Interactively Simulating Fluid based on SPH and CUDA

Yige Tang
Beijing Normal University
No. 19 XinJieKouWai St
Haidian District
100875, Beijing, China
solidsnake1905@gmail.com

Zhongke Wu
Beijing Normal University
No. 19 XinJieKouWai St
Haidian District
100875, Beijing, China
zwu@bnu.edu.cn

Mingquan Zhou
Beijing Normal University
No. 19 XinJieKouWai St
Haidian District
100875, Beijing, China
mqzhou@bnu.edu.cn

ABSTRACT

In this paper, we propose a novel method of interactive fluid simulating based on SPH, and implement it on CUDA (Compute Unified Device Architecture). Firstly we use SPH (Smoothed Particle Hydrodynamics) theory to simulate the motion of fluids. Secondly we propose an interactive method between fluid and rigid objects. We treat the rigid objects as two different types, static one and dynamic one. We deal with the two types in separately suitable ways in order to enforce their motion similar to the real world. By taking advantages of CUDA which are greatly effective for large scale numeric computation in parallel, our simulation achieves real time with low cost.

Keywords

Simulation, Fluid, SPH, CUDA, Interactive

1. Introduction

As a common phenomenon in nature, simulation of fluid including water and smoke is an important part in visual reality. After Reeves' proposal of particle system in 1983, which is used to present non-solid objects such as water, fire and smokes, researchers have done a lot of work for simulating fluid vividly and effectively. Fluid simulation is usually applied in medical image visual reality and video game. Simulating fluid in computer is a tough issue. Although the theory of computational fluid dynamics has existed for many years, some properties of fluid which contains convection, turbulence and surface tension are difficult to be expressed through simple modeling. However, due to the fact that the real time simulation is more important than computational precision in computer graphics, the mathematical model and implementation focus on real time and visual effects more than precision in fluid computation.

The earliest approach of fluid simulation is simple particle system, which is proposed by Reeves in simulating fire and flake [Ree83a]. After that, Shinya and Stam improved particle system respectively in their work by introducing random turbulences [Shi92a] [Sta93a]. Fluid simulation based on Navier-Stokes Equations is implemented in 2D space firstly, both Gamito and Yaeger et al. have made contributions to it [Gam95a] [Yae86a]. In 1997, Stam et al. proposed an approach based on grid to simulating smoke [Sta99a], which is the first interactive method on fluid simulation.

There are two approaches of simulating fluid based on Navier-Stokes equations, the Eulerian viewpoint and the Lagrangian viewpoint.

In the Eulerian approach, which is based on

position, the fluid properties on some fixed points are computed. The absolute locations of fixed points never change, and the properties need to be computed are velocity, pressure, density, etc. The Eulerian method is appropriate for simulating gases, while is not able to present liquid well in wave and foam. The Lagrangian approach is different from Eulerian one as it is based on particles. Desbrun et al. and Tonnesen use particles to present soft objects [Ses96a] [Ton98a]. Witkin et al. use particles to control implicit surface [Wit94a]. Dan et al. simulate lava by using particles [Sto99a]. Comparing with Eulerian approach, Lagrangian approach has several following advantages. First, it's not grid based, so fluid can move in the whole scene and interact with other object. Second, it can present more details of fluid, such as the mergence and dispersing of water drop.

Recently, the most common Lagrangian approach is based on smoothed particle hydrodynamics (SPH), which is proposed by Lucy in 1977 [Luc77a] firstly used in astronomical phenomenon. Muller et al. introduced SPH theory to compute fluid simulation in 2003 [Mul03a]. Their approach simplifies solution method of Navier-Stokes equations, while the amount of calculation is so great that it's hard to implementing animation in real time when the quantity of particles is huge.

Development of programmable GPU technique makes large-scale numeric computation be solved effectively under low cost. Due to the feature, SPH method can be solved in parallel. Through programming on GPU, real-time simulation for large-scale-particles fluid becomes possible. The GPU based radix sorting approach designed by Satish et al. combining with spatial uniformed-grid, reduces the cost in finding neighbors of particles [Sat08a].Then the method improves the computational speed. In 2004 Amada et al. implemented forces' computation of particles [Ama04a], while the neighbor finding task is still done by CPU. The method completely implemented on GPU is firstly proposed by Kolb and Cunts in 2005 [Kol05a]. This approach emerges earlier than CUDA.

CUDA is a general-purpose GPU programming toolkit released by nVIDIA in 2007, which makes it possible to use C language program on GPU. With the help of CUDA, the processors of GPU can run parallelly by independently executing the same groups of operations on different sets of data. The above features are well suited for SPH method, because the same groups of operations such as force computation, speed and position update are completely same and have to be executed for each particle.

2. Fundamentals

2.1 Smoothed Particle Hydrodynamics

Essentially, SPH is a computational model to compute the interactive result of each particle in the fluid system. It defines a way to compute properties of a fix point impacted by other particles in the continuous space. A distance related weighted function $W(r)$ which is called kernel function is the key of SPH method.ris the distance between some position x and particle $i's$ position x_i. Another form of kernel function is $W(|x - x_i|)$. Kernel function satisfies the following equation $\int W(|x - x_i|)dx = 1$.We use poly6 kernel as our kernel function, its form is

$$W_{poly6}(x) = \frac{315}{64\pi h^9}\begin{cases}(d^2 - r^2)^3 & 0 \le r \le d \\ 0 & \text{otherwise}\end{cases} \quad (1)$$

After affirming kernel function, smoothed fields $A_s(x)$ of arbitrary attributes A_i of the particle as

$$A_s(x) = \sum_j m_j \frac{A_j}{\rho_j} W(|x - x_j|) \quad (2)$$

After replacing the kernel in formula (2) by the gradient of the kernel, we easily get the gradients of the field as following.

$$\nabla A_s(x) = \sum_j m_j \frac{A_j}{\rho_j} \nabla W(|x - x_j|) \quad (3)$$

In SPH simulation, we only need consider pressure, viscous force and external force. Pressure and viscous force can be calculated by above formulations.

2.1.1 Pressure

Formula (2) is used on computing pressure yields

$$f_i^{pressure} = -\nabla p(x_i) = -\sum_j m_j \frac{p_j}{\rho_j} \nabla W(|x_i - x_j|) \quad (4)$$

If there are only two particles, the pressure force calculated by formula (4) will not be symmetric because the pressures at the locations of the two particles are not equal. Following equation is a simple, stable and fast solution.

$$f_i^{pressure} = -\nabla p(x_i) = -\sum_j m_j \frac{p_i + p_j}{2\rho_j} \nabla W(|x_i - x_j|) \quad (5)$$

Since particles only carry the three quantities mass, position and velocity, the pressure at particle locations has to be evaluated firstly.
We compute pressure by using equation (6)

$$p = k(\rho - \rho_0) \quad (6)$$

Here k is a gas constant and ρ_0 is the environmental pressure.

2.1.2 Viscosity

We use following equation to calculate viscosity.

$$f_i^{viscosity} = \mu \nabla^2 v(x_i) = \mu \sum_j m_j \frac{v_j}{\rho_j} \nabla^2 W(|x_i - x_j|) \quad (7)$$

Since viscosity forces are dependent on velocity differences and not on absolute velocities, there is a natural way to symmetrize the viscosity forces by using velocity differences:

$$f_i^{viscosity} = \mu \sum_j m_j \frac{v_j - v_i}{\rho_j} \nabla^2 W(|x_i - x_j|) \quad (8)$$

2.1.3 External Forces

External forces include gravity, collision forces and interaction forces with environment. They are applied directly to the particles without the use of SPH.

2.1.4 Computing Procedure

The SPH model is executed under the following steps:

1) Find neighbors of each particle
2) Calculate the particle density
3) Calculate forces on particles
4) Update position and velocity of particles in next time step

After finishing these steps, the particles move obeying SPH rules can be rendered shown in Figure 1.

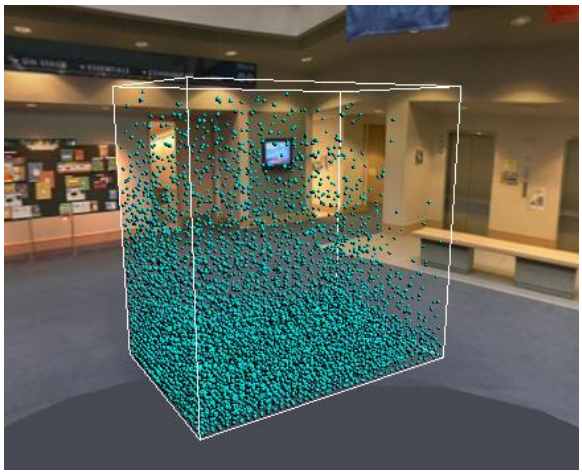


Figure 1. A group of SPH particles. Their motions are computed by above formulas.

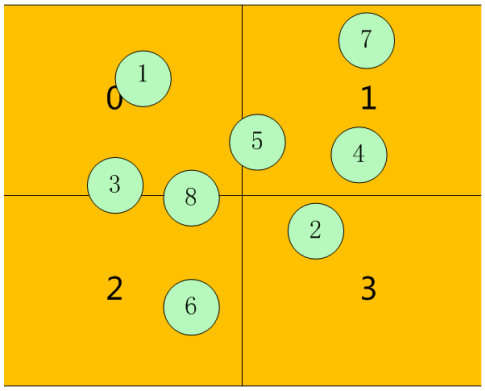
2.2 Neighbors Search

SPH is computationally heavy. The first step is finding neighbors of each particle. In worst case, each particle should be compared with all of others, whose complexity is $O(N^2)$.To avoid this, using uniform grid reduces most of cost. We use the algorithm presented in [Gre08a], which can be summarized as follows:

1) Divide the simulation domain into a uniform grid.
2) Use 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
4) Sort the particle according to their spatial hash.
5) Reorder the particles in a linear buffer according to their hash value.

After those steps, it satisfies that particles in the same cell will lie consecutively in the linear buffer, which makes finding neighbors much more effectively.

Radix sort's implementation on GPU is proposed by Satish et al. [5]



(a)

Before sort:

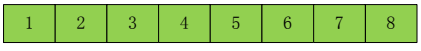
ParticleIndex

ParticleCell

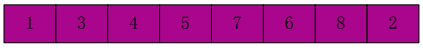
(b)

After sort:

ParticleIndex

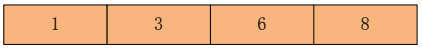


OldIndex



ParticleCell

CellStart



CellEnd

(c)

Figure 2. Data structure used by radix sort. (a) presents particles' position in the cells. (b)is status of arrays before sort. (c) shows the arrays needed after sort. The particles in same cell are consecutive in ParticleIndex array. We use OldIndex array to get the particle index before sort in order to access the velocity and position of particles.

3. Interaction with Rigid Objects

We classify rigid objects into two types. One type of them is static object such as glass with water. However the water acts, the glass keeps still. Another type is dynamic object such as a block on the water. We use different methods to simulate them respectively.

3.1 Interaction with Static Objects

It is easy to compute the interaction between particles and static rigid objects. Since the object can't move, there is no need to compute a force from the particles on the object. We only compute the penalty force, which forces the particle back into the fluid region in the opposite direction. Deformation will not happen on that static object.

When a particle collides with a static object, a penalty force is applied. Moore and Wilhelms provide a comprehensive introduction to the penalty force method [Mat88a]. Here we use the penalty force applied to a fluid particle can be calculated by equation (9).

$$f^{col} = k_s dn + k_d(v \cdot n)n \qquad (9)$$

In equation (9), d is the distance by which the particle has interpenetrated the static object, k_s is a

spring constant, k_d is a damping constant, n is the normal vector at the collision location, and v is the relative velocity of the particle to the static object.

3.2 Interaction with Dynamic Rigid Bodies

Computing interaction with dynamic objects is more complicated.

We treat dynamic rigid bodies as a portion of fluid whose initial density is greater than the initial density of the fluid. The only difference is in computing the pressure impaction between fluid and particles of rigid bodies. The rigid particles push the fluid away from the rigid bodies. Likewise, the fluid particles apply a pressure that results in a pure translation or rotation of the rigid bodies.

The pressure applied on a rigid body particle is given by equation (10), and the pressure at a fluid particle is given by equation (11).

$$p = \begin{cases} k^{rigid}\left(\rho - \rho_0^{rigid}\right) & \rho \geq \rho_0^{rigid} \\ 0 \text{ otherwise} \end{cases} \quad (10)$$

$$p = \begin{cases} k^{fluid}\left(\rho - \rho_0^{fluid}\right) & \rho \geq \rho_0^{fluid} \\ 0 \text{ otherwise} \end{cases} \quad (11)$$

Here, ρ_0^{rigid} is the rigid body's initial density, while ρ_0^{fluid} is the fluid's.

The rigid objects construct of rigid particles, whose motion is restricted to translation and rotation without deformation. Pressure is applied individually to each rigid particle, causing them to move independently. After that, we need modify the position of those particles to keep the rigid body's shape.

The mass M of whole object formed by rigid particles is calculated by formula (12).

$$M = \sum_j m_j \quad (12)$$

Here, m_j is the single particle's mass. Assuming all of the particles are the same, then the velocity of the center of mass can be computed by formula (13).

$$v_g = \frac{1}{N}\sum_j v_j \quad (13)$$

N is the amount of particles, and v_j is velocity of a single particle j.The angular velocity of the rigid body ω is approximated by equation (14).

$$\omega = \frac{1}{I}\sum_j q_j \times v_j \quad (14)$$

Here q_j is the location of a single particle j relative to its corresponding rigid body center of mass.

By using those above equations, we can get a rigid object's velocity and angular velocity. Then we can simulate the motion of that rigid object.

4. CUDA Computation

In our implementation, we use three texture arrays to store positions, velocities and densities of particles in last computing procedure, and we write new values to global memory of GPU respectively. The following Table 1 outlines the steps of our SPH algorithm with CUDA.

```
SPHCompute() //for each frame
{
    Copy particles' properties from CPU to GPU
    Set physical parameters of environment
    Hash particles by their spatial position
    Sort particles using radix-sort
    /*--ComputeDensity--*/
    Launch CUDA kernel function for each thread
    Each thread calculate one particle
    Compute new densities using other particles in 27 neighbor grids by using SPH kernel function
    __syncthreads()
```

/*--Compute Force--*/
Launch CUDA kernel function for each thread
Each thread calculate one particle
Calculate forces using densities computed above, including pressure and viscosity
Handle external forces such as gravity
}

Table 1.The procedure of SPH computing in our implementation

5. Rendering

By using density we have computed in above work, the Marching Cubes method is applied in our work. An image spaced method is applied to simulate the refractive effect. The rendering method not only obtains a good visualization effect, but also bring little calculation burden. The rendering results are shown in Figure 3.

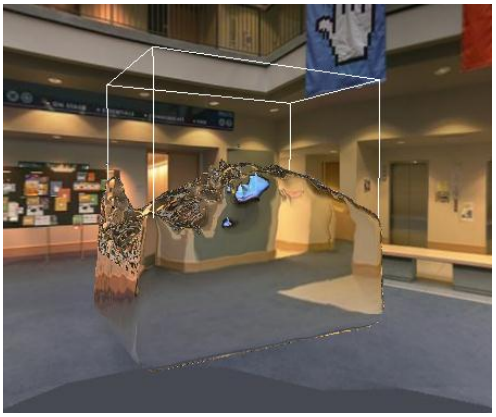
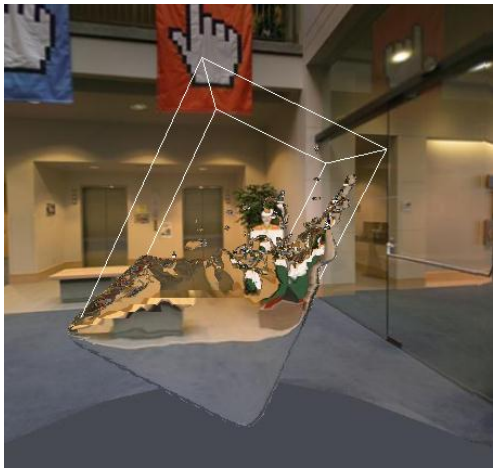


Figure 3. Water rendered by using marching cube. The refraction effects are generated by CGSL

.

6. Conclusions

In this paper, we presented an interactive fluid simulating method based on Smoothed Particle Hydrodynamics. Our implementation can simulate fluid in real time and vividly with the help of CUDA and GLSL. The fluid we simulate looks like in real world with wave and foam. Additionally, the method which we propose in this paper also makes the fluid interacting with rigid objects well. The fluid in the box can move with the rotation of the box. The wood block in the box is floated under the force from the water. Figure 4 shows the interactive results of our implementation.



(a) Result of interacting with static rigid objects.

(b) Result of interacting with dynamic rigid objects.

Figure 4. Results of interacting with rigid objects

Through using CUDA, the method achieves real time simulation, since we take the advantage of the capacity of parallel computation afforded by GPU. Our implementation runs on below platform:

Windows7 OS 64-Bit， Intel(R) Core(TM) i7 CPU @3.07GHZ, 6GB RAM and GeForce GTX 570 with 1280MB video memory.

The results of frame rates are shown in table 2.

number of Particles	frame rate (FPS)
65,536	98.1
131,072	42.2
262,144	18.8

(a) results without free surface rendering

number of Particles	frame rate (FPS)
65,536	50.9
131,072	22.8
262,144	10.4

(b) results with free surface rendering

Table 2. Runtime result of our implementation on above platform

The large performance gap between the results in Table 2(a) and the ones in Table 2(b) is due to the surface rendering. The surface rendering takes extra cost on memory and time. The above table presents that our implementation performs very well.

7. References

[Ama04a] T. Amada, M. Imura, Y. Yasumuro, Y. Manabe, K. Chihara. Particle - based fluid simulation on the GPU. Proc. ACM Workshop on General - purpose Computing on Graphics Processors, 2004.

[Gam95a] M. N. Gamito, P. F. Lopes, and M. R. Gomes. Two dimensional Simulation of Gaseous Phenomena Using Vortex Particles. In Proceedings of the 6th Eurographics Workshop on Computer Animation and Simulation, pages 3–15. Springer - Verlag, 1995.

[Gre08a] S. Green. Cuda Particles. Technicle report, NVIDIA.

[Kol05a] A. Kolb, N. Cuntz. Dynamic particle coupling for GPU-based fluid simulation. Proc. 18th Symposium on Simulation Technique, 722-727, 2005.

[Luc77a] L. B. Lucy. A numerical approach to the testing of the fission hypothesis. The Astronomical Journal, 82: 1013-1024, 1977.

[Mat88a] M. Matthew, J. Wilhelms, Collision Detection and Response for Computer Animation. Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques, ACM Press: p. 289-298, 1988.

[Mul03a] M. Muller, D. Charypar, M. Gross. Particle-Based Fluid Simulation for Interactive Applications.Eurographics/SIGGRAPH Symposium on Computer Animation, 2003

[Ree83a] W. T. Reeves. Particle systems: a technique for modeling a class of fuzzy objects. ACM Transactions on Graphics 2(2), pages 91-108, 1983.

[Sat08a] N. Satish, M. Harris, M. Garland. Designing efficient sorting algorithms for many core gpus. NVIDIA Technical Report NVR-2008-001, NVIDIA Corporation, Sept, 2008.

[Ses96a] M. Desbrun and M. P. Cani. Smoothed particles: A new paradigm for animating highly deformable bodies. In Computer Animation and Simulation ’96 (Proceedings of EG Workshop on Animation and Simulation), pages 61-76. Springer - Verlag, Aug 1996.

[Shi92a] M. Shinya and A. Fourier Stochastic Motion: Motion Under the Influence of Wind. In Proceedings of Eurographics’92, pages 119-128, September 1992.

[Sta93a] J. Stam and E. Fiume. Turbulent Wind Fields for Gaseous Phenomena. In Proceedings of SIGGRAPH ’93, pages 369–376. Addison-Wesley Publishing Company, August 1993.

[Sta99a] J. Stam. Stable fluids. In Proceedings of the 26th annual conference on Computer graphics and interactive techniques, pages 121–128. ACM Press/Addison-Wesley Publishing Co., 1999.

[Sto99a] D. Stora, P. Agliati, M. Cani, F. Neyret, J. Gascuel. Animating lava flows. In Graphics Interface, pages 203-210, 1999.

[Ton98a] D. Tonnesen. Dynamically Coupled Particle Systems for Geometric Modeling, Reconstruction, and Animation. PhD thesis, University of Toronto, November 1998.

[Wit94a] A. Witkin and P. Heckbert. Using particles to sample and control implicit surfaces. In Computer Graphics (Proc. SIGGRAPH ’94), volume 28, 1994.

[Yae86a] L. Yaeger and C. Upson. Combining Physical and Visual Simulation. Creation of the

Planet Jupiter for the Film 2010. ACM Computer Graphics (SIGGRAPH ’86), 20(4):85–93, August 1986.

Acknowledgements

The work is partially supported by National Natural Science Foundation of China (No: 61170170) and the Fundamental Research Funds for the Central Universities (No: 2009SD-11)
Corresponding author, Email: zwu@bnu.edu.cn