Fast Animation of Amorphous and Gaseous Phenomena

Scott A. King
Roger A. Crawfis
Wayland Reid
Department of Computer and Information Science, The Ohio State University
Columbus, Ohio, United States of America

Abstract

We present a technique to animate amorphous materials such as fire, smoke and dust in real-time on graphics hardware with dedicated texture memory. Our method uses a coarse voxel grid to model object dynamics, and texture cycling to create local and global dynamics. Detail is added by encoding high-frequency components, which are normally spread uniformly throughout the volume, into the volume integration. The individual voxels are rendered using a splatting approach with a table of anisotropic footprint functions. Our method produces a truly three-dimensional volume effect that can interact with the rest of the environment.

Using different spectral scales for the volume's appearance allows for motion at three distinct and disjoint scales. Local dynamics are achieved by phase-shifting through a set of textures within a voxel. Global dynamics, such as eddies, are propagated through the volume using inter-voxel dynamics. Object dynamics are achieved using procedural or keyframe animation techniques on the low-resolution voxel grid. We also develop an automated technique for texture selection by sampling a single large image having various frequency components.

Keywords: Fire, smoke, clouds, gaseous phenomena, volume rendering, atmosphere, splatting, textured splats, animation.

1 Introduction

Virtual environment technology such as flight simulators, medical simulators and games is making vast steps in realism and speed. For convincing interactive environments, gaseous effects, such as fire, dust and smoke, are needed. To achieve this realism, we have several design goals we wish to meet:

- Animation must support real-time applications.
- The method should be able to represent a variety of phenomena.
- The effect must occupy three-dimensional space.
- Other objects in the environment must be able to interact with the effect.
- A variety of shapes should be supported.
- The basic shape should allow for easy deformations or propagation.

Rendering gaseous phenomena has been an active area of research since the visualization of the rings of Saturn [1] and the Genesis effect in the film Star Trek II: The Wrath of Khan [2]. Blinn creates new illumination methods to handle the scattering of light [1]. Reeves uses particle systems to simulate fire, grass and trees [2][3]. Kajiya ray traces volume densities [4]. Perlin procedurally generates realistic 2D fires based on a noise function [5], which is extended to generate 3D fires using hyper-textures [6]. Gardner uses 3D textures with transparency to represent clouds [7]. Ebert and Parent combine volume ray casting with a scanline A-buffer to render scenes containing both volume and geometry models [8].

Recent research has focused on physically-based modeling. Stam and Fiume reformulate the advection-diffusion equations for densities composed of "warped blobs", which more accurately model the distortions that gases undergo when advected by wind fields, using a model for the flame and its spread [9]. Stam and Fiume formulate global illumination in the presence of gases and fire. Sakas uses spectral turbulence theory using Kolmogorov's exponential law and a phase-shift in the frequency domain [10]. Sakas develops a spectral synthesis model to generate a voxel containing densities and stochastically places and migrates eddies and other turbulent features. Chiba et al. model vortices in a turbulent field with particles acting as tracers within this field giving both vortices and particles behavior which allow them to appear, disappear, and interact with each other and the environment [11]. Foster and Metaxes use specialized forms of the equations of motion for a hot gas, solving the differential equations at low resolutions for speed [12]. The method of Foster and Metaxes is useful for rendering rolling, billowing gases.

The performance of the above systems is dependent on the resolution of the volume grid. Increasing the resolution slows down the generation and rendering process substantially, especially in 3D. For example, Sakas achieves near real time performance with a medium resolution (64x64) 2D field [10]. Sakas reports when the method is extended to 3D the computation time increases exponentially, achieving near real-time speeds only for 8x8x8 fields, while high resolution grids require minutes to calculate. For a medium resolution grid (60x60x45), Foster requires 49 seconds to calculate and 23 minutes to render (ray-tracing) [12]. These physically-based methods [10,11,12] create volumetric models of fire which then need to be rendered, making them candidates to be combined with our rendering method. This combination will be discussed further in section 6.

The above methods either focus on rendering [1, 3, 8, 9] or on modeling [2, 4, 7, 9, 10, 11, 12] the effect. To date, most, if not all techniques require a high number of primitives to generate realistic detail, or assume the camera is viewing the material from a distance. This paper focuses on very fast, visually realistic animation of three-dimensional amorphous materials, that allows objects or the camera both inside and

outside the volume. We use volume rendering of a coarse voxel grid for speed of rendering, in combination with texture cycling for apparent motion.

Several different volume rendering techniques have been developed for regular grids: raycasting [13,14], shear-warp [15], Fourier-domain rendering [16], cell projection [17, 18], and splatting [19]. Our approach capitalizes on the efficient voxel-based projection paradigm of the splatting algorithm.

In splatting, the volume is thought of as a field of overlapping interpolation kernels *h*. One such kernel is placed at each voxel location *j* and weighted by the voxel's value v_j. The ensemble of overlapping voxel kernels then reconstructs a continuous representation of the volume. The task of volume rendering can be interpreted as the process of casting viewing rays into the volume and integrating the volume along these rays. Raycasting calculates this integral by sampling the volume along the ray and compositing the samples in front-to-back order. This sampling operation is an expensive one, and furthermore, a voxel may be involved in many such sampling operations, depending on the sampling distance. Splatting provides a more efficient way to generate the ray integral. It works by observing that a voxel v_j's contribution to a ray is given by

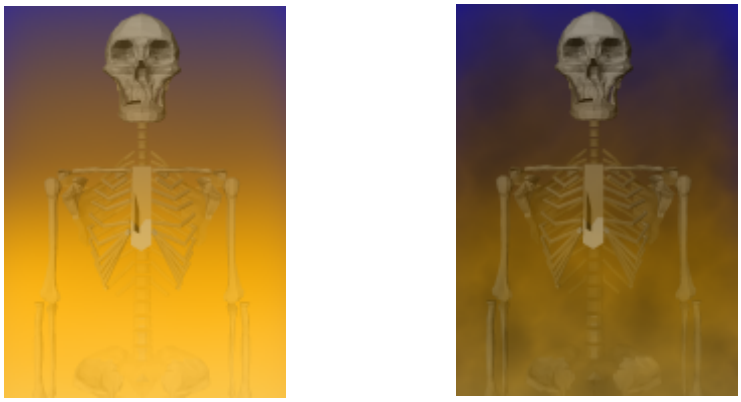
$$v_j \cdot \int h(s)ds$$

where *s* follows the integration of the interpolation kernel in the direction of the viewing ray. If the viewing direction is constant for all voxels or if the interpolation kernel is radially symmetric, then we may pre-integrate

$$\int h(s)ds$$

into a lookup-table, termed a kernel *footprint*, and use this table for all voxels. For volume projection, we map the voxel footprints, scaled by the voxel values, to the screen where they accumulate into the projection image [20]. We use the splatting algorithm in its first incarnation, i.e., in the *composite-every-sample* mode [19], where each footprint is considered an atomic entity and is immediately composited on the image plane, in back-to-front order. We see, that in contrast to raycasting, splatting considers each voxel only once (for a 2D interpolation on the screen), and not several times (for a 3D interpolation in world space). Also, in contrast to raycasting, line integrals across a voxel are now continuous or approximated with good quadrature, and don't require normalization of α to compensate for sample distance. We also only need to project the voxels with relevant values, which reduces the projection task tremendously. Finally, the efficient pre-integrated kernel representation allows splatting to use qualitatively better kernels (with larger extents) than the trilinear filter typically employed by raycasting. Possible kernels are the Gaussian function or the Crawfis-Max kernel [21], which is a kernel optimized for splatting and designed to yield a low-variance volume reconstruction by the field of interpolation kernels.

Laur and Hanrahan [20] extended this technique for octree data structures. They also approximated the Gaussian with a triangular mesh of varying opacities. Crawfis and Max support octree data structures and employ texture mapping to render the splats [21]. Crawfis and Max develop an optimal splat with a small footprint extent to render smoothly varying functions. Crawfis and Max added anisotropic icons within



(a) (b)

Figure 1. Modeling fire with a 7x20x5 voxel grid. a) Using traditional splatting. b) Using our method of incorporating detail into the splats.

the textures to represent vector fields by using a simple phase-shift through the overlapping textures, which provided the illusion of coherent motion. In this paper, we provide a framework for texture synthesis and animation of gaseous volumes that is closely related to the textured splats in [21].

The rest of the paper will discuss our technique. Section 2 describes our algorithm in detail and compares it to more general volume rendering. Section 3 explains the three types of motion possible with our method. Section 4 discusses the creation of textures, which add the detail to the splat, using various techniques. Section 5 presents some results from our method applied to various types of gaseous effects, with some performance measurements. In section 6 we give our conclusions and discuss future work.

2 Our Approach

We model an effect with a small regular grid, assigning colors and opacities at each voxel. Using an optimal splat footprint function [21] rather than a Gaussian, the splatting algorithm by Westover [19] has been used to render the volume shown in figure 1a. The resulting image resembles more of a semi-transparent blob than a raging fire. In comparison, using the same grid, our new technique achieves the rendering shown in figure 1b. For distant views, the traditional volume rendering may be sufficient, as would a single texture mapped facade. Neither volume rendering nor 2D facades are effective when the viewpoint is near or inside the fire.

The cornerstone of our algorithm is the realization that the detail within an amorphous volume is ill-defined, with edges and shapes (other than the overall gross shape) that are not only difficult to perceive visually, but are also constantly changing. We model an effect with a coarse, regular voxel grid of opacity values. Each voxel or point is then rendered independently in a back-to-front order, with respect to the viewpoint, using the splatting technique. To model fine details, we use anisotropic footprints which provide interpolation kernels that are not monotonic nor smooth,

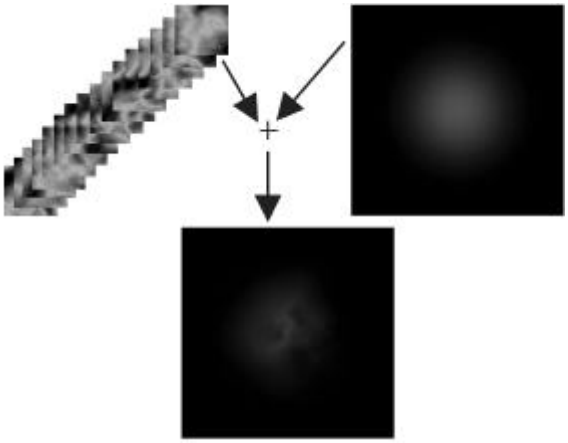


Figure 2. A texture is weighted with the splat footprint function to produce a textured splat containing embedded detail

but rather exhibit a more fractal-like nature. As will be shown in section 5, our technique is applicable to a wide variety of volumetric effects. For purposes of discussion, and without loss of generality, we primarily focus on the techniques to model and render fire in this and the next two sections.

Our basic algorithm consists of the following steps:

1) Select a set of textures to represent the volume and create the footprint table.
2) Create a volume out of regularly spaced voxels.
3) Assign an initial texture to each voxel.
4) Render the current frame, drawing the volume in back-to-front order.
5) Assign new textures to each voxel.

Steps 4 and 5 are done for every frame, and by varying the implementation of steps 3 and 5 and the choice of textures in step 1, different visual effects are achieved.

The first step is to create a small set of textures that contain the frequencies we wish to model. These textures can be created in a variety of ways including procedurally, from measured data, from photos, manually, or by subsampling other textures in various ways. The set of textures is converted to grayscale and windowed with an optimal splat footprint function [21], creating a splat that contains high frequency detail while preserving the overall characteristics of a footprint function. Figure 2 illustrates how a textured splat is constructed from a texture and a footprint. The texture selection process is discussed in section 4.

Step two is to represent the overall shape of the effect as a regular grid. A regular grid allows for quick geometric modeling. We allow for sparse grids by describing the volume as a set of grid points. Opacities represent how much of the gaseous material is present within each voxel.

In step three, the voxels are given an initial textured splat, currently chosen randomly. The dynamic updates, discussed below, quickly change the initial value in most situations, however, the choice should be consistent with the criteria employed in step five.

In step four, the volume is rendered by drawing each voxel's weighted footprint on the screen in a back-to-front manner with respect to the viewpoint. The textured splat is used as the opacity and the color is generated from a color lookup table or the color and opacity can be combined directly into the textured splat. For fire, we developed an inexpensive method to approximate temperature by using the voxel height as the index into the color lookup table. Mixing effects, such as smoke and fire, can be achieved with multiple color tables and differing texture sets.

Finally, in step five, a new texture for each voxel is determined for the next frame. This cycling of textures creates the illusion of motion. The possible dynamics are discussed more fully in section 3.

3 Dynamics

Our method allows the control of motion in three complementary ways. These can be viewed as three different scales or frequencies of motion. For fire, these are the spread of the fire, material density mixture propagation, and the local turbulent mixing. For general applicability, we label these as:

Object Dynamics: Large movements or deformations of the volume which change the perceived shape or location of the entire effect are the object dynamics. These deformations are accomplished by procedurally (or explicitly) changing the coarse voxel grid.

Global Dynamics: The preservation of the relative mixture of a finite volume as it moves from one area of the volume to another area are global dynamics. Turbulent structures, such as eddies can be passed through the volume using this type of dynamics. Global dynamics are accomplished with texture propagation through the volume.

Local Dynamics: Movements within or through a local neighborhood of the volume, the turbulent structures themselves, are local dynamics. For a fire, these movements would be the twirling vortex or licking flames. These are accomplished by phase-shifting through coherent textures to provide the illusion of movement.

Our work primarily addresses the latter two scales of movements since object dynamics is traditional object animation, for which effective solutions exist. A physically-based fire modeling technique [10, 11, 12] can be used for this purpose. The next three subsections will discuss each of these dynamics in turn.

3.1 Object Dynamics

By shifting the detail to the textured splats, we can use very coarse voxel grids for the overall or gross shape of the fire. Many of the images presented in this paper use grid sizes containing less than three-hundred voxels (10x5x6). This allows a simple Eulerian or fluid flow calculation to be performed across the grid, propagating substructures and properties through the mesh. More flexible tools and procedural algorithms

are areas of future interest and are discussed in the conclusions. The focus of this paper is on the local and global dynamics.

3.2 Global Dynamics

Many phenomena exhibit dynamics even if the gross shape of the volume remains static. A pillar of smoke, for instance, has the appearance of upward motion with many complex interactions and details within the rather static plume. These movements must occasionally be handled across voxel cells, such as when a feature migrates through the volume. For the upward motion of fire we match the texture characteristics used from one frame to the next, while propagating those characteristics upward through the volume. This motion is accomplished by grabbing the texture index of the voxel below the current voxel from the previous frame. This texture index is then offset slightly before a final texture is selected and used in the splat. The resulting animations provide a smooth appearance of motion.

3.3 Local Dynamics

For local motion, we can employ texture animation techniques to cycle through a set of textures. If these textures have a cyclical pattern to them, a continuous and smooth animation results. A surprising result is the apparent motion perceived when textures were selected randomly. Using texture cycling by itself can also give global motion, but when it is used in conjunction with the above technique better results are achieved. By careful selection of the textures, we can produce different motion, such as licking flames, rolling clouds, or bellowing smoke.

4 Creating Textures

Creating textures for computer graphics imagery is a difficult process that takes talent, practice and patience. We developed a system that allowed quick texture creation from existing images, to aid the animator.

For the images in figures 1b, 2, 3, 4 and 5 we used textures generated from a version of Perlin's [5] turbulence function. These textures, in general, provide adequate results for many of the effects we are after. We have experimented with several dozen different textures and while not all were suitable for fire, many provided other interesting effects which may be useful for rendering water, clouds or other phenomena.

We also examined the use of textures selected from stock photography. Here, the user can specify a set of positions within a much larger image. For each position, a texture is extracted from the image and added to the set of textures. This allows us to easily experiment with real images of fire. Figure 6 shows an image of an actual bonfire from which a set of textures is selected, shown in figure 7. The set of textures is then used to render the fire in figure 8.

The texture locations can also be selected automatically. Spectral analysis can be used to select only those portions of the image with certain frequency components, or to maximize a variation in frequencies across the set of textures. Generating textures

along certain curves within the image is useful for local dynamics. Checking existing images for interesting motion can be done quickly by randomly selecting textures from the images. Random selection was used in figures 8, 9, 10, and 11 with surprisingly good results for extremely simple and regular textures

Sampling along a specified path through the image can create desired local and global dynamics. For instance, selecting textures along a linear path, with the selected textures overlapping, will create linear motion. A piecewise linear path will create linear motion in multiple directions, possibly causing intricate interference patterns.

5 Applications / Results

5.1 Fire

As previously mentioned, figures 1 and 8 show images of fire created with our method. Figure 3 shows several stills from an animation in which a cow walks through fire unscathed. Since our method produces a three dimensional volume effect, objects can easily interact with that effect. So far we have only concentrated on the local and global dynamics. By adding a method that physically models fire [10, 11, 12], other fire effects, such as cross wind motion, fuel simulation, conversion to smoke, can be created.

5.2 Smoke / Steam

Figure 9 shows a still from an animation of a steam engine moving along a track. The smoke is fully three-dimensional. It consists of a voxelized cone and cylinder, requiring less than 400 non-empty voxels stored as a linear list and passed to our renderer. Since the motion of the steam results from the motion of the train, this effect looks very natural. Adding object dynamics will allow for the smoke/steam to interact with the environment. One such interaction would be changing shape when going under a bridge.

Figure 5 shows steam coming from the spout of a teapot. A simple 2x2x20 regular grid was used to model the steam with sixteen turbulence textures. Inter-voxel advection is used to aid in the upward motion.

5.3 Other effects

By using textures from photos of people, we can produce more surreal images. Figure 10 shows a laser-scanned head clouded with images of those close to him. For figure 10, we cropped faces from photographs and used the faces as the set of textures for our algorithm. When animated, these faces appear to flash across the head and dissipate above it. Adding a few faces to the turbulence textures used for fire, gives the appearance of ghosts in the fire.

In figure 3, we have a herd of cattle kicking up a cloud of dust. This technique has also been used for effects such as snow, rain, clouds and water caustics. We

believe it can be extended to phenomena such as bloody tissue, running water, and blowing grass.

5.4 Performance

By using coarse voxel grids and textured splats we achieve real-time speed from our method. Table I shows some performance statistics for various machines. Note that our implementation was interrupt based causing it to only display at 60Hz on the Onyx. Our method requires fast 2D texture mapping capabilities and the ability to change the associated texture for each splat quickly. We have found that a 32x32 texture gives very good results and only requires 16K of texture memory for each texture used. We generally use eight textures which requires 128K of texture memory to achieve real-time results. For machines with more memory, more complex dynamics can be achieved by using more and larger textures. Our method is pixel fill limited, that is, the speed is determined by the actual number of pixels that are displayed.

Table I. Performance measurements of method on various machines

Window size	**Voxels**	**02** 200Mhz R5000	**Indy** 133MHz R4800	**Octane/MXI** 195MHz R10000	**Onyx2/IR** 194MHz R10000
320x240	1560	12Hz	.39Hz	36Hz	60Hz
640x480	1560	12Hz	.063Hz	24Hz	60Hz
1280x960	1560	5Hz	.02Hz	8Hz	30Hz
320x240	780	20Hz	.8Hz	36Hz	60Hz
320x240	390	30Hz	1.6Hz	40Hz	60Hz
320x240	195	60Hz	3Hz	72Hz	60Hz

6 Conclusions

We have presented a novel extension to volume rendering of density clouds for animated gaseous or amorphous materials. The technique gives visually realistic animation, yet is efficient enough to be embedded into the next generation of video games or real-time virtual environments. Our technique has a wide variety of uses including, but not limited to: games, medical simulations, flight simulators, industrial simulations, scientific visualization, and as a modeling aid for more realistic effects. Our method is capable of representing various gaseous phenomena at real-time rates by using hardware texture support and low resolution volumes.

Several extensions to our method are possible. One of the more promising extensions is the ability to specify color in the detail along with the opacity. One such use would be to create moving highlights such as caustics. Our biggest thrust has been on dynamics and amorphous internal volume shape. We wish to explore methods to control the global shape using voxel opacities and dynamic voxel locations.

We do not consider illumination issues, such as for fire, in which the volume is a light source. Also, a spot light directed at the volume should illuminate part of the

volume. A possible solution to the first problem is to apply the textures to the objects that are nearby in the same manner as the fire is generated, but as an additive color texture.

We need to combine our method with one of the physically-based methods on a low resolution grid to create better global and object dynamics, while using our method to create the local dynamics. With the ability to use splat locations, instead of a rectangular grid, particle systems, could also be investigated as a control for the object dynamics.

Visit our web site, ***http://www.cis.ohio-state.edu/graphics/research/fire/***, for the latest information, images, and a demo version.

7 Acknowledgments

We would like to thank the Ohio Supercomputer Center for the use of an SGI Onyx2 for timing calculations. We would like to thank the reviewers for their insightful comments. We also thank Naeem Shareef, Klaus Mueller and Matthew Lewis for their help in writing this paper. This work was partially supported by an Ohio State University interdisciplinary seed grant and an NSF career award.

References

1 Blinn JF, Light reflection functions for simulation of clouds and dusty surfaces. Computer Graphics (SIGGRAPH ‘82 Proceedings), 16(3), pp 21 - 29, 1982.

2 Reeves WT, Particle systems. A technique for modeling a class of fuzzy objects. Transactions on Graphics, 2(2), pp 91-108, April 1983.

3 Reeves WT, Blau R, Approximate and probabilistic algorithms for shading and rendering structured particle systems. Computer Graphics (SIGGRAPH ’85 Proceedings), 19(3), pp 313-322, July 1985.

4 Kajiya JT, Ray tracing volume densities. Computer Graphics (SIGGRAPH ‘84 Proceedings), 18(3), pp 165-174, 1984.

5 Perlin K, An image synthesizer. Computer Graphics (SIGGRAPH ’85 Proceedings), 19(3), pp 287-296, July 1985.

6 Perlin K, Hoffert E, Hypertexture. Computer Graphics (SIGGRAPH ‘89 Proceedings), vol 27, pp 57-64, aug, 1989.

7 Gardner G, Visual simulation of clouds. Computer Graphics (SIGGRAPH ‘85 Proceedings), 19(3), 297-303, 1985.

8 Ebert DS and Parent RE, Rendering and animation of gaseous phenomena by combining fast volume and scanline a-buffer techniques. Computer Graphics (SIGGRAPH ‘90 Proceedings), 24(4), pp. 357-366, August 1990.

9 Stam J, Fiume E, Depicting fire and other gaseous phenomena using diffusion processes. Computer Graphics (SIGGRAPH ’95 Proceedings), pp 129-136, August 1995.

10 Sakas G, Modeling and animating turbulent gaseous phenomena using spectral synthesis. The Visual Computer, vol 9, pp 200-212, 1993.

11 Chiba N, Ohkawa S, Muraoka K, Miura M, Two-dimensional visual simulation

of flames, smoke and the spread of fire. The Journal of Visualization and Computer Animation, 5, (1), pp 37-54 , 1994
12 Foster N, Metaxas D, Modeling the motion of a hot, turbulent gas. SIGGRAPH 97, pp 181-188, Aug 1997
13 Levoy M, Display of surfaces from volume data. IEEE Computer Graphics and Applications, 8(5), pp 29-37, May 1988.
14 Drebin R, Carpenter L, Hanrahan P, Volume rendering. Computer Graphics (SIGGRAPH '88 Proceedings), vol 22, pp 65-74, August 1988.
15 Lacroute P, Levoy M, Fast volume rendering using a shear-warp factorization of the viewing transformation. Computer Graphics Proceedings, Proccedings of SIGGRAPH '94, pp 451-458, July 1994.
16 Malzbender T, Fourier volume rendering, ACM Trans. Graph. 12(3), pp 233-250, July 1993., Pages 233 - 250]
17 Shirley P, Tuchman A, A polygonal approximation to direct scalar volume rendering. Computer Graphics (SIGGRAPH '90 Proceedings) 24(5), pp 63-70, 1990.
18 Max N, Hanrahan P, Crawfis R, Area and volume coherence for efficient visualization of 3D scalar functions. Computer Graphics (San Diego Workshop on Volume Visualization), 24(5), pp 27-33, Nov 1990.
19 Westover L, Interactive volume rendering. Proceedings of the Chapel Hill Workshop on Volume Visualization, Chapel Hill, NC, May, 1989
20 Laur D, Hanrahan P, Hierarchical splatting: a progressive refinement algorithm. PH '91, 25(4): pp 285-288, July 1991.
21 Crawfis R and Max N, Texture splats for 3d vector and scalar field visualization. in Proceedings Visualization '93, San Jose, CA: IEEE Computer Society Press, Oct, 1993.

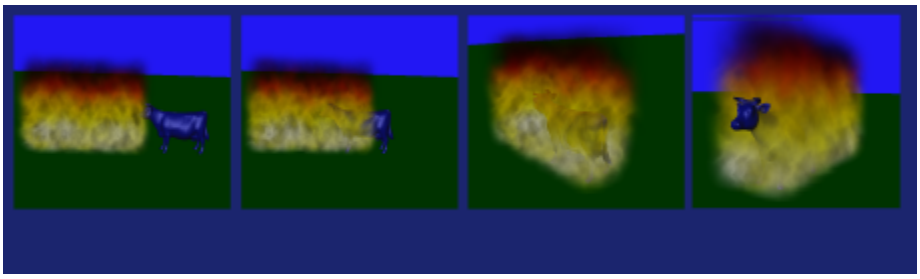


Figure 3. Several stills from an animation of a cow moving through fire.

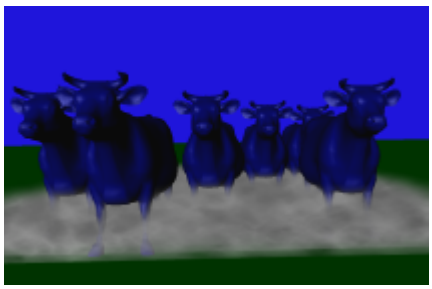


Figure 4. A herd kicking up a cloud of dust

Figure 5. A steaming Utah teapot.

Figure 6. A bonfire cut from a photograph of real fire.

Figure 7. Resulting set of textures generated by randomly sampling figure 6.

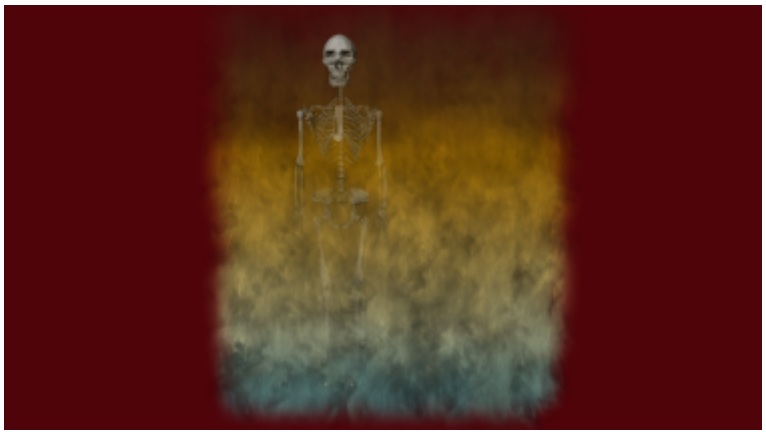


Figure 8. The resulting fire image using the textures from figure 7.

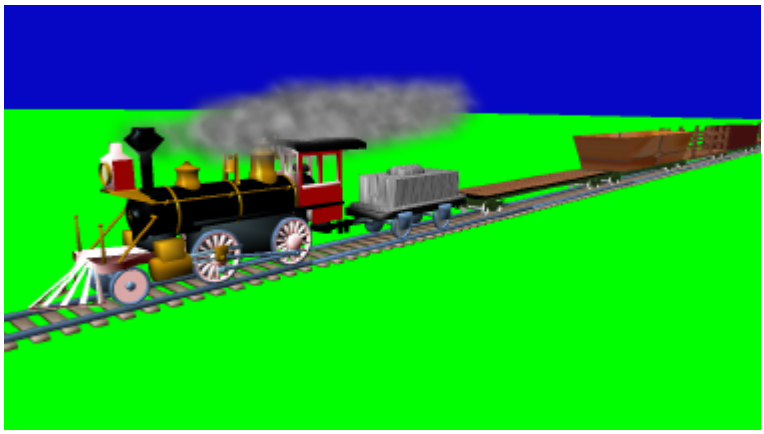


Figure 9. Image from real-time animation of smoke emanating from a locomotive.

Figure 10. Artistic application. A low-resolution volume textured with human faces, gives the effects of ghosts.

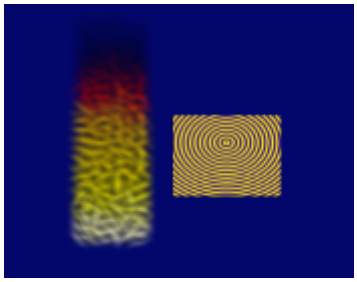


Figure 11. The volume effect on the left was created by randomly sampling the texture on the right.