

Virtuelle Realität Partikelsysteme



G. Zachmann
Clausthal University, Germany
cg.in.tu-clausthal.de

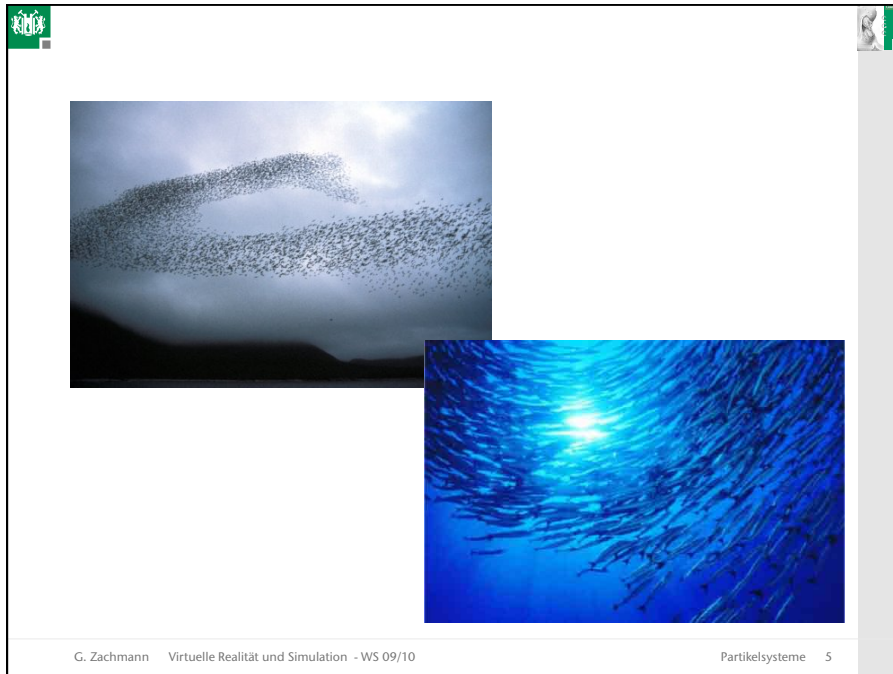
 

Modellierung/Simulation/Rendering natürlicher Phänomene



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 2

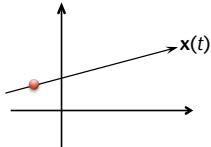
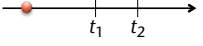




Dynamik eines Massenpunktes

- **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:**

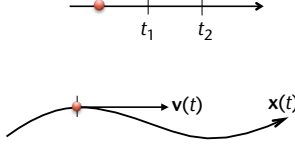
$$\mathbf{v} = \frac{\text{Weg}}{\text{Zeit}} = \frac{\mathbf{x}(t_2) - \mathbf{x}(t_1)}{t_2 - t_1}$$
 - Einheit: m/s
 - Beachte: Geschwindigkeit = Vektor
Ort des Partikels = Punkt!

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 7

- **Momentangeschwindigkeit:**

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

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

- **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 :**

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

└──────────┘
Newtons 2. Gesetz

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 8

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:

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

$$\mathbf{x}_0(t) = \mathbf{x}_0 + \int_{t_0}^t \mathbf{v}(t) dt$$
- Diskretisieren und Linearisieren:

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

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

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

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 9

Der Phasenraum

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

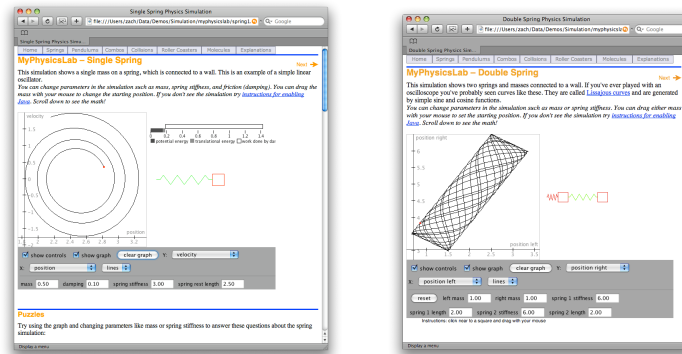
$$\mathbf{q} = (\mathbf{x}, \mathbf{v}) = (x_1, x_2, x_3, v_1, v_2, v_3)$$

$$= (x_1, x_2, x_3, \dot{x}_1, \dot{x}_2, \dot{x}_3) \in \mathbb{R}^6$$
- Der Raum aller möglicher Zustände heißt **Phasenraum** (*phase space*)
- Die Dimension ist $6n$, n = Anzahl Partikel
- Bewegungsgleichungen im Phasenraum:

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

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 10

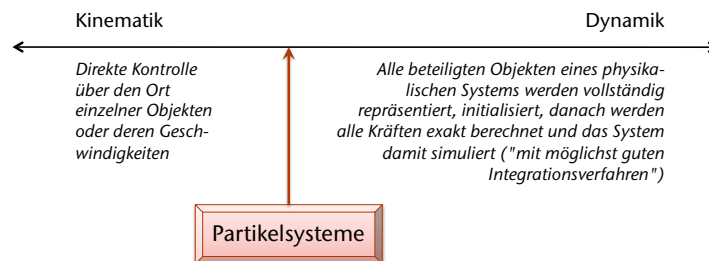
- Beispiel für ein Partikel, das sich nur auf der x-Achse bewegen kann und durch eine Feder in einer Ruhelage gehalten wird:



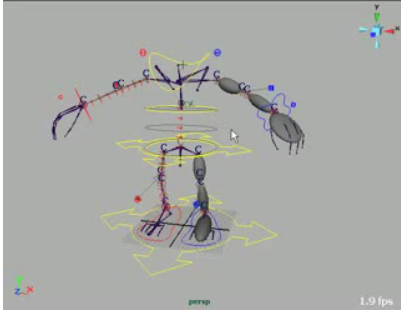
www.myphysicslab.com

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



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 13

Partikelsysteme

■ Definition: Ein **Partikelsystem** besteht aus

1. Einer Menge von **Partikeln**; jedes Partikel i hat (mindestens) folgende Attribute:
 - Masse, Position, Geschwindigkeit ($m_i, \mathbf{x}_i, \mathbf{v}_i$)
 - Alter a_i
 - Kräfteakkumulator F_i
 - evtl.: wie z.B. Farbe, Transparenz, Optische Größe, Lebensdauer, Typ/Art ...
2. Einer Menge **Partikelquellen**; jede 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

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 14

- **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 intuitive Art, den stochastischen Prozess für die initiale Position von Partikeln zu beschreiben
 - Häufig: Kreisscheibe, Würfel, Kegel, etc.

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 15

Das Execution Model

- Der Ablauf eines Partikelsystems:

```

loop forever:
  rendere alle Partikel
  Δt := Rendering-Zeit
  kille alle Partikel mit Alter > TTL (max. 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 gegebenenfalls Geschwindigkeiten (*)
  aktualisiere Positionen (ein weiterer Eulerschritt)
  modifiziere eventuell Positionen (z.B. wg. Constraints)
  sortiere Partikel nach Tiefe (für Alpha-Rendering)
  
```

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 16

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 $t_{\text{current}} - t_{\text{gen}} > \text{TTL}$ testen
 - Wird später bei paralleler Implementierung wichtig
- Zu (*) im Algorithmus:
 - Ist "un-physikalisch", erlaubt aber bessere kinematische Kontrolle durch den Programmierer / Animator
 - Ist auch bei Kollisionen nötig
- Der Rest ist Intuition und Kreativität ...
- Oft speichert man eine kleine Historie der Positionen der Partikel, um einen einfachen "motion 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)

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 17

Beispiel eines Partikelsystems

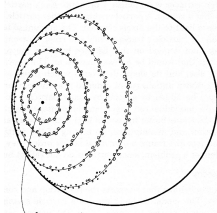
- Ausschnitt aus "Wrath of Khan":



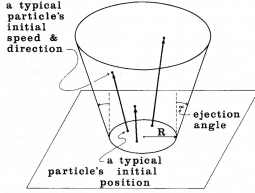
(William Reeves, 1984)

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 18

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

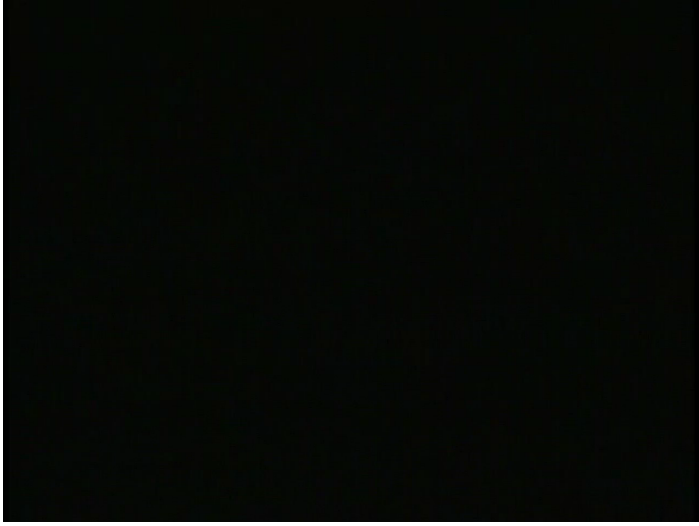


- Stochastische Prozesse für Partikelgenerierung:
 - Kegelstumpf senkrecht zu Kugeloberfläche
 - Varianz für Lebensdauer
- Farbe = $f(\text{Alter})$



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 19

Exkurs: die Panspermie-Hypothese

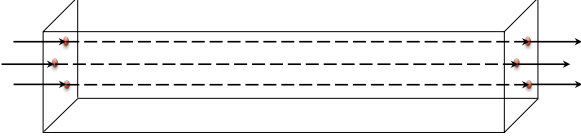


Karl Sims, 1990

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 20

Operationen auf Partikeln

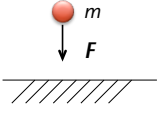
- Positionsooperationen:
 - Eher selten
 - z.B. "Tunneln"



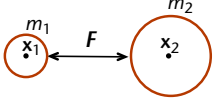
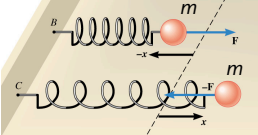
G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 21

Physikalische Effekte

- Schwerkraft:

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

- 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
 

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 22

■ 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

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 23

■ Elektromagnetische Kraft (Lorentz-Kraft):


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

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


Faraday'sches Horn

Dauer: 45s

Inhalt: Lorentzkraft


 Produziert im Rahmen des
eLearnPhysik Projektes

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

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 24

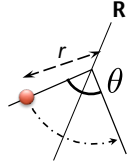
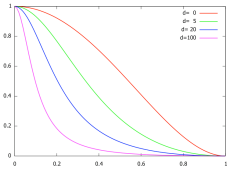
Nicht-physikalische Effekte

- **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

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

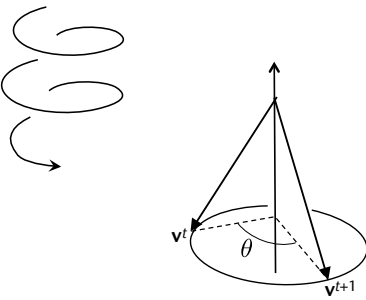



- Erweiterungen:
 - Masse des Partikels einbeziehen
 - B-Spline als Achse des Vortex (für Tornados z.B.)
 - Achse des Vortex animieren

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 25

- 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



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 26

Kollisionen

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

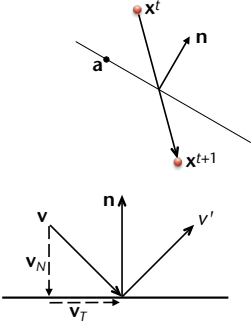
$$(\mathbf{x}^t - \mathbf{a}) \cdot \mathbf{n} > 0 \wedge (\mathbf{x}^{t+1} - \mathbf{a}) \cdot \mathbf{n} < 0$$
- Koll.-Behandlung: reflektiere \mathbf{v}

$$\mathbf{v}_N = (\mathbf{v} \cdot \mathbf{n}) \mathbf{n}$$

$$\mathbf{v}_T = \mathbf{v} - \mathbf{v}_N$$

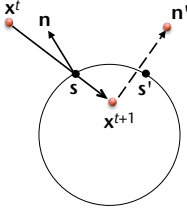
$$\mathbf{v}' = \mathbf{v}_T - \mathbf{v}_N = \mathbf{v} - 2(\mathbf{v} \cdot \mathbf{n}) \mathbf{n}$$
- Erweiterung um Reibung und elastischer/inelastischer Stoß:

$$\mathbf{v}' = (1 - \mu) \mathbf{v}_T - \varepsilon \mathbf{v}_N$$
 mit μ = Reibungskoeffizient (friction parameter) und ε = Federung / Elastizität (resilience)

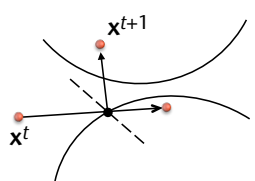
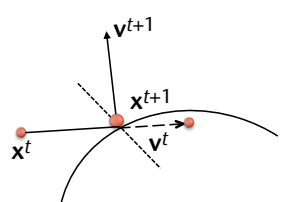


G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 27

- Fazit: Kollisionserkennung für Partikel = "Punkt-in-Geometrie-Test" bzw. Schnitttest zwischen Geradensegment und Geometrie
- Analog für Kugeln:
 - Exakten Schnittpunkt s und Normale \mathbf{n} bestimmen
 - Dann weiter wie eben
- Für Polyeder und implizite Flächen: siehe CG1
- Für Höhenfeld (Terrain): siehe CG2 (Raytracing)



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 28

- Achtung: stelle konsistenten Zustand nach der Kollisionsbehandlung her!
 - Problem: "Doppelkollisionen" an engen Stellen
 - Beispiel:
 
 - Korrekte Behandlung:
 
 - Gibt noch weitere Möglichkeiten

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 29

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.

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 30

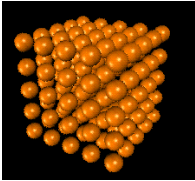
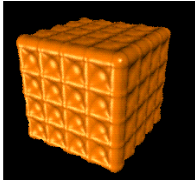
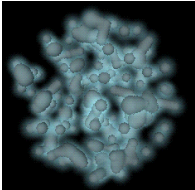
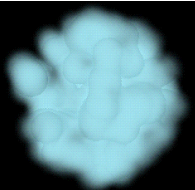
Rendering

- 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

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 31

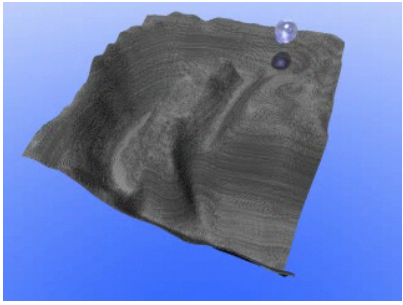
Rendering von "blobby objects"

- 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

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 32

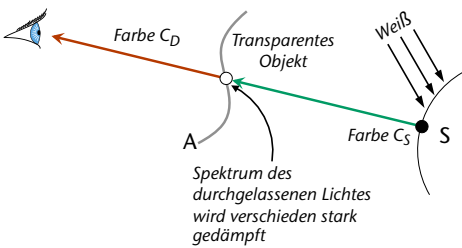
Beispiel



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 33

Rendering von transparenten Objekten

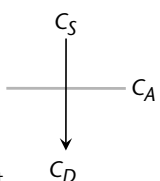
- Transparenz \approx Licht wird von einem Material teilweise durchgelassen, wobei verschiedene Wellenlängen verschieden stark gedämpft werden



- Extremfall: Farbfilter

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 34

- **Approximation: Alpha-Blending**
 - $\alpha \in [0, 1]$ = Transparenz / Opacity
 - $\alpha = 0 \rightarrow$ komplett durchsichtig,
 - $\alpha = 1 \rightarrow$ 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 $\rightarrow C_S$
 - Formel auswerten $\rightarrow C_D$
 - C_D in Framebuffer schreiben

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 35

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

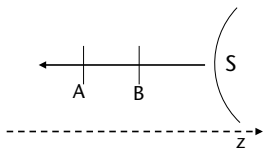
$$C'_D = \alpha_A C_A + (1 - \alpha_A) C_S$$

$$C_D = \alpha_B C_B + (1 - \alpha_B) C'_D$$

$$= \alpha_B C_B + (1 - \alpha_B) \alpha_A C_A + (1 - \alpha_B) (1 - \alpha_A) C_S$$
 - Erst B, dann A (ohne z-Test) ergibt:

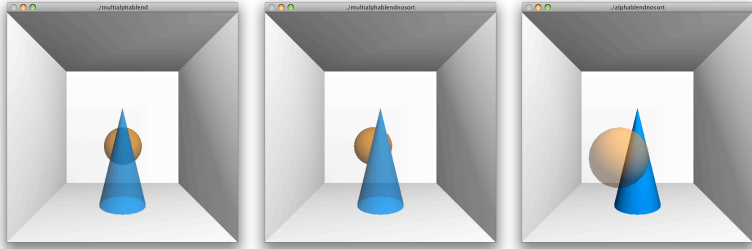
$$C'_D = \alpha_B C_B + (1 - \alpha_B) C_S$$

$$C_D = (1 - \alpha_A) \alpha_B C_B + \alpha_A C_A + (1 - \alpha_B) (1 - \alpha_A) C_S$$
- **Fazit: man muss die Polygone/Partikel von hinten nach vorne rendern, selbst dann, wenn der Z-Buffer abgeschaltet wird!**



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 36

▪ Beispiele:



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

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 37

▪ In Open GL:

- Einschalten mit:


```
glEnable( GL_BLEND );
```
- Blending-Funktion festlegen:


```
glBlendFunc( 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`)

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 38

Partikel-Systeme-Demos

<http://www.ihlabs.com/java/particles.html>

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 39

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

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 40

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

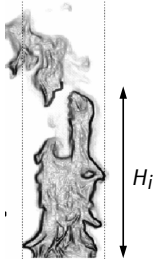
$$\mathbf{v}_P^{t+1} = \mathbf{v}_P^t + w(\mathbf{x}_P, t) + b(T_P) + d(T_P)$$
 wobei
 - w = Windfeld
 - b = Auftrieb
 - d = Diffusion = Rauschen;
 - T_P = Temperatur des Partikels = Alter
 (Vereinfachung hier: Partikel haben keine Masse)
 - In Folge-Frames weitere solche Partikel generieren, bis Max.-Anzahl für ein Spine erreicht
 - Spine-Partikel durch B-Spline verbinden

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 41

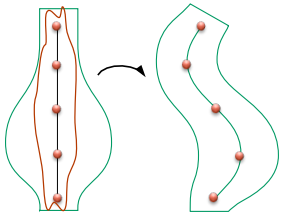
- Aufbrechen von Flammen-Elementen:
 - Das obere Stück des Spines wird zu einem zufälligen Zeitpunkt abgetrennt, wenn Höhe $> H_i$
 - Lebensdauer nach der Abtrennung:

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

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



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



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 42

■ **Rendering:**

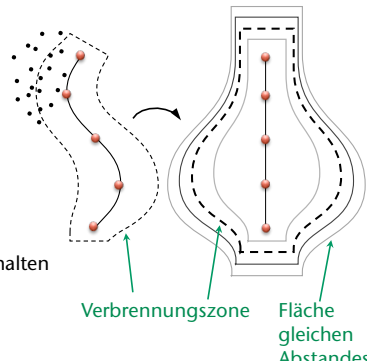
- Raum um die Flamme durch eine große Menge von Punkten sampeln gemäß der Dichtefunktion

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

wobei \mathbf{x}' der Punkt auf der (deformierten) Profilfläche ist, der zu \mathbf{x} am nächsten ist:

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

- Lege Referenzfoto als Textur auf die Profilfläche → Basisfarbe für \mathbf{x}



Verbrennungszone Fläche gleichen Abstandes

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 43

- Helligkeit eines Samples an Position \mathbf{x} :

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

wobei k = Faktor zur Kontrolle, n = Anzahl Samples

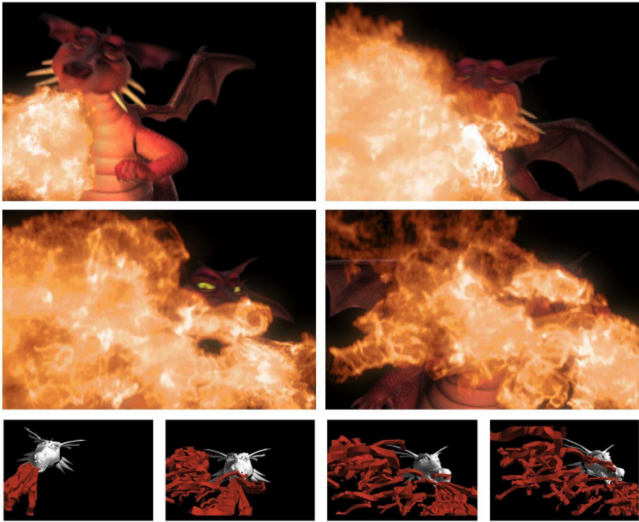
- Anzahl: ca. 10 Samples pro Pixel, ca 10,000 Samples pro Flamme
- Samples im Inneren von anderen Objekten werden verworfen
- Rauch: Samples > "Rauchhöhe" werden grau/schwarz gerendert

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 44

Results

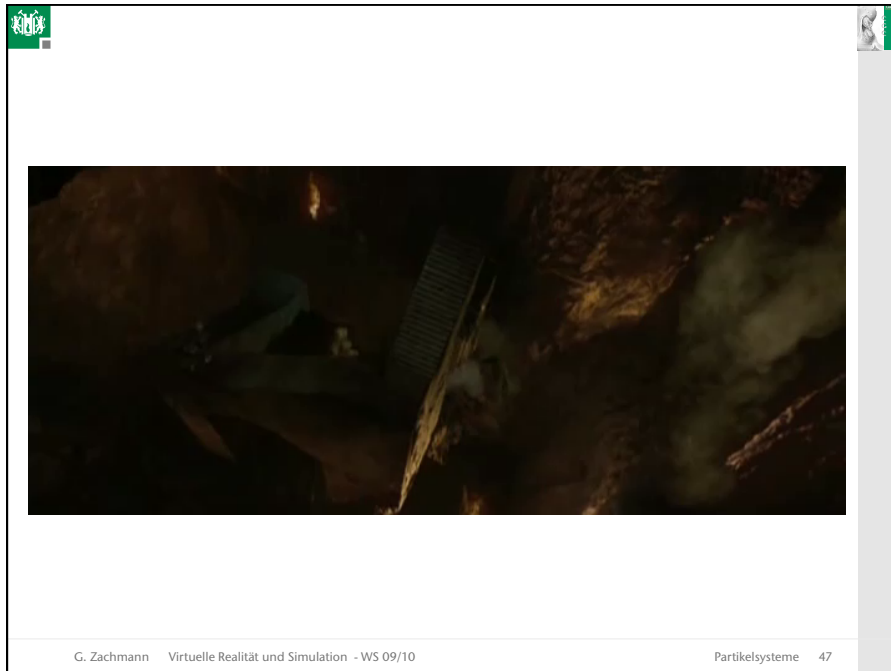


G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 45



Arnaud Lamarlette and Nick Foster, PDI/DreamWorks

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 46



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

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 48

- Übersicht des Algorithmus:

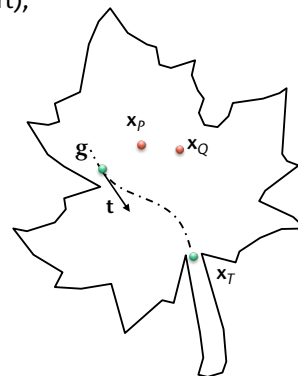
```

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

- x_p = aktuelle Position des Partikels P ,
- x_T = Position des Ziels (Blattstiel),
- g = nächster Punkt zu x_p auf einer Bahn,
- t = Tangente in g (normiert),
- x_Q = nächstes Partikel zu P



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

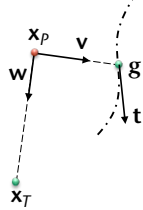
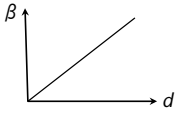
- Setze:

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

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

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

$$\beta = \beta(\|\mathbf{x}_P - \mathbf{g}\|)$$
- Ein (ungefähr) lineares β ergibt z.B. Partikelbahnen, die in der Nähe der bestehenden Bahn tangential dazu verlaufen, weiter weg senkrecht darauf zu

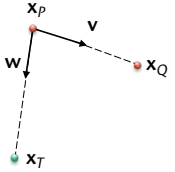
G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 51

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:

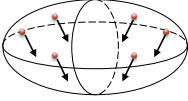
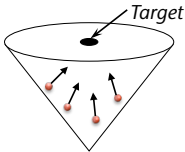
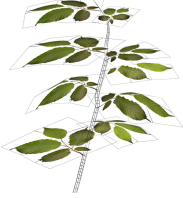
$$\mathbf{x}' = \mathbf{x}_P + \gamma \mathbf{v} + (1 - \gamma) \mathbf{w}$$
- Dicke der Adern:
 - Jedes Partikel hat Größe = Betrag der Querschnittsfläche der Ader
 - Zu Beginn: alle Partikel haben Einheitsgröße
 - Bei Verschmelzen: Größen addieren
 - Bei Auftreffen auf bestehende Bahn: Größe des Partikels zu Größe des Querschnitts der Bahn ab dem Auftreffpunkt bis zum Ziel dazuzaddieren



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 52

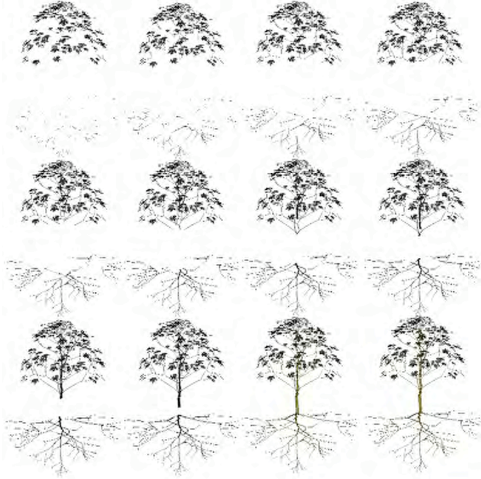
Modellierung von Bäumen

- 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 platziert werden
 - "sweep a disk along the path"
- Zweig-Primitive an die Äste setzen:

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 53

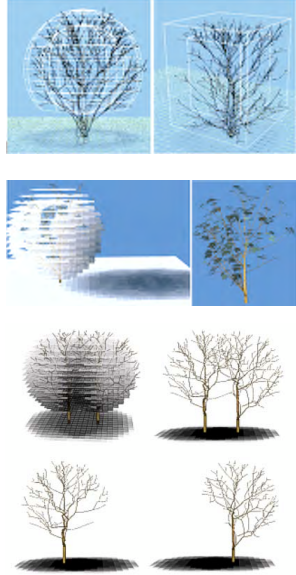
Beispiel-Ablauf:



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 54

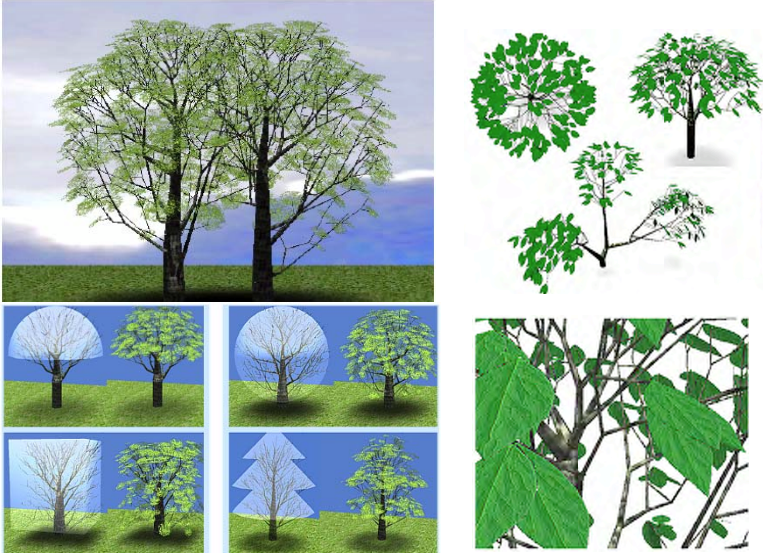
Berücksichtigung der Lichtverhältnisse

- 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)



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 55

Beispiele



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 56



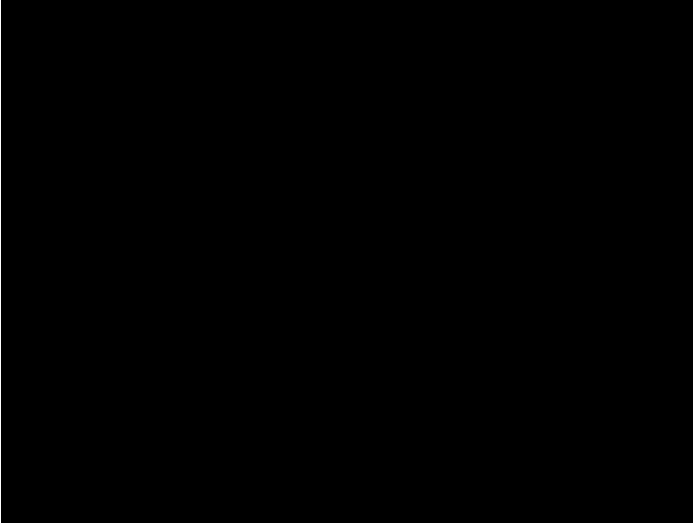
Videos

**Approximate Image-Based Tree Modeling
using Particle Flows**

Boris Neubert, Thomas Franken, Oliver Deussen
University of Konstanz

This slide features a title "Videos" in the top-left corner. Below it is a grey header box with the title "Approximate Image-Based Tree Modeling using Particle Flows". Underneath the header, the authors "Boris Neubert, Thomas Franken, Oliver Deussen" and their affiliation "University of Konstanz" are listed. The main visual is a row of several 3D rendered trees with varying heights and foliage colors, ranging from green to yellow. The slide also includes a small green logo in the top-left corner and a small portrait icon in the top-right corner. At the bottom, it contains the text: "G. Zachmann Virtuelle Realität und Simulation - WS 09/10" on the left and "Partikelsysteme 58" on the right.

Historischer Video

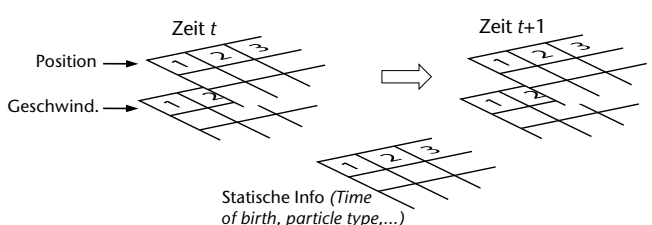
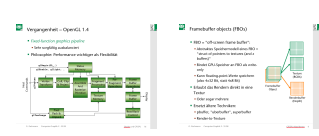


Andre and Wally B (Pixar)

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 59

Massiv-parallele Simulation

- Exkurs / Erinnerung: die GPU als massiv-parallele general-purpose Architektur
- Speicherung der Daten in Texturen:
 - Verwende 2D-Textur, da so mehr Partikel gespeichert werden können (reines Impl.-Detail)
 - Indizes nach 2D-Indizes umrechnen, oder gleich mit 2D-Indizes arbeiten

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 60

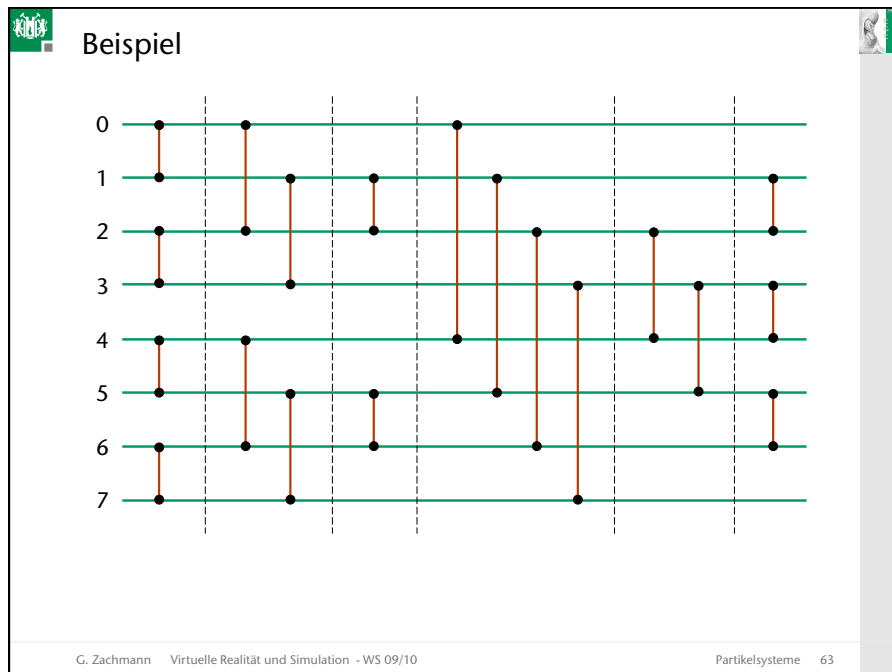
- Verwaltung freier Speicherplätze (memory management):
 - Wenn Partikel stirbt, trage Textur-Index in Liste ein
 - Bei Partikel-Generierung: hole freie Indizes aus Liste
 - Eventuell besser: Queue statt Liste, sortiert nach Index
 - Vorteil: keine Fragmentierung (keine "Löcher")
 - Nachteil: man kann nicht en bloc/parallel neue Partikel generieren und allozieren

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 61

Paralleles Sortieren

- Erinnerung: Sortierung wird für Alpha-Blending benötigt
- Lösung: **Sortiernetzwerke**
- Informelle Definition:
 - Bestehen aus einer Menge von "Leitungen"
 - Daten D_i laufen von links nach rechts durch die Leitungen i
 - Zwei Leitungen können vertikal durch einen Komparator verbunden werden
 - Falls $D_i > D_j \wedge i < j$,
dann werden die beiden Daten
durch den Komparator vertauscht
- Eigenschaft: Ein Sortiernetzwerk ist **datenunabhängig**, d.h., die Laufzeit ist unabhängig von der "Sortiertheit" der Eingabe!

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 62



Das 0-1-Prinzip

- Definition (*monoton*):
 Seien A, B zwei Mengen mit Ordnungsrelation,
 und $f : A \rightarrow B$ eine Abbildung.
 f heißt *monoton* genau dann, wenn

$$\forall a_1, a_2 \in A : a_1 \leq a_2 \Rightarrow f(a_1) \leq f(a_2)$$
- Lemma:
 Sei $f : A \rightarrow B$ *monoton*. Dann gilt

$$\forall a_1, a_2 \in A : f(\min(a_1, a_2)) = \min(f(a_1), f(a_2))$$
 Analoges gilt für *max*.
- Beweis:
 Fall 1: $a_1 \leq a_2 \Rightarrow f(a_1) \leq f(a_2)$
 $\min(a_1, a_2) = a_1, \min(f(a_1), f(a_2)) = f(a_1)$
 $f(\min(a_1, a_2)) = f(a_1) = \min(f(a_1), f(a_2))$
 Fall 2: $a_2 < a_1 \rightarrow$ analog

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 64

■ Erweiterung von $f : A \rightarrow B$ auf Folgen über A bzw. B :

$$f(a_0, \dots, a_n) = f(a_0), \dots, f(a_n)$$

■ Lemma:
 Sei f eine monotone Abbildung und \mathcal{N} ein Komparatornetzwerk.
 Dann kommutieren \mathcal{N} und f , d.h.

$$\forall n \forall a_0, \dots, a_n : \mathcal{N}(f(a)) = f(\mathcal{N}(a))$$

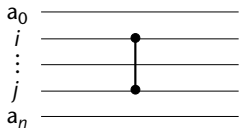
G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 65

■ Beweis:

■ Sei $a = (a_0, \dots, a_n)$ Folge

■ Notation: schreibe einen Komparator zwischen Leitung i und j so:



$[i : j](a)$



■ Es gilt:



$$\begin{aligned}
 [i : j](f(a)) &= [i : j](f(a_0), \dots, f(a_n)) \\
 &= (f(a_0), \dots, \underbrace{\min(f(a_i), f(a_j))}_i, \dots, \underbrace{\max(f(a_i), f(a_j))}_j, \dots, a_n) \\
 &= (f(a_0), \dots, f(\min(a_i, a_j)), \dots, f(\max(a_i, a_j)), \dots, a_n) \\
 &= f(a_0, \dots, \min(a_i, a_j), \dots, \max(a_i, a_j), \dots, a_n) \\
 &= f([i : j](a))
 \end{aligned}$$

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 66

- **Satz (0-1-Prinzip):**
 Sei \mathcal{N} ein Komparatornetzwerk.
 Falls \mathcal{N} jede 0-1-Folge sortiert, dann sortiert es auch jede beliebige Folge!

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 67

- **Beweis (durch Widerspruch):**
 - Annahme: Folge a wird nicht durch \mathcal{N} sortiert
 - Dann ist $\mathcal{N}(a) = b$ nicht korrekt sortiert, d.h. $\exists k : b_k > b_{k+1}$
 - Definiere $f : A \rightarrow \{0,1\}$ wie folgt:

$$f(c) = \begin{cases} 0, & c < b_k \\ 1, & c \geq b_k \end{cases}$$
 - Nun gilt:

$$f(b) = f(\mathcal{N}(a)) \stackrel{f \text{ monoton}}{=} \mathcal{N}(f(a)) = \mathcal{N}(a')$$

wobei a' eine 0-1-Folge ist.
 - Aber: $f(b)$ ist unsortiert, denn $f(b_k) = 1$ und $f(b_{k+1}) = 0$
 - Also ist auch $\mathcal{N}(a')$ unsortiert, d.h., wir haben aus a eine 0-1-Folge konstruiert, die von \mathcal{N} **nicht** sortiert wird.

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 68

Batcher's Odd-Even-Mergesort [1968]

- Im Folgenden sei die Länge n einer Folge a_0, \dots, a_n immer eine Zweier-Potenz, d.h. $n = 2^k$
- Zunächst die Sub-Routine "odd-even merge":

```

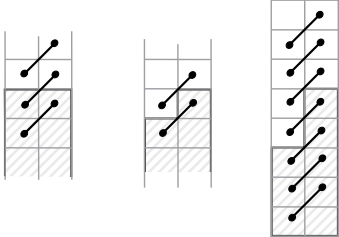
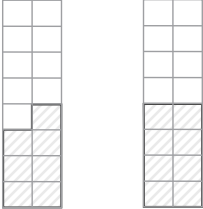
oem(  $a_0, \dots, a_{n-1}$  ):
precondition:  $a_0, \dots, a_{n/2-1}$  und  $a_{n/2}, \dots, a_{n-1}$  sind beide sortiert
postcondition:  $a_0, \dots, a_{n-1}$  ist sortiert
if  $n = 2$ :
  compare  $[a_0 : a_1]$  (1)
if  $n > 2$ :
   $\bar{a} \leftarrow a_0, a_2, \dots, a_{n-2}$  (even sub-sequence)
   $\hat{a} \leftarrow a_1, a_3, \dots, a_{n-1}$  (odd sub-sequence)
   $\bar{b} \leftarrow \text{oem}(\bar{a})$ 
   $\hat{b} \leftarrow \text{oem}(\hat{a})$  (*)
  copy  $\bar{b} \rightarrow a_0, a_2, \dots, a_{n-2}$ 
  copy  $\hat{b} \rightarrow a_1, a_3, \dots, a_{n-1}$  (**)
  for  $i \in \{1, 3, 5, \dots, n-3\}$ 
    compare  $[a_i, \dots, a_{i+1}]$  (2)
  
```

G. Zachmann Virtuelle Realität und Simulation - WS 09/10
Partikelsysteme 69

Korrektheit:

- Mittels Induktion und dem 0-1-Prinzip
- Induktionsanfang: $n = 2$
- Induktionsschritt: $n = 2^k, k > 1$
- Betrachte 0-1-Folge a_0, \dots, a_{n-1}
- Schreibe diese in 2 Spalten
- Markiere 0 = weiß, 1 = grau
- Offensichtlich: \bar{a} und \hat{a} bestehen beide aus zwei sortierten Hälften, d.h., Vorbedingung von *oem* ist erfüllt
- Nach Zeile (*) haben wir diese Situation (die ungerade Teilfolge kann höchstens zwei 1-en mehr enthalten)

G. Zachmann Virtuelle Realität und Simulation - WS 09/10
Partikelsysteme 70

- Nach Zeile (**) werden diese Vergleiche vorgenommen und es gibt nur diese 3 Fälle:
 
- Danach ist in jedem Fall eine dieser beiden Situation hergestellt:
 
- D.h., die Ausgabefolge ist sortiert
- Fazit: jede 0-1-Folge wird sortiert
- Laufzeit : $T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{2} - 1 \in O(n \log n)$

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 71

- Der Sortier-Algorithmus:


```

oemSort(a0, ..., an-1) :
  if n = 1:
    return
  a0, ..., an/2-1 ← oemSort(a0, ..., an/2-1)
  an/2, ..., an-1 ← oemSort(an/2, ..., an-1)
  oem(a0, ..., an-1)
      
```
- Laufzeit:

$$T(n) \in O(n \log^2 n)$$

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 72

Die Abbildung auf Shader (Stream Programming Model)

- Daten werden zu Beginn in einer Textur auf die GPU geladen
- Auf der CPU läuft folgendes Programm:

```

oemSort(n):
  if n = 1 → return
  oemSort(n/2)
  oem(n,1)
  
```

```

oem( n, step ):
  if n = 2:
    oemEndShader ausführen
  else:
    oem( n/2, step*2 )
    oemRekursionShader ausführen
  
```

- Mit dem Step-Parameter erreicht man eine Sortierung "in situ"

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 73

- Der Shader (stream kernel) für das Ende einer Rekursion:

```

oemEndShader( i, step ):
  // are we on the even or the odd side?
  if i/step is even:
    div = 1
  else:
    div = -1
  a0 ← SortData[i]           // SortData = Textur =
  a1 ← SortData[ i+div+step ] // globales "Array"
  if div > 0:
    output max(a0,a1)        // schreibe ins
  else:
    output min(a0,a1)        // Ausgabe-Array
  
```

- Der oemEndShader bildet Zeile (1) aus dem Algorithmus ab
- Erinnerung: ein Shader wird für jeden Index i einmal (parallel) aufgerufen

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 74

- Der Shader für die Rekursion selbst:

```

oemRecursionShader( i, step, n ):
if i < step || i ≥ n-step:
    output SortData[i]
else:
    a_i ← SortData[i]
    a_i_plus_1 ← SortData[ i+step ]
    if i/step is even:
        output max( a_i, a_i_plus_1 )
    else:
        output min( a_i, a_i_plus_1 )

```

- Der oemRecursionShader bildet die Zeile (2) aus dem Algorithmus ab

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 75

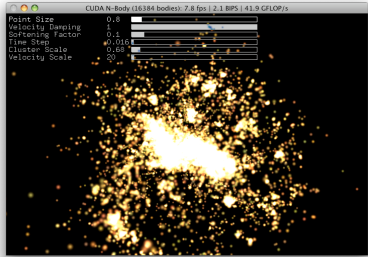
- Laufzeit:

$$\frac{1}{2} \log^2 n + \frac{1}{2} \log n \quad \text{rendering passes}$$

- Ergibt 210 Passes für 1024 x 1024 Partikel
 - Kann man inkrementell machen, also eine kleine Anzahl Sortier-Passes pro Frame

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 76

Demos



The screenshot shows a window titled "CUDA N-Body (16134 bodies) 7.8 fps / 2.3 BPP / 41.9 GLOPs". The window contains a control panel with the following settings:

Point Size	0.8
Velocity Ramping	0
Collision Factor	0.1
Time Step	0.01
Cluster Scale	0.05
Velocity Scale	0.1

The main area of the window displays a dense cluster of yellow and orange particles, with a bright central region, representing an N-body simulation.

N-body simulation

<http://www.nvidia.com/cuda>

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 Partikelsysteme 77