Simulation of Smoke Improve Computing Coordinate Performance

Yongzhe Xu[1], Eunju Kim[1] and Byungsoo Lee[1,*]

[1]*Department of Computer Engineering, University of Incheon, Korea*

yongzhexu@hotmail.com, leebone28@hanmail.net, bsl@incheon.ac.kr

Abstract

Smoke simulation is an interesting topic in computer graphics these days. The realistic rendering of smoke scene is one of the most challenging tasks in fluid dynamics of computer graphics. Smoke simulation rendering is based on basics particle system. In this paper, our proposed method based on FDS's (Fire Dynamic Simulation) output the result of the Excel file which improves DirectX for effective smoke and fire model. Using Excel file coordinates to DirectX particle system, and changes the original particle systems coordinates value. After computing the coordinates of particle moving location and boundary the flow simulation's real world smoke fluid mechanism of fluid dynamics. Recent game engine computer graphic developers usually control resources focused on smoke visualization and fire model. In this paper, we focused on smoke location boundary. The proposed method can change smoke particles more accurately and efficiently than the rendering is implemented by making smoke particles after putting the coordinates form FDS into DirectX Experiment result shows that 500 million particles have 5~8% performance improvement, and gives neutrality and reality to user's view.

Keywords: *Computer Graphics, Fire Dynamics Simulator (FDS), Smoke, Particle System, Fluid Dynamics, Unity 3D, Fire Simulation*

1. Introduction

The first particle systems in computer graphics used the generation of the explosion of a planet for the effect of special film tricks. Over the past few years, researchers working on simulated smoke [8] Ronald Fedkiw, Jose Stam, and Henrik Wann Jesen by using 300,000 particles. That was the first computational particle system simulation of smoke using fluid dynamics. Now, developers use 2 to 3 million particles for simulation of smoke for 3D Application program. Real flow of [18] 2012 particle system used more than 5 million particles simulation smoke and fire model. From this simulation or game engine, FDS requires a great amount of time to simulate real-world fire and smoke. Therefore, performance is not good enough. Working with real time simulation and finding out bug from 3D fluid solver are a challenging task. Currently, Computation of fluid dynamics simulation used for more accuracy, high interactivity possible application industrial design validation medical simulator, games of smoke effective, firefighter simulator, virtual reality. In the following explanation of our proposed approach, the focus is on how to skip the computing of coordinates, but the other methods only focus on real time and visualization which looks like more effective. Many of algorithm mused prepare time simulation fluid dynamics save boundary data. In Section 3, The proposed method based on FDS result in smoke data, by getting smoke boundary from FDS output excel file, after computing smoke boundary graphic card skip complex coordinates computing time which improves the performance up. In the Section 3.5.2 experiment result part, we perform comparative study of NVIDIA soft particle

demo and Intel fluid demo. In the Section 4, we discuss limitation part of our algorithm and some delay problem, and last section describes further work.

2. Related Works

In computer graphics area, developers have researched on visual realistic simulation effects such as fire, water, smoke and fog which are very challenging tasks. From 1981s until now [11, 26], Researchers have been working on new methods for simulating fluid dynamics. The first algorithm is the basics of the particle system. That algorithm is too complex and demands long time for calculation. Therefore, doing simulation in real time computing is not an easy task and results are not sufficient for live work [13]. From the figure 1 opacity Mapping demo Particle system is the point of 3D space that is determined by the position x, y, z, and orientation is given by three vectors x, y, z. NVIDIA smoke is based on fluid physics mechanism [1, 2]. The approach from the 2d fluid simulation and implementation of 3D fluid simulation by Harris [5] is difficult for real time simulation. Form formulation (1) represents is Partial differential equations (PDEs) function

$$\left(\frac{\partial}{\partial t}\right) x = f(x,t). \quad (1)$$

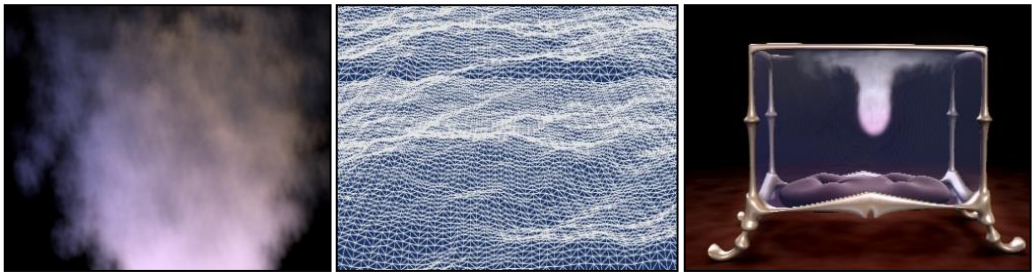
Formulation (2) the x is changed by wavelet f, which may be depended on x and t.

$$x^{n+1} = x^n + f(x^n, t^n). \quad (2)$$

2D fluid simulation as shown by author Harris doesn't have height coordinate z [7].
Improved NVIDIA Box of smoke demo is added function z from the formulation (3) [3]

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} = 0. \quad (3)$$

Set world space coordinate. But this function occupies all graphic card resources, so the performance is slow.



(a) (b) (c)

Figure 1. (a) NVIDIA Opacity Mapping Demo, (b) Harris 2004 2D Fluid Simulation, (c) NVIDIA Box of Smoke Demo

Common physically approachable Navier-Stokes Equations.

The original form of the equations of fluid motion is General Navier-Stokes equations. The

formulation is (4) [17]

$$\rho\left(\frac{\partial_v}{\partial_t} + v \cdot \nabla v\right) = -\nabla p + \nabla \cdot \mathrm{T} + f \qquad (3)$$

V is the flow velocity, ρ is the fluid density, p is the pressure, T is the stress tensor, f represents body force acting on fluid and ∇is the del operator.

2.1. Texturing Fluid

Texturing fluid is the method for rendering approach in simulation of fluid dynamics. The main target of texturing fluid considers special effects, animations, games, movies and medical simulations. Texturing fluid is a novel texture synthesis algorithm for fluid flow. It supports a simulator for 3D velocity fields and free surface of the fluid where each iteration produce individual outputs [16].Texture fluid also supports bump, displacement and alpha mapping. Texturing fluid handle topological changes simulated fluid merge and multiple fluids of its volumes. Figure 2 shows a flow chart of how to texturing fluid components interact with each other for fluid texturing. Texturing Fluid has three steps to do

(i) Fluid simulator based on velocity information computing dynamics surface.

(ii) Performing textures synthesis on the fluid surface that coherents with temporally near surfaces

(iii) The method of transporting texture information(velocity, surface size) from current surfaces to near surfaces

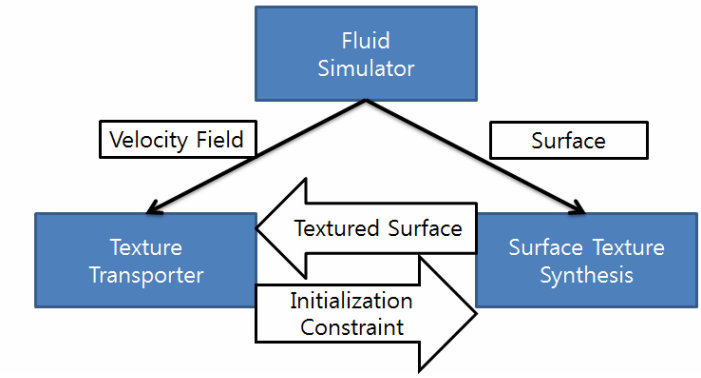


Figure 2. Texturing Fluid over View

2.1.1. Height Field Fluids

Height field's [19] main goal is reducing dimension from 3D to 2D. Algorithm cannot support captures of breaking wave, breaking smoke, breaking fire. The idea of algorithm is wave equation and decreases dimensions. Height field position is column height, velocity changes with the column height. Wave equation [22] based on Newton's Second Law of motion, where Formulation (5) is replaced k/m by c2 utt= c2· uxx is 1D wave equation.

$$\mu_{tt} = \frac{f}{m} = k \cdot \frac{u_{xx}}{m} \ . \qquad (5)$$

Formulation (6) shows Solution is function X c is velocity of waves travel.

$$\mu(x,t) = f(x+ct) + g(x-ct). \quad (6)$$

2.1.2. Original Smoke Simulation Algorithm Problem

Newton's Second Law i.e. f = m × a changes to a = f ÷ m i.e. the change of velocity in per unit time is the equal force divided by mass. Formulation (7) ρ is m, (Vt +V. ∇t) is a, (-∇ρ+f) means f. Simulated fluid loop is like Formulation (7)

$$\rho\,(V\_t + V.\nabla t) = -\nabla\_\rho + f. \quad (7)$$

Compute force position and velocities. If we want to compute the velocity, we need "force ÷ mass × time" to calculate the position. We need "velocity × time". Figure 3 shows using original smoke simulation algorithm simulation particle moving like fluid dynamics. This demo supports multi core part (OpenMP). If we want to know performance, we need to reduce multicore parts, or add all comparative program codes to support multicore.

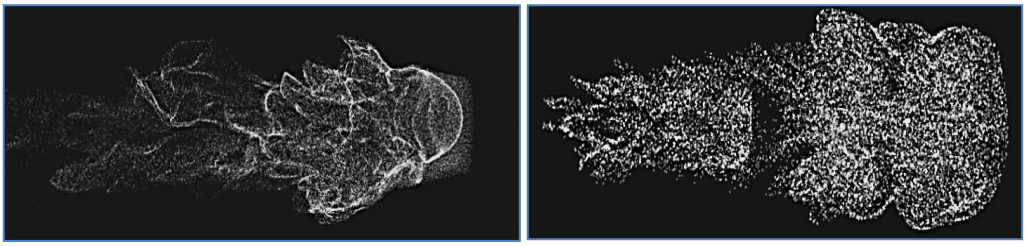


Figure 3. Intel Fluid Simulation for Video Game Demo

2.2. Current Research Focus

Particles based on fluids are simple and fast simulation fluid dynamics. Each particles attribute has mass, position, velocity, external forces, life times (life cycle). Particles emitters, particle life time, position and density are changed. Recently SPH (Smoothed Particle Hydrodynamics) often use for real time fluids in Computer Graphics [20, 21]. Figure 4 shows this algorithm result. Conservation of mass is Formulation (8) and (9).

$$\frac{\partial p}{\partial t} + \nabla \cdot (\rho v) = 0. \quad (8)$$

- Navier Stokes Equation conservation of momentum (9)

$$\rho\left(\frac{\partial_v}{\partial_t} + v \cdot \nabla v\right) = -\nabla p + \rho g + u\nabla^2 v.$$

Evaluate the pressure in particle locations and compute pressure the ideal gas stat equation.

Figure 4 NVIDIA Physic Fluid Demo (SPH)

2.3. Our Proposed Approach

Our approach is different from the existing method, here our proposed method focused on reducing computing complexity. From Section 2.2, if we know the mass, position, velocity, external forces, life times (life cycle) values then we can skip the computation of the same function again. Figure 5 shows the method of our proposal system flowchart. First step is to make Game Environment and input environment setting in FDS simulator. Second, Use game environment in FDS (input smoke detector into detection smoke density and 3D space coordinate). Third, after FDS simulation is done, load the output parameters (HRR, Device detection output file). Lastly, use data rendering NVIDIA soft particle system to find out simulator result.

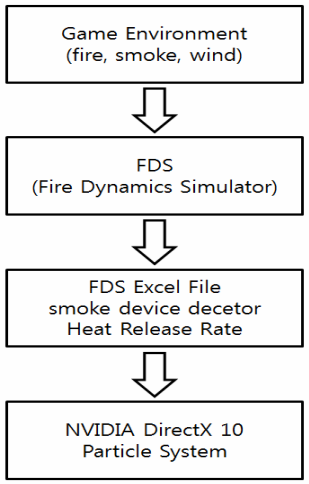


Figure 5. Proposed Method System Flowchart

3. FDS

Fire Dynamics Simulator (FDS) is a computational fluid dynamics model of fluid flow. The software based on Navier-Stokes equation which is suitable for low speed, thermally flow, smoke and heat simulation from fires. FDS is a free software by the NIST (National Institute of Standards and Technology of the USA Department of Commerce [24], VTT Technical Research Center of Finland. Execute “smokeview.exe”, load “example.smv”, and

visualization of fire. The "smokeview.exe" is a companion program that reads FDS's output of the excel file and produces animations on the user's screen. "smokeview.exe" has a simple menu-driven interface. Click the mouse right button and select the menu which shows only Smoke, HRR (Heat Release Rate) [25], Fire, or combines altogether.

3.1. FDS Simulation Environment

In Figure 6, simulation Environment mesh is x, y, z = 27 X 27 X 27 (meter). The mesh has small room 9 X 9 X 9 (meter). Burner size is 1 X 1 X 0.1 (meter). Window size is 3 X 3 X 3 (meter). Material size is 1 X 5 X 1 (meter). Simulation time is 700 second. Mesh x, y, z cells are 30 and all mesh cells are 531441000. Simulation time is 3days 4hour (75 hours).

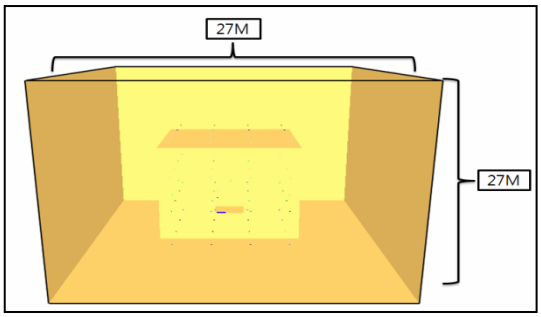
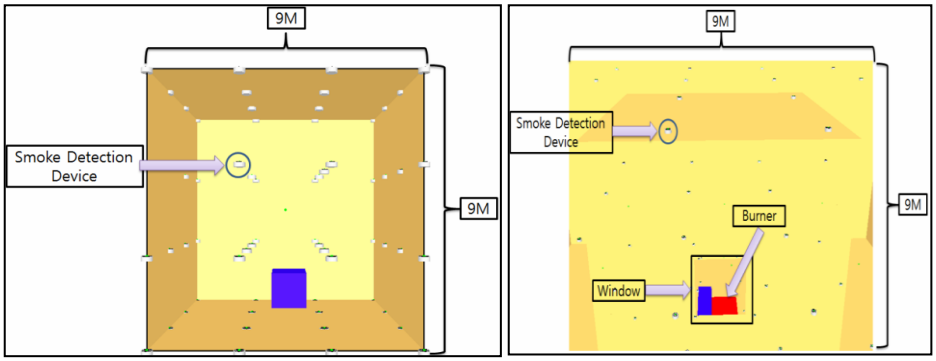


Figure 6. FDS Simulation Environment

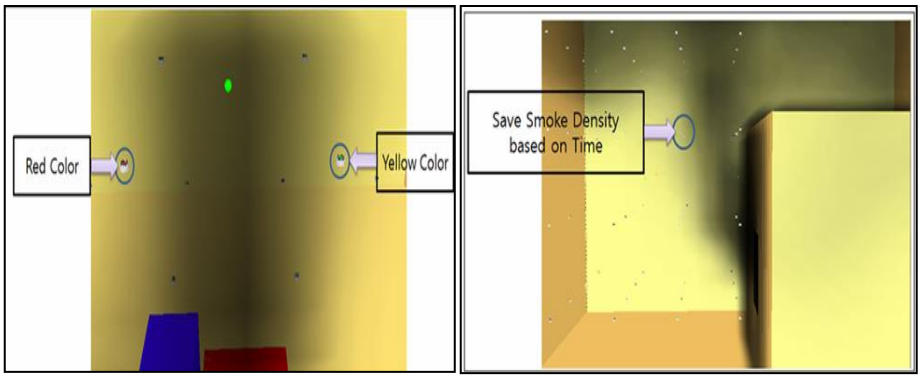
3.2. Smoke Detection Device coordinates

Figure 7 (a) shows FDS checking smoke density in the simulation. Here, we have displayed the experiment simulation environment. Figure 7 (b) shows room and smoke detection devices. A blue box is a burning material. Simulation programming part requires setting smoke device coordinate position near fire burner and window. Each smoke detector distance is 1 X 1 X 1m. Smoke detector records more than value 7 density of smoke, light color changes from red to stable yellow color.



(a) (b)

Figure 7. (a) Room Environment (b) Smoke Detector Coordinate (1x1x1m)



(a) (b)

Figure 8. (a) Smoke Detector Color, (b) Record Smoke Density by Times

Figure 8 (a) red areas is fire Burner simulation scenario and 1st room is fire start. 2nd smoke initial and flow fluid dynamics from window to outside place. Figure 8 (b) shows 3rd all smoke boundary record in output excel file by user setting times (seconds).

3.3. FDS Result Data

Section 3.2 discusses FDS output of the excel file. One is HRR and the other is Smoke Device detector file. Table 1 shows excel file data value with a human vision distance.

Table 1. Smoke Density Detector Relationship Field of Vision

Visible value	Vision distance	statement
0.1	20~30m	smoke detector start alarm
0.3	5m	It is very difficult for human to find out exit door
0.5	3m	Human can't find exit door when distance is more than 3 meter.
1.0	1~2m	human cannot see front object
10	0.2~0.5m	If the distance is more than 1 meter, human can't find escape lampe.
30	-	-

3.4. Using FDS Data Simulation Fluid Dynamics

In order to work with FDS data, particles slices of boundaries are needed. 3.5.1 Section 'transform.position + next_position' loads 3D space of next transform position location coordinates. Figure 9, Life-cycle of the particles move from 'Start' to 'End', and the particles disappear when arrives at End-position. Each space send particles to next Random Function space coordinate. From game environment we need the first found slicing boundary. From FDS, we need to know the start point and end point of particles of smoke, we need to compute how many boundaries are needed to slice. Based on the rate of change curve we can

find parameter wave when have big choice. FDS output parameter had change set that time space is slicing boundary.

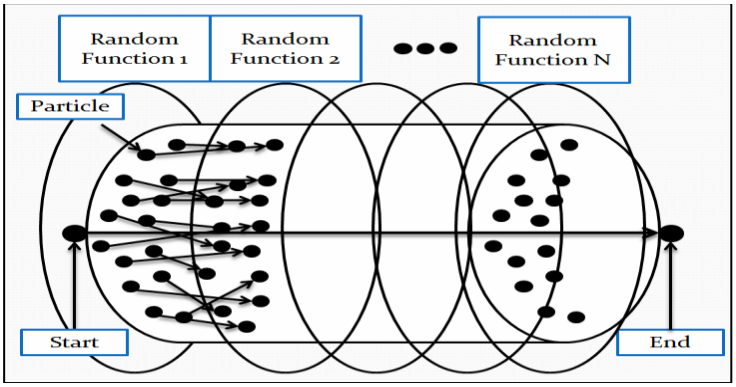


Figure 9. Slicing Particle Boundary

3.5. Proposal Source Code

Source code <1> program code, 'struct Particle Attribute', is Particle Attribute setting part [6, 9]. We can change '_position' to change particle system to world space coordinate. The particle's position is parameter '_position'. The particle is speed emitter counts (less than 5000), the particle's life-time follows location and boundary. Source code <2> particle in DirectX 10 advanced particle system change boundary limit [12, 14] smoke particle initial boundary. Source code <3> shows the change of coordinate and next coordinates values.

3.5.1. Source code <1>

```
struct Particle Attribute

{

       D3DXVECTOR3 _position = FDS excel data; // particle world
coordinate

       D3DXVECTOR3 _velocity;     //particle speed emitter counts

       D3DXVECTOR3 _acceleration; //particle acceleration

       float _lifeTime;           // particle life time

       float _age;                // particle age

       D3DXCOLOR _color;          // particle color

       D3DXCOLOR _colorFade;      // particle color change

       bool _isAlive;             // particle destroy or not

};
```

Source code <2>

```
public ParticleSystem particles;

{

       If(Vector3.Distance(lastPosition, transform.position)
>life_cycle)

       {

              Destory (particles);

       }

       Else

       {

              lastPosition = transform.position +
location_next_position

       }

}
```

Source code <3>

```
    float3 GetVolumeCoords( float3 pos )

    { // input load from fds pos to this source code

       Particle.coords.x = current.pos.x + (VolumeSize/2.0);

       Particle.coords.x /= VolumeSize;

       Particle.coords.y = current.pos.z + (VolumeSize/2.0);

       Particle.coords.y /= VolumeSize;

       Particle.coords.y = 1 - coords.y;

       Particle.coords.z = current.pos.y / VolumeSize;

       Particle.coords += VolumeOffsets;

       return current.coords;

    }

       // sample volume and velocity textures

       float3 coords = GetVolumeCoords( input.pos );

       float4 planeEq = g_txVolume.SampleLevel( g_samVolume, coords,
0 );

       float3 worldVel = g_txVelocity.SampleLevel( g_samVolume,
```

```
coords, 0 );
       //color = planeEq;
       //planeEq = float4(0,0,0,0);
       //float3 worldVel = float3(0,0,0);
       float distToPlane = dot( planeEq, pos );
       if( distToPlane != 0 && distToPlane < g_fParticleRadius )
       {
              //particle velocity in boundary
              pos.xyz += (particleRadius - distToPlane)*planeeq.xyz;
              //count how many particle in sense
              float impartVel = max( 0, dot( normalize(worldVel),
planeEq.xyz ) );
              // if particle in initial time or destroy time input
NULL
              vel = (1-impartVel)*reflect( vel, planeEq.xyz ) * 0.5;
              // else impart velocity
              vel += impartVel * ( worldVel );
       }

       current.pos.xyz = current.pos.xyz + vel*elapsedTime;
       vel = vel + frameGravity*elapsedTime;
       output.pos = pos;
       output.lastpos = lastpos;
       output.vel = vel;
       output.color = color;
       output.id = input.id;
       return output;
   }
```

3.5.2. Experiment Result

Table 2 is our experiment, we used CPU I5-2500, Samsung 4GB Memory, NVIDIA 650GTX Graphic Card, and FDS Simulation Version is 5. Figure 10 [15] shows source codes in rendering time. Performances are different in LOD (Level of Detail). We input coordinate boundaries in game scene to change LOD. The performance result got better than original algorithm (real time coordinate compute).

Table 2. Environment of Experiment

Hardware	System Specification
CPU	Intel I5-2500
Memory	Samsung 4Gb
Graphic Card	NVIDIA 650GTX
Simulation Version	FDS 5

Figure 10. Experiment Performance Result

Figure 11 shows the experiment result, mesh is like FDS simulation 27 X 27 X 27 smoke particle initial 1000, Particle dumping is 5, Particle life time is 3second, Particle boundary is grow 0.3 , and limitation is 1.0. Figure 11 shows particle boundary in a game sense. The particle follows program code coordinates and boundaries. The left of Figure 9 shows smoke in 3D space mesh. The center picture shows particle initial follow box-line. The right picture shows the change of texture in life time of particle. Particle texture was changed from black to bright.

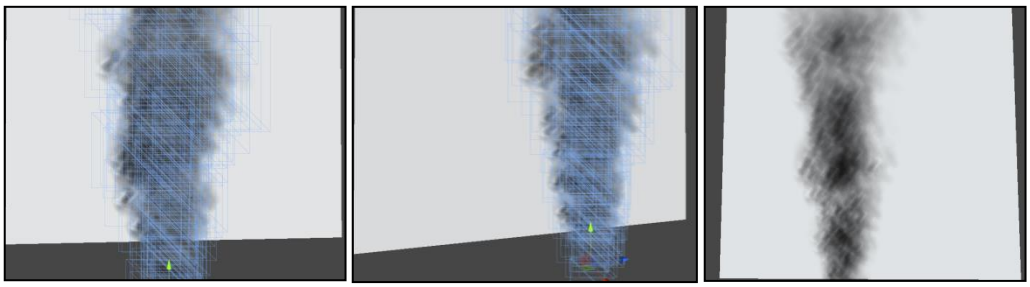


Figure 11. Experiment Result

4. Limitations

In this approach, we have experimented 27 X 27 X 27 (meter) mesh. In a big environment scene simulation FDS output data is very big. It is desirable particles in lifetime requires move one location to another location. Thus particle's lifecycle grow longer, similar original particles life cycle. FDS simulation needs a long time to compute a simulation.

5. Conclusions and Future Work

Our proposed algorithm reduces coordinates computation to improve a performance. Existing algorithms focus on real time simulation and fluid dynamics. The excel file based on

FDS smoke detector device coordinate computes smoke boundary coordinate. The result can improve the performance better than other visualization smoke particle systems. Because graphic card can skip computing of the particle coordinate not like fluid mechanism particle system. This method is possible to be used in steam effect not only smoke particle system but also fire. This approach is different from common simple particle system visualization as it has more accuracy and realism. The problem of this approach is that FDS simulation part needs a long time to simulate.

References

[1] B. Eberhardt, A. Weber and W. Strasser, "A Fast, Flexible, Particle-System Model for Cloth Draping", IEEE, Computer Graphics and Application, vol. 16, no. 50, **(1996)** September, pp. 52-59.
[2] J. Jansen and L. Bavoil, "Fast rendering of opacity-mapped particles using DirectX 11 tessellation and mixed resolutions", Opacity Mapping SDK White Paper NVIDIA Graphics SDK 11 document.
[3] C. Keenan, L. lamas Ignacio and T, Sarah, "Real Time Simulation and Rendering of 3D Fluids. GPU GEMS 3, **(2007)**.
[4] M. Kevin, K. Bryan, H. Simo and F. Jason, "Fire Dynamics Simulator (version 5)", NIST Special Publication 1019-5. NIST.
[5] J. Lander, "The Ocean Spray in Your Face. Game Developer Magazine", **(1997)** July, pp. 13-20.
[6] F. Luna, "3D Game Programming with DirectX 9.0", Plano, Texas: Wordware Publishing, **(2003)**.
[7] J. Harris Mark, "Fast Fluid Dynamics Simulation on the GPU, GPU GEMS, chapter 38.
[8] F. Ronald, S. Jos and J. Henrik Wann, "Visual Simulation of smoke", Proceedings of the 28th annual conference on Computer graphics and interactive techniques ACM, **(2001)**.
[9] W. Reeves, "Particle Systems A Technique for Modeling a Class of Fuzzy Objects", ACM Transactions on Computer Graphics, vol. 17, no. 3, **(1983)**, pp. 359-376.
[10] Unity Team, Unity Manual, Particle System, positioning Game Objects.
[11] Jos Stam, Stable Fluids, In SIGGRAPH 99 Conference Proceedings, Annual Conference Series, **(1999)** August, pp. 121-128 PDF.
[12] D. Wensheng, Z. Xinyan and Z. caijun, "Smoke Simulation based on particle system in Virtual Environments", Multimedia Communications (Mediacom), 2010 International Conference.
[13] M. Caniato Renhe, A. Oliveira, C. Esperanca and R. Marroquim, "Enhanced Target Driven Smoke Morphing", 2012 XXV SIBGRAPI Conference on Graphics, Patterns and Images.
[14] H. Won Byun and H. Moon Jung, "Drawing Style Capture for Cartoon Rendering", International Journal of Multimedia and Ubiquitous Engineering, vol. 8, no. 1, **(2013)** January.
[15] J.-H. Shin, S.-m. Lee and D.-J. Kim, "An Analysis in the Correlation between Frontal Lobes/Occipital Lobes Parts' Neural Waves in Case of 3D Syndrome Outbreak While Watching 3D Object", International Journal of Multimedia and Ubiquitous Engineering, vol. 7, no. 2, **(2012)** April.
[16] H. Yeom and U. Yoon, "ECG Artifact Removal from Surface EMG Using Adaptive filter Algorithm", International Journal of Multimedia and Ubiquitous Engineering, vol. 7, no. 2, **(2012)** April.
[17] S. Feng He, H.-C. Wong and U.-H. Wong, "An Efficient Adaptive Vortex Particle Method for Real-Time Smoke Simulation", IEEE 2011 12th International Conference on Computer-Aided Design and Computer Graphics.
[18] T. Schlick, RF_toolfactory.http://thevault.realflow.com/docs/realflow_manual.pdf , realfolow, **(2012)**.
[19] N. Thurey, M. Muller-Fischer, S. Schirm, M. Gross and J. J. Monaghan, "Smoothed particle hydrodynamics", Annual Review of Astronomy and Astrophysics, vol. 30, **(1992)**, pp. 543-574.
[21] S. Premože, T. Tasdizen, J. Bigler, A. Lefohn and R. T. Whitaker, "Particle based simulation of fluids", Eurographics 03, pp. 401-410.
[22] B. Miklós and M. Müller, "Real-Time Fluid Simulation Using Height Fields", eth zurich (swiss federal institute of technology), Switzerland summer, **(2004)**.
[23] D. Enright, S. Marschner and R. Fedkiw, "Animation and Rendering of Complex Water Surfaces", SIGGRAPH '02 Proceedings of the 29th annual conference on Computer graphics and interactive techniques, **(2002)**, pp. 736-744.
[24] M. Kevin, K. Bryan, H. Simo and F. Jason, "Fire Dynamics Simulator (version 5)", NIST Special Publication 1019-5. NIST.
[25] G. P. Forney, "User's Guide for Smokeview Version 5 - A Tool for Visualizing Fire Dynamics Simulation Data", NIST Special Publication 1017-1 CODEN: NSPUE2, Fire Research Division Building and Fire Research Laboratory, **(2007)** August.

[26] W. Magnus, F. Henrik, A. Chris and M. Stephen, "Capturing Thin Features in Smoke Simulations", Siggraph Talk, **(2011)**.

Authors

Yongzhe Xu received his MS in Computer science and engineering from Univesity of Incheon, Korea, in 2009.

He is currently working toward a PhD in computer science and engineering at the same university. His research interests include computer graphic, FDS, virtualization, and Pattern Recognization, Machine Learning.

EunJu Kim received her MS in computer science and engineering from University of Incheon, Korea, in 2010.

She is currently working toward a PhD in computer science and engineering at the same university. Her research interests include computer vision, image processing, and Artificial intelligence.

Byungsoo Lee received his MS in MBA from University of Dongguk, Korea, 1980.

Received his Doctor of Science from University of KyongGi, Korea, 1998.

He is a Professor of Department of Computer Engineering at the University of Incheon, Korea.

His research interests include software design, decision making system, eCRM, RFID/USN, IT convergence.