Fluid simulation with two-way interaction rigid body using a heterogeneous GPU and CPU environment

José Ricardo da S. Junior
Media Lab - UFF

Esteban W. Clua
Media Lab - UFF

Paulo A. Pagliosa
FACOM - UFMS

Anselmo Montenegro
Media Lab - UFF

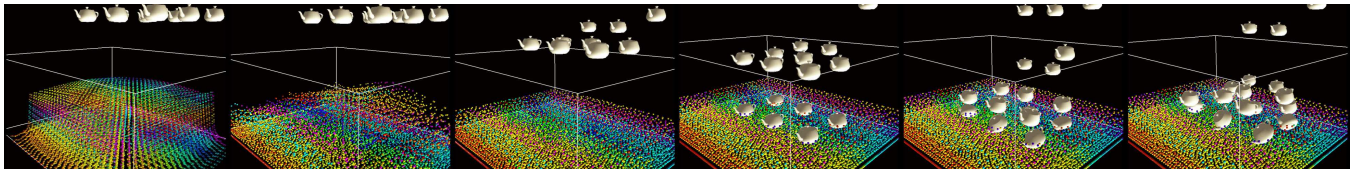

Figure 1: *Fluid simulation with two-way interaction between rigid bodies*

Abstract

Simulation of natural phenomena, such as water and smoke, is a very important topic to increase real time scene realism in videogames. Besides the graphical aspect, in order to achieve realism, it is necessary to correctly simulate and solve its complex governing equations, requiring an intense computational work. Fluid simulation is achieved by solving the Navier-Stokes set of equations, using a numerical method in CPU or GPU, independently, as these equations do not have an analytical solution. The real time simulation also requires the simulation of interaction of the particles with objects in the scene, requiring many collision and contact forces calculation, which may drastically increase the computational time.

In this paper we propose an heterogeneous multicore CPU and GPU hybrid architecture for fluid simulation with two-ways of interaction between them, and with a fine granularity control over rigid body's shape collision. We also show the impact of this heterogeneous architecture over GPU and CPU bounded simulations, which is commonly used for this kind of application. The heterogeneous architecture developed in this work is developed to best fit the Single Instruction Multiple Thread (SIMT) model used by GPUs in all simulation stages, allowing a high level performance increase.

Keywords:: Smoothed Particle Hydrodynamics, Fluids, CUDA, GPU aceleration

Author's Contact:

{jricardo,esteban,anselmo}@ic.uff.br
pagliosa@facom.ufms.br

1 Introduction

Simulating natural phenomena like flow rivers or smoke has became popular in computer games and movie films. Using aerodynamics effects in races and flight simulation augment even more the immersion during its usage. These type of phenomena is simulated through solving fluid's governing equations with different attributes according to the desired effect. Due its important aspect in many areas, a new knowledge area called Computational Fluid Dynamics (CFD) has emerged for studying methods to solve these governing equations numerically. Even more, to not break the scene immersion, interaction with dynamic and static rigid bodies in the ambient where the fluid is being simulated must be considered, performing collision with them and computing forces coming from these rigid bodies and to these ones.

For solving its governing equations numerically, fluid simulation can be done by using Eulerian and/or Lagrangian approach. In the Eulerian approach, the domain where the simulation occurs is discretized into a fixed grid and variables like pressure and density are calculated in these cells. On the other hand, in Lagrangian approach, these variables are calculated in the material that moves with the flow, avoiding unnecessary processing in space where no fluid is being at. Additionally there is the ALE (Arbitrary Lagrangian-Eulerian) method, which is a mix between Eulerian and Lagrangian methods used in the same simulation.

The first real time fluid simulations were performed in CPU, most time without any interaction with ambient in order to achieve interactive frame rates. Later, with the horsepower provided by the GPU over CPU, additional phenomena could be performed in real time during fluid simulation like allowing more objects to be simulated and two-way interaction between them and its surrounded objects.

Unfortunately, in many works where the GPU is used for performing simulations like fluid and rigid bodies, the CPU is subutilized, being responsible only for coordinating what needs to be done by the GPU during the simulation, sitting and waiting it to be ready. In this case, some CPU's cores stay idle during the simulation, which could be used to make some useful processing while waiting for the GPU to be done. For many cases, this subutilization is due the fact long time required to do memory data transfer between CPU and GPU, leading a low overall system performance than doing all tasks using only GPU without data memory transfer or low data transfer.

In this paper we present a new architecture for simulating fluid and rigid body using a heterogeneous multicore CPU and GPU system, allowing two-way interaction between them, as shown in the simulation of fluid and rigid body in Figure 1. At the same time we fill the lack of GPU bounded architecture during collision detection, which in many times processes all elements in the scene at each frame without making any pre-sort of which ones need really to be processed. For performance increase, memory independent regular grids are used for fluid, static and dynamic rigid bodies with an efficient way communication between them. After search in literature, the author did not find any related work of using a heterogeneous system for performing this kind of simulation. In this work, the fluid simulation in done by using the Smoothed Particle Hydrodynamics (SPH), a mesh free particle Lagrangian based method.

The remainder of this paper is organized as follows. After referring to related works of fluid simulation in section 2, we describe the fluid and rigid body approach used in this work in section 3. In section 4 we describe acceleration data structures employed in the simulations. In section 5 we present our heterogeneous architecture of multicore CPU e GPU. In section 6 we show the results and, in section 7, the conclusions of the paper.

2 Related work

Physical simulation using Eulerian grid-based approach started by Foster and Metaxas [Foster and Metaxas 1996; Foster and Metaxas 1997] which are the first to propose solving the full 3D Navier-Stokes equations in order to re-create visual properties of dynamic fluids. Stam [Stam 1999] simulates dynamic gases using a semi-Lagrangian integration scheme that archives unconditional stability using artificial viscosity and rotational damping. Foster and Fedkiw [Foster and Fedkiw 2001] extended the technique to liquids using both a level-set method and particles inside the liquid. Enright et al. [Enright et al. 2002] added particles outside the fluid for free

surface tracking of the fluid.

Allowing rigid body two-way interaction, Takahashi et al. [Takahashi et al. 2002] presents a simple method for coupling between fluids and buoyant rigid bodies on regular grids using a combined Volume of Fluid method and Cubic Interpolated Propagation system. Génevaux et al. [Arash et al. 2003] uses marker particles for free surface representation and performs interaction with deformable rigid bodies represented as a collection of spring-mass set of particles. In [Cohen et al. 2010], the authors use a moving grid to simulate fluid. In the approach the grid moves together with the fluid flows, allowing the use of small grid with the benefit of being able to calculate the fluids variable anywhere in space.

After introduction of particle system by Reeves [Reeves 1983], natural phenomena that uses mesh less methods starts to appear. Fluid simulation using SPH is proposed by Desbrun and Gascuel [Desbrun and paule Gascuel 1996] for simulating deformable objects, augmenting it later [Stora et al. 1999] for allowing lava simulation through the viscosity coupling to temperature.

Müller et al. [Müller et al. 2003] uses the SPH method for fluid simulation in real time, deriving the density and viscosity forces from the Navier-Stokes equation. After this, the authors augment this work [Müller et al. 2004] by allowing fluid interaction with rigid bodies to simulate virtual surgery using a Gaussian quadrature to distribute ghost particles on rigid bodies surfaces, which are responsible for generating repulsive forces.

Kipfer and Westermann [Kipfer and Westermann 2006] use SPH to simulate fluid flows over deformable terrains in GPU using shader, without considering collision with rigid bodies. Kurose and Takahashi [Kurose and Takahashi 2009] simulate fluids and rigid bodies with two-way interaction between them using SPH in GPU. For the interaction, the authors discretize rigid body's polygons into a set of particles and solve Linear Complementary Problem (LCP) to solve collision forces between them.

3 Fluid and rigid body models

In this section, the fluid and rigid body models employed in our approach are explained. The model used for solving fluid's governing equations is based on the one proposed by Müller [Müller et al. 2003].

3.1 SPH model

Simulating fluids' behavior requires the solving of the equations

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}.\nabla\mathbf{v}\right) = -\nabla p + \rho\mathbf{g} + \mu\nabla^2\mathbf{v}, \tag{1}$$

and

$$\frac{\partial \rho}{\partial t} + \nabla.(\rho\mathbf{v}) = 0, \tag{2}$$

which is known as Navier-Stokes equation for modeling the flow of incompressible Newtonian fluids. In these equations, ρ represents the fluid's density, $\mathbf{v}$ the velocity field, p is the pressure field, $\mathbf{g}$ is the resultant of external forces (like gravity) and μ represents the fluid's viscosity.

Equation (1) is known as *equation of motion* and states that changes in linear momentum must be equal to all forces that act in the system. The convective term $\mathbf{v}.\nabla\mathbf{v}$ represents the change of a fluid's element properties that moves from one position to another, not used in Lagrangian methods as the material flows with the fluid.

Equation (2) is known as *continuity equation* or *mass conservation* and states that in absence of sinks and drains the mass in the system must be constant. For particle based method this equation is unnecessary as each particle carries a constant mass [Müller et al. 2003].

The Navier-Stokes equation is solved in this paper using the SPH method, introduced by Lucy [Lucy 1977] and Gingold and Monaghan [Gingold and Monaghan 1977] for simulation of astrophysical problems. The SPH is a mesh free Lagrangian particle based method that allows for field quantities defined only at discrete particles to be evaluated anywhere in space using a radial symmetrical smoothing kernel over a set of neighborhood particles inside a compact support radius [Monaghan 1992]. Evaluating a continuous field $A(\mathbf{x})$ scalar quantity is achieved by calculating a weighted summation of contributions for all particles $i \in [1...N]$, with position $\mathbf{x}_i$, mass m_i and additional attributes A_i as

$$A(\mathbf{x}) = \sum_j m_j \frac{A_j}{\rho_j} W(\mathbf{r}, h), \tag{3}$$

where ρ_i is the density of particle i, $\mathbf{r} = \mathbf{x} - \mathbf{x}_j$ and $W(\mathbf{r}, h)$ is the smoothing kernel. Computing the density of ρ_i can be evaluated using the equation (3) as

$$\rho_i = \rho(\mathbf{x}_i) = \sum_j m_j W(\mathbf{r}, h). \tag{4}$$

The kernel function $W(\mathbf{r}, h)$ must have a finite support, i.e. $\int W(\mathbf{r}, h)dr = 1$ and $W(\mathbf{r}, h) = 0$ for $|\mathbf{r}| > h$. According to [Liu and Liu 2003], the value of h must be chosen to allows for a particle i to have 5, 21 and 27 neighborhood in one, two and three dimensions, respectively. The gradient and Laplacian of a smoothed attribute function $A(\mathbf{x})$ is the gradient and Laplacian of the kernel function, respectively

$$\bigtriangledown A(\mathbf{x}) = \sum_j m_j \frac{A_j}{\rho_j} \bigtriangledown W(\mathbf{r}, h), \tag{5}$$

$$\bigtriangledown^2 A(\mathbf{x}) = \sum_j m_j \frac{A_j}{\rho_j} \bigtriangledown^2 W(\mathbf{r}, h). \tag{6}$$

Using equation (3) for calculating pressure and viscosity forces generates asymmetrical forces as can be seen when considering only two particles. To avoid this, pressure and viscosity forces are evaluated using the following functions

$$f_i^{pressure} = -\sum_j m_j \frac{p_i + p_j}{2\rho_j} \bigtriangledown W(\mathbf{r}, h), \tag{7}$$

$$f_i^{viscosity} = \mu \sum_j m_j \frac{v_j - v_i}{\rho_j} \bigtriangledown^2 W(\mathbf{r}, h), \tag{8}$$

where v_i is the particle's velocity and μ is the viscosity coefficient, respectively. The kernels used to compute these functions are the ones proposed in [Müller et al. 2003]. The pressure is computed using a modified ideal gas law state suggested by Desbrun [Desbrun and paule Gascuel 1996]

$$p_i = k(\rho_i - \rho_0), \tag{9}$$

where k is the stiffness constant of the fluid and ρ_0 is its rest density. At the end, the acceleration of particle i is computed as

$$a_i = \frac{f_i^{pressure} + f_i^{viscosity} + f_i^{external}}{\rho_i}. \tag{10}$$

In this paper, $f_i^{external}$ is the resultant force between the gravitational force and rigid body's acting forces during collision.

3.2 Rigid body model

A simple rigid body simulation basically only applies gravitational forces in their center of mass when any collisions occur. In this case, one of the most expensive tasks in simulating rigid body is checking for collision and performing calculations of the generated forces together with others information like normal of the contact point and depth of interpenetration between the colliders. In general, performing collision detection between rigid bodies is done through the use of polygons or a polygon's mathematical function representation. Collision detection using mathematical function is far away very fast than using polygonal collision as only an equation needs to be solved. Due its performance, the first step during collision detection is made by using a coarse representation of the whole rigid body in a polygon that can be evaluated mathematically, like a bounding sphere or bounding box. After these broad phase, rigid bodies that have passed goes through a narrow phase, where a more accurate collision is done, many times using triangle level [Moore and Wilhelms 1988]. At the same time, CUDA uses a SIMT (Single Instruction Multiple Thread) architecture, where an instruction is performed for a collection of different data elements. This way, solving the same function for collision detection using different data elements is even fast using CUDA.

For all the benefits listed, this work discretizes rigid body's polygons into a set of spheres with the same radius. As fluid elements are represented using particles, which in turns is a collection of spheres with the same radius, using spheres to represents rigid body's polygons allow for ease collision detection with rigid bodies and fluids, performing two-way interaction between fluid and rigid bodies.

Discretizing a polygonal model into a set of particles is made by using a modified of depth peeling [Everitt 2001], originally used for rendering transparent polygons. The technique has been also applied to various other operations like collision detection [Trapp and Döllner 2008] and polygonal discretization [Harada 2007]. In this work, we use a pre-processing step to discretize the rigid body's polygons into a set of particles, as shown in Figure 2. These particles are only used during collision detection with fluid and others rigid bodies in the scene. Collision response is made through repulsive forces using Discrete Elements Method (DEM) as explained in section 5.4.

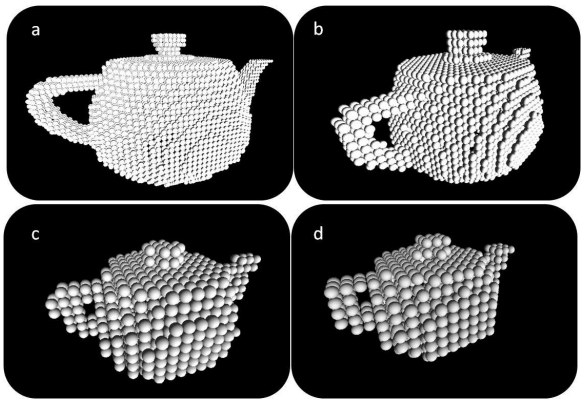

Figure 2: *Teapot processing using variable radius parameter: (a) 0.3 radius with com 7337 spheres; (b) 0.5 radius with 2674 spheres; (c) 0.9 radius with 600 spheres; (d) 1 radius with 668 spheres.*

4 Acceleration structure

SPH uses neighborhood particles to calculate fluid's variable using an interpolation function. In this case, the complexity is $O(n^2)$ for a set of n particles. To reduce this, a spatial subdivision structure needs to be used. In this work, a regular grid sized *R* is used, where *R* is the kernel compact support radius [Monaghan 1992] of particles interpolation. This way, only 27 grid cells needs to be checked in 3 dimension.

For rigid bodies, a subdivision data structure is also required, as it is represented by a set of particles for performing collision detection. Unfortunately, the rigid body's particle radius maybe not be the same as the one used to represents the fluid particle's volume. Allowing for two-way interaction between fluid and rigid body requires communication between these data structures.

One solution to maintain the search over 27 grid cells is to have a cell's size capable of storing the greatest rigid body volume in the scene. Unfortunately this will cause a degradation of the system as more particles will need to be computed for fluid processing when using a smaller radius than rigid body's particles radius.

To solve this problem, a new data structure presented in [da Silva Junior 2010] is used. In this work, the author presents a method that allows for independent regular unbounded hash grids localized at different memory space of GPU and CPU to communicate each other in a efficient way, which fits best for the purpose of the presented work.

During the simulation, three regular grids are used. First a regular grid for fluid's particles is used, responsible to sort particles based on their compact support radius. Another one is used for static rigid body's particles and finally a regular grid for dynamic rigid body's particles. The use of different grids for static and dynamic rigid body is based on the fact that static rigid body's particles did not change its cell location over the simulation, particles are fixed in space. In this case, only the fluid's and dynamic rigid body's grids need to be updated when necessary.

5 Heterogeneous architecture

Load balance over CPU and GPU is generally made for generic tasks [Joselli et al. 2009]. In this case, these tasks running in parallel must be independent of each other in order to avoid problems of data corruption over multiple threads. This fact makes hard using CPU and GPU together for solving a simulation. Also, sharing processing between CPU and GPU may slow down the overall system performance than using only GPU due data memory exchange.

In this section a new heterogeneous architecture using multiple CPU cores and GPU for fluid and rigid body simulation is presented, allowing two-way fluid and rigid body interactions, minimizing data exchange and concurrent data access during the simulation.

5.1 Load balance strategy between CPU/GPU

Simulating fluid and rigid body require the execution of some ordered tasks. To allow for two-way interaction between them, data interchanging between rigid body and fluid simulation stages is necessary as can be seen by dashed lines in Figure 3. This dependency may cause overall system performance degradation in case one of these steps takes too long time to process data necessary by a dependent stage. To minimize this, these tasks need to be well distributed over GPU and CPU cores in an heterogeneous system, considering the parallel GPU's power over CPU.

Usually, fluid simulation archiving two-way rigid body interaction, where rigid body is discretized into some sort of representation like particles is made in GPU by performing operations like classification and collision detection in all of these particles each frame [Zhang et al. 2008; Zhang et al. 2007; Kurose and Takahashi 2009; Harada 2007]. Considering for example, *400* rigid bodies discretized into *76* particles each, these operations need to be performed for *30400* particles including the fluid's particles, which needs to be in a great number for obtaining correct numerical values in SPH [Liu and Liu 2003].

Considering that in most scenes the fluid only occupies a small subset of the world in relation to rigid bodies, some optimization can be made to avoid the processing of all rigid body's particles each frame. This way, doing some sort of broad phase between rigid bodies and fluid and between rigid bodies itself may eliminate the processing of many particles or even the processing of some tasks, as can be seen in Figure 4. This optimization can be made as rigid body simulation only requires its center of mass integration, being

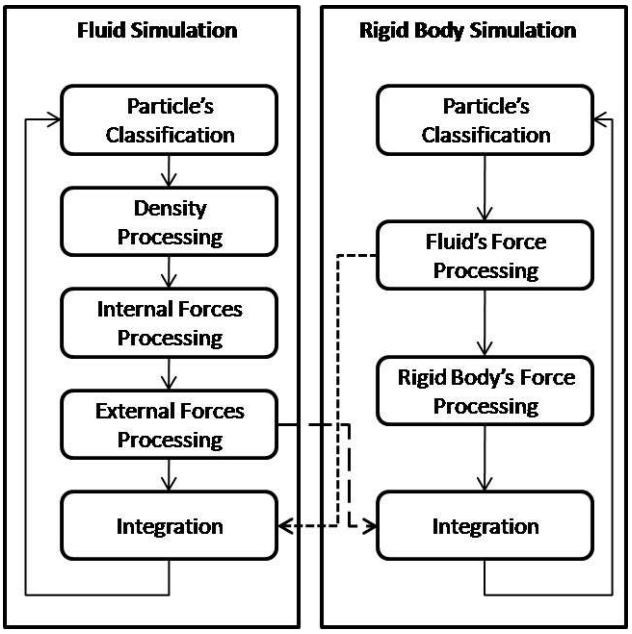

Figure 3: *Stages necessary for performing fluid and rigid body simulation and its dependency showed by the red lines.*

its particles only used for collision detection and response. This way, when any collisions occur between rigid bodies and fluid, the simulations can be processed independently of each other. On the other hand, in case of collision between them, only potential particles must take further processing for a more accurate collision and response calculation.

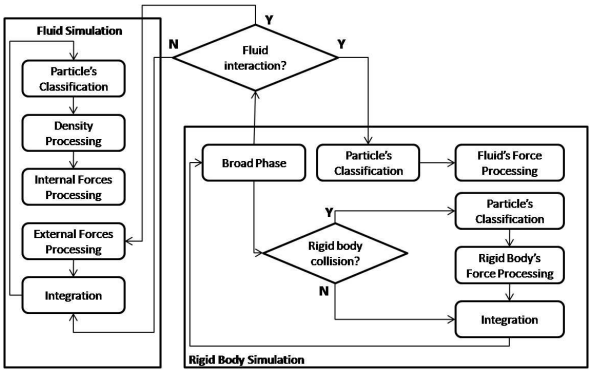

Figure 4: *Implemented model for fluid and rigid body simulation with two-way interaction.*

Figure 5 shows the workflow and tasks distribution between CPU and GPU in the proposed environment. As can be seen, many of these tasks are processed in GPU, due its capacity of execution in parallel, except for the broad phase, which is processed in parallel using multiple CPU cores during fluid's particles spatial subdivision. As can be seen by this figure, some tasks may not be performed according to results of the broad phase step. In this case, the broad phase is an important process for performance increase, as some tasks that may be avoided require a large amount of computational effort.

5.2 Broad phase

There are the following possibilities during collision detection: rigid body with fluid, rigid body with other rigid body only, and rigid body with rigid body and also fluid. By detecting these types of collision, the two-way interaction between rigid body and fluid and its associated tasks can be performed only when they are necessary.

For effectiveness, broad phase collision detection is made between fluid and rigid bodies by using an *axis aligned bounding box*

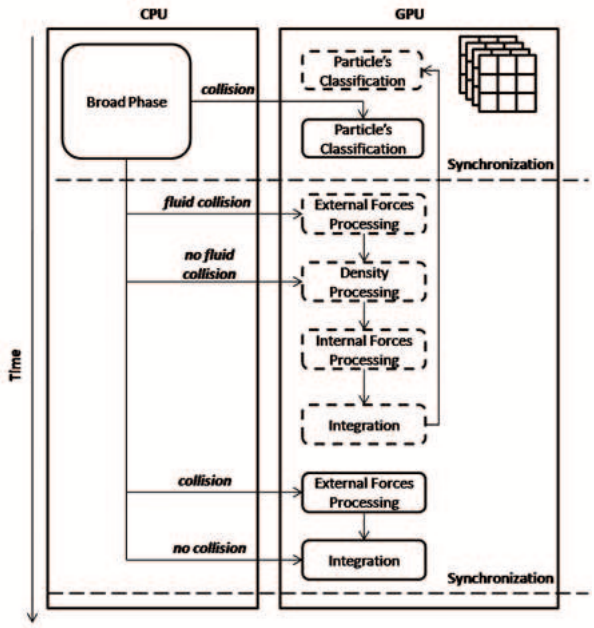

Figure 5: *Task distribution over GPU and CPU. Dashed blocks represents fluid's task while full blocks represents rigid body's task.*

(AABB) approach. For this, all the available CPU cores are used through the *OpenMP* library, with allows CPU multithread programming. During this, each CPU core is responsible for performing the collision detection in a subset of all rigid bodies in the scene and the fluid's volume using its bounding box. In case of collision, a flag is checked in a flag array, indicating if this collision was with a rigid body, fluid or both. For more clarity, this process is shown in Figure 6, where the dashed blocks store the number of collisions of each CPU core.

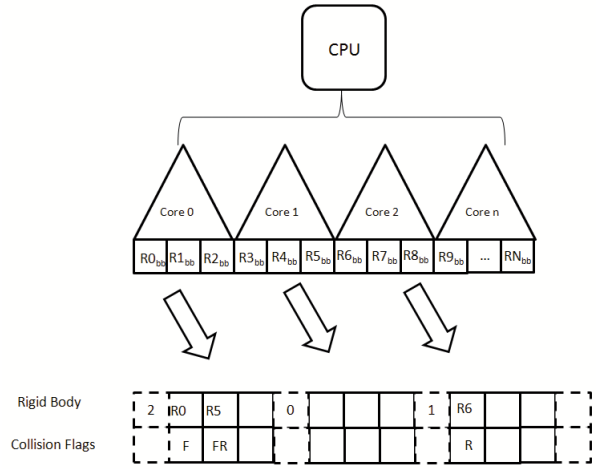

Figure 6: *Performing broad phase using multiple CPU's cores. Rk_{bb} represents the k-th rigid body's bounding box. F flag represent fluid collision while R flag represents rigid body collision.*

At the end of this process an array containing the potential rigid body colliders and a flag array indicating with type of collision has occurred is generated. Using these data, a position, offset and count array is created independently for the ones that collided with fluid and rigid body. These arrays are responsible to store the rigid body's relative particles position, the offset index for the starting particle set of each rigid body's in the position array and the particle number of each rigid body, respectively, as presented in Figure 7. Processing interaction between rigid bodies and fluid is only necessary in this array set instead of all elements presented in the simulation, as is commonly done.

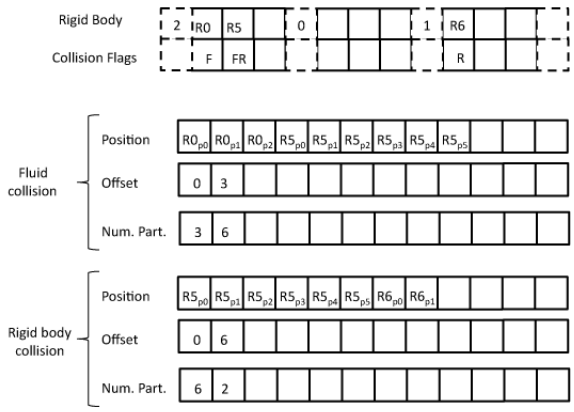

Figure 7: *Data array generated after broad phase stage in CPU. Rigid body's particles that collide with fluids are stored independently of the ones that collided with fluid.*

5.3 Fluid and rigid bodies collision

After broad phase step, rigid bodies' particles that collided with the fluid and other rigid bodies are stored in two distinct arrays. For narrow phase processing, these rigid bodies' particles need to be sorted using its own regular grid. As said before, three independent regular grids are used for this step, these particles are classified using the dynamic rigid bodies' regular grid.

First this classification is made for rigid bodies that may collided with the fluid and forces between them are solved. After that the classification is made based on the particles that collided with other rigid bodies, processing the collision interaction forces. In both steps, these particles need to be transformed from rigid body relative position to global position. This is done during classification through the use of rigid body's center of mass position and orientation. With these particles in global position, the collision is made by using the mathematical sphere equation

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r^2, \qquad (11)$$

which informs if collision occur and the depth of interpenetration.

5.4 External forces calculation

During fluid and rigid body simulation, external forces coming from collision as well the ones caused by gravity is computed. Also, as shown above, the two-way coupling between rigid body and fluid is only computed for the rigid body's particles that had collided in the broad phase step, allowing the fluid to continue its own processing in case any collisions occur.

These forces are calculated using the discrete elements method (DEM), which is used for simulate granular materials [Mishra 2003] like sands. The repulsive force f_{ij}, acting on particle i through interaction with particle j is computed using a spring force $f_{i,s}$

$$f_{i,s} = -k(t_{rad} - |r_{ij}|)\frac{r_{ij}}{|r_{ij}|}, \qquad (12)$$

and a damping force $f_{i,d}$

$$f_{i,d} = \eta v_{ij}, \qquad (13)$$

where k is the spring coefficient, η is the damping coefficient, r_{ij}, v_{ij} and t_{rad} represents the relative distance, relative velocity and i and j radius sum, respectively.

The forces generated during particles interaction need to obey third Newton's law, which states that for every action there is a reaction of the same magnitude but with opposite direction. So, the same approach presented here is used for fluid/rigid body and rigid body/rigid body interaction, storing each force in its respective particle.

5.5 Integration

After fluid and rigid body particles forces are computed, it is necessary to integrate these forces to compute the acceleration, velocity and position. In this paper, the explicit Eulerian approach is used for.

For fluid's particles, the internal and external forces previously computed are integrated for each particle, and its velocity and position variation is calculated.

For rigid bodies, another approach must be taken. Rigid bodies are discretized into a set of particle for performing collision. During collision, each particle stores its external force and torque amount coming from interaction with other particles. At the end, particle's forces and torque are added and the resulting force and torque applied directly in the rigid body's center of mass. In this work this is performing by using a scan operation [Harris et al. 2007] in GPU. However as rigid body's particles forces and torques are stored in a unique array, scan would add all forces from all rigid body's particles that had collided. To solve this, a segmented scan operation [Sengupta et al. 2007] is used instead scan. The segmented scan employs an auxiliary array indicating the starting of each segment in the array to be processed, doing the addition over all particles from different subsets of the whole array. This processing is done using the CUDPP library, which implements the most common operations in GPU using CUDA. Unfortunately, the kernel for segmented scan operation only enables primitive data as float and integer; so a modification in this library was made for allowing vector structure, as the one used for storing particle's forces. With this modification, doing a back warded inclusive segmented scan make the segment addition, storing its results in the first element of the segment, which result can be applied in the rigid body's center of mass.

6 Results

This section presents the results of the heterogeneous multi core CPU and GPU processing. For these tests, a PC equipped with an Intel Core 2 Quad Q6600 using 4 GB of RAM and a NVidia 9600 GT with 512 DDRAM was used.

For each simulation, a number of different configuration was used. In all simulation, the *teapot* model discretized into *76* particles is used and the amount of time are presented in seconds. Also, the results is presented without time to rendering, as some approach that can be used to increase it performance was not employed in this work.

First, Table 1 shows the simulation of rigid body and fluid made in CPU and GPU without considering interaction between them. In the table, the column labeled FPS represents the *frames per second* which measure a time necessary to update and render the simulation. As the graph in Figure 8 shows, increasing the number of fluid's particles causes almost an exponential increasing of time to process the simulation. On the other hand, increasing the number of rigid bodies makes almost a more linear increase in time to process the simulation.

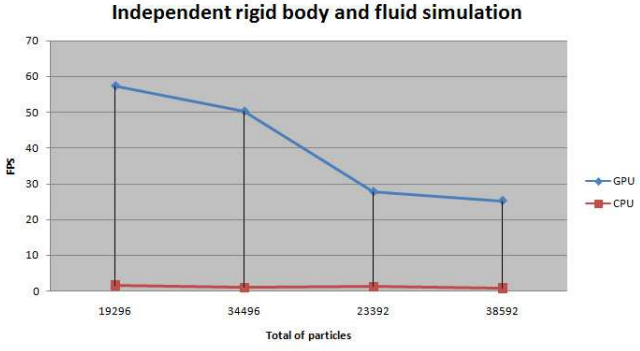

Figure 8: *Graph of rigid body and fluid simulation without interaction in CPU and GPU.*

Table 1: *Result of fluid and rigid body simulation without interaction between them. RB: number of rigid bodies; CRP: total of rigid body's particles; CF: total of fluid's particles; TPS: total of particle's system.*

				GPU		CPU		
CF	RB	CRP	TPS	FPS	Time	FPS	Time	Speedup
4096	200	15200	19296	57,4	0,0174	1,6	0,6028	35,87
4096	400	30400	34496	50,2	0,0199	0,9	1,0941	55,77
8192	200	15200	23392	27,7	0,0361	1,2	0,8045	23,08
8192	400	30400	38592	25,1	0,0397	0,7	1,3158	35,85

Using the new architecture of heterogeneous multi core CPU and GPU for processing fluid and rigid body simulation with two-way interaction between them is presented in Table 2 and its graph in Figure 9. As can be seen, the exponential increase time due fluid's particles increase was preserved in the new architecture but the overall system performance reach a speedup over two in relation GPU bound, as show in Table 3.

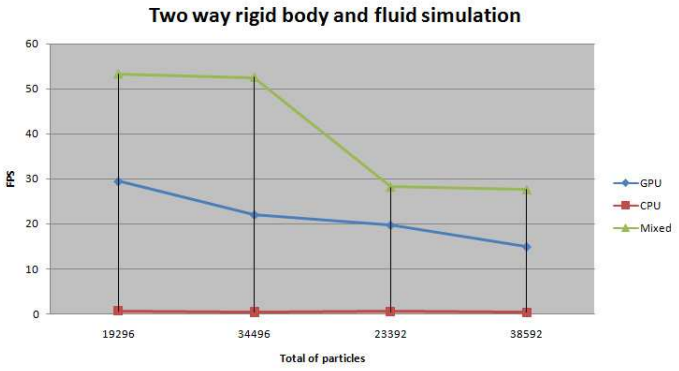

Figure 9: *Graph of rigid body and fluid simulation with two-way interaction in CPU, GPU and heterogeneous (GPU e CPU) system.*

Table 3: *Overall system increase using the heterogeneous architecture of multiple CPU cores and GPU over GPU bound. RB: number of rigid bodies; CRP: total of rigid body's particles; CF: total of fluid's particles; TPS: total of system's particles.*

CF	RB	CRP	TPS	GPU FPS	Het. FPS	Speedup
4096	200	15200	19296	29,5	53,4	1,81
4096	400	30400	34496	22,0	52,6	2,39
8192	200	15200	23392	19,7	28,2	1,43
8192	400	30400	38592	14,9	27,6	1,85

7 Conclusions and future works

As presented the simulation of fluid and rigid bodies using a heterogeneous system of multicore CPU and GPU over a GPU bound system increase the system performance almost in 60% during the simulation. It's due the fact that many complex and time consuming tasks can be avoided during the simulation processing using a broad phase step as done in this work. In this case, only rigid body that potentially collided with fluid and/or other rigid bodies needs further processing, performing two-way between fluid and rigid bodies only in the required one. In most cases these number is very low when compared with the amount of rigid bodies in the scene.

According to the results presented in this section, the proposed architecture can be extended to enable the use of multiple CUDA kernels using the new FERMI GPUs, which allows for more than one kernel to be executed at the same time. In this case, one kernel can be used to process fluid simulation and another one to process rigid body simulation at same time, processing interaction between them only when necessary. Also, the architecture of heterogeneous system can be extended to support a cluster of GPUs, allowing even more fluid's particles and rigid bodies to be processed in real time.

Using the rigid body's discretization in this work and the multiple regular grids, level of detail for processing rigid body collision could be used. In this case, rigid bodies which are in the simulation focus uses a large number of small particles to better perform collision detection and response, while rigid bodies that are not in the simulation focus can use fewer number of big particles to increase the simulation performance during the narrow phase processing.

Due the fact that rigid bodies are discretized into a set of particles, the simulation can be extended for simulating deformable models. Also rigid body melting and solidification can be implemented, allowing them coupling with fluid during these phase transition.

Visualization of fluid's free surface is also being developed, instead of using a collection of spheres, as shown by this work. The visualization will be presented using a point splat method.

8 Acknowledgements

The author thank all the Computation Institute at Federal Fluminense University for their support. Financial support from CAPES is acknowledged.

References

ARASH, O. E., GÈNEVAUX, O., HABIBI, A., AND MICHEL DISCHLER, J. 2003. Simulating fluid-solid interaction. In *in Graphics Interface*, 31–38.

COHEN, J. M., TARIQ, S., AND GREEN, S. 2010. Interactive fluid-particle simulation using translating eulerian grids. In *I3D '10: Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, ACM, New York, NY, USA, 15–22.

DA SILVA JUNIOR, J. R. 2010. *Simulação computacional em tempo real de fluidos utilizando o método SPH em ambiente heterogêneo CPU/GPU*. Master's thesis, Universidade Federal Fluminense.

DESBRUN, M., AND PAULE GASCUEL, M. 1996. Smoothed particles: A new paradigm for animating highly deformable bodies. In *In Computer Animation and Simulation ί96 (Proceedings of EG Workshop on Animation and Simulation*, Springer-Verlag, 61–76.

ENRIGHT, D., MARSCHNER, S., AND FEDKIW, R. 2002. Animation and rendering of complex water surfaces. *ACM Trans. Graph. 21*, 3, 736–744.

EVERITT, C. 2001. Interactive order-independent transparency. Tech. rep., NVidia Corporation.

FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 23–30.

FOSTER, N., AND METAXAS, D. 1996. Realistic animation of liquids. *Graph. Models Image Process. 58*, 5, 471–483.

FOSTER, N., AND METAXAS, D. 1997. Modeling the motion of a hot, turbulent gas. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 181–188.

Table 2: *Result of fluid and rigid body simulation with two-way interaction between them. RB: number of rigid bodies; CRP: total of rigid body's particles; CF: total of fluid's particles; TPS: total of particle's system.*

				GPU		CPU		Heterogeneous (CPU and GPU)	
CF	RB	CRP	TPS	FPS	Time	FPS	Time	FPS	Time
4096	200	15200	19296	29,5	0,0339	1,6	0,6028	53,4	0,0187
4096	400	30400	34496	22,0	0,0455	0,9	1,0941	52,6	0,0190
8192	200	15200	23392	19,7	0,0508	1,2	0,8045	28,2	0,0355
8192	400	30400	38592	14,9	0,0672	0,7	1,3158	27,6	0,0363

GINGOLD, R. A., AND MONAGHAN, J. J. 1977. Smoothed particle hydrodynamics - theory and application to non-spherical stars. In *Royal Astronomical Society, Monthly Notices, vol. 181*, 375–389.

HARADA, T. 2007. Real-time rigid body simulation on gpus. In *GPU Gems 3*, H. Nguyen, Ed. Addison-Wesley, 611–631.

HARRIS, M., SENGUPTA, S., AND OWENS, J. D. 2007. Parallel prefix sum (scan) with cuda. In *GPU Gems 3*, H. Nguyen, Ed. Addison Wesley, Aug.

JOSELLI, M., ZAMITH, M., CLUA, E., MONTENEGRO, A., LEAL-TOLEDO, R., CONCI, A., PAGLIOSA, P., VALENTE, L., AND FEIJÓ, B. 2009. An adaptative game loop architecture with automatic distribution of tasks between cpu and gpu. *Comput. Entertain. 7*, 4, 1–15.

KIPFER, P., AND WESTERMANN, R. 2006. Realistic and interactive simulation of rivers. In *GI '06: Proceedings of Graphics Interface 2006*, Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 41–48.

KUROSE, S., AND TAKAHASHI, S. 2009. Constraint-based simulation of interactions between fluids and unconstrained rigid bodies. In *Proceedings of Spring Conference on Computer Graphics*, 197–204.

LIU, G. R., AND LIU, M. B. 2003. *Smoothed Particle Hydrodynamics: A Meshfree Particle Method; electronic version*. World Scientific, Singapore.

LUCY, L. B. 1977. A numerical approach to the testing of the fission hypothesis. In *Astronomical Journal, vol. 82*, 1013–1024.

MISHRA, B. K. 2003. A review of computer simulation of tumbling mills by dem part i - contact mechanics. In *International Journal of Mineral Processing, Vol. 71(1-4)*, 73–93.

MONAGHAN, J. J. 1992. Smoothed particle hydrodynamics. In *Annual review of astronomy and astrophysics. Vol. 30*, 543–574.

MOORE, M., AND WILHELMS, J. 1988. Collision detection and response for computer animationr3. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 289–298.

MÜLLER, M., CHARYPAR, D., AND GROSS, M. 2003. Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 154–159.

MÜLLER, M., SCHIRM, S., TESCHNER, M., HEIDELBERGER, B., AND GROSS, M. 2004. Interaction of fluids with deformable solids. *Comput. Animat. Virtual Worlds 15*, 3-4, 159–171.

REEVES, W. T. 1983. Particle systems—a technique for modeling a class of fuzzy objects. *ACM Trans. Graph. 2*, 2, 91–108.

SENGUPTA, S., HARRIS, M., ZHANG, Y., AND OWENS, J. D. 2007. Scan primitives for gpu computing. In *Graphics Hardware 2007*, ACM, 97–106.

STAM, J. 1999. Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 121–128.

STORA, D., AGLIATI, P.-O., CANI, M.-P., NEYRET, F., AND GASCUEL, J.-D. 1999. Animating lava flows. In *Proceedings of the 1999 conference on Graphics interface '99*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 203–210.

TAKAHASHI, T., UEKI, H., KUNIMATSU, A., AND FUJII, H. 2002. The simulation of fluid-rigid body interaction. In *SIGGRAPH '02: ACM SIGGRAPH 2002 conference abstracts and applications*, ACM, New York, NY, USA, 266–266.

TRAPP, M., AND DÖLLNER, J. 2008. Real-time volumetric tests using layered depth images. In *Eurographics 2008*, The Eurographics Association, 235–238.

ZHANG, Y., SOLENTHALER, B., AND PAJAROLA, R. 2007. Gpu accelerated sph particle simulation and rendering. In *SIGGRAPH '07: ACM SIGGRAPH 2007 posters*, ACM, New York, NY, USA, 9.

ZHANG, Y., SOLENTHALER, B., AND PAJAROLA, R. 2008. Adaptive sampling and rendering of fluids on the gpu. In *Eurographics/IEEE VGTC Symposium on Point-Based Graphics*, 137–146.