Advanced Game Physics

Schnelle Kollisionserkennung

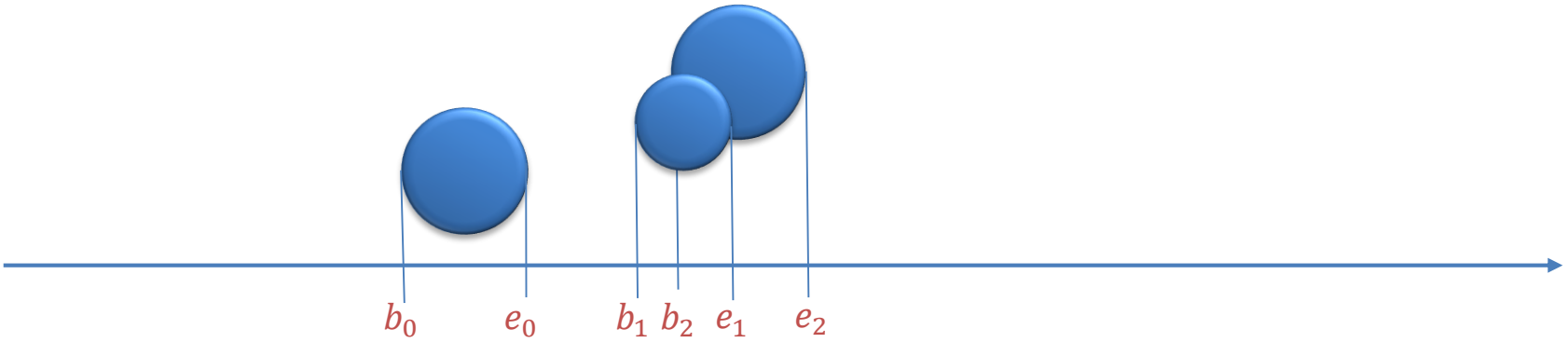
Prof-Dr. Günther Greiner,
Matteo Colaianni M.Sc., Benjamin Keinert M.Sc.
Darius Rückert B.Sc.

Lehrstuhl für Graphische Datenverarbeitung

Wintersemester 2015/16

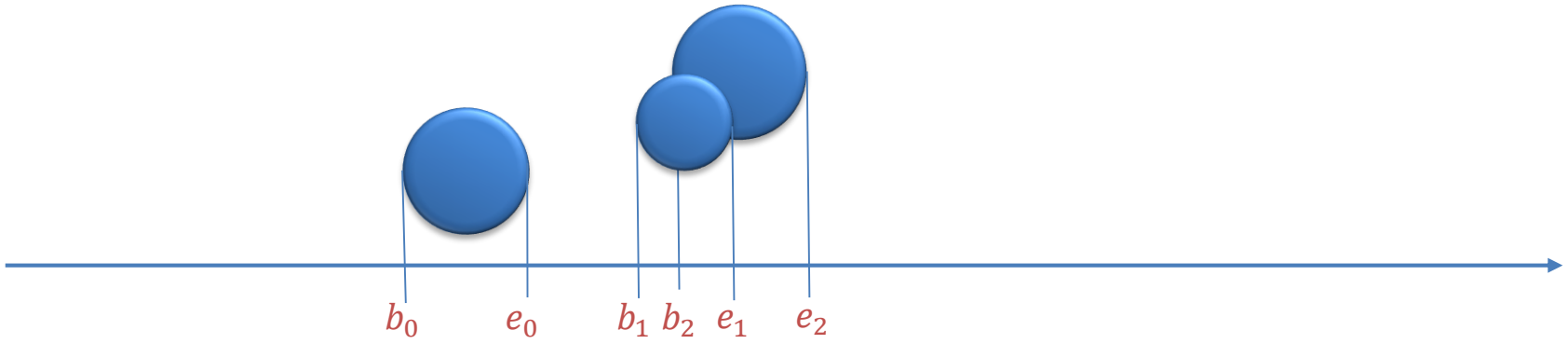
Sort and Sweep

- Projektion der Szene auf eine Achse
- Mittelpunkt projizieren, begin und end jeder Kugel in eine gemeinsame Liste speichern.



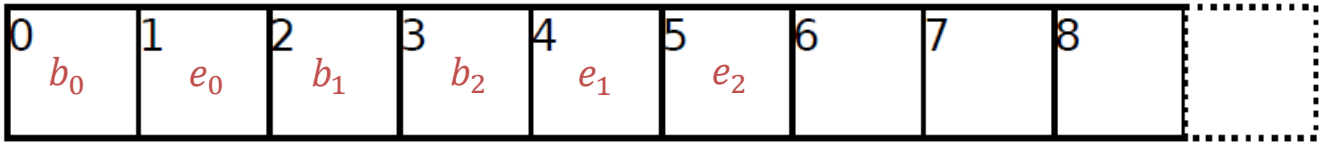
Sort and Sweep

- Projektion der Szene auf eine Achse
- Mittelpunkt projizieren, begin und end jeder Kugel in eine gemeinsame Liste speichern.
- Die Liste nach aufsteigender Koordinate sortieren



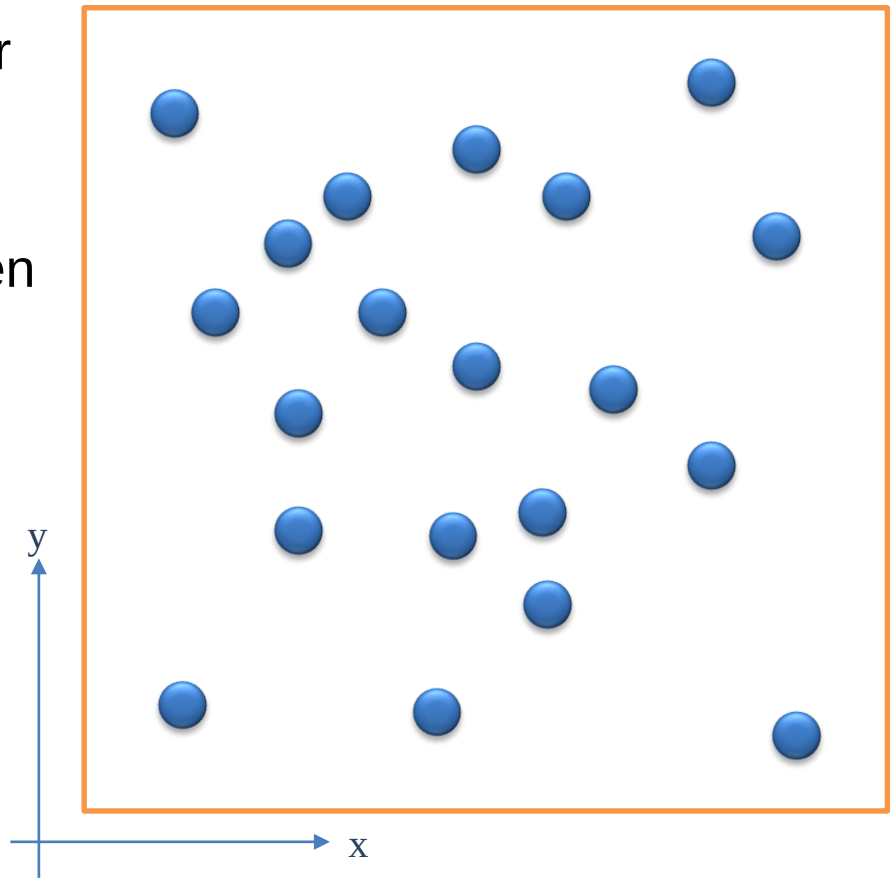
Sort and Sweep

- Pro Listeneintrag einen Thread ausführen
  - Ist der Eintrag ein *end*, terminiert der Thread
  - Ist der Eintrag ein *begin*, geht der Thread die Liste zu seinem zugehörigen end durch und testet für jeden “anderen” *begin* die Kollision



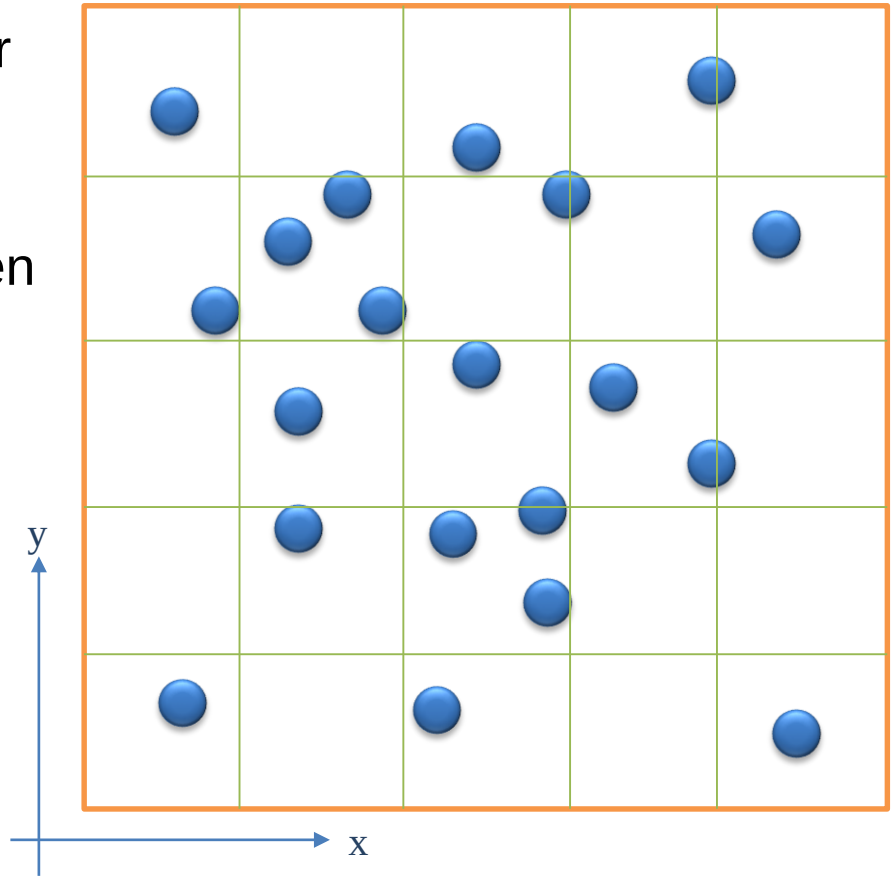
Linked Cell

- Kollisionserkennung innerhalb der Kollisionsdomäne ist $O(n^2)$
- Unmögliche Kollisionen eliminieren



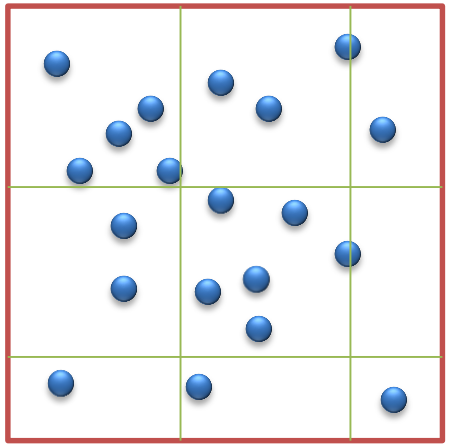
Linked Cell

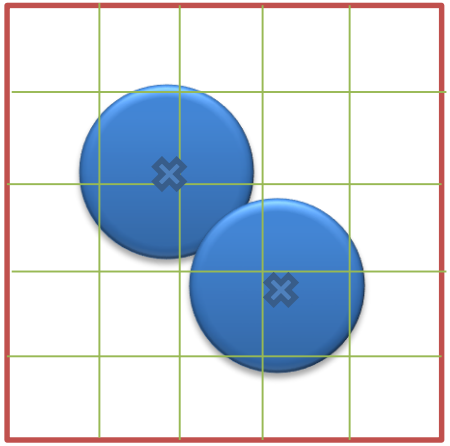
- Kollisionserkennung innerhalb der Kollisionsdomäne ist $O(n^2)$
- Unmögliche Kollisionen eliminieren
- Die Domäne wird durch ein Gitter aufgeteilt



Linked Cell

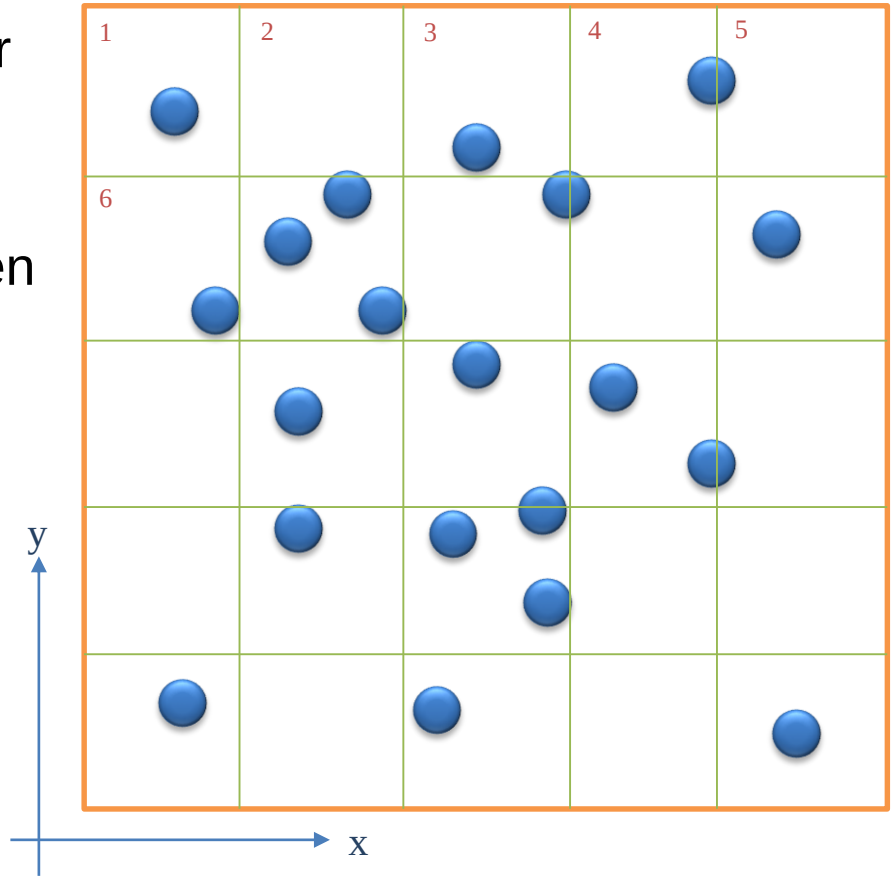
- Zellgröße beeinflusst die Laufzeit
- Zu Groß: Viele Partikel in einer Zelle
- Zu Klein: Partikel „fallen durch“ den Test
- Min. $\Delta_{Cell} \geq r_{min}$
- Alle Nachbarn im Einzug sind in einer 3x3 Zell-Nachbarschaft





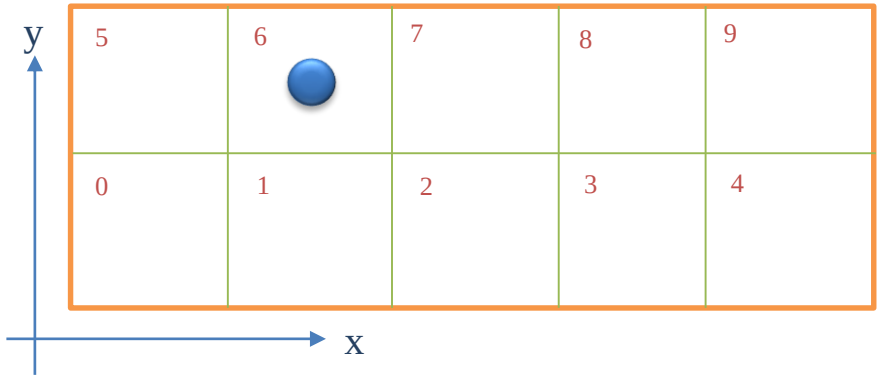
Linked Cell

- Kollisionserkennung innerhalb der Kollisionsdomäne ist $O(n^2)$
- Unmögliche Kollisionen eliminieren
- Die Domäne wird durch ein Gitter aufgeteilt
- Alle Partikel werden in die Zellen einsortiert



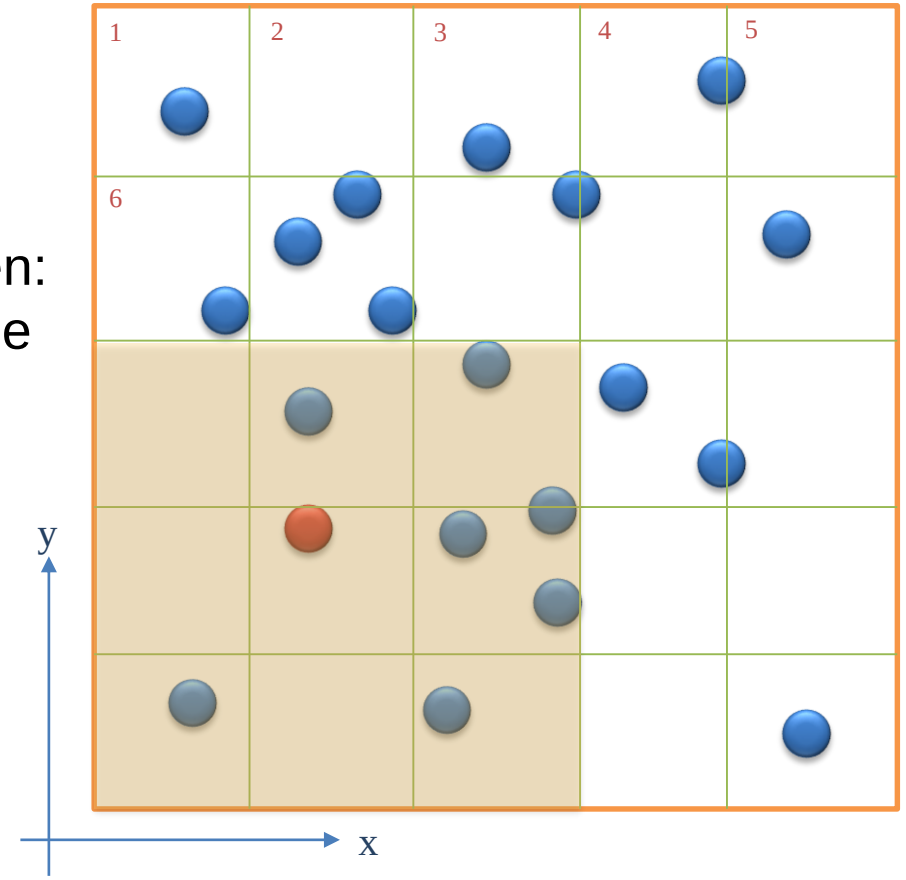
Linked Cell

- Cell-Index muss eine Funktion der Partikelposition sein
- Geometrisch motiviertes Hashing: $\mathbb{R}^d \rightarrow \mathbb{N}_0$
- Einfach: $i = \lfloor y \rfloor \cdot N_x + \lfloor x \rfloor$
- *Beispiel:* $\lfloor 1.5 \rfloor \cdot 5 + \lfloor 1.5 \rfloor = 6$



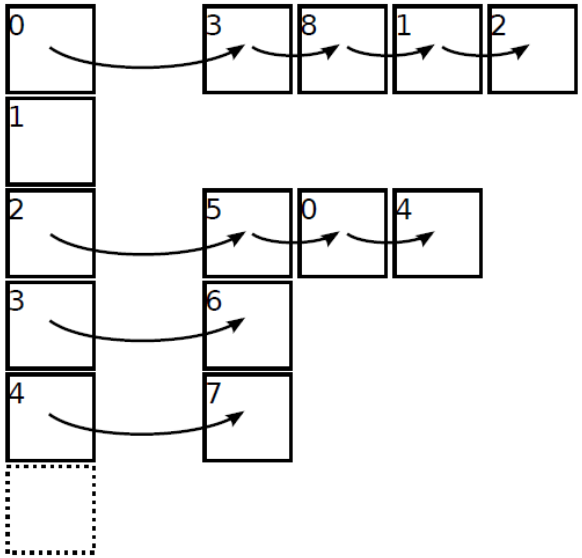
Linked Cell

- Einsortieren aller Partikel in das Gitter ist O(n)
- Generieren potentieller Kollisionen: erneutes nachsehen in der Tabelle
- 3x3(x3) Zell-Nachbarschaft



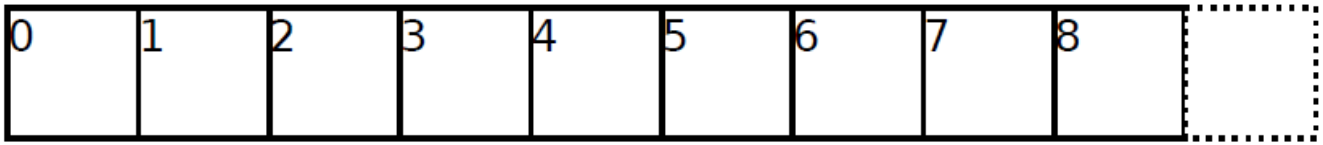
Linked Cell - GPU

- Zellen beinhalten mehrere Partikel
- Darstellung durch eine Verkettete Liste pro Zelle
- Auf der CPU kein Problem!
- Auf der GPU…

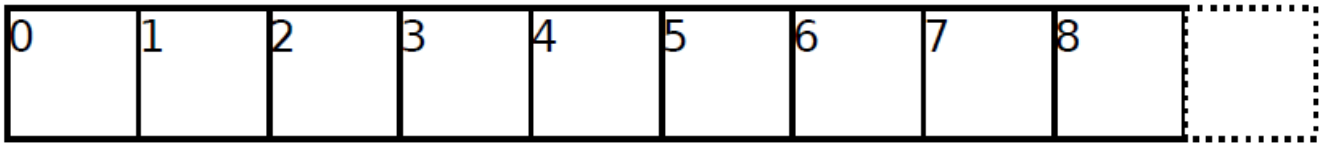


Linked Cell - GPU

Zellen



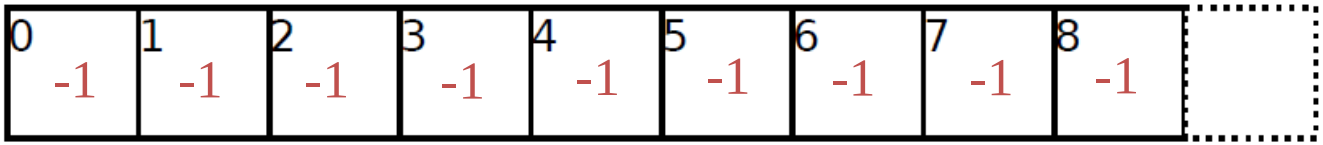
Partikel



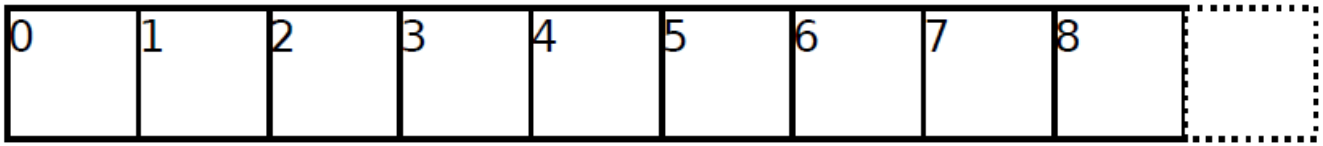
Je ein Speicher für Zellen und Partikel

Linked Cell - GPU

Zellen

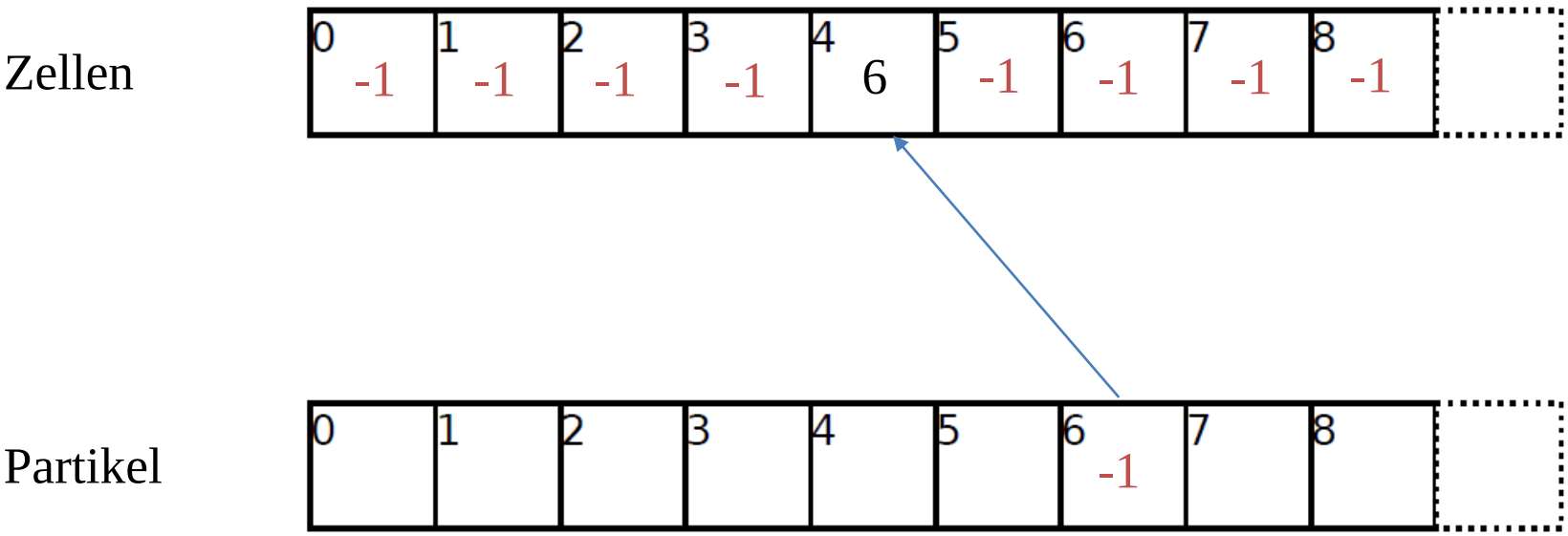


Partikel



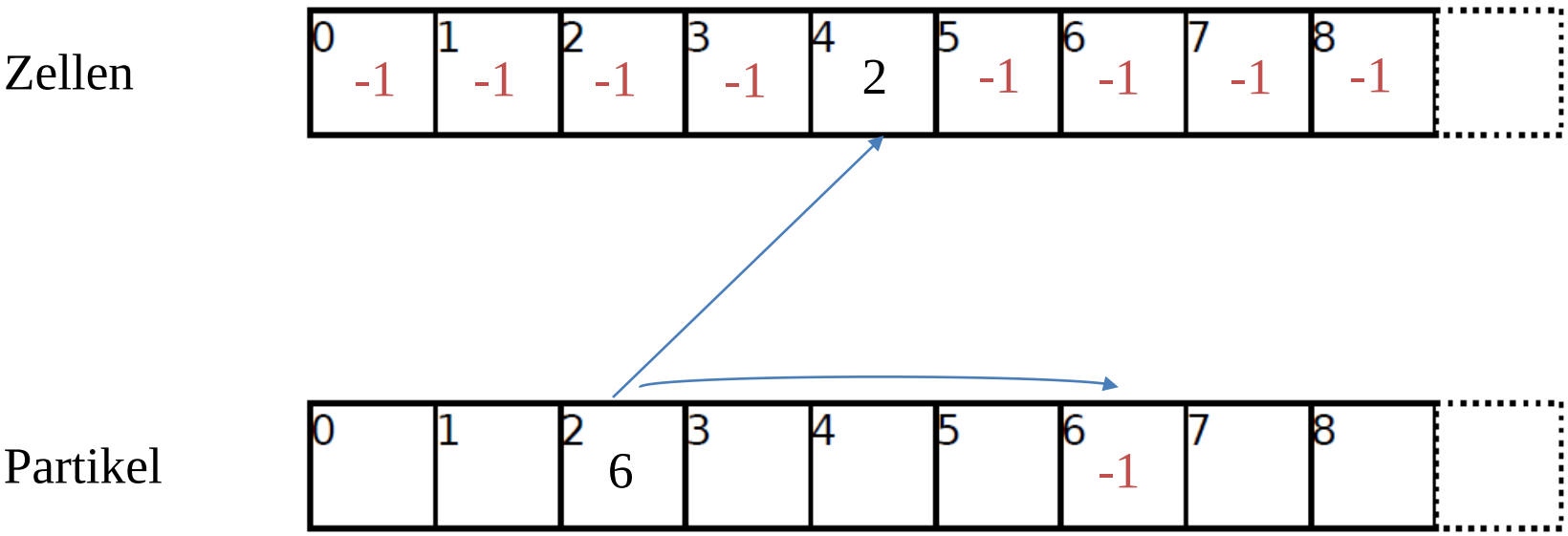
Initialisiere Zell-Speicher mit -1

Linked Cell - GPU



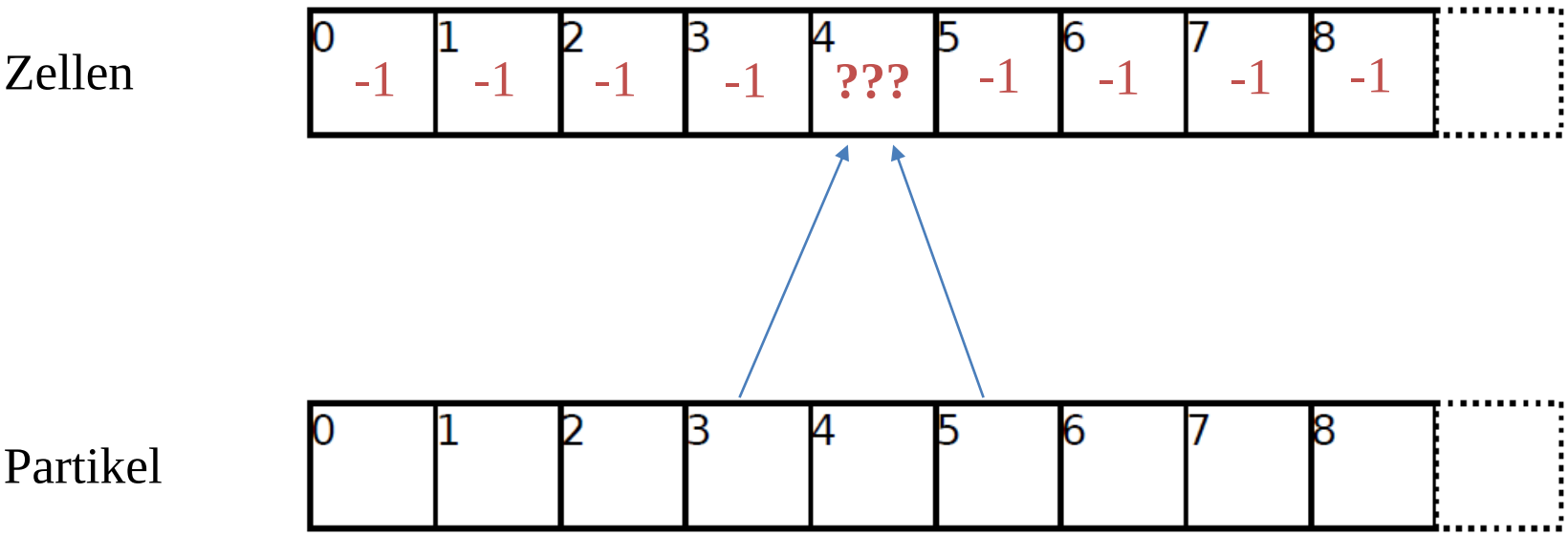
Einfügen eines Partikels in eine Zelle → Einträge tauschen

Linked Cell - GPU



Erneutes Einfügen eines Partikels in diese Zelle → Einträge tauschen

Linked Cell - GPU



Wenn 2 Threads gleichzeitig in eine Zelle einfügen wollen???

Linked Cell - GPU

- Atomic-Operationen verhindern zeitgleichen Zugriff auf Speicher

- atomicAdd: addiert einen Wert auf Speicher atomar
- atomicSub: subtrahiert einen Wert von Speicher atomar
- atomicExch: tauscht atomar einen Wert mit dem Wert im Speicher
- […]

http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#atomic-functions