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

Defining physically-based materials

Measured materials

MDL ecosystem

Become part of the ecosystem
Introduction
What is NVIDIA MDL?

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 Rendering Modes

- Realtime Rasterizer
- Interactive Raytracer
- Pathtracer

Share Scene Database and MDL materials for a consistent look

Switching modes with 1 call 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
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
- Volume
- Geometry
- Backface
  - ...
MDL Material Model

- **material**
  - surface
    - bsdf scattering
  - backface
    - ...
  - volume
  - geometry
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

- 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

**backface**
- ... ior
- thin_walled

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

**geometry**
- displacement
- cutout_opacity
- normal
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
MDL    Layered Material Example
diffuse bsdf
red
diffuse bsdf
yellow

diffuse bsdf
red

custom-curve layering
diffuse bsdf
red

custom-curve layering

diffuse bsdf
yellow

glossy bsdf
roughness 0.5

weighted layering
diffuse bsdf
red

diffuse bsdf
custom-curve
layering

glossy bsdf
roughness 0.5

glossy bsdf
roughness 0.1

weighted
layering

weighted
layering
diffuse bsdf yellow

custom-curve layering

diffuse bsdf red

custom-curve layering

weighted layering

glossy bsdf roughness 0.5

weighted layering

glossy bsdf roughness 0.1

weighted layering

fresnel layering

specular bsdf
Defining physically-based materials with source code
Defining a Material Using MDL

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

```c
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

Creating new materials

```mdl
material mymaterial ()
  = material ();
```
Defining a Material Using 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

Added displacement since 2017

Cloth example

4 anisotropic glossy highlights + translucency
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

MDL is ‘C’ like

```
type-of-return-value function-name ( parameters )
{
    statements
}
```
Defining a Function Using MDL

Function access render state through standard modules

```cpp
color uv_as_color()
{
    return color( state::texture_coordinate(0) );
}
```
Defining a Function Using MDL

Use functions to drive BSDF or material parameters

```cpp
color uv_as_color()
{
    return color(state::texture_coordinate(0));
}

material uv_as_color_material_v2()
= plaster( plaster_color: uv_as_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

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

```python
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
- **MDL package space**: nvidia::vMaterials::Design::Metal::chrome
UDIM and uv-tiles
New in MDL 1.4

UDIM texture layout in Autodesk Maya, rendering in Iray
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
Measured materials
**Spatially Varying (SV)BRDF**

Set of textures for an analytic material model

<table>
<thead>
<tr>
<th>Texture Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>diffuse</td>
</tr>
<tr>
<td>specular</td>
</tr>
<tr>
<td>glossy</td>
</tr>
<tr>
<td>kurtosis</td>
</tr>
<tr>
<td>normal</td>
</tr>
</tbody>
</table>

**Fixed MDL material**
Spatially Varying (SV)BRDF

Allegorithmic: Substance Designer

Sample

Simple acquisition method

Raw measurements

Images courtesy of Allegorithmic
Spatially Varying (SV)BRDF

Allegorithmic: Substance Designer

Postprocessing with Substance Designer: crop, repair, tile
Spatially Varying (SV)BRDF

Allegorithmic: Substance Designer

Render with Iray and export to MDL

Easy modifications

Images courtesy of Allegorithmic
Spatially Varying (SV)BRDF

X-Rite: TAC 7, Pantora, AxF file format

Sample TAC 7 Virtual Lightbooth

Images courtesy of X-Rite
Spatially Varying (SV)BRDF

X-Rite: TAC 7, Pantora, AxF file format

Pantora

Images courtesy of X-Rite
Spatially Varying (SV)BRDF

X-Rite: TAC 7, Pantora, AxF file format

AxF file format importer to Iray and MDL

Added dirt with MDL layers

Substance Designer combines AxF and MDL

Images courtesy of X-Rite
Light Path Expressions

Paths that interact with wall + Paths that do not
Light Path Expressions

Paths that interact with wall

Paths that do not

\[ \text{Paths that interact with wall} \times + \text{Paths that do not} = \]
Light Path Expressions

Paths that interact with wall

Paths that do not

=
Light Path Expressions

Paths that interact with wall

Paths that do not

*
Light Path Expressions

Paths that interact with wall

Paths that do not

Edit the Wall Color easily in Post - and get proper reflections and color bounce
MDL Complements Light Path Expressions

LPEs can select individual DF components

- Light falling onto the ground without first passing through the glass
- Caustics cast by the glass
- Specular reflections on the glass
- Specular reflections on the ice cube
- All remaining interactions
MDL ecosystem
MDL - Past, Present and Future

- MDL 0.x ...
- MDL 1.0
  - Iray 2013
- MDL 1.1
  - Public specification
  - Nvidia Iray Plugins
  - mental ray (3ds Max, Maya)
  - Bunkspeed
  - Substance Designer
  - Catia V6
- MDL 1.2
  - Vray
  - Adobe Dimension
  - Daz 3d
- MDL 1.3
  - ESI IC.IDO
- MDL 1.4
  - Advisory Council
  - Public SDK
  - 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 Iray
Shipping integrally within commercial products

Catia V6 and Industrial Designer

Substance Designer & Painter

DAZ Studio

Patchwork 3D

SOLIDWORKS

SIEMENS PLM

migenius

Visualize

NX 11

SketchUp
Iray Plugin Solutions
www.irayplugins.com

Iray for 3ds Max

Iray Server

Iray for Maya

Iray for Rhino
MDL in VRAY

MDL MATERIALS
NVIDIA's physical material format
MDL Adobe Dimension and Adobe Stock

http://www.adobe.com/products/dimension.html
MDL in Substance Designer
MDL in Substance Designer
MDL in Substance Designer
MDL in Substance Designer
MDL in Substance Designer
MDL in Substance Designer
Focus on Material Exchange
Freely choose where to author material content

- Substance Designer: create
- Chaosgroup V-RAY: consume
- Iray for Rhino: modify
NVIDIA vMaterials 1.4
~1400 MDL materials verified for accuracy - FREE TO USE
NVIDIA vMaterials 1.5 - Coming Q2 2018
More than 200 new material variants

vMaterials 1.4
ground asphalt

old variant is hidden

vMaterials 1.5: new presets
NVIDIA vMaterials 1.4
Harder to control parameters and texture parameters
NVIDIA vMaterials 1.5 - Coming Q2 2018
More flexible and user-centric parameters
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

License the MDL SDK

Contact us for licensing information
Write Your Own Compiler

MDL Specification can be downloaded @
http://www.nvidia.com/mdl/

MDL conformance test suite

Syntactic conformance tests - available at request

Semantic conformance tests
MDL SDK 2018.0

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 2018.0 - What is New

Features

MDL 1.4 support
Class compilation support in all modes
Link mode
Full material compilation with BSDF reference implementation
Improved distilling quality
Flexible render state binding in backends

API to enumerate all dependent resources
Access to SDK version at API entry point
Auto shutdown
SDK helper class for simplified access to annotations
New samples for all back-ends
Search in MDL Specification finds now names with ‘_’
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 in Realtime Rendering

Three approaches

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

All based on MDL SDK
Compilation: On-demand Shader Generation

Example deployment
Distillation to Fixed Material Model

MDL Material

- Complex BSDF layering
- Complex procedurals

Fixed Material Model

- Simple BSDF structure
- One texture per parameter
- Examples
  - Disney-principled BRDF model
  - UE4 model (moving target)
  - normal map, cutout
Distillation to Fixed Material Model

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

**Fixed Material Model**
- Simple BSDF structure
- One texture per parameter
- Examples
  - Disney-principled BRDF model
  - UE4 model (moving target)
  - normal map, cutout

Approximate render result: Some materials will look quite different

Fast projection of material instances: Realtime editing

Flexible framework to target different fixed models (moving target) not a fixed MDL subset (no “MDL lite”)
Distillation to Fixed Material Model
Results on vMaterials

diffuse-only
Fresnel( glossy, diffuse)
original
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
MDL Takeaways

What is MDL

- Declarative Material Definition
- Procedural Programming Language

MDL Ecosystem

- NVIDIA vMaterials
- MDL Advisory Council

Starting Material

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

- Iray SDK
- OptiX SDK
- MDL SDK
- NV Pro Pipeline
- vMaterials

**PHYSICS**

- PhysX

**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

Further Information on MDL

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

Documents

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

MDL@GTC

Monday, March 26, 10:00 AM - 10:50 AM – Room 230C
Integrating the NVIDIA Material Definition Language MDL in Your Application

Monday, March 26, 10:30 AM – 10:55 AM – Hilton Santa Clara
MDL & Substance in Automotive

Wednesday, March 28, 2:00 PM – 5:00 PM – Westin Sainte Claire Ballroom
Iray Leaders