Asychronous Smoothed Particle Hydrodynamics

ANDREW PREGENT

ACM Reference Format:
Andrew Pregent. 2022. Asychronous Smoothed Particle Hydrodynamics. 1, 1 (November 2022), 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 ABSTRACT

In this paper, a new method for parallelizing smooth-particle hydrodynamics using CUDA on the GPU will be presented, building upon asychronous SPH. The method will then be evaluated for performance against single and multi-core implementations, and its accuracy will be compared to a commercial simulation package.

2 INTRODUCTION

Particle-based fluid simulations are a difficult problem to parallelize efficiently. Similar to the notorious N-Body problem, particles in CFD simulations may potentially interact with any other particle in the system, at any point in time. The random access which this requires is particularly unfriendly to GPUs, which are typically optimized for streamed memory. To make matters worse, to obtain any meaningful results requires not only a very large number of particles (typically on the order of millions) but the calculations must be done at extremely small time-scales in order to ensure the stability of the system. To this end, a large amount of research has been done on attempting to reduce the computational time required.

3 LITERATURE REVIEW

Computational Fluid Dynamics is predominantly concerned with the integration of the Navier-Stokes equations through time. These equations rely on the continuum hypothesis of a fluid, which is the assumption that despite the true nature of fluids being a result of their molecular interactions, the macroscopic behaviours of a fluid can be predicted with sufficient accuracy with smooth vector fields.

Fluid simulation can be broadly grouped into two categories. Eulerian simulations observe physical quantities which move past fixed locations in space. The use of regular grids is often employed, however some methods also use meshes instead. Lagrangian simulations track the movement of physical quantities through the flow of the fluid. There also exist hybrid approaches which attempt to make use of both paradigms.

Smoothed Particle Hydrodynamics (SPH) is a Lagrangian simulation method first proposed independently by L.B. Lucy[12] and R.A. Gingold and J.J. Monaghan[4] for simulations in astrophysics. J.J. Monaghan later extended the method to free surface flows[13]. Refer to Ting Ye, et al. for a modern review of various advances in the method.

Mathieu Desbrun and Marie-Paule Gascuel applied the Courant-Friedrichs-Lewy criterion to SPH, providing an upper bound on the time step based on the kernel support size and the maximum particle velocity[3]. This means that in practice the time step must often be very small in order for the simulation to remain stable. Predictive Corrective Incompressive SPH (PCISPH) attempts to address this problem for incompressible fluids such as water, where the problem is exacerbated by the high stiffness required in the equation of state (EOS)[16].

Another approach is to avoid a global time-step altogether. Prashant Goswami and Christopher Batty propose segmenting the time-step by spatial chunks.[5]. Asychronous SPH allows every particle to have its own time frame[14][2][3]. This is more efficient when there are only a few fast particles, as is often the case. Reinhardt also suggest using multiple queues in parallel, a method due to Kale and Lew, though they are unsure if this will scale well in practice on the GPU.[14][11].

Much research has been done to bring SPH to the GPU. Much of the difficulty lies in efficiently searching a fixed distance neighborhood of each particle, since a brute force search of every particle pair is infeasible. The work of Ihmsen et al. provided much of the groundwork with an early parallel implementation.[10] For this search they used a Z-index sort, a space filling curve which provides a cache-friendly ordering for the particles. Amada et al. present a partial GPU implementation which relies on the CPU for the neighborhood search, providing the information to the GPU as a texture.[1] Harada et al. present an early fully GPU implementation[7]. Later Hérault make use of the programmable pipeline to create a CUDA implementation[9], which they later released as open source[8]. Finally, Rustico et al. extend this to multiple GPUs.[15]

The neighborhood search has also garnered interest on its own. Ohno, Nitta and Nakai look at optimizing the neighborhood search for the GPU. Recently, Groß and Köster look at utilizing modern features of GPUs[6]. In particular, they leverage atomic memory synchronization between work groups in order to optimize the search further.

Author's address: Andrew Pregent.

XXXX-XXXX/2022/11-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

4 PROBLEM STATEMENT

5 METHOD

All particles in the simulation start at the same age, and ages of particles are tracked individually. A particle ages as it steps forward in time. All the particles in a cell are stepped forward in time at once. The time-step is determined by the youngest particle in the cell, and calculated using the CFL condition. This ensures that the simulation remains stable.

Previous methods of asychronous SPH integrate each particle forward in time individually. However, this was found to be prohibitively expensive to implement on the GPU. Instead, all the particles in a cell are integrated at once. This allows us to re-use the backtracked neighborhood surrounding the cell for each particle in the cell, greatly reducing the overhead of backtracking while still allowing different areas of the simulation to be integrated at different rates.

A cell is only updated if it is younger than all of its neighboring cells. This ensures that no region of the simulation is stepped too far ahead of other regions. This also ensures that it is never necessary to extrapolate the location of particles, only interpolation of physical values is required to bring a neighborhood of particles to the same frame of reference in time, increasing the stability of the simulation.

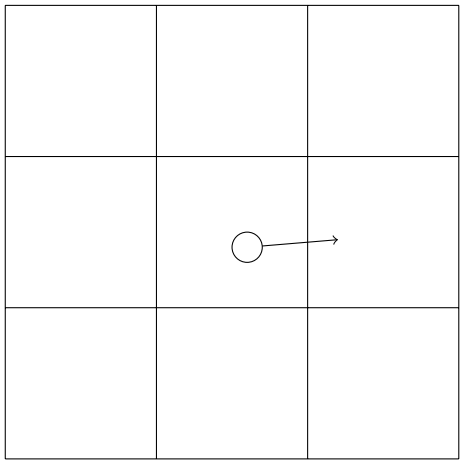

Fig. 1. The youngest particle determines the time step of a cell.

6 RESULTS

7 CONCLUSION

8 REFERENCES

REFERENCES

[1] Takashi Amada et al. “Particle-based fluid simulation on GPU”. In: *ACM workshop on general-purpose computing on graphics processors*. Vol. 41. Citeseer. 2004, p. 42.

[2] Xiaojuan Ban et al. “Adaptively stepped SPH for fluid animation based on asynchronous time integration”. In: *Neural Computing and Applications* 29.1 (2018), pp. 33–42.

[3] Mathieu Desbrun and Marie-Paule Gascuel. “Smoothed particles: A new paradigm for animating highly deformable bodies”. In: *Computer Animation and Simulation'96*. Springer, 1996, pp. 61–76.

[4] Robert A Gingold and Joseph J Monaghan. “Smoothed particle hydrodynamics: theory and application to non-spherical stars”. In: *Monthly notices of the royal astronomical society* 181.3 (1977), pp. 375–389.

[5] Prashant Goswami and Christopher Batty. “Regional time stepping for SPH”. In: *Eurographics 2014*. Eurographics Association. 2014, pp. 45–48.

[6] Julian Groß, Marcel Köster, and Antonio Krüger. “Fast and Efficient Nearest Neighbor Search for Particle Simulations.” In: *CGVC*. 2019, pp. 55–63.

[7] Takahiro Harada, Seiichi Koshizuka, and Yoichiro Kawaguchi. “Smoothed particle hydrodynamics on GPUs”. In: *Computer Graphics International*. Vol. 40. SBC Petropolis. 2007, pp. 63–70.

[8] A. Hérault et al. *GPU-SPH*. http://www.ce.jhu.edu/dalrymple/GPU/

[9] Alexis Hérault, Giuseppe Bilotta, and Robert A Dalrymple. “Sph on gpu with cuda”. In: *Journal of Hydraulic Research* 48.sup1 (2010), pp. 74–79.

[10] Markus Ihmsen et al. “A parallel SPH implementation on multi-core CPUs”. In: *Computer Graphics Forum*. Vol. 30. 1. Wiley Online Library. 2011, pp. 99–112.

[11] Kedar G Kale and Adrian J Lew. “Parallel asynchronous variational integrators”. In: *International Journal for Numerical Methods in Engineering* 70.3 (2007), pp. 291–321.

[12] Leon B Lucy. “A numerical approach to the testing of the fission hypothesis”. In: *The astronomical journal* 82 (1977), pp. 1013–1024.

[13] Joe J Monaghan. “Simulating free surface flows with SPH”. In: *Journal of computational physics* 110.2 (1994), pp. 399–406.

[14] Stefan Reinhardt et al. “Fully asynchronous SPH simulation”. In: *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 2017, pp. 1–10.

[15] Eugenio Rustico et al. “A journey from single-GPU to optimized multi-GPU SPH with CUDA”. In: *7th SPHERIC Workshop*. 2012, p. 56.

[16] Barbara Solenthaler and Renato Pajarola. “Predictive-corrective incompressible SPH”. In: *ACM SIGGRAPH 2009 papers*. 2009, pp. 1–6.