Fast Automatic Level of Detail for Physically-based Materials

Lutz Kettner
NVIDIA Corporation
lkettner@nvidia.com

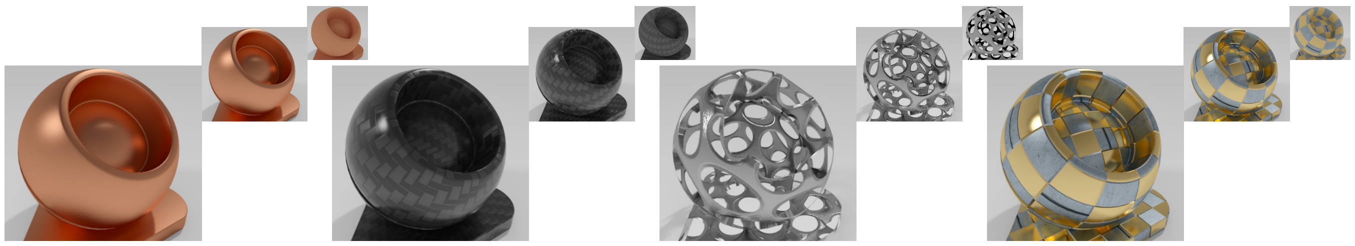

Figure 1: Examples from a 600 materials test set, showing the original material (large images), a simplification to a Fresnel-layered glossy over diffuse material (medium images), and a simplification to a diffuse-only material (small images).

ABSTRACT

Using a term rewriting system, simplifications of physically-based materials described in a declarative programming language can be created automatically. Sets of rules for the term rewriting system allow for customizing simplifications according to use cases. Examples include automatic level-of-detail generation or simplification of materials for faster rendering in realtime viewports and games.

CCS CONCEPTS

•**Computing methodologies →Rendering;** *Rasterization; Ray tracing;* Reflectance modeling;

KEYWORDS

level of detail, physically-based material, term rewriting system

ACM Reference format:
Lutz Kettner. 2017. Fast Automatic Level of Detail for Physically-based Materials. In *Proceedings of SIGGRAPH'17 Talks, Los Angeles, CA, USA, July 30 - August 03, 2017,* 2 pages.
DOI: http://dx.doi.org/10.1145/3084363.3085062

1 INTRODUCTION

Simplification and level of detail can reduce the cost of rendering while preserving the fidelity of the result. Here, we simplify material complexity with a novel level-of-detail framework in a physically-based rendering (PBR) context.

To define materials of rich visual fidelity, renderers mostly used—and still use—procedural shading languages such as RSL, GLSL, HLSL, CgFX, or C/C++ with an API. Existing work on material simplification focuses on such shaders, where the challenge is the semantic gap between the material domain and the chosen programming language. Consequently, solutions rely on a full compiler implementation, its semantic analysis, heuristics and costly optimization strategies to reduce shader complexity.

With PBR, a natural separation of materials into texturing functions and distribution functions emerged, such as the Bidirectional Reflectance Distribution Function (BRDF), which models the surface reflectance behavior. While procedural languages remain the solution for texturing functions, more declarative language paradigms are used to create rich distribution functions. One example is the NVIDIA Material Definition Language (MDL) [Kettner et al. 2015], which uses an algebra of elemental distribution functions, modifiers, and combiners to define a structured representation of a combined distribution function. Another example is the Open Shading Language (OSL) [Sony Pictures Imageworks Inc., et al. 2016], which uses the concept of closure types to embed distribution functions with a restricted algebra in a procedural shader.

Such higher-level declarative definitions can be more easily analyzed, optimized, and simplified. In particular, our level-of-detail framework applies the ideas of term rewriting systems to these declarative definitions. Term rewriting rules transform complex distribution function expressions into simpler distribution function expressions while possibly moving the desired effect of the complex expression into more complex texture functions.

1.1 Related Work

Related methods look at simplification as a non-realtime pre-processing and optimization step. They look at GPU shader programs and, for example, apply compiler technology to migrate fragment shader work loads to interpolated vertex shader computations.

[Olano et al. 2003] use the idea of loophole compiler optimization and focus on reducing texture accesses. [Pellacini 2005] simplifies procedural shaders by applying simplification rules on the abstract-syntax-tree representation. The lower abstraction level and the tree size lead to an optimization algorithm with long runtimes. [Sitthi-Amorn et al. 2011] extend these ideas with genetic programming.

[Wang et al. 2014] use code transformations, surface subdivision, and approximation rules. The approximation rules approximate per pixel computations in the fragment shader based on high-order polynomials fitting on surfaces.

Recent work by [He et al. 2016] uses reduction of detail by far-field approximations or complete elimination of textured effects, such as the transition from normal maps to a glossy BRDF, where dropping normal maps also eliminates the need for tangent vector computations in the vertex shader.

2 TERM REWRITING FOR MATERIALS

The declarative definition of the distribution functions of a physical-based material model can be represented as an expression tree in an algebra that consists of all distribution functions specified in the language and the operations to combine them. Argument values for the distribution function parameters are determined by literal values and function call graphs. Directed acyclic call graphs may be handled by unrolling them to a tree.

A rule for a term rewriting system consists of a *pattern* that can match a sub-expression in the declarative part of the expression tree, and a *replacement expression*. The rule

$$\texttt{mix}(w_1, \texttt{diff}(c_1), w_2, \texttt{diff}(c_2)) \rightarrow \texttt{diff}(w_1 c_1 + w_2 c_2)$$

is a simple example for a pattern that matches a weighted mix of two colored diffuse distribution functions and replaces the pattern with a single diffuse distribution function with a weighted sum of colors.

Given a suitable set $\mathcal{R}$ of rules and applying rule after rule to an initial material expression M_1 creates a sequence

$$M_1 \overset{\mathcal{R}}{\Longrightarrow} M_2 \overset{\mathcal{R}}{\Longrightarrow} \cdots \overset{\mathcal{R}}{\Longrightarrow} M_n$$

of materials, where assuming termination R reduces M_1 to a final expression M_n. Term rewriting is a simple and efficient means to create one simplified material M_n or a whole sequence $<M_l>$ for a level-of-detail representation in one process.

2.1 Creating Suitable Sets of Rewriting Rules

Important questions for a rule set of a term rewriting system are *termination*, *confluence* (is there a unique normal form), and runtime *efficiency*. Typical evaluation strategies for efficient systems are *top-down* or *bottom-up*.

In particular, a bottom-up evaluation strategy leads to a systematic process of deriving a rule set whose correctness follows easily by induction. We start by defining the normal forms that are terminal states in the derivation chain, such as a single diffuse material, or a Disney-principled material model. The full process consists of the following steps:

(1) Define the set $\mathcal{N}$ of all normal form expressions.
(2) For all distribution functions D of the language and combinations of how normal forms in $\mathcal{N}$ can be used as arguments for the parameters of D, create a matching pattern.
(3) Eliminate patterns that are equal to a normal form in $\mathcal{N}$.
(4) For each of the remaining patterns, choose a normal form from $\mathcal{N}$ that is a good replacement expression and define its arguments based on arguments of the match. The choice of the normal form and its arguments can introduce approximations and give room for optimizations.

Rules, like the example above, may transfer complexity from the declarative expression to the regular function call graph. Other rules might lower total complexity by just picking parts with the highest weight. Function call graph complexity can be further reduced with the complementary and orthogonal technique of texture baking, i.e., evaluating function subgraphs over a suitable domain and replacing the subgraph with a simple bitmap texture lookup.

Rules can be complemented with partial evaluation, where functions are evaluated or sampled at simplification time. The resulting value can control rule behavior or be baked for use at render time.

The ideas of the framework presented here apply analogously to all kinds of distribution functions, such as for transmission, emission, volume effects, as well as other parts of a material, such as normal maps, and general function call graphs, assuming their components are specified as algebra.

3 RESULTS

We have implemented the framework based on the NVIDIA Material Definition Language (MDL). We first created rule sets targeting a single diffuse model and a Fresnel-layered glossy over a diffuse material including multi-layer normal maps and cutout masks. The rule sets contain between seven and 23 rules. We applied those rules on a set of 600 diverse materials from different domains; see Figure 1 for examples rendered with NVIDIA Iray. The method nicely transfers important features of a material within the limits of the target models. Simplification (without texture baking) takes 1–2 ms on a 2.6 GHz Intel Core i7 for one material.

The system is very fast for practical material complexities and can be used just-in-time in applications editing original materials and their parameters. It bridges the gap between a rich material model and the limited material models common in game engines and editing viewports. It can also accelerate offline renderers by replacing the detailed materials in certain contexts, such as indirect lighting contributions.

4 CONCLUSION

We presented a framework that shows how a term rewriting system can simplify materials represented by expressions. It uses rule sets, a powerful and easy-to-use way to specify simplifications, which can target fixed material models or create sequences of materials for a level-of-detail representation.

ACKNOWLEDGMENTS

The author would like to thank Jan Jordan for valuable contributions to this project as well as Alexander Keller and Mike Blake for helpful suggestions on this paper.

REFERENCES

Y. He, T. Foley, and K. Fatahalian. 2016. A System for Rapid Exploration of Shader Optimization Choices. *ACM Trans. Graph.* 35, 4 (July 2016), 112:1–112:12.

L. Kettner, M. Raab, D. Seibert, J. Jordan, and A. Keller. 2015. The Material Definition Language. In *Proc. of the Workshop on Material Appearance Modeling*. 1–4.

M. Olano, B. Kuehne, and M. Simmons. 2003. Automatic Shader Level of Detail. In *Proc. of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware (HWWS '03)*. 7–14.

F. Pellacini. 2005. User-configurable Automatic Shader Simplification. In *ACM SIGGRAPH 2005 Papers (SIGGRAPH '05)*. 445–452.

P. Sitthi-Amorn, N. Modly, W. Weimer, and J. Lawrence. 2011. Genetic Programming for Shader Simplification. In *Proc. of the 2011 SIGGRAPH Asia Conference (SA '11)*.

L. Gritz (Ed.). 2016. *Open Shading Language 1.7: Language Specification.* Sony Pictures Imageworks Inc., et al.

R. Wang, X. Yang, Y. Yuan, W. Chen, K. Bala, and H. Bao. 2014. Automatic Shader Simplification Using Surface Signal Approximation. *ACM Trans. Graph.* 33, 6 (2014).