Smoke Simulation For Large Scale Phenomena

Category: Process

Abstract

In this paper, we present an efficient method for simulating highly detailed large scale participating media such as the explosions shown in figure 1. We capture this phenomena by simulating the motion of particles in a fluid dynamics generated velocity field. A novel aspect of this paper is the creation of highly detailed three-dimensional turbulent velocity fields at interactive rates using a low to moderate amount of memory. The key idea is the combination of two-dimensional high resolution physically based flow fields with a moderate sized three-dimensional Kolmogorov velocity field tiled periodically in space. We also present a new technique for rendering the particles as a participating volume.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Ray Tracing;

Keywords: smoke, incompressible flow, Navier-Stokes equations, computational fluid dynamics, semi-Lagrangian methods, stable fluids, vorticity confinement, Kolmogorov spectrum, wind fields

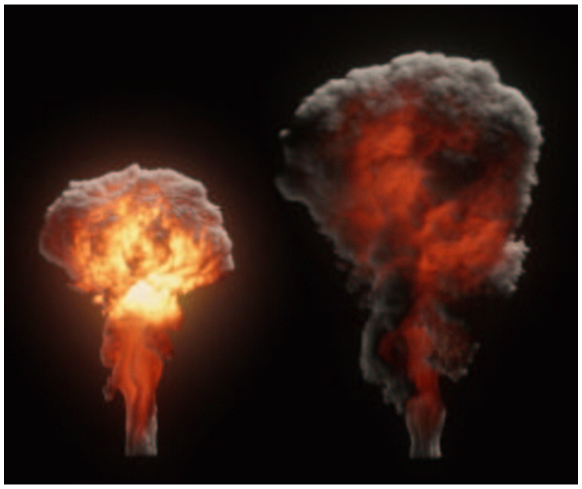
Figure 1: **Large scale explosions.**

1 Introduction

Although numerical simulations of natural phenomena such as smoke, fire and water are now routinely used in the special effects industry, some of the larger scale phenomena remain challenging. For example, it is difficult to simulate the nuclear destruction of an entire city with the level of detail necessary for a feature film. Moreover, reference video footage tends to be of low quality and resolution, so computer simulations of these phenomena are preferable.

Previous method such as [Foster and Metaxas 1997; Stam 1999; Fedkiw et al. 2001] can produce near real time results on small grids and interactive results on moderate sized grids. However, on very large grids of the scale $2000 \times 2000 \times 2000$ these methods are impractical. In fact, a grid of this size requires about 120GB of memory just to store the density and velocity field as floats (and twice that much for doubles), which is well beyond the capability of a high end work station. And while large parallel computers may be available to some, the algorithms tend to scale rather poorly.

Instead of using voxel grids defined throughout space, a more efficient use of memory can be obtained via particle methods. We are not referring to the particles methods such as [Foster and Metaxas 1996; Foster and Fedkiw 2001; Enright et al. 2002] that still require a three-dimensional grid to store the velocity fields, but instead methods that are grid independent such as Smoothed Particle Hydrodynamics (SPH), see e.g. [Gingold and Monaghan 1977; Desbrun and Cani 1996; Hadap and Magnenat-Thalmann 2001]. Unfortunately, the standard SPH approach can be rather expensive when using a large number of particles, since one has to keep track of the nearest neighbors and solve fluid equations for velocity and pressure. As an alternative, we advocate the integration of noninteracting particles forward in time using a wind field that does not require a large three-dimensional grid for its representation. Traditional methods (e.g. [Sims 1990; Wejchert and Haumann 1991; Rudolf and Raczkowski 2000]) create wind fields using the superposition principle inherent to solutions of a simplified Laplace equation model, and then advect either particles or a grid based density through these fields. We instead propose deriving a more realistic and detailed velocity field from a number of two-dimensional fluid dynamics simulations that can be carried out rather efficiently even for high levels of detail. Thus to obtain results on the scale of $2000 \times 2000 \times 2000$, we only need to simulate a few 2000×2000 grids saving about a factor of 2000 in both simulation time and memory. This reduces the amount of memory to around 60 megabytes without sacrificing detail. Of course, we still need to store the position of the particles but only those at an optical depth that the render can "see", i.e. since the particles don't interact, we don't need the ones deep inside (or behind) a plume. This is similar to the optimization of storing the particles only near a water surface proposed in [Foster and Fedkiw 2001].

We also use a periodic spatial tiling of three-dimensional Kolmogorov spectrum, see e.g. [Stam and Fiume 1993; Lamorlette and Foster 2002], in order to add more three-dimensional effects to the simulation, and this requires another 24 megabytes of memory. Both the two-dimensional computational fluid dynamics (CFD) solution and the Kolmogorov velocity field can be precomputed rather quickly (just a few seconds per frame).

The particles are rendered using a view-dependent voxel-based method that is efficient enough to allow sequences of several hundred frames to be rendered overnight at resolutions of over 2000 pixels across while still including all the effects necessary for visual realism. Details in the variation of density and illumination as small as a single pixel are visible. The volume is motion blurred according to both its motion and the camera motion. Direct illumination of the volume by external light sources with correct shadowing is supported, as is incandescence. Diffuse scattering of light within a participating medium is an important effect, and so this is simulated as well.

2 Previous Work

Early approaches focused on the smoke's density (mostly ignoring velocity) [Gardner 1985; Perlin 1985; Ebert and Parent 1990; Sakas 1990], and detail was added using time animated solid textures. [Stam and Fiume 1993] modeled random velocity fields using a Kolmogorov spectrum, and [Stam and Fiume 1995] proposed an advection-diffusion approach for densities composed of "warped blobs" to model gaseous distortions by wind fields.

[Kajiya and von Herzen 1984] were the first to simulate the equations of fluid dynamics directly, and some two-dimensional models were considered in [Yaeger and Upson 1986; Gamito 1995], but one of the more significant works in three-dimensional simulations was [Foster and Metaxas 1996; Foster and Metaxas 1997]. Since their explicit time integration scheme limits the time step increasing the computational cost, [Stam 1999] proposed an unconditionally stable method that uses a semi-Lagrangian advection scheme. [Fedkiw et al. 2001] introduced the notion of a vorticity confinement [Steinhoff and Underhill 1994] to add more small scale rolling motions to these simulations.

The compressible version of the Navier-Stokes equations can be used to model explosions. For example, [Neff and Fiume 1999] model and visualize the blast wave portion of an explosion based on a blast curve approach, and [Yngve et al. 2000] modeled explosions in air coupling their results to the brittle fracture model of [O'Brien and Hodgins 1999]. Since solving the compressible equations requires a very small time step, others have modeled fire (without shock waves) using the incompressible version of the equations. For example, [Nguyen et al. 2002] simulated fire using two phase incompressible flow. Also, a very practical procedural model for fire was proposed in [Lamorlette and Foster 2002].

Another large scale problem (but one that we do not pursue in this paper) is the simulation of clouds. See for example, [Dobashi et al. 2000] who used lattice gas solvers based on cellular automata, and [Miyazaki et al. 2002] who used an approach similar to [Fedkiw et al. 2001] including vorticity confinement.

There are several approaches to the problem of high-quality volume rendering, see e.g. [Brodlie and Wood 2001]. Early work [Blinn 1982; Kajiya and von Herzen 1984] traced rays directly through a set of spherical particles to render a volume defined in this way. However, it is difficult to support the high albedo scattering that occurs in an optically dense medium with this method. An approach that makes the problem more tractable is surface extraction [Wyvill et al. 1986; Lorensen and Cline 1987] where an isosurface is found by thresholding the density function, and then either rendered directly or tessellated and input into a standard renderer. This offers good speed and the ability to render arbitrarily small details efficiently, but it cannot easily model the true volumetric effects required.

Other work used a three-dimensional grid with density and/or lighting information stored in each voxel, see e.g. [Levoy 1988; Stam 1999; Fedkiw et al. 2001]. Isotropic scattering can be calculated by exchange of light between the voxels. The problem with this method is that a large number of voxels are required to represent fine details in the volume particularly if features at varying distances to the camera are visible in a perspective projection. This means that the method generally has very high memory requirements, and is best suited to rendering volumes without detailed density or lighting variations.

Some recent work has used photon mapping [Jensen and Christensen 1998; Fedkiw et al. 2001] to simulate light transport in a participating medium. This uses bidirectional ray tracing with a structure of spatially sorted particles (photons) to store indirect illumination. The method is general enough to handle both isotropic and non-isotropic scattering and volume caustics. However, as the resolution of the rendered image and the detail present in the density of the volume and the lighting increases, the method becomes excessively memory and processor intensive. There is also no explicit mechanism to store density, direct illumination or self-illumination, although the method could be extended to do so.

3 Simulation Method

3.1 Two-dimensional Incompressible Flow

We model the smoke's velocity, $\mathbf{u} = (u, v)$, with the two-dimensional incompressible Euler equations [Landau and Lifshitz 1998]

$$\nabla \cdot \mathbf{u} = 0 \tag{1}$$

$$\mathbf{u}_t = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \nabla p + \mathbf{f} \tag{2}$$

where p is the pressure of the gas, and $\mathbf{f}$ accounts for the external forces. Note that we have arbitrarily set the constant density of the fluid to one. Equation 2 is solved by first computing an intermediate velocity $\mathbf{u}^*$ ignoring the pressure term, and then adding the pressure correction term using $\mathbf{u} = \mathbf{u}^* - \Delta t \nabla p$ where the pressure is found by solving $\nabla^2 p = \nabla \cdot \mathbf{u}^* / \Delta t$. We use a semi-Lagrangian stable fluids approach to find the intermediate velocity $\mathbf{u}^*$ and solve the linear system of equations for the pressure using a preconditioned conjugate gradient method. See [Stam 1999; Fedkiw et al. 2001] for the details.

The smoke's temperature and density are passively convected by the velocity field, $T_t = -(\mathbf{u} \cdot \nabla)T$ and $\rho_t = -(\mathbf{u} \cdot \nabla)\rho$, and thus both can be solved for using the semi-Lagrangian stable fluids method. Heavy smoke tends to fall downwards due to gravity while hot gases tend to rise due to buoyancy. Although we don't account for the temperature in the simulation (only using it for rendering), the external buoyancy force is directly proportional to the density, $\mathbf{f}_{buoy} = -\alpha\rho\mathbf{z}$ where $\mathbf{z} = (0, 1)$ points in the upward vertical direction and α is positive constant with appropriate units.

Nonphysical numerical dissipation damps out interesting flow features, and we use vorticity confinement (see [Steinhoff and Underhill 1994; Fedkiw et al. 2001; Nguyen et al. 2002]) to generate the swirling effects. First the vorticity $\omega = \nabla \times \mathbf{u}$ is identified as the ("paddle-wheel") source of this small scale structure, and then normalized vorticity location vectors, $\mathbf{N} = \nabla|\omega| / |\nabla|\omega||$ that point from lower to higher concentrations of vorticity are constructed. The magnitude and direction of the vorticity confinement force is computed as $\mathbf{f}_{conf} = \varepsilon h (\mathbf{N} \times \omega)$ where $\varepsilon > 0$ and is used to control the amount of small scale detail added back into the flow, and the dependence on the grid size h guarantees that the physically correct solution is obtained as the mesh is refined. Figure 2 shows the results of a two-dimensional simulation.

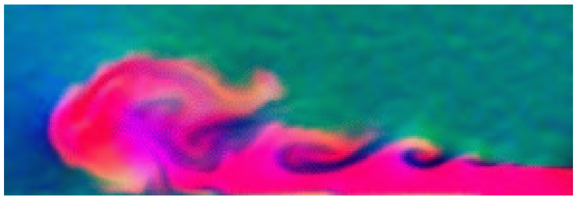
Figure 2: **Density contours from a two-dimensional simulation.**

3.2 Interpolation

After generating a few two-dimensional velocity fields (with density and temperature), we define a three-dimensional velocity field via interpolation. Traditional methods created wind fields using the superposition principle inherent to solutions of a simplified Laplace equation model, and then advected either particles or a grid based density through these fields. More recently there has been a surge of CFD algorithms, e.g. [Foster and Metaxas 1997; Stam 1999; Fedkiw et al. 2001; Nguyen et al. 2002] arising as competing methods. While wind field methods are much faster than their CFD counterparts, the CFD algorithms produce highly detailed and realistic flow fields. Thus, we propose using the two-dimensional versions of these CFD algorithms as two-dimensional cross-sections of flow fields in three-dimensional space, and define interpolation methods to fill in the empty regions between these two-dimensional cross-sections. That is, we piece together velocity fields in three-dimensional physical space with the aid of interpolation, as opposed to piecing together analytic flow fields in solution space as is typical for the traditional wind field methods. Many of the benefits of both approaches are retained. For example, there is no need to solve the Navier Stokes equations in three spatial dimensions. Only very efficient two-dimensional simulations are needed, and thus the bottle neck is the particle advection for both our method and the traditional wind field approach. Moreover, we obtain highly detailed physically based velocity fields from full, but two-dimensional, Navier-Stokes simulations removing the need for guessing how to piece together traditional wind field flow fields in solution space to obtain the desired look.

For simplicity, suppose that we begin with two separate (but similar) two-dimensional computations and place them side by side in three-dimensional space as shown on the left hand side of figure 3. Then the velocity field (and temperature and density) at a point P is defined using linear interpolation between the two closest points as shown in the figure. In general, these closest points will not lie exactly on two-dimensional grid points, so linear interpolation is used to find an appropriate value. Interpolation is carried out in time as well, i.e. the two-dimensional solutions are cached once per frame, and subframe values are defined at the closest points using linear interpolation in time. Although one cannot readily define flow fields throughout all of space using this method, the participating media phenomena we are concerned with usually lie in a finite region.

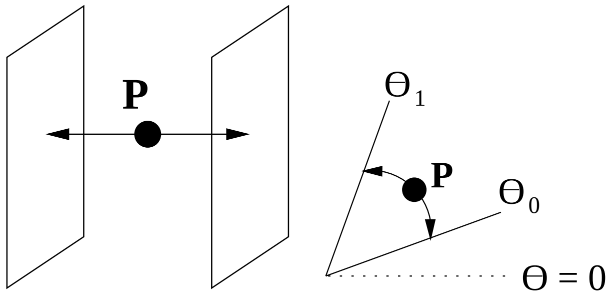

Figure 3: **Parallel interpolation (side view) and cylindrical interpolation (top view), respectively.**

There are many variants on this method, e.g. one could place the two-dimensional solutions in more creative ways, use different methods for interpolation, etc. The right hand side of figure 3 shows a top view of two planes tiled in a cylindrical fashion, and here the interpolation is carried out by finding the closest points along an arc of constant radius from the vertical axis. This variant is especially useful for the many phenomena that have approximate axial symmetry. For example, to generate a three-dimensional smoke plume, we calculate a number of similar (but slightly different) two-dimensional plumes, cut each one in half, and tile them in a circle at varying θ locations. Then the interpolation illustrated in figure 3 is carried out using the two solutions slices that a point happens to fall in between. Note that one could use the axisymmetric Navier Stokes equations here as well, but we have found this unnecessary especially since a number of lower dimensional calculations are required either way to provide a degree of variance from one tile to the next.

3.3 Kolmogorov Spectrum

Since we build a three-dimensional velocity field from two-dimensional solutions of Navier-Stokes equations, it is desirable to add a fully three-dimensional component to the velocity field. This is accomplished using a Kolmogorov spectrum which was described in detail in [Stam and Fiume 1993] and used in many application, e.g. to model fire in [Lamorlette and Foster 2002]. We also note that the Kolmogorov technique is similar to the Phillips spectrum techniques used by [Tessendorf 2002] to simulate water waves on oceans.

The main idea is to use random numbers to construct an energy spectrum in Fourier space that subsequently determines the structure of the velocity field. There are a wide variety of different models in the turbulence literature, but the most popular is probably the Kolmogorov energy spectrum

$$P_h(\mathbf{k}) = \begin{cases} 0 & if \quad k < k_{inertial} \\ 1.5\varepsilon^{\frac{2}{3}}k^{-\frac{5}{3}} & otherwise \end{cases}$$

where energy introduced at frequency $k_{inertial}$ is propagated to higher frequencies at a constant rate ε. After constructing an energy spectrum in Fourier space, one enforces the divergence free condition and uses and inverse FFT to obtain a velocity field full of small scale eddies. See [Stam and Fiume 1993] for more details.

Since the velocity field is periodic, a single grid can be used as a tiling of all of space. Moreover, one can use two grids of different sizes to increase the period of repetition to the least common multiple of their lengths alleviating visually troublesome spatial repetition (although this is a minor point for us since we blend the Kolmogorov velocity field with our non-periodic fluid dynamics generated velocity field). We also fill the time domain by constructing a few Kolmogorov velocity fields, assigning each one to a different point in time, and linearly interpolating between them at intermediate times. In practice, two spectrums are usually enough and we alternate between them every 24 frames. Finally, at any point in space and time, we define the total velocity field as a linear combination of the Kolmogorov field and the wind field constructed from the two-dimensional fluid dynamics simulations.

3.4 Particle Advection

Now that we have a definition of our flow field at every point of interest in three spatial dimensions, we can passively advect particles through the flow using $\mathbf{x}_t = \mathbf{u}$ where $\mathbf{x}$ is the particle position. If desired, copies of two-dimensional flow fields and the three-dimensional Kolmogorov velocity field can be distributed to multiple machines where particles can be passively evolved with no intercommunication requirements. This allows one to generate an incredibly large numbers of particles, although we have found that even one processor can generate enough particles to move the bottleneck to the rendering stage.

There are many advantages to using particles to represent the flow field. For example, one can rapidly visualize the results of a calculation by simply drawing points at every particle location, density and temperature fields can be interpolated from the two-dimensional grids to the particle locations and stored there for subsequent rendering, and both an orientation and an angular velocity can be evolved with each particle to provide additional information (such as an evolving coordinate system, see e.g. [Szeliski and Tonnesen 1992]) for the rendering stage.

4 Rendering

In order to alleviate the memory difficulties associated with mapping all the particles to a large three-dimensional voxel grid for rendering, we use a truncated pyramid shaped grid aligned with the view frustum, i.e. each voxel is a small truncated pyramid. This offers several advantages. As the grid is aligned with the pixels of the image, aliasing artifacts are greatly reduced. Features near the camera are automatically resolved with the appropriate higher level of detail. The resolution of the grid can be relatively low in the axis perpendicular to the view plane resulting in a smaller grid with a corresponding reduction in memory use and render time. Note that in addition to the particle densities, the voxel grid also stores the total radiance (sum of direct illumination, incandescence and scattered light). The direct illumination and incandescence are calculated first, and then diffuse scattering of light within the volume is simulated.

4.1 Particle Sampling

The first step in rendering the volume defined by the particles is to sample the particle density onto the frustrum shaped voxel grid. This can be done in many ways depending on the application, but we sample each particle as one or more ellipsoids. Each ellipsoid is assigned a radius on each axis of its particles local coordinate system, and has a density at a point $\mathbf{x}$ in the local coordinate system given by $D(\mathbf{x}) = 1 - f(1-s, 1, |\mathbf{x}|/r)$ where $0 \le s \le 1$ is a softness factor, r is the particle radius, and f is defined as

$$f(a,b,t) = \begin{cases} 0 & if \quad t \le a \\ 1 & if \quad t \ge b \\ -2\left(\frac{t-a}{b-a}\right)^3 + 3\left(\frac{t-a}{b-a}\right)^2 & if \quad a < t < b \end{cases} .$$

A turbulence function is used to modulate the density function to add extra detail [Perlin 1985]. We also use motion blur to sample the ellipsoids onto the voxel grid according to the relative motion between the corresponding particle and the camera.

4.2 Lighting Model

The transfer of light in a participating medium is described by [Chandrasekhar 1960]. For a medium that emits light, we can express the change in radiance at a point $\mathbf{x}$ as [Jensen and Christensen 1998]:

$$\frac{\partial L(\mathbf{x},\omega)}{\partial \mathbf{x}} = \sigma_a L_e(\mathbf{x},\omega) - \sigma_t L(\mathbf{x},\omega) + \sigma_s \int_{4\pi} p(\mathbf{x},\omega,\omega')L(\mathbf{x},\omega')d\omega'$$

where $p(\mathbf{x},\omega,\omega')$ is a phase function that describes how much of the light at point $\mathbf{x}$ from direction ω is scattered in the direction ω'. This equation is in general difficult to solve. However, in the case of a high albedo medium, it can be shown that that the scattering becomes effectively isotropic and can be modeled as a diffusion process [Stam and Fiume 1995; Jensen et al. 2002].

Direct illumination of the volume is calculated at each voxel that has a non-zero density by tracing rays to each light source in the scene to calculate attenuation of the incoming light. The light from the source is attenuated as the ray is traced through the voxel grid using the method described in section 4.3 below. The volume can also be incandescent, and this is simulated by treating each particle as a small blackbody radiator that illuminates the surrounding voxels. The radius and shape of the illumination function is based on the density function of the particle and we allow the user explicit control over the mapping from blackbody temperature to color values. The self-illumination is then accumulated with the direct external illumination stored in the voxels.

Having stored illumination values at each voxel, we then simulate isotropic light scattering as diffusion. We use a simple but efficient method that assumes that light is scattered from each voxel uniformly in all directions, and attenuated exponentially with optical distance as in equation 3 (below). Since a high albedo medium can scatter light over large distances, we accelerate this using a hierarchical method [Jensen and Buhler 2002].

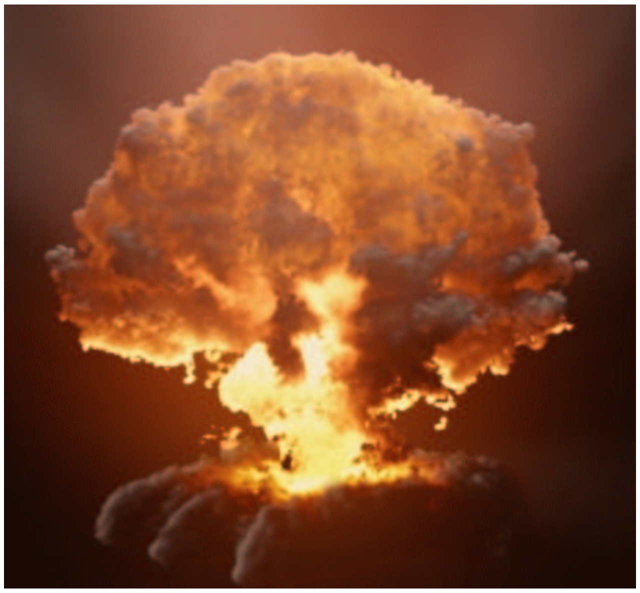
Figure 4: **Nuclear explosion.**

4.3 Ray Marching

The volume is rendered by tracing rays from the camera through the voxel grid. As the voxel grid is aligned with the view frustum, ray traversal is extremely efficient (just incrementing an index). We accumulate color and opacity, where an opacity of 0 corresponds to zero attenuation, and an opacity of 1 corresponds to complete attenuation. The opacity of a voxel centered at point $\mathbf{x}$ is calculated as $a = 1 - exp(-\tau D(\mathbf{x})dz)$ where $D(\mathbf{x})$ is the density of the voxel, dz is the depth of the voxel in the direction of the ray and τ is a constant that controls the conversion from density to opacity. We accumulate opacity along the ray by

$$A_{n+1} = A_n + a(1 - A_n). \tag{3}$$

An advantage of ray tracing in this direction is that the trace can be terminated as soon as full opacity $(> .999)$ is reached, and this happens relatively early for many of the phenomena that we are interested in. At each point we also accumulate the illuminated color of the volume, weighted by the opacity calculated as above. This is $C_{n+1} = C_n + a(1 - A_n)I(\mathbf{x})$ where $I(\mathbf{x})$ is the stored illumination at the voxel. This gives us a correctly premultiplied image which can be directly composited over other elements in the scene.

5 Results

We have run all of our simulations on a Pentium4 2.2GHz or comparable machine. Figure 1 shows a large explosion obtained by advecting a few million particles through a flow field constructed from two dimensional 250×250 grid cell simulations and a $128 \times 128 \times 128$ Kolmogorov grid. The simulation times were a few seconds per frame for both the two-dimensional simulations and the Kolmogorov spectrum, but the particle advection took an average time of about two minutes per frame since we used as many as 6 million particles. Figure 4 is characteristic of an even larger scale nuclear explosion, and we increased the resolution of the two dimensional simulations to about 500×500 grid cells which still simulated in less than 10 seconds per frame. (We tried two-dimensional simulations as large as 2000×2000 grid cells, and the computational cost was only about two minutes per frame.) Both the size of the Kolmogorov grid and the number of particles were approximately the same as in the first example. Note that this example uses one radially interpolated set of velocity fields for the large plume and another for the ground elements. Rendering times for all simulations ranged from 5-10 minutes per frame.

6 Conclusions and Future Work

In this paper, we modeled large scale phenomena using the combination of a few highly detailed two-dimensional flow fields and a moderate sized three-dimensional Kolmogorov velocity field. This technique is a few orders of magnitude more efficient than fully three-dimensional fluid dynamics calculations, and thus can obtain a level of detail unattainable using those methods.

It is easy to sculpt and control flow fields in two spatial dimensions, since the results can often be obtained in real or interactive time. Moreover, one can create a library of two-dimensional simulations complete with the parameters used to generate them, and then an animator can more easily choose a starting point for constructing future simulations.

We also described a view-dependent rendering method that is particularly efficient for rendering large detailed volumes from a particular perspective viewpoint. As future work, it might be interesting to combine this view dependent technique with the photon mapping method.

References

BLINN, J. F. 1982. Light Reflection Functions for Simulation of Clouds and Dusty Surfaces. *Computer Graphics 16*, 3, 21–29.

BRODLIE, K., AND WOOD, J. 2001. Recent Advances in Volume Visualization. *Computer Graphics Forum 20*, 1, 125–148.

CHANDRASEKHAR, S. 1960. *Radiative Transfer*. Dover, New York.

DESBRUN, M., AND CANI, M.-P. 1996. Smoothed particles: A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation '96 (Proceedings of EG Workshop on Animation and Simulation)*, Springer-Verlag, R. Boulic and G. Hegron, Eds., 61–76. Published under the name Marie-Paule Gascuel.

DOBASHI, Y., KANEDA, K., OKITA, T., AND NISHITA, T. 2000. A Simple, Efficient Method for Realistic Animation of Clouds. In *SIGGRAPH 2000 Conference Proceedings, Annual Conference Series*, 19–28.

EBERT, D. S., AND PARENT, R. E. 1990. Rendering and Animation of Gaseous Phenomena by Combining Fast Volume and Scanline A-buffer Techniques. In *Proceedings of SIGGRAPH 1990*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 357–366.

ENRIGHT, D., MARSCHNER, S., AND FEDKIW, R. 2002. Animation and Rendering of Complex Water Surfaces. In *Proceedings of SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 736–744.

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual Simulation of Smoke. In *Proceedings of SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, E. Fiume, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 15–22.

FOSTER, N., AND FEDKIW, R. 2001. Practical Animation of Liquids. In *Proceedings of SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 23–30.

FOSTER, N., AND METAXAS, D. 1996. Realistic Animation of Liquids. *Graphical Models and Image Processing 58*, 471–483.

FOSTER, N., AND METAXAS, D. 1997. Modeling the Motion of a Hot, Turbulent Gas. In *Proceedings of SIGGRAPH 1997*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 181–188.

GAMITO, M. N. 1995. Two dimensional Simulation of Gaseous Phenomena Using Vortex Particles. In *Proceedings of the 6th Eurographics Workshop on Computer Animation and Simulation*, Springer-Verlag, 3–15.

GARDNER, G. Y. 1985. Visual Simulation of Clouds. In *Proceedings of SIGGRAPH 1985*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 297–384.

GINGOLD, R. A., AND MONAGHAN, J. J. 1977. Smoothed Particle Hydrodynamics-Theory and application to nonspherical stars. *Mon. Not. R. Astron. Soc 181*, 375.

HADAP, S., AND MAGNENAT-THALMANN, N. 2001. Modeling Dynamic Hair as a Continuum. *Computer Graphics Forum 20*, 3.

JENSEN, H. W., AND BUHLER, J. 2002. A Rapid Hierarchical Rendering Technique for Translucent Materials. In *Proceedings of SIGGRAPH 2002*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 576–581.

JENSEN, H. W., AND CHRISTENSEN, P. H. 1998. Efficient Simulation of Light Transport in Scenes with Participating Media using Photon Maps. In *SProceedings of SIGGRAPH 2002*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 311–320.

JENSEN, H. W., MARSCHNER, S., LEVOY, M., AND HANRAHAN, P. 2002. A Practical Model for Subsurface Light Transport. In *Proceedings of SIGGRAPH 2002*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 511–518.

KAJIYA, J. T., AND VON HERZEN, B. P. 1984. Ray Tracing Volume Densities. In *Proceedings of SIGGRAPH 1984*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 165–174.

LAMORLETTE, A., AND FOSTER, N. 2002. Structural Modeling of Flames for a Production Environment. In *Proceedings of SIGGRAPH 2002*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 729–735.

LANDAU, L. D., AND LIFSHITZ, E. M. 1998. *Fluid Mechanics, 2nd edition*. Butterworth-Heinemann, Oxford.

LEVOY, M. 1988. Display of Surfaces from Volume Data. *IEEE Computer Graphics and Applications 8*, 3, 29–37.

LORENSEN, W., AND CLINE, H. 1987. Marching cubes: A high-resolution 3d surface construction algorithm. *Computer Graphics (SIGGRAPH Proc.) 21*, 168–169.

MIYAZAKI, R., DOBASHI, Y., AND NISHITA, T. 2002. Simulation of Cumuliform Clounds Based on Computational Fluid Dynamics. *Proc. EUROGRAPHICS 2002 Short Presentation*, 405–410.

NEFF, M., AND FIUME, E. 1999. A Visual Model for Blast Waves and Fracture. In *Proceedings of Graphics Interface 1999*, 193–202.

NGUYEN, D., FEDKIW, R., AND JENSEN, H. W. 2002. Physically Based Medeling and Animation of Fire. In *Proceedings of SIGGRAPH 2002*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 721–728.

O'BRIEN, J. F., AND HODGINS, J. K. 1999. Graphical Modeling and Animation of Brittle Fracture. In *Proceedings of SIGGRAPH 1999*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 137–146.

PERLIN, K. 1985. An Image Synthesizer. In *Proceedings of SIGGRAPH 1985*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 287–296.

RUDOLF, M. J., AND RACZKOWSKI, J. 2000. Modeling the Motion of Dense Smoke in the Wind Field. *Computer Graphics Forum 19*, 3.

SAKAS, G. 1990. Fast Rendering of Aritrary Distributed Volume Densities. In *Proceedings of Eurographics 1990*, 519–530.

SIMS, K. 1990. Particle Animation and Rendering Using Data Parallel Computation. *Computer Graphics 24*, 4, 405–413.

STAM, J., AND FIUME, E. 1993. Turbulent Wind Fields for Gaseous Phenomena. In *Proceedings of SIGGRAPH 1993*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 369–376.

STAM, J., AND FIUME, E. 1995. Depicting Fire and Other Gaseous Phenomena Using Diffusion Process. In *Proceedings of SIGGRAPH 1995*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 129–136.

STAM, J. 1999. Stable Fluids. In *SIGGRAPH 99 Conference Proceedings, Annual Conference Series*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 121–128.

STEINHOFF, J., AND UNDERHILL, D. 1994. Modification of the Euler Equations for "Vorticity Confinement": Application to the Computation of Interacting Vortex Rings. *Physics of Fluids 6*, 8, 2738–2744.

SZELISKI, R., AND TONNESEN, D. 1992. Surface modeling with oriented particle systems. *Computer Graphics (SIGGRAPH Proc.)*, 185–194.

TESSENDORF, J. 2002. Simulating Ocean Water. In *Simulating Nature: Realistic and Interactive Techniques*, SIGGRAPH 2002, Course Notes 9.

WEJCHERT, J., AND HAUMANN, D. 1991. Animation Aerodynamics. *Computer Graphics 25*, 4, 19–22.

WYVILL, B., MCPHEETERS, C., AND WYVILL, G. 1986. Animating Soft Objects. *The Visual Computer 2*, 4, 235–242.

YAEGER, L., AND UPSON, C. 1986. Combining Physical and Visual Simulation - Creation of the Planet Jupiter for the Film 2010. In *Proceedings of SIGGRAPH 1986*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 85–93.

YNGVE, G. D., O'BRIEN, J. F., AND HODGINS, J. K. 2000. Animating Explosions. In *Proceedings of SIGGRAPH 2000*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 29–36.