Simulation of Massive Particle Systems with Lennard-Jones Force on the GPU

Bastian Hassel

I. Introduction

THE Lennard-Jones potential is a model to approximate the interaction between a pair of neutral atoms or molecules. First proposed was a form of the potential in 1924 by John Lennard-Jones [1]. A common expression of the potential is

$$\mathbf{v}_{LJ} = 4\varepsilon \left[\left(\frac{\delta}{r}\right)^{12} - \left(\frac{\delta}{r}\right)^{6} \right] \tag{1}$$

In order to simulate the forces of the Lennard-Jones potential they need to be calculated for all particles against ever other particle in the system, with just $256k$ particles 65.5 billion calculations are needed, resulting in long computation times. Therefore the main focus in this article is to reduce the time complexity of $O(n^2)$ towards the best case $O(n)$.
Particles too far away to influence the current particle can be sorted out to reduce overall workload. The best way to achieve this is by dividing the space into grid cells, that way any particle need only to be calculated against their 9 adjacent cells. This also defines the size of the cells to the range at which particles could influence each other.

II. Implementation

A. Sorting

In order to place particles into their corresponding cell they must be sorted. All particles inside a particular cell do not need to be sorted any further, their order does not matter. Thus the sorting can be sped up a little by keys for their positions, so all particles that belong to the same cell get the same key. Also to make comparisons on the GPU faster the key size should be limited to 4 bytes, the size of an integer. To make byte alignment easier the 3 axes x,y,z are stored in 3 bytes. This results in a limitation to 256 possible values for each axis.
With these limitations the key value can be calculated by dividing the particle position by the cell size and using the dot product to align the bytes.

$$\mathbf{v}_{key} = \left\lfloor \frac{\vec{\mathbf{x}}_{particle}}{\vec{\mathbf{s}}_{cellsize}} \right\rfloor \cdot \begin{pmatrix} 65536 \\ 256 \\ 1 \end{pmatrix} \tag{2}$$

Since the particles need to be assigned with their key it is necessary to save the key-value pair. The index of the particle is sufficient to ensure that actually the particle is sorted and not just its key alone.
Once the key is calculated the sorting can start. Because the particle data is on the GPU and copying the data to the CPU and back causes too much latency an efficient sorting algorithm for the GPU is needed. As this question is already answered a bitonic sorter is a good solution for smaller particle counts and radix sort is generally a good solution [2], especially since the key value will not be larger than 3 bytes in this simulation. Especially to improve the speed of a radix sort it is necessary to reduce the key size to the bare minimum.

B. Grid building

With a sorted particle list the grid for the simulation can be build. Therefore every grid cell gets 2 values. The first one is the start index and the other one the end index of the particles in the cell. Since all particles are sorted it is ensured that indices in between belong to particles in the same cell.

C. Simulating

For simulation itself every particle still has to be processed but the workload is much smaller due to the preparatory work. First the corresponding grid cell must be calculated, then all adjacent cells are needed, so they have to be included too. All particles that affect the current one can then easily be acquired by iterating from the start index to the end index for all cells that were gathered in the first step. For the force itself only the distance between 2 affecting particles has to be calculated and passed

into the function mentioned in the introduction. All forces then need to be accumulated and then converted into an acceleration.
The final step consist than by applying the acceleration to the velocity and moving the position by the velocity.

$$\mathbf{v}^{t+1} = \mathbf{v}^t + \mathbf{a}^t \cdot \delta t \tag{3}$$

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \mathbf{v}^t \cdot \delta t \tag{4}$$

To keep all particles closer together applying forces from arbitrary walls would be wise. The alternate, more accurate method would be a torus-topology. Instead of reflecting particles beyond the walls is to replace them instant to the other side of the system:

$$\mathbf{x}^{'} = \mathbf{x} + \mathbf{w}.xyz \cdot (2\mathbf{w}.w) \tag{5}$$

But that alone does not allow to interact with particles on the other side, to achieve this the modulo operator can be used instead of clamping the grid cell indices between 0 and their maximum. The position of particles in those cells that are actually on the other side needs to be adjusted to represent their position for the force calculation.

III. Results

Particles will generally form cluster, greater attraction will cause smaller and denser cluster, greater repulsion will cause them to spread more widely while also resulting in affected particles having lower velocities resulting from lesser particles being very close.

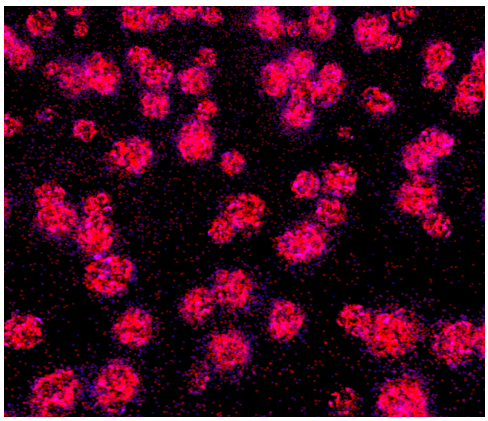

Fig. 1. Simulation Results.

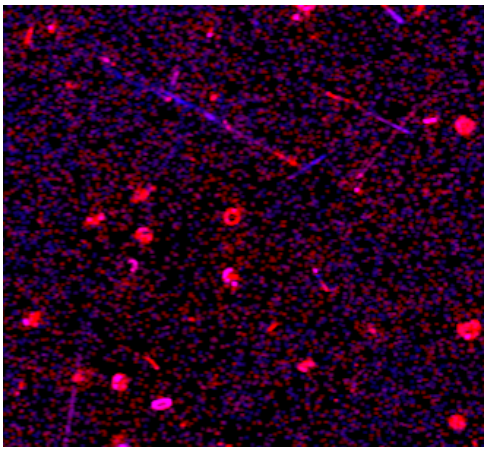

Fig. 2. Simulation Results.

Although computing could be sped up sorting the particles into cells is also created additional artefacts. So with bigger time steps the formed cluster tend to move towards the center of the grid cells. Also particles vibrate more the bigger the time steps are instead of staying in place.

References

[1] J. E. Jones, "On the determination of molecular fields. ii. from the equation of state of a gas," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 106, no. 738, pp. 463–477, Oct 1924. [Online]. Available: http://dx.doi.org/10.1098/rspa.1924.0082

[2] N. Satish, M. Harris, and M. Garland, "Designing efficient sorting algorithms for manycore gpus," in *Proceedings of the 2009 IEEE International Parallel & Distributed Processing Symposium*. Institute of Electrical and Electronics Engineers, May 2009, pp. 1–10. [Online]. Available: http://dx.doi.org/10.1109/IPDPS.2009.5161005