# A multi-sampling approach for smoke behavior in real-time graphics

Henrik Gustavsson[†]
University of Skövde

Henrik Engström[‡]
University of Skövde

Mikael Gustavsson[*]

## Abstract

Smoke simulation is a key feature of serious gaming applications for fire-fighting professionals. A perfect visual appearance is not of paramount importance, the behavior of the smoke must however closely resemble its natural counterpart for successful adoption of the application. We therefore suggest a hybrid grid/particle based architecture for smoke simulation that uses a cheap multi-sampling technique for controlling smoke behavior. This approach is simple enough for it to be implemented in current generation game engines, and uses techniques that are very suitable for GPU implementation, thus enabling the use of hardware acceleration for the smoke simulation.

**Keywords:** Smoke Simulation, Particle System, Multi-sampling, GPU, Serious Gaming

## 1 Introduction

Smoke simulation is a topic that has received much attention from computer graphics researchers. Commonly, work on smoke simulation is focused on graphical appearance, either producing photo realistic smoke [Fedkiw et al. 2001, Losasso et al. 2004] or less photo-realistic, cartoon-like smoke [Selle et al. 2000]. When simulated smoke is used in animations, graphical appearance is considered much more important than real-time execution of the particle system. In a serious gaming application, the behavior of the smoke is more important than the appearance. Computational efficiency is also important since a computer games engine is used for rendering as well as other tasks related to the game. For realistic behavior and appearance of smoke, rendering of a single frame of animated smoke typically takes up to a minute [Losasso et al. 2004] even very old, and comparably simplistic solutions [Ebert and Parent 1990] are prohibitively expensive to implement in a current generation game engine.

Particle systems in computer games typically use simple rule-based engines executed by the CPU of the computer used to run the game engine [Sele et al. 2000]. Such an approach tends to produce very simple particle systems with simplistic behavior and very limited interaction with the environment. We suggest an approach which uses a simple multi-sampling technique for a particle system that is similar to a rule-based particle engine but has a behavior that is more similar to advanced particle systems. This engine is sufficiently simple that it can be implemented in a current generation game engine, and in the future implemented using a GPU for hardware acceleration of the computations.

[†] e-mail: henrik.gustavsson@his.se

[‡] e-mail: henrik.engstrom@his.se

[*] e-mail: mikgust@gmail.com

## 2 Related Work

Particle systems are available in all of the major 3d animation packages and game engines. The particle engines available in computer games are however much less sophisticated than their counterparts in 3d animation software. This is mainly due to the stringent real-time requirements of computer games. Recently, several authors have investigated hardware acceleration of particle systems by offloading particle calculations to the GPU [Kipfer et al. 2004]. This would allow simulation of a significantly larger set of particles on the same hardware. If main memory can be avoided by using the GPU, the strain on the often congested system bus can be reduced significantly. There have also been suggestions for executing more advanced particle engines on the GPU such as fluid simulation [Amada et al. 2004, Hegeman et al. 2006]. Even with GPU acceleration, these systems are not sufficiently fast to simulate a particle system in a game engine.

Serious gaming is a topic that has recently emerged as a light-weight tool for training professionals or military personnel. Existing video game engines can be readily adapted for training [Fong 2004]. To the player, a video game consists of a playing environment, characters, tools, and missions. These elements can then be altered to better fit for training purposes. There are several games available today for various types of training. Many of these are military simulators, but others have been developed, such as games for training of firefighting professionals [Stapleton 2004]. A key issue when it comes to the use of games for training of professionals is user acceptance. Acceptance is related to the perceived realism of the simulated environment and poor acceptance may lead to reduced efficiency of the training [Lampton et al. 2002]. In a game for fire-fighting professionals such elements as the color of clothing and details of equipment may be important for user acceptance [Stapleton 2004]. In a game focused on extinguishing fires, the behavior of the smoke is one of the most important properties for user acceptance. If the smoke does not behave realistically the game is less useful for training purposes.

Currently, there are environments such as NIST Fire Dynamics Simulator (FDS) available for fire simulation in virtual buildings [Hyeong-Jin and Lilley 2006] or even smaller structures [Hamis 2006] which portray fire behavior in a very realistic manner. These simulations are very realistic but the performance is not high enough to be suitable for real-time execution using a game engine. There has also been work in producing realistic graphical representation of fire [Nguyen et al. 2002], but since fire simulation uses many of the same techniques as in smoke simulation fire simulation it has most of the same drawbacks as smoke simulation for real-time games environments.

## 3 A simplified physics model

The most common approach to realistic smoke motion is to either use Navier-Stokes or Euler equations. For a standard simulation system, the equation might use variables such as density, pressure, volume force, velocity and viscous coefficients [Fedkiw 2001]. The equations are then used to compute the properties of the fluid or smoke for each element of a grid. In a Lagrangian or grid-less approach, particles are often updates using a variations of the Navier-Stokes equations used by the grid methods where the acceleration of the particle is decided by the velocity, density and additional forces that may affect the particle.

For this work a Lagrangian approach was selected since particles can be updated using a simplified equation based on the pressure surrounding the particle. In the simplified context of smoke simulation for an in-door fire, the equations that affect particle motion can be simplified significantly. Firstly, there are no external forces such as wind inside a building. Furthermore, since the smoke is light, the effect of gravity is also negligible. There are thus two main forces that affect the smoke, the pressure of surrounding particles and the upward motion due to the heat generated from the fire.

We suggest an approach that uses a medium sized 3D grid implemented as a 3D texture that contains the density of particles in each of the grid cells. Each particle is controlled by the pressure in the surrounding grid elements, which is derived through texture sampling. The size of the grid needs to be roughly the size of the polygons used to render the smoke so that the polygons cluster together to form a solid block when the maximum density has been reached. A low density grid cell should attract particles and a high density grid cell should repel particles. A function can then be used to control the strength of the effect on the particle dependent on the pressure in grid cell. This gives us the equation:

$$\mathbf{a}_i = \frac{\sum \mathbf{s}_j \cdot f(\rho_i)}{\sum f(\rho_i)}$$

**Figure 1. Particle acceleration equation**

Where **a** is the acceleration for particle *i*, **s** is the sample vector and *f* the attraction function and $\rho$ is the particle density at the position of the particle *i* offset by the sample vector **s**. We simplify this further by using the acceleration as the particle velocity, thereby removing the need to keep track of both the particle position, velocity and acceleration.

This very simple approach also allows simplification of the particle updates since particle to particle collisions are not required and collisions with the environment can be simulated by grid elements with maximum density. This means that the particle update code will only need to know the density of particles and the particle position. A standard particle system commonly uses the position, velocity and acceleration of the particles [Kipfer et al. 2004]. By using the acceleration as the velocity, the update procedure is even simpler than a standard particle system.

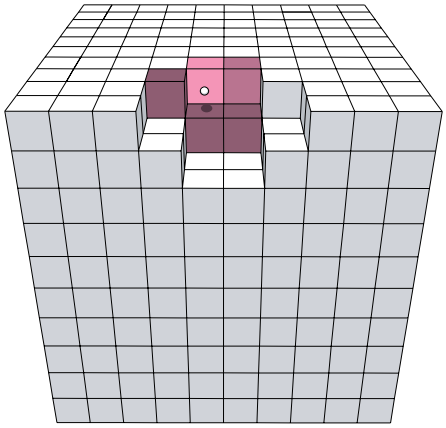

**Figure 2. Multi-sampling hybrid grid/particle approach**

## 4 Multisampling approach

When multisampling is used, it is important to select a sampling strategy that gives the best output for the least amount of samples. In the case of the particle engine it is important that the sampling occurs farther away from the location of the particle than the closest grid so that the particle will be attracted to low density areas some distance away from its current location. For this application we chose a sampling pattern that samples all 26 grid points one unit away from the particle so that no low density grid is missed regardless of the direction. The second level of samples are taken two steps away from the centre grid in a pattern similar to the Nvidia quincunx (pattern looks like the number 5 on a die) pattern used for anti-aliasing. The second level sample vectors are twice as long as the first level but since there are fewer of them, the total influence of the second level is very similar to the first level. On the third level there are only sample vectors for the six ordinal directions. so even though the vectors are longer, total influence is lower than both the first and second level.

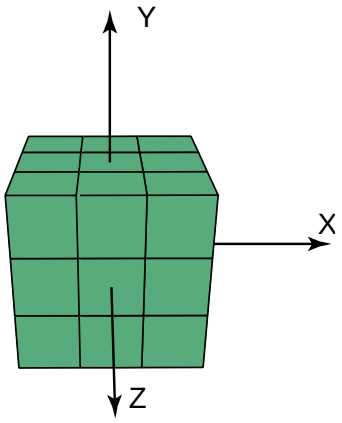


**Figure 3. First level (26 samples)**

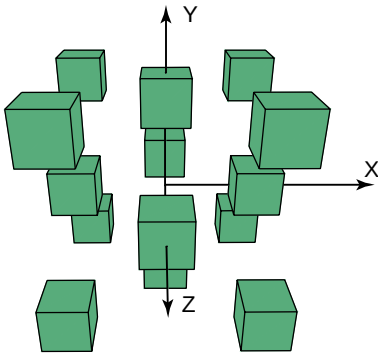


**Figure 4. Second level (14 samples)**

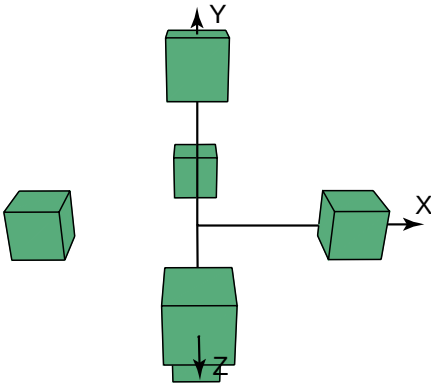


**Figure 5. Third level (6 samples)**

When the sampling pattern has been decided it is necessary to select the function that will decide how the particles will react. A high value means that the particles will be attracted and a value below zero means that the particles will be repelled. The steepness of the curve determines how fast that will happen and how many particles that will end up in each element of the grid.

For this application we chose a variation of the logistics function, with parameters that gave us a sigmoid curve that has the value 1 at 0.0 and reaches zero at 5 and -1 at 10. This means that it is unlikely that a grid element will contain more than 5 particles since any value above 5 will repel particles. For a room that has a very large number of grid elements it is necessary to use a steeper curve

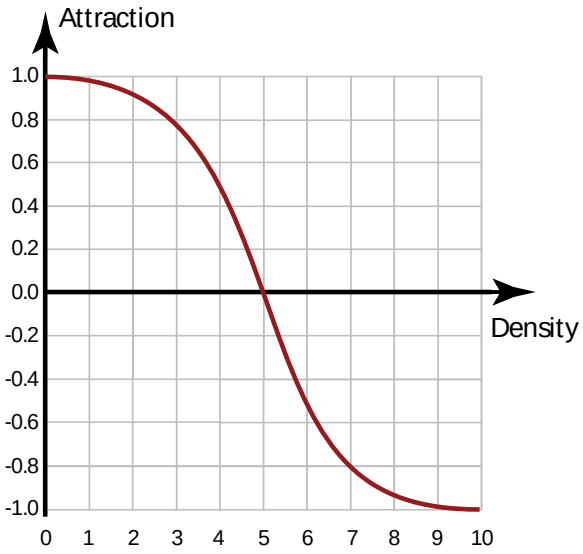


**Figure 6. Sigmoid particle density function**

If a simulation is started without any bias toward a specific direction, there is no incentive for the smoke particles to rise since all directions have equal influence on the particle speed. To introduce a system where the particle can rise while it is hot and close to the fire we must introduce a bias for each of the sample points so that when the particle is hot the sample points above 0 on the y axis get a slight advantage. This advantage can be reduced, either when the particle becomes older or when the particle has been in a low density (fewer warm particles) environment for a number of time steps. These parameters can be recorded into a bias texture, an example of which is shown below. Note that the differences between high and low bias has been exaggerated.

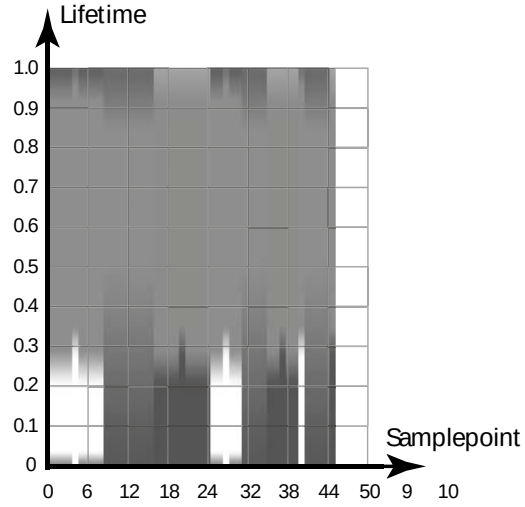


**Figure 7. Sigmoid particle density function**

The resulting pseudo-code can be seen below. Note that three textures are used, one 3D texture for the grid, one 2D texture for the bias and one 1D texture for the sigmoid function lookup table. This means that each particle will require 138 samples per particle which is not much since a GPU can perform millions if not billions of samples per second.

```
for(int j=0;j<46;j++){
        x=xk+sample_x[j];
        y=yk+sample_y[j];
        z=zk+sample_z[j];

        int griddata=Grid[x][y][z];
        cnt+=abs(sigmoid[griddata]);

        sumx+=sigmoid[griddata]*
        sample_x[j]*bias[lifetime][idx];
        sumy+=sigmoid[griddata]*
        sample_y[j]*bias[lifetime][idx];
        sumz+=sigmoid[griddata]*
        sample_z[j]*bias[lifetime][idx];
}
```

**Figure 8. Particle update pseudo-code**

## 5 Results

Through visualization of the density grid it is possible to see that the particle system works well when it comes to equalizing the particle pressure. The particles strive to move away from the high density areas into lower density areas. The largest dot at the top of figure 8 repels surrounding particles with maximum force.

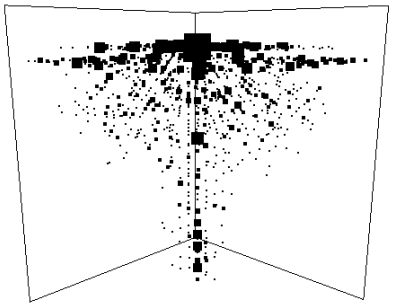

**Figure 9. Particle density plot**

Particle behavior over time is also close to the expected result, the hot recently born particles rise with a high speed and the colder older particles are more stationary or are even dropping slightly when cold. The figures below depict the simulation at t=1 second, t=10 seconds and t=25 seconds and finally t=50 seconds.

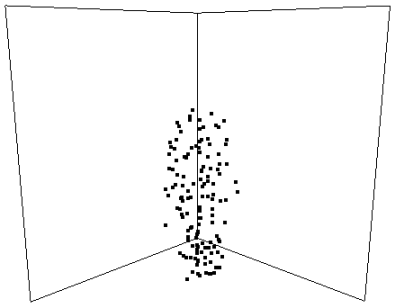

**Figure 10. Simulation at t=1 second**

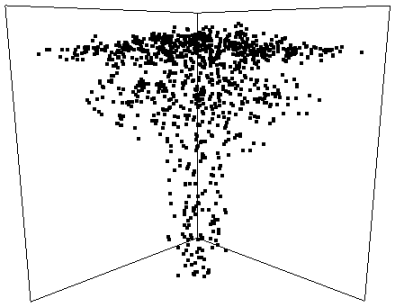

**Figure 11. Simulation at t=10 seconds**

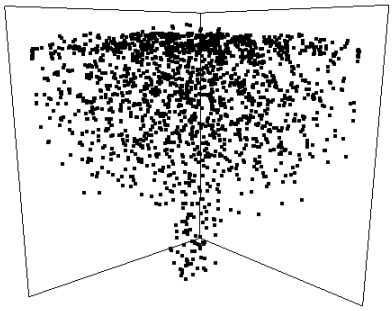

**Figure 12. Simulation at t=25 seconds**

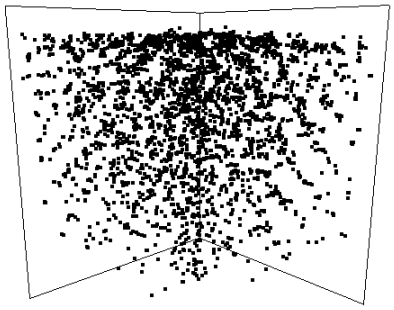

**Figure 13. Simulation at t=50 seconds**

## 6 Conclusions

We have shown that it is possible to construct a very simple particle system that produces smoke that behaves realistically in that the smoke goes up to the top of the room and then slowly fills the room from above until the room is filled with a uniform cloud of smoke. The simulation uses only approximately 150 texture samples per particle which means that thousands of particles can be simulated on a modern computer without large negative effects on the performance of the game engine.

The bias texture has a big effect on the behavior of the particles and construction of the bias texture does require some attention for realistic behavior. Similarly, the particle emission and density also affects the visual quality. If the grid size is too large or particles are released at locations that are very close, particles will not move realistically.

## 7 Future Work

There are large amounts of future work to conduct with this smoke simulator.

- Examining the effects of the bias texture and the sigmoid function and see if other values or configurations are better than the current.
- More realistic rendering through billboards for smoke-puffs and using the density grid as a basis for putting soot on the walls of the room.
- Benchmarking the solution in real-life situations to see where for instance the number of particles gives the best visuals and the lowest negative effect on game performance
- One of the most pressing issues is the implementation of this smoke simulator in a real games engine to see how it reacts in a real game. Another pressing issue is to make the GPU implementation and examine the effect of the GPU on for example number of particles.

## 8 References


AMADA, T., IMURA, M., YASUMURO, Y., MANABE, Y. AND CHIHARA, K. 2004. *Particle-Based Fluid Simulation on GPU*, ACM Workshop on General-Purpose Computing on Graphics Processors and SIGGRAPH 2004 Poster Session.

EBERT, D. S. AND PARENT, R. E. 1990. *Rendering and Animation of Gaseous Phenomena by Combining Fast Volume and Scanline A-buffer Techniques.* Computer Graphics (SIGGRAPH 90 Conference Proceedings) 24(4):357–366

FEDKIW, R., STAM, J. AND WANN JENSEN, H. 2001. *Visual Simulation of Smoke*. ACM SIGGRAPH 2001, 23-31.

FONG, G. 2004. *Adapting COTS games for military simulation.* Proceedings of the 2004 ACM SIGGRAPH International Conference on Virtual Reality Continuum and its Applications in Industry, 269-272.

HEGEMAN, K., CARR, N. A. AND MILLER, G. S. P. 2006. *Particle-Based Fluid Simulation on the GPU*, International Conference on Computational Science 2006, Part IV, LNCS 3994, 228–235

HAMINS, A., BUNDY, M. AND DILLON, S. E. 2006. *Characterization of Candle Flames.* 44th AIAA Aerospace Sciences Meeting and Exhibit. 1-13

HYEONG-JIN, K. AND LILLEY, D. G. 2006. *Computer Modeling of Developing Structural Fires.* 44th AIAA Aerospace Sciences Meeting and Exhibit

KIPFER, P., SEGAL, M. AND WESTERMANN, R. 2004. *UberFlow: A GPU-Based Particle Engine.* HWWS'04 Proceedings of the ACM SIGGRAPH / EUROGRAPH conference on graphics hardware.

LAMPTON, D. R., BLISS, J. P. AND MORRIS, C. S. 2002. *Human performance measurement in virtual environments.* K. M. Stanney (Ed.), Handbook of virtual environmentvs: Design, implementation, and applications, Mahwah, NJ: Lawrence Erlbaum Associates, 701-720

LOSASSO, F., GIBOU, F. AND FEDKIW, R. 2004. *Simulating water and smoke with an octree data structure.* ACM Transactions on Graphics (TOG) 23(3) 57–462

NGUYEN, D. Q., FEDKIW, R. AND WANN JENSEN, H. 2002. *Physically based modeling and animation of fire.* Proceedings of the 29th annual conference on Computer graphics and interactive techniques. 721-728

STAPLETON, A. 2004. *Serious Games: Serious Opportunities.* Paper presented at the Australian Game Developers Conference, Academic Summit, Melbourne

SELLE, A, MOHR, A. AND CHENNEY, S. 2000. *Cartoon Rendering of Smoke Animations.* ACM SIGGRAPH