Sharing Physically Based Materials Between Renderers with MDL

Jan Jordan  Software Product Manager MDL
Lutz Kettner  Director Advanced Rendering and Materials
Introduction to NVIDIA Material Definition Language MDL

Matching the appearance of a single material within different rendering techniques

MDL ecosystem

Become part of the ecosystem
Introduction
The **NVIDIA Material Definition Language (MDL)** is technology developed by NVIDIA to define **physically-based** materials for physically-based rendering solutions.
MDL in Substance Designer
MDL in Substance Designer
Matching the appearance of a single material within different rendering techniques
One Scene for Different Renderers

Realtime Rasterizer
Interactive Raytracer
Pathtracer

Share scene and MDL materials for a consistent look

Switching renderers with no scene modifications
## Traditional Shading Language Parts

<table>
<thead>
<tr>
<th>Texturing</th>
<th>Material Definition</th>
<th>Material Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Texture lookups</td>
<td>Glossy reflection</td>
<td>Light loops / trace N rays</td>
</tr>
<tr>
<td>Procedurals</td>
<td>Transparency</td>
<td>OIT / ray-continuation</td>
</tr>
<tr>
<td>Uv-transforms</td>
<td>Translucency</td>
<td>Ray marching</td>
</tr>
<tr>
<td>Projectors</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Noise functions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Math functions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Procedural Programming Language
- Texture lookups
- Procedurals
- Uv-transforms
- Projectors
- Noise functions
- Math functions

Declarative Material Definition
- Glossy reflection
- Transparency
- Translucency

Renderer
- Rasterizer
  - Light loops / OIT
- Raytracer
  - Trace N rays
- Pathtracer
  - Ray-marching
**MDL is not a Shading Language**

MDL defines what to compute, **not** how to compute it
- no programmable shading
- no light loops or access to illumination
- no trace call
- no sampling
- no camera dependence
MDL Material Model

**Material**

- **Surface**
  - BSDF: scattering
  - EDF: emission
    - intensity

- **Volume**
  - VDF: scattering
    - scattering_coefficient
    - absorption_coefficient

- **Geometry**
  - Displacement
  - Cutout opacity
  - Normal

- **Backface**
  - IOR
  - Thin walled
MDL Elemental Distribution Functions

Bidirectional Scattering Distribution Functions

- Diffuse Reflection
- Diffuse Transmission
- Glossy (various)
- Backscatter Glossy
- Specular Reflection
- Measured BSDF
MDL Elemental Distribution Functions

Emissive Distribution Functions

Volume Distribution Functions

Diffuse

Spot

IES Profile

Henyey-Greenstein
MDL Distribution Function Modifiers

- Tint
- Thin Film
- Directional Factor
- Measured Curve Factor
MDL Distribution Functions Combiners

- Normalized Mix
- Clamped Mix
- Weighted Layer
- Fresnel Layer
- Custom Curve Layer
- Measured Curve Layer
MDL 1.4: New BSDF

Modifier:
Complex ior factor

Combiners:
All weights can be color now

Copper
Gold
Silver

Custom curve layer
Defining a Material Using MDL

Create complex materials by layering

```mdl
material plastic(
    color diffuse_color = color(.15,0.4,0.0),
    float roughness = 0.05
) = material(
    surface: material_surface(
        scattering: df::fresnel_layer ( 
            ior: color(1.5),
            layer: df::simple_glossy_bsdf ( 
                roughness_u: glossy_roughness
            ),
            base: df::diffuse_reflection_bsdf ( 
                tint: diffuse_color
            )
        )
    )
);
```
Defining a Function Using MDL

Functions allow control flow like loops, switches, conditionals

```cpp
float summed_perlin_noise (  
    float3 point,  
    int level_count=4,  
    float level_scale=0.5,  
    float point_scale=2.0,  
    bool turbulence=false)
{
    float scale = 0.5, noise_sum = 0.0;
    float3 level_point = point;
    for (int i = 0; i < level_count; i++)
    {
        float noise_value = perlin_noise(level_point);
        if (turbulence)
            noise_value = math::abs(noise_value);
        else noise_value = 0.5 + 0.5 * noise_value;
        noise_sum += noise_value * scale;
        scale *= level_scale;
        level_point *= point_scale;
    }
    return noise_sum;
}
```
MDL Handbook
www.mdlhandbook.com

Added displacement since 2017

Cloth example

4 anisotropic glossy highlights + translucency
Additional MDL Benefits

**Measured Materials**
- Spatially Varying BRDF
- AxF from X-Rite
- Measure Isotropic BSDF

**Designed for Parallelism**
- Little data dependencies
- Side-effect free functions

**Material Catalogs**
- Modules and packages
- Archives
MDL ecosystem
MDL - Past, Present and Future

- MDL 0.x ...
- MDL 1.0
  - Iray 2013
- MDL 1.1
  - Public specification
- MDL 1.2
  - Nvidia Iray Plugins
  - mental ray (3ds Max, Maya)
- MDL 1.3
  - Bunkspeed
  - Substance Designer
- MDL 1.4
  - Catia V6
  - Vray
  - Daz 3d
  - ESI IC.IDO
  - Advisory Council
  - Public SDK
  - Unreal Studio 4.20
  - Solidworks Visualize

MDL Advisory Council
Companies sharing our vision of MDL

Joint direction of MDL and the MDL eco system
Include expertise other companies have gained in the field and with MDL
NVIDIA vMaterials 1.5
~1600 MDL materials verified for accuracy - FREE TO USE
Become part of the ecosystem
Become Part of the Ecosystem

Integrate MDL enabled renderer

MDL is included

Write your own compiler

Based on the freely available MDL Specification

Use the MDL SDK

Published under the NVIDIA Designworks License and...
Write Your Own Compiler

**MDL Specification**

Language specification document
Free to use

http://www.nvidia.com/mdl/

**MDL conformance test suite**

Syntactic conformance tests
Semantic conformance tests
Available on request
MDL SDK 2018.1

Features

MDL 1.4
DB for MDL definitions
DAG view on materials
several compilation modes
MDL editing
Code generators
PTX, LLVM IR, x86, GLSL (fcts. only)
Distiller and texture baker
Samples
Documentation and tutorials
MDL SDK and OptiX
We will give you all you need…

• MDL SDK PTX backend produce PTX code suitable to be called by OptiX available since MDL SDK 2017.1
Full Material Compilation
OptiX and CUDA rendering samples with MDL SDK 2018.0 and OptiX 5.1
MDL Distilling

Released as part of Iray/MDL SDK

Example: UE4 target with clearcoat and transparency through alpha

GLSL rendering sample using Distilling and baking comes with MDL SDK 2018.1
May the Source Be with You

NVIDIA Open Sources the MDL SDK

Today  https://github.com/NVIDIA/MDL-SDK

BSD 3-clause license

Full MDL SDK
• 48 modules, 570 files, 310 KLOC
• Excluding
  MDL Distilling and texture baking
  GLSL compiler back-end
• Added MDL Core API
• Includes MDL Core Definitions and more
# MDL Core API

A Lower-level Compiler API in the MDL SDK

<table>
<thead>
<tr>
<th>MDL SDK API</th>
<th>MDL Core API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Higher-level API for easy integration</td>
<td>API close to the compiler</td>
</tr>
<tr>
<td>Reference counted interfaces</td>
<td>Objects managed in arenas</td>
</tr>
<tr>
<td>Mutable objects</td>
<td>Immutable objects</td>
</tr>
<tr>
<td>In-memory store</td>
<td>Stateless compiler</td>
</tr>
<tr>
<td>Texture and resource importer</td>
<td>Callbacks</td>
</tr>
</tbody>
</table>
MDL Takeaways

What is MDL

- Declarative Material Definition
- Procedural Programming Language

MDL Ecosystem

- NVIDIA vMaterials
- MDL Advisory Council

Starting Material

- Open Source release
- MDL Specification
- MDL Handbook
- MDL SDK
- MDL Backend Examples
- Conformance Test Suite
Further Information on MDL

www.nvidia.com/mdl
raytracing-docs.nvidia.com/mdl/index.html

Documents

- NVIDIA Material Definition Language ▸ Technical Introduction
- NVIDIA Material Definition Language ▸ Handbook
- NVIDIA Material Definition Language ▸ Language Specification