\mathbf{t}$ = Tangente in $\mathbf{g}$ (normiert),
  - $\mathbf{x}_Q$ = nächstes Partikel zu P



- Falls $\|\mathbf{x}_P - \mathbf{g}\| < \|\mathbf{x}_P - \mathbf{x}_Q\|$:
  - Setze:

$$\mathbf{v} = \frac{\mathbf{g} - \mathbf{x}_P}{||\mathbf{g} - \mathbf{x}_P||}$$

$$\mathbf{w} = \frac{\mathbf{x}_T - \mathbf{x}_P}{||\mathbf{x}_T - \mathbf{x}_P||}$$



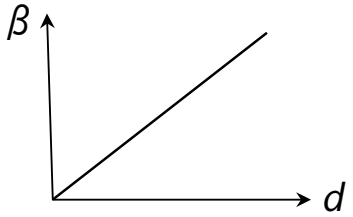
  - Neue Position:

$$\mathbf{x}'_P = \mathbf{x}_P + \alpha\mathbf{w} + (1-\alpha)\left(\beta\mathbf{v} + (1-\beta)\,\mathbf{t}\right)$$

wobei

$$\beta = \beta\left(||\mathbf{x}_P - \mathbf{g}||\right)$$

- Ein (ungefähr) lineares β ergibt z.B. Partikelbahnen, die in der Nähe der bestehenden Bahn tangential dazu verlaufen, weiter weg senkrecht darauf zu

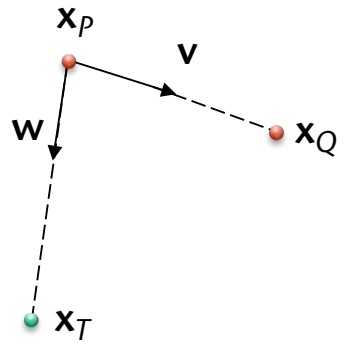


- Sonst ($||\mathbf{x}_P - \mathbf{x}_Q|| < ||\mathbf{x}_P - \mathbf{g}||$):
  - Setze

$$\mathbf{v} = \frac{\mathbf{x}_Q - \mathbf{x}_P}{||\mathbf{x}_Q - \mathbf{x}_P||}$$

  - Neue Position:

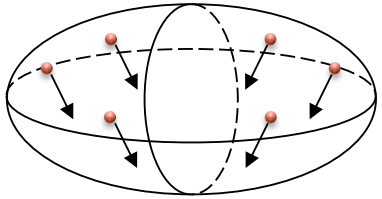
$$\mathbf{x}' = \mathbf{x}_P + \gamma\mathbf{v} + (1-\gamma)\mathbf{w}$$

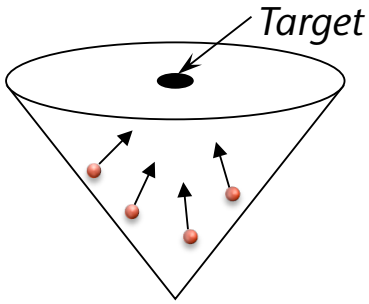


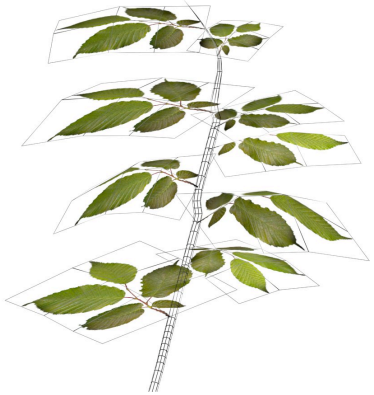
- Dicke der Adern:
  - Jedes Partikel hat Größe = Betrag der Querschnittsfläche der Ader
  - Zu Beginn: alle Partikel haben Einheitsgröße
  - Bei Verschmelzen: Größen addieren
  - Bei Auftreffen auf bestehende Bahn: Größe des Partikels zu Größe des Querschnitts der Bahn ab dem Auftreffpunkt bis zum Ziel dazuaddieren

Modellierung von Bäumen

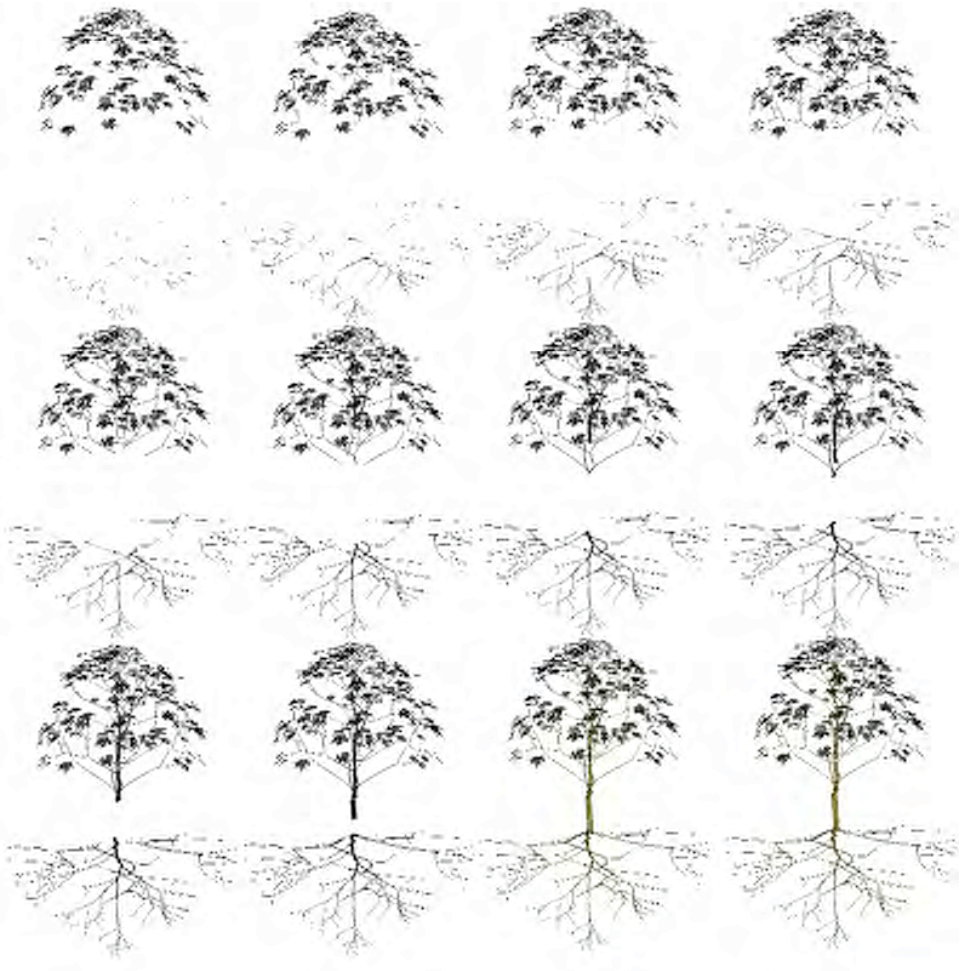
- Funktioniert genau gleich
- Vorgabe: Geometrie für die initialen Positionen der Partikel
  - Nur Hüllgeometrie
  - Erzeuge Partikel darin mittels stochastischem Prozeß
- Geometrie der Zweige: verbinde Kreisscheiben, die senkrecht zur Bahn entlang der Bahn plaziert werden
  - "sweep a disk along the path"
- Zweig-Primitive an die Äste setzen:





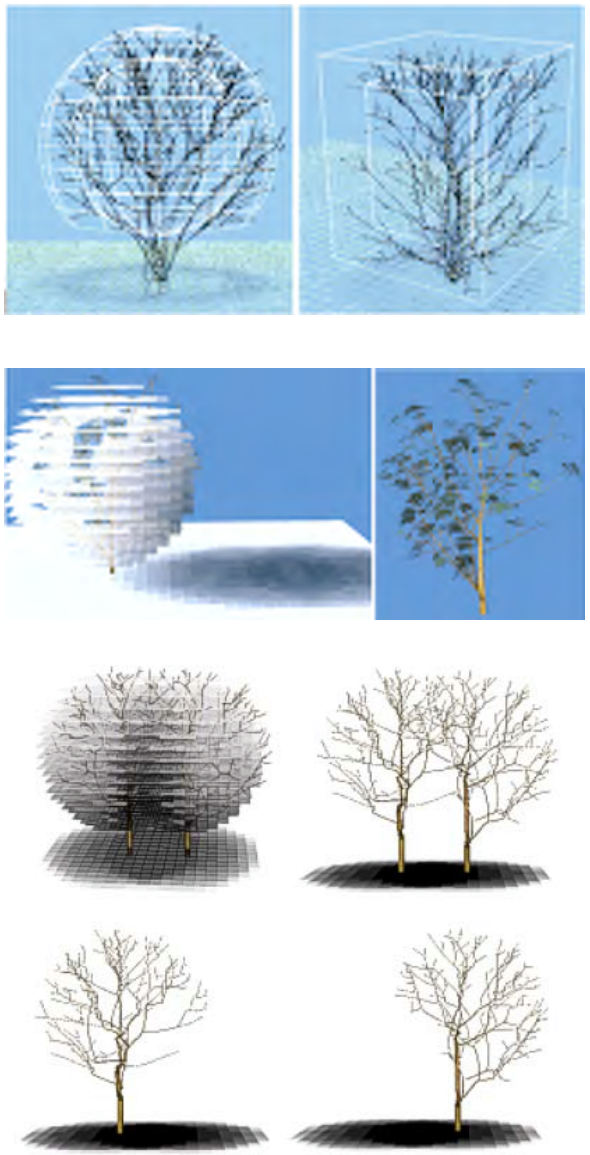


- Beispiel-Ablauf:

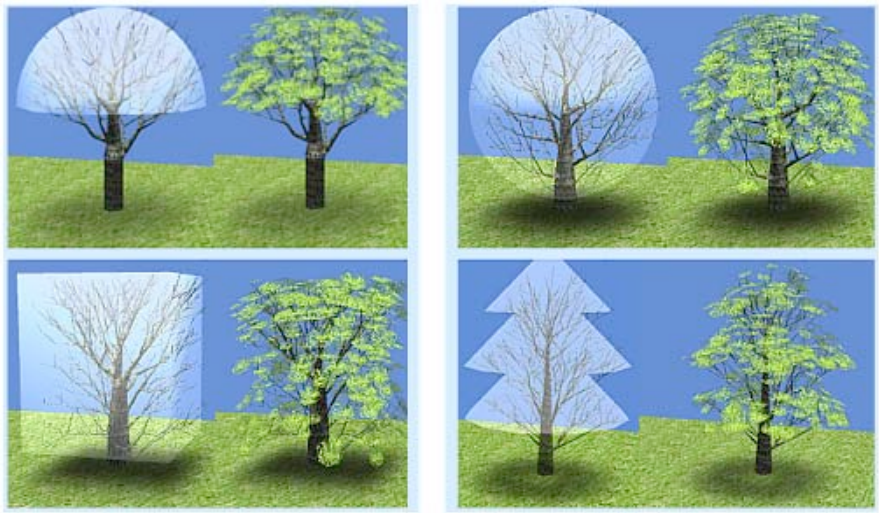


Berücksichtigung der Lichtverhältnisse

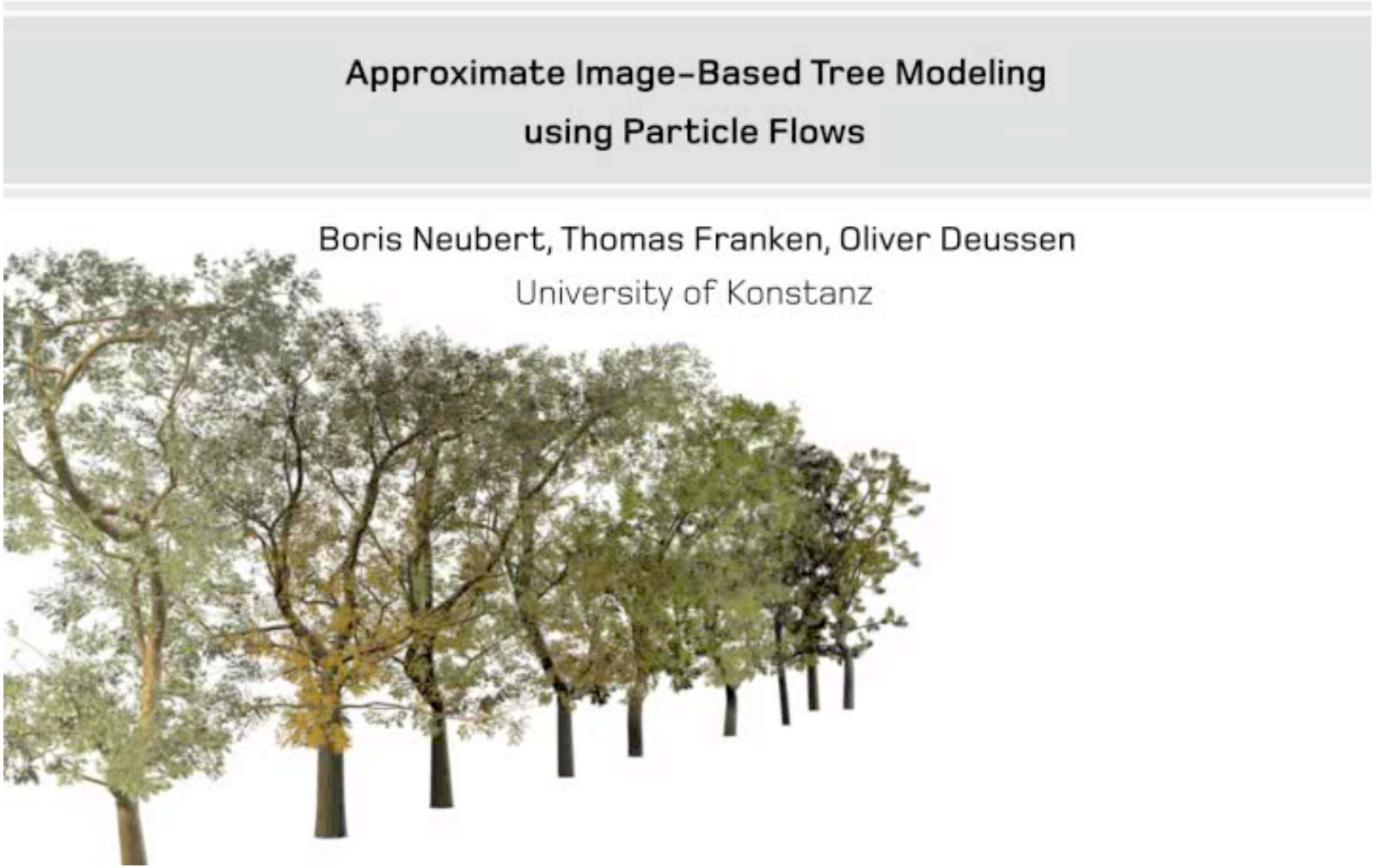
- Beobachtung: Stellen mit weniger Licht haben weniger Zweige / Blätter
- Lässt sich relativ einfach modellieren:
  - Lege den Baum in ein Gitter
  - Approximiere die (noch nicht existierende) Blätterschicht durch eine Kugel- oder Würfelschale
  - Berechne Lichteinfall für jeden Gitterknoten durch die Schale hindurch (ray casting)
  - Bei der Partikelerzeugung: passe Wahrscheinlichkeit einer Erzeugung dem Lichteinfall an (trilinear interpolieren)



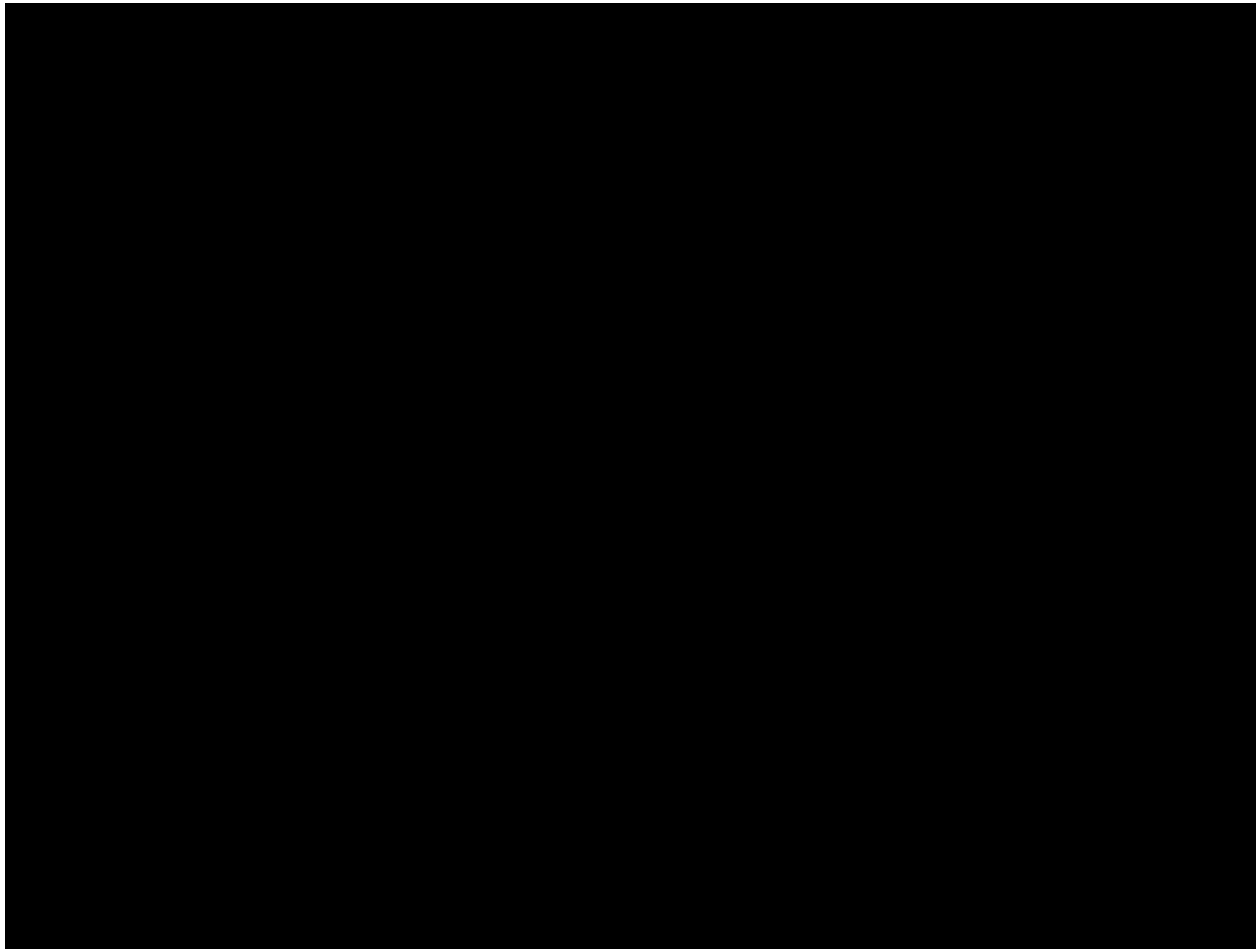
Beispiele



Videos


Approximate Image-Based Tree Modeling
using Particle Flows
Boris Neubert, Thomas Franken, Oliver Deussen
University of Konstanz

Historischer Video



Andre and Wally B (Pixar)