Sharing Physically Based Materials Between Renderers with MDL

Jan Jordan  Software Product Manager MDL
Lutz Kettner  Director Advanced Rendering and Materials

March 18, GTC San Jose 2019
Agenda

Introduction to NVIDIA Material Definition Language MDL

Matching the appearance of a single material within different rendering techniques

Defining physically-based materials

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.
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

Texturing
- Texture lookups
- Procedurals
- Uv-transforms
- Projectors
- Noise functions
- Math functions

Material Definition
- Glossy reflection
- Transparency
- Translucency

Material Implementation
- Light loops / trace N rays
- OIT / ray-continuation
- Ray marching
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
**Procedural Programming Language**

**Declarative Material Definition**

**Rasterizer**
- Light loops / AOIT

**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

<table>
<thead>
<tr>
<th>material</th>
<th>surface</th>
<th>volume</th>
<th>geometry</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>backface</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

...
MDL Material Model

**material**

**surface**
- bsdf
- scattering

**volume**

**geometry**

**backface**
- ...

MDL Material Model

- **material**
  - **surface**
    - bsdf scattering
    - **emission**
      - edf emission
      - intensity
  - **backface**
    - ...
  - **volume**
  - **geometry**
MDL Material Model

material

surface
- bsdf scattering
- emission
  - edf emission
  - intensity
- backface
  ...

volume
- vdf scattering
- scattering_coefficient
- absorption_coefficient

geometry
MDL Material Model

**material**

**surface**
- **bsdf** scattering
- **emission**
  - **edf** emission
  - intensity

**volume**
- **vdf** scattering
  - scattering_coefficient
  - absorption_coefficient

**geometry**
- displacement
- cutout_opacity
- normal

backface

...
MDL Material Model

- **Material**
  - **Surface**
    - bsdf: scattering
  - **Emission**
    - 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  Layered Material Example
Defining Physically-based Materials With Source Code
MDL is a ‘C’ like language. The material viewed as a struct

```c
struct material {
    bool thin_walled;
    material_surface surface;
    material_surface backface;
    color ior;
    material_volume volume;
    material_geometry geometry;
};
```
Defining a Material Using MDL

MDL is a ‘C’ like language. The material and its components viewed as a struct

```c
struct material {
    bool thin_walled;
    material_surface surface;
    material_surface backface;
    color ior;
    material_volume volume;
    material_geometry geometry;
};

struct material_surface {
    bsdf scattering;
    material_emission emission;
};
```
Defining a Material Using MDL

MDL is a ‘C’ like language. The material and its components viewed as a struct

```
struct material {
    bool thin_walled = false;
    material_surface surface = material_surface();
    material_surface backface = material_surface();
    color ior = color(1.0);
    material_volume volume = material_volume();
    material_geometry geometry = material_geometry();
};

struct material_surface {
    bsdf scattering = bsdf();
    material_emission emission = material_emission();
};
```
Defining a Material Using MDL

Material struct is already fully defined

```cpp
material();
```
Defining a Material Using MDL

Material struct is already fully defined

material();
Defining a Material Using MDL

Creating new materials

```
material name ( material-parameters )
  = material ( material-arguments );
```
Defining a Material Using MDL

```mdl
material plaster() = material(
    surface: material_surface(
        scattering: df::diffuse_reflection_bsdf()
    )
);
```
Defining a Material Using MDL

New materials can have parameters

```mdl
material plaster ( color plaster_color = color(.7))
  = material(
      surface: material_surface (  
        scattering: df::diffuse_reflection_bsdf (  
          tint: plaster_color  
        )  
      )  
  );
```

Defining a Material Using MDL

Create complex materials by layering

```plaintext
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  
            )
        )
    )
);
```
MDL Handbook
www.mdlhandbook.com

January 10th update: more on procedural texturing and displacement

Upcoming: advanced volumes
MDL Procedural Programming Language
C-like language for function definitions

Function results feed into material and function parameters

“Shader graphs” are equivalent to function call graphs

```
texture_coordinate
  texture_space`: 0

  summed_perlin_noise
    position

  color_constructor
    value

  Material plaster
    plaster_color
```
Defining a Function Using MDL

Functions allow control flow like loops, switches, conditionals

```mdl
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;
}
```
Defining a Function Using MDL

Call graph of functions substitute shader graphs

```cpp
material perlin_noise_material()
  = plaster(
    plaster_color: color(
      summed_perlin_noise(
        point: state::texture_coordinate(0)
      )
    )
  )
)```

Call graph of functions substitute shader graphs
MDL Module System

MDL is program code

MDL is a programming language allowing dependencies among modules and materials

```cpp
import nvidia::vMaterials::Design::Metal::chrome::*;
```

We use search paths to resolve imports
MDL Module System

MDL is program code

MDL is a programming language allowing dependencies among modules and materials

```plaintext
import nvidia::vMaterials::Design::Metal::chrome::*;
```

We use search paths to resolve imports

```
C:\Users\Jan\Documents\mdl\nvidia\vMaterials\Design\Metal\chrome.mdl
```

- **search path**: C:\Users\Jan\Documents\mdl\nvidia\vMaterials\Design\Metal\chrome.
- **MDL package space**: nvidia::vMaterials::Design::Metal::chrome
MDL 1.5 Preview

MDL Encapsulated File Format (MDLE)

One material - fully self contained in one file
Includes textures, previews, etc. in the file
Renaming and copying works

... work just like textures
MDL 1.5 Preview
Internationalization (i18n)

Localization of all MDL string annotations

Based on OASIS standard XLIFF 1.2: XML Localisation Interchange File Format
http://docs.oasis-open.org/xliff/xliff-core/xliff-core.html

Package and module XLIFF files in MDL file hierarchy

**Example**

```
C:\Users\%USERNAME%\Documents\mdl\nvidia\vMaterials\fr.xlf  MDL search path
nvidia\vMaterials\AEC\Glass\Mirror_fr.xlf  French vMaterial package XLIFF file
nvidia\vMaterials\AEC\Glass\Mirror_fr.xlf  French Mirror module XLIFF file
```
struct material {
    ...
    hair_bsdf hair;
};

hair_bsdf chiang_hair_bsdf {
    float diffuse_reflection_weight = 0.0;
    color diffuse_reflection_tint = color(1.0);
    float2 roughness_R = float2(0.0);
    float2 roughness_TT = roughness_R;
    float2 roughness_TRT = roughness_TT;
    float cuticle_angle = 0.0;
    color absorption_coefficient = color();
    float ior = 1.55;
};
MDL 1.5 Preview
Microfacet coloring to support flip-flop car paints and more

1D measured curve (MDL >=1.4)

2D measured curve (new in MDL 1.5)
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 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 Iray
Iray 2019 roadmap

Iray RTX 2019
• Release in May
• RTX support, up to 5 times speedup!
• MDL 1.5 support for
  • MDLE
  • localization
  • 2d measured curve
SOLIDWORKS Visualize
MDL import since 10/2018, tweaking + viewport preview coming
Epic Unreal Studio

“Real-time workflows for enterprise” www.unrealengine.com/studio

MDL support through DATASMITH
Vizoo xTex

MDL export in the next release

“Vizoo is the number one supplier of Soft-and Hardware solutions for the physically accurate digitization of material swatches in the fashion industry. “

www.vizoo3d.com
MEDULR
Online MDL editor and material library preparing for opening

Discover, create and share materials. We're building a global community to create the worlds largest material library.

www.medulr.com
Quixel Megascans
“Incredible scans and tools for creatives.”

quixel.com
Focus on Material Exchange
Freely choose where to author material content

- Substance Designer
  - create

- Chaosgroup V-RAY
  - consume

- Iray for Rhino
  - modify
NVIDIA vMaterials 1.6 - SIGGRAPH 2019
~1700 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
RENDERING

Iray SDK
OptiX SDK
MDL SDK
NV Pro Pipeline
vMaterials

VOXELS

GVDB Voxels
VXGI

VIDEO

GPUDirect for Video
Video Codec SDK

MANAGEMENT

GRID SW MGMT SDK
NVAPI/NVWMI

DISPLAY

Multi-Display
Capture SDK
Warp and Blend

https://developer.nvidia.com/designworks
MDL SDK 2019 (.0.1)

Features

MDL 1.4 (1.5 feature previews)
DB for MDL definitions
DAG view on materials
several compilation modes
MDL editing
Code generators
PTX, LLVM IR, x86, HLSL, GLSL (fcts. only)
Distiller and texture baker
Samples
Documentation and tutorials
MDL SDK 2019 - What is New

Features

Preview of MDL 1.5 features:

- Localisation
- MDL encapsulated format

Improved BSDF reference implementation (libbsdf) including measured brdf and emissive distribution functions

Additional distilling mode (transmissive PBR)

HLSL backend (2019.0.1)

Automatic derivatives for texture filtering

Open source release available on Github

- Includes exclusive MDL core compiler API

More samples

- Updated CUDA sample for transmissive materials
- (CPU rendering sample)?
- MDL browser sample
MDL and RTX

Materials tricky for today's game engines become feasible with RTX

- Anisotropic glossy reflections
- True refractive and volumetric materials
- Measured BRDF
- Proper translucency
- Complex glossy lobe shape and color

MDL materials make RTX shine!
**MDL SDK and RTX**

The MDL SDK directly generates material code for use in RTX enabled renderer

**Microsoft DXR**

- HLSL back-end with MDL SDK 2019.0.1 and sample path tracer in the SDK

**NVIDIA OptiX**

- PTX back-end since MDL SDK 2018.1 sample program available as part of Optix 5.1 & 6

Integrating MDL with an RTX based renderer is simple!
Automatic Derivatives for Texture Filtering

OptiX sample renderer integration: Derivatives off
Automatic Derivatives for Texture Filtering

OptiX sample renderer integration: Derivatives on
MDL in Realtime Rendering

Three approaches

1. Ubershader
2. Compilation: on-demand shader generation
3. Distillation to fixed material model

All based on MDL SDK
Distillation to Fixed Material Model

**MDL Material**
- Complex BSDF layering
- Complex procedurals

**Fixed Material Model**
- Simple BSDF structure
- One texture per parameter
Distillation to Fixed Material Model

**MDL Material**
- Complex BSDF layering
- Complex procedurals

**Distillation**

**Fixed Material Model**
- Simple BSDF structure
- One texture per parameter
Distillation to Fixed Material Model

**MDL Material**
- Complex BSDF layering
- Complex procedurals

**Fixed Material Model**
- Simple BSDF structure
- One texture per parameter
Distillation to Fixed Material Model

**MDL Material**
- Complex BSDF layering
- Complex procedurals

**Distillation**

**Fixed Material Model**
- Simple BSDF structure
- One texture per parameter

Approximate render result: Some materials will look quite different
Distillation to Fixed Material Model

**MDL Material**
- Complex BSDF layering
- Complex procedurals

**Distillation**

**Fixed Material Model**
- Simple BSDF structure
- One texture per parameter

Fast projection of material instances: Realtime editing

Approximate render result: Some materials will look quite different
Distillation to Fixed Material Model

**MDL Material**
- Complex BSDF layering
- Complex procedurals

**Fixed Material Model**
- Simple BSDF structure
- One texture per parameter

Fast projection of material instances: Realtime editing

Flexible framework to target different fixed models not a fixed MDL subset (no "MDL lite")

Approximate render result: Some materials will look quite different
Distillation to Fixed Material Model

Results on vMaterials

diffuse-only

Fresnel( glossy, diffuse)

original
MDL Distilling

Released as part of Iray/MDL SDK

Multiple distilling targets (diffuse only, diffuse_glossy, UE4, new: transmissive PBR)

Original:
Iray MDL

Projection:
Dassault Stellar with Enterprise PBR
May the Source Be with You

NVIDIA Open Sourced the MDL SDK

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

4 releases shipped since SIGGRAPH 2018
# 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

- MDL Specification
- MDL Handbook
- MDL SDK
- MDL Backend Examples
- Conformance Test Suite

Starting Material

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

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

MDL@GTC
Mon 9 AM SJCC 230B Sharing Physically Based Materials Between Renderers with MDL
Mon 10 AM SJCC 230B Integrating the NVIDIA Material Definition Language MDL in Your Application
Mon 11 AM Hilton Hotel Almaden 2 A New PBR Material Serving Mobile, Web, Real-Time Engines and Ray Tracing
Tue 9 AM Hilton Hotel Almaden 2 Multi-Platform Photo-Real Rendering: Utilizing NVIDIA’S MDL and Allegorithmic’s Substance Suite for Product Imaging
Thu 10 AM SJCC 230C Real-Time Ray Tracing with MDL Materials

GTC On-Demand
on-demand-ghtc.gputechconf.com