Para 2010 – State of the Art in Scientific and Parallel Computing – extended abstract no. 139
University of Iceland, Reykjavik, June 6–9 2010
http://vefir.hi.is/para10/extab/para10-paper-139.pdf

Fast GPU-based Fluid Simulations Using SPH

Smoothed Particle Hydrodynamics (SPH) on Graphics Processing Units (GPUs)

Øystein E. Krog*[1] and Anne C. Elster†[1]

[1]*Dept. of Computer and Information Science, Norwegian University of Science and Technology (NTNU)*

Abstract Graphical Processing Units (GPUs) are massive floating-point stream processors, and through the recent development of tools such as CUDA and OpenCL it has become possible to fully utilize the bandwidth and computational power they contain. A computationally challenging problem is how to model movements of liquids. We have developed a GPU-based framework for 3-dimensional Computational Fluid Dynamics (CFD) using Smoothed Particle Hydrodynamics (SPH). This paper describes the methods used for implementing fast simulations of fluids dynamics using GPUs, and compares the performance of the implementation to previous SPH implementations. Our implementation uses the acceleration data structures found in the NVIDIA "Particles" demo, but implements SPH instead of its simpler mass-spring system. The implementation uses CUDA and has been highly optimized to the point where a scaled simulation can run in "real-time". We implement two different SPH models, a simplified model for newtonian fluids, and a complex model for non-newtonian fluids. The complex SPH model is used to simulate flowing snow avalanches. Using our simple SPH model and a NVIDIA GeForce 260 GTX we achieve 38 FPS with 256K particles, 63 FPS with 128K particles and 99 FPS with 64K particles. Open source code will be provided. This should make our work very useful not only for our current work on simulating snow avalanches, but also for other CFD applications that need faster simulations of many particles.

Keywords GPU, CFD, SPH, GPGPU, CUDA, Fluid

1 Introduction

The simulation of fluids is an interesting problem due to the importance of fluids in the physical world. Simulating fluids also present a large challenge due to the large computational demands that arise from the complex behaviors of fluids, especially in 3 dimensions. Due to these demands most fluid simulations are not done in "real-time", in fact many previous SPH implementations have been constrained to 2D. Our framework makes it possible to do SPH simulations in 3D for large problem sizes in "real-time". So far it has not been possible to achieve beliveable "real-time" fluid simulations for all but the smallest and coarsest models. By using an accelerator such as the GPU, the number of particles modeled in the simulation can be increased considerably, and the overall simulation speed is greatly increased compared to CPU-based simulations.

*Email: oystein.krog@gmail.com
†Email: elster@idi.ntnu.no

2 Computational Fluid Dynamics

Computational Fluid Dynamics (CFD) is a large field, in which the Navier-Stokes equations play an important role. These equations can be solved numerically, and several such methods have been developed in order to simulate fluids on computers.

2.1 Navier-Stokes Equations

The Navier-Stokes equations in simplified Lagrangian form consist of mass and momentum conservation:

$$\frac{d\rho}{dt} = -\rho \nabla \cdot \mathbf{v} \tag{1}$$

$$\frac{d\mathbf{v}}{dt} = -\frac{1}{\rho}\nabla p + \frac{1}{\rho}\nabla \cdot \mathbf{S} + \mathbf{f} \tag{2}$$

Where $\mathbf{v}$ is the velocity field, ρ the density field, ∇p the pressure gradient field resulting from isotropic stress, $\nabla \cdot \mathbf{S}$ the stress tensor resulting from deviatoric stress and $\mathbf{f}$ an external force field such as gravity.

For incompressible newtonian fluids the momentum conservation reduces to:

$$\frac{d\mathbf{v}}{dt} = -\frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla^2 \mathbf{v} + \mathbf{f} \tag{3}$$

Where the term μ is the dynamic viscosity of the fluid.

2.2 Smoothed Particle Hydrodynamics

In SPH the different effects of Navier-Stokes are simulated by a set of forces that act on each particle. These forces are given by scalar quantities that are interpolated at a position $\mathbf{r}$ by a weighted sum of contributions from all surrounding particles within a cutoff distance h in the space Ω. In integral form this can be expressed as follows [5]:

$$A_i(\mathbf{r}) = \int_\Omega A(\mathbf{r}')W(\mathbf{r}-\mathbf{r}', h)d\mathbf{r}' \tag{4}$$

The numerical equivalent is obtained by approximating the integral interpolant by a summation interpolant [5]:

$$A_i(\mathbf{r}_i) = \sum_j A_j \frac{m_j}{\rho_j} W(\mathbf{r}_{ij}, h) \tag{5}$$

where j iterates over all particles, m_j is the mass of particle j, $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$ where $\mathbf{r}$ is the position, ρ_j the density and A_j the scalar quantity at position $\mathbf{r}_j$.

For a more comprehensive introduction to SPH, please refer to [5].

2.3 Avalanche SPH

Snow avalanches wary greatly in behaviour, from powder-snow avalanches to so called dense-flow, or flowing snow avalanches. Snow avalanches usually appear as a viscous flow down a slope, and it is this obvious property which has prompted the use of fluid dynamics in avalanche simulation [1]. Several viscosity models exist for modelling non-newtonian fluids, and rheological parameters have been collected for flowing snow [6]. Many SPH models exist for viscoplastic fluids, from melting objects [8] to lava flows [9] and generalized rheological models [4].

3 Methods and Implementation

CUDA is a parallel computing architecture developed by NVIDIA. We use CUDA for C, which is basically the C language with some added syntax. GPUs are massively parallel, with several thousand threads available. We parallelize the calculation of SPH by assigning a thread to each particle in the simulation. Each thread is then responsible for calculating the SPH sums over the surrounding particles. When accessing memory on the GPU, *coalesced* (correctly structured) access is very important. Due to the nature of the algorithm, fully coalesced access is unfortunately not possible. By utilizing the texture cache this problem is greatly alleviated.

3.1 Nearest-Neighbor Search

The summation term in the SPH-formulation is computationally heavy, it requires looking at many nearby particles and computing interactions between them. To avoid a naive brute-force $O(N^2)$ search for neighbors, a nearest-neighbor search algorithm is commonly used, such as a linked list or a uniform grid. We use the algorithm presented in [2], which can be summarized as follows:

1. Divide the simulation domain into a uniform grid.
2. Use to the spatial position of each particle to find the cell it belongs to.
3. Use the particle cell position as input to a hash function (a spatial hash)
4. Sort the particle according to their spatial hash.
5. Reorder the particles in a linear buffer according to their hash value.

Particles in the same cell will then lie consecutively in the linear buffer, and finding "neighbors" is simply a matter of iterating over the correct indices in the buffer. For the sorting we used the fastest radix sort available for the GPU at the time of implementation (by Satish *et al* [10]).

3.2 Non-Newtonian Fluids

non-newtonian fluids differ from newtonian fluids in that their viscosity is not constant. In a newtonian fluid the relation between shear stress and the strain rate is linear, with the constant of proportionality being the viscosity. For a non-newtonian fluid the relation is nonlinear and can even be time-dependent. There exist many classes of non-newtonian fluids, and many types of models, of which we implement several. The complex SPH model differs primarily in that it includes the much more complex stress calculation presented in [4].

3.3 SPH Models

We use our framework to implement two different SPH models, a simplified model for interactive use, based on a model by Müller *et al.*[7] and a complex model for non-newtonian fluids based on a model by Hosseini *et al.*[4]. The simplified model is focused on interactive performance and creates a visually pleasing water-like fluid, and the complex model is used to simulate flowing-snow avalanches through the use of different rheological(the study of the flow of matter) models.

By using the SPH formulation we end up with the following simulation equations:

$$\rho_i = \sum_j m_j W(\mathbf{r}_{ij}, h) \tag{6}$$

$$\mathbf{f}_i^{pressure} = -\frac{1}{\rho}\nabla p(\mathbf{r}_i) = \sum_{j\neq i} m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right)\nabla W(\mathbf{r}_{ij}, h) \tag{7}$$

The incompressible fluid is simulated as a weakly compressible fluid where the incompressibility constraint is applied to the pressure p by using an equation of state given by the ideal gas law with an added rest density: $p = k(\rho - \rho_0)$

For calculating non-newtonian fluids we use the formulation presented by [4].

$$\nabla \mathbf{v}_i = \sum_j \frac{m_j}{\rho_j}(\mathbf{v_i} - \mathbf{v_j}) \otimes \nabla W(\mathbf{r}_{ij}, h) \tag{8}$$

Where $\otimes$ is the outer product and $\nabla \mathbf{v}$ is a tensor field (a 3x3 matrix). Giving us $\mathbf{D}_i = \frac{1}{2}(\nabla \mathbf{v} + (\nabla \mathbf{v})^T)$, and the stress tensor $\mathbf{S}_i = \eta(D)\mathbf{D}$, with $D = \sqrt{\frac{1}{2} trace(\mathbf{D})^2}$. The term $\eta(D)$ is essentially the viscosity for the fluid, which for a newtonian is constant. By using a rheological model for calculating this term, various non-newtonian fluids can be simulated. Finally the stress tensor is calculated:

$$\mathbf{f}_i^{stress} = \frac{1}{\rho}\nabla \cdot \mathbf{S}_i = \sum_{j\neq i} \frac{m_j}{\rho_i \rho_j}(\mathbf{S}_i + \mathbf{S}_j) \cdot \nabla W(\mathbf{r}_{ij}, h) \tag{9}$$

Thus we have that the acceleration for a particle is given by:

$$\mathbf{a}_i = f_i^{pressure} + f_i^{stress} + f_i^{external} \tag{10}$$

3.4 SPH Algorithm

Due to the data dependencies between the various force, some of them must be calculated separately. Each calculation step is essentially a summation over neighboring particles, and we combine the force calculations using loop

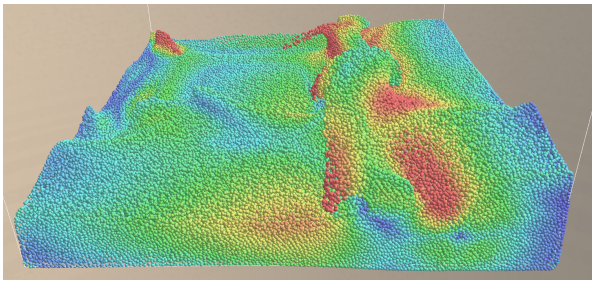

Figure 1: A screenshot of the simple SPH model with 256K particles. Hue-based gradient shading for the velocity of the particles.

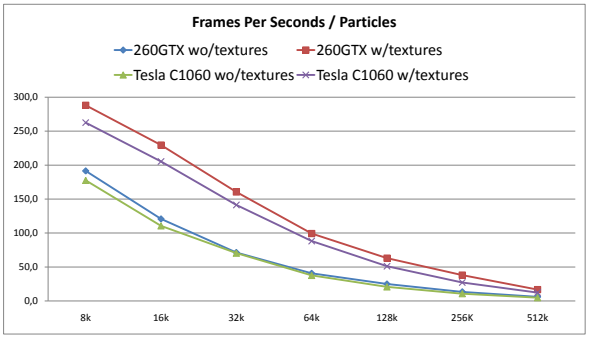

Figure 2: Performance scaling for the simple SPH model.

fusion as far as it is possible. For the complex SPH models we end up with the following steps:

1. Update the hashed, radix sorted, uniform grid.
2. Calculate the SPH density
3. Calculate the SPH velocity tensor
4. Calculate the SPH pressure and SPH stress tensor
5. Integrate in time.

For the simplified SPH model, the stress tensor is replaced with a simplified viscosity approximation which ignores deviatoric stress, since it is not strictly nescessary for incompressible newtonian fluids. As a result the viscosity calculation does not need the velocity tensor, step 3 can be dropped, and the viscosity force can be computed together with the pressure in step 4. For integrating the velocity change we use the Leap-Frog method, since it is more accurate than simple Euler integration while still mainting low memory and computational costs.

3.5 Simulation Framework

The implementation of the simulation framework is coded in C++ and the GPU code using C for Cuda. The simulation itself is separated logically into a separate library. Since the entire simulation is done on the GPU, an API for integration of rendering is nescessary to avoid the high cost of copying rendering buffers to the CPU. Using the API it is possible to implement direct rendering of simulation buffers for both Direct3D and OpenGL. For the case of OpenGL this is possible using Vertex Buffer Objects (VBOs), which can be rendered directly using shaders with fairly low overhead. The CUDA simulation kernels have been highly optimized. By manually optimizing register usage, reording memory accesses and optimizing the block sizes for the CUDA code, performance gains as large as 40% were realized.

4 Results

For performance testing two different GPUs were used, a Geforce 260GTX and a Tesla C1060. The computer had a Intel Core2 Quad Q9550, 4GB DDR3 ram and was running Windows Vista 64-bit .

4.1 Choice of Metrics

Comparing and evaluating the performance of the simulations is difficult due to the large amount of parameters and their effect on performance. In addition it is hard to compare to other SPH implementations due to different SPH models and paramaters. For the simple SPH model we compare against Müller and Harada which use a very similar SPH model, using rest density selected to simulate water, with the dynamic viscosity set to 1. For the complex SPH model we have not completed our performance evaluation, but compare against the implementation of the simple SPH model. An additional difficulty with the non-newtonian model is that the viscosity is not constant, but varies according to stress, which means that it is hard to give exact performance numbers. For the purposes of measuring performance, we choose the same parameters as the simple model, and a simplified rheology model with a (very high) constant viscosity of 300. To obtain absolute performance numbers we use a fairly simple simulation setup; a square simulation domain with simple repulsive forces as walls where a cubic volume of fluid is dropped into a shallow pool of water. The performance numbers were measured when the fluid had reached a stable equilibrium. We use FPS as our performance unit, because our simulation is frame-locked with the rendering.

Memory Usage Due to the hashed uniform grid structure the memory usage is fairly sparse. Had the uniform grid been allocated directly the usage would be much larger. The memory usage in bytes is $176N$ where N is the number of particles. The memory usage of the complex SPH models is $212N$ bytes. This means that it is possible to simulate very large systems even on commodity hardware. We have tested systems up to 2048K particles on a NVIDIA Geforce 260GTX.

4.2 Performance

Müller *et al.*[7] achieves 20 FPS at 2200 particles on a 1.8 GHz Pentium 4. Harada *et al.*[3] achieves 17 FPS at 60000 particles on an NVIDIA GeForce 8800GTX using OpenGL and CG.

Using our simple SPH model and a NVIDIA GeForce

260 GTX we achieve 38 FPS with 256K particles, 63 FPS with 128K particles and 99 FPS with 64K particles. On the NVIDIA Geforce 260GTX the complex SPH model runs at 17 FPS with 256K particles, 31 FPS at 128K particles and 55 FPS at 64K particles.

It is worth noting that direct comparison of FPS is not always a good measure of performance. An SPH implementation can have large variations in performance depending on the parameters of the system.

By changing only the number of particles and the particle mass (to keep the fluid volume constant), the scaling in Figure 2 is observed. Comparing the 260GTX and the Tesla C1060, the performance on the Tesla is somewhat lower, which is likely due to the lower memory clock on the Tesla. This finding is congruent with the theory that the SPH algorithm is highly bandwidth constrained on the GPU. This is due to imperfect coalescing of memory reads.

4.3 Real-Time Appearance

By scaling the simulation domain, and relaxing the accuracy requirements by selecting large timesteps, the fluid simulations produce belivable "real-time" behavior. We have tested up to 2048K particles at which point the simulation can no longer be considered "real-time", but it is still very fast compared to previous implementations. In fact the CPU implementation by Müller *et al.*[7] achieves 20 FPS with 2200 particles, in contrast to our implementation on the GPU which achieves 17 FPS with 512K particles. It should be noted that this is not truly "real-time" simulation of a volume of liquid, since both the fluid volume and the time is not to scale. Our complex SPH model is not as well suited to "real-time" simulation due to the nescessity of a somewhat lower timestep in order to support higher viscosities.

5 Conclusions

In this paper, we presented an implementation of Smoothing Particle Hydrodynamics (SPH) on the GPU. Our implementation achieves very good performance since we take advantage of the massive amounts of parallelism available on modern GPUs, as well as use specialized acceleration data structures. Our SPH implementations achieve very good performance compared to previous implementations, and can be used to produce believable "real-time" behavior.

Using our simple SPH model and a NVIDIA GeForce 260 GTX we achieve 38 FPS with 256K particles, 63 FPS with 128K particles and 99 FPS with 64K particles.

5.1 Current and Future Work

Simulations of snow can be used for everything from gaming to avalanche prediction. For games snow simulation can help create complex environments, which can lead to numerous possibilities for game-play mechanics. Predicting the behavior of snow avalanches can help prevent loss of both life and property.

The resource usage of our model has been investigated and it was found that it does not consume much memory but is very memory bandwidth intensive. To further increase performance it might be interesting to investigate the possibility of using multiple GPUs.

Finally, the visualization of the fluid model can be improved, at the moment a very simple method of direct particle rendering is used. By using a surface reconstruction model such as Marching-Cubes a real surface can be rendered. It is also possible to use screen-space surface rendering techniques to approximate the fluid surface without the large computation cost associated with true surface reconstruction.

References

[1] E. Bovet, B. Chiaia, and L. Preziosi. A new model for snow avalanche dynamics based on non-newtonian fluids. *Meccanica*.

[2] S. Green. Cuda particles. Technical report, NVIDIA.

[3] T. Harada, S. Koshizuka, and Y. Kawaguchi. Smoothed particle hydrodynamics on gpus. 2007.

[4] S. M. Hosseini, M. T. Manzari, and S. K. Hannani. A fully explicit three-step sph algorithm for simulation of non-newtonian fluid flow. *International Journal of Numerical Methods for Heat & Fluid Flow*, 17(7):715–735, 2007.

[5] M. Kelager. Lagrangian fluid dynamics using smoothed particle hydrodynamics. Master's thesis, University of Copenhagen, Department of Computer Science, 2006.

[6] M. A. Kern, F. Tiefenbacher, and J. N. McElwaine. The rheology of snow in large chute flows. *Cold Regions Science and Technology*, 39(2-3):181 – 192, 2004.

[7] M. Müller, D. Charypar, and M. Gross. Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 154–159, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[8] A. Paiva, F. Petronetto, T. Lewiner, and G. Tavares. Particle-based non-newtonian fluid animation for melting objects. *Computer Graphics and Image Processing, Brazilian Symposium on*, 0:78–85, 2006.

[9] A. Paiva, F. Petronetto, T. Lewiner, and G. Tavares. Particle-based viscoplastic fluid/solid simulation. *Computer-Aided Design*, 41(4):306 – 314, 2009. Point-based Computational Techniques.

[10] N. Satish, M. Harris, and M. Garland. Designing efficient sorting algorithms for manycore gpus. NVIDIA Technical Report NVR-2008-001, NVIDIA Corporation, Sept. 2008.