Cartoon Rendering of Smoke Animations

Andrew Selle* Stanford University
Alex Mohr Pixar
Stephen Chenney UW Madison

Abstract

We describe a technique for generating cartoon style animations of smoke. Our method takes the output of a physically-based simulator and uses it to drive particles that are rendered using a variant of the depth differences technique (originally used for rendering trees). Specific issues we address include the placement and evolution of primitives in the flow and the maintenance of temporal coherence. The results are visually simple, flicker-free animations that convey the turbulent, dynamic nature of the gas with simple outlines.

CR Categories: I.3.0 [Computer Graphics]: General

Keywords: smoke rendering, smoke simulation, cartoon rendering, non-photorealistic rendering

1 Introduction

Consider a puff of smoke. A cartoonist might draw a swirly boundary with a solid white interior, and maybe some internal swirls. A photograph of smoke reflects complex volumetric light interactions, driven by the equations of fluid dynamics. For stylized animations we prefer the abstract, swirly form, while for special effects a photorealistic result is desired.

This paper describes a technique for cartoon rendering of animated smoke. Our aim is to generate stylized animations such as the one shown on the left in Figure 1, which conveys the turbulent, fluid nature of the gas using only solid blocks of color and hard silhouettes. Our method is driven by the output of a physically-based simulator, ensuring that the animated flow is consistent with gaseous behavior. We also have access to physical properties of the flow, such as density, which we can use to modify the rendering as in Figure 1.

Our contribution is a method for producing an artistic, animated smoke rendering from a physically-based simulation. The simulator (in our case Fedkiw et. al.'s [2001] algorithm) ensures that the basic fluid motion is plausible; the velocity field evolves correctly, mass is conserved, and scalar fields such as pressure and temperature are consistent. We embed particles in the fluid and advect them with the flow. These are then rendered using an extension of Deussen and Strothotte's [2000] *depth differences* tree rendering technique. Particle primitives are rendered and then pixels whose depth differs significantly from their neighbors are drawn as silhouettes. Several innovations were required to adapt the basic algorithm to smoke animation, including a method for evolving the primitives to reflect properties of the flow (densities, temperatures, etc.), shaping and orienting particles dynamically to convey information, and strategies for maintaining temporal coherence.

*email: aselle@cs.stanford.edu

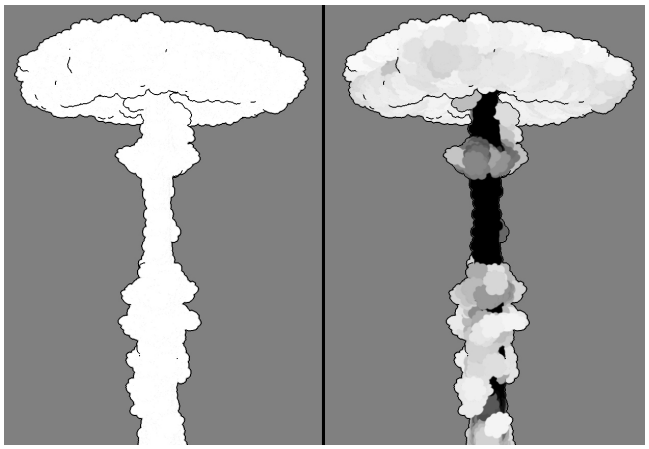

Figure 1: *Left: frame from a fluid animation generated by our system, in an abstract rendering style reminiscent of hand-drawn smoke and clouds. Right: another frame from our system, colored by velocity magnitude. Our stylized renderings capture the same sense of billowing, turbulent flow as photorealistic smoke.*

1.1 Related Work

Particle systems are typically used to simulate and render smoke in real time applications, such as computer games. Procedural rules guide the evolution of billboard particles that are blended together to give the appearance of smoke. A stylized look can be achieved using non-photorealistic billboards; Lamorlette and Foster's [2002] fire modeling work is a very high quality example of this approach. An alternative for stylized rendering is advected non-photorealistic textures [Witting 1999; Neyret 2003], in which a texture is deformed by the flow. This approach, however, typically removes detail as the flow evolves and the texture mixes.

Our target styles, however, require hard edges in some regions to define the shape and motion of the flow, but in coexistence with large low detail regions. Blended billboard or texture approaches cannot easily produce this result: rendering hard-edge billboards will produce unnecessary detail in the middle of the cloud, while soft edge billboards would fail to define the edges. It is similarly difficult to define textures to capture this effect and maintain it as the flow evolves. Billboard particle systems have the second disadvantage of requiring a particle-based procedural model of the smoke's motion, which is not always easy to derive. While our work is at heart a particle-based approach, we use physically-based simulations to ensure plausible flow, and for the first time apply a depth differences algorithm to the rendering of smoke billboards.

Yu et. al. [2000] describe a procedural model for generating still images of cartoon smoke. They define a spine for the smoke-trail consisting of a sine curve with frequency and amplitude that increases with height from the source. Their model only applies to

smoke rising from a source (a cigarette or smokestack) and is not designed for animation.

There has been extensive work on the photorealistic rendering of smoke and other gaseous flows. Fedkiw, Stam and Wann Jensen's [2001] work represents the current state of the art in photorealistic smoke. This and other methods [Kajiya and Von Herzen 1984; Gardner 1985; Sakas 1990] simulate or approximate the scattering, emission and adsorption of light by smoke particles. These methods are slow compared to billboard and texture approaches, and it is not clear how to adapt them for stylized renderings.

Non-photorealistic rendering has previously been applied to the problem of visualizing fluids, with the aim of improving a viewer's understanding of the underlying physics. Kirby, Marmanis and Laidlaw [1999] encode a range of flow-derived variables in both the strokes and layers of a painting. For example, stroke size and orientation shows the velocity field magnitude and direction, while an under-painting shows the vorticity direction. Healey and Enns [2002] present weather data visualizations that use texture and stroke to convey information. While these techniques produce images that are both visually appealing and scientifically useful, we aim for a more abstract representation that explicitly hides information in the rendering.

The following section discusses the details of our technique. Section 3 and our video show some results, while Section 4 discusses the limitations of our work and suggests directions for improvement.

2 Implementation

Our smoke rendering system consists of three main components: a simulator for driving the motion of embedded smoke particles and associated scalar fields (such as temperature); a rendering system that uses depth differences [Deussen and Strothotte 2000] to draw the particles in a cartoon style; and an interface layer that manages the evolution of the particles over time.

2.1 Smoke Simulation

The smoke simulator must produce motion for the smoke particles, as well as associated data such as temperatures. Particle systems are one common approach to this problem, having the advantage of intuitive parameterized control. However, for a general purpose smoke simulator it is unclear how to create realistic rules that capture all the possible effects while retaining usability. The need to track scalar fields adds further complications.

Physical models have the advantage of producing realistic particle motion and any associated parameters. Recent advances in physically-based fluid simulations have made them a viable method for real-time smoke animation. In particular, Stam's [1999] semi-Lagrangian *stable fluids* has made smoke simulation more robust and less error-prone. Fedkiw, Stam and Wann Jensen [2001] improved upon Stam's approach. For larger simulations, Rasmussen et. al.'s [2003] layering approach enables efficient, realistic simulation. Finally, Treuille et. al. [2003] introduced a method for the key-frame control of smoke simulations.

We generate our smoke motion with the physically-based method of Fedkiw et. al. [Fedkiw et al. 2001]. However, we require only the *output* of a smoke simulator – we do not modify the simulation algorithm – so any procedural or physically-based approach could be used with our system if it generates the required data.

To generate a particle set, we simply run the smoke simulation introducing massless marker particles periodically at the smoke source. These particles are advected through the simulation velocity vector field using Euler integration. Position, velocity, and density are linearly interpolated from the simulation grid. This is easy to implement because semi-Lagrangian solvers already include particle tracing as one of the basic building blocks of the simulation.

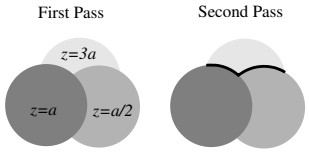

Figure 2: *The image on the right shows the result of the depth differences algorithm performed on the left image with threshold value $\gamma = a$. Note the pixels with depth difference greater than γ are colored black.*

2.2 Rendering

To produce a cartoon style rendering we must generate sharp silhouettes at interesting points in the smoke, but otherwise use constant color chunks to indicate the body of smoke. It is particularly important to use the silhouettes to convey the appearance of turbulence and vortices.

Particle based renderings are advantageous in that, moving in concert, they tend to capture rapid changes such as turbulence and vortices. One approach might be to identify the interesting particles in the smoke and restrict silhouette rendering to those. However, it is difficult to identify such regions by looking at only one particle at a time. It is also not clear how to set the thresholds for interesting behavior, as these might change throughout the simulation.

An alternative is to forgo particles and work with isosurfaces derived from the simulation. The silhouettes of such surfaces could be rendered directly. However, isosurfaces are not well resolved from the relatively coarse grid of a physically-based simulation model, and particle-system models are ruled out entirely. Additionally, using an isosurface makes the implicit and incorrect assumption that all the interesting behavior of smoke occurs at a small, readily defined set of isovalues.

Our system works with particles, but employs a global algorithm to find places where silhouettes should be drawn. We adapt the depth differences technique originally introduced by Deussen and Strothotte [2000] for generating pen-and-ink illustrations of trees. A rendering primitive, such as a disk, is associated with each particle and rendered to the depth and color buffers. A second pass looks at the depth buffer for every pixel. If the central difference of the depth buffer z across any of its four faces exceeds the depth threhsold γ then its color is set to black as shown in Figure 2. Specifically, the color of pixel i, j is overridden to be the silhouette color (e.g. black) if $\left|\frac{z_{i+1,j}-z_{i,j}}{\Delta x}\right| > \gamma$ or $\left|\frac{z_{i,j}-z_{i-1,j}}{\Delta x}\right| > \gamma$ or $\left|\frac{z_{i,j+1}-z_{i,j}}{\Delta y}\right| > \gamma$ or $\left|\frac{z_{i,j}-z_{i,j-1}}{\Delta y}\right| > \gamma$. Otherwise, the color from the first pass is retained.

We created a range of smoke primitives including a circle cloud, a puffy cloud, and a "hairy" cloud shown in Figure 3. We use OpenGL to rasterize these primitives and read back the color and depth buffers to perform the depth difference computation.

The depth differences algorithm works well on a single frame, but when applied to an animated sequence distracting visual artifacts may be introduced. These arise when there are sudden shifts in silhouettes edges due to primitives crossing the depth difference threshold as they move forward and backward. Advecting primitives in the flow removes most of these artifacts because, when the flow is coherent, the depth switches will not cause flickering. Incoherent flow will still cause flickers, but in our case that corresponds to turbulence and vortices – precisely the locations where high frequency effects are desirable.

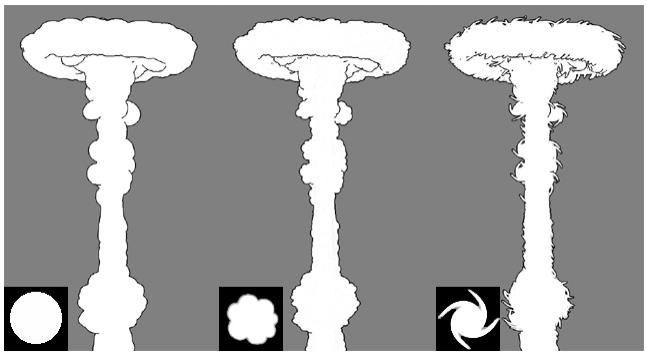

Figure 3: *Three different stencils used as rendering primitives, and the resulting images.*

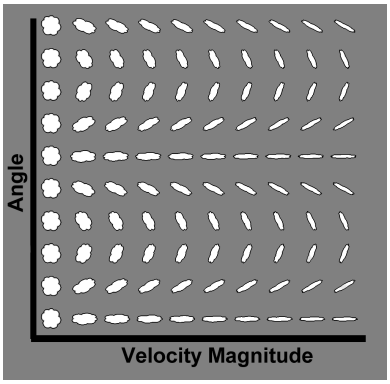

Figure 4: *Velocity direction and magnitude's effect on particle rendering. Note the stretch is proportional to the velocity magnitude while the orientation of the stencil is proportional to the angle.*

Simple advected particles are not sufficient for high quality cartoon renderings. One would like to eventually remove particles if their density falls below a certain threshold. Simply removing the particles will create jumps, so care is required in managing the evolution of particles over time. Correspondingly, particles may not be advected into all areas of density, so new particles should be added in areas of sufficient density.

2.3 Simulation-Rendering Interface

To improve temporal coherence and give more flexibility to the animator we modulate particle drawing based on properties of the particles. The information our particles provide from the simulation suggest intuitive controls to the look of our renderer. All these modulations are modeled as simple linear interpolation ramps, but more complicated schemes are possible.

The density around a particle drives the particle size. The higher the density the larger the particle and thus the more influence it has on the final image. Conversely, as a particle diverges from the main action of the simulation, it enters low density regions and thus is rendered smaller. This not only conveys the influence of particles but also helps temporal coherence. As the particle reaches lower density regions, its resulting small size allows for its elimination without causing a large temporal discontinuity.

The color of the primitive rendering can be modulated by either temperature or density. For example, a nuclear explosion could be modeled with a simulation. At the highest temperature, in the core of the explosion, the color of the primitive could be red fading to black as the soot cools. Or, at the highest density, the smoke could be rendered black, while at the lower density the particle primitive could be rendered white.

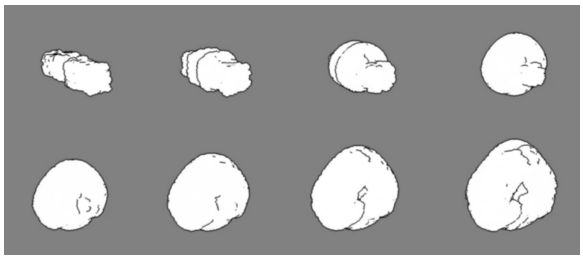

Figure 5: *Two blobs of smoke colliding and forming a donut, showing our renderer handles topological changes gracefully.*

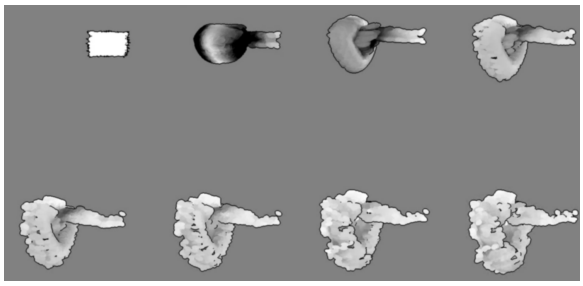

Figure 6: *An explosion where the smoke is propelled sideways until it cools and falls downward.*

The velocity of the particle is used to determine both the rotation of the primitive and the amount of stretch. While the primitive always faces the viewer, there is a degree of freedom in its in-plane rotation which it is important to control. Keeping a fixed alignment produces uninteresting renderings, while choosing it randomly per frame produces temporal incoherence. Instead, we orient the stencil in the velocity direction. In addition to maintaining continuity, it also makes smoke direction changes more visually apparent. Squash and stretch, first introduced to the graphics community by Lasseter [1987] can be introduced by stretching the primitive in the velocity direction. This approach to setting orientation and stretch is similar to that used by Chenney et. al. [2002]. Figure 4 depicts the effect velocity and angle have on stretch and orientation of the rendered particle.

3 Results

We have simulated and rendered several examples to illustrate our system. Each example started with the simulation of the target scenario, and the results were subsequently fed into our renderer. One of the chief advantages of our system is that it can render any smoke behavior that can be generated with the smoke simulator.

In Figure 5, two separate clouds of smoke are propelled towards each other. When they collide, they form a donut. This shows that even when topology changes, our advected particles track the change producing the proper rendering. In Figure 6, a cloud of smoke is violently propelled horizontally. The darkness is proportional to the magnitude of the particle velocity. Figure 7 depicts smoke rapidly emitted from a source at the bottom of the frame. As the smoke moves toward the top of the frame it collides with a ceiling causing the smoke to turn over on itself, causing interesting silhouette contours to form.

The smoke simulation required a maximum of 3 seconds per frame on a 128,000 cell grid (40x80x40 for Figure 7) on a 3 GHz Pentium IV. A 2D layered simulation could probably reduce this to real-time performance. Our unoptimized renderer produced one frame in about a second for a 640x480 image, varying with the number of particles. This time could be reduced by implementing the depth discontinuity algorithm in graphics hardware.

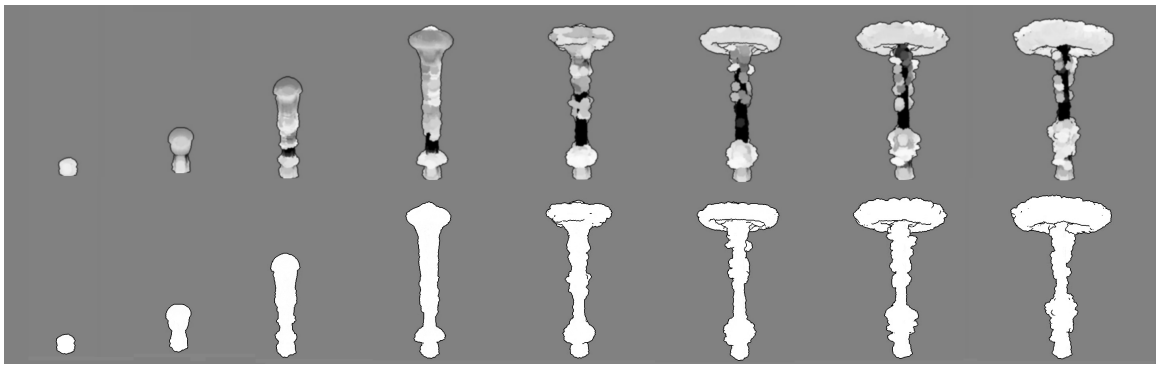

Figure 7: *A smoke animation where the smoke rushes upward and collides with the ceiling causing the smoke to be pushed out and tumble over itself. The top is rendered with color modulated by velocity magnitude while the bottom is the same sequence rendered with a fixed white coloring.*

4 Conclusion

We have introduced a technique for rendering stylized smoke. The underlying dynamics of the smoke is generated with a standard Navier-Stokes fluid simulator and output in the form of advected marker particles. These particles are then rendered as texture mapped 2D stencils, and silhouette edges are added using a depth difference technique to emphasize shape and depth.

The most significant contribution of our technique is dispelling the notion that physical simulation is incompatible with stylized rendering. Indeed, animating smoke by hand is difficult due to the complex, turbulent nature of fluids. Smoke simulation and realistic rendering have successfully alleviated this difficulty for photorealistic applications. Our technique shows that simulation, when combined with a stylized rendering technique, is similarly useful for expressive applications.

Our technique has some limitations. There are several thresholds and parameters to tune. However, while these parameters need to be set, they also provide artists more control over the output. The technique does not work well with varying viewpoints because the feature lines will move as the "depth" direction changes, introducing artifacts due to the viewer motion, rather than the flow. Finally, the 3D smoke simulations are relatively slow, taking several seconds per frame. While layered 2D simulations could be used to increase the speed, this problem suggests examining simpler methods that could replace the physical simulation. In particular, it is unclear whether the full detail generated by the simulation is necessary. Moreover, it may be possible to derive a fast procedural model that captures the salient turbulence and billowing. Despite these limitations, our system produces visually rich and compelling stylized renderings.

Acknowledgments

This work was done at the University of Wisconsin – Madison. Funding and equipment was provided by the NSF, Intel and Microsoft.

References

CHENNEY, S., PINGEL, M., AND IVERSON, R. 2002. Simulating cartoon style animation. In *Proceedings of Non-Photorealistic Animation and Rendering (NPAR) 2002*, 133–138.

DEUSSEN, O., AND STROTHOTTE, T. 2000. Computer-generated pen-and-ink illustration of trees. In *SIGGRAPH 2000 Conference Proceedings*, ACM SIGGRAPH, 13–18.

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

FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. In *SIGGRAPH 2001 Conference Proceedings*, 23–30.

GARDNER, G. Y. 1985. Visual simulation of clouds. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, ACM Press, 297–304.

HEALEY, C. G., AND ENNS, J. T. 2002. Perception and painting: A search for effective, engaging visua lizations. *IEEE Computer Graphics & Applications 22*, 2, 10–15.

JINHUI, Y., XIAOGANG, X., AND QUNSHENG, P. 2000. Computer generation of cartoon smoke. *Journal of Computers 23*, 9, 987–990.

KAJIYA, J. T., AND VON HERZEN, B. P. 1984. Ray tracing volume densities. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, ACM Press, 165–174.

KIRBY, R. M., MARMANIS, H., AND LAIDLAW, D. H. 1999. Visualizing multivalued data from 2D incompressible flows usi ng concepts from painting. In *Proceedings Visualization '99*, 333–340.

LAIDLAW, D. H. 2001. Loose, artistic "textures" for visualization. *IEEE Computer Graphics & Applications 21*, 2, 6–9.

LAMORLETTE, A., AND FOSTER, N. 2002. Structural modeling of flames for a production environment. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 729–735.

LASSETER, J. 1987. Principles of traditional animation applied to 3D computer animation. In *Computer Graphics: SIGGRAPH '87 Conference Proceedings*, 35–44.

NEYRET, F. 2003. Advected textures. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 147–153.

RASMUSSEN, N., NGUYEN, D. Q., GEIGER, W., AND FEDKIW, R. 2003. Smoke simulation for large scale phenomena. *ACM Transactions on Graphics (TOG) 22*, 3, 703–707.

SAKAS, G. 1990. Fast rendering of arbitrary distributed volume densities. In *Proceedings of EUROGRAPHICS '90*, Elsevier Science Publishers B.V.(North-Holland), 519–530.

STAM, J. 1999. Stable fluids. In *Computer Graphics: Proceedings of SIGGRAPH 99*, 121–128.

TREUILLE, A., MCNAMARA, A., POPOVIĆ, Z., AND STAM, J. 2003. Keyframe control of smoke simulations. *ACM Transactions on Graphics (TOG) 22*, 3, 716–723.

WITTING, P. 1999. Computational fluid dynamics in a traditional animation environment. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 129–136.