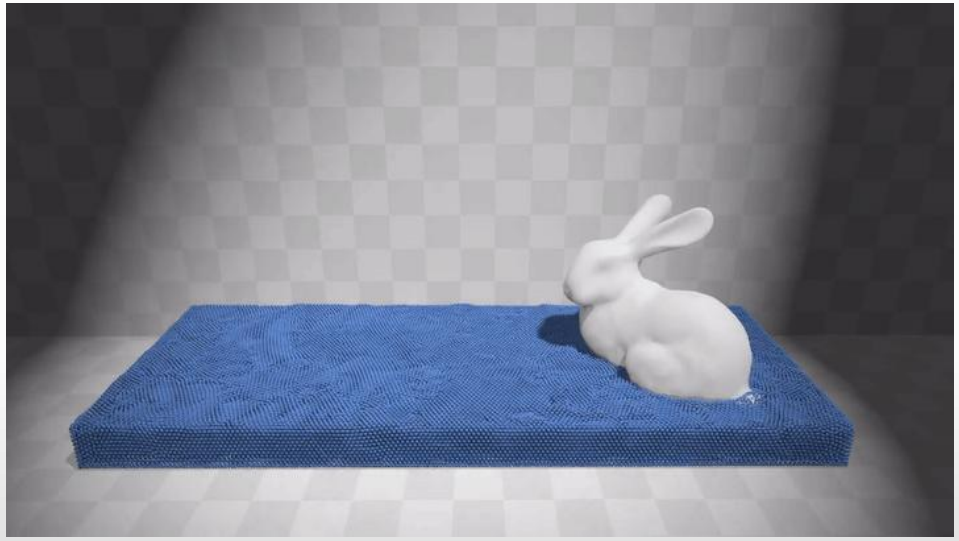
Flüssigkeitssimulation in Echtzeit



Beschleunigung auf der GPU

Material: lfranke.github.io/fluess

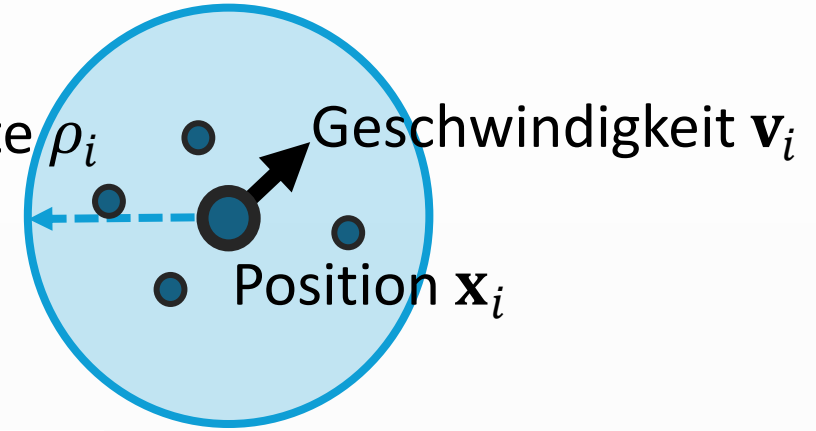
Linus Franke

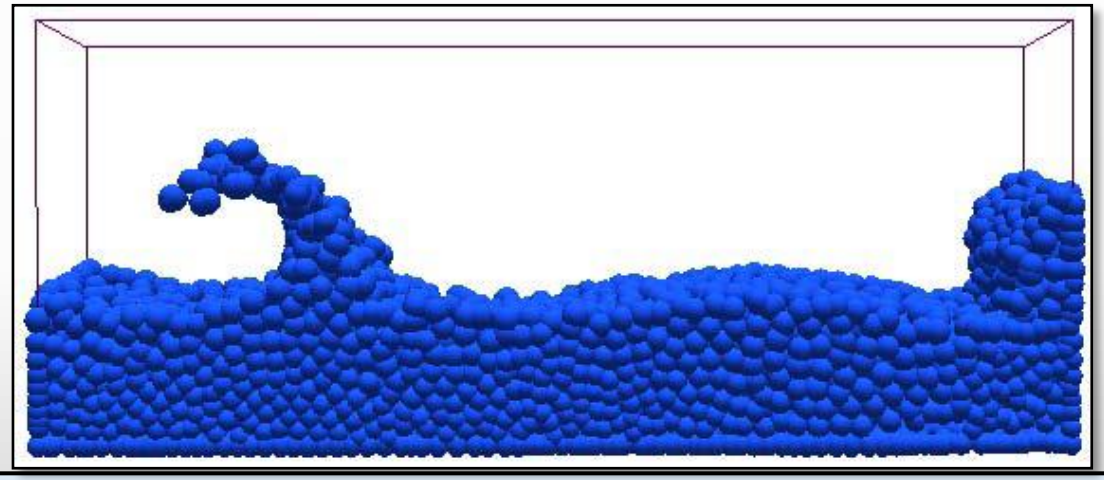
Letzte Woche

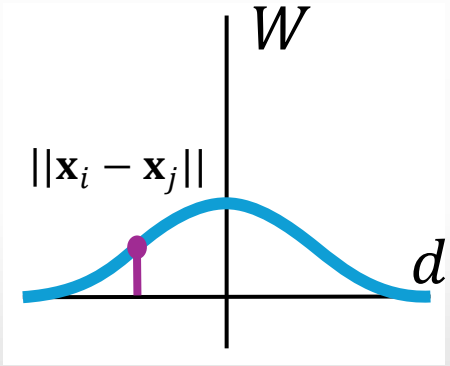
- Navier-Stokes Gleichung (Lagrange)
  - Änderung der Geschwindigkeit:
    - entgegen dem Druckgradient
    - mit viskoser Dämpfung
    - unter Einwirkung der Gravitation
  - Masse bleibt erhalten (inkompressibel)
- Partikelbasierte Flüssigkeiten
  - SPH-Kernel (Glättungsfunktion)

Momentumserhaltung $\frac{\mathrm{D}\mathbf{v}}{\mathrm{D}t} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{v} + \mathbf{g}$

Masseerhaltung $\nabla \cdot \mathbf{v} = 0$





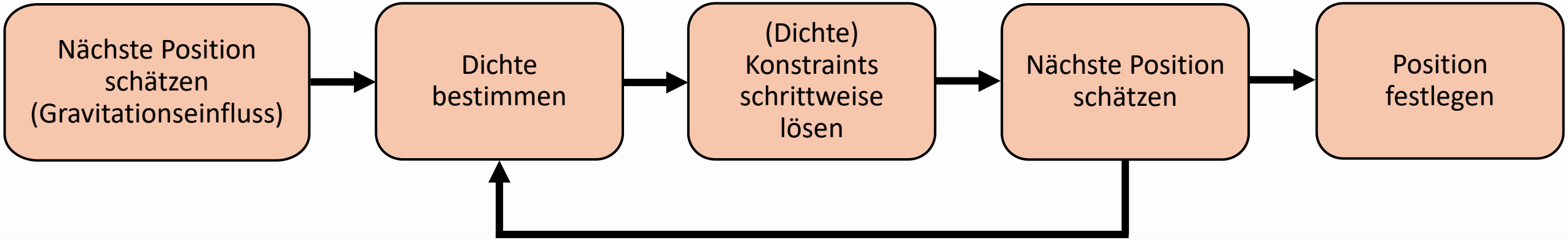


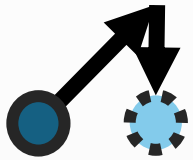
Material: lfranke.github.io/fluess

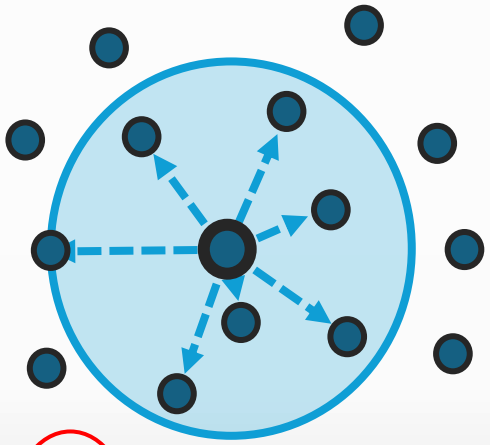
Letzte Woche

- Position-based Fluids [Macklin and Müller 2013]

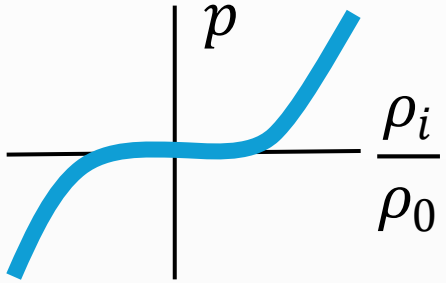
Heute:
In Echtzeit, auf der GPU, viele Partikel



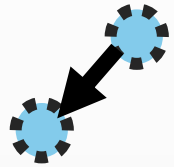




$$A_i \approx \sum_j \frac{m_j}{\rho_j} A_j W(\mathbf{x}_i - \mathbf{x}_j, h)$$



ρ_0: Ruhedichte



Nachbarsuche

- Häufige Summen über Partikel

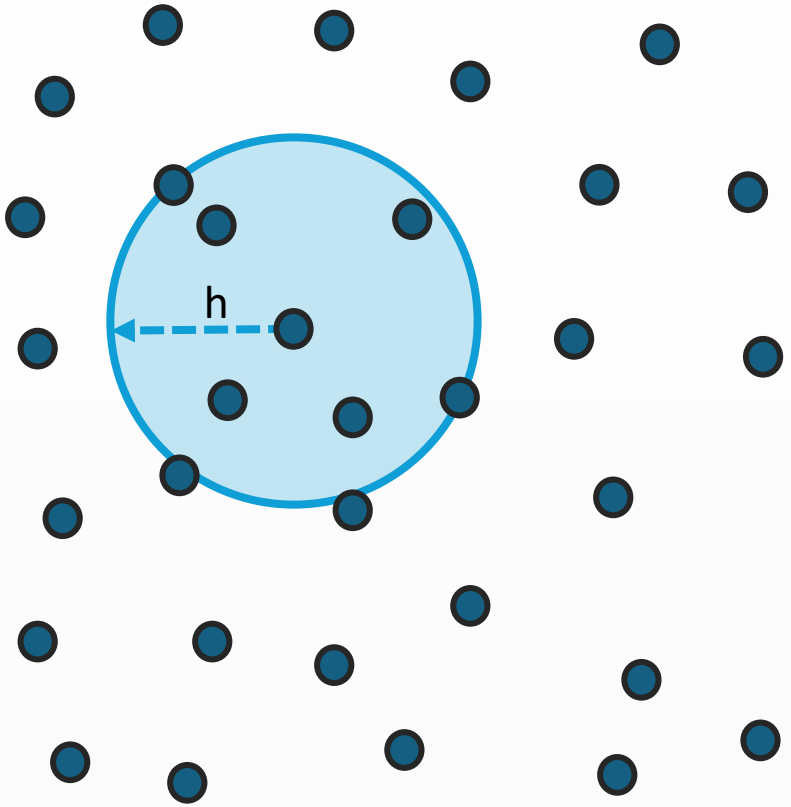
$$\sum_j (\cdot) W(\mathbf{x}_i - \mathbf{x}_j, h)$$

  - Wenn für jedes Partikel: $\mathcal{O}(n^2)$
  - Optimieren? Vorberechnen der Nachbarn

- Glättungsfunktion (Smoothing Kernel) kompakt
  - $W(r,h) = 0$ wenn $\|r\| \geq$ h
  - Funktion und Ableitungen haben keinen Einfluss außerhalb des Stützradius h

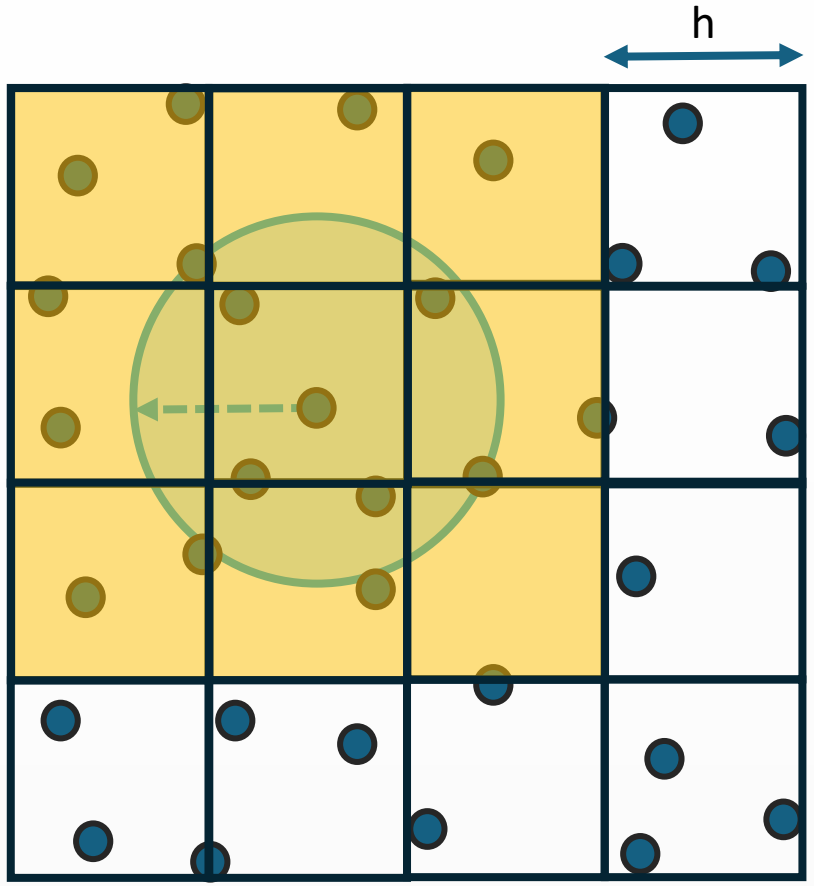
- Wenn ein Partikel im Durchschnitt k Nachbarn hat und wir alle Nachbarn pro Partikel kennen:

$$\mathcal{O}(kn)$$

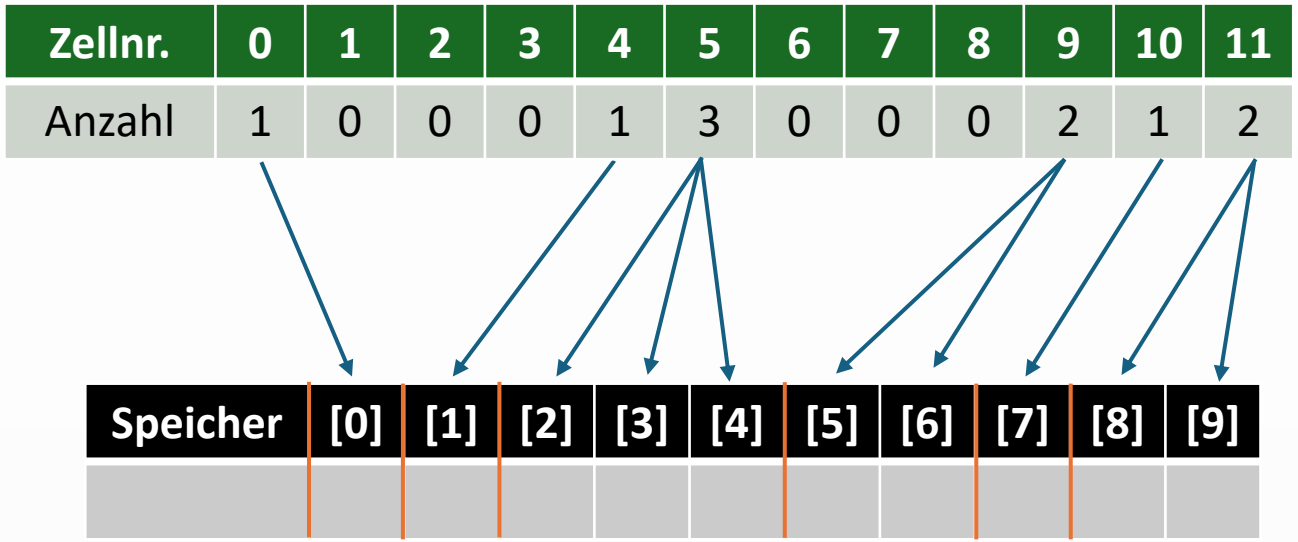


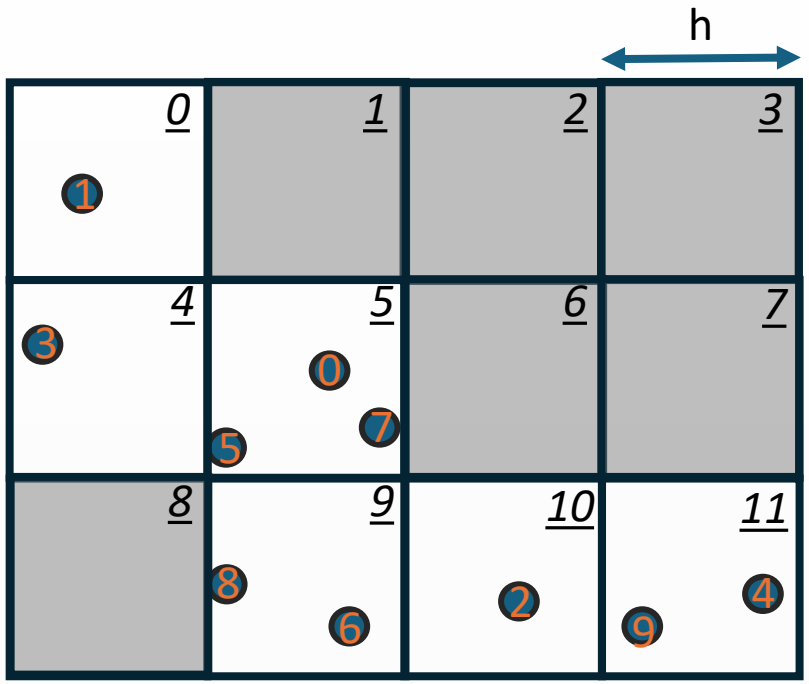
Gitterbasiert

- Gitterstruktur mit $n \times m$ Zellen
  - Zelllänge gleich dem Stützradius h
  - Nachbaren sind in
    - derselben Zelle
    - direkter Nachbarzelle
  - $i_{cell} = \left\lfloor \frac{x}{h} \right\rfloor + \left\lfloor \frac{y}{h} \right\rfloor n$
- Jede Zelle enthält variable viele Partikel
- GPU hat kein direktes Memory Allokationssystem, nachallokieren schwierig
  - Verkettete Liste? Keine Cachekoherenz, langsam
  - Array?
    - Vorallokieren? Maximales Fassungsvermögen pro Zelle begrenzen?
      - Fehler in der Simulation
    - Maximales Fassungsvermögen pro Zelle: Anzahl aller Partikel
      - Viel Speicherverbrauch, GPU Speicher evtl. zu klein
    - Viel verschwendeter Speicher für leere Zellen



Gitterbasiert GPU Kompakt - Dreistufig





Gitterbasiert GPU Kompakt - Dreistufig

- Zählpass: Wie viele Partikel fallen in jede Zelle (atomicAdd, wegen Nebenläufigkeit)

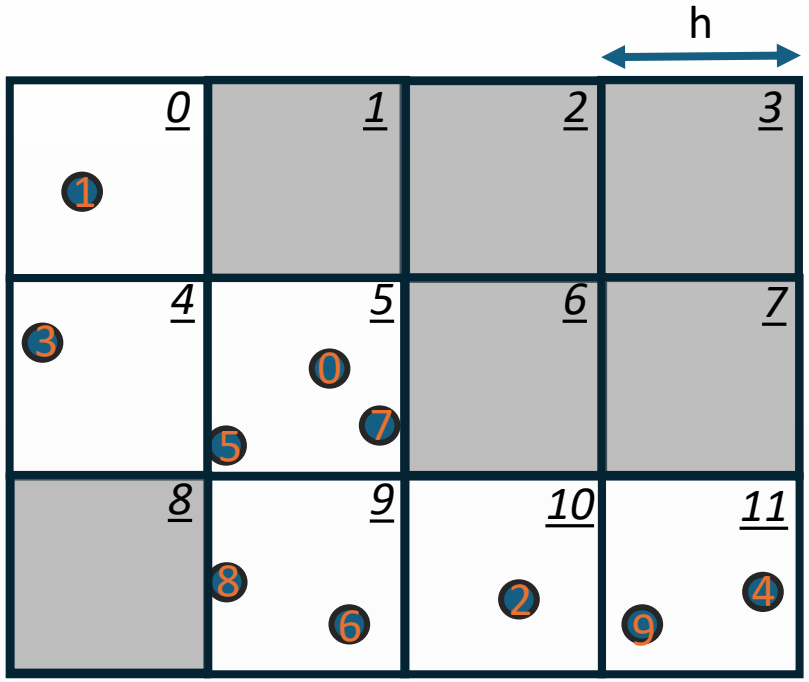
Zellnr.	0	1	2	3	4	5	6	7	8	9	10	11
Anzahl	1	0	0	0	1	3	0	0	0	2	1	2

- Offsets bestimmen: Exklusiver Scan
  - summiert alle vorherigen Elemente, ohne das aktuelle

Offset	0	1	1	1	1	2	5	5	5	5	7	8

  - Offset und Anzahl ergeben Speicherbereiche

Speicher	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Zelle	0	4	5	5	5	9	9	10	11	11



- Beispielzugriff auf Zelle 5: Anzahl[5] = 3, Offset[5] = 2
  -> Speicher[2] bis Speicher[(2+3)] ist Zelle 5
- Beispielzugriff auf Zelle 7: Anzahl[7] = 0 -> Leer

Gitterbasiert GPU Kompakt - Dreistufig

- Zählpass: Wie viele Partikel fallen in jede Zelle (atomicAdd, wegen Nebenläufigkeit)

Zellnr.	0	1	2	3	4	5	6	7	8	9	10	11
Anzahl	1	0	0	0	1	3	0	0	0	2	1	2

- Offsets bestimmen: Exklusiver Scan
  - summiert alle vorherigen Elemente, ohne das aktuelle

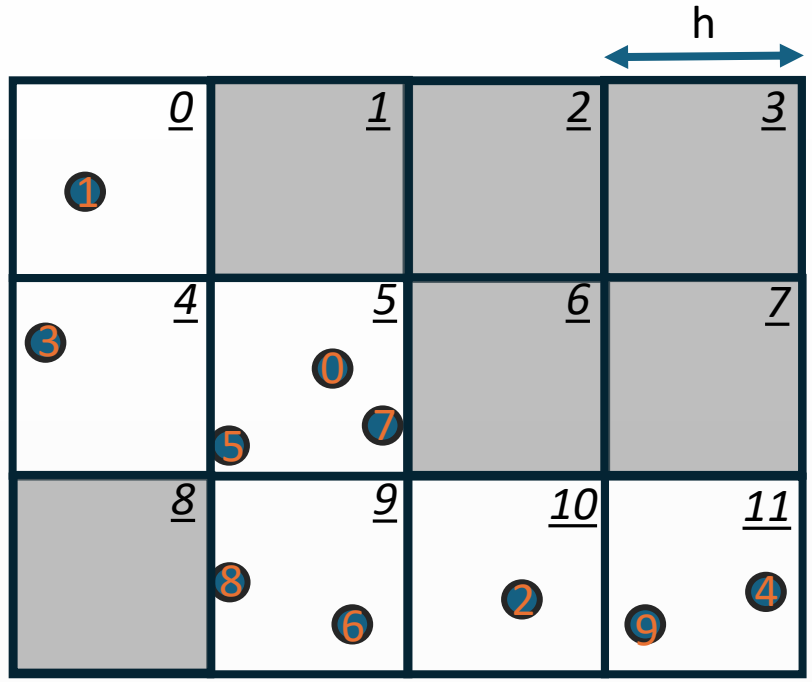
Offset	0	1	1	1	1	2	5	5	5	5	7	8

  - Offset und Anzahl ergeben Speicherbereiche

Speicher	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Zelle	0	4	5	5	5	9	9	10	11	11

- Partikelpass: Einsortieren
  - Weiteres Zählerarray für aktuelle Befüllung
  - Partikel ID eintragen

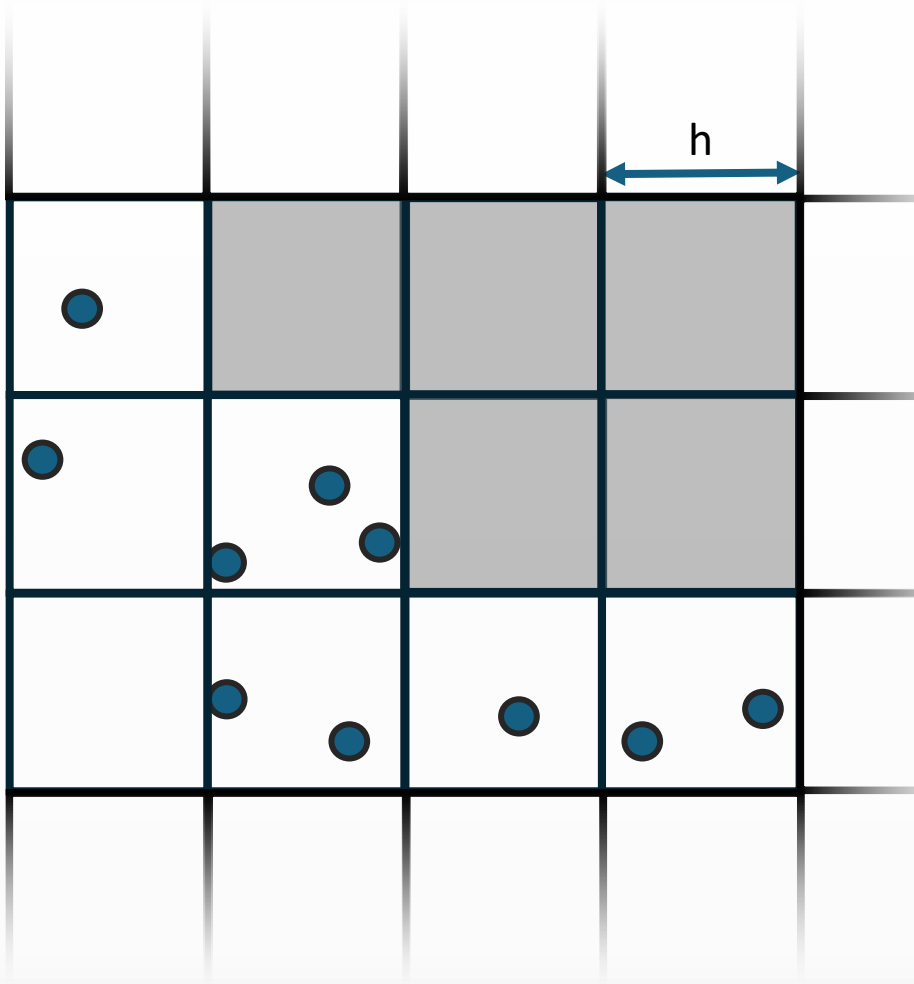
Data	1	3	5	0	7	8	6	2	4	9



Maximale Ausmaße des Grids müssen bekannt sein, 3 zusätzliche Arrays

Open World? Hash-basiert

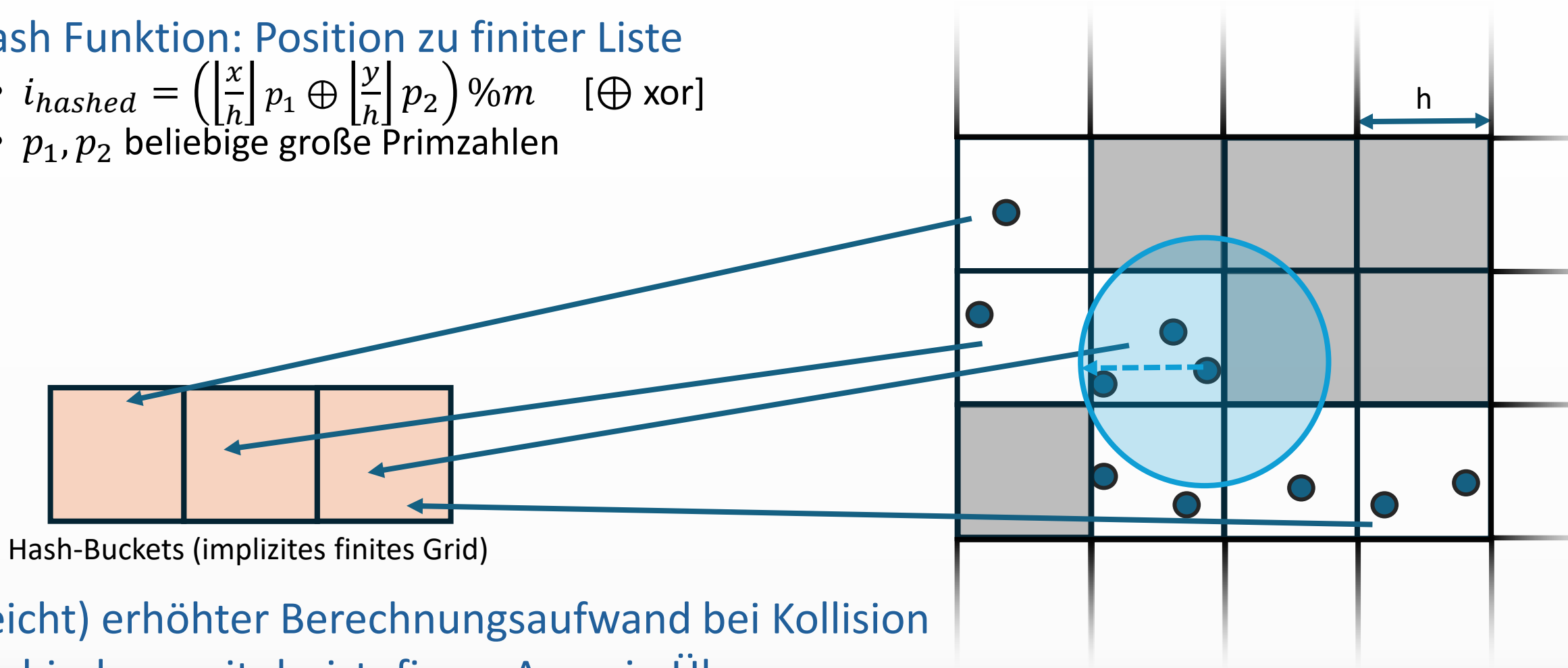
- Hash-basiert erweitern mit Hash-Tabelle mit m Buckets
- Hash Funktion: Position zu finitem Index
  - $i_{hashed} = \left(\left\lfloor \frac{x}{h} \right\rfloor p_1 \oplus \left\lfloor \frac{y}{h} \right\rfloor p_2\right) \% m$ [$\oplus$ xor]
  - p_1, p_2 beliebige große Primzahlen
- Unendliches Grid wird auf finite Einträge reduziert
- Hashkollisionen können ignoriert werden
  - Glättungsfunktion entfernt Partikel mit zu hoher Distanz



Weitere Informationen: [CUDA Tech Report]

Hash-basiert

- Hash Funktion: Position zu finiter Liste
  - $i_{hashed} = \left(\left\lfloor\frac{x}{h}\right\rfloor p_1 \oplus \left\lfloor\frac{y}{h}\right\rfloor p_2\right) \% m$ [$\oplus$ xor]
  - p_1, p_2 beliebige große Primzahlen



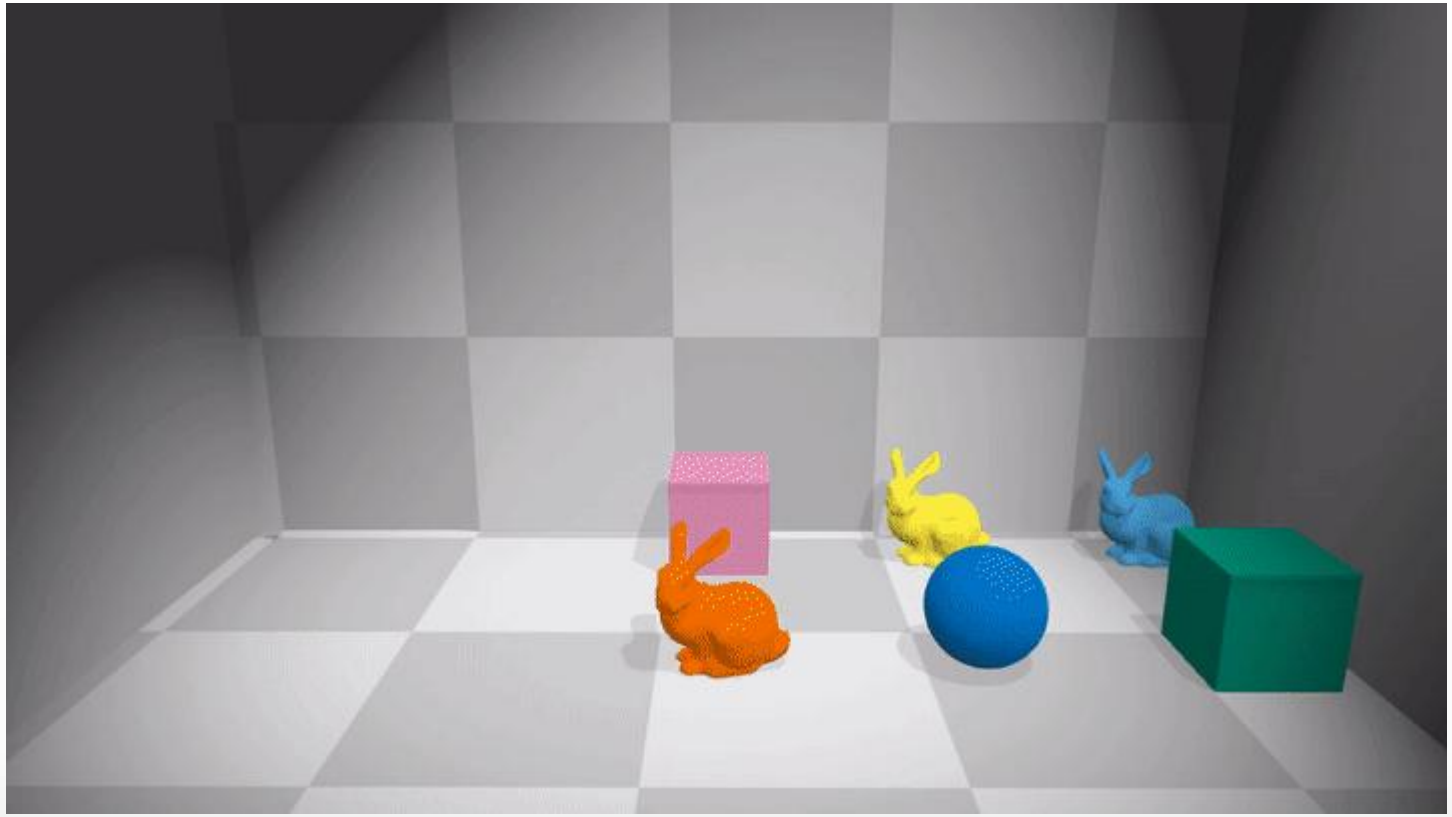
- (leicht) erhöhter Berechnungsaufwand bei Kollision
- Verbindung mit dreistufigem Array in Übung
  - Erweiterung auf 3D

Zusammenfassung

- Nachbarsuche in Particle-based Fluids langsam
  - Komplexität $\mathcal{O}(n^2)$
- Kompakte Glättungsfunktion (Smoothing Kernel) erlaubt Beschleunigung
- Gitterbasiert
- Gitterbasiert, Dreistufig auf GPU
  - Schnell
  - Ausmaße des Grids müssen bekannt sein
- Hashbasierte Erweiterung
  - Unbegrenzt nutzbar
  - Hashkollisionen erhöhen leicht den Berechnungsaufwand

Nächste Woche

- Oberfläche extrahieren
- Lichtbrechung



Material: lfranke.github.io/fluess