Institut für Informatik — TU München WS 2010/11
Scientific **C**omputing in **C**omputer **S**cience
Prof. Dr. H.-J. Bungartz
Dipl.Inf. T. Auckenthaler, Dipl.Inf. W. Eckhardt

PSE Molekulardynamik
Sheet 4: Thermostats, Rayleigh-Taylor-Instability and "falling drops"

Exercise at 15th December 2010

Task 1 „Thermostats "

In following simulations we will expose the molecules to gravity or heat. That external influences would lead to an increase in the kinetic energy of the system simulated and thus to a rising temperature.
In order to keep the simulated material at a constant temperature, as well as to be able to simulate heating or cooling processes, it is neccessary to have a thermostat.
The temperature of the phase is determined by the velocity of the particles. The dependency is given by

$$E_{kin} = \frac{\texttt{dim}N}{2} k_B T$$

or dimensionless:

$$E_{kin} = \frac{\texttt{dim}N}{2} T$$

where E_{kin} can be calculated as the sum of the kinetic energy of all particles:

$$E_{kin} = \sum_{pariclesi} \frac{1}{2} m v_i^2$$

As seen in the tutorial, the scaling factor can be calculated as:

$$\beta_\gamma = \sqrt{\frac{E^D}{E(t)}}$$

- Implement such a thermostat.
- The input parameters are
  - initial temperature:
    the initial scaling value for the velocity for the Maxwell-Boltzmann distribution is calculated as:

    $$v = \sqrt{\frac{2E_{kin}}{\texttt{dim}Nm}}$$

    The initialization of the phase to a given temperature may be moved to the thermostat and be called after all particles have been instantiated.
    For our applications (at least for the moment) it is ok to set $m = 1$ and hardcode it.
    However, the initialization with the Brownian Motion should be optional.
  - the number of timesteps after which the thermostat is applied
- Furthermore for simulations with a different target temperature:
  - target temperature
  - temperature difference, i.e. the step size in which the temperature should be changed
  - the number of timesteps after which the temperature has to be changed
- Adapt the XML and tests accordingly.

Task 2 „Simulation of the Rayleigh-Taylor instability "

The Rayleigh-Taylor instability occurs at the interface of two fluids when a fluid with higher density resides on a fluid with lower density and is subject to gravity.
In this task we will perform the simulation of a Rayleigh-Taylor instability. We will impose reflecting boundary conditions on the horizontal boundaries of the domain and periodic boundary conditions on the vertical boundaries.
The two different fluids are represented by two molecular cuboids.

- Implement periodic boundaries:
  - particles which left the domain at a boundary have to be inserted at the opposite boundary
  - particles from the boundary layer of the opposite boundary have to be inserted into the halo layer of the boundary.
- The fluids are exposed to gravity. Implement a force calculation routine which adds a gravitational force G (along the y-axis) according to

  $$G = m \cdot g_{grav}$$

  It should be possible to specify the factor g_{grav}.

- We have to model two different fluids. Extend the molecules so they can be of different type. Especially they may have different
  - mass
  - Lennard-Jones Parameters ϵ and σ

  For two molecules of different type the Lennard-Jones parameters have to be determined according to the Lorentz-Berthelot mixing rule:
  $\sigma_{12} = \sigma_{21} = \frac{\sigma_{11}+\sigma_{22}}{2}$ $\epsilon_{12} = \epsilon_{21} = \sqrt{\epsilon_{11}\epsilon_{22}}$

- For testing purpose, perform a small experiment:

  $g_{grav} = -12.44$ | delta_t $= 0.0005$ | $t_{end} = 25$
  Size of domain L = 63, 36 | initial temperature $T_{init} = 40$ | $n_{thermostat} = 1000$
  $r_{cutoff} = 2.5\sigma$

  Liquid 1:
  $m_1 = 1.0$ | $\epsilon_1 = 1.0$ | $\sigma_1 = 1.0$
  $x_1 = \{0.6; 2\}$ | $v_1 = \{0; 0\}$ | $n_1 = \{50; 14\}$
  $h_1 = 1.2$

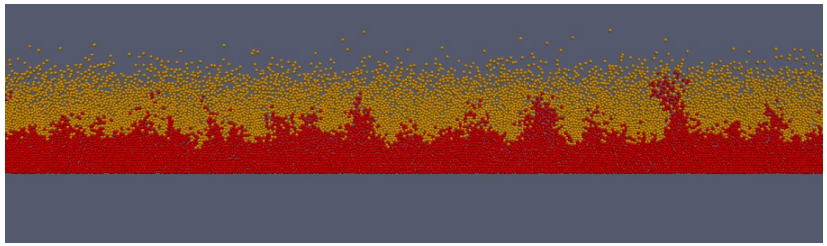
  Liquid 2:
  $m_2 = 2.0$ | $\epsilon_2 = 1.0$ | $\sigma_2 = 0.9412$
  $x_2 = \{0.6; 19\}$ | $v_2 = \{0; 0\}$ | $n_2 = \{50; 14\}$
  $h_2 = 1.2$

- Perform the bigger experiment:

  $g_{grav} = -12.44$ | delta_t $= 0.0005$ | $t_{end} = 50$
  Size of domain L = 300, 54 | initial temperature $T_{init} = 40$ | $n_{thermostat} = 1000$
  $r_{cutoff} = 2.5\sigma$

  Liquid 1:
  $m_1 = 1.0$ | $\epsilon_1 = 1.0$ | $\sigma_1 = 1.2$
  $x_1 = \{0.6; 2\}$ | $v_1 = \{0; 0\}$ | $n_1 = \{250; 20\}$
  $h_1 = 1.2$

  Liquid 2:
  $m_2 = 2.0$ | $\epsilon_2 = 1.0$ | $\sigma_2 = 1.1$
  $x_2 = \{0.6; 27\}$ | $v_2 = \{0; 0\}$ | $n_2 = \{250; 20\}$
  $h_2 = 1.2$



Task 3 „Simulation of a falling drop “

In this task we want to simulate a drop falling into a bassin filled with liquid.

- Write an output component which saves the phase space to a text file, in a similar format like the input file for the simulation of the solar system. Similarily, the output file will be used as the input for a new simulation.

  Note: Here you should to save the complete state of the molecules to the file (including old force, type, and so on...)

- Initially, we let gravity act on the fluid in the bassin until $t = 15$ so that the molecules arrange in a natural way (equilibration). Then the phase space has to be saved to a file which we use as input for the further simulation.

  Use the following parameters:

  $g_{grav} = -12.44$ | delta_t $= 0.0005$ | $t_{end} = 40$
  Size of domain L = 303, 180 | initial temperature $T_{init} = 40$ | $n_{thermostat} = 1000$
  $r_{cutoff} = 2.5\sigma$

  Liquid:
  $m_1 = 1.0$ | $\epsilon_1 = 1.0$ | $\sigma_1 = 1.2$
  $x_1 = \{1.5; 2\}$ | $v_1 = \{0; 0\}$ | $n_1 = \{250; 50\}$
  $h_1 = 1.2$

- Now perform the actual simulation. Basically, we use the equilibrated fluid, a drop (represented by a sphere), and let gravity accellerate the drop.

  Use the parameters and the output of the above simulation and additionally specify a sphere with: $x_1 = \{150; 150\}$ $n_1 = 20$

Task 4 „Performance Measurement “

As simulations are pretty long-running, we have to

- Change the application so that you can measure the runtime of an iteration (take the avarage of several iterations).

  Note: Use the function gettimeofday()

- Create a profile of your application, e.g. with gprof. Use a large, representative scenario ($\geq$ 10000 molecules), with io switched off. Which parts of the code consume the most runtime?
- Think about how you could improve the performance of your code, especially of the compute intensive parts (consider features of the c++-language, e.g. inlining, templates, as well as algorithmic features (e.g. reuse calculated values, avoid unnecessary square-roots, ...)

Good luck!

Deliver by 11th January 2011, 12 pm per mail to `eckhardw@in.tum.de` and `auckenth@in.tum.de`!