```
/
**************************************************************************
****
 * smoke.sl
 *
 * Description:
 *    This is a volume shader for smoke.  Trapezoidal integration is
 *    used to find scattering and extinction.
 *
 * Parameters:
 *   opacdensity - overall smoke density control as it affects its
ability
 *          to block light from behind it.
 *   lightdensity - smoke density control as it affects light
scattering
 *          toward the viewer.
 *   integstart, integend - bounds along the viewing ray direction of
the
 *          integration of atmospheric effects.
 *   stepsize - step size for integration
 *   smokefreq, smokeoctaves, smokevary - control the fBm of the noisy
smoke
 *          If either smokeoctaves or smokevary is 0, there is no
noise
 *          to the smoke.
 *   scatter - when non-1, can be used to give wavelength-dependent
 *          extinction coefficients.
 *
 * Author: Larry Gritz
 *
 * Reference:
 *   _Advanced RenderMan: Creating CGI for Motion Picture_,
 *   by Anthony A. Apodaca and Larry Gritz, Morgan Kaufmann, 1999.
 *
 * $Revision: 1.1 $     $Date: 2004/03/02 04:50:34 $
 *

**************************************************************************
****/


#include "noises.h"


/* For point P (we are passed both the current and shader space
 * coordinates), gather illumination from the light sources and
 * compute the smoke density at that point.  Only count lights tagged
 * with the "__foglight" parameter.
 */
void
```

```
smokedensity (point Pcur, Pshad;
              uniform float smokevary, smokefreq, smokeoctaves;
              float stepsize;
              output color Lscatter; output float smoke)
{
    Lscatter = 0;
    illuminance (Pcur) {
        extern color Cl;
        float foglight = 1;
        lightsource("__foglight",foglight);
        if (foglight > 0)
            Lscatter += Cl;
    }
    if (smokeoctaves > 0 && smokevary > 0) {
        point Psmoke = Pshad * smokefreq;
#pragma nolint
        smoke = snoise (Psmoke);
        /* Optimize: one octave only if not lit */
        if (comp(Lscatter,0)+comp(Lscatter,1)+comp(Lscatter,2) > 0.01)
            smoke += 0.5 * fBm (Psmoke*2, stepsize*2, smokeoctaves-1,
2, 0.5);
        smoke = smoothstep(-1,1,smokevary*smoke);
    } else {
        smoke = 0.5;
    }
}



/* Return a component-by-component exp() of a color */
color colorexp (color C)
{
    return color (exp(comp(C,0)), exp(comp(C,1)), exp(comp(C,2)));
}



volume
smoke (float opacdensity = 1, lightdensity = 1;
       float integstart = 0, integend = 100;
       float stepsize = 0.1, maxsteps = 100;
       color scatter = 1;   /* for sky, try (1, 2.25, 21) */
       float smokeoctaves = 0, smokefreq = 1, smokevary = 1;)
{
    point Worigin = P - I;    /* Origin of volume ray */
    point origin = transform ("shader", Worigin);
    float dtau, last_dtau;
    color li, last_li;

    /* Integrate forwards from the start point */
```

```
    float d = integstart + random()*stepsize;
    vector IN = normalize (vtransform ("shader", I));
    vector WIN = vtransform ("shader", "current", IN);

    /* Calculate a reasonable step size */
    float end = min (length (I), integend) - 0.0001;
    float ss = min (stepsize, end-d);
    /* Get the in-scattered light and the local smoke density for the
     * beginning of the ray
     */
    smokedensity (Worigin + d*WIN, origin + d*IN,
                   smokevary, smokefreq, smokeoctaves, ss, last_li,
last_dtau);

    color Cv = 0, Ov = 0;   /* color & opacity of volume that we
accumulate */
    while (d <= end) {
        /* Take a step and get the local scattered light and smoke
density */
        ss = clamp (ss, 0.005, end-d);
        d += ss;
        smokedensity (Worigin + d*WIN, origin + d*IN,
                       smokevary, smokefreq, smokeoctaves, ss, li,
dtau);

        /* Find the blocking and light scattering contribution of
         * the portion of the volume covered by this step.
         */
        float tau = opacdensity * ss/2 * (dtau + last_dtau);
        color lighttau = lightdensity * ss/2 * (li*dtau +
last_li*last_dtau);

        /* Composite with exponential extinction of background light
*/
        Cv += (1-Ov) * lighttau;
        Ov += (1-Ov) * (1 - colorexp (-tau*scatter));
        last_dtau = dtau;
        last_li = li;
    }

    /* Ci & Oi are the color and opacity of the background element.
     * Now Cv is the light contributed by the volumee itself, and Ov
is the
     * opacity of the volume, i.e. (1-Ov)*Ci is the light from the
background
     * which makes it through the volume.  So just composite!
     */
    Ci = Cv + (1-Ov)*Ci;
    Oi = Ov + (1-Ov)*Oi;
}
```