

Übung 3

Dipl.-Inf. Domenic Jenz
(jenz@hirs.de)

HLRS, Universität Stuttgart

Übungen zur Vorlesung Molekulardynamik und Lattice
Boltzmann Methoden

Outline

Ein kleines MD Programm

Einführung

Das Grundgerüst

Particle

Domain

Velocity-Störmer-Verlet

Ausprobieren 1

Lennard-Jones Potential

Ausprobieren 2



Übungstermine (geplant)

25.11.2008

9.12.2008

13.1.2009

27.1.2009

10.2.2009



Einführung

- ▶ Im folgenden werdet Ihr ein kleines MD Programm basteln.
- ▶ Ein Grundgerüst ist gegeben auf der Seite <http://www.hlrs.de/people/jenz> unter **uebung3.tar.bz2**
- ▶ An den grundlegenden Algorithmen dürft ihr euch in den nächsten Übungen versuchen.
- ▶ Das wäre diese Woche:
 - ▶ Velocity-Störmer-Verlet-Verfahren
 - ▶ Lennard-Jones Potential
- ▶ Wir beginnen natürlich mit einer sequentiellen Variante !



- ▶ Der Code ist wie folgt aufgeteilt:
 - ▶ **Hauptprogramm**: `smd.cpp`
Nimmt die Kommandozeilenparameter entgegen und startet die Simulation.
 - ▶ **Partikel Klasse**: `Particle.cpp` und `Particle.h`
kapselt die Eigenschaften eines Partikels.
 - ▶ **Domain Klasse**: `Domain.cpp` und `Domain.h` verwaltet das Simulationsgebiet und stellt Funktionalität zum Speichern und Einlesen zur Verfügung.
 - ▶ **Komponenten Klasse**: `Component.cpp` und `Component.h` enthält die Informationen einer Komponente, wie z.B. ε und σ
- ▶ Eure Hauptarbeit findet in **`domain.cpp`** statt.
- ▶ in **`config.h`** ist DIM momentan mit 2 vordefiniert (wir bleiben also erst mal in 2D)



Particle

Die Klasse **Particle** repräsentiert ein Partikel u.a. mit:

- ▶ double mass: Masse des Partikels
- ▶ double[DIM] vel: Geschwindigkeit
- ▶ double[DIM] pos: Position
- ▶ double[DIM] force: Kraft, die auf das Partikel wirkt
- ▶ double sigma : σ für LJ-Potential
- ▶ double eps: ε für LJ-Potential
- ▶ void zeroForce (): Funktion, welche die Kraft auf 0 setzt
- ▶ double addForce(int index, double f): addiert f zur index-Kraftkomponente

Diese Daten sind über diverse getter-Methoden erreichbar und (teilweise) über setter-Methoden veränderbar.



Domain

Wichtig sind hier:

- ▶ Particle* particles: das Array mit allen Partikeln
- ▶ double dt : Zeitschrittweite δt
- ▶ int actStep : aktuelle Schritt Nummer
- ▶ double getParticleNumber (): Funktion, welche die Gesamtzahl an Partikeln zurückgibt
- ▶ void forceGravitation (): Kraftberechnungen anhand des Gravitationspotentials

$$F_{ij} = \frac{m_i m_j}{d_{ij}^3} \cdot (x_j - x_i)$$

- ▶ void updatePreF(), void updatePostF(): Berechnungen vor und nach Kraftberechnung

Velocity-Störmer-Verlet

Das Velocity-Störmer-Verlet Verfahren sieht wie folgt aus:

$$\vec{x}(t + \Delta t) = \vec{x}(t) + \Delta t \vec{v}(t) + \frac{\vec{F}(t) \cdot \Delta t^2}{2m_i} \quad (1)$$

$$\vec{v}(t + \Delta t) = \vec{v}(t) + \frac{(\vec{F}(t) + \vec{F}(t + \Delta t))\Delta t}{2m_i} \quad (2)$$

Gleichung 1 lässt sich komplett vor den Kraftberechnungen implementieren (also Kandidat für `updatePreF()`), während Gleichung 2 auch aktuelle Kraftwerte benötigt und somit am Besten aufgespalten wird in: vor und nach Kraftberechnung. (ein Teil in `updatePreF()` und der andere in `updatePostF()`)



Ausprobieren 1a

Nachdem ihr **updatePreF** und **updatePostF** implementiert habt (Zur Kraftberechnung wird vorerst **forceGravitation** benutzt):

- ▶ Kompilieren mit **make** im Quellcodeverzeichnis
- ▶ Führt **./smd planets.smd 10000 planetsOut 100** aus.
- ▶ Startet **dx** und betet, dass es auf der Maschine vor euch installiert und nicht völlig verhunzt ist.
- ▶ Wählt "Run Visual Programs" an und wählt im Unterverzeichnis **OpenDX** eures Quellcodeverzeichnisses die Datei **sad.net** aus.
- ▶ Jetzt sollte ein **sad Control Panel** aufgegangen sein, wo man unter "Import file name" nun **planetsOut.series.dx** auswählt.



Ausprobieren 1b

Auf den VisGS Rechnern geht es anders:

- ▶ Auf einer Konsole gebt ihr folgendes ein:
 - ▶ bash
 - ▶ export LD_LIBRARY_PATH=/opt/dx/lib:./
 - ▶ /opt/dx/bin/dx -uionly &
 - ▶ /opt/dx/bin/dx -execonly &
- ▶ Auf **Connection->Start Server gehen** und dann im neuen Fenster unter **Options** das Feld **Connect to already running server** anwählen , das Optionsmenü wieder verlassen und **Connect** drücken.
- ▶ Jetzt kann man mit **File->Open Program** die Datei **sad.net** öffnen.
- ▶ **Windows->Open All Control Panels** auswählen und dann im Control Panel unter "Import file name" nun **planetsOut.series.dx** angeben



Ausprobieren 1c

Sollte jetzt nichts zu sehen sein ist wohl die Kamera verstellt.
Zum Einstellen:

- ▶ Im Visualisierungsfenster auf **Options->View Control** gehen
- ▶ **Mode** auf **Camera** stellen
- ▶ **To** auf (0,0,0) setzen
- ▶ **From** auswählen und auf (0,0,100) setzen
- ▶ **Up** auswählen und auf (0,1,0) setzen

Danach noch einmal **Execute->Execute Once** ausführen.



Lennard-Jones Potential

Ein sehr beliebtes Potential (für ungeladene, nicht chemisch aneinander gebundenen Atome) ist das Lennard-Jones-12-6 Potential:

$$U(r_{ij}) = 4 \cdot \varepsilon \left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\left(\frac{\sigma}{r_{ij}} \right)^6 - 1 \right) \quad (3)$$

Die resultierende Kraft also zwischen Molekül i und j ist:

$$\vec{F}_{ij} = 24 \cdot \varepsilon \cdot \frac{1}{r_{ij}^2} \cdot \left(\frac{\sigma}{r_{ij}} \right)^6 \cdot \left(1 - 2 \cdot \left(\frac{\sigma}{r_{ij}} \right)^6 \right) \cdot (\vec{x}_j - \vec{x}_i) \quad (4)$$

Hierbei war r_{ij} die Distanz zwischen den beiden Molekülen



Ausprobieren 2a

- ▶ Implementiert mit der Funktion **forceLJ()** in `domain.cpp` das Lennard-Jones-12-6-Potential und ersetzt in **oneStep** das Gravitationspotential durch eure neue Funktion
- ▶ Diesmal lasst es mit `./smd collision.smd 10000 collOut 100` laufen
- ▶ Habt Ihr "Actio = Reactio" schon berücksichtigt ?
- ▶ Eine weitere Optimierung ist ein Cut-Off Radius von 2.5σ
- ▶ Letztendlich können da sicher auch noch Schleifen mit OpenMP parallelisiert werden



Ausprobieren 2b - Online Visualisierung

Statt nur die Sequenz am Ende abzuspielen, bietet es sich an auch schon während der Simulation den aktuellen Zustand zu visualisieren.

- ▶ Auf einer Konsole gebt ihr folgendes ein:
 - ▶ `/opt/dx/bin/dx -uionly -mdf simpleImport.mdf &`
 - ▶ `/opt/dx/bin/dx -execonly -mdf simpleImport.mdf &`
- ▶ Connect wie vorher auch
- ▶ wählt das Programm **onlineVis.net** aus
- ▶ mit **Execute->Execute on Change** sollte dann alle 10s das Bild aktualisiert werden.

