Thin Film Shader

Overview

This shader reproduces the effect of thin film interference on a surface. It is based on code found here:

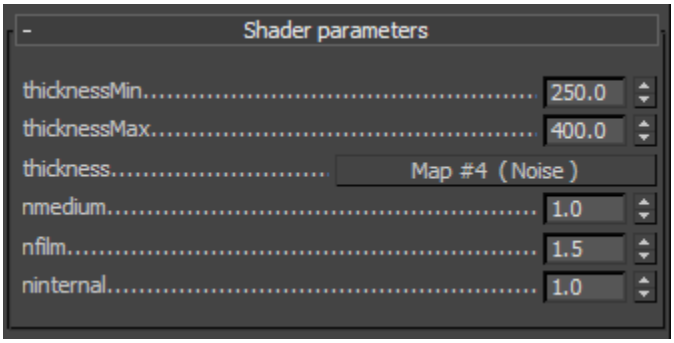
http://www.gamedev.net/page/resources/_/technical/graphics-programming-and-theory/thin-film-interference-for-computer-graphics-r2962

The shader can be used in the reflection color slot of a **VRayMtl** material through a **VRayOSLTex,** with the **Fresnel** option for the material turned off (the shader does its own Fresnel calculations).

The shader code and a sample scene are here: thinfilm.zip

The hdr image is not provided as part of the sample scene. Feel free to replace it for a similar lighting result.

Parameters



thicknessMin - the minimum thickness of the thin film, in nanometers.

thicknessMax - the maximum thickness of the thin film, in nanometers.

thickness - a texture map that specifies the actual thickness of the film between the specified min and max thickness values. Normally this would be something like a noise map to give some variation to the interference effect. If this is left blank, the **thicknessMin** parameter specifies the film thickness.

nmedium - refractive index of the medium surrounding the material. Normally this is set to 1.0 for air.

nfilm - refractive index of the medium that the thin film is made of.

ninternal - refractive index of the medium below the thin film.

Shader code

Here is the shader code:

irridescence.osl

```
/* Amplitude reflection coefficient (s-polarized) */
float rs(float n1, float n2, float cosI, float cosT) {
    return (n1 * cosI - n2 * cosT) / (n1 * cosI + n2 * cosT);
}

/* Amplitude reflection coefficient (p-polarized) */
float rp(float n1, float n2, float cosI, float cosT) {
    return (n2 * cosI - n1 * cosT) / (n1 * cosT + n2 * cosI);
}

/* Amplitude transmission coefficient (s-polarized) */
float ts(float n1, float n2, float cosI, float cosT) {
    return 2 * n1 * cosI / (n1 * cosI + n2 * cosT);
}

/* Amplitude transmission coefficient (p-polarized) */
float tp(float n1, float n2, float cosI, float cosT) {
    return 2 * n1 * cosI / (n1 * cosT + n2 * cosI);
}

// cosI is the cosine of the incident angle, that is, cos0 = dot(view angle, normal)
// lambda is the wavelength of the incident light (e.g. lambda = 510 for green)
// From http://www.gamedev.net/page/resources/_/technical/graphics-programming-and-theory/thin-film-
```

```
interference-for-computer-graphics-r2962
float thinFilmReflectance(float cos0, float lambda, float thickness, float n0, float n1, float n2) {
        float PI=3.14159265358979323846264338327795;

    // compute the phase change term (constant)
    float d10 = (n1 > n0) ? 0 : PI;
    float d12 = (n1 > n2) ? 0 : PI;
    float delta = d10 + d12;

    // now, compute cos1, the cosine of the reflected angle
    float sin1 = pow(n0 / n1, 2) * (1 - pow(cos0, 2));
    if (sin1 > 1) return 1.0; // total internal reflection
    float cos1 = sqrt(1 - sin1);

    // compute cos2, the cosine of the final transmitted angle, i.e. cos(theta_2)
    // we need this angle for the Fresnel terms at the bottom interface
    float sin2 = pow(n0 / n2, 2) * (1 - pow(cos0, 2));
    if (sin2 > 1) return 1.0; // total internal reflection
    float cos2 = sqrt(1 - sin2);

    // get the reflection transmission amplitude Fresnel coefficients
    float alpha_s = rs(n1, n0, cos1, cos0) * rs(n1, n2, cos1, cos2); // rho_10 * rho_12 (s-polarized)
    float alpha_p = rp(n1, n0, cos1, cos0) * rp(n1, n2, cos1, cos2); // rho_10 * rho_12 (p-polarized)

    float beta_s = ts(n0, n1, cos0, cos1) * ts(n1, n2, cos1, cos2); // tau_01 * tau_12 (s-polarized)
    float beta_p = tp(n0, n1, cos0, cos1) * tp(n1, n2, cos1, cos2); // tau_01 * tau_12 (p-polarized)

    // compute the phase term (phi)
    float phi = (2 * PI / lambda) * (2 * n1 * thickness * cos1) + delta;

    // finally, evaluate the transmitted intensity for the two possible polarizations
    float ts = pow(beta_s, 2) / (pow(alpha_s, 2) - 2 * alpha_s * cos(phi) + 1);
    float tp = pow(beta_p, 2) / (pow(alpha_p, 2) - 2 * alpha_p * cos(phi) + 1);

    // we need to take into account conservation of energy for transmission
    float beamRatio = (n2 * cos2) / (n0 * cos0);

    // calculate the average transmitted intensity (if you know the polarization distribution of your
    // light source, you should specify it here. if you don't, a 50%/50% average is generally used)
    float t = beamRatio * (ts + tp) / 2;

    // and finally, derive the reflected intensity
    return 1 - t;
}

surface irridescence
[[ string description = "Thin film coating shader. Use as reflection color for the material, with Fresnel for
the material OFF (this texture computes its own Fresnel)" ]]
(
 float thicknessMin = 250 [[ string description = "Minimum thickness of the film, in nm" ]],
 float thicknessMax = 400 [[ string description = "Maximum thickness of the film, in nm" ]],
 string thickness = "test.png" [[ string description = "Texture map (usually noise) that determines the
thickness of the film as fraction between the min and max" ]],
 float nmedium = 1 [[ string description = "Refractive index of the outer medium (typically air)" ]], //
approximate refractive index of air
 float nfilm = 1.5 [[ string description = "Refractive index of the thin film itself" ]] , // approximate
refractive index of water
 float ninternal = 1 [[ string description = "Refractive index of the material below the film" ]], //
approximate refractive index of the lower material
 output color reflColor = 0
)
{
 float cos0 = abs(dot(I , N));

 color thickTex=texture(thickness, u, v);
 float t=(thickTex[0]+thickTex[1]+thickTex[2])/3.0;
 float thick=thicknessMin*(1.0-t)+thicknessMax*t;

 float red=thinFilmReflectance(cos0, 650, thick, nmedium, nfilm, ninternal);
 float green=thinFilmReflectance(cos0, 510, thick, nmedium, nfilm, ninternal);
 float blue=thinFilmReflectance(cos0, 475, thick, nmedium, nfilm, ninternal);
```

```
 reflColor=color(red, green, blue);
}
```