A Real-Time, Controllable Simulator for Plausible Smoke

Morgan McGuire
Brown University
morgan@cs.brown.edu
March 5, 2006

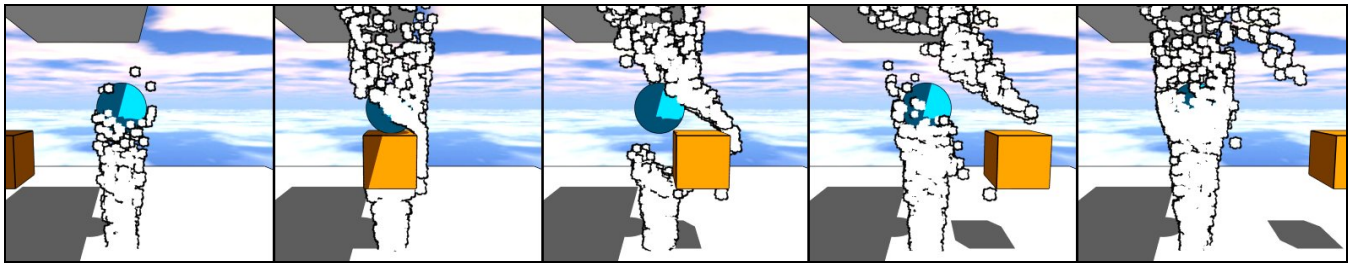

Figure 1. Animation of a rising white particle stream colliding with static (sphere, ceiling tile) and dynamic (box) objects.

Abstract

While many games and other virtual world applications include real-time simulation of rigid bodies, fluids and gasses have proven harder to animate efficiently. This report describes a method for extending previous work with particle systems to efficiently simulate smoke that interacts with rigid bodies, using an existing rigid body simulator. The method is applicable compressible fluids and gasses other than smoke.

Although the animations produced by this method are not physically correct, they exhibit the phenomenology of real smoke, including dissipation, vortices, and compression. The animations can also be controlled through aesthetically meaningful parameters like *drag* and *vortex axis*. Thus the simulator is well-suited to applications where interactive performance and expressive power are preferred to realism; for example, video games and rapid development of animation.

1 Introduction

Rendering is only half the challenge of creating compelling animation sequences for entertainment applications—objects must also be animated.

Most recent work on physically correct simulation of smoke and fluid flow involves three-dimensional grids. These methods are too slow and require too much memory to simulate real-time flow using current computers. Most also do not provide expressive artistic controls (one exception is the key-framed smoke work by Treuille et al. [2003]).

This report is motivated by the needs of video game and film production studios. Video games can benefit from fast smoke simulation at runtime for a myriad of effects including dynamic fog, explosions, missile trails, magic spells, and steam. In comparison, film production studios have the luxury of time for offline simulation and rendering. However, methods that accelerate simulation and rendering are also useful for film. Such methods reduce the cost of computer animation, making it available for small-budget and television production, and allow real-time preview that improves artist workflow.

Production of rigid body animation by physical simulation has become increasingly common in both the game and film context. Because of this, a base of real-time rigid body middleware has emerged. This includes software like Havok, ODE, Tokamak, and Novodex[1], and potentially hardware as graphics cards become more general purpose and companies like Aegia investigate dedicated physics processors.

This prevalence of rigid-body simulation affects the design of the smoke simulator. I designed smoke simulator to leverage existing infrastructure by integrating state and enforcing constraints with a rigid-body simulator (in this case, ODE). This design choice so also simplifies the challenge of making smoke interact with both static meshes and dynamically simulated rigid bodies.

Smoke is primarily a small number of carbon molecules suspended in a large volume of hot air. The carbon is highly compressible, however the hot air is not. However, in order to achieve real-time performance, this simulator approximates the air as perfectly compressible. This leads to the limitation that it is not possible to fill an environment with smoke without some form of advection.

1.1 Related Work

Particle systems [Reeves 1983] are a classic idea in computer graphics. They represent amorphous and fluid shapes like water, fire, and smoke as a set of point masses that do not affect one another during simulation. Particle systems are rendered by either drawing each particle as a one-pixel point or as a small alpha-blended billboard.

Sims [1990] introduced a series of controls for particle systems that include vortices, air currents, and particle generators. These expanded the set of effects possible with particle systems and gave strong artistic control over the result. A limitation of Sims' system is that many controls are needed to simulate interactions between particles and the environment like vortex shedding and constrained fluid flow.

In order to model these kinds of realistic phenomena, Foster and Metaxes [1997] introduced simulation of gas and fluids via grid-based simulation. Many simulation methods have since followed their lead, working with grids but improving the quality, performance, and space requirements of the original technique.

[1] Havok: http://www.havok.com
ODE: http://ode.org
Tokamak: http://www.tokamakphysics.com
Novodex: http://www.ageia.com

However, there is currently no grid method that can reasonably simulate phenomena in real time that have the dimensions and resolution of jet plane contrails during a dog fight or even the wisps of smoke from a candle. Most grid methods also lack the artistic controls that made Sims' approach so appealing for entertainment applications.

To achieve real-time results and artistic control (at the expense of physical correctness), this report returns to a particle-system approach. I am not the only ones to return to particles; Selle, Rasmussen, and Fedkiw [2005] recently seeded their offline grid simulation with thousands of explicit vortex particles to capture detail too small to represent on the grid; a previous technical report by Gamito [1995] reports a similar idea. Much of the recent work by Lin Shi and Yizhou Yu (e.g. [Shi2005]) has focused on particles, although none of it achieves real-time performance for 3D flow.

To introduce environment interaction in a particle system without losing artistic control or performance, the method in this report extends Sims' ideas with vortex shedding and constrained rigid-body simulation for particles. This creates dynamic advection and object-smoke interactions similar to those observed in physically correct methods.

In 1990, Sims' system required seconds to minutes per frame for simulation. Performance numbers in this report benefit from 15 years of hardware advances, but also are the result of algorithmic improvements. The new simulator leverages spatial subdivision by hash spaces and bounding hierarchies inside the OPCODE (http://www.codercorner.com) and ODE (http://ode.org) libraries to accelerate collision detection and vortex simulation to approximately linear time. ODE uses first-order forward Euler integration, which is more than sufficient for smoke phenomena, and resolves interpenetration constraints with an iterative matrix solver called Quickstep that is at least as fast as quadratic time.

2 Forces

Smoke particles have little mass or volume, so smoke simulation is actually fluid simulation of the surrounding medium (air), with smoke tracers to make the air currents visible. Particle systems maintain and integrate only the particle state, ignoring the medium. Explicit controls create forces that approximate the effects of the surrounding air.

Our simulator models six superimposing forces on particles: *vortex*, *wind*, *turbulence, buoyancy*, *gravity*, and *drag*. Vortices are special (invisible) particles that exert a swirling force in a small neighborhood. The others are global scene or parameters.

The interaction of particles and the scene is through non-penetration constraints. Particles experience inelastic rigid body collisions with objects, but the objects are not constrained by the collisions (as if the particles had zero mass). Particles do not collide with each other, so the smoke is compressible. Enabling particle-particle collisions simulates an incompressible fluid like water, but the tremendous number of collisions creates an unstable system. Figure 1 shows several frames of animation in which a column rising smoke is deflected around a fixed sphere and knocked aside by a moving box. The smoke is also contained by the ceiling at the top of the frame.

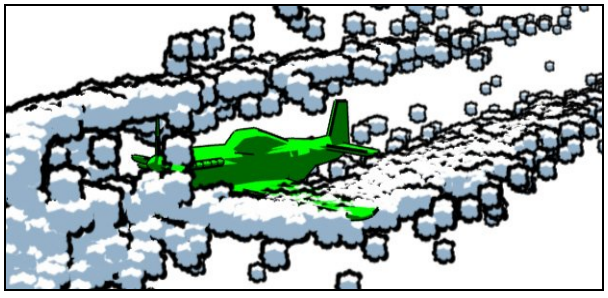

Figure 3. Clouds deflected by collisions with an airplane and swirling from interaction with propeller blades and vortices.

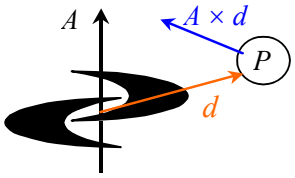

Figure 2. Vortices exert centripetal and centrifugal forces.

2.1 Vortices

The simulator models *vortices* with a position V, peak strength (in Newtons) p, rotation axis A, radial falloff exponent $1 \le e \le 2$, effect range r, and unitless amount of suction $0 \le s \le 1$. We found that peak strengths on the range 0.01 N $\le p \le$ 10 N effectively covered the range of candle-powered convection through shed vortices from airplane wings. Larger values remain stable but are uninteresting because the smoke dissipates to fast.

A vortex applies centripital acceleration (along the radial direction) and centrifugal or "suction" acceleration perpendicular to the vortex rotation and radial direction (Figure 3), like a whirlpool.

Let V be the center of the vortex, A be the vortex axis, P be the position of the particle. The radial direction is $d = (P - V)/\|P–V\|$. When the particle is in range, the force has magnitude that falls off with distance and is clamped to avoid excessive force near the center:

$$\|F_V\| = p \cdot \min\left(\|P-V\|^{-e}, 5\right) \text{ if } \|P-V\| < r, \text{ otherwise } 0 \qquad \textbf{Eq. 1}$$

The direction of the force is primarily perpendicular to the axis and radial direction, augmented by the artist-controlled centrifugal component:

$$\hat{F}_V = A \times d - d \cdot s. \qquad \textbf{Eq. 2}$$

Because a vortex only effects nearby particles, the simulator needs to locate them efficiently (i.e., in approximately constant time). The implementation exploits existing collision detection routines in a rigid body simulator to this end. Each vortex is represented by an invisible sphere of radius r that acts like a particle under simulation. Collisions between particles and vortices are enabled. However, when such a collision occurs the simulator computes and applies the vortex force instead of introducing a non-penetration constraint.

2.2 Wind

Wind force is a time-varying, spatially-uniform force field parameterized by direction, mean magnitude, and standard deviation (in Newtons). Wind is also subject to *gusts*, which have their own direction and statistical magnitude as well as a probability of occurring in a given time slice.

2.3 Turbulence

Turbulence approximates complex but local and small-magnitude effects like Brownian motion. It is modeled with a smooth, time varying 3D noise function (e.g., Perlin's) that indicates the magnitude and direction of force at a point. We found that for the approximately 1 kg/m^3 density particles in our simulations, turbulence forces on the order of 0.01 Newtons added a natural feel without completely dispersing clouds.

2.4 Buoyancy and Gravity

Buoyancy force is a constant force applied to all particles of the same material type. Particles accelerate at different rates, depending on their density. *Gravity* is a constant downward acceleration applied to all particles.

2.5 Drag

Drag simulates the presences of an incompressible medium by applying friction to oppose motion. It creates damping and a terminal velocity so that particles do not accelerate indefinitely under the other forces. We follow Sims and approximate drag as linearly proportional to and opposed to velocity (non-linear drag factors are more physically correct but hard to control). It is always important when numerically integrating to clamp the sum of frictional forces ||*mass* * *velocity* / *timeslice*|| to avoid adding energy to the system.

3 Particle and Vortex Generation

Particles are created by *generators* parameterized by a mean and standard deviation for the density, radius, and time until dissipation of the generated particles. Generators also have a generation rate and a volume within which the new particles will appear. They may be attached to simulated objects in the scene, e.g., at the tail of a missile.

Particles may also be explicitly created by programmer scripts, for example, to create clouds that exist when the simulation begins. Vortices are also created by generators at runtime, simulating vortex shedding as objects move through space, or explicitly before simulation to indicate a priori movement patterns. Pre-created vortices can be rigidly attached to objects and have strength proportional to linear velocity to create flow patterns.

Figure 4 shows examples of static and dynamically created vortices. The vortices are visualized as red wireframe cylinders.

4 Performance

An implementation of this system, including a billboard renderer, can process hundreds to thousands of particles per frame at 30 fps on an AMD 2.4 GHz CPU with a GeForce6800 graphics card. Including the renderer in the performance measurement is important because the bottleneck is actually between the simulator and the renderer.

To render at high speed with OpenGL (or DirectX) on modern graphics hardware, it is necessary to transfer the billboard geometry from the CPU to the graphics processor (GPU) in an array called a *vertex array*. The CPU executes the code that creates the vertex array containing billboard geometry for each particle and copies it to the GPU every frame, and that process is more limiting than rendering or even simulation.

Copying particle data is slower than simulating it for two reasons. First, simulation performance is limited by CPU clock and memory speed (which are both high), whereas the copy is limited by the PCI-express bus speed, which is much slower. Second, the copy also requires expanding each particle into four vertices and texture coordinates and rotating the vertices so that the billboard faces the viewer.

According to the DirectX 10 specification, the next generation of hardware will support new "geometry shaders" that can generate a billboard from a single point on the graphics card. We anticipate that this will remove the bottleneck.

Simulation is extremely fast because most objects do not interact. The simulation system alone can process up to 5000 particles and 100 vortices at 30 fps on a 1.4 GHz Pentium. Rendering is also very efficient—current hardware has sufficient fill and vertex-processing rate to handle up to 500k particles, so we are not fully exploiting the available graphics power because of the CPU bottleneck.

Porting general-purpose CPU algorithms to graphics hardware is an active area of research today. Our simulator is built on a rigid body simulator (ODE), so a hardware port of ODE should directly enable a fully hardware accelerated smoke system.

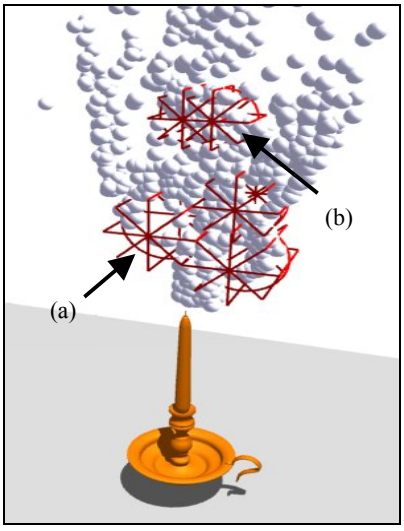

Figure 4. Particle flow through a fixed vortices (a) and dynamically created vortices (b) that are themselves rising.

5 Acknowledgements

Thanks to Colin Hartnett and Andi Fein for their work implementing this project. This research was supported by an NVIDIA Fellowship.

6 References

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual simulation of smoke. In *SIGGRAPH 2001 Conference Proceedings*, ACM SIGGRAPH, 251 – 260

GAMITO, M. N. 1995. Simulation Of Turbulent Flow Using Vortex Particles. Technical report, INESC, Portugal http://virtual.inesc.pt/virtual/tr/gamito95/artigo.html

REEVES, W. T. 1983. Particle Systems—A Technique for Modeling a Class of Fuzzy Objects. *ACM Trans. Graph.*, 2:2, 91—108

SELLE, A., RASMUSSEN, N., FEDKIW, R. 2005. A vortex particle method for smoke, water and explosions. *ACM Trans. On Graphics*, 910—914

SHI, L., YU, Y., WOJTAN, C., AND CHENNEY, S. 2005. Controllable Motion Synthesis in a Gaseous Medium. *The Visual Computer*, 21: 7, 474—487

SIMS, K. 1990. Particle animation and rendering using data parallel computation. *Computer Graphics*, 24:4, August, 405—413

TREUILLE, A., MCNAMARA, A., POPOVIĆ, Z., AND STAM, J. 2003. Keyframe control of smoke simulations. ACM Trans. On Graphics, 22:3, July, 716—723