AxF - Appearance exchange Format

Gero Müller, Jochen Tautges, Alexander Gress, Martin Rump,
Max Hermann, and Francis Lamy
X-Rite, Inc., 4300 44th St. SE, Grand Rapids, MI 49505

January 15, 2021
Version 1.8

Abstract

In this paper X-Rite presents a file format for exchanging digital material appearance. The format is a vital part of X-Rite's appearance initiative consisting of a new generation of appearance capturing devices. In order to make the usage of measured appearance as simple as possible for users, a broad support by software vendors is essential. Therefore, X-Rite aims to initiate the assembly of a consortium of hard- and software vendors, members of the scientific community and users, which would be responsible for the further development and standardization of the format.

1 Motivation

Driven by the ongoing virtualization of the product design process we see a growing demand for measured material appearance. This trend pushes the need for efficient and standardized exchange, communication, and archiving of digital appearance data.

Nonetheless, current digital material workflows are far from being standardized. In practice, most often proprietary data formats are used. If materials need to be transferred between software packages, data usually needs to be stripped down to a least common denominator like an image or a simple reflectance model like Blinn-Phong, which is used for instance by OpenGL's and Direct3D's fixed function pipeline. While workarounds can be found for individual cases, this situation is not acceptable for users of appearance measurement devices because a physical material measurement should be independent of specific 3D rendering software platforms.

We are aware of the fact that due to the highly competitive and innovative nature of the field it is not an easy task to reach for standardization in graphics. On the other hand, starting this endeavor from the measurement side seems to be a promising approach. It should be the common goal of both software and hardware vendors to achieve a consistent reproduction of measured appearance across different systems. This, and nothing less, is expected by the users.

2 Requirements

Besides mandatory requirements like efficiency and platform independence we want to bring attention to some special requirements for appearance data.

Scalability The format should be able to store raw measurement data, which easily can range up to several gigabytes. Therefore the format must be scalable, which means access times should not depend on the file size. This requirement rules out text-based formats like XML, which lack indexing capabilities.

Generality The structure of appearance data can be quite diverse, ranging from a single spectrum up to complex combinations of data from hundreds of sensors or even procedural descriptions. Therefore, the format must be extensible and self-describing in order to accommodate all these different kinds of representations. This requirement does not only hold for payload but also for metadata.

Workflow compatibility Supporting a least common denominator of appearance representations in addition to a full blown BTF or even BSSRDF is critical for format adoption. This requires the ability to store different versions of a material in a single file. We propose to define an SVBRDF variant as least common denominator since most state of the art rendering software supports at least a material model based on diffuse and specular color textures, normal and/or height map and some specification of gloss.

In an industrial environment, data security and protection are also important and will become an integral part of the format in future versions.

3 Design Decisions

Based on the above requirements, AxF is designed in layers. Since a binary format is inevitable for efficiency and scalability, we first define a binary layer. On top of the binary layer we propose a basic structuring mechanism, which allows to define the hierarchical semantic structure of payload and metadata. The final layer consists of the explicit definition of the semantic structures.

We propose to use the HDF5 [HDF15] format as *binary base layer*. HDF5 defines a versatile data model, which can represent any kind of complex data objects and metadata. The format imposes no (practical) limits on the number or size of data objects in the file and is available for all major platforms.

The second layer implements a simple but powerful *property node tree* concept on top of HDF5 (cf. Figure 1). This concept is analog to a basic filesystem concept where nodes correspond to folders (structuring) and properties to files (payload). Properties support various typical datatypes like integral types, strings, and unbounded multi-dimensional data arrays.

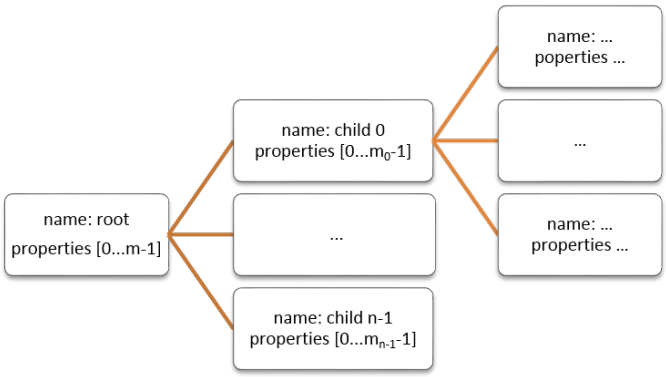


Figure 1: Property node tree concept

The third layer defines a *baseline* for valid AxF files that partly should be supported by all applications integrating AxF. Essentially it defines a set of node and property types and their semantics as described in the following section.

4 Baseline AxF

The basic semantic structure of an AxF file is related to the Color Exchange Format (CxF), which has been defined by X-Rite as a portable format for system and device independent exchange of color data and which has been published as ISO Standard "*ISO17972 Graphic technology – Colour data exchange format (CxF/X)*" [ISO15].

An exhaustive list of all node types and their semantics is beyond the scope of this document, but to convey the idea we will briefly sketch the root-level node collections as shown in Figure 2:

- The **DeviceSpecs** collection (*optional*) contains specification and calibration data for measurement instruments. It allows the specification of geometric and radiometric device properties.
- The **ColorSpecs** collection specifies arbitrary (linear) color spaces and spectral samplings. Since AxF stores floating point color values, non-linear color spaces are not part of the current specification.
- The **Materials** collection contains one or multiple *AxF materials*. Each AxF material node is the parent of multiple material-level node collections:
  - **Metadata** is a container for arbitrary, user-defined metadata for the given material (such as textual descriptions).
  - **Representations** is a set of one or multiple *AxF representations*. An AxF representation is a specific digital encoding of the full appearance of a material, suitable for *rendering* the material under ar-

bitrary viewing and lighting conditions. A representation may have been derived from measurements (by fitting a specific model to the measured data, such that the resulting representation approximates and generalizes the original measurements) and/or may have been edited manually.

  - **Measurements** (*optional, AxF version ≥ 1.8*) is a set of *AxF measurements*. An AxF measurement is a sampling of certain appearance properties of a physical material sample, captured by an instrument under well-defined conditions. A measurement provides a standardized set of values, suitable e.g. for comparison with measurements on other material samples in the context of *quality control*.
  - **Resources** is a collection of *AxF resources*, i.e. textures and uniform data, which constitutes the payload data for the representations and measurements of the given AxF material.

- The **Resources** collection at root level (*optional, AxF version ≥ 1.8*) stores *AxF resources* that are shared by multiple AxF materials. This allows to reduce storage space in particular when the AxF file contains a set of materials which are different variants derived from a common original material.

Note that a *Representation* node itself contains only the rendering *semantics* (such as the used BRDF / BTDF models), but no payload data. Instead it references the payload data stored in either the root-level or the respective material-level *Resources* collection.

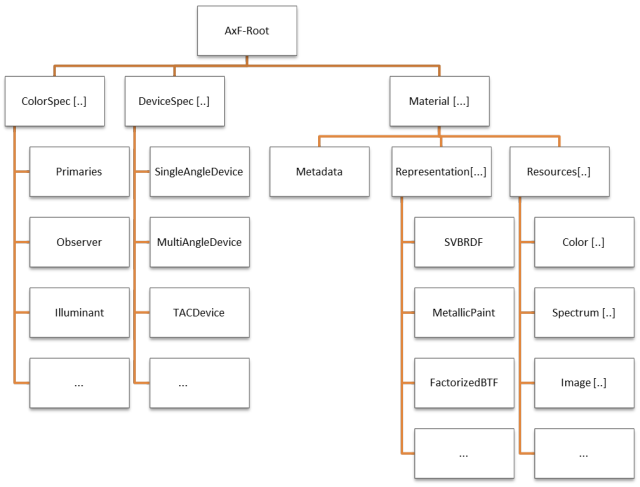


Figure 2: Excerpt from the definition of *Baseline AxF*. Collections are denoted by [..].

AxF allows storing an arbitrary number of different representations for a single material. It is, however, important to understand that each such representation is assumed to represent the very same (physical) material sample. In other words: Different representations of the same material approximate the same appearance, but potentially at different quality due to the use of different models.

The main motivation for AxF materials with multiple representations is backwards compatibility: By storing a representation with reduced feature set in addition to a representation with full feature set, third-party renderers that do not (yet) support the full feature set can use this compatibility representation as fallback.

If multiple representations are available for a material, a renderer should automatically choose the representation that is most suitable, more precisely the best representation that is compatible with the capabilities of the respective renderer. Since version 1.7.1, the AxF SDK provides convenience functions for exactly this purpose and also supports – to some extent – automatic conversions of the stored material representation(s) to match the capabilities of the given renderer (for the case that no suitable fallback representation is stored explicitly in the AxF file). See the AxF SDK documentation for details.

5 AxF Representation Classes

AxF supports different types of so-called *Representation Classes*, which refer to specific classes of related reflectance models and have been chosen to support a wide range of material types and target applications. For each class, the so-called *AxF compatibility profiles* specify a baseline of specific appearance models which should be implemented by rendering applications aiming to support the respective compatibility profiles. The basic idea is that at file creation, the user may enforce that representations for a certain compatibility profile will be included in the AxF file. If the client application is known to support all features of that compatibility profile, it is assured that it will be able to pick a representation that it can handle. See the AxF SDK documentation for details.

In addition to surface models, AxF supports since version 1.6 also volumetric materials (via representation class *Volumetric*). This way translucent materials can be represented by storing the general volumetric scattering and phase function parameters of the radiative transfer equation [Cha60].

In the following sections we will give brief overviews on each of the representation classes. The goal is to give a reader experienced in computer graphics and appearance modelling a good notion of the scope of the AxF baseline. We leave out most of the gritty details and refer to the AxF SDK documentation for details. If you need further background in general computer graphics topics like BRDFs, BTFs, texture mapping, or volumetric scattering, we advise you to consult the relevant literature. In the context of material appearance e.g. [DRS07] is a great starting point.

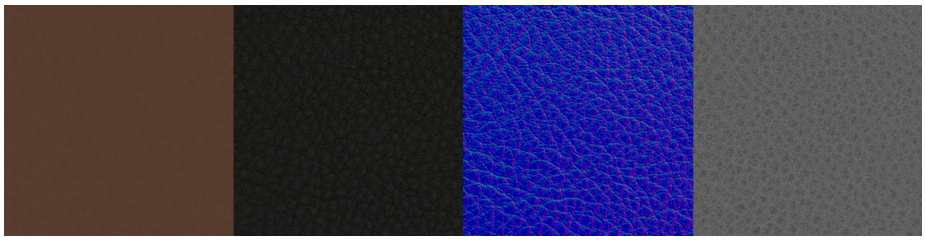


Figure 3: Parameter textures of an AxF 1.0 isotropic Ward SVBRDF for a measured leather material (from left to right): *diffuse color, specular color, normal, specular roughness.*

5.1 Spatially Varying BRDF

AxF defines a general SVBRDF representation based on the following evaluation of a BRDF consisting of a diffuse and a specular component with spatially varying parameters stored in textures (as shown in Figure 3):

$$f_r(\mathbf{x},\mathbf{i},\mathbf{o}) = f_d(\mathbf{i},\mathbf{o};T_{\rho_d}(\mathbf{x}),T_{\mathbf{N}}(\mathbf{x})) + f_s(\mathbf{i},\mathbf{o};T_{\rho_s}(\mathbf{x}),T_{\mathbf{N}}(\mathbf{x}),T_{\alpha}(\mathbf{x}),...)$$

The texture operator $T(\mathbf{x})$ represents a discrete 2D table of values of various types indexed by the texture coordinate $\mathbf{x}$. In particular, $T_{\rho_{\{d,s\}}}(\mathbf{x})$ contains color values (either trichromatic RGB values or spectra), which are stored for the diffuse and specular BRDF part, respectively. $T_{\mathbf{N}}(\mathbf{x})$ represents the *normal map*, which is used to vary the local coordinate system in order simulate a bumpy surface. $T_{\alpha}(\mathbf{x}),...$ denotes the texture operators for model-dependent specular parameters such as roughness.

In the following we assume that the incoming and outgoing directions $\mathbf{i}$, $\mathbf{o}$ are given in the local coordinate system that has been aligned to the normal $T_{\mathbf{N}}(\mathbf{x})$ from the normal map, and thus we omit the explicit normal parameter for f_d and f_s.

The only diffuse model f_d supported by AxF so far is the Lambert model:

$$f_d^{\text{Lambert}}(\mathbf{i},\mathbf{o};\rho_d) = \frac{\rho_d}{\pi}$$

where $\rho_d = T_{\rho_d}(\mathbf{x})$ is the diffuse color.

The default specular model f_s of AxF 1.0 is a variant of the Ward model as proposed by Geisler-Moroder [GMD10], which in the isotropic case is:

$$\begin{aligned} f_s^{\text{Ward}_{\text{GM2010}}}(\mathbf{i},\mathbf{o};\rho_s,\alpha) &= \rho_s \frac{1}{\pi\alpha^2} N_{\text{GM2010}}\, e^{\frac{1}{\alpha^2}\left(1-\frac{1}{h_z^2}\right)} \\ N_{\text{GM2010}} &= \frac{1}{4\,\langle\mathbf{i},\mathbf{h}\rangle^2 h_z^4} \end{aligned}$$

where $\mathbf{h}$ denotes the half-angle vector, and the model parameters are specular color $\rho_s = T_{\rho_s}(\mathbf{x})$ and roughness $\alpha = T_{\alpha}(\mathbf{x})$. Note that AxF also supports an anisotropic variant of this model, see the AxF SDK documentation for details.

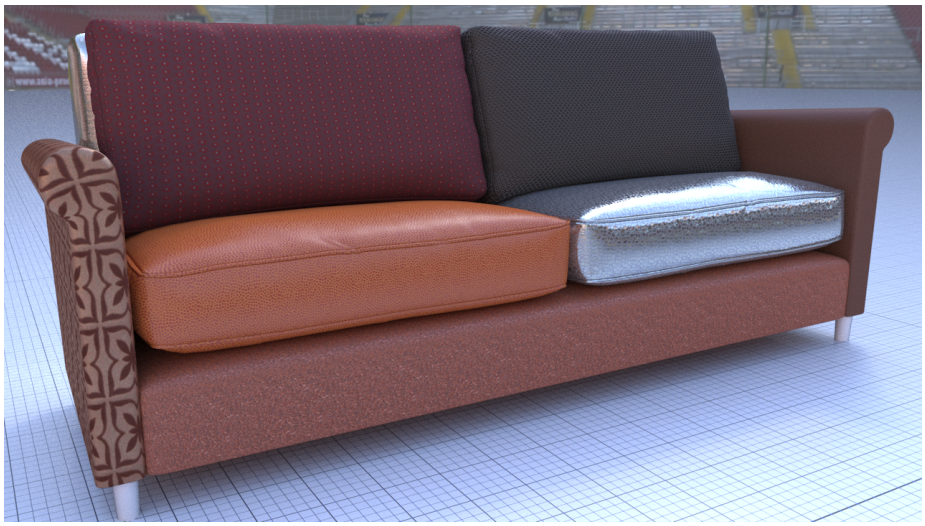


Figure 4: The *"AxF Sofa"* covered with several measured AxF SVBRDF materials like fabrics, plastics, leather, and metals rendered using Autodesk VRED. Some of the materials are taken from the MAM-2014 sample set [MAM14].

Beyond this basic model of AxF 1.0, several enhanced models are introduced with subsequent AxF versions: Starting with version 1.1, an extension of the Ward BRDF based on Schlick's Fresnel approximation is supported, of benefit for appearance representation for many textiles, leathers, and even plastics. Version 1.5 adds support for the increasingly popular GGX BRDF model [WMLT07], that is becoming sort of an industry-standard for realtime visualization. Version 1.7 introduces an *Energy Preserving SVBRDF* model based on GGX, which is compatible to the Dassault Systèmes Enterprise PBR Shading Model (DSPBR) [DS19]. Details on all these BRDF models are given in the AxF SDK documentation.

For best results, third-party renderers should optimally support all three BRDF models (Ward, GGX, and the Energy Preserving GGX variant) natively. Alternatively, if a third-party application supports only one of the two GGX-based models, the AxF SDK can convert between GGX-based SVBRDF representations and Energy Preserving SVBRDF representations while preserving the appearance as well as possible.

Figure 4 shows a rendering using several examples of measured SVBRDFs, represented using AxF's SVBRDF model.

In the following paragraphs, further SVBRDF extensions are described: coated materials (via the clear coat extension), alpha mapping, colored transparency, and displacement mapping.

Coated Materials Many real-world materials are made of multiple layers. In particular materials with a single semi-transparent coating layer on top of an

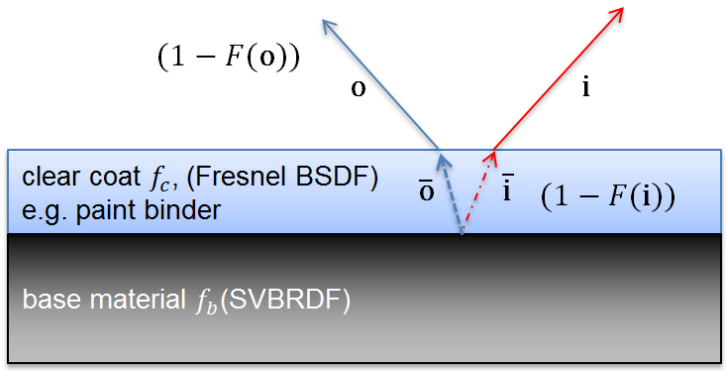


Figure 5: Base SVBRDF material covered by a transparent Fresnel BSDF layer.

opaque or semi-transparent base layer occur often, and these kind of materials can be modeled and rendered with acceptable additional effort compared to single-layer materials. In AxF versions 1.1 - 1.7, the *clear coat* extension for SVBRDF-based (or carpaint-based) representations was the only way to model such materials. It assumes an achromatic coating with ideal Fresnel reflection at the coating surface, as sketched in Figure 5. In version 1.8, AxF introduces *multi-layer materials* as a more generic concept to represent materials made of multiple semi-transparent layers, which supersedes the clear coat extension feature-wise and can also model colored coatings and rough reflections at the coating surface, see Section 5.5. Nonetheless, the legacy *clear coat* extension is still supported, and we restrict the discussion in this section to the capabilities of this legacy extension.

The clear coat is assumed to be completely transparent (no absorption and scattering despite at the layer boundaries), infinitesimal thin (refracted rays from the coating hit the lower layer at the same point), and solely defined by its (spatially varying) index of refraction and an (optional) normal map, which allows to model mesoscopic effects like the *Orange Peel* effect shown in Figure 6.

In addition to the refractive clear coat model introduced in AxF version 1.1, AxF also supports the following simplified clear coat models in order to ensure compatibility with a large number of existing rendering systems:

- Since version 1.3, AxF also supports a non-refractive clear coat model, which corresponds to the refractive model except that the base SVBRDF layer is evaluated for the original incoming and outgoing directions rather than for the corresponding refracted directions.

- Since version 1.7.1, AxF supports another variant of the non-refractive clear coat model with an alternative transmission term, based on a simple Schlick approximation of the Fresnel term with a fixed F_0 value (similar to [DS19]).

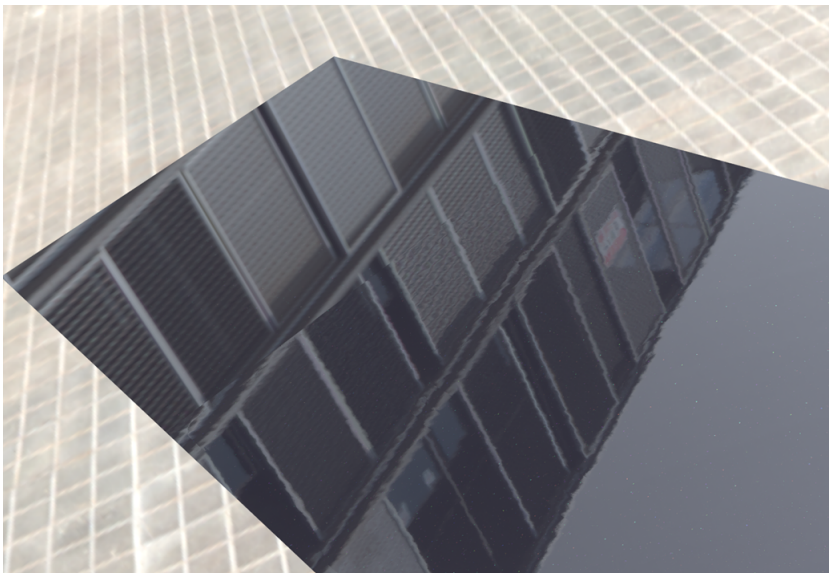


Figure 6: The Orange Peel effect shown for a measured paint material. For comparison the rear patch has a perfectly smooth coating.

See the AxF SDK documentation for a precise specification of all supported clear coat models.

If a third-party application does not support the AxF 1.1 refractive clear coat model, it may request the AxF SDK – since version 1.7.1 – to convert SVBRDF representations with refractive clear coat to one of the two simplified clear coat models listed above while preserving the appearance as well as possible. (This means that certain effects of refraction, such as the "widening" of specular lobes, will be "baked" into the base SVBRDF model.)

Alpha Mapping Starting with version 1.4, AxF supports also standard *alpha mapping* (see e.g. [PD84]), often referred to as *cut-out* or *opacity mapping*, for SVBRDFs. Containing spatially varying opacity values between 0 (completely transparent) and 1 (fully opaque), this map can be used to model simple transparency for an infinitesimal thin surface. The left image in Figure 7 shows a typical example how alpha mapping can be used to represent a perforated leather material.

Colored Transparency Starting with AxF version 1.8, SVBRDF representations can also be extended by a *non-refractive Dirac transmission model*. In comparison to alpha mapping, which is achromatic and does not consider Fresnel transmittance, this model enables to encode colored transparency. Assuming a material surface of infinitesimal thickness, the light passing through it, as determined by the Fresnel transmittance, is modulated by a transmission color.

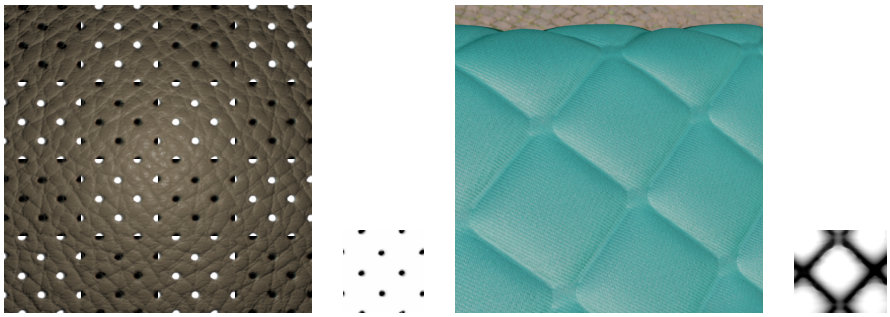


Figure 7: Left: Perforated leather overlaid onto a checkerboard pattern and the corresponding measured alpha map representing the holes. — Right: Textile rendering with measured displacement. Note the geometry's silhouette, which would have been a perfect line without displacement.

Note that the light direction is not altered by that, unlike by a refractive coating. With this model, for instance the appearance of thin, colored plastics or film can be described. In order to achieve an additional cut-out effect, e.g. for decals, it might be used in combination with alpha mapping.

Displacement Mapping Another convenient and widely used technique is *displacement mapping* (first introduced in [Coo84]), which AxF supports since version 1.4. A displacement map (a.k.a. height map) stores spatially varying height values, which are used by the rendering engine to locally offset and re-tesselate the geometry. The right part of Figure 7 shows an example of a textile material with displacement.

5.2 Measured BTF

The appearance model for representation class *FactorizedBTF* is based on a BTF compression technique as in [MMK03]. BTFs are naturally understood as a multi-dimensional table of reflectance measurements of a material sample.

Let us consider that each such measurement corresponds to sampling the general 8-dimensional reflectance field $R_V(\mathbf{x}_i, \omega_i; \mathbf{x}_o, \omega_o)$, which transfers incident light fields to their corresponding outgoing light fields parameterized on a bounding surface V. See Figure 8 for an illustration.

The discrete version of this operator is called the *light transport matrix* $\mathbf{R}$. It can be obtained by discretizing the domain of the incoming and outgoing light field:

$$\mathbf{L}_o = \mathbf{R}^{(K \times I)} \mathbf{L}_i$$

By assuming distant incoming lighting a typical parameterization of the incoming lighting is by directions of incoming light sources:

$$I = \{(\theta_{i,0}, \phi_{i,0}), (\theta_{i,1}, \phi_{i,1}), \ldots, (\theta_{i,l}, \phi_{i,l})\}$$

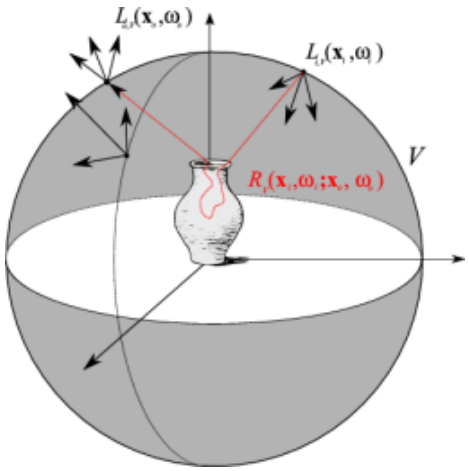


Figure 8: The reflectance field.

K describes the discretization of the outgoing light field and is usually given by the image or texture size $E = W \times H = \{1, 2, \ldots, w\} \times \{1, 2, \ldots, h\}$ and a set of outgoing directions $O = \{(\theta_{o,0}, \phi_{o,0}), (\theta_{o,1}, \phi_{o,1}), \ldots, (\theta_{o,v}, \phi_{o,v})\}$ similar as above and hence

$$K = E \times O = \{(1, 1, \theta_{o,0}, \phi_{o,0}), (1, 1, \theta_{o,1}, \phi_{o,1}), \ldots, (w, h, \theta_{o,v}, \phi_{o,v})\}.$$

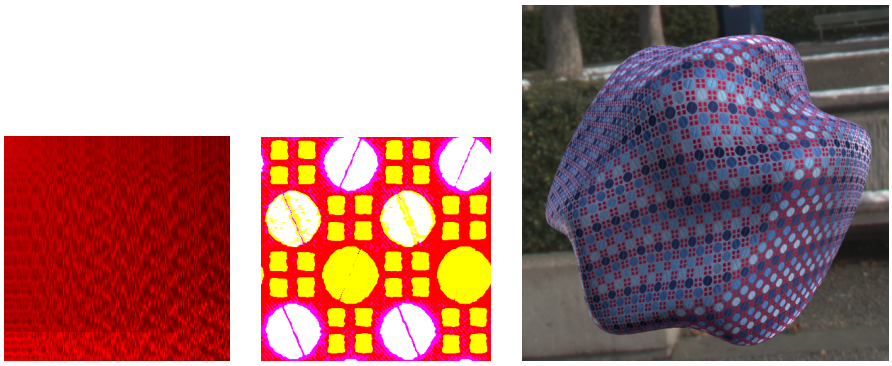
In this sense the light transport matrix $\mathbf{R}_{(K \times I)}$ as given above can be considered as a 2D table where each of the two dimensions enfolds a higher-dimensional index set: the 2D incoming directions are enfolded along the columns and the four dimensions of spatial position and outgoing direction are enfolded along the rows of the matrix.

The basic idea of matrix factorization-based appearance representations is now to apply matrix factorization techniques to the sampled data arranged into a 2D matrix and use these techniques for data compression.

Since the mathematical field of matrix factorization is huge, we'll leave it here with the most common definition of the Singular Value Decomposition, which states that each $m \times n$ matrix $\mathbf{A}$ of rank r can be decomposed into the product $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices and $\mathbf{S}$ is a diagonal matrix with $s_{ii} \geq s_{i+1,i+1} > 0$ for $1 \leq i < k$ and $s_{ii} = 0$ for $k + 1 \leq i \leq \min(m, n)$. The numbers s_{ii} are the nonnegative square roots of the eigenvalues of $\mathbf{A}\mathbf{A}^T$. The columns $\mathbf{u}_j$ of $\mathbf{U}$ are the eigenvectors of $\mathbf{A}\mathbf{A}^T$, and the columns $\mathbf{v}_j$ of $\mathbf{V}$ are the eigenvectors of $\mathbf{A}^T\mathbf{A}$.

The key insight for data compression is the well-known *Eckhart-Young Theorem*, which states that the matrix

$$\mathbf{A}_{r_0} := \sum_{j}^{r_0} \mathbf{u}_j s_{jj} \mathbf{v}_j^T$$



EigenBRDF EigenTexture Rendering (OpenGL)

Figure 9: The first three angular and spatial components of a factorized BTF encoded in a color mapped RGB image. The right image shows a rendering of the material.

is the best rank-r_0 approximation of $\mathbf{A}$ in the least-squares sense. Since the storage requirements for $\mathbf{A}_{r_0}$ are $O\left((m+n)r_0\right)$ compared to $O(mn)$ we have a compression ratio of $\left(\frac{r_0}{n}+\frac{r_0}{m}\right)$.

The 2D layout of 6D appearance data (which is in fact a BTF) as sketched above is not optimal for data compression since the compression ratio becomes optimal if n and m, i.e. the number of rows and columns, are roughly of the same size. We won't go too much into detail here, but a more balanced arrangement which puts the spatial dimension E in the rows and the angular dimension $I \times O$ in the columns has proven to be quite convenient in practice:

$$\mathbf{R}_{E\times(O\times I)} = \mathbf{U}_E \mathbf{S} \mathbf{V}^T_{I\times O}$$

Now the matrix $\mathbf{U}_E$ contains in its columns the so-called *EigenTextures*, and the columns of $\mathbf{V}_{I\times O}$ are the *EigenBRDFs*. Figure 9 shows an example for these factorized components by depicting the first three rows or columns, respectively, where the 4D EigenBRDFs have been enrolled into a 2D image.

5.3 Measured Carpaint

Since metallic paints, as they are typically used in the automotive industry, are not well represented by either SVBRDFs or standard BTFs, AxF supports a specialized representation for measured metallic paints in the spirit of the work of Rump et al. [RMS+08]. Again, we do not go too much into details here but refer the reader to the related scientific literature.

We'll leave it with the introduction of the main components such that general complexity and expressiveness of the model can be judged quickly by the experienced reader.

The model for representation class *CarPaint2* consists of three main components: first, a BRDF part which models the angular brightness variation of the

Figure 10: The left image shows a single slice from the flake texture table. The other images show renderings with different illumination conditions.

paint, second, an angular dependent color table which captures low-frequency color shift of modern effect paints, and finally, a BTF part which captures the visible flakes. The three components are sketched in the following paragraphs.

Brightness BRDF The BRDF is modeled using a multi-lobe Cook-Torrance:

$$f_r^{\text{brightness}}(\bar{\mathbf{i}}, \bar{\mathbf{o}}) = \frac{\rho_d}{\pi} + \sum_{k=1}^{K} \rho_{s_k} \frac{D(\bar{\mathbf{h}}; \alpha_k)\, F^{\text{Schlick}}(\bar{\mathbf{i}}, \bar{\mathbf{h}}; F_{0,k})\, G(\bar{\mathbf{i}}, \bar{\mathbf{o}}, \bar{\mathbf{h}})}{\pi\, \bar{i}_z\, \bar{o}_z}$$

where D is the Beckmann distribution, F^{Schlick} Schlick's Fresnel approximation, and G the Cook-Torrance geometry term [CT82]. $K = 3$ is the default number of lobes for fitting measurement data and can be considered the de-facto standard.

We use bars over the direction vectors in this section, to indicate that these directions are defined *below* an optional coating layer, which can be described via the *clear coat* extension similarly as for SVBRDF representations (see Section 5.1). E.g., in case of refractive clear coat, $\bar{\mathbf{i}}$ stands for the refracted incoming direction $\bar{\mathbf{i}} = \mathbf{refract}(-\mathbf{i}, \mathbf{n}_s^{\text{coating}}, \frac{\eta_{\text{air}}}{\eta^+})$, where $\mathbf{n}_s^{\text{coating}}$ denotes the normal of the clear coat and η^+ its index of refraction. $\bar{\mathbf{o}}$ is defined accordingly.

Color Table The BRDF brightness term is modulated by a 2D color table $\chi(\theta_{\bar{\mathbf{h}}}, \theta_{\bar{\mathbf{i}}})$ parameterized by the angle between half vector and normal $\theta_{\bar{\mathbf{h}}} = \arccos(\bar{h}_z)$ and the angle between half vector and incoming direction $\theta_{\bar{\mathbf{i}}} = \arccos(\langle \bar{\mathbf{h}}, \bar{\mathbf{i}} \rangle)$. In classical microfacet modeling $\theta_{\bar{\mathbf{h}}}$ is interpreted as the angle between the overall surface normal and the microfacet normal. In the case of a metallic paint we identify microfacets with metallic flakes, hence $\theta_{\bar{\mathbf{h}}}$ can be interpreted as the angle between the overall surface normal and the normal of a flake (modeled as a perfect mirror). Correspondingly, $\theta_{\bar{\mathbf{i}}}$ is the angle between the refracted illumination direction and the flake normal.

Flake BTF Textures The spatially varying part, i.e. the visible flakes, is also parameterized based on $\theta_{\bar{\mathbf{h}}}$ and $\theta_{\bar{\mathbf{i}}}$, but now each entry does not contain a single

color value but a complete flake texture slice as shown on the left of Figure 10, which actually leads to a 4D table $\Xi(\mathbf{x}, \theta_{\bar{\mathbf{h}}}, \theta_{\bar{\mathbf{i}}})$. In practice the sampling of the angular space can actually be quite sparse, which results in overall texture sizes of 20-50 Megabytes.

Combined Model The complete spatially varying model for the carpaint base material *below* clear coat reads as follows:

$$f_r(\mathbf{x}, \bar{\mathbf{i}}, \bar{\mathbf{o}}) = \chi(\theta_{\bar{\mathbf{h}}}, \theta_{\bar{\mathbf{i}}})\, f_r^{\text{brightness}}(\bar{\mathbf{i}}, \bar{\mathbf{o}}) + \Xi(\mathbf{x}, \theta_{\bar{\mathbf{h}}}, \theta_{\bar{\mathbf{i}}})$$

For more details on how to evaluate the combined model including the clear coat, see the AxF SDK documentation.

5.4 Volumetric Scattering

Due to the advent and success of using Monte-Carlo integration for physically correct rendering, more and more rendering software packages are capable of solving the full radiative transfer equation (RTE) [Cha60], which describes the full light transport in participating media. Again, this document shall not give a full-fledged introduction into rendering and modeling of volumetric scattering, but instead we refer the interested reader to publications like [NGHJ18], which give a great overview on the topic.

It is important to note, though, that for representing the scattering properties of homogeneous translucent materials the generic physical parameters derived from the RTE are nowadays readily supported by many production rendering packages:

$\sigma_a(\lambda)$**,** $\sigma_s(\lambda)$ – spectral absorption and scattering coefficients (unit $\frac{1}{mm}$), depending on wavelength λ;

$p(\omega_i, \omega_o)$ – phase function, which describes the contribution of in-scattering light.

AxF uses the well-known Henyey-Greenstein phase function [HG41], which only depends on the angle θ between in-scattered and the outgoing direction and the scattering parameter $g \in [-1, 1]$:

$$p^{\text{HG}}(\theta; g) = \frac{1}{4\pi} \frac{1 - g^2}{(1 + g^2 - 2g\cos\theta)^{3/2}}$$

AxF does not specify specialized models or approximations for volumetric scattering, but stores the generic physical parameters: σ_a, σ_s, g. Thereby it leaves it to the rendering software to either fully solve the RTE for the given parameters using Monte-Carlo integration or to employ a specific approximation whose parameters can be derived from the original physical parameters.

Figure 11 shows an example where such a description is used to model a translucent plastic material. The parameters can be derived from measurements of the material, e.g. over a black and white backing.

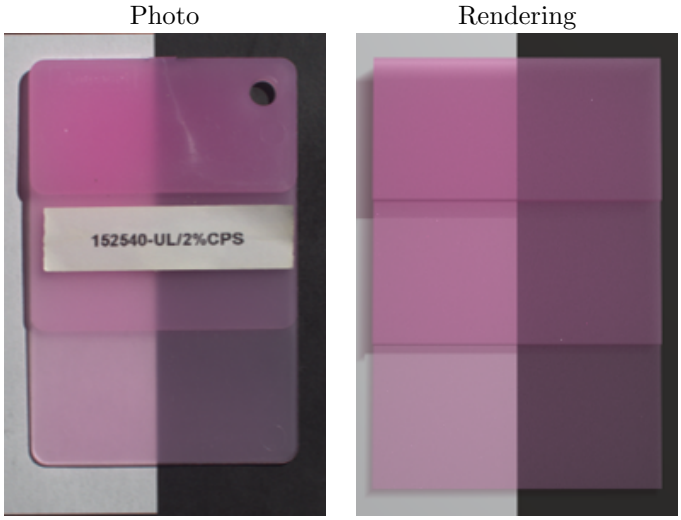


Figure 11: The parameters used for rendering the 3-step plastic chip on the right (homogeneous spectral absorption and scattering coefficients, isotropic phase function ($g = 0$)), have been derived from the photo on the left. Note the blurring of the black/white line in the photo which is a result of the non-smooth surface of the real plastic chip. The model on the right assumes a perfectly smooth interface.

In case of a stand-alone volumetric AxF representation, the boundary of the volume is restricted to an ideally flat Fresnel surface. Starting with AxF version 1.8, a volumetric representation can also be used as base layer within a multi-layer material (see Section 5.5), in which case the boundary of the volume can be specified by an *SVBSDF* representation with GGX-based BRDF and BTDF models [WMLT07] and custom parameter textures, including a normal map to account for structures on mesoscopic scale and a roughness map to account for microscopic surface roughness.

5.5 Multi-Layer Materials

Since version 1.8, AxF has a generic concept to represent materials made of multiple semi-transparent layers. Such *multi-layer materials* contain a separate sub-representation for each layer, each of which encodes the appearance properties of that particular layer. This is useful

- to model coated materials;
- for volumetric materials to separate the appearance properties of the material surface from those of the material volume.

While this concept would support an arbitrary number of layers, for the current AxF version, multi-layer materials may consist of at most two layers,

in order to simplify their implementation. This restriction may be lifted in a future AxF version.

The new concept for multi-layer materials improves upon what was possible in previous AxF versions via the clear coat extension sketched in Section 5.1. Clear coat appearance effects that could already be achieved previously are:

Stand-alone Representation	Clear Coat (AxF $\geq$ 1.1)		Appearance Effects
	reflection	transmission	
SVBRDF- or Carpaint-based	Dirac	Dirac	Smooth clear coat Clear coat with orange peel
Volumetric	Dirac	Dirac	Smooth surface

AxF 1.8 introduces multi-layer representations supporting the following combinations of base layers and upper layers, where the upper layer is specified by an *SVBSDF* sub-representation, for extended appearance effects:

Multi-layer Representation			New Appearance Effects (AxF $\geq$ 1.8)
Base Layer	Upper Layer		
	BRDF	BTDF	
SVBRDF-based	GGX	Dirac	Rough coating (due to GGX BRDF) Colored coating (due to transmission color)
Volumetric	GGX	GGX	Mesostructure (via SVBSDF normal map) Rough surface (rough BRDF and BTDF) Hazy transmission (rough BTDF only) Colored surface (BTDF transmission color)

Details on multi-layer materials can be found in the SDK documentation.

6 Integration and Performance

For the integration of AxF into third-party applications, X-Rite provides a C/C++ based SDK, which is made available on request under a royalty-free license (contact one of the authors or LicensingTeam@XRITE.com directly).

For each AxF material, the SDK allows to decode the metadata, the AxF representations, and since version 1.8 also the AxF measurements (when stored in the file), along with their associated payload data (AxF resources). The SDK is fully documented and comes with example code.

6.1 Decoding AxF Representations

While the AxF SDK contains an easy-to-use implementation of CPU-based rendering (`axf::decoding::CPUDecoder`), this implementation is not aggressively optimized, but rather should be considered as reference implementation, which should be used primarily for reference and testing purposes. In particular, we do not recommend to use this way of decoding in performance critical

production environments. Instead, we encourage client applications to implement the AxF representations, which we sketched in Section 5, internally and to use the `axf::decoding::TextureDecoder` interface for retrieval of the appearance model parameters (i.e. textures). The full technical details required for implementing a particular representation class can be found in the SDK documentation.

6.2 Handling Colors and Spectra

AxF supports storing color resources in monochromatic, trichromatic, pseudo-spectral, and full spectral color spaces. To facilitate that, each color resource is associated with a certain *ColorSpec* node (see the *ColorSpecs* collection sketched in Section 4), which specifies the corresponding color space precisely.

In any case, AxF uses a *linear* color space (i.e. a linear subspace of the spectral space) to store a color resource. Please note that we consider a color (from a color resource) to correspond to a reflectance spectrum (i.e. *spectral reflectance* in physical terms) or to a projection thereof into the respective linear subspace that the *ColorSpec* represents. A unity reflectance spectrum represents a neutral material, i.e. one for which the spectrum of the reflected light corresponds exactly to the spectrum of the incoming light.

Currently, the information from the *ColorSpec* node cannot be retrieved explicitly via the SDK. Instead the client application defines a target color space into which all retrieved color resources will be converted (based on the AxF-internal *ColorSpec* node) before returning them to the client. Details can be found in the SDK documentation.

Spectral Rendering Starting with version 1.6, the AxF SDK supports querying of spectral color resources for representations that actually store spectral data. Since AxF version 1.6, this is supported for *Volumetric* and *SVBRDF*-based representations and since AxF version 1.8 also for the *CarPaint2* representation class and for multi-layer representations.

If a representation does not store full spectral data, it has to be decoded in a trichromatic target color space. Nonetheless, in this case it is possible to query a "spectralization transformation", which is a matrix that can be used to transform a trichromatic color from the selected target color space to a corresponding spectrum (for any given spectral sampling).

For details on both techniques for spectral rendering, we refer to the SDK documentation.

6.3 Performance and Memory Consumption

Measured appearance has often been considered as being more resource demanding than *tweaked* materials based on manually edited texture images. This has been true for BTF-based material representations which rely on dense sampling of both the spatial and the angular dimension of appearance, but it does not apply to measured materials based on the SVBRDF representation. The reason

is simply that even hand-tweaked materials are essentially SVBRDFs: every texel stores parameters of a certain BRDF model. As a result most of the optimization techniques which are applied to hand-tweaked materials can also be applied to measured materials with a similar effect.

While the numbers given in the following for illustrating AxF storage requirements are based on SVBRDFs, most of the techniques could also be applied to BTFs and Carpaint representations.

A typical AxF 1.1 anisotropic SVBRDF resulting from a TAC7 measurement consists of the following parameters:

- Diffuse Color, $|T_{c_d}(\mathbf{x})| = 3$ channels
- Normal Map, $|T_N(\mathbf{x})| = 3$ channels
- Specular Color, $|T_{c_s}(\mathbf{x})| = 3$ channels
- Anisotropic Roughness, $|T_{p_s}(\mathbf{x})| = 3$ channels[1]
- Fresnel, $|T_{F_0}(\mathbf{x})| = 1$ channel
- Material dependent:
  - Clearcoat Layer IOR, 1 channel
  - Clearcoat Layer Normal Map, 3 channels

Without clear coat, such an SVBRDF has 13 channels and given a spatial resolution of 1024×1024 pixels this leads to $13 \times 1024 \times 1024 \times 4 = 52\,\mathbf{MB}$ (additionally 16 MB for clear coat) of uncompressed data in 32-bit precision.

Note that the storage requirement for the AxF file itself is in general considerably lower, since AxF applies common lossless compression techniques for storing the data. However, this kind of compression is not suitable for random access to the data, thus the data is decompressed transparently when reading it from the AxF file. Therefore, we only consider the uncompressed in-memory storage requirements of AxF materials in the following.

Using roughly 50 MB of storage per material can impose a restriction for the use of AxF materials in real-time applications and on low-performance systems, for instance at the point-of-sale. Fortunately, the in-memory storage requirements can be reduced in numerous ways for most practical applications:

1. Before measurement: By selecting the optimal model for the material and use case. This may correspond simply to using less BRDF parameters (channels) like in the isotropic vs. anisotropic case. Or to the reduction of spatial variation by assuming that some parameters are constant over the sample, for instance by using the *global F0* option, which assumes a constant Fresnel parameter, or the *dielectric material* option, which assumes colorless specular reflections and thereby eliminates chromaticity variation in the specular color texture.

[1] In AxF, this is in general represented by two resources, a 2-channel *Roughness* map (representing roughness values along the two principal axes) and a single-channel *Anisotropic Rotation* map (containing rotation angles).

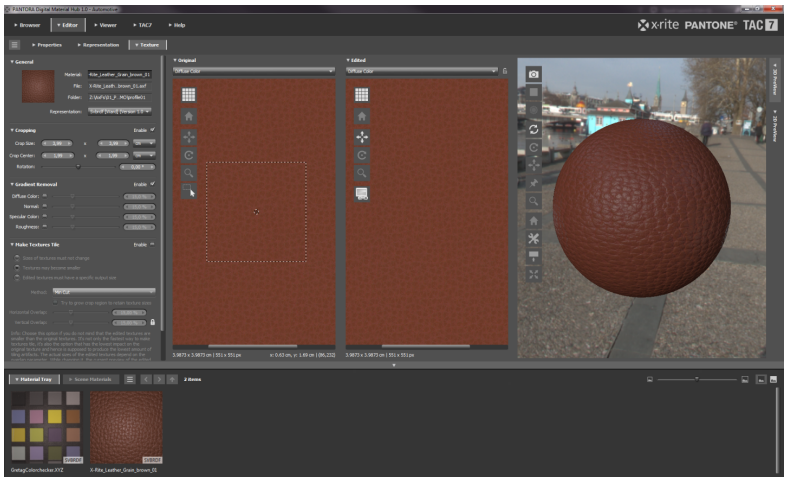


Figure 12: AxF Editor integrated into X-Rite's Digital Material Hub.

2. During data preparation: By using the AxF Editor of X-Rite's **Digital Material Hub** (cf. Figure 12) to reduce the spatial variation in certain BRDF parameter maps and by sharing AxF resources between materials where appropriate (shown in Figure 13).

   This approach is recommended in particular if the material has a dominant base color or BRDF. Think of a material like single-colored molded plastic, which has an almost constant BRDF but high variation in the normal and/or displacement map.

3. While requesting data from the SDK: By choosing an appropriate data type (for instance half-precision floating point compared to 32-bit precision) and the required (maximum) spatial resolution, in which to store textures in memory.

4. By further reducing the memory footprint of texture resources in the client application, for instance by applying certain texture compression techniques or by supporting the sharing of resources as represented in the AxF file.

By applying and combining theses techniques, storage reductions from 50%-80% compared to the original 32-bit precision can easily be achieved without losing significant visual fidelity. In our example this might result in in-memory storage requirements of only 10 MB.

Again, further details can be found in the SDK documentation.

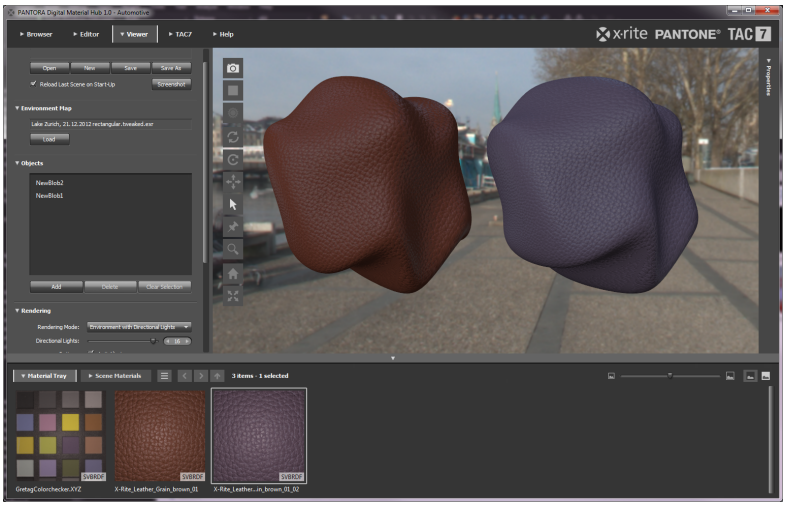


Figure 13: Two leather materials with different diffuse base color, but sharing the same diffuse and specular intensity map and normal map resources.

7 Wrap-up

In this paper we introduced X-Rite's proposal for an appearance exchange format, which handles the low-level details like efficient storage and access of payload and metadata, how colors are specified etc. Although implementations into commercial software like NVidia's MDL [MDL15] or Autodesk VRED are available (cf. Figure 4 for an example), the definition of the top-layer semantic (Baseline AxF) and especially its material representations are not meant to be complete.

Further definition of the format needs to focus on additional representations which extent the gamut of materials that can be well represented in a portable, i.e. exchangeable way. Obviously, this has to be done in close collaboration with both the software vendors, who decide which kind of material models they will support and integrate in their software, and the users, who define the types of materials that need to be digitized as accurate as possible in order to increase the quality and credibility of the digital design process.

Then goal would be to build a consortium of hardware and software vendors, members of the research community, and users of measured material appearance to define future requirements and demands for additional AxF features and especially material representations. If a broad support for an agreed baseline of representations can be achieved, this can be the first step towards a standardization of digital material appearance making efficient exchange and archival of appearance possible for industry and research.

A Document History

Version 1.8, January 2021 Added description of the new *Measurements* and root-level *Resources* collections. Added paragraph on the new *Colored Transparency* feature for SVBRDF-based representations. Added section on the new *Multi-Layer Materials* and referred to it in the sections on coated SVBRDF-based materials and volumetric representations. Added note on the extended support of representations with spectral data.
Corrected the isotropic Ward formula in Section 5.1. Additional minor changes in Sections 4 - 6 for clarification (not color coded).
Removed the "Appendix on BRDF models" from this white paper. Please refer to the AxF SDK documentation instead, which contains an extensive description of all BRDF and BTDF models supported by this AxF version.

Version 1.7.1, February 2020 Added information about the new simplified clear coat variant and about the new AxF SDK feature for converting representations to match the capabilities of a given third-party renderer that supports a subset of AxF compatibility profiles only.
Removed outdated information such as old integration guidelines, while referring to the AxF SDK documentation instead, which contains an updated implementation guide and was extended by a lot of detail information in AxF SDK version 1.7.1.

Version 1.7, April 2019 Added section on the new Energy Preserving GGX BRDF model.

Version 1.6, March 2019 This version introduced a lot of changes:

- Started color coding changes between consecutive versions of the paper and added this *Document History* section.
- Corrected Geisler-Moroder Term - the old version was the one for the *non-normalized* half-angle vector.
- Added new *Volumetric* representation class for volume scattering.
- Extended paragraph on spectral rendering.
- Moved updated and extended part on BRDF models to an Appendix.
- Minor updates in Wrap-up section.

Version 1.5, March 2018 Added note on support of the GGX Microfacet model as part of the SVBRDF representation.

Version 1.4, February 2017 Added paragraphs on alpha and displacement mapping.

Version 1.3, July 2016 Added a paragraph on refractive and non-refractive coatings.

Version 1.2, April 2016 Added documentation for new measured Carpaint representation. Added a section on the integration of the AxF SDK and performance considerations.

Version 1.1, January 2016 First public version of this white paper.

References

[Cha60] S. Chandrasekhar. *Radiative Transfer*. Dover Books on Intermediate and Advanced Mathematics. Dover Publications, 1960.

[Coo84] Robert L. Cook. Shade trees. *SIGGRAPH Comput. Graph.*, 18(3):223–231, January 1984.

[CT82] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Trans. Graph.*, 1(1):7–24, January 1982.

[DRS07] Julie Dorsey, Holly Rushmeier, and François X. Sillion. *Digital Modeling of Material Appearance*. Morgan Kaufmann / Elsevier, December 2007.

[DS19] Dassault Systèmes. Enterprise PBR shading model, spec. version 2019x / 2020x, 2019. https://github.com/DassaultSystemes-Technology/EnterprisePBRShadingModel.

[GMD10] David Geisler-Moroder and Arne Dür. A new Ward BRDF model with bounded albedo. *Computer Graphics Forum*, 29(4):1391–1398, 2010.

[HDF15] Hierarchical Data Format, version 5, 1997-2015. http://www.hdfgroup.org/HDF5/.

[HG41] K. G. Henyey and J. L. Greenstein. Diffuse radiation in the galaxy. *Astrophysical Journal*, 93:70–83, January 1941.

[ISO15] ISO 17972-1:2015. Graphic technology – Colour data exchange format – Part 1: Relationship to CxF3 (CxF/X). 2015. https://www.iso.org/standard/61500.html.

[MAM14] MAM-2014 sample set, 2014. https://sites.google.com/site/mam2014samples/.

[MDL15] NVidia MDL - Material Definition Language, 2015. http://www.nvidia.com/object/material-definition-language.html.

[MMK03] Gero Müller, Jan Meseth, and Reinhard Klein. Compression and real-time rendering of measured BTFs using local PCA. In *Vision, Modeling and Visualisation 2003*, pages 271–280, November 2003.

[NGHJ18] Jan Novák, Iliyan Georgiev, Johannes Hanika, and Wojciech Jarosz. Monte Carlo methods for volumetric light transport simulation. *Computer Graphics Forum (Proceedings of Eurographics - State of the Art Reports)*, 37(2), May 2018.

[PD84] Thomas Porter and Tom Duff. Compositing digital images. *SIGGRAPH Comput. Graph.*, 18(3):253–259, January 1984.

[RMS+08] Martin Rump, Gero Müller, Ralf Sarlette, Dirk Koch, and Reinhard Klein. Photo-realistic rendering of metallic car paint from image-based measurements. *Computer Graphics Forum*, 27(2):527–536, April 2008.

[WMLT07] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR'07, pages 195–206. Eurographics Association, 2007.

