Appendix A: Refractoring Precomputed F Texture to Per-Frame F' Texture

Scattering inside participating media is generally computed using the standard airlight equation [Nishita et al. 1987], described in Equation 1 from the paper:

$$L_{sctr} = \int_0^{d_s} \kappa_s \rho(\alpha) \frac{I_0}{d^2} e^{-(\kappa_a+\kappa_s)(d+x)} dx.$$

Sun et al. [2005] introduced an interactive single scattering model for participating media based upon a preintegration of a reorganized version this equation:

$$L_{sctr}(\gamma, d_s, d_l, \kappa_s) = \frac{\kappa_s I_0 e^{-\kappa_s d_l \cos\gamma}}{2\pi d_l \sin\gamma} \int_{\gamma/2}^{\frac{\pi}{4}+\frac{1}{2}\arctan\frac{\kappa_s(d_s - d_l\cos\gamma)}{\kappa_s d_l \sin\gamma}} e^{-\kappa_s d_l \sin\gamma \tan\xi} d\xi.$$

Where (as shown in Figure 2 in the paper) I_0 is the light intensity, γ is the angle between the viewing and light rays, d_l is the distance from eye to light, d_s is the distance from eye to the visible surface point, and κ_s is the participating media's homogeneous scattering coefficient.

To ease readability and more readily simplify the equation, they introduced two auxiliary expressions $A_0(d_l, \gamma, \kappa_s)$ and $A_1(d_l, \gamma)$. We use a slightly modified version of these auxilary expression (to remove dependance on I_0):

$$A_0(d_l, \gamma, \kappa_s) = \frac{\kappa_s e^{-\kappa_s d_l \cos\gamma}}{2\pi d_l \sin\gamma},$$

and

$$A_1(d_l, \gamma, \kappa_s) = \kappa_s d_l \sin\gamma.$$

Using these new expressions, the integral becomes:

$$L_{sctr}(\gamma, d_s, d_l, \kappa_s) = I_0 A_0(d_l, \gamma, \kappa_s) \int_{\gamma/2}^{\frac{\pi}{4}+\frac{1}{2}\arctan\frac{\kappa_s(d_s - d_l\cos\gamma)}{\kappa_s d_l \sin\gamma}} e^{-A_1(d_l,\gamma,\kappa_s)\tan\xi} d\xi.$$

In our paper, we introduce a third auxilary expression $A_2(d_l, \gamma, d_s)$ to help further simplify this equation:

$$A_2(d_l, \gamma, d_s) = \frac{\pi}{4} + \frac{1}{2}\arctan\frac{d_s - d_l\cos\gamma}{d_l \sin\gamma},$$

which, after dropping the parameters for the A_i terms results in:

$$L_{sctr} = I_0 A_0 \int_{\gamma/2}^{A_2} e^{-A_1 \tan\xi} d\xi.$$

The key to Sun et al.'s [2005] work is the observation that this scattering integral can be represented as a difference of factors of the form:

$$F(u,v) = \int_0^v e^{-u \tan\xi} d\xi,$$

which is an integral that can be numerically precomputed and stored in a 2D texture. While there is no analytical solution, $F(u,v)$ is a smooth function and behaves well in the range of values needed for volumetric scattering.

Using this idea, the scattering function can then be defined as (Equation 2 in the paper):

$$L_{sctr} = I_0 A_0 \left[F(A_1, A_2) - F(A_1, \frac{\gamma}{2}) \right].$$

This works exceedingly fast when rendering participating media without shadows, where the only lookups into $F(u,v)$ correspond to the eye $(F(A_1, \gamma/2))$ and the visible geometry $(F(A_1, A_2))$. Unfortunately, when stepping along this ray to sample the illumination at various points in the volume, this equation becomes:

$$L_{sctr} = A_0 \sum_{i=1}^{N} I_0(i) \left[F(A_1, A_2(i)) - F(A_1, A_2(i-1)) \right],$$

where A_2 must be recalculated at every step. In Sun et al.'s [2005] work the expense of recomputing A_2 was relatively unimportant, since it was computed once per pixel. When stepping along a ray, it must be recomputed numerous times. Since each step's computation requires an arctan (an operation currently implemented using many GPU instructions), simply computing A_2 can become quite costly when sampling 30 or more times. Furthermore, this becomes numerically unstable for small step sizes.

We observe that when stepping along each pixel's ray, we traverse a single column in the precomputed F texture. A_1 is only a function of d_l (which is fixed for all pixels in a given frame), κ_s (which is constant in homogeneous media), and γ (which is constant inside each pixel). By computing the new texture $F'(\cos\gamma, x)$ suggested in Section 3.4 once per frame, we reduce the pixel shader computations to a minimal set: the current distance x from the eye, and the dot product $\cos\gamma$ between the ray and the direction from the eye to the light.

Because F and F' vary extremely smoothly, F' can be computed at a low resolution (we use 256^2, which is cheap enough that we did not experiment to find if lower resolution textures remain usable). Furthermore, since the texture is continuously recreated only useful ranges of angles and distances for each frame need be stored, and the texture is indexed by easily computed and understandable values instead of complex mathematical formulae.

We use:

$$F'(\cos\gamma, x) = \frac{\kappa_s e^{-\kappa_s d_l \cos\gamma}}{2\pi d_l \sin\gamma} F\left((\kappa_s d_l \sin\gamma)/u_{max}, \left(\frac{\pi}{4} + \frac{1}{2}\arctan\frac{x - d_l\cos\gamma}{d_l \sin\gamma}\right)/v_{max}\right),$$

where u_{max} and v_{max} describe the ranges of u and v in the original F texture (i.e., $0 \le u \le u_{max}$ and $0 \le v \le v_{max}$) and all the other values can either be computed from $\cos\gamma$ or are constant over the frame (i.e., κ_s and d_l).

Rewriting this to be slightly cleaner:

$$F'(\cos\gamma, x) = A_0(d_l, \gamma, \kappa_s) F\left(\frac{A_1(d_l, \gamma, \kappa_s)}{u_{max}}, \frac{A_2(d_l, \gamma, x)}{v_{max}}\right),$$

A GLSL shader implementing this computation is included, though due to our implementation's internal storage format for F we use:

$$F'(\cos\gamma, x) = A_0(d_l, \gamma, \kappa_s) F\left(1 - \frac{A_1(d_l, \gamma, \kappa_s)}{u_{max}}, 1 - \frac{A_2(d_l, \gamma, x)}{v_{max}}\right),$$