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

NVIDIA® Iray®

NVIDIA® mental ray®

central for *physically-based* rendering


B TZ 2012
D
Iray Photoreal


CAPTUR
RENAULT
Renault Design ©2014

courtesy of BLUEMOTION

Photographer: Simon Puschmann/Rendering and Retouching: [zerone]

courtesy Lee Henshall


diamond
emerald
aquamarine
iolite
morganite
tanzanite
ametrine
tourmaline
topaz
silver
turquoise
peridot
sapphire
amethyst
ruby
citrine
garnet
alexandrite
gold
pearl
zircon
jade
opal
onyx
NVIDIA vMaterials with Iray Photoreal

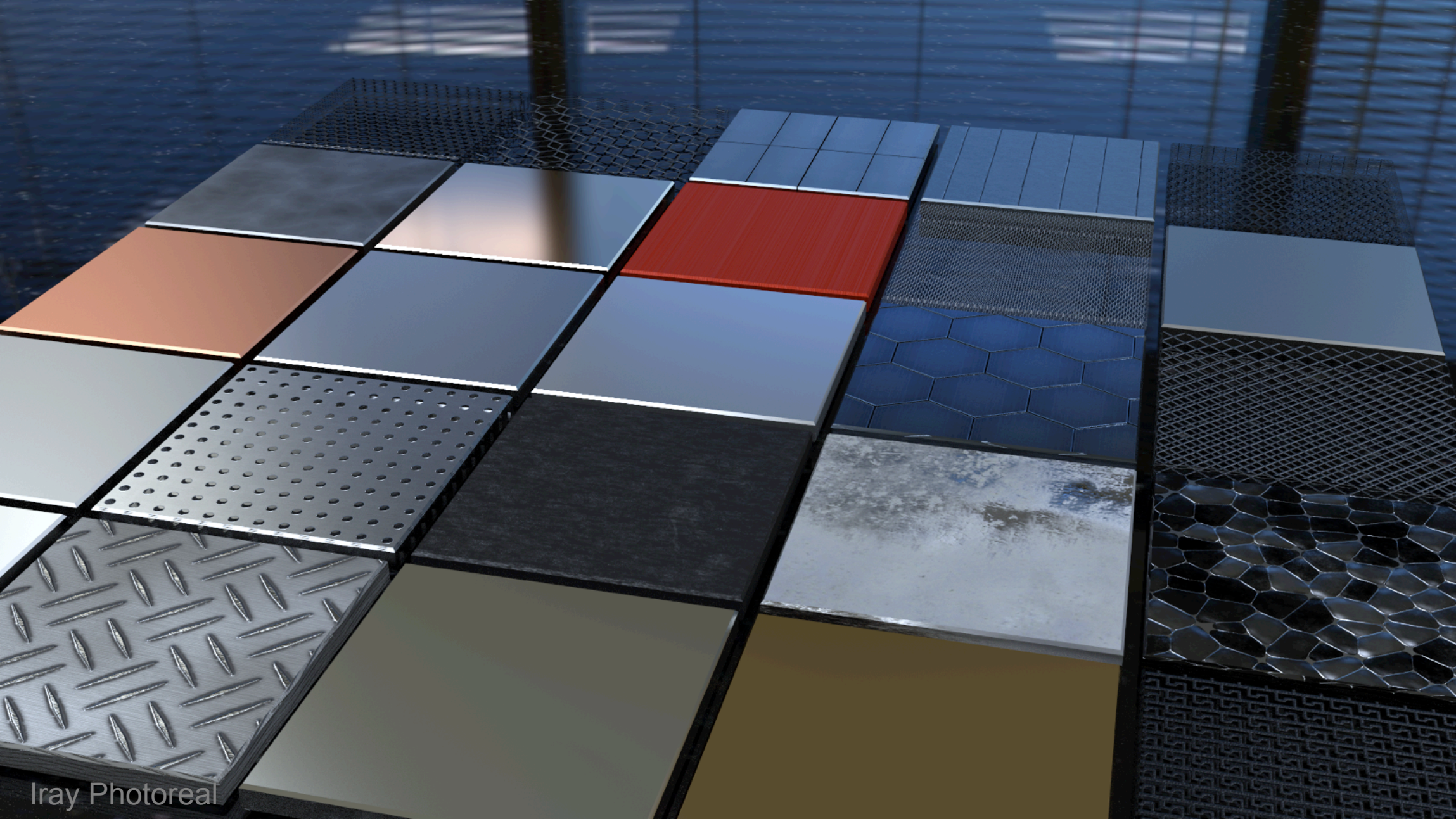
courtesy Harley Davidson

courtesy Harley Davidson

Iray Photoreal


Iray Photoreal

Iray Photoreal


Iray Photoreal


Iray Photoreal


Iray Photoreal


MDL Material Sample
Material Collection, Interior design 2014
MECHANICAL
Iray Photoreal

Iray 2016 Rendering Modes

Realtime | **Interactive** | **Photoreal**

60 FPS ⟷ 15 FPS*

20 FPS ⟷ 2 FPS*.

10 FPS ⟷ Minutes

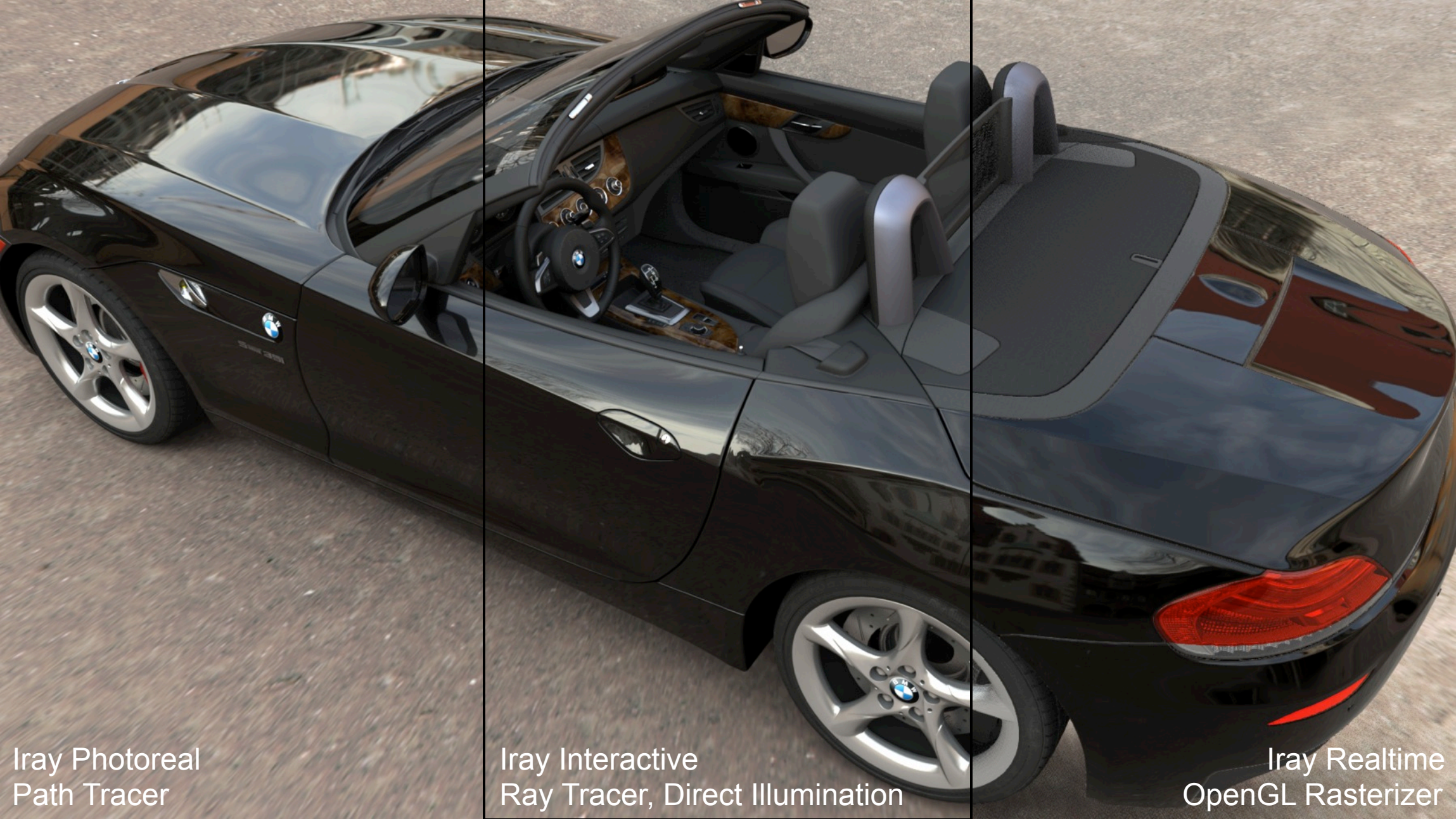
Shares Scene Database and Material Description for a consistent look

Common Materials within easy to create material catalogues


Iray Realtime
OpenGL Rasterizer


Iray Interactive
Ray Tracer, Direct Illumination


Iray Photoreal
Path Tracing


Iray Photoreal
Path Tracer
Iray Interactive
Ray Tracer, Direct Illumination
Iray Realtime
OpenGL Rasterizer

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

Procedural Programming Language → Declarative Material Definition

Renderer

Rasterizer
- Light loops
- OIT

Raytracer
- Trace N rays

Pathtracer
- Ray-marching

MDL

Procedural Programming Language → Declarative Material Definition

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

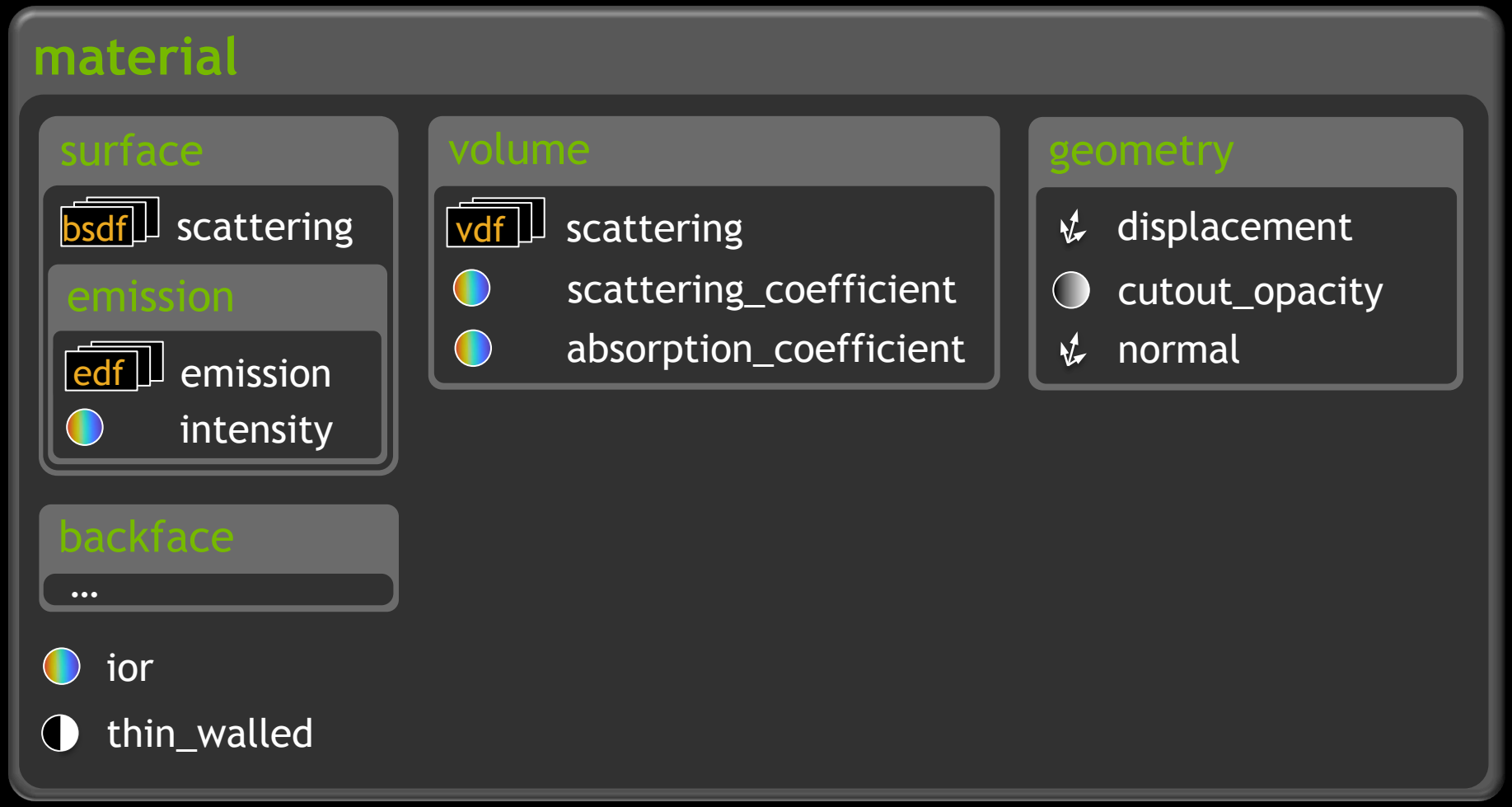
Renderer

Rasterizer
- Light loops
- OIT

Raytracer
- Trace N rays

Pathtracer
- Ray-marching

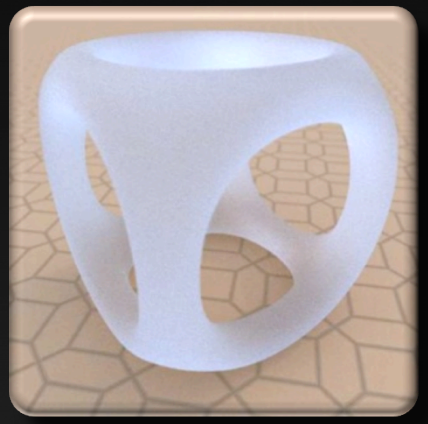
MDL Material Model



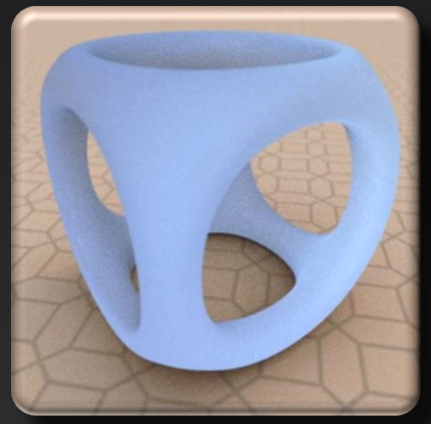
MDL Elemental Distribution Functions

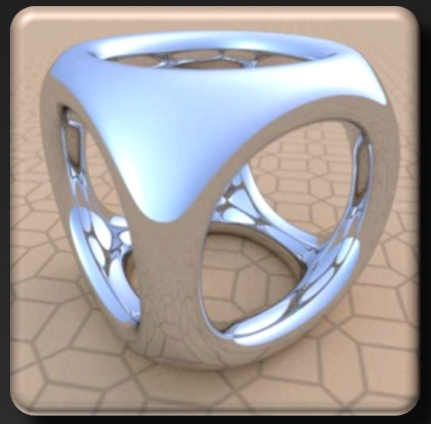
Bidirectional
Scattering
Distribution
Functions

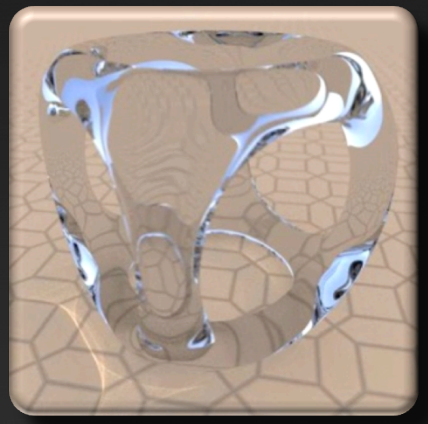

Diffuse Reflection


Diffuse Transmission


Simple Glossy


Backscattering Glossy


Specular
Pure Reflection

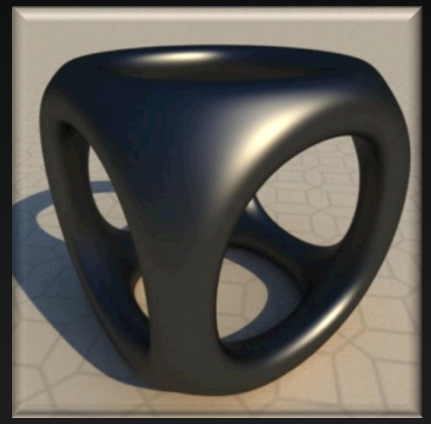

Specular Reflection
& Transmission


Measured BSDF

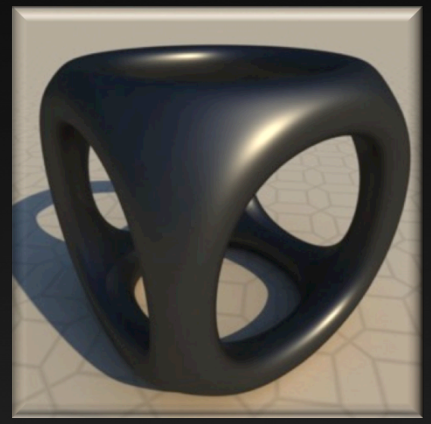
MDL Elemental Distribution Functions

Bidirectional
Scattering
Distribution
Functions

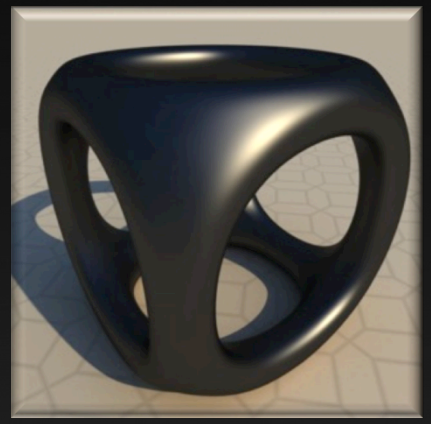
New in MDL 1.3:



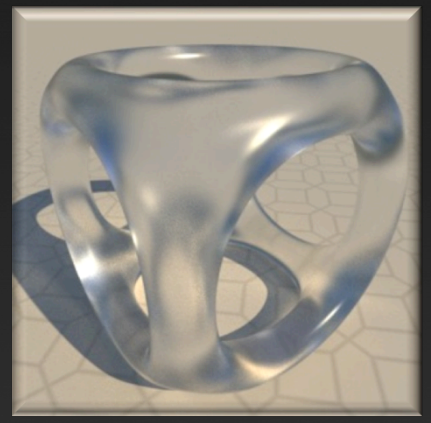
Beckman-Smith/V-Cavities



GGX-Smith/V-Cavities



Ward Geisler Moroder



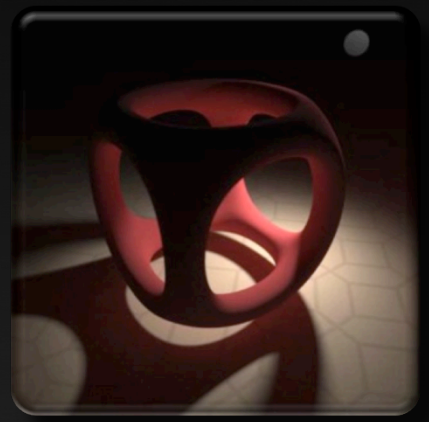
...also with transmission

...also with transmission

MDL Elemental Distribution Functions

Emissive Distribution Functions

Diffuse



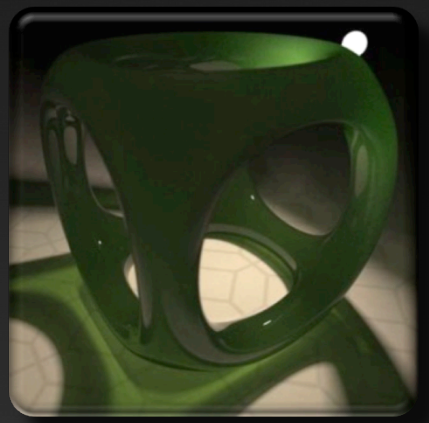
Spot

IES Profile

Volume Distribution Functions

Anisotropic
Absorption & SSS

Anisotropic + IOR
& Internal Scattering



Anisotropic
w/ Light

MDL Distribution Function Modifiers

Tint

Thin Film

Directional Factor

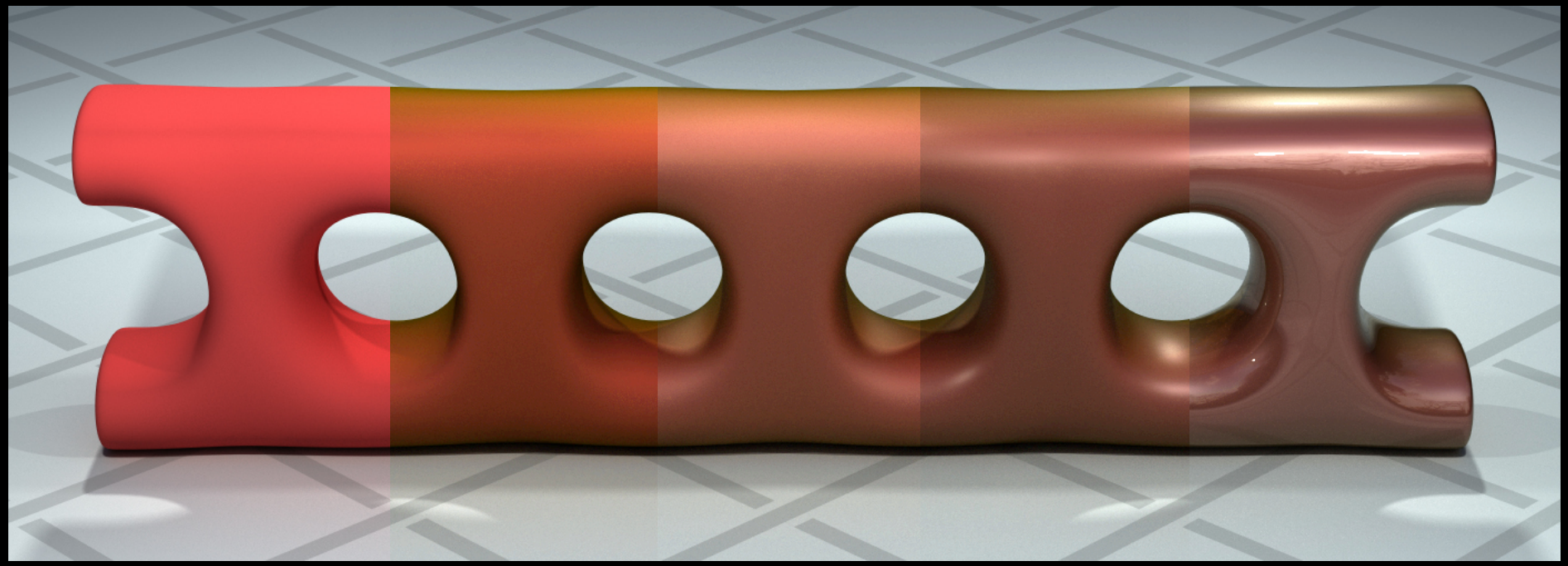
Measured Curve Factor

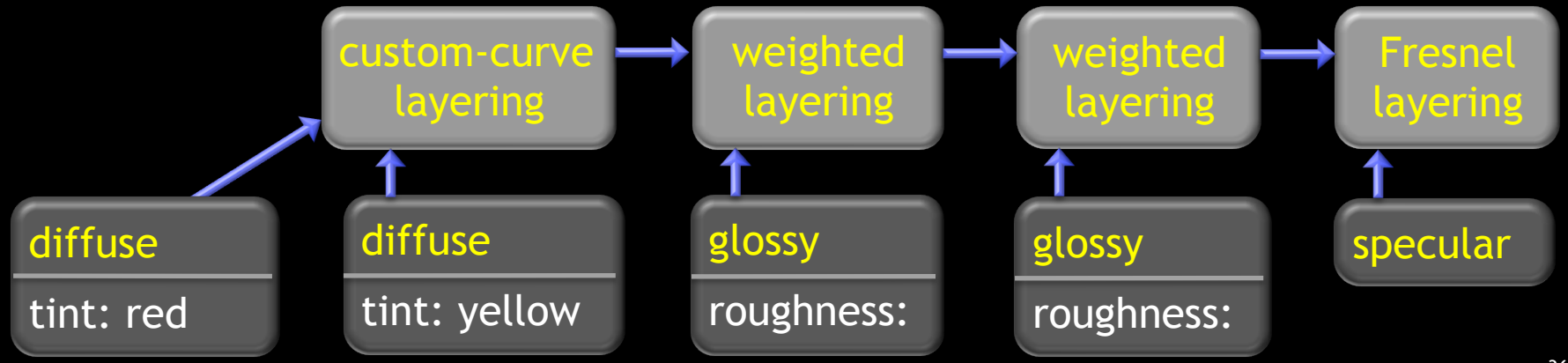
MDL Distribution Function Combiners

Normalized Mix
Clamped Mix
Weighted Layer

Fresnel Layer
Custom Curve Layer
Measured Curve Layer

MDL Layered Material Example




custom-curve layering
weighted layering
weighted layering
Fresnel layering
diffuse
tint: red
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

```
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

```
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

```
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 );
```

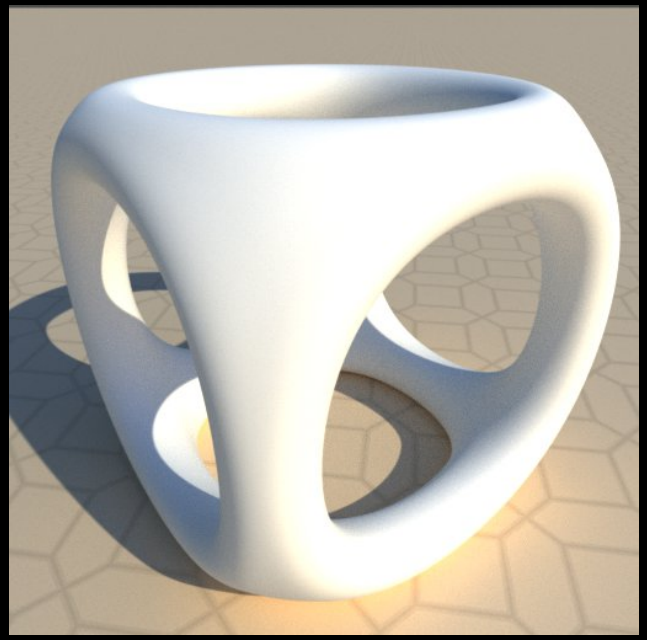
Defining a Material Using MDL

Creating new materials

```
material mymaterial ( )
    = material      ( );
```

Defining a Material Using MDL

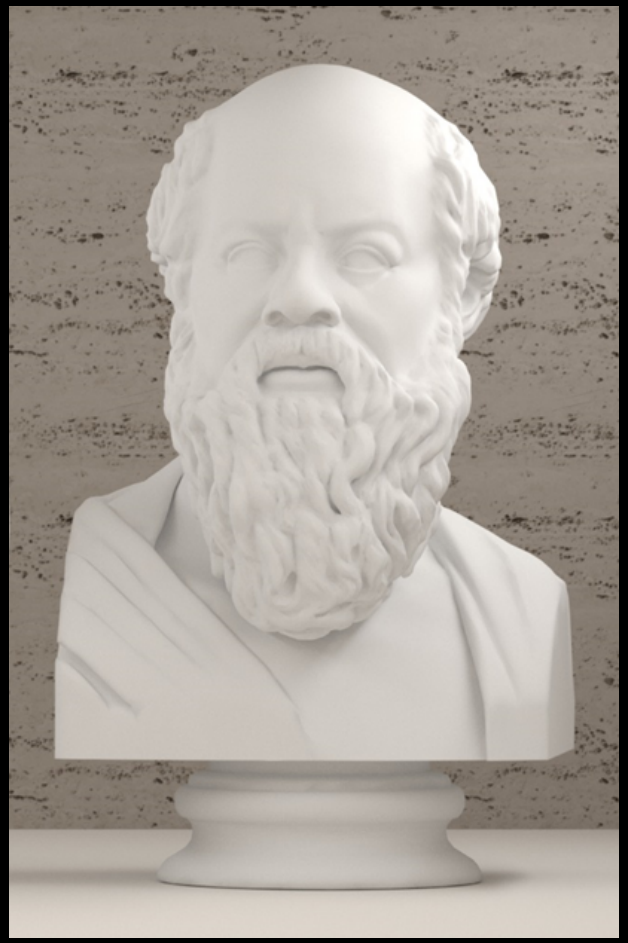
```
material plaster( )
    = material(
        surface: material_surface(
            scattering: df::diffuse_reflection_bsdf()
        )
    );
```



Defining a Material Using MDL

New materials can have parameters

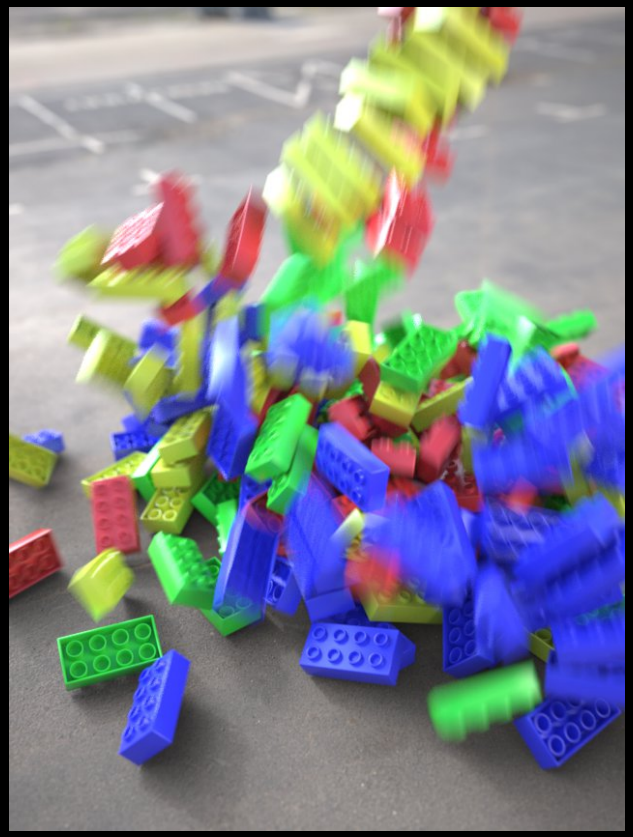
```
material plaster ( color plaster_color = color(.7))
    = material(
        surface: material_surface (
            scattering: df::diffuse_reflection_bsdf (
                tint: plaster_color
            )
        )
    );
```



Defining a Material Using MDL

Create complex materials by layering

```
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 )
        )
    )
);
```

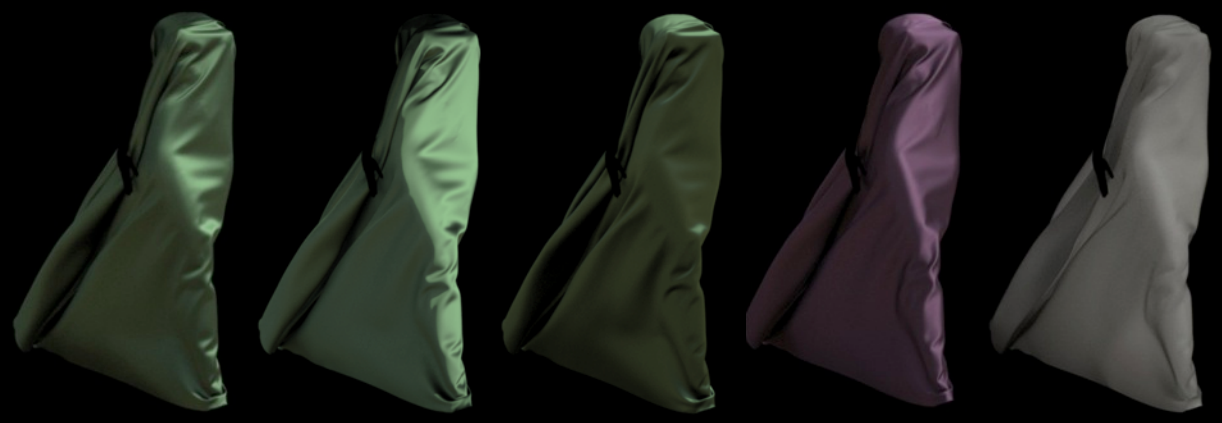


MDL Handbook

www.mdlhandbook.com

60+ new pages since SIGGRAPH 2015

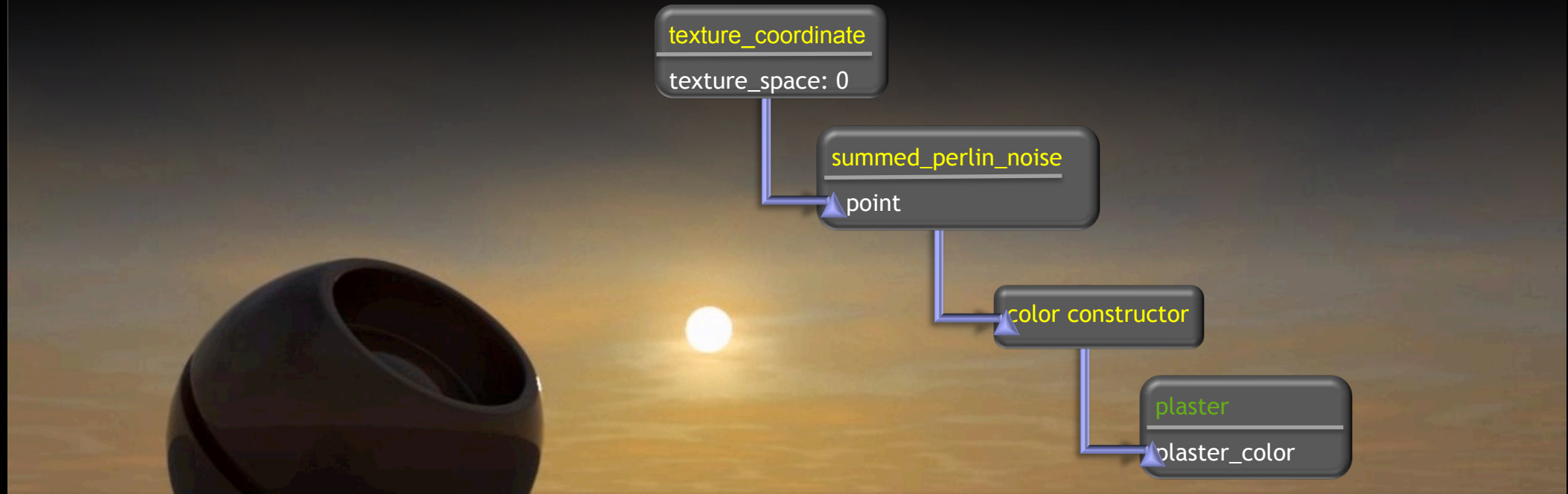
Example



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

```
type-of-return-value function-name ( parameters )
{
    statements
}
```

Defining a Function Using MDL

Function access render state through standard modules

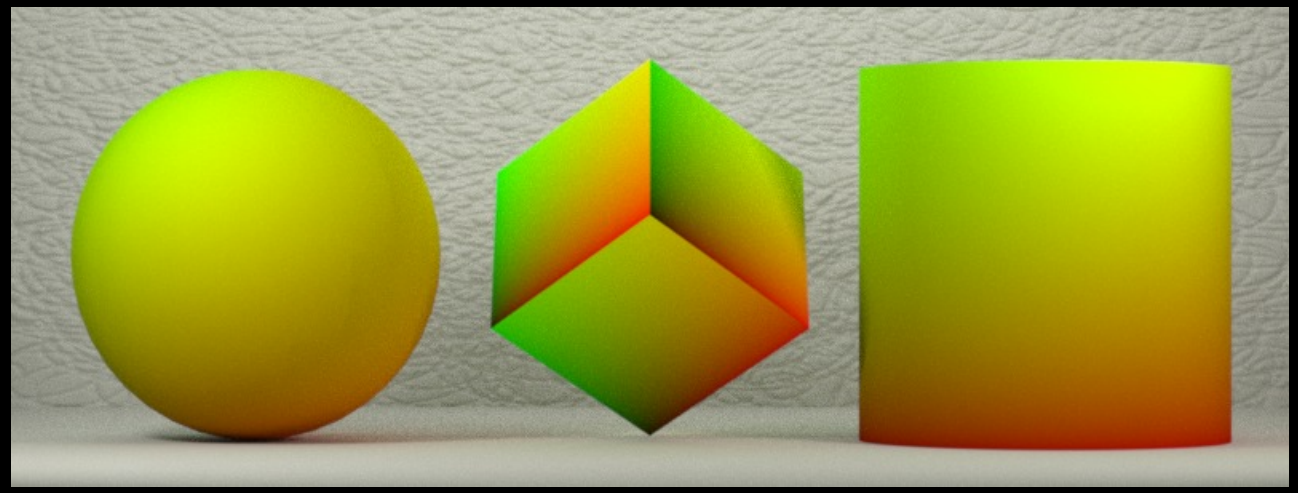
```
color uv_as_color()
{
        return color( state::texture_coordinate(0) );
}
```

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

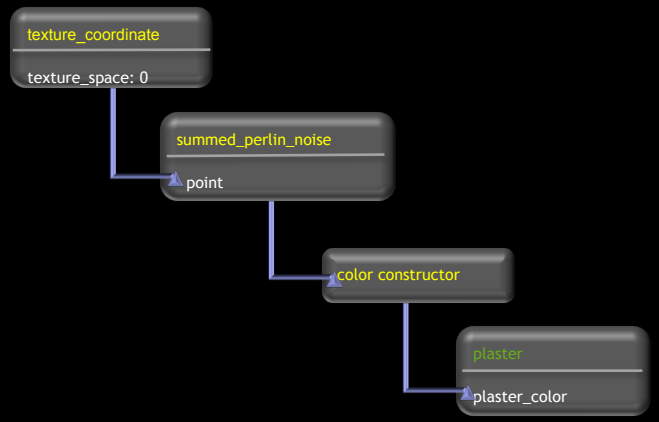
```
float summed_perlin_noise (
     float3 point,
     int level_count=4,
     float level_scale=0.5,
     float point_scale=2.0,
     bool turbulence=false)
{
     float scale = 0.5, noise_sum = 0.0;
     float3 level_point = point;
     for (int i = 0; i < level_count; i++)
     {
          float 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;
}
```

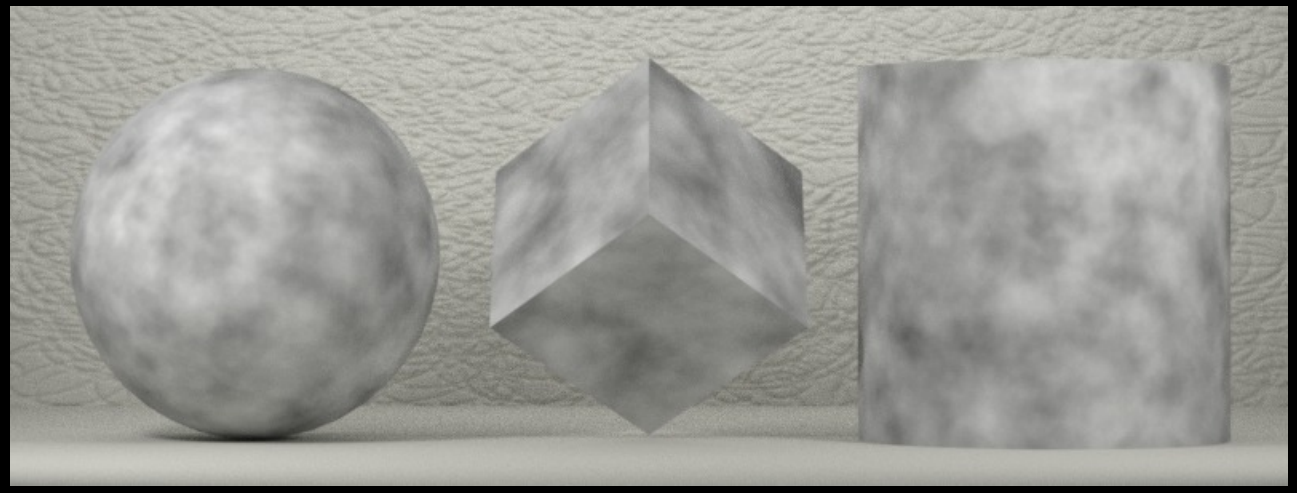
See
www.mdlhandbook.com
For implementation

Defining a Function Using MDL

Call graph of functions substitute shader graphs

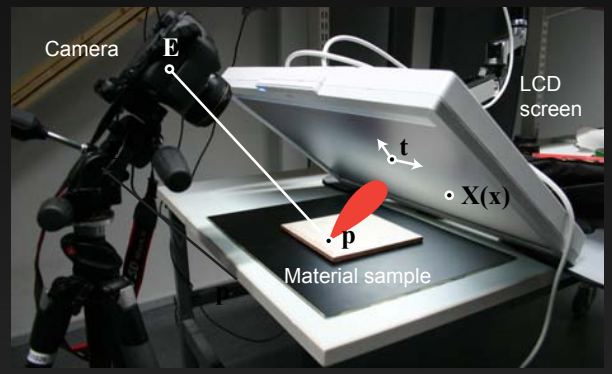
```
material perlin_noise_material()
= plaster(
    plaster_color: color(
        summed_perlin_noise(
            point: state::texture_coordinate(0)
        )
    )
)
```



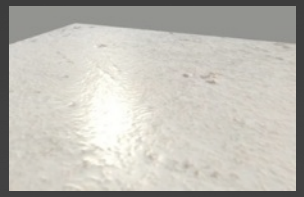


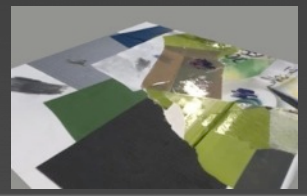
MDL Measured Materials

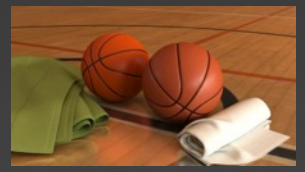
Fast scan for believable materials

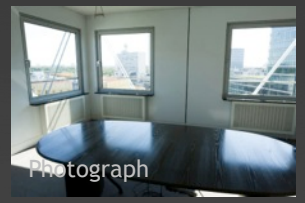


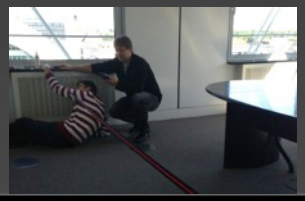
Quantitative measurements for predictive rendering











Spatially Varying (SV)BRDF

- Analytic material model
- Measurement drives model parameters

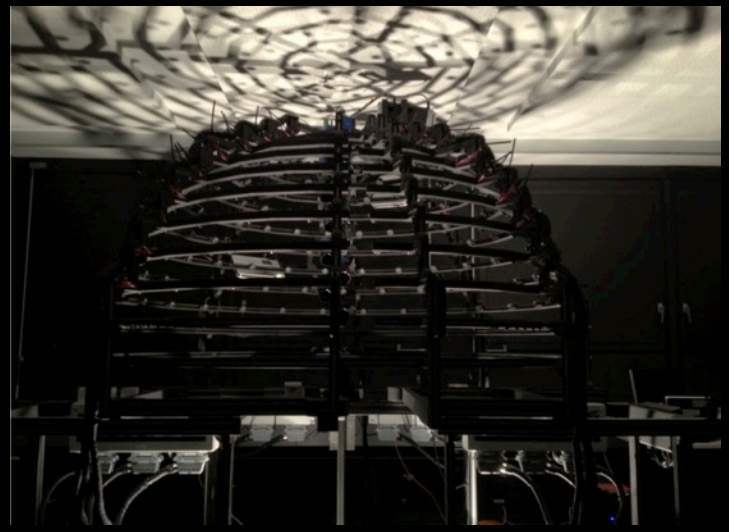
Practical SVBRDF Capture In The Frequency Domain, SIGGRAPH 2013

Miika Aittala and Jaakko Lehtinen, Aalto University, NVIDIA Research

Tim Weyrich, University College London

BTF Measurement Technology from X-Rite

Total Appearance Capture (TAC)



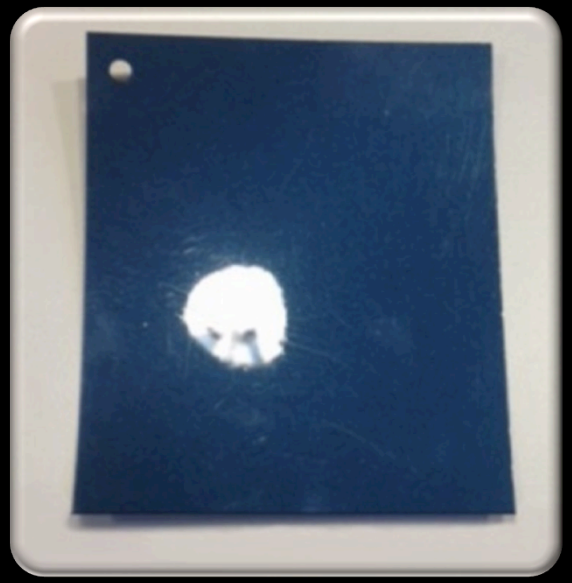
Dome TAC

TAC7 Prototype

- Measurement stored in Appearance eXchange Format (AxF)
- Iray supports SVBRDF representation and carpaint of AxF 1.3

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

MDL

Additional Benefits

Physically-based materials are an easy-to-use paradigm

Allows simple compilers and early optimizations

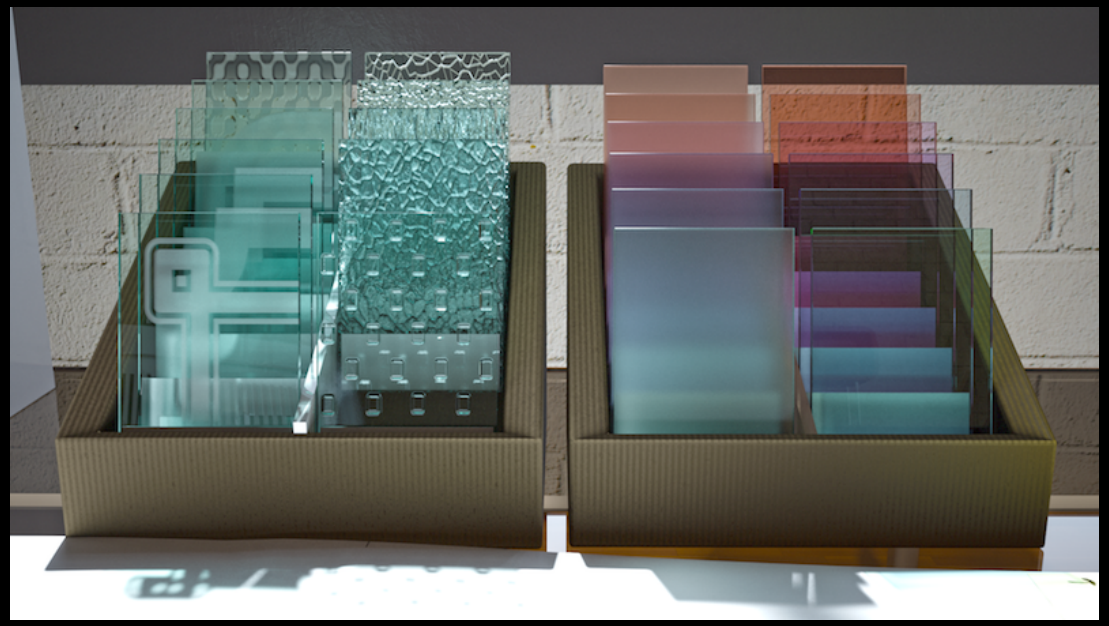
Supports modern rendering algorithms

Enables fast renderers, especially on parallel architectures

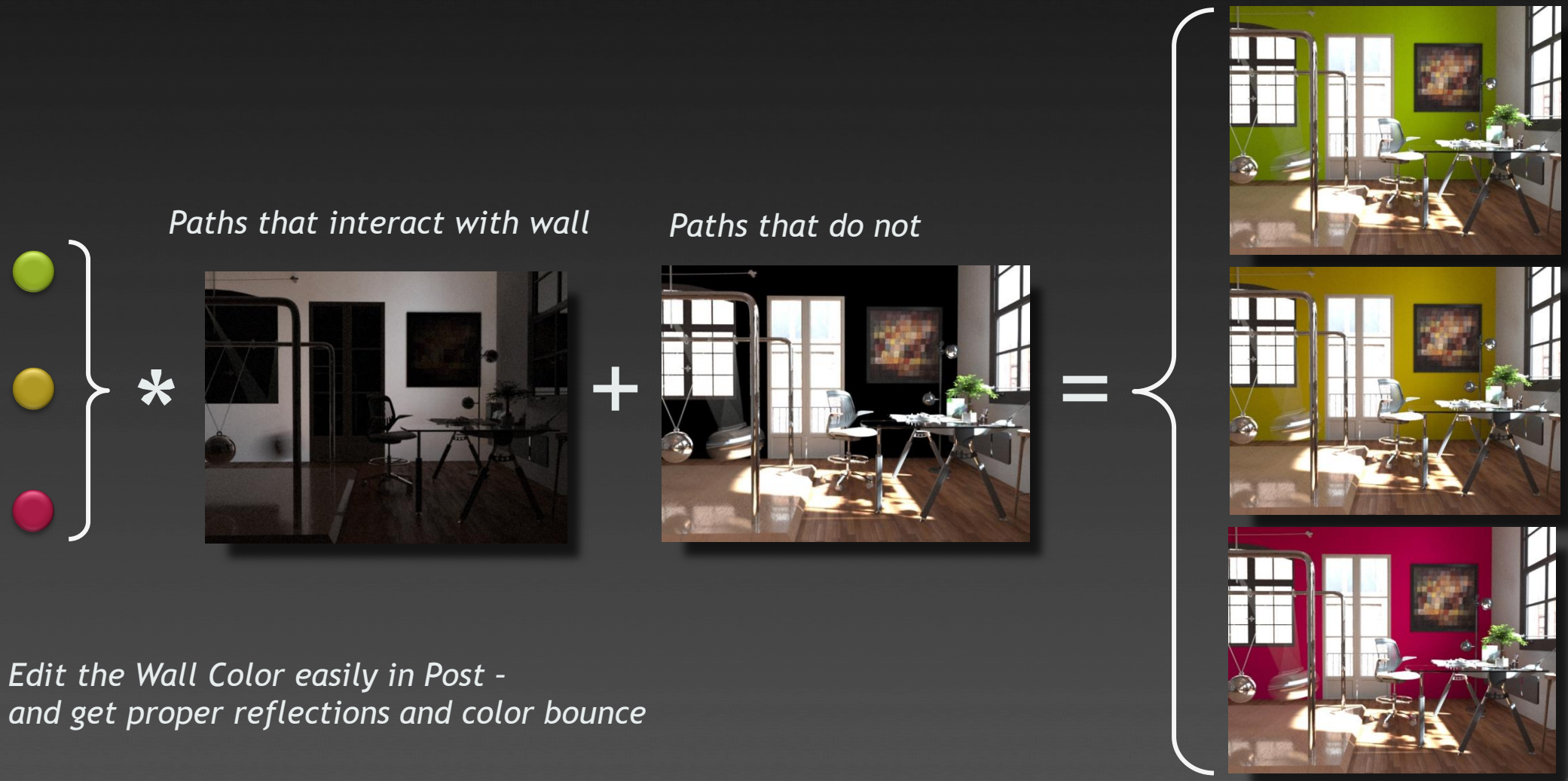
GPU friendly

Supports material catalogs

Complements Light path expressions



Light Path Expressions

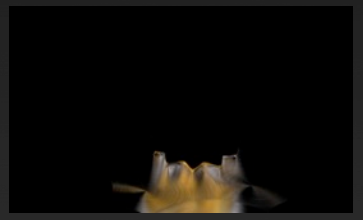


Edit the Wall Color easily in Post -
and get proper reflections and color bounce

MDL Complement Light Path Expressions

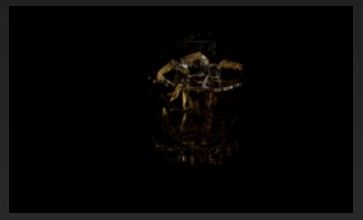
LPEs can select individual DF components

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

Jan 2013

MDL 1.0, shipment with Iray 2013

May 2014

Made Specification public

2015

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

May 2012

Kick-off of MDL Spec

Jan 2014

MDL 1.1, support for measured data

Bunkspeed, Catia start using MDL

Jan 2015

MDL 1.2, resource handling, units

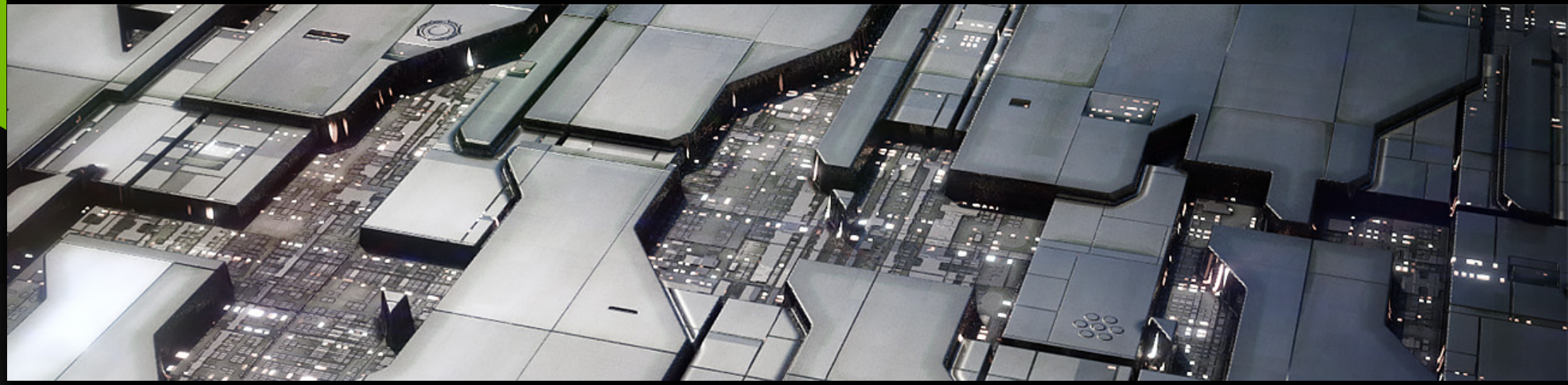
Support in mental ray, exposed in Autodesk 3ds Max, Maya

2016

Vray, ESI IC.IDO, MDL 1.3

MDL in Commercial Products

MDL in Substance Designer



Focus on Material Exchange

Freely choose where to author material content

create

modify

consume

NVIDIA vMaterials

~800 MDL Materials Verified for Accuracy - FREE TO USE

Become Part of the Eco-System

Integrate MDL enabled renderer

- MDL is included

Write your own compiler

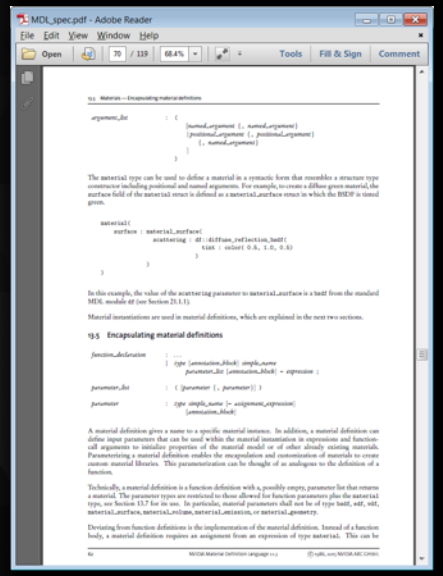
- Based on the freely available MDL Specification

License the MDL SDK

- Contact us for licensing information

Become Part of the Eco-System

Write your own compiler



MDL Specification can be downloaded @

http://www.nvidia-arc.com/products/iray/mdl-materials.html

MDL conformance test suite

Syntactic conformance tests - available at request

Semantic conformance tests

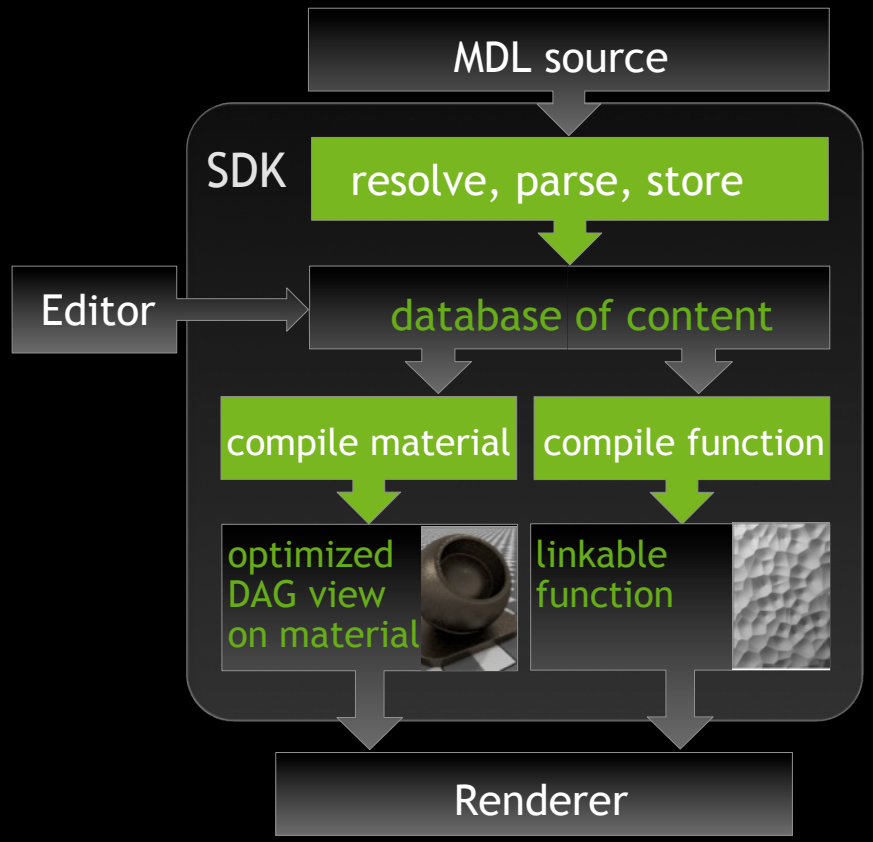
Become Part of the Eco-System

License the MDL SDK

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

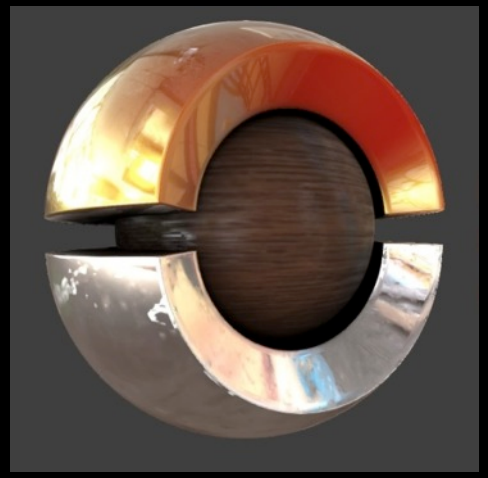
Contact us for details on availability and licensing

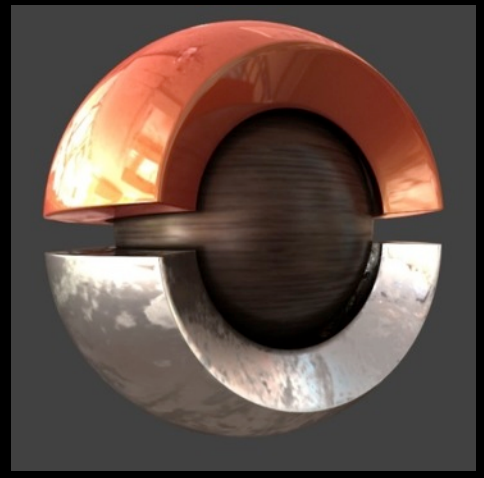


MDL SDK in Use

IC.IDO

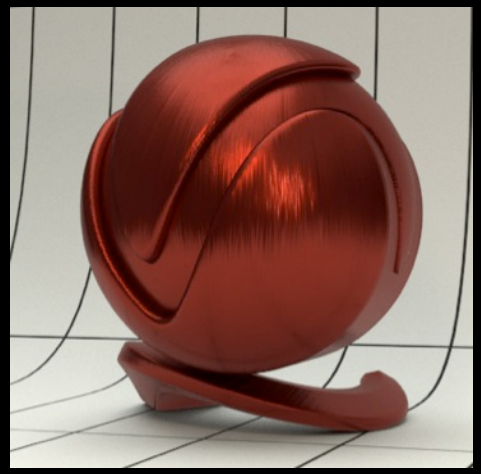
esri

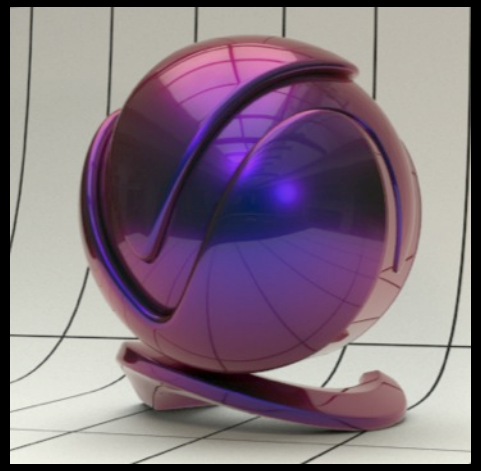




CHAOSGROUP

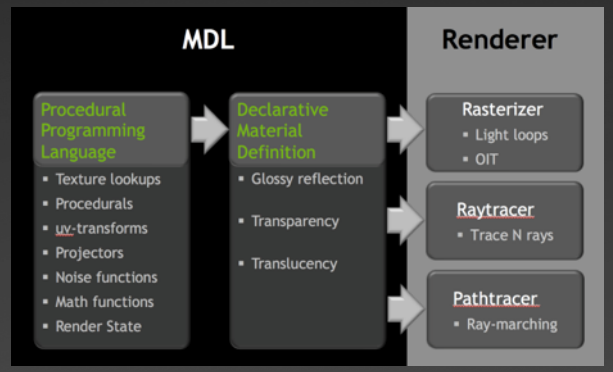
V-ray





MDL Takeaways

What is MDL



MDL Eco-system

- vMaterials

Become part of the Eco-system

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

More Information

http://www.nvidia.com/object/material-definition-language.html

Demos and Talks at NVIDIA booth on the show floor

The Great Wizard MEC4D 2015

Iray Photoreal in Daz 3D