International Journal of Scientific & Engineering Research, Volume 4, Issue 5, May -2013
ISSN 2229-5518

Smoke Dispersion Modelling Based on a GPU Computation

Sugeng Rianto[1,2], Arinto Y.P.Wardoyo.[2], Soemarno[1], M, Nurhuda[2]

Abstract—A computation of a 3D fluid flow simulation for virtual environment with user interaction can be a non-trivial issue. This is especially how to reach good performances and balancing between visualization, user feedback interaction, and computations. In this paper, we describe our approach of computation methods based on parallel programming on a GPU. The 3D fluid flow solvers have been developed for smoke dispersion simulation by using combinations of the cubic interpolated propagation (CIP) based fluid flow solvers and the advantages of the parallelism and programmability of the GPU. The fluid flow solver is generated in the GPU-CPU message passing scheme to get rapid development of user feedback modes for fluid dynamic data. A rapid solution in fluid flow solvers is developed by applying cubic interpolated propagation (CIP) fluid flow solvers. From this scheme, multiphase fluid flow equations can be solved simultaneously. To get more acceleration in the computation, the Navier-Stoke Equations (NSEs) is packed into channels of texel, where computation models are performed on pixels that can be considered to be a grid of cells. Therefore, despite of the complexity of the obstacle geometry, processing on multiple vertices and pixels can be done simultaneously in parallel. The data are also shared in global memory for CPU to control the haptic in providing kinaesthetic interaction and felling. The results show that GPU based parallel computation approaches provide effective simulation of compressible fluid flow model for real-time interaction in 3D computer graphic for PC platform. This report has shown the feasibility of a new approach of solving the fluid flow equations on the GPU. The experimental tests proved that smoke dispersion on various obstacles with user interactions on few model obstacles can be effectively and efficiently simulated on the reasonable frame rate with a realistic visualization. These results confirm that good performances and balancing between visualization, user feedback interaction, and computations can be applied successfully.

Index Terms—Compressible Fluid, GPU programming, Parallel Computation, Real-time Visualisation.

1 Introduction

In the two decades, the visualisations of volumetric real-world phenomena become more popular. This is not just to show the data of the research to be easily understood but also it is applicable in virtual reality environment and computer games, where the major goal of using such system are to provide "realistic" method to allow an easy creation of digital equivalents for natural phenomena (Rianto and Li, 2008, Zhang et al., 2008). Therefore, the demanding power of simulation and rendering increase continuously. In the case of compressible fluid such as: simulation and rendering the effects of explosion, cloud and fog, fire and smoke will drastically increase the realism level improving the immersiveness of the computer generated environment.

Other approaches, applying metamorphic modelling to visualise structures that exhibit similar statistical properties as those evolving in nature such as the works that have been done by: (Gardner, 1985) simulated and visualised clouds, (Ebert et al., 1994) used procedural approach for modelling, (Sakas and Westermann, 1992, Stam and Fiume, 1993) applied a functional approach to produce visual simulation of wind and gaseous turbulence.

- *Sugeng Rianto is currently pursuing doctoral degree in PDKLP in Universitas Brawijaya Indonesia,PH+62341571260, priantos@ub.ac.id*
- *ArintoYudi P.W., is univerity lecture in Department of Physic in Universitas Brawijaya Indonesia,PH+62341575833.*
- *Soemarno, is director of PDKLP in Universitas Brawijaya Indonesia,PH-+62341571260*
- *M. Nurhuda, is univerity lecture in Department of Physic in Universitas Brawijaya Indonesia,PH+62341575833.*

Researcher on other area worked on compressible flow using pre-computed density, velocity maps, and diffusion to explicitly specify the shape of large-scale structures in flames and wind fields (Perry and Picard, 1994, Stam and Fiume, 1995). All of these works show their beneficial properties and useful for realistically modelling small-scale variations in the simulated fields. However, this studies has problems in controlling the dynamic behaviour when showing the physically structures of the model being simulated.

Currently, a flexible programming interface for the powerful floating point hardware can be undertaken based on high performance graphics processing units (GPUs). GPUs are modern graphics cards that have a highly parallel nature. Several studies show that GPUs can already outperform CPUs and have better performance in computations (Semiconductor Industry Association, 2002, Khailany et al., 2003). These developments make GPUs are applied as high performance computational engines for floating point intensive numerical computations. GPUS also become a first commercially successful examples of a class of future computing architectures which may be the key to high performance, cost effective portable super computers (Owens et al., 2002, Khailany et al., 2001) that never exist before. Traditionally, the computational resources of GPUs have been used mostly to solve traditional graphics problems such as the shading models enhancement and their effects applied at the pixel levels (Olano, 2002). The current study on modern GPUs show their application not just for graphics but also far more general computation such as application that have been revealed by (Purcell et al., 2002, Carr et al., 2002) showing ray tracing engines based on the GPU pro-

International Journal of Scientific & Engineering Research Volume 4, Issue 5, May -2013
ISSN 2229-5518

gramming.

In recent years, it is possible to run real-time fluid dynamic computation in a portable system using GPU. For instance, (Matthias and ller, 2009) attempted to solve surface tracking for simulations and renderings of liquids with free surface based on GPU programming. This study claimed the efficiency of computation and memory consumption during direct control of volume and feature preservation of fluid simulation in computer graphics. Several studies also show that GPU has many advantages compared to CPU in the way of calculations. Besides better visualisation power, GPU supports efficient computations with its parallelism and programmability. GPU is now widely applied, not only for generating better visualisations, but also for high-performance computations.

The application of GPU for flow simulation was introduced by (W.Li et al., 2003b). This study not only focuses on visual fidelity as in (Stam and Fiume, 1993, Stam, 1999, Foster and Metaxas, 1996, Mulller et al., 2005, Premože et al., 2003, Losasso et al., 2006), but also attempts to accelerate the flow simulation to real time, while maintaining physical accuracy. Moreover, in (Liu et al., 2004, Harris, 2004, Crane et al., 2007b) an advanced study on fluid dynamic computation is conducted to achieve parallelisms for more efficient computations and real time rendering based on GPU. Other methods (Harada et al., 2007) (Figure 2.4d) (Kolb and Cuntz, 2005) implement algorithm on GPU to compute a smoothed particle hydrodynamic (SPH) scheme. This study claim that their algorithm enable to implement the SPH simulation entirely on GPUs and increase the simulation speed many times compare to CPU processing. While all these studies have demonstrated significant improvement in fluid visualisation and user interactivity, there are only few studies that apply this advancement that focus on compressible fluid and virtual reality application that involving 3D interactions such as force feedback in the fluid visualisations.

In this paper we describe the mapping of four fundamental works: developing compressible fluid flow solver based on CIP scheme that is adapted from (Kim et al., 2008), paralysation based on GPU programming (Venetillo and Celes, 2007, Crane et al., 2007a), adaptive sampling of fluid particle, efficient simulation and visualisation of compressible fluid (Zhang et al., 2008). All of them are workhorses of physical modelling and optimization applications and this demonstrate excellent performance on GeForce GT mobile hardware in realistic challenge applications.

2. Related Work

The modeling of fluid dynamic has received much attention in the computer graphics society in the last few decades. Although, there are many good works related to this field, this paper only introduce some recent closely related works.

Firstly, a fluid modeling with interactive animation was pioneered by (Stam and Fiume, 1993). This study simulated a turbulence of smoke based on advection-diffusion model to animate particle-based gaseous phenomena. He claims that efficient animations and renderings fluid motion can be reached by a clustering algorithm. Few years later, this work had been extended by (Foster and Metaxas, 1996) where a more complex behavior on fluid animation is applied. However, those works suffering from blowing when a bigger time step is applied. A semi-Lagrangian method with unconditionally stable fluid simulation, then, was introduced by (Stam, 1999) to overcome that problem. However this scheme is still suffering from numerical dissipation. Some other researches to augment fluid modeling have been continuously developed for various applications such as particle-based fluid animation (Mulller et al., 2005), turbulent water over natural terrain, and melting and burning (Losasso et al., 2006). All of these studies mainly discuss on the ways of finding efficient NSEs solution of motion for liquid with realistic looking behaviors for practical animations.

In recent years the introduction of graphic processor unit (GPU) has also embossed fluid modeling in several ways. Beside this device can provide better visualizations, it also can support efficient computations with its parallelism and programmability. The application of GPU for flow simulation was introduced by (W.Li et al., 2003a) (W.Li et al., 2003b). This study not only focuses on visual fidelity as done in (Stam and Fiume, 1993, Stam, 1999, Foster and Metaxas, 1996, Mulller et al., 2005, Losasso et al., 2006), but also attempts to accelerate the flow simulation in real time speed, while maintaining physical accuracy. In (Liu et al., 2004, Harris, 2004), an advanced study on fluid dynamic computation is done to get parallelisms for more efficient computations and realistic animations. Moreover, the GPU applications for interactive fluid motion in terrain rendering with erosions (Benes et al., 2006, Mei et al., 2007, Anh et al., 2007) also become a current research interest in computer graphics and animation. Although, these studies shows potential outcome in increasing the visual quality and the user interactivity, all of them have not involved force feedback interactions yet.

Although, many studies have done to explore force feedback design and computations for haptic instruments, there only a few that concern with force feedback design for fluid media. Haptic rendering method was introduced by (Avila and Sobierajski, 1996). This study, then, inspired many applications such as: surgical cutting or trimming (Thomas V and Cohen, 1999), sculpting (Kim and Park, 2004), and volume visualization (Lundin et al., 2005a). Whereas a force feedback integration with interactive fluid model has been introduced in (Baxter and Lin, 2004). This method enables force and torque generations in virtual painting applications. There is no other study exploring this field except in (Lundin et al., 2005b) for presenting fluid dynamic data and in (Bhasin et al., 2005) for simulating droplet fluid. Both of them actually only provide visualization rather than force feedback interaction into fluid models.

This section will review only fluid dynamic solver based on a cubic interpolated propagation scheme. The cubic interpolated propagation (CIP) method was firstly introduced in (Yabe et al., 1991) as a universal solver for hyperbolic equation by cubic interpolation. It is then used for solving fluid dynamic equation as a conservative semi-Lagrangian solver for a solid, liquid and gas. Some studies prove that this method is efficient enough to solve hydrodynamic equation in all states

International Journal of Scientific & Engineering Research Volume 4, Issue 5, May -2013
ISSN 2229-5518

(Yabe et al., 2002). More advance CIP study (Song et al., 2005) attempts to improve the *stable fluid* (Stam, 1999) from numerical dissipation. This scheme is not just applicable for dissipative media (like fog and smoke), but also non-dissipative liquid (water). Lastly, although the CIP has been accepted as an alternative method for fluid dynamic solver that fits the need of computer animations, to the best of author knowledge, there is no other study to date that uses this method in virtual environments with haptic interactions.

This paper introduce a framework which combines CIP and parallelism in GPU, trying to achieve the most efficient solution of fluid dynamic equations for real time rendering in virtual environment with haptic interaction.

The rest of the paper is organized as follows: description of fluid computation background; GPU implementation for visual rendering and fluid solver computation; the experimental setups and the results; and finally concludes the paper.

3. Teoretical Background

3.1 Fluid flow solver computation

The fluid dynamic equations usually are described by Navier-Stokes equations (NSEs). In Newtonian fluids, the motion equations are derived from a combination of the transport of a *momentum* in the fluid;

$$\frac{\partial u}{\partial t} = -(u \cdot \nabla)u - \frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla^2 u + F_c \quad (1)$$

and mass conservation function;

$$\nabla \cdot u = 0 \quad (2)$$

where u is a velocity field of fluid, p is a pressure, ρ is the density of the fluid, μ is a viscosity coefficient, F_c represents external force (the gravity, forces delivered through haptic), and ∇ is the differential operator, respectively. Then, we can predict the behavior of a fluid by solving (1) and (2), then the flow simulation is run in staggered grids. The grids define velocity components at cell faces and scalar variables in cell centers.

Assume that φ is the function of the volume of fluid fraction, the fraction of fluid volume in each cell is transported using advection equation

$$\frac{\partial \varphi}{\partial t} = -(u \bullet \nabla)\varphi \quad (3)$$

This equation can be solved by using a combination of the CIP method and the advection form ($-(u \cdot \nabla)u$). Moreover, the interface between liquid and solid is traced by distributing (4) into advection phase (6) and non-advection (5).

$$\frac{\partial \varphi}{\partial t} + u \cdot \nabla \varphi = g \quad (4)$$

$$\frac{\partial \varphi}{\partial t} = g \quad (5)$$

$$\frac{\partial \varphi}{\partial t} + u \cdot \nabla \varphi = 0 \quad (6)$$

Where $g = -(u \cdot \nabla) f$. The solution of those equations in the spatial domain (x, y, z) can be performed by solving the non-advective part (5) using a finite difference and then advect the result as shown in (6). Numerical solution for this, on the other hand, can be done by partition the space into a grid of cells.

In the CIP scheme the advection phase is computed by shifting a cubic interpolated profile into space according to the total derivative equations. Equation (6) is also called as level set equation (Song et al., 2005), where the surface of liquid can be obtained by tracking the locations for which $\varphi = 0$. As the exact solution of (6) is

$$\varphi(x,t) = \varphi(x - ut, 0) \quad (7)$$

If the velocity is assumed to have a constant value within a short time, we get

$$\varphi(x, t + \Delta t) \cong \varphi(x - u\Delta t, t) \quad (8)$$

It is noted that the CIP method uses not only the function values at the grid points, but also the spatial derivatives at those points for constructing the profile inside the grid cell. Here $x\text{-}u\Delta t$ is not always located on grid points and interpolated locally by Hermite polynomials within the calculation grid point as the value for the next time step. Within the discretised function a cubic-Hermite polynomial can be expressed as (9).

$$F_j(x) = [a_j X - b_j)X + \Delta x \varphi_j^{'}]X + \varphi_j \quad (9)$$

where

$$a_j = \Delta x(\varphi_j^{'} + \varphi_{j+1}^{'}) - 2(\varphi_{j+1} - \varphi_j)$$

$$b_j = \Delta x(\varphi_j^{'} + 2\varphi_{j+1}^{'}) - 3(\varphi_{j+1} - \varphi_j)$$

and

$$X = \frac{x - x_j}{\Delta x}$$

$$\Delta x = x_j - x_{j-1}$$

However, using cubic-Hermite profile (9) can lead to instabilities. A modification as proposed in (Song et al., 2005) can solve this problem and is claimed always stable. This scheme implement higher dimensional CIPs based on the monotonic CIP solver (see (Song et al., 2005) for detail).

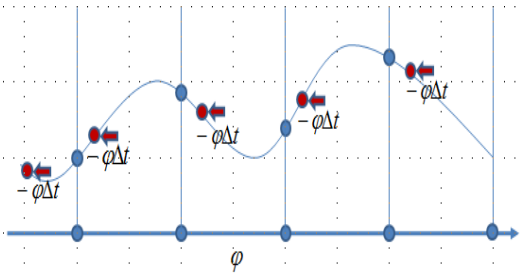


Fig. 1. The physical value estimation in advection phase on the departure point of the arbitrary contour.

The CIP scheme uses a spatial profile that rely on different interpolations to propagate the solution along the characteristics (Yabe et al., 2004). The physical value on φ_t is calculated locally within the calculation cell (the original equation). Then, an estimate value is directly advected toward the grid point

International Journal of Scientific & Engineering Research Volume 4, Issue 5, May -2013
ISSN 2229-5518

calculations at the next time step value (Fig. 1). This results is more accurate modeling of the real situation, even in fairly coarse grid. While other methods require multi-points to construct the profile in the one dimensional case, the CIP can be constructed only from a single cell. This advantage is useful for treating boundaries. Beside the CIP method provides a stable result, less diffusive, and has third order accuracy (Utsumi et al., 1997, Song et al., 2005); the numerical solvers of fluid motion can provide the solutions simultaneously to all interfaces.

3.2 Fluid force and user interaction

The external forces F_c from (1) consist of gravity F_g, surface tension f_t, and a user interactive force (haptic) F_u. These forces can be expressed as

$$F_c = F_g + f_t + F_u \tag{10}$$

The gravitational force is equal to ρg, where g is gravitational acceleration. When we treat the surface tension as a body force and does not require explicit information on the geometry and position of the liquid surface, the surface tension can be expressed as continuum surface force as proposed in (Brackbill et al., 1992) as follow:

$$f_t = -\sigma\rho k(\varphi)\delta_\varsigma(\varphi)\nabla(\varphi) \tag{11}$$

where σ and $k(\varphi)$ are surface constant coefficient and local curvature value respectively, and $\delta(\varphi)$ delta function that is formulated as:

$$\delta_\varsigma(\varphi) = \begin{cases} \frac{1}{2\varsigma}(1+\cos\left(\frac{\pi\varphi}{\varsigma}\right) ==> |\varphi| \le \varsigma \\ 0 ==> |\varphi| > \varsigma \end{cases} \tag{12}$$

When the simulation time steps are small enough, this force has less visually significant. Interactive force from user through a haptic device can be formulated as the equation of motion for a rigid body as follow:

$$m\ddot{x} = g - \int pnds \tag{13}$$

where m and x are the mass and the centre of gravity of the rigid body respectively. Then, the moment (**I**) that act on the centre of the mass is given by:

$$I\dot{\omega} = \int (r \times pnds) \tag{14}$$

r is the cell positions, $\dot{\omega}$ is angular velocity vector, s is the surface area of the marked cells, and p is the pressure of the cell.

Lastly, from (1) we need to solve the pressure of the fluid. The pressure projection is solved using the Poisson's equation with assumption that the liquid has no viscosity and has an external force that is determined by the force released from/to haptic interactions. After the volume of fluid fraction of each cell is obtained in the simulation, a fluid surface is created and it is smoothed by subdivision, if necessary. The detail explanation of this method can be described below. Assume that u' is the velocity result obtained by processing Equation (1), and then the pressure can be expressed as the Poisson equation,

$$\nabla \cdot \left(\frac{\nabla p}{\rho}\right) = \frac{\nabla \cdot u'}{\Delta t} \tag{15}$$

The (15) can be discretised as

$$\sum_{\psi=(i,j,k)} (\rho^{-1}_{\psi+\frac{1}{2}} + \rho^{-1}_{\psi-\frac{1}{2}}) p_\psi - (\rho^{-1}_{\psi+\frac{1}{2}} p_{\psi+1} + \rho^{-1}_{\psi-\frac{1}{2}} p_{\psi-1}) =$$

$$-\frac{1}{\Delta t} \sum_{\psi=(i,j,k)} (u'_{\psi+\frac{1}{2}} - u'_{\psi-\frac{1}{2}}) \tag{16}$$

Here, velocity and density values are taken from cell faces, whereas, the pressure values are taken from the centers of neighboring cells (see Fig. 2).

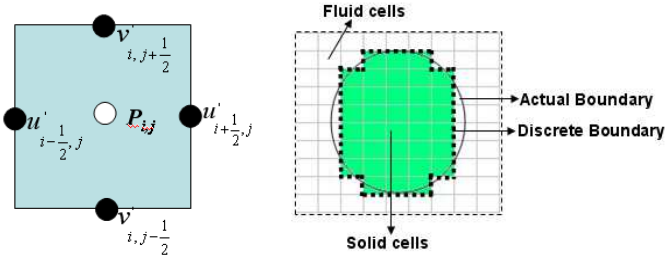


Fig. 2. (a) A typical composition of a grid cells. P is pressure, u and v are velocity in x and y axis respectively. (b) A solid geometry in relation with boundary discretisation and composition of a grid cells.

4. GPU SETUP FOR VISUAL RENDERING AND FLUID SOLVER COMPUTATION

4.1. GPU programming

The computation on GPU is done by mapping the formulation domain directly to texture memories as explained in (Mei et al., 2007, Liu et al., 2004, Harris, 2004, Crane et al., 2007a). All fluid solver described above is mapped in combination of vertex and fragment programs. Cells attributes (such as pressure, velocity, density) are stored in several 3D textures. These values, then, are updated in each simulation step by running computation kernels over the grid. This computation is implemented as a vertex shader that executes on every cell in the grid and writes the results to an output texture. Since GPUs are designed to render into 2D buffer, the execution must be done for each slice of a 3D volume by iterating slice indices over entire grid.

The momentum equation computations are solved by splitting the equations into advection and non advection equation groups. The advection equations consist of velocity, density, and level set advections, whereas non advection equations have pressure projection and diffusions. The detail implementations of the corresponding kernels refer to (Harris, 2004) with modification in the contained equations to be solved.

4.2. Volume Visualization

The visual appearance of the volume object being simulated is generated using direct volume rendering technique with opacity peeling as suggested in (Malik et al., 2007, Rezk-Salama and Kolb, 2006). This approach can occlude and show the object of interest better and faster than other segmentation techniques.

International Journal of Scientific & Engineering Research Volume 4, Issue 5, May -2013
ISSN 2229-5518

5. Experimental Setup

The experiments are implemented using the system with space mouse with six degrees of freedom (DOF), an NVDIA Quadro GTX8800/512Mb and 400 MHz RAMDAC Clock Speed GPU and dual display running on a PC (PIV Dual Core 3.2 GHz, 2GB RAM, 80GB HD). The volume rendering demonstration was generated through a combination of direct volume rendering and opacity peeling to show a bounding surface, velocity, viscosity, pressure gradient, and friction.

Experiments were created from a computer generated environments. The test results demonstrated the efficiency and effectiveness of our proposed framework for a 3D fluid dynamic simulation in real time interactions with user interaction. To support the immersion and kinesthetic felling, the 3D space mouse with dual display were used. The visual rendering is developed based on Visual Studio C++ with NVIDIA CUDA and HellHeaven-Fx Tools. Mostly All computations in this study use 32-bit floating points on programmable graphics hardware that is assumed suitable to real-world problems.

6. Results and Discussion

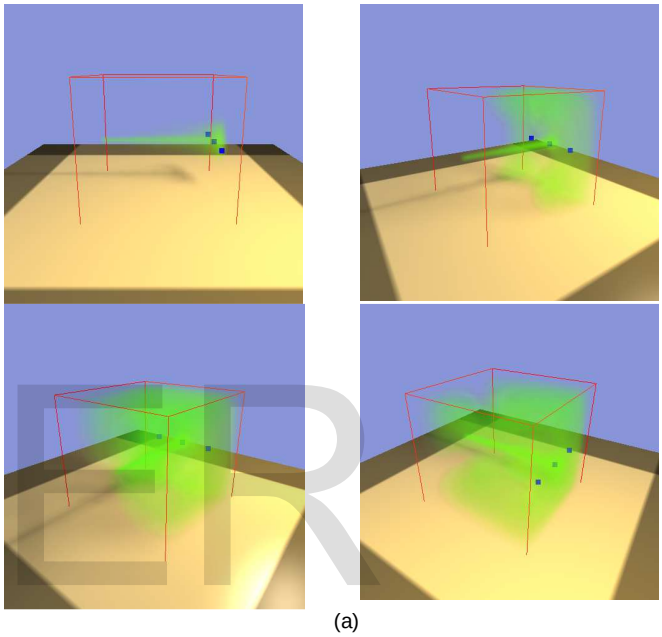
The experimental results utilised PC in combination with commodity 3D graphics hardware (***Note:*** the common streaming graphics processor is called as Graphics Processor Unit or GPU) as specified above) show high quality volume visualisations as shown in Table 1. The table shows the average frame rates data of visuals rendering, real-time-rendering and user interaction, and the comparison of frame rate between rendering with and without using the GPU for the 3D space with smoke emitter.

The results also shows stable visualisation for virtual reality environment application, felling the smoke, and sensing the number of PM (Fig. 3). These demonstrations had released more than 1000 particles every time computation steps and were visualised direcly on the display. The snapsots of various real-time smoke dispersion from souces and fires can be seen in Fig. 3 and 4. In Fig. 4a and 4b were emmited from sources (4a. single souce; 4b. few sources) and disperse continously to the roof,wheas Fig. 4c and 4d demonstrated fire injector with smoke dispersion from fires. Fig. 4e demonstrated few big fires that release thik smoke and disperse continousely. This study examines real-time computation with 3D visualisation in several different scenarios. The average performance observed during the experiment can be seen in the Table 1. The data is observed based on the visual frame rate per second (fps) during real-time computation and visual rendering simultaneously.

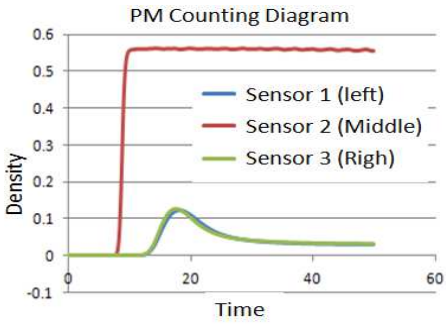
From Tabel 1 can be summarised as follows: Fluid-solid interaction in this project describes a fluid simulation to respond realistically to a moving polygonal object (wall, sensor,space mouse). The simulator can emit smoke interactivelly; the source can be normal emitter or injector. Normal emitter (continously) can be done application one time press to the mouse button, however continuous pressing will lead to emmiter to be an injector where the releasing smoke speed is double every second. The data also show that GPU implementation increase the visual rendering performance that also mean that the computation performance increase somehow (about four time in average).

Table 1: Visual Performance for Real-time Computation (GPU vs CPU)

Type of Simulation	Performance (Fps)	
	CPU	GPU
Smoke with UserInteraction	14	58
Smoke emitters	15	60
High speed smoke Injection	15	60
Complex obstacle	13	58



(a)



(b)

Fig.3. Smoke injection in cube space. a) Smoke are injected into space with three PM counter sensors; b). The density number of PM between sensor.

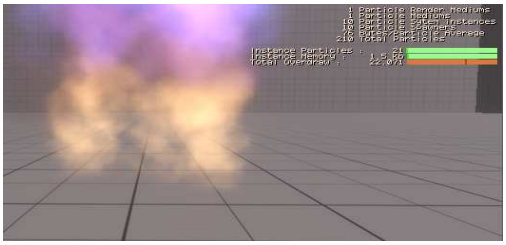
7. Conclusion

This paper offers a concept of an approximation fluid dynamic computation for fluid dynamic modeling in virtual reality application. This work focuses on the development of the real-time fluid dynamic simulation in three dimensional virtual reality environments, particularly for smoke dispersion models. The simulation and modeling are desktop based applications that compatible with parallel computations. To generate more realism fluid effects in virtual environment

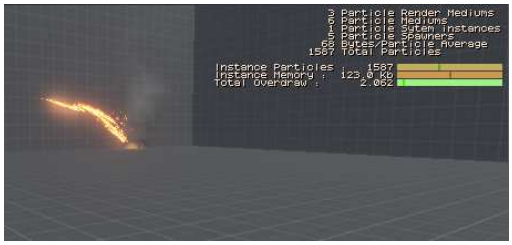
International Journal of Scientific & Engineering Research Volume 4, Issue 5, May -2013
ISSN 2229-5518

simulator, the system is designed based on the CPU-GPU balancing. By taking advantages of GPU parallelism in coupling with CIP fluid solver that solve multiphase fluid simultaneously, the fluid dynamic simulation for smoke dispersion can be created in real-time in 3D virtual reality environment without involving supercomputer.


(a)


(b)


(c)


(d)


(c)

Fig.4. Smoke and fire demonstration in various scenarios. a) smoke from single emitter; b) Smoke from 2 source; c),d) fire injection with smoke; e) Bigger fire smoke emissions from 3 sources.

The results demonstrated that the potential application for designing plausible fluid dynamic (smoke dispersion) simulator with complex fluid effects in virtual reality simulator with various 3D interactions. It is proved that the basic need of smoke dispersion effects in virtual reality environment can be created such as: smoke source emission, PM counting, smoke decompositions, condensation, flow rate and obstacle deposition. Those applications can be effectively simulated on realistic visualization on reasonable frame rate with real time interactions. More results in smoke dispersion modelling application will be explored in the future studies such as: multi-fluid interactions, architectur and building design, and other complex solid-fluid interactions such as smoke blending, high pressure, and supersonic application.

REFERENCES

ANH, N. H., SOURIN, A. & ASWANI, P. 2007. Physically based hydraulic erosion simulation on graphics processing unit. *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia.* Perth, Australia: ACM New York, NY, USA.

AVILA, R. S. & SOBIERAJSKI, L. M. Year. A haptic interaction method for volume visualization. *In:* Visualization '96. Proceedings., 1996. 197-204, 485.

BAXTER, W. & LIN, M. C. 2004. Haptic interaction with fluid media. *Proceedings of the 2004 conference on Graphics interface.* London, Ontario, Canada: Canadian Human-Computer Communications Society.

BENES, B., TENSINSKY, V., HORNYS, J. & S.K.BHATIA 2006. Hydraulic erosion. *Computer Animation and Virtual Worlds*, 17**,** 99-108.

BHASIN, Y., LIU, A. & BOWYER, M. 2005. Simulating Surgical Incisions Without Polygon Subdivision. *IOS Press*, 111**,** 43-49.

BRACKBILL, J. U., KOTHE, D. B. & ZEMACH, C. 1992. A continuum method for modeling surface tension. *J. Comput. Phys.*, 100**,** 335-354.

CARR, N. A., HALL, J. D. & HART, J. C. Year. The Ray Engine. *In:* In SIGGRAPH/Eurographics Workshop on Graphics Hardware, 2002.

CRANE, K., LAMAS, I. & TARIQ, S. 2007a. Real-Time Simulation and Rendering of 3D Fluids. *In:* NGUYEN, H. (ed.) *GPU Gems 3.* Addison-Wesley Professional.

CRANE, K., LLAMAS, I. & TARIQ, S. 2007b. Real-Time Simulation and Rendering of 3D Fluids. *In:* NGUYEN, H. (ed.) *GPU Gems 3 1 Book Cased (Hardback) plus DVD (1 ISBN).* Addison-Wesley.

EBERT, D., MUSGRAVE, K., PEACHEY, D., PERLIN, K. & WORLEY, M. 1994. *Texturing and Modeling: A Procedural Approach*, Academic Press.

FOSTER, N. & METAXAS, D. 1996. Controlling Realistic Animation.

GARDNER, G. Year. Visual simulation of clouds. *In:* In ACM Computer Graphics (Proc. SIGGRAPH '85), 1985. 297-384.

HARADA, T., KOSHIZUKA, S. & KAWAGUCHI, Y. 2007. Smoothed particle hydrodynamics on GPUs. *The Visual Computer manuscript.*

HARRIS, M. J. 2004. Fast Fluid Dynamics Simulation on the GPU. *In:* FERNANDO, R. (ed.) *GPU Gems - Triks for Real-Time Graphics.* Addison Wesley.

KHAILANY, B., DALLY, W. J. & CHANG, A. 2001. Imagine: Media

International Journal of Scientific & Engineering Research Volume 4, Issue 5, May -2013
ISSN 2229-5518

Processing with Streams. *IEEE Micro*, 21**,** 35-46.

KHAILANY, B., DALLY, W. J., RIXNER, S., KAPASI, U. J., OWENS, J. D. & TOWLES, B. Year. Exploring the VLSI Scalability of Stream Processors. *In:* In Proceedings of the Ninth Symposium on High Performance Computer Architecture, 2003.

KIM, D., SONG, O.-Y. & KO, H.-S. 2008. A Semi-Lagrangian CIP Fluid Solver without Dimensional Splitting. *Computer Graphics Forum (Proc. Eurographics)*, 27**,** 467-475.

KIM, L. & PARK, S. H. 2004. A Haptic Sculpting Technique Based on Volumetric Representation. *In:* PERALES, F. J. & DRAPER, B. A. (eds.) *Lecture Notes in Computer Science*. 3179 ed.: Springer-Verlag.

KOLB, A. & CUNTZ, N. Year. Dynamic particle coupling for GPU-based fluid simulation. *In:* Proc. 18th Symposium on Simulation Technique, 2005 Erlangen, Germany. ASIM, 722-727.

LIU, Y., LIU, X. & WU, E. Year. Real-time 3D Fluid Simulation on GPU with Complex Obstacles. *In:* Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on, 2004. 247-256.

LOSASSO, F., IRVING, G., GUENDELMAN, E. & FEDKIW, R. 2006. Melting and Burning Solids into Liquids and Gases. *Visualization and Computer Graphics, IEEE Transactions on*, 12**,** 343-352.

LUNDIN, K., GUDMUNDSSON, B. & YNNERMAN, A. Year. General proxy-based haptics for volume visualization. *In:* First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC), 2005a. 557-560.

LUNDIN, K., SILLEN, M., COOPER, M. & YNNERMAN, A. Year. Haptic Visualization of Computational Fluid Dynamics Data Using Reactive Forces. *In:* the Conference on Visualization and Data Analysis, part of IS&T/SPIE Symposium on Electronic Imaging January 2005b San Jose, CA USA.

MALIK, M. M., MOLLER, T. & GROLLER, M. E. 2007. Feature peeling. Institute of Computer Graphics and Algorithms Vienna University of Technology.

MATTHIAS, M. & LLER 2009. Fast and robust tracking of fluid surfaces. *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* New Orleans, Louisiana: ACM.

MEI, X., DECAUDIN, P. & HU, B.-G. Year. Fast Hydraulic Erosion Simulation and Visualization on GPU. *In:* DECAUDIN, P., ed. Computer Graphics and Applications, 2007. PG '07. 15th Pacific Conference on, 2007. 47-56.

MULLLER, M., SOLENTHALER, B., KEISER, R. & GROSS, M. 2005. Particle-based fluid-fluid interaction. *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation.* Los Angeles, California: ACM Press.

OLANO, M. 2002. Real-Time Shading Languages. *Course Notes. ACM SIGGRAPH.*

OWENS, J. D., KHAILANY, B., TOWLES, B. & DALLY, W. J. Year. Comparing Reyes and OpenGL on a Stream Architecture. *In:* In SIGGRAPH/Eurographics Workshop on Graphics Hardware, 2002. 47-56.

PERRY, G. & PICARD, R. Year. Synthesizing flames and their spread. *In:* SIGGRAPH '94 Technical Sketch Notes, 1994.

PREMOŽE, S., TASDIZEN, T., BIGLER, J., LEFOHN, A. & WHITAKER, R. T. 2003. Particle-Based Simulation of Fluids. *EUROGRAPHICS 2003 / P. Brunet and D. Fellner (Guest Editors)*, 22**,** 401-410.

PURCELL, T. J., BUCK, I., MARK, W. R. & HANRAHAN, P. 2002. Ray Tracing on Programmable Graphics Hardware. *ACM Transactions on Graphics*

REZK-SALAMA, C. & KOLB, A. 2006. Opacity Peeling for Direct Volume Rendering. *EUROGRAPHICS 2006*, 25.

RIANTO, S. & LI, L. 2008. A 3D Fluid Dynamic Rendering For Real-Time Surgical Cutting Simulation. *Seminar Nasional Aplikasi Teknologi Informasi (SNATI)***,** 6.

SAKAS, G. & WESTERMANN, R. 1992. A Functional Approach to the Visual Simulation of Gaseous Turbulence. *Computer Graphics Forum*, 11**,** 107-116.

SEMICONDUCTOR INDUSTRY ASSOCIATION. 2002. *International Technology Roadmap for Semiconductors* [Online]. Available: http://public.itrs.net/ [Accessed December 2011].

SONG, O.-Y., SHIN, H. & KO, H.-S. 2005. Stable but nondissipative water. *ACM Trans. Graph.*, 24**,** 81-97.

STAM, J. 1999. Stable Fluids. *Proc. of the 26th Annual Conference on Computer Graphics and Interactive Techniques***,** 121-128.

STAM, J. & FIUME, E. 1993. Turbulent Wind Fields for Gaseous Phenomena. *Computer Graphics Proceedings***,** 369-372.

STAM, J. & FIUME, E. Year. Depiction of fire and other gaseous phenomena using diffusion processes. *In:* In ACM Computer Graphics (Proc. SIGGRAPH '95), August 1-6 1995. 129-136.

THOMAS V, T. I. & COHEN, E. Year. Direct Haptic Rendering Of Complex Trimmed Nurbs Models. *In:* Proceedings of Symposium on Haptic Interfaces, November 14-19 1999 Providence. ASME International Congress and Exposition.

UTSUMI, T., KUNUGI, T. & AOKI, T. 1997. Stability and accuracy of the Cubic Interpolated Propagation scheme. *Computer Physics Communications*, 101**,** 9-20.

VENETILLO, J. S. & CELES, W. 2007. GPU-based particle simulation with intercollisions. *The Visual Computer*, 23.

W.LI, Z.FAN, X.WEI & A.KAUFMAN 2003a. GPU-Based flow simulation with complex boundaries. Computer Science Department, SUNY at Stony Brook.

W.LI, Z.FAN, X.WEI & A.KAUFMAN 2003b. GPU-Based flow simulation with complex boundaries. *Technical Report 031105, Computer Science Department, SUNY at Stony Brook.*

YABE, T., ISHIKAWA, T., WANG, P. Y., AOKI, T., KADOTA, Y. & IKEDA, F. 1991. A universal solver for hyperbolic equations by cubic-polynomial interpolation II. Two- and three-dimensional solvers. *Computer Physics Communications*, 66**,** 233-242.

YABE, T., MIZOE, H., TAKIZAWA, K., MORIKI, H., IM, H.-N. & OGATA, Y. 2004. Higher-order schemes with CIP method and adaptive Soroban grid towards mesh-free scheme. *Journal of Computational Physics*, 194**,** 57-77.

YABE, T., OGATA, Y., TAKIZAWA, K., KAWAI, T., SEGAWA, A. & SAKURAI, K. 2002. The next generation CIP as a conservative semi-Lagrangian solver for solid, liquid and gas. *Journal of Computational and Applied Mathematics*, 149**,** 267-277.

ZHANG, Y., SOLENTHALER, B. & PAJAROLA, R. Year. Adaptive Sampling and Rendering of Fluids on the GPU. *In:* HEDGE, H. C., LAIDLAW, D., PAJAROLA, R. & STUADT, O., eds. IEEE/EG Symposium Volume and Point-Based Graphics, 2008. The Eurographics Asoc.