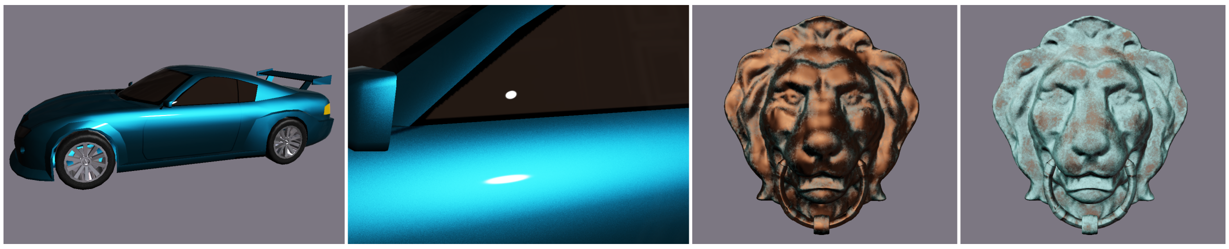
Layered Materials in Real-Time Rendering

Oskar Elek

Faculty of Mathematics and Physics
Charles University
Prague / Czech Republic



Abstract

Today's games and other real-time 3D applications often use only basic empirical models for modelling the appearance of materials and rely on complex geometry and texturing to make them more visually appealing. In this paper we explore the possibilities of bringing more physically plausible models to real-time 3D graphics.

We do this by implementing the layered BRDF of Weidlich and Wilkie on GPU. This model utilizes the well-known Torrance-Sparrow and Oren-Nayar microfacet models. We show how to make this layered model useful for real-time rendering through various optimizations. Then we derive two specialized models based on this basic layered model. These two models attempt to simulate the appearance of metallic car paints and metallic patinas.

Keywords: surface reflectance models, appearance modelling, layered materials

1 Introduction

Our environment contains a large variety of objects and materials with surfaces composed of multiple layers, for instance coated ceramics and plastics, varnished and patinated metals, organic tissues etc. Rendering such materials in real-time applications is difficult — traditionally used empirical BRDFs, such as the Phong reflectance model, are not capable of reproducing the appearance of most of them, and possibility of utilization of measured BRDF or BTF data is limited on GPU. There is a need for a simple analytical BRDF that can reproduce the appearance of layered materials in interactive applications.

This paper extends the work of Weidlich and Wilkie [21], who presented a simple analytical physically-based BRDF suitable for rendering layered materials. Their model is based on combination of commonly used BRDFs, such as Torrance-Sparrow [20] and Oren-Nayar [15] reflectance models. We show how to transfer their work into the environment of real-time evaluation in fragment shaders, along with various convenient optimizations. We then demonstrate the potential of this model by employing it in two specialized models for rendering metallic car paint and copper patina.

The paper is organized as follows: we first give an overview of related work in the field and review the layered model of Weidlich and Wilkie. Then we discuss a real-time adaptation of this model, along with two specialized models for rendering metallic car paint and metallic patinas. Finally, we measure the performance of our implementation and discuss its possible utilization in real-time applications.

2 Background and Related Work

2.1 Microfacet Reflectance Models

Analytical reflectance models used in 3D computer-generated imagery can be generally divided between two groups: *empirical* and *physically-based*. Empirical models are based on direct observation and therefore are usually physically implausible (except for the limit cases — ideal diffuse and mirror reflectors). The most common BRDFs from this group are Phong [16] model and its modification by Blinn [1], both widely used in real-time rendering applications. The prevalence of these models in real-time rendering is due to their low computational requirements and fairly good reproduction of overall object appearance.

In contrast to these, physically-based BRDFs model reflectance properties of materials from first optical principles, which leads to a more plausible appearance of materials. A BRDF is physically plausible, if it conserves energy (i.e. albedo is always at most 1), obeys Helmholtz reciprocity principle and is non-negative. To retain a closed analytical form, they often use a statistical surface representation, instead of an explicit one. Surfaces are represented by statistically distributed tiny microfacets or V-shaped cavities. We will discuss two physically-based BRDFs, namely the Torrance-Sparrow [20] and Oren-Nayar [15] reflectance models.

The Torrance-Sparrow reflectance model builds on the assumption that the material surface consists of microscopic V-shaped cavities that behave like perfect mirrors with Fresnel reflectance. The amount of reflected light f_r is defined as

$$f_r = \frac{FDG}{\pi(N\cdot L)(N\cdot V)} \tag{1}$$

- $F(\beta, n, \kappa)$ is the Fresnel term. It expresses the reflectance coefficient of each individual microfacet. β is the angle between incident light direction L (or view direction V) and half vector H, n and κ are real and imaginary components of the material's index of refraction (IOR). Note that F is wavelength-dependent. The full Fresnel term equations can be found in [2] or [7].
- $D(\alpha, m)$ is the microfacet slope distribution function representing the amount of microfacets oriented towards the observer. α is the angle between half vector H and surface normal N, $m \in \langle 0, 1\rangle$ is the surface roughness parameter. Small values of m represent very smooth surfaces, while values close to 1 correspond to rough surfaces. For $m \to 0$ the term D converges to Dirac δ-function. Commonly used distributions are for instance the Beckmann distribution ($D = \frac{1}{m^2 \cos^4 \alpha} e^{-(\tan\alpha/m)^2}$), the Blinn distribution or the Gaussian distribution.
- $G(L, V, N)$ is the geometry attenuation term, which expresses the amount of light attenuated after shadowing-out in the surface microfacet structure. Please refer to [3] for the complete formula.

The Torrance-Sparrow model was introduced to computer graphics by Cook and Torrance [3], who added a diffuse term and some minor modifications into the model. It is suitable for modelling wide variety of surfaces, especially metals. It is also successful in predicting phenomena such as off-specular reflection and specular backscattering. The HLSL code of this model can be found in Appendix B.

The Oren-Nayar model represents a generalization of the standard Lambertian reflectance. It is similar to the Torrance-Sparrow model in its assumptions; the only difference between them is that the Oren-Nayar model treats each individual microfacet as diffuse reflector, not as a prefect mirror. This complicates the situation, because it is now necessary to take multiple interreflections between neighbouring microfacets into account. The resulting model from Oren and Nayar is therefore only an approximation of the full solution, although a very good one. Its full formulation can be found in [15].

The model is suitable for modelling rough materials with dominant diffuse component, such as clay, stone or uncoated paper. It is capable of reproducing the effect of diffuse backscattering, characteristic for example for the full Moon. Despite its moderate computational costs, it is rarely used in real-time rendering. The reason for this is that the standard 'N-dot-L' Lambertian reflectance exhibits very similar reflectance behaviour, but is far superior in terms of computational performance.

2.2 Layered Reflectance Models

Due to the frequent occurrence of layered materials in our environment, the search for model capable of rendering such materials has received adequate attention. Kubelka and Munk [12, 10, 11] developed a physical model for modelling subsurface scattering within multiple layers. Hanrahan and Krueger [8] presented a model for subsurface scattering computation in the context of computer graphics, and made it directly usable in a Monte-Carlo renderer. These models are comprehensive, but lack an analytical closed form, and therefore are not suitable for real-time applications.

As for analytical models, Neumann and Neumann [14] proposed a simple model for modelling multiple layers. However, they considered only perfectly smooth, transparent layers without including internal reflection. Kelemen and Szirmay-Kalos [9] presented a composite BRDF derived from the Cook-Torrance model. Their model does not explicitly consider surface layers and therefore does not account for absorption and internal reflections, but thanks to the tight coupling between diffuse and specular BRDF components, their model estimates the appearance of single-layered surfaces fairly well.

Finally, the model of Weidlich and Wilkie [21] accounts for both internal reflection and absorption and supports unlimited number of layers. Each layer can have any arbitrary BRDF; the only requirement on these BRDFs is that for all layers except the lowest a transmission component, which allows the light enter the layer underneath, must exist (for example the Cook-Torrance model does contain a transmissive component, while the Oren-Nayar model does not). The only limitation is that the model does not support scattering within the layers. Since this model forms the starting point for our work, we will briefly review it now.

It is worth mentioning that the model relies on the assumption of layers, which are thin in comparison with the surface geometric features. This is not uncommon; in fact all previously mentioned models rely on this. This allows to assume that all incident and refracted rays meet at a single point at every layer (which is very convenient, as it allows purely local evaluation of the model, without including surface spatial position into the model). Fortunately, this condition, along with the assumption of nonpresent subsurface scattering, still hold true for wide variety of materials.

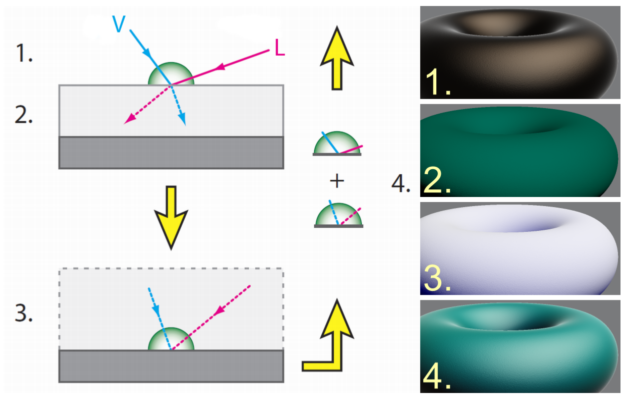


Figure 1: The recursive evaluation scheme. Each stage is also shown graphically. Scheme used with permission.

The evaluation of the model is carried out in a recursive manner and can be described in four steps for two given layers i and $i+1$ (see also Equation 2 and Figure 1):

1. The BRDF of the upper layer f_{r_i} is evaluated for given light and view directions L and V. This also produces the trans-

mittance coefficient $T_{i \to i+1} = 1 - F_i$. Two refracted directions L' and V' are calculated, according to Snell's law for given refractive indices n_{i-1} and n_i ($n_0 \approx 1$ if the object is in the air).

2. The refracted light is attenuated by the absorption term a_i.
3. The BRDF of the lower layer $f_{r_{i+1}}$ is evaluated for L' and V'. If the layer $i+1$ is not the lowest one, we recursively continue from Step 1 ($f_{r_{i+1}} \equiv f_r^{(i+1)}$).
4. On return from the recursion, the light coming from the lower layer is attenuated by $T_{i+1 \to i}$ and subjected to possible total internal reflection t_i. The contributions from both layers are added together.

This can be expressed in the form of a recurrent equation for the composite BRDF at layer i as:

$$f_r^{(i)} = f_{r_i}(L,V) + T_{i \to i+1} \cdot f_{r_{i+1}}(L',V') \cdot a_i \cdot t_i \qquad (2)$$

- a_i is the attenuation term according to Bouguer-Lambert-Beer law. The portion of absorbed light depends on the material-specific wavelength-dependent absorption coefficient σ and the distance the light travels in a particular layer:

$$a_i = e^{-\sigma l_i} \qquad l_i = d_i \cdot \left(\tfrac{1}{N \cdot L'} + \tfrac{1}{N \cdot V'}\right) \qquad (3)$$

  where d_i is the thickness of the layer i.

- t_i is the internal reflection term. It compensates for the energy lost during the potential total internal reflection of light when crossing an inter-layer boundary from denser into a less dense medium on its way upwards. It is defined as

$$t_i = (1 - G_i) + T_{i+1 \to i} \cdot G_i \qquad (4)$$

  where G_i is the Torrance-Sparrow geometry attenuation term.

The final value of the entire model is simply obtained as $f_r = f_r^{(1)}$. For the detailed discussion of the model, please refer to the original paper [21].

2.3 Specialized Material Models

Specialized material models are utilized when the available general BRDFs cannot reproduce the desired material's appearance well enough. As a consequence there is a large variety of such models, each aiming to simulate a single particular effect. Therefore we will list only those few which are relevant for us here; for a comprehensive overview of these models, please refer to [5].

Modelling of car paint is a subject of intensive research, since it is important for virtual prototyping in the automotive industry. Takagi et al. [18, 19] developed techniques for both acquisition and rendering of car paints, that are directly applicable in the industry. Ershov et al. [6] developed an interactive model for pearlescent car paint rendering by simulating scattering between virtual thin sublayers. Recently, Rump et al. [17] introduced a realistic hybrid model for metallic car paint rendering that combines acquired BTF data and classical BRDFs, such as Cook-Torrance model.

As for modelling of metallic patinas, Dorsey and Hanrahan [4] presented a method for simulating growth of patinas on metallic objects by considering multiple layers of material and applying eroding operators on them. For rendering they used the Kubelka-Munk theory with three layers for copper substrate, tarnish and the patina itself. Yao-Xun and Zen-Chung [22] simulated patina growth on objects buried underground, using L-systems. However, these works are focused on patina development simulation and not on rendering.

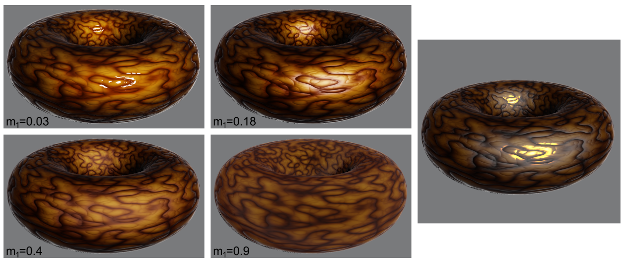


Figure 2: Variation of the upper layer's roughness influences shape of the reflection from the lower layer with $m_2 = 0.3$. The image on the right shows a torus with $m_1 = 0.4$ and $m_2 = 0.05$ without the correction.

3 Layered Materials in Real-Time

The basic version of the real-time layered model adaptation is a relatively straightforward implementation of the evaluation scheme presented in Section 2.2. Algorithm 1 shows a Cg fragment shader for the model with two layers, one light source and an environment reflection from the upper layer. Both layers use the Torrance-Sparrow BRDF, plus a diffuse component for the lower layer, and are normal-mapped.

The main difference is of course the lack of capacity to explicitly cast sampling rays. This has several implications. First, we must strictly stick to the evaluation of the local model with given L and V directions (or L' and V' for the lower layer, respectively). Also the environment reflection must take into account the upper layer roughness, and without sampling this can be achieved only by providing an adequately blurred environment map for the texture lookup at Line 29. Otherwise an inconsistency between the environment reflection sharpness and the specular highlight shape will occur.

Another consequence of the inability to actually sample the BRDF is that a discrepancy between the roughness of the layers might cause an incorrect appearance of the surface, specifically when $m_1 > m_2$. Figure 2 depicts the problem. This cannot happen in Monte-Carlo rendering, because the light gets properly blurred during the refraction on the upper layer. The solution here is to clamp the value of m_2 to the value of m_1 (see Line 21), so that the lower layer's roughness is always greater or equal to the one of the upper layer.

Although the shader we show uses only two layers, there is no obstacle to using more layers (except performance considerations), thanks to the recurrent character of Equation 2. Adding a layer would mean calculation of a new pair of refracted vectors L'' and V'', terms a_2 and t_2 for the new layer and of course of its own BRDF. Another means of enhancing the model's visual richness would be for example adding a thickness map for the upper layer (as in Figure 2), or using a roughness map to vary m across the surface.

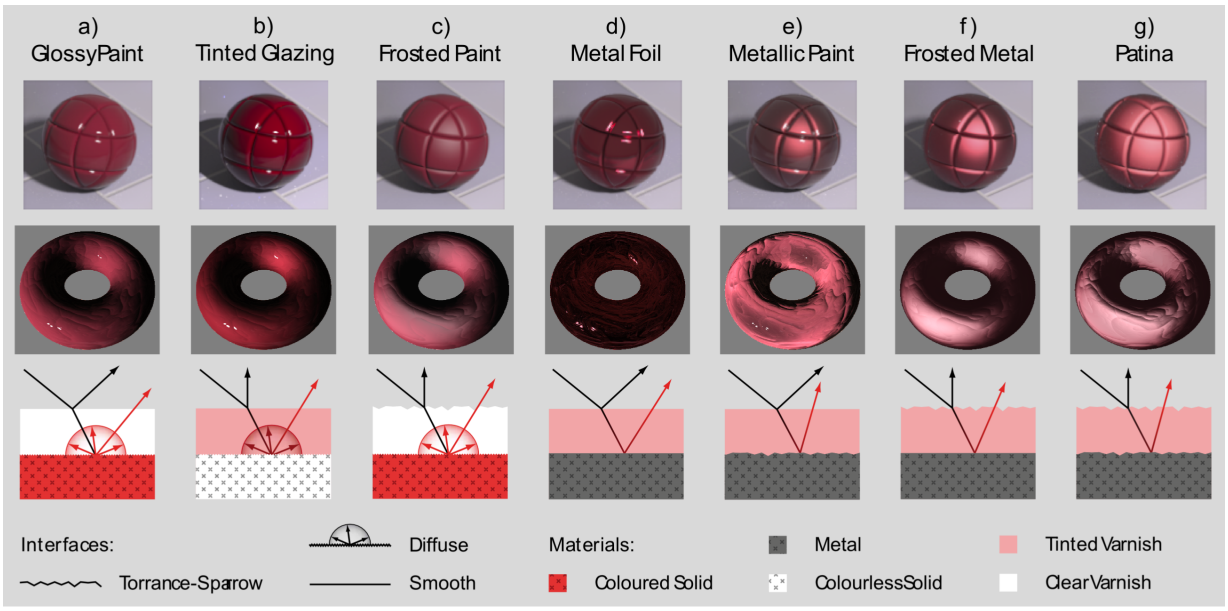


Figure 3: Examples of surfaces that can be generated by the layered model. Top row: Results from the original paper. Bottom row: Images generated by our real-time implementation, with parameters adjusted to the best visual match with their corresponding Monte-Carlo versions. Note that the only qualitative difference is the absence of global illumination. The original image used with permission.

3.1 Optimizations

Undoubtedly the largest performance impact is caused by evaluation of the Torrance-Sparrow model. We will therefore try to speed up its evaluation, what practically means optimizing evaluation of F, D and G terms. We will show that it is possible to precompute F and D.

Fresnel term $F(\theta, n, \kappa)$: The Fresnel term evaluation is the most expensive operation in the model, and is called three times in the basic version of the layered model. The value of F depends on the angle θ between two considered vectors and on the material index of refraction (n,κ). This yields 7 degrees of freedom (for RGB colour components), making a naive precomputation impossible. Of course, it would be possible to separate the R, G and B components of (n,κ) and create three 3D tables, one for each colour component. This is however not the best solution, because it would increase the number of lookups needed to evaluate F from one to three. The solution lies in the fact that for a given material, the value of (n,κ) is fixed, which allows creation of a 1D table parameterized only by the incident angle θ. Such 1D table would contain all possible values of F for that particular material and would be very compact — since the gradient of F in respect to θ is low, the resolution of this table can be small, for example 128 samples. For R8G8B8 texture (which is sufficient, since the value of F is always $\in \langle 0,1 \rangle$) this means the size of only 384 bytes. This allows for creation of a 2D texture atlas for hundreds of materials in the scene with the size in the order of tens of kB.

If precomputation is not desired for some reason, F can be approximated. Lazányi and Szirmay-Kalos [13] presented an accurate and inexpensive approximation of the full Fresnel term, which deviates from the full formula in 5% at most.

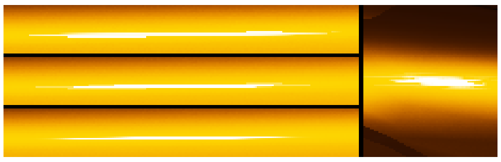


Figure 4: Zoom on a sharp specular highlight ($m_1 = 0.01$) using tabulated D term with linear mapping (top), nonlinear mapping (middle) and the full evaluation (bottom). The right image shows a highlight when MIP-mapping is enabled on the texture which contains D.

Distribution term $D(\alpha, m)$: Dimensionality is not an issue here, since both α and m are scalars, so a 2D table can hold the entire distribution term. The complication here is the convergence of D towards the Dirac δ-function when $m \rightarrow 0$, independent of which distribution is used. This implies that for very small values of m and $\alpha \rightarrow 0$, the large gradient of D cannot properly be reproduced, even if a large sampling rate is used for α (see Figure 4). To remedy this problem, a non-linear mapping must be used for α — on the coordinate $u \in \langle 0,1 \rangle$ the texture holds a value of D, which would be stored in the linearly-mapped texture on the coordinate u^x. We use $x = 8$, since u^8 can be computed in 3 multiplications. But still, even with linear mapping, this issue starts to be apparent just for very smooth surfaces (with $m < 0.02$).

It is also better to disable MIP-mapping for the texture containing D. The reason for this is that the higher MIP levels will blend together adjacent values of the texture (which correspond the different values of m), resulting in an incorrect size and jaggedness of the specular highlight, when viewed from distance. As for the texture resolution, we use a single-channel 16b

```
Data: vertex shader output structure IN, uniform variables $m_1$, $m_2$, d
      (float) and $n_1$, $\kappa_1$, $n_2$, $\kappa_2$, $\sigma$, lightCol (float3), texture samplers
Result: fragment colour
begin
        // calculate involved vectors...
    float3 N = normalize(2 * tex2D(normalMap, IN.UV).xyz - 1);
    float3 L = normalize(IN.lightDir);
    float3 V = normalize(IN.eyePos - IN.fragmentPos);
    float3 H = normalize(V + L);
    float3 R = reflect(-V, N);
    float3 L' = -refract(L, N, 1/$n_1$);
    float3 V' = -refract(V, N, 1/$n_1$);
    float3 H' = normalize(V' + L');
        // ...and their dot products
    float NdotL = dot(N, L);
    float NdotH = dot(N, H);
    float NdotV = dot(N, V);
    float VdotH = dot(V, H);
    float NdotL' = dot(N, L');
    float NdotH' = dot(N, H');
    float NdotV' = dot(N, V');
    float V'dotH' = dot(V', H');
        // BRDFs for both layers
    float3 $F_1$, $F_2$;
    float $G_1$, $G_2$;
        // F and G are 'out' parameters
    float3 $f_1$ = TorranceSparrow(NdotL, NdotV, NdotH, VdotH, $n_1$,
    $\kappa_1$, $m_1$, $F_1$, $G_1$);
    float3 $f_2$ = TorranceSparrow(NdotL', NdotV', NdotH', V'dotH',
    $n_2$, $\kappa_2$, max($m_2$, $m_1$), $F_2$, $G_2$);
        // diffuse contribution of lower layer
    $f_2$ += (1 - $F_2$) * max(NdotL, 0) * tex2D(diffuseMap, IN.UV);
        // internal reflection term
    float3 $T_{12}$ = 1 - $F_1$;
    float3 $T_{21}$ = $T_{12}$;
    float3 t = (1 - $G_1$) + $T_{21}$ * $G_1$;
        // attenuation term
    float $l$ = d * (1/NdotL' + 1/NdotV');
    float3 a = exp(-sigma * $l$);
        // environment reflection for upper layer
    float3 $F_{1_{env}}$ = FresnelTermNP(NdotV, $n_1$, $\kappa_1$);
    float3 envCol = $F_{1_{env}}$ * texCUBE(environmentMap, R);
        // final summation
    float3 $f_r$ = lightCol * ($f_1$ + $T_{12}$ * $f_2$ * a * t);
    return float4($f_r$ + envCol, 1);
end
```

Algorithm 1: The layered model shader code.

floating-point texture with 512 samples for α and 512 samples for m, resulting in the size of 0.5MB.

Geometry term $G(L,V,N)$**:** The precomputation of the geometry term is not necessary, as most of the involved dot products have to be calculated anyway, leaving only a few multiplications and two divisions to be evaluated. However, should such need arise, Kelemen and Szirmay-Kalos [9] showed a way to exclude the evaluation of G from the Torrance-Sparrow model.

Of course, a completely different way of speeding up the computation can be taken — instead of using the Torrance-Sparrow model, one can use a simpler BRDF for the layers. Even the Phong model can be used, with the transmission term T derived from the amount of reflected light or by explicitly evaluating the Fresnel term. However, usage of the Phong model decreases the richness of appearance that can be achieved with physically plausible models.

A comparison between the full and precomputed model versions can be seen on Figure 5. A comparison between the results of the real-time version of the model and the original Monte-Carlo implementation of Weidlich and Wilkie can be seen in Figure 3. Figure 10 shows more examples of the model usage.

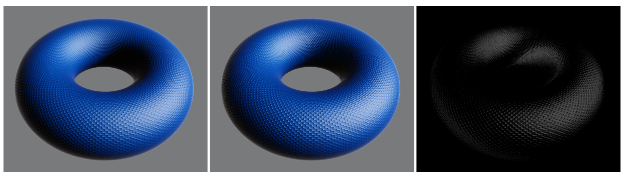


Figure 5: Visual comparison of the full evaluation (left) and using precomputed tables for F and D (middle). The right image shows a magnified difference between the two ($|full - precomputed|$); the largest error is less than 6%.

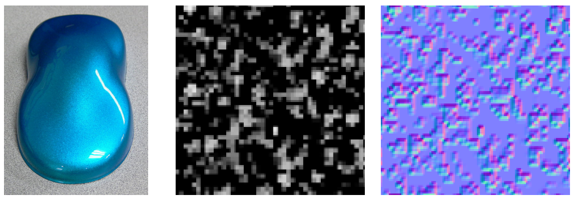


Figure 6: Left: Sparkling effect on an object coated with metallic paint. Right: An example of a texture representing the metallic flakes heightfield and the corresponding normal map.

4 Specialized Materials Modelling

In this section we will present two specialized models for modelling the appearance of metallic car paint and metallic patina. Both are straightforward modifications of the model discussed previously.

4.1 Metallic Paint

Standard solid car paints usually consist of an opaque pigment layer sprayed over base substrate. Metallic paints use translucent pigments which in addition contain very small metallic flakes and can optionally be covered with a clear coating. This structure is responsible for two characteristic appearance features of metallic car paints — an overly metallic look (naturally caused by the material the flakes are made of, e.g. chromium) and sparkling effect, which is especially visible under direct sunlight. This sparkling effect is caused by the fact that the flakes spread in the medium are almost randomly oriented and reflect the incoming light to different directions (see figure 6 for illustration).

To model a surface with such structure it is natural to use the layered model. We use two layers, a lower 'substrate' layer made of chromium and a tinted 'coating' layer. The sparkling effect is achieved by perturbing the surface normal with the texture shown in Figure 6, but only for the lower layer. This normal map is tiled many times across the surface, so that the individual flakes are not visible.

Two problems arise from this approach. The first problem is the consequence of the fact that the flakes are smaller than the size of a pixel. This is modelled by the aforementioned tiling of the perturbing normal map. By doing this, however, a MIP map of the texture is used instead of the full texture to fetch the value of the perturbed normal. This effectively smoothes out all

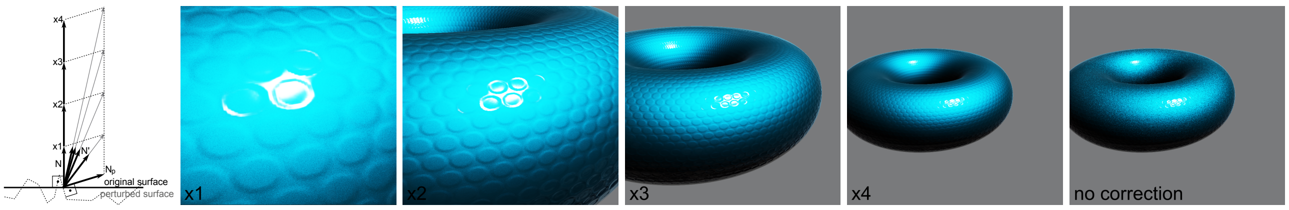


Figure 7: Left: The scheme depicting the scaling of the perturbed normal. Middle: Four distances from the object corresponding to the scheme. Right: The object without the scaling correction.

details provided by the flakes normal map. To overcome this, all filtering must be disabled for the normal map. This produces the effect of noisy sparkling, which dynamically changes as the observer moves.

The second problem arises from the solution to the first one. Minification filtering and MIP mapping ensure that high-frequency features on the texture are filtered so aliasing does not occur. By disabling these, the noisy sparkling is apparent even from distance. In reality this cannot happen, because as the observed object gets further from the eye, more flakes are projected onto the same area on the retina, effectively averaging and smoothing the perceived image. This can be solved by scaling the original surface normal by the distance of the observer from the fragment and add this scaled normal to the perturbed normal (see Figure 7). This effectively decreases the influence of the perturbed normal, making the surface look smooth from distance. In code, this can be written as:

```
float distance = length(IN.eyePos - IN.fragmentPos);
    // N currently contains the normal-mapped normal
    // also remember we are in tangent space
N += max(distance, 1) * float3(0, 0, 1) +
  (2 * tex2D(flakesNormalMap, 1000 * IN.UV).xyz - 1);
N = normalize(N);
```

Multiplication of the UV coordinates by a large number tiles the normal map. This block of code is to be inserted between Lines 13 and 14 in the Algorithm 1 (to influence only the angles between the refracted rays).

4.2 Patina

Patination is a chemical oxidation process which occurs on metals. It changes the chemical composition near the surface of the material, often resulting in a layer of substance with different optical properties than the original material. Unlike rusting, patination does not destructively erode the metal; instead, it forms a solid protective layer atop of the metal substrate, which then stays stable. Patination occurs on many common metals and alloys, for example on copper, brass, aluminium, tin and even on silver. We chose to model copper patina, because of its distinct appearance.

Copper patinas often have complex chemical composition, which tend to differ with the atmospheric conditions the copper object is exposed to. The involved substances are cuprite Cu_2O, antlerite, brochantite and possibly others. Since the physical constants for these are not widely available, we use for the patina the IOR of cuprite and an empirically matched light-green absorption spectrum.

An important thing is to have control over the patina growth, i.e. to determine places where patina is already developed and where not yet. Naturally, this changes over time. To model the development we use a single grayscale texture (see Figure 9) and two parameters: *transition* $\in \langle 0,1\rangle$ and *extent* $\in \langle -\mathit{transition}, 1+\mathit{transition}\rangle$. The first one controls the sharpness of the patina regions and the second one controls the extent of patina development (the larger is the *extent* parameter, the more of the surface is covered with patina). As for the texture, the darker the value, the earlier will the patina develop on that particular position. The texture can be derived from the surface curvature (as in our case, since the patina develops earlier in places like cracks and wrinkles) or can be an output from an actual weathering simulation. The code for this looks as follows:

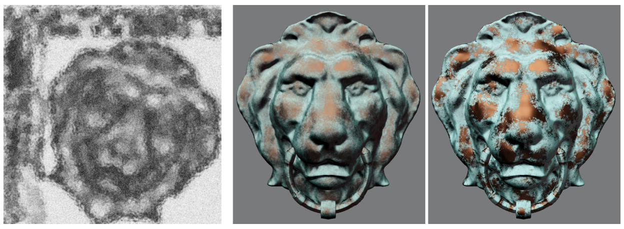


Figure 9: Left: A grayscale map that controls the patina development. Right: Blurry (`transition` $= 0.9$) and sharp (`transition` $= 0.2$) boundaries of the developed patina regions.

```
float patinaValue = tex2D(patinaMap, IN.UV).r;
float extentFactor = 1-smoothstep(extent - transition,
                                  extent + transition,
                                  patinaValue);
```

The variable *extentFactor* $\in \langle 0,1\rangle$ is then used through the entire shader to control the amount of patina.

It is necessary to realize that on the places where the patina is not developed yet, only the reflection from the lower layer has to be taken into account. The development of patina also changes some properties of the lower layer, for example its roughness. Figure 8 demonstrates this process. Therefore the model has to be evaluated for both cases (with and without the upper patina layer) and the *extentFactor* variable should be used to linearly interpolate between them. The *extentFactor* should be also used to interpolate between the model parameters influenced by the patina growth, namely the aforementioned roughness parameter m_2 and the IOR value (n, κ) used to calculate the environment reflection (Line 28 in Algorithm 1).

Unfortunately, the substance that forms the patina layer does not conform to the assumption that no light is scattered within the layer. Quite the contrary — it is the scattering that makes the patina appear primarily as a diffuse reflector. To avoid subsurface

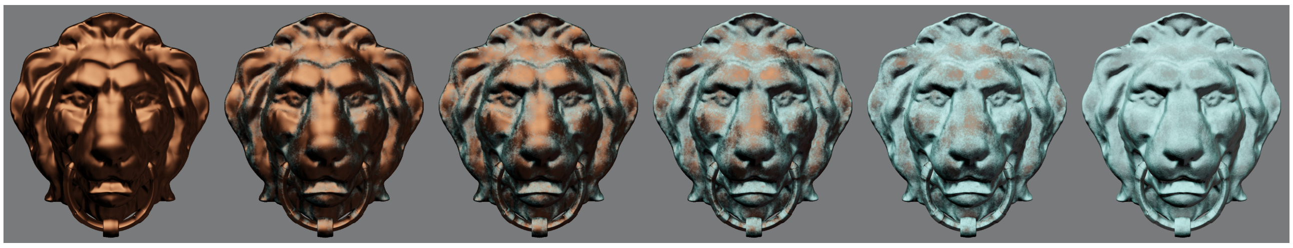


Figure 8: Patina development 'over time' by changing the *extent* parameter value (from left to right -0.65, -0.05, 0.15, 0.45, 0.75 and 1.65; *transition* $= 0.65$).

scattering computation we approximate this effect by adding a diffuse component to the BRDF of the patina layer, that is:

```
float3 patinaDiffuse = (1 - sigma)
                * (1 - F_1) * max(NdotL, 0);
f_1 += patinaDiffuse;
```

We modulate the diffuse component by the remainder of the absorption spectrum σ to strengthen the characteristic colour of the patina. To further improve the appearance of patina, we control its thickness with an additional texture (as proposed in Section 3) and modulate the base copper layer with a dark-orange tone to mimic the appearance of tarnish.

5 Results and Conclusions

We implemented the basic model and the derived specialized models in HLSL, using NVIDIA FX Composer 2.5 for development and measurements. We test the basic two-layer model in two versions, using the full evaluation and using precomputed tables for F and D, against a single-layer Phong shader with similar features (normal mapping, environment reflection). We do not measure the two specialized models, as these are simple modifications of the basic model and do not add significant computational overhead. The measurements used GeForce 8800 GTX (G80) as a reference GPU. Table 1 summarizes the measurements.

Model	GPU cycles	MPix/s
Layered (full)	436	348
Layered (precomp.)	236	757
Phong	104	1648

Table 1: Performance comparison of the layered model and standard Phong model. The measured quantities are the number of G80 GPU cycles used for model's evaluation and the corresponding pixel throughput.

So, the discussed layered model is still roughly 2.3 times slower than the Phong model. This is expected however, since we are evaluating two BRDFs instead of one, and each of them being far more complex than the Phong model.

Conclusions Although the discussed layered model is slower than the commonly used reflectance models in real-time 3D applications, it provides superior appearance reproduction of a wide variety of surfaces, and is physically plausible. Moreover, it is very likely that the performance impact of using this model in a real-time application would be only a few percent, because:

- It is not necessary to use the model on all objects in a scene, but only on objects in the user's primary attention (e.g. cars in a racing game).
- The performance of any renderer is not given solely by the performance of the used reflectance models, but is mainly a consequence of many other tasks the renderer have to perform.

This indicates that the model is a viable alternative for use in today's real-time 3D applications, including games.

To conclude the paper; we have presented a real-time implementation of the physically-based layered BRDF introduced by Weidlich and Wilkie [21]. We have then shown how to optimize this model to be useful in real-time 3D applications. Furthermore, we have explored the capabilities of this model by deriving two specialized models for modelling the appearance of metallic car paint and metallic patina from it. The FX Composer project containing all discussed shaders will be made available on the author's webpage.

6 Acknowledgements

I would like to thank Alexander Wilkie for his insights and advice regarding this work.

References

[1] J. F. Blinn. Models of light reflection for computer synthesized pictures. In *Proceedings of SIGGRAPH '77*, pages 192–198, 1977.

[2] M. F. Born and E. Wolf. *Principles of Optics*. Cambridge University Press, 7th edition, 1999.

[3] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. In *Radiometry*, pages 42–59, 1992.

[4] J. Dorsey and P. Hanrahan. Modeling and rendering of metallic patinas. In *Proceedings of SIGGRAPH '96*, pages 387–396, 1996.

[5] J. Dorsey, H. Rushmeier, and F. Sillion. *Digital Modeling of Material Appearance*. Morgan Kaufmann Publishers, 2008.

[6] S. Ershov, K. Kolchin, and K. Myszkowski. Rendering pearlescent appearance based on paint-composition modelling. *Comput. Graph. Forum*, 20(3), 2001.

[7] A. S. Glassner. *Principles of Digital Image Synthesis Volume Two*. Morgan Kaufmann Publishers, 1995.

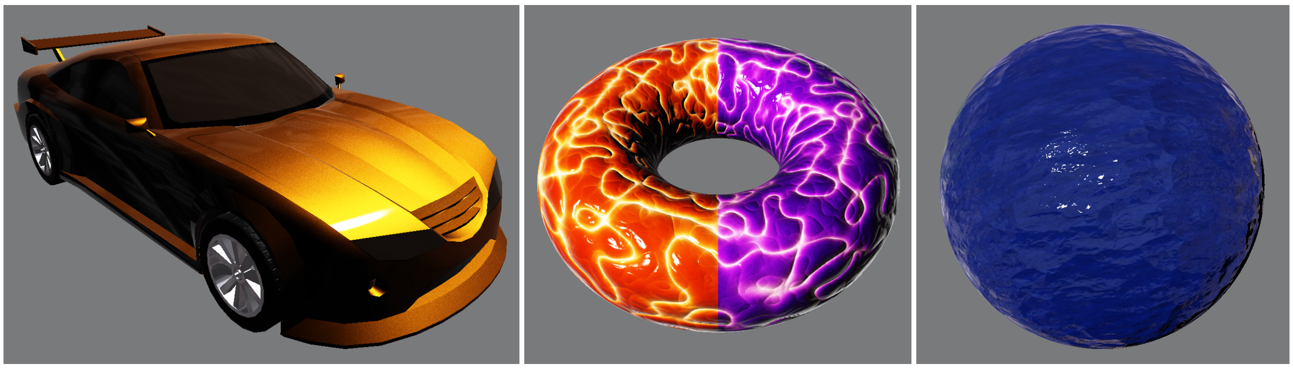


Figure 10: Examples of the model usage. Left: Car paint with strong sparkling effect. Middle: Varying upper layer thickness (orange and purple lacquers used). Right: Concrete ball coated in a blue transparent varnish. The lower concrete layer uses the Oren-Nayar BRDF.

[8] P. Hanrahan and W. Krueger. Reflection from layered surfaces due to subsurface scattering. In *Proceedings of SIGGRAPH '93*, pages 165–174, 1993.

[9] C. Kelemen and L. Szirmay-Kalos. A microfacet based coupled specular-matte BRDF model with importance sampling. In *Eurographics Short Presentations*, pages 25–34, 2001.

[10] P. Kubelka. New contributions to the optics of intensely light-scattering materials. Part I. *J. Opt. Soc. Am.*, 38(5):448–448, 1948.

[11] P. Kubelka. New contributions to the optics of intensely light-scattering materials. Part II: Nonhomogeneous layers. *J. Opt. Soc. Am.*, 44(4):330–334, 1954.

[12] P. Kubelka and F. Munk. Ein beitrag zur optik der farbenstriche. In *Z. tech. Physik 12*, pages 593–601, 1931.

[13] I. Lazányi and L. Szirmay-Kalos. Fresnel term approximations for metals. In *WSCG 2005 Short Communications Proceedings*, 2005.

[14] L. Neumann and A. Neumann. Photosimulation: Interreflection with arbitrary reflection models and illumination. *Comput. Graph. Forum*, 8(1):21–34, 1989.

[15] M. Oren and S. K. Nayar. Generalization of Lambert's reflectance model. In *Proceedings of SIGGRAPH '94*, pages 239–246, 1994.

[16] B. T. Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317, 1975.

[17] M. Rump, G. Müller, R. Sarlette, D. Koch, and R. Klein. Photo-realistic rendering of metallic car paint from image-based measurements. *Comput. Graph. Forum*, 27(2), 2008.

[18] A. Takagi, H. Takaoka, T. Oshima, and Y. Ogata. Accurate rendering technique based on colorimetric conception. In *Proceedings of SIGGRAPH '90*, pages 263–272, 1990.

[19] A. Takagi, A. Watanabe, and G. Baba. Prediction of spectral reflectance factor distribution of automotive paint finishes. *Color Research and Application*, 30(4), 2005.

[20] K. E. Torrance and E. M. Sparrow. Theory for off-specular reflection from roughened surfaces. In *Radiometry*, pages 32–41, 1992.

[21] A. Weidlich and A. Wilkie. Arbitrarily layered micro-facet surfaces. In *GRAPHITE 2007*, pages 171–178, 2007.

[22] Ch. Yao-Xun and S. Zen-Chung. Physically-based patination for underground objects. *Comput. Graph. Forum*, 19(3), 2000.

A Selected coefficients

The following table lists a few selected indices of refraction for materials that have been used in the paper (in the models for metallic car paint and metallic patina rendering).

Material\ λ	**r**[690nm]	**g**[550nm]	**b**[450nm]
Copper (Cu)			
n	0.213	1.04	1.17
κ	4.05	2.59	2.36
Chromium (Cr)			
n	3.84	3.18	1.99
κ	4.37	4.41	4.22
Cuprite (Cu_2O)			
n	2.83	3.10	3.06
κ	0.083	0.19	0.6

B Torrance-Sparrow model

```
float3 TorranceSparrow(float NdotL, float NdotV,
                       float NdotH, float VdotH,
                       float3 n, float3 k, float m,
                       out float3 F, out float G)
{
    //D term - Beckmann distribution
   float D;
   float tg = sqrt(1 - NdotH * NdotH) / NdotH;
   D = 1 / (m * m * NdotH * NdotH * NdotH * NdotH)
            * exp(-(tg/m) * (tg/m));

    //F term - Lazanyi-Szirmay-Kalos approximation
   float q = 1 - VdotH;
   F = ((n - 1)*(n - 1) + 4 * n * q*q*q*q*q + k*k)
      / ((n + 1)*(n + 1) + k*k);

    //G term
   G = min(1, min(NdotV * (2 * NdotH) / VdotH,
                  NdotL * (2 * NdotH) / VdotH));

    //entire model
   return F * D * G / (4 * NdotV);
}
```