5.3

Volumetric Clouds and Mega-Particles

Homam Bahnassi, **InIFramez**
Wessam Bahnassi, **Electronic Arts**

Introduction

Volumetric effects are an important part of 3D game scenes. The two most common approaches for implementing these effects are screen-aligned particles and sliced volumetric rendering. Other approaches exist but are mostly variants of these two.

This article lists drawbacks of existing techniques and then introduces a novel, yet simple, technique that overcomes many of these drawbacks (see Figure 5.3.1). This novel technique targets volumetric effects and phenomena that range from clouds to thick smoke, dust, fog, and even space nebulae.

FIGURE 5.3.1 A volumetric cloud rendered using the described technique, featuring correct lighting and depth interaction.

The Problems

Many attempts have been made to make volumetric effects real-time in computer graphics. There always is, however, a catch. Either the output is a crude approximation or it lacks flexibility or other important features. The following categorization summarizes common drawbacks of existing implementations.

Rendering Artifacts

Missing lighting: Most particle sprite approaches are unlit. This lack of lighting is usually due to missing information (e.g., no normals) or due to calculations being too expensive for real-time execution [Laeuchli05].

No proper depth interaction: The implementation is 2D in nature. Existing objects in the scene are not taken into account. Techniques based on fractal generation [Pallister01] typically suffer from this limitation.

Sharp cutouts: Cutouts are common with most sprite-based techniques. The problem is obvious whenever particle sprites intersect the ground or other objects.

User Inconvenience

Camera placement limitations: Several slice-based approaches require that the camera never enters the volume of a cloud. Alternatively, these approaches increase rendering complexity to handle this case [Baker01].

Unintuitive controls and inflexibility: Fractal-based approaches typically suffer from this problem.

Performance

High fill-rate requirements: Particle-based approaches typically consume large amounts of fill rate. High overdraw and the cost of alpha-blending combine to cripple frame rate.

Expensive dynamics calculations: This drawback is specific to particle-based approaches. Simulating a large number of particles consumes a lot of CPU processing power.

Unfriendly to GPU optimizations: Some volumetric techniques alter pixel depth in the pixel shader, which forces graphics hardware to abandon early-z optimizations, resulting in suboptimal rendering speed.

Hardware Requirements

Requirement of advanced GPU capabilities: Navier-Stokes fluid simulations [Stam00] and GPU-generated fractals [Tarantilis04] fall into this category because they require high GPU programmability to execute their algorithms.

High demand of hardware resources: Techniques incorporating volumetric textures typically suffer from this drawback. For example, ray-tracing into volume textures [Gottlieb04] requires large texture dimensions to render a cloud with enough detail. Large volumetric textures tend to suffer from cache misses, thus reducing performance.

The Approach

Most existing particle-based techniques build on the concept of "more is better" to produce good results. The approach presented here is also particle based, yet it depends on rendering a small number of large particles instead. These so-called mega-particles are not 2D billboards, but rather 3D spheres. The spheres render close to each other in different sizes and layouts to give a cloud a basic structure. Later, these grouped spheres turn into a puffy, realistic-looking cloud through a series of operations, outlined below and shown in Figure 5.3.2.

Generating cloud color and depth information: Mega-particles render to a separate render-target texture, while using the scene's depth buffer. The particles render as lit 3D spheres, just like any other object in the scene. Z-test and z-write are enabled for proper interaction with the scene's depth buffer.

Blurring and fractal distortion: Next, the render-target texture is blurred using your favorite blurring method (e.g., separable Gaussian and down-sampling [Mitchell04]), followed by rendering a 3D box covering all the mega-particles. A fractal texture textures this 3D box to add detail to the result.

Final blending: Finally, the blurred texture is blended with the original back buffer. The blending is done by rendering a single, full-screen quad over the existing color information of the back buffer.

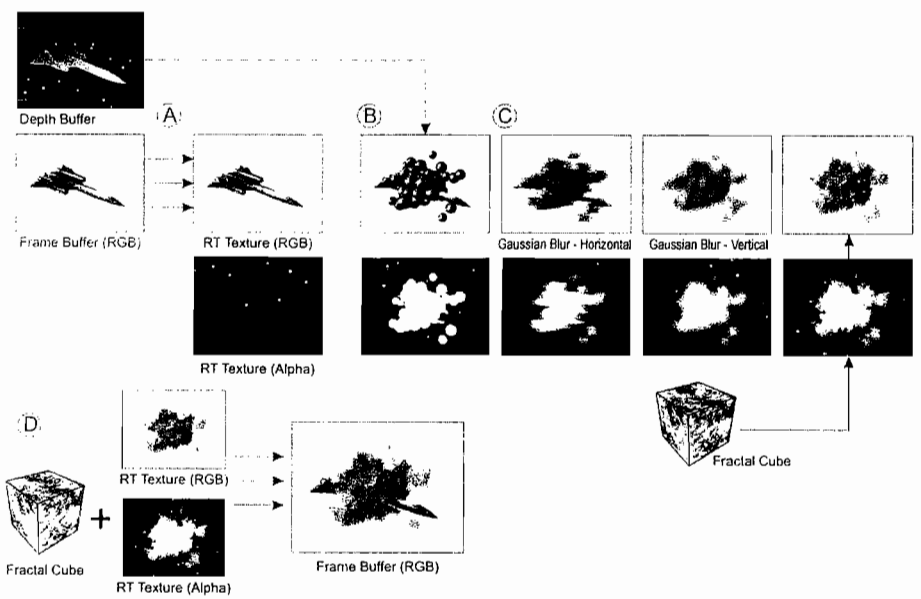

FIGURE 5.3.2 The pipeline for rendering volumetric clouds using mega-particles.

Ingredients and Workflow

This technique requires a render-target texture that is of the same dimensions as the back buffer. The format of this texture depends on the planned effect usage: colored clouds require three channels of color information, while grayscale clouds only need a single channel. Coloring here means variance of chromacity (e.g., differently colored lights or mega-particles). Simple color tinting, however, is still achievable with a single color channel for intensity. The format must also contain an additional alpha channel.

Another requirement is the mega-particles' geometry. These 3D spheres shape the cloud and therefore should be created and sized under artistic control. Animation may be procedural or under artists' control, and may cover all aspects of rendering of these spheres (e.g., position, size, and color).

The final requirement is a 3D box that encloses all mega-particles of a single cloud. This box is textured with a fractal map that covers all faces seamlessly. This texture may be a pregenerated fractal of moderate size (e.g., 256×256 texels for each face). A single-channel alpha format, such as A8, suffices for this texture.

The time for cloud rendering should be chosen carefully. The ideal rendering order is to first render all objects that are behind the cloud (from the camera's viewpoint) or that could potentially penetrate it. Next, the cloud's rendering process is carried out. Finally, the remaining objects in front of the cloud are rendered.

The following sections provide a detailed description of the technique's rendering steps.

Generation of Cloud Color and Depth Information

The operation starts by filling the render-target texture with the current color contents of the back buffer and an alpha value of 0. Next, this texture is set as the current render target, while ensuring that the existing depth buffer is still intact and active.

Now the cloud renders all of its mega-particle spheres with full lighting just like any other scene object. The cloud spheres thus test against existing depth values and write to the depth buffer. During rendering, sphere pixels that pass the depth test also write the value 1.0 into the alpha channel of the render-target texture (i.e., alpha-blending and testing is disabled).

Blurring and Fractal Distortion

At this point, the render-target texture contains the existing scene and the newly rendered mega-particles. The next step is to blur this texture, that is, not only the color channels, but the alpha channel as well. While any blurring technique works, a separable Gaussian blur with down-sampling [Mitchell04] is used in the sample application, which quickly produces a good amount of blurring. Note that the sampling kernel should be big enough to remove any sharp edges that interpenetrating mega-particles might create, yet not so big as to smudge lighting details. The sample application on the companion CD-ROM exposes the parameter `g_BlurAmount` to control this kernel size per cloud.

The next step renders the fractal-textured 3D box so that it bounds all cloud spheres (i.e., by applying the required transformations). The purpose of rendering this box is to add detail to the render-target texture's alpha channel. Two techniques achieve this goal: using alpha-modulation or using displacement.

With alpha-modulation, the box is rendered using the scene's camera with z-test and z-writing disabled. Alpha blending is set to a multiplicative blending mode (e.g., ColorOut = ColorSrc × ColorDest + ColorDest × 0), and the color-write mask is set to only write the alpha channel. Backface culling is enabled if the camera is outside of this box or disabled if the camera is within the box.

When using displacement, the fractal box is used as a source to displace pixels in the alpha channel, which gives better results if tuned correctly. [Hargreaves04] describes a similar technique in its "Displacement Effects" section. This technique requires a two-channel format for the fractal texture.

The fractal-box rendering operation guarantees that as the camera orbits around a cloud, it sees true 3D details that identify each part of the cloud.

Final Blending

At this point we have a texture containing blurred and fractal distorted spheres. The final step is to combine this texture and the back buffer. Rendering a full-screen quad to the back buffer that samples the texture at a 1-to-1 texel-to-pixel ratio achieves this combination. During this blit, alpha blending is on, and the blending mode is:

$$\text{Color}_{\text{Out}} = \text{Color}_{\text{Src}} \times \text{Alpha}_{\text{Src}} + \text{Color}_{\text{Dest}} \times (1\text{-Alpha}_{\text{Src}})$$

Z-test and z-write are disabled (see Figure 5.3.3).

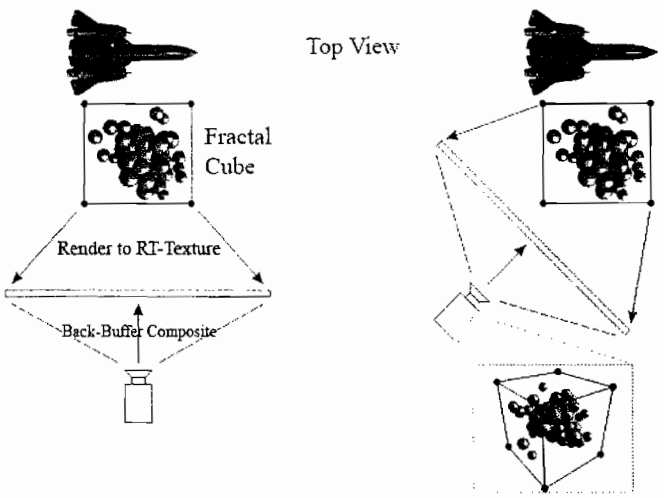

FIGURE 5.3.3 Data setup for the final blit.

Discussion of Results

The described technique has many advantages over existing approaches. The following list shows the differences using the categories mentioned in the preceding section "The Problems."

Rendering

Depth interaction: Rendering clouds using this technique interacts correctly and accurately with depth values of existing objects in the scene.

Colored clouds: Each sphere can be colored independently to give variation over the whole cloud (e.g., dark and bright areas or lightning, causing flashing areas).

Fully lit: The 3D spheres are lit using your favorite lighting calculations; these lighting calculations directly influence the cloud's final coloring. For more complex lighting effects, shadowing and ambient occlusion can be included.

3D and volumetric: The cloud is viewable from any camera angle. Cameras are free to move inside a cloud as long as the fractal texture-mapped box correctly overlays the blurred render-target texture.

User Convenience:

Flexibility and control over shape and animation: The generic layout of the spheres shapes a cloud. Because the number of spheres is small, shaping a cloud is manageable. Spheres can gradually spread and scale over time to fade out a cloud. Moreover, animating the fractal details gives an additional sense of motion (e.g., rotating the fractal box).

Does not require a specific scene setup: The technique fits into existing rendering mechanisms.

Performance:

Low fill-rate consumption: The amount of alpha-blended overdraw is limited to at most two full-screen blits even when clouds fill the whole screen.

High potential for optimization: Sorting the spheres in front-to-back order saves overdraw. Blurring can be limited to the cloud-covered portion instead of the whole screen.

GPU friendly: The technique takes advantage of optimizations offered by today's graphics cards, for example, early z-rejection.

GPU Requirements:

Does not require advanced shader models: The technique is achievable on DirectX 7 hardware or GameCube, Playstation 2, or XBox consoles as long as a simpler blurring method is used.

A side-effect of blurring and fractal distortion is that clouds fuzzily interact with objects interpenetrating them. Instead of a sharp cutout, a cloud smoothly climbs an object's edges, resulting in a realistic look (see Figure 5.3.4).

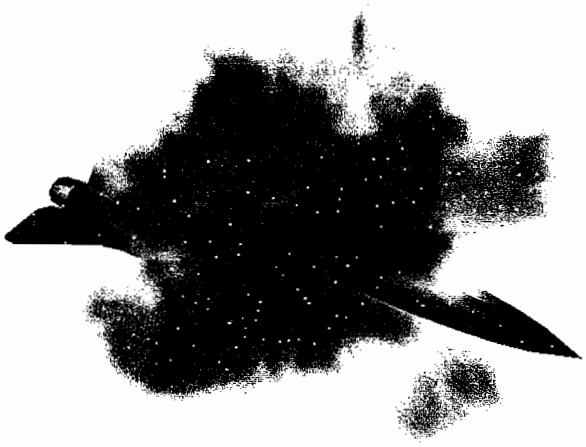

FIGURE 5.3.4 Clouds climb an interpenetrating object's edges in a fuzzy way.

For more interesting results, a second fractal box can be added. The purpose of this second box is to rotate slowly over time and blend with the first box's values, resulting in apparent internal structure changes of the cloud.

For this technique to work, clouds must render in a certain order within the scene. This order is necessary because the fuzzy-edge-climbing effect should not happen to objects totally in front of a cloud. Since these fuzzy edges are generated by the blur spreading color values outside of the mega-particles, they are not backed up by z-values. Thus, they always blend with what is underneath them, even if it is closer to the viewer. Failure to render in the appropriate order eliminates the fuzzy edges feature, which may still be acceptable if changing the rendering order is not an option.

Clouds rendered with this implementation are fully opaque except at the edges. Constant translucency is achievable during the final back buffer blend, but varying translucency should be handled by changing translucency of each mega-particle.

Conclusion

This article presents a simple technique to render volumetric clouds cheaply. The core concept is to render fully lit mega-particles that are blurred, fractal-distorted, and then alpha-blended with the frame buffer. The technique eliminates many drawbacks

of previous volumetric rendering approaches and offers artistic control and flexibility. Finally, performance overhead of this technique is low; in particular, its fill-rate consumption is small and almost constant.

Acknowledgments

Special thanks to Sergei Savchenko of Electronic Arts and Abdo Haji-Ali of In|Framez for their guidance and their reviews of this article.

References

[Baker01] Baker, Dan and Charles Boyd. "Volumetric Rendering in Real-time." Available online at *http://www.gamasutra.com/features/20011003/boyd_01.htm*, 2001.

[Gottlieb04] Gottlieb, Eli Z. "Rendering Volumes in a Vertex & Pixel Program by Ray Tracing." *ShaderX²: Shader Programming Tips & Tricks with DirectX 9*, edited by Wolfgang Engel. Wordware Publishing, 2004: pp. 177–184.

[Hargreaves04] Hargreaves, Shawn. "Non-Photorealistic Post-processing Filters in MotoGP 2." *ShaderX²: Shader Programming Tips & Tricks with DirectX 9*, edited by Wolfgang Engel. Wordware Publishing, 2004: pp. 469–480.

[Laeuchli05] Laeuchli, Jesse. "Volumetric Clouds." *ShaderX³: Advanced Rendering with DirectX and OpenGL*, edited by Wolfgang Engel. Charles River Media, 2005: pp. 611–616.

[Mitchell04] Mitchell, Jason L., Marwan Y. Ansari, and Evan Hart. "Advanced Image Processing with DirectX9 Pixel Shaders." *ShaderX²: Shader Programming Tips & Tricks with DirectX9*, edited by Wolfgang Engel. Wordware Publishing, 2004: pp. 450–452.

[Pallister01] Pallister, Kim. "Generating Procedural Clouds Using 3D Hardware." *Game Programming Gems 2*, edited by Mark A. DeLoura. Charles River Media, 2001: pp. 463–473.

[Stam00] Stam, Jos. "Interacting with Smoke and Fire in Real-time." *Communications of the ACM*, *43*(7), (2000): pp. 76–83.

[Tarantilis04] Tarantalis, Georgios E. "Simulating Clouds with Procedural Texturing Techniques Using the GPU." Naval Postgraduate School Master Thesis, September 2004.