SHARING PHYSICALLY BASED MATERIALS BETWEEN RENDERERS WITH MDL

Jan Jordan  
Software Product Manager MDL

Lutz Kettner  
Senior Manager, Advanced Rendering and Materials
Agenda

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 eco-system

Become part of the eco-system

Outlook
What is NVIDIA MDL?

The NVIDIA Material Definition Language (MDL) is technology developed by NVIDIA ARC to define physically-based materials for its rendering solutions. It is central for physically-based rendering.
courtesy Harley Davidson
Iray 2016

Rendering Modes

Realtime

60 FPS

Interactive

20 FPS

Photoreal

10 FPS

Minutes

Shares Scene Database and Material Description for a consistent look

Common Materials within easy to create material catalogues
Iray Realtime OpenGL Rasterizer
Iray Interactive Ray Tracer, Direct Illumination
Iray Photoreal Path Tracer
Traditional Shading Language Parts

Texturing
- Texture lookups
- Procedurals
- uv-transforms
- Projectors
- Noise functions
- Math functions
- Render state

Material Definition
- Glossy reflection
- Transparency
- Translucency

Material Implementation
- Light loops / Trace N rays
- OIT / ray-continuation
- Ray-marching
MDL

Procedural Programming Language
- Texture lookups
- Procedurals
- uv-transforms
- Projectors
- Noise functions
- Math functions
- Render State

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
<table>
<thead>
<tr>
<th>MDL</th>
<th>Material Model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>material</strong></td>
<td></td>
</tr>
<tr>
<td><strong>surface</strong></td>
<td><strong>volume</strong></td>
</tr>
<tr>
<td>bsdf</td>
<td>scattering</td>
</tr>
<tr>
<td>emission</td>
<td>scattering</td>
</tr>
<tr>
<td>edf</td>
<td>emission</td>
</tr>
<tr>
<td></td>
<td>intensity</td>
</tr>
<tr>
<td></td>
<td>absorption_coefficient</td>
</tr>
<tr>
<td></td>
<td>scattering_coefficient</td>
</tr>
<tr>
<td>backface</td>
<td>ior</td>
</tr>
<tr>
<td></td>
<td>thin_walled</td>
</tr>
</tbody>
</table>
MDL Elemental Distribution Functions

- Bidirectional Scattering Distribution Functions
  - Diffuse Reflection
  - Diffuse Transmission
  - Simple Glossy
  - Backscattering Glossy
  - Specular Pure Reflection
  - Specular Reflection & Transmission
  - Measured BSDF
Bidirectional Scattering Distribution Functions

New in MDL 1.3:

Beckman-Smith/V-Cavities
GGX-Smith/V-Cavities
Ward Geisler Moroder

...also with transmission...also with transmission
MDL

Elemental Distribution Functions

Emissive Distribution Functions
- Diffuse
- Spot
- IES Profile

Volume Distribution Functions
- Anisotropic Absorption & SSS
- Anisotropic + IOR & Internal Scattering
- Anisotropic w/ Light
MDL Distribution Function Modifiers

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

- Normalized Mix
- Clamped Mix
- Weighted Layer

- Fresnel Layer
- Custom Curve Layer
- Measured Curve Layer
MDL

Layered Material Example
diffuse
tint: red
custom-curve layering
weighted layering
weighted layering
Fresnel layering
diffuse
tint: yellow
glossy
roughness:
glossy
roughness:
specular
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

material();
Defining a Material Using MDL

Material struct is already fully defined

material();
Defining a Material Using MDL

Creating new materials

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

Creating new materials

```java
material mymaterial ()
    = material ();
```
Defining a Material Using MDL

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

New materials can have parameters

```plaintext
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

```cpp
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
            )
        )
    )
);```

Create complex materials by layering.
MDL Handbook

www.mdlhandbook.com

60+ new pages since GTC 2015

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
Defining a Function Using MDL

MDL is ‘C’ like

```c

typedef function-name ( parameters )
{
    statements
}
```

Defining a Function Using MDL

Function access render state through standard modules

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

Use functions to drive BSDF or material parameters

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

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

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

Fast scan for believable materials

Measured Materials

Quantitative measurements for predictive rendering

---

A recent trend recognizes the need for practical high-resolution digital assets in games and films. However, capturing the spatially-varying reflectance and small geometric variations that greatly contribute to their look is challenging. Most current systems require either specialized hardware, long capture times, user intervention, or rely heavily on heuristics. Our work takes this further, offering a wide range of materials from diffuse to almost mirror-like specular reflectance. We believe that automatic capture of such models is highly desirable; however, currently systems require either specialized hardware, or that are time-consuming, user-intensive, or rely heavily on heuristics.

Spatially-varying reflectance and small geometric variations play a vital role in the appearance of real-world surfaces. Consequently, roscales, they still exhibit scratches, scuffing and other local material properties. Even if perhaps mostly flat and homogeneous over large areas, they exhibit spatially-varying surface reflectance. Most natural materials exhibit spatially-varying surface reflectance, making strong assumptions on the spatial material distribution and making spatially-varying BRDF (SVBRDF) acquisition infeasible. To still capture the most prominent features of the material, using a single viewpoint and illuminating the sample using a planar light source significantly smaller than a full spherical light-source.

First, we confine observations to a smaller range of the angular domain, which complicates its acquisition. Exhaustive sampling of the six-dimensional space leads to prohibitive acquisition times. Second, we use fully automated commodity hardware only, allowing applicability to near-planar surfaces, but we argue that this case is common enough to cover a majority of real-world scenarios.

To still capture the most prominent features of the material, using a single viewpoint and illuminating the sample using a planar light source significantly smaller than a full spherical light-source.

Our concrete setup consists of a single LCD screen and one camera, for capture and processing with no intermittent user intervention. Our system is the first to offer such generality, while requiring only current systems require either specialized hardware, long capture times, user intervention, or rely heavily on heuristics. We describe our acquisition setup that utilizes only portable commodity hardware, which complicates its acquisition. Exhaustive sampling of the six-dimensional space leads to prohibitive acquisition times. To still capture the most prominent features of the material, using a single viewpoint and illuminating the sample using a planar light source significantly smaller than a full spherical light-source.

Our results exhibit a good qualitative match to photographs taken under novel viewing and lighting conditions for a range of materials. Our setup, in particular, a laptop screen can be used for illumination. Our setup, in particular, a laptop screen can be used for illumination. Our results exhibit a good qualitative match to photographs taken under novel viewing and lighting conditions for a range of materials. Our setup, in particular, a laptop screen can be used for illumination. Our setup, in particular, a laptop screen can be used for illumination.
Spatially Varying (SV)BRDF

- Analytic material model
- Measurement drives model parameters

Iray Photoreal

Practical SVBRDF Capture In The Frequency Domain, SIGGRAPH 2013
Miika Aittala and Jaakko Lehtinen, Aalto University, NVIDIA Research
Tim Weyrich, University College London
BTF Measurement Technology from X-Rite
Total Appearance Capture (TAC)

- Measurement stored in Appearance eXchange Format (AxF)
- Iray supports the base profile (SVBRDF representation) of AxF
X-Rite AXF Support
X-Rite AXF Support
X-Rite AXF Support
Measured Isotropic BSDFs

Radiant Zemax: Imaging Sphere
Measured Isotropic BSDFs

- Scanned BSDF
Measured Isotropic BSDFs

• Clear coat added with a specular BSDF layer
Measured Isotropic BSDFs

- Scratches added with another layer for a bump map
Measured Isotropic BSDFs
Physically-based materials are an easy-to-use paradigm

Supports modern rendering algorithms

Allows simple compilers and early optimizations

Enables fast renderers, especially on parallel architectures

GPU friendly

Supports material catalogs
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 Complement 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 - past, present and future

June 2011
First Ideas, influence from mental ray shader API, MetaSL

May 2012
Kick-off of MDL Spec

Jan 2013
MDL 1.0, shipment with Iray 2013

Jan 2014
MDL 1.1, support for measured data
Bunkspeed, Catia start using MDL

May 2014
Made Specification public

Jan 2015
MDL 1.2, resource handling, units
Support in mental ray, exposed in Autodesk 3ds Max, Maya

2015
NVIDIA Iray plugins, DAZ 3d, Allegorithmic Substance Designer, NVIDIA vMaterials

2016
Vray, ESI IC.IDO, MDL 1.3
Watch out for announcements @ GTC2016!
MDL in Commercial Products

- Lightworks
  - Iray for 3ds Max
- UI Composer
- Bunkspeed Shot & Drive
- Sketchup (Bloom Unit) RealityServer
- [0x1]
- mental ray
- CATIA Live Rendering SolidWorks Industrial Designer
- Bunkspeed
- Iray for Maya
- Iray for Rhino
- Iray for Cinema 4D
- Iray for 3ds Max
- Iray for Rhino
- Iray for Cinema 4D
- mental ray
- Iray Server
- VRay
- IC_IDO
- SIEMENS PLM
- DAZ 3D
- Substance Designer
Focus on Material Exchange
Freely choose where to author material content

create

modify

consume

Substance Designer

Iray for Rhino

Iray for 3ds Max

Lightworks
NVIDIA vMaterials
~600 MDL Materials Verified for Accuracy - FREE TO USE
Become Part of the Eco-System

Integrate Iray

MDL is included

Write your own compiler

Based on the freely available MDL Specification

License the MDL SDK

MDL SDK can be licensed independently of Iray.
Become Part of the Eco-System

Write your own compiler

MDL Specification can be downloaded @


MDL conformance test suite

Syntactic conformance tests - available at request

Semantic conformance tests
MDL SDK can be licensed independently of Iray.

Features:

- **MDL 1.3**
- DB view on available definitions
- DAG view on materials, several compilation modes
- MDL editing features
- Backends for compilation of texturing functions
  - PTX
  - LLVM IR
  - GLSL

Contact us for details on availability and licensing.
MDL SDK to Support MDL in a Viewport
Interpretation versus JIT compilation

1. Ubershader in Iray Photoreal
2. On demand shader generation, example code

Cross-compilation of MDL functions to GLSL in MDL SDK
MDL to GLSL Example Code

MDL SDK

MDL Material

Compiled Material
(DAG Backend)

Map MDL DFs, stdlib and other known functions

Upcoming GLSL back-end to cross-compile all functions

Application

OpenGL Renderer

MDL to GLSL Example

Tree Representation
(Data Annotations)

GLSL Shader

GLSL Snippets

Upcoming GLSL back-end to cross-compile all functions
MDL SDK in Use

Monday, Room 210E

11:00 – 11:50
MDL Materials to GLSL Shaders: Theory and Practice (S6311)

14:00 – 14:50
Implement Physically Based Ray Tracing with NVIDIA OptiX and MDL (S6244)
MDL SDK in Use

Tuesday, Room LL21B

14:00 – 14:25

Advances in V-Ray RT GPU (S6345)
MDL Takeaways

**What is MDL**

MDL
- Procedural Programming Language
  - Texture Lookups
  - Procedurals
  - uv-transforms
  - Projectors
  - Noise Functions
  - Math Functions
  - Render State
- Declarative Material Definitions
  - Glossy Reflection
  - Transparency
  - Translucency

**MDL Eco-system**

- Renderer
  - Rasterizer
    - Light loops
    - OIT
  - Raytracer
    - Trace N rays
  - Pathtracer
    - Ray-marching

- vMaterials

**Become part of the Eco-system**

- MDL Specification
- MDL Handbook
- MDL SDK
- MDL to GLSL Code Example
- MDL Conformance Test Suite
MORE INFORMATION

JOIN THE CONVERSATION
#GTC16 -twitter-facebook-linkedin


Demos and Talks at NVIDIA booth on the show floor