Programmieren einer Partikelsimulation für kurzreichweitige Interaktionen

Projektbericht

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

Vorgelegt von:	Oliver Heidmann, Benjamin Warnke
E-Mail-Adresse:	adresse@email.de, 4bwarnke@informatik.uni-hamburg.de
Matrikelnummer:	1234567, 6676867
Studiengang:	Informatik
Betreuer:	Philipp Neumann

Hamburg, den 09.03.2017

Abstract

Ziel des dieses Projektes ist die Implementierung einer Partikel-Simulation für kurzreichweitige Partikel-Interaktionen mit Autotuning.

Contents

1 Einleitung

Bei Partikel Simulationen müssen normalerweise die Wechselwirkungen zwischen jedem möglichen Partikelpaar berechnet werden. Die Laufzeit des Programms steigt quadratisch zur Anzahl der Partikel. Dies ist besonders bei hohen Anzahlen von Partikeln kritisch für die Laufzeit. Die in diesem Projekt implementierte Partikel-Simulation ist für kurzreichweitige Interaktionen optimiert. Dadurch lassen sich die Interaktionen zwischen weit auseinanderliegenden Partikeln vernachlässigen, wodurch die Laufzeit kürzer werden kann. Es gibt verschiedene Möglichkeiten, die Interaktionen auf die kurzreichweitige Interaktion zu beschränken um das Programm zu beschleunigen. Das Problem bei diesen verschiedenen Möglichkeiten der Beschleunigung besteht darin, dass je nach Eingabe eine andere Art der Vereinfachung eine bessere Programm-Laufzeit ermöglicht. Die Besonderheit dieser Partikel Simulation liegt darin, dass das Programm zu beginn selbst entscheiden kann, welche Optimierungsstrategie für die gegebene Eingabe am sinnvollsten ist. Hieraus resultiert der Vorteil gegenüber anderen Programmen, dass die Laufzeit für jede beliebige Eingabe besonders schnell ist, und nicht nur wenn die Eingabe passend ist. Dies ist besonders dann spannend, wenn man selbst nicht sicher ist, welches verfahren für die Eingabe am besten geeignet ist, ohne selbst vorher alle Möglichkeiten einmal auszuprobieren.

2 Realisierung

2.1 Vorgehen

planung aufteilung + zeitplan -> realität zeiteinschätzung vs reale zeit unerwartete schwierigkeiten

2.2 Design

klassendiagramm welche klasse tut was für spätere programierer erklären

2.3 Implementierung

beschreiben der benutzung(parameter)

2.4 Analyse

2.4.1 Korrektheit

Um sicherzustellen, dass das Programm korrekte Ausgaben liefert, wurden verschiedene Arten von Test durchgeführt. Zum einen wurden einzelne Komponenten getestet, zum anderen wurde auch das Gesamtverhalten des Programms überprüft.

- **unit-Tests** Um die Korrektheit von den einzelnen Komponenten des Programms zu gewährleisten wurden unit-Tests eingesetzt. Schon beim schreiben der unit-tests wurden viele Fehler gefunden und behoben. Bei späteren Refactoring-Maßnahmen wurden durch diese Testfälle viele neue Fehler verhindert.

- **Energieerhaltung** Aus dem Physikalischen Gesetz 'Aktion gleich Reaktion' ergibt sich, dass die Energie in einem geschlossenen System immer erhalten belieben muss. Eingebaute Funktionen im Programm ermöglichen die Ausgabe der aktuellen Energie im System, sodass leicht überprüft werden kann, ob die Energie in einem Akzeptablen Rahmen beleibt. Durch Rechenungenauigkeiten ist es sehr unwahrscheinlich, dass die Energie exakt gleich beleibt.

- **Visualisierung** Durch Visualisieren der Ausgabedaten wurde auch das fertige Ergebnis stichprobenartig überprüft. Bei den getesteten Parametern verhielt sich das Programm wie erwartet.

2.4.2 Laufzeit

Der Fokus dieses Projektes liegt auf dem auto-tuning, wodurch auch das Laufzeitverhalten des Programms genau analysiert werden muss. Die Laufzeit des Programms, und somit die Wahl der besten Internen Datenrepräsentation hängt von verschiedenen Parametern ab.

- **Dichte**
- **Volumen**
- **Anzahl**
- **cut-off**
- **cut-off-factor**
- **Startgeschwindigkeit**
- **Startanordnung**

3 Zusammenfassung

4 Literatur

5 Anhang

Verwendete Bibliotheken und Programme zum ausführen des Programms

- Boost
  - unit-tests
- CMake
- Make
- clang-format
- paraview
  - Visualisierung der Ausgabe
- lcov
  - Testabdeckung ermitteln und visualisieren
- slurm
  - Messtabellen berechnen
- doxygen
  - Dokumentation des Quelltextes
- latex
  - dieses Dokument
  - Präsentationen