

Oz: The Great and Volumetric

Magnus Wrenninge
Sony Pictures Imageworks

Chris Kulla
Sony Pictures Imageworks

Viktor Lundqvist
Sony Pictures Imageworks



Figure 1: Renders from *Oz: The Great and Powerful*. The left image shows a water simulation with volumetric spray and foam elements. The right image shows the effect of the multiple scattering approximation. ©2013 Disney Enterprises Inc.

For *Oz: The Great and Powerful*, Imageworks provided over 1100 shots, 400 of which included substantial volumetric elements. We discuss the challenges of various sequences and optimizations to our renderer performed during the course of production.

Path traced volumes in production Starting on *The Amazing Spiderman* and *Men In Black 3*, Imageworks’ in-house version of Arnold has supported path tracing of volumes alongside surfaces. The pipeline is built around the Field3D file format, and volume primitives support all standard light sources, including area lights and textured skydomes. We maintain reasonable render times with the importance sampling techniques detailed in [Kulla and Fajardo 2012].

To lighters, volumes were seen as just another scene graph location in Katana, with a geometric proxy representation for fast previewing and layout. Named attributes of each field automatically bind to similarly named shader parameters, so that TDs can write custom shaders easily. Under the hood, the Field3D primitive supported arbitrary numbers of fields in each .f3d file, and had full support for dense, sparse and MAC fields of mixed bit depths.

Multiple scattering approximation A large portion of the volumetric work on *Oz* consisted of fog, mist, steam and clouds, all of which are high albedo media. Inspired by the work of Bouthors [2011] on cloud rendering, where multiple shadow maps were used to approximate different orders of multiple scattering, we developed an approximation for path tracing that incurs little render time overhead, adds little extra noise, and can be importance sampled. The main idea is to artificially lower the extinction coefficient σ_t along the shadow ray to let more light reach the shaded point. But rather than use a fixed scaling factor, we use a summation over several scales. We also adjust the local phase function eccentricity g and local scattering coefficient σ_s such that the total contribution of light at a given point is:

$$L = \sum_{i=0}^{N-1} L_i \quad (1)$$

$$L_i = \sigma_s b^i L_{\text{light}}(\omega_i) p(\omega_i, \omega_o, c^i g) e^{-a^i \int_0^t \sigma_t(s) ds} \quad (2)$$

where N (“octaves”), a (“attenuation”), b (“contribution”), and c (“eccentricity attenuation”) are user-tunable parameters. Defaults of $N = 8$ and $a = b = c = 1/2$ are a good starting point for all the cases we tried. Since the change in eccentricity in the phase function makes the choice of sampling direction more complex, we stochastically choose a single octave with probability of b^i . This allows keeping the ray budget constant and renders

with little extra noise. These parameters gave artists simple control over the multiple scattering effect, without having to tweak scattering lengths manually. This approximation can be used on camera-shadow rays when only simulating direct lighting, or on camera-indirect-shadow rays if already simulating one bounce. The latter option is more expensive but more accurately captures the diffusion effect.

Shading system Volumes were also integrated into Arnold’s OSL-based shading system, which allowed TDs to build custom shaders for volumes in Katana, just as they would for surface shaders. In order to provide a standardized interface to the lighting pipeline, a “root” illumination shader was provided to the TDs, into which arbitrary shader nodes were hooked. Several sequences relied heavily on custom shaders, for effects such as fire, explosions, and ash clouds, and also for procedural noise banks used to quickly layout generic fog.

Efficient traversal of frustum buffers For the case of sparse uniform volumes, we limit ray marching to occupied blocks by using Bresenham traversal. However, for frustum buffers there is no guarantee that an arbitrary ray remains straight in voxel space, as the perspective division may bend a ray into a curved path. Rather than attempt some form of curved ray marching, we observe that the projection mapping does not bend axis-aligned planes and build a kd-tree over the sparse blocks in voxel space. During ray intersections tests, the planes are transformed into world space which avoids testing against curved rays. To support motion blurred mappings and reduce the size of the tree we perform the plane transform on the fly during traversal. Since the planes are axis-aligned, this is a very simple operation on the perspective matrix entries. We pay special attention to numerical robustness to ensure the continuity of the detected occupied intervals. This sped up the integration of high resolution buffers greatly, saving the overhead of shader calls in regions that are known to be empty. As our method makes no assumption about ray direction, it applies equally well to shadow rays and other non-view-aligned rays.

References

- KULLA, C., AND FAJARDO, M. 2012. Importance sampling techniques for path tracing in participating media. *Computer Graphics Forum* 31, 4, 1519–1528.
- WRENNINGE, M., BIN ZAFAR, N., HARDING, O., GRAHAM, G., TESSENDORF, J., GRANT, V., CLINTON, A., AND BOUTHORS, A. 2011. Production volume rendering 2: Systems. ACM SIGGRAPH 2011 Courses.