A Practical Analytic Single Scattering Model for Real Time Rendering

Published: 2005
By: Bo Sun, Ravi Ramamoothi, Srinivasa G. Narasimhan, Shree K. Nayar

Presentation by: Maria Lemón Sjölander
Comparing
Made for

- Single scattering
- Isotropic light source
- Homogeneous participating media
Related Work

- Multiple scattering and non-homogeneous media
About Scattering

- Multiple Scattering
- Single Scattering
About Single Scattering

(a)

Point Source

Direct Transmission

Viewer

Surface Point

(b)

Point Source

Single Scattered

Direct Transmission

Viewer

Surface Point

(c)

Point Source

Single Scattered

Direct Transmission

Viewer

Surface Point

(d)

Point Source

Single Scattered

Direct Transmission

Viewer

Surface Point
The Airlight Integral
and
The Surface Radiance Integral
Math - The Airlight Integral

\[ F(u, v) = \int_{0}^{v} \exp[-u \tan \xi] \, d\xi \]

Stored numerically in a 2D look-up table

\[ L_a = A_0(T_{sv}, \gamma, \beta) \left[ F(A_1(T_{sv}, \gamma), \frac{\pi}{2}) - F(A_1(T_{sv}, \gamma), \frac{\gamma}{2}) \right] \]

Computed in real time in the vertex or pixel shader
Math – The Surface Radiance Integral

\[ G_n(T_{sp}, \theta'_s) = \int_{\Omega_{2\pi}} \frac{e^{-T_{sp}} \cos \gamma'}{\sin \gamma'} \left[ F(A_1, \frac{\pi}{2}) - F(A_1, \frac{\gamma'}{2}) \right] \cos^n \theta_i d\omega_i \]

Stored numerically in a 2D look-up table

\[ L_p = I_0 k_d \left[ \frac{e^{-T_{sp}}}{D_{sp}^2} \cos \theta_s + \beta^2 \frac{G_0(T_{sp}, \theta_s)}{2\pi T_{sp}} \right] + I_0 k_s \left[ \frac{e^{-T_{sp}}}{D_{sp}^2} \cos^n \theta'_s + \beta^2 \frac{G_n(T_{sp}, \theta'_s)}{2\pi T_{sp}} \right] \]

Computed in real time in the vertex or pixel shader
Scattering coefficient of the medium

- Only value needed for the math functions
  - High $\beta$ = thicker fog
  - Low $\beta$ = thinner fog
  - Works best for low $\beta$
Implementation

- Easy to implement
- Can be added to almost any interactive application

- Works with:
  - Complex Lightning
  - Environment mapping
  - Precomputed radiance transfer
frag2app fmain
    float4 objPos : TEXCOORD3, // 2D texture coords
    ...
    uniform samplerRECT F, // 2D special functions
    uniform samplerRECT G0,
    uniform samplerRECT Gn)
{
    frag2app OUT; // output radiance
    // Set up and calculate Tqv, γ, Dqv, TvP, θz and θz′

    ******* Compute Lq from equation 11 *******
    A0 = (β * I0 * exp[-Tqv * cos(γ)])/(2π * Dqv * sin(γ)); // equation 7
    A1 = Tqv * sin(γ); // equation 8
    v = π/4 + (1/2) arctan([(TvP - Tqv * cos(γ))/(Tqv * sin(γ))]):
    f1 = texRECT(F, float2(A1, v)); // v is one of texture coords
    f2 = texRECT(F, float2(A1, γ / 2)); // 2D texture lookup
    airlight = A0 * (f1 - f2); // equation 11

    ******* Diffuse surface radiance from equation 17 *******
    d1 = k_d * exp[-TvP * cos(θz) * I0/(Dzp * Dzp)];
    d2 = (k_d * I0 + β * β)/(2π * Tzp) * texRECT(G0, float2(Tzp, θz));
    diffuse = d1 + d2;

    ******* Specular surface radiance from equation 18 *******
    s1 = k_s * exp[-TvP * cos(θz′) * I0/(Dzp * Dzp)];
    s2 = (k_s * I0 + β * β)/(2π * Tzp) * texRECT(Gn, float2(Tzp, θz′));
    specular = s1 + s2;

    ******* Final Color (equation 10) *******
    OUT.color = airlight + (diffuse + specular) * exp[-TvP];
    return OUT;
}