Workshop on Material Appearance Modeling (2015)
H. Rushmeier and R. Klein (Editors)

The Material Definition Language

L. Kettner, M. Raab, D. Seibert, J. Jordan, and A. Keller[†]

NVIDIA

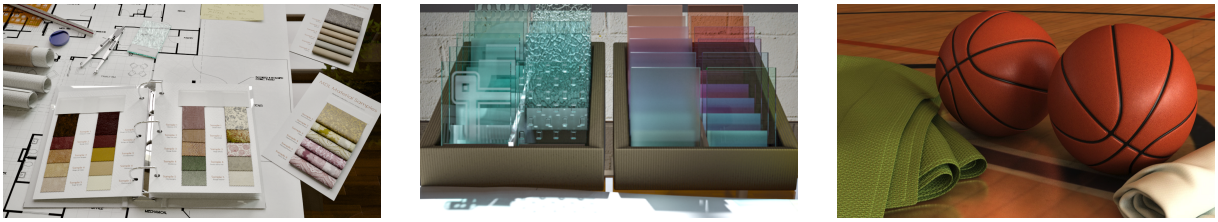

Figure 1: *A variety of materials created with the Material Definition Language (MDL).*

Abstract

We introduce the physically-based Material Definition Language (MDL). Based on the principle of strictly separating material definition and rendering algorithms, each MDL material is applicable across different rendering paradigms ranging from realtime over interactive solutions to advanced light transport simulation.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1. Introduction

The paradigm shift towards physically-based rendering (PBR) has happened throughout the whole rendering industry, from VFX to design and games, over the course of the last years. Being physically-based is a prerequisite for almost all modern rendering algorithms [Vea97] and has received great attention with the 2013 Technical Achievement Awards and the seminal book with the very same title [PH11]. As a consequence, material appearance modeling evolved from conventional shading languages to the use of Bidirectional Scattering Distribution Functions (BSDFs) and related concepts in order to separate the concerns of rendering algorithms and material definition.

For its various benefits, physically-based workflows have been adopted by different classes of rendering algorithms such as for example ray-tracers and rasterizers. Hence, there is a need for a physically-based material definition that works across different rendering algorithms.

In this position paper, we introduce the NVIDIA Material Definition Language (MDL) and discuss its expressiveness and advantages as well as its limitations. The language specification [NVI15] is publicly available and enables others to adopt MDL.

e-mail:lkettner|mraab|dseibert|jjordan|akeller@nvidia.com

2. Yet another Material Language?

Modern renderers typically offer a C/C++ API for material development, such as PBRT, Arnold, or the RIS framework of RenderMan 19 do. MetaSL [men10] successfully introduced programmable BRDFs (Bidirectional Reflectance Distribution Functions) with version 1.2, but kept its legacy of a conventional programming language and later has been discontinued. Besides MDL, the only domain specific language to model physically-based materials is the Open Shading Language (OSL) [Son14].

OSL innovated with the representation of distribution functions by *closures* in an otherwise RenderMan-Shading-Language-like language. Conventional functions are used to compute a closure from standard components and an algebra of operators. In contrast, MDL keeps the definition of the distribution functions and their layering structure strictly separated from the conventional functions. An MDL compiler thus easily understands the structure of the distribution functions of a material at compile time. While OSL offers a direct trace call and additional data communication channels to serve the needs of VFX productions, parallel architectures are targeted more efficiently by MDL, in particular,

DOI: 10.2312/mam.20151195

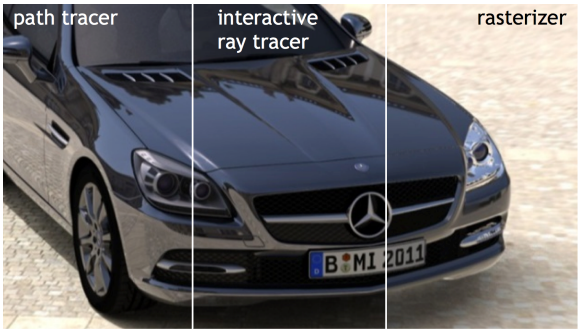

Figure 2: *The same car paint material across three rendering algorithms.*

due to the side-effect free functions, the read-only state, less data dependencies at runtime, and easy access to the material structure for optimizations.

In summary, MDL addresses the requirements of modern light transport simulation algorithms better than traditional shading languages and provides the benefits of a domain specific language over the common C/C++ API solutions. In comparison to OSL, it has a clearer separation of concerns into functions, material definition, and rendering algorithms, and its design contains choices in favor of highly performant renderers especially on massively parallel architectures including GPUs.

3. One Material across many Rendering Algorithms

MDL is a small domain-specific language to define *materials* and *functions*, which can be organized in *modules* and *packages* to create flexible, custom-built material catalogs. Materials are defined in a declarative style; they define what to compute for a material and not how to compute it, which is the key to make one MDL material usable across many rendering algorithms while targeting the same material appearance (see Fig. 2), of course within the limitations of the selected light transport simulation algorithm. As a consequence, renderer specific parts commonly found in other shading languages, such as light loops or ray tracing calls, are not available in MDL. The function definitions in MDL are written in a procedural programming style. Their use is limited to computing material parameters, which allows for optimizing rendering algorithms independent of the material definition.

3.1. Declarative Definition of Materials

Modeling with physically-based entities, a material's characteristics are defined by the means of Bidirectional Scattering Distribution Functions (BSDF), Emission Distribution Functions (EDF), and Volume Distribution Functions (VDF). They define the interface between the material definition and the rendering algorithm.

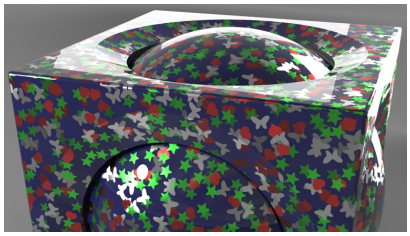

Figure 3: *Procedural flakes of varying geometric shapes.*

Unlike other monolithic or strictly layered solutions, in MDL these distribution functions can be composed in a flexible and powerful way from simple building blocks. For example, a BSDF can be one of a set of simple elemental BSDFs, or it can be constructed using a directed acyclic graph of operators and modifiers where only the leaves are elemental BSDFs. Combination operators include simple weight-driven mixing, (Fresnel-)layering, and color curve modifiers, often with configurable custom normal mapping options. The graph of operators is equivalent to a declarative expression and does not allow for control flow constructs that would prevent compile-time understanding of the material's structure, like for example variable-dependent loops. Similarly, EDFs and VDFs can be built from elemental distribution functions and operators.

3.2. Procedural Language for Function Definitions

Since MDL has been designed for massively parallel architectures like GPUs, we only allow for the definition of pure (side-effect-free) functions that have access to a read-only rendering state.

The C-like procedural programming language offers flexibility to material artists to program material input parameters or additional material properties, such as displacement mapping, normal mapping, and cutouts. The renderer state available in MDL enables material authors to program everything from simple texture blending and projection mapping pipelines to noise-driven normal mapping. See Fig. 3 for an example of procedurally generated flakes.

4. Expressive Power for Material Appearance Modeling

In the following, we discuss aspects of the expressive power of MDL and its limitations, and in particular its support of measured materials.

4.1. Synthesizing Materials

Seen from the perspective of a conventional shading language, the declarative material model of MDL certainly

Figure 6: *Five textures comprising an SVBRDF measurement [AWL13] rendered on a sun-lit plane.*

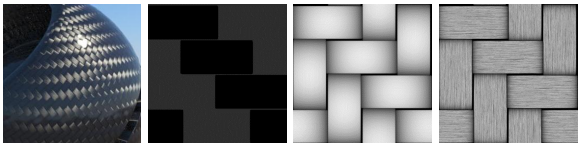

Figure 4: *Synthesizing a carbon fiber reinforced polymer: The material uses three maps with a simple layered BRDF foundation, which are (from the second image on the left to the far right): An anisotropy map to model reflections of the fibers embedded in the resin. A bump map for the fiber bundles, while the coating itself appears flat. A reflectivity map for the fiber self shadowing. A similar material structure can be used to create lively woods.*

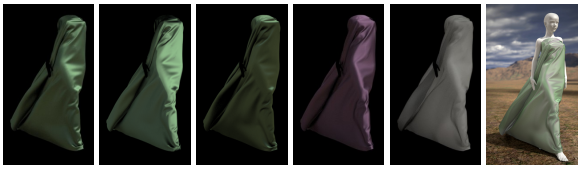

Figure 5: *Cloth features complex anisotropic behavior determined by threads running orthogonally to each other (warp and weft). Threads are bent which can create secondary highlights and warp and weft might be of different color, creating complex color variation. From left to right: a) The basis for warp (Anisotropic glossy BSDF). b) Adding a tilted glossy lobe to simulate thread bending. c) Adding another tilted glossy lobe to simulate thread bending. d) Weft: 90 degrees rotated and different color (warp and weft together create the complex reflective behavior of cloth). e) Translucency of the fabric. f) Final result.*

could be perceived as a kind of restrictive fixed-function model. For example, the elemental distribution functions and their operators are fixed indeed; they are part of the interface between the renderer and the material definition. However, the algebra of operators on distribution functions is very powerful and flexible. The forthcoming MDL Handbook (see www.mdlhandbook.com) provides an idea of the vast diversity of materials that can be represented in a straightforward and natural way in MDL, such as materials with transparent clear coats, varnishes, plastics, metals, glas, translu-

Figure 7: *Three materials of the sample material set [Rus14] measured with the Radiant Zemax Imaging Sphere device: Basketball, green felt, and gold silver.*

cent materials (see Fig. 1), resins with embedded fibers (see Fig. 4), cloth (see Fig. 5), or procedural metallic paints.

Such materials efficiently can be evaluated and sampled in a renderer, even under realtime constraints. Obviously, the cost grows with the complexity of the material, for example, when combining many BSDFs or using complex texture function networks and procedures. MDL's support for material measurements provides an efficient alternative, especially if such complex materials are required.

4.2. Measured Materials

Material measurements are a simple way to create believable materials and a necessity for quantitative results in predictive rendering. Measurement devices for either purpose exist and create a Spatially Varying BRDF (SVBRDF) representation, i.e., a set of textures controlling an analytic BRDF model representing their measurement. Typically, these are easy to represent in MDL, such as for a recent research system to capture believable materials [AWL13] (see Fig. 6), or the base profile in the AxF format from the Bidirectional Texture Function (BTF) scanning technology by X-Rite targeting predictive workflows [Sch14] (see Fig. 1 on the right, where the green cloth and the basketball materials are from the sample material set [Rus14]). Currently, a complete BTF cannot be represented in MDL.

Additional components in MDL support measured reflectance curves, for example, for realistic glass, and measured isotropic BSDFs. We used the latter representation with the Radiant Zemax Imaging Sphere device. The device does not provide spatial variation as it averages a material BRDF over a small area of a porthole, which may provide a

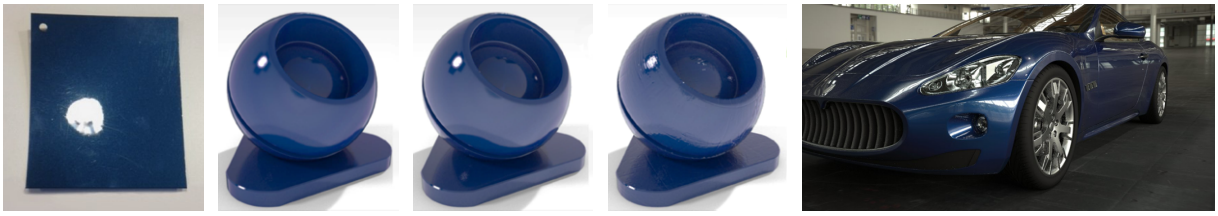

Figure 8: *From left to right: a) A physical car-paint sample, b) its measurement, c) with an added clear coat (a Fresnel-layered specular BRDF), d) a normal map for scratches in the paint, and e) a rendering of the final material on a car model.*

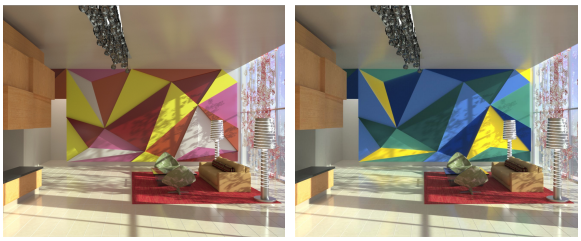

Figure 9: *Light path expressions are used to render the tiles on the back wall into individual images and composite tinted versions to make the final images including tinted secondary effects on floor and ceiling. For example, a final image (left) can be adjusted in post processing to a high-quality recolored image (right).*

good result for a far field appearance (see Fig. 7 for results using the sample material set [Rus14]).

The component model of MDL integrates measurements seamlessly with all other components, for example, allowing for their flexible use in a creative design concept phase, where an artist develops new materials by modifying existing ones by adding a different surface finish with a bump map (see Fig. 8(d)). For a lack of efficient compression, the measured BSDF representation is currently limited to isotropic BSDFs.

A typical limitation of measurement devices is their insufficient capture of reflections at grazing angles. In MDL, additional knowledge about grazing-angle reflectivity of a material, for example, through a different measurement or human knowledge, can be used to amend the measurement with a suitable Fresnel-layered specular reflection (see Fig. 8(c)).

5. Conclusion

Physically-based materials work reliably across different lighting situations and allow for the creation of future-proof material catalogs. Compared to conventional shader programming, look development is dramatically simplified and material specific programming can be well separated from rendering specific code.

Instead of compromising on the correctness of the material model, the demand for artistic control over the images generated by physically-based renderers can be met with the orthogonal render concept of light path expressions (LPEs). In particular, MDL material components, such as an individual elemental BSDF responsible for a specular highlight or car-paint flakes, can be individually selected by regular expressions and rendered into separate passes for subsequent, possibly non-linear, compositing tasks (see Fig. 9).

The physically-based material model provided by MDL is key for high rendering performance, as verified by the existing integrations in NVIDIA Iray (see Fig. 2) and mental ray. Our implementation supports JIT compilation with CUDA kernels on GPUs, CPUs, and OpenGL GLSL. All images are rendered with Iray Photoreal, except Fig. 2, which shows three renderers.

Acknowledgements

We would like to thank Holly Rushmeier for the material samples distributed at MAM 2014, X-Rite for the respective measured AxF files, TurboSquid for various models used in our scenes, and Thomas Zancker for the BMW scene.

References

[AWL13] AITTALA M., WEYRICH T., LEHTINEN J.: Practical SVBRDF capture in the frequency domain. *ACM Trans. Graph. 32*, 4 (July 2013), 110:1–110:12.

[men10] MENTAL IMAGES GMBH, BERLIN, GERMANY: *Design Specification for MetaSL*, Version 1.2.1, Dec. 15th, 2010.

[NVI15] NVIDIA CORP., SANTA CLARA, CA, USA: *NVIDIA Material Definition Language 1.2: Language Specification*, 2015.

[PH11] PHARR M., HUMPHREYS G.: *Physically Based Rendering*. Morgan Kaufmann, 2nd Ed., 2011.

[Rus14] RUSHMEIER H.: The MAM2014 sample set. In *Eurographics Workshop on Material Appearance Modeling* (2014), Klein R., Rushmeier H., (Eds.).

[Sch14] SCHWARTZ C.: X-Rite demo. Eurographics Workshop on Material Appearance Modeling, 2014.

[Son14] SONY PICTURES IMAGEWORKS INC., ET AL.: *Open Shading Language 1.6: Language Specification*, 2014.

[Vea97] VEACH E.: *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, 1997.