Dr. Roberto Grosso, Matteo Colaianni, Darius Rückert *Erlangen, 23.11.2015*

Fast Collision Detection (Aufgabenblatt 3)

Aufgabe 1 [3 Punkte] (Timing)

a) Erweitern Sie das Programm um die Möglichkeit, die Zeit, die ein Simulationsschritt benötigt, zu messen. Verwenden Sie hierzu cudaEvents.

b) Geben Sie ungefähr jede Sekunde die Partikelanzahl und die durchschnittliche Zeit eines Schrittes auf der Konsole aus.

c) In Aufgabe 2 und 3 soll jeweils ein neuer Algorithmus zu Kollisionsdetektion implementiert werden. Per Tastendruck soll zwischen diesen zu Laufzeit gewechselt werden können.

- Taste 1: Brute-Force (Aufgabenblatt 2)
- Taste 2: Sort and Sweep (Aufgabe 2)
- Taste 3: Linked Cell (Aufgabe 3)

d) Erstellen sie ein Diagramm, das die Zeitmessung pro Frame für unterschiedliche Partikelanzahlen mit der Brute-Force Kollisionsbehandlung visualisiert.

Aufgabe 2 [8 Punkte] (Sort and Sweep)

Implementieren Sie den Algorithmus Sort and Sweep (Witkin and Baraff 1997), um die Kollisionserkennung zu beschleunigen.

a) Die Szene wird zunächst auf eine Achse projiziert. Jede Kugel wird durch ein Intervall $[b_i, e_i]$ definiert. Erstellen sie parallel eine Liste, welche für jede Kugel ihren Start- und Endpunkt enthält.

b) Nun soll die Liste nach dem Wert auf der Achse aufsteigend sortiert werden. Verwenden Sie hierzu den Sortieralgorithmus von Thrust.

c) Als letztes wird für jeden Eintrag im Array ein Thread ausgeführt. Handelt es sich um einen Endpunkt, terminiert der Thread. Handelt es sich hingegen um einen Startpunkt, geht der Thread so lange das Array entlang, bis der zugehörige Endpunkt gefunden wurde. Alle auftretenden Startpunkte anderer Kugeln lösen eine Kollisionsantwort aus.

d) Erstellen Sie ein Partikel-Laufzeit-Diagramm für die modifizierte Kollisionsbehandlung.

Aufgabe 3 [9 Punkte] (Linked Cell)

In dieser Aufgabe soll der *Linked Cell* Algorithmus implementiert werden.

a) Die Simulationsdomäne soll zunächst durch ein uniformes 3-dimensionales Gitter unterteilt werden. Wählen sie hierzu eine Zellengröße Δ, die mindestens doppelt so groß ist wie der Größte Kugelradius r_{max}.

b) Implementieren Sie die Hashfunktion, welche die Abbildung $\mathbb{R}^3 \rightarrow \mathbb{N}_0$ realisiert. Sortieren Sie nun alle Kugeln in das Gitter ein, indem Sie eine verkettete Liste auf der GPU realisieren (Informationen hierzu finden Sie in den Folien der Vorlesung).

c) In einem letzten Schritt sollen die Kollisionen aufgelöst werden, indem jede Kugel eine $3 \times 3 \times 3$-Zellnachbarschaft untersucht und die Listen traversiert. Achten Sie dabei darauf, dass jede Kollision nur einmal aufgelöst wird.

d) Erstellen Sie auch für diesen Algorithmus ein Partikel-Laufzeit-Diagramm.

Viel Erfolg !

Abgabe: Mittwoch, 6.12.2015, vor 24:00 Uhr.