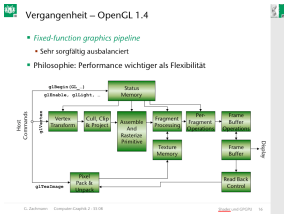
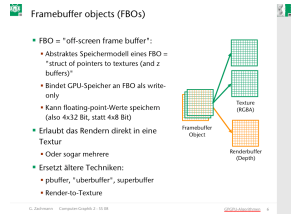
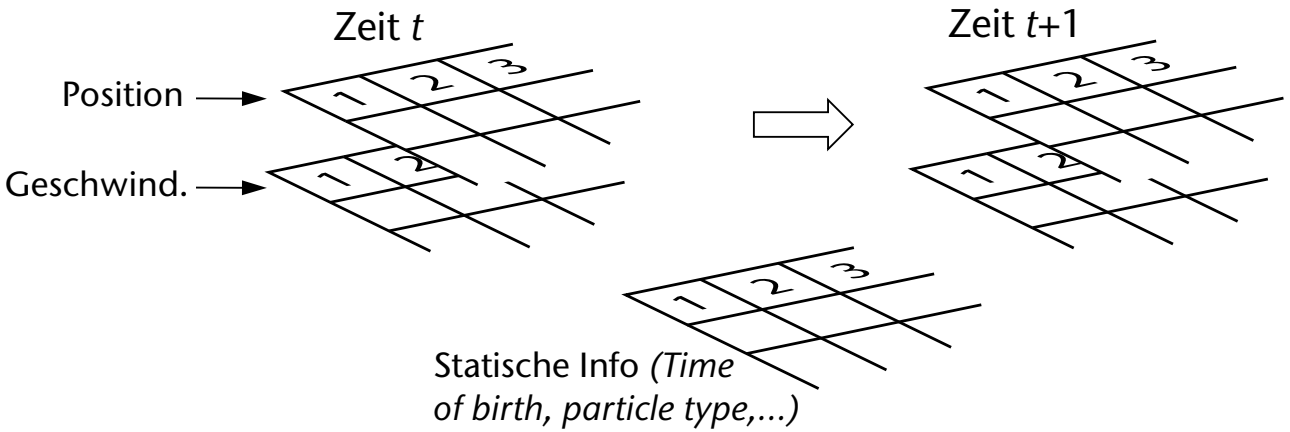
Massiv-parallele Simulation

- Exkurs / Erinnerung: die GPU als massiv-parallele general-purpose Architektur





- Speicherung der Daten in Texturen:

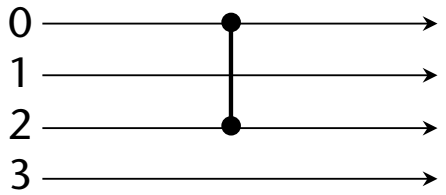


  - Verwende 2D-Textur, da so mehr Partikel gespeichert werden können (reines Impl.-Detail)
    - Indizes nach 2D-Indizes umrechnen, oder gleich mit 2D-Indizes arbeiten

- 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

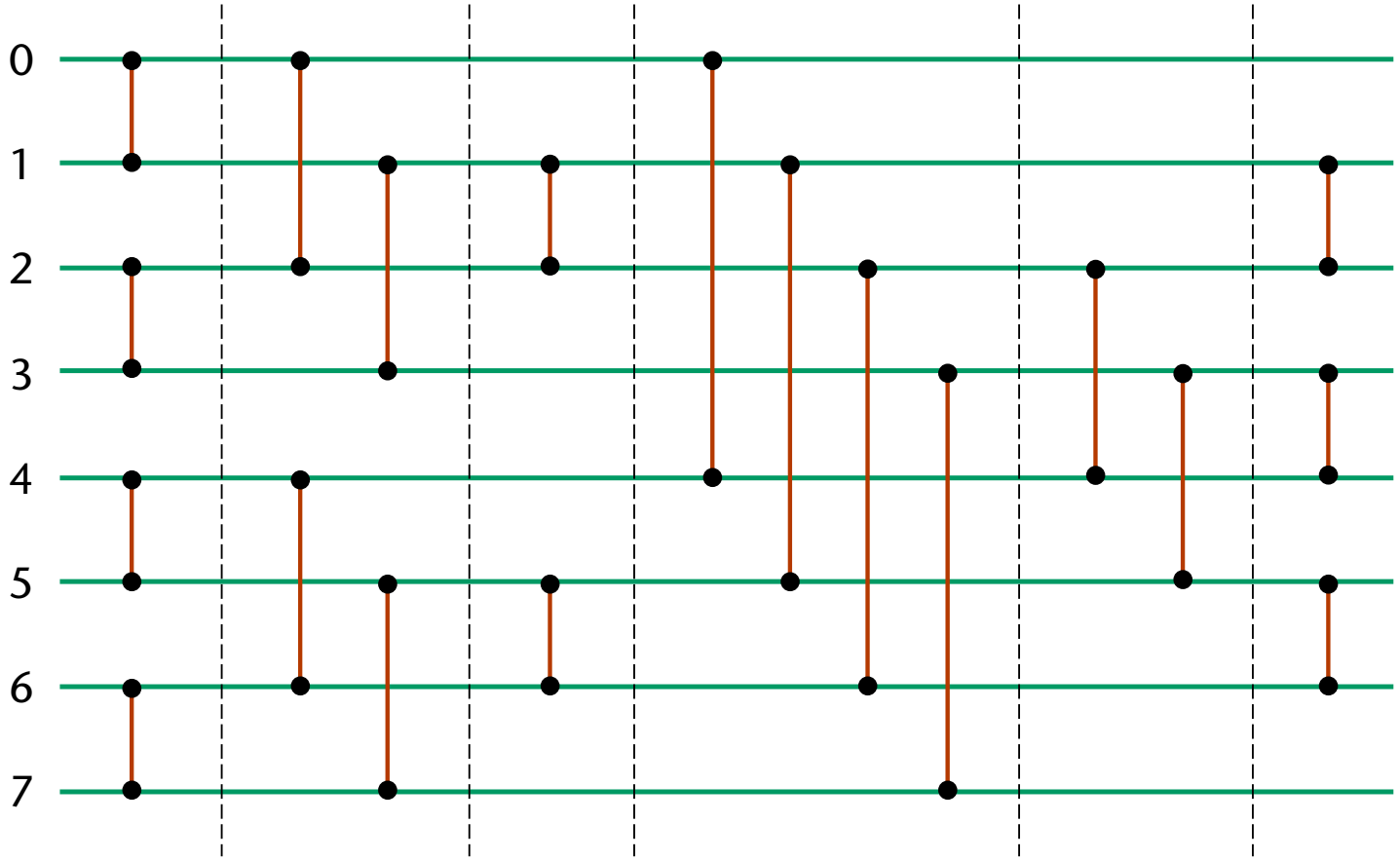
Paralleles Sortieren

- 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!

Beispiel

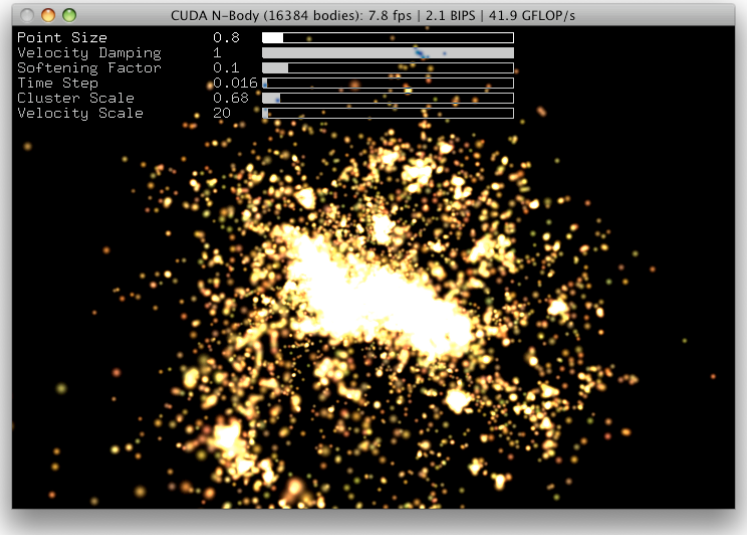


- Laufzeit:

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

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

Demos



N-body simulation

http://www.nvidia.com/cuda