Sharing Physically Based Materials Between Renderers with MDL

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

July 29, SIGGRAPH 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

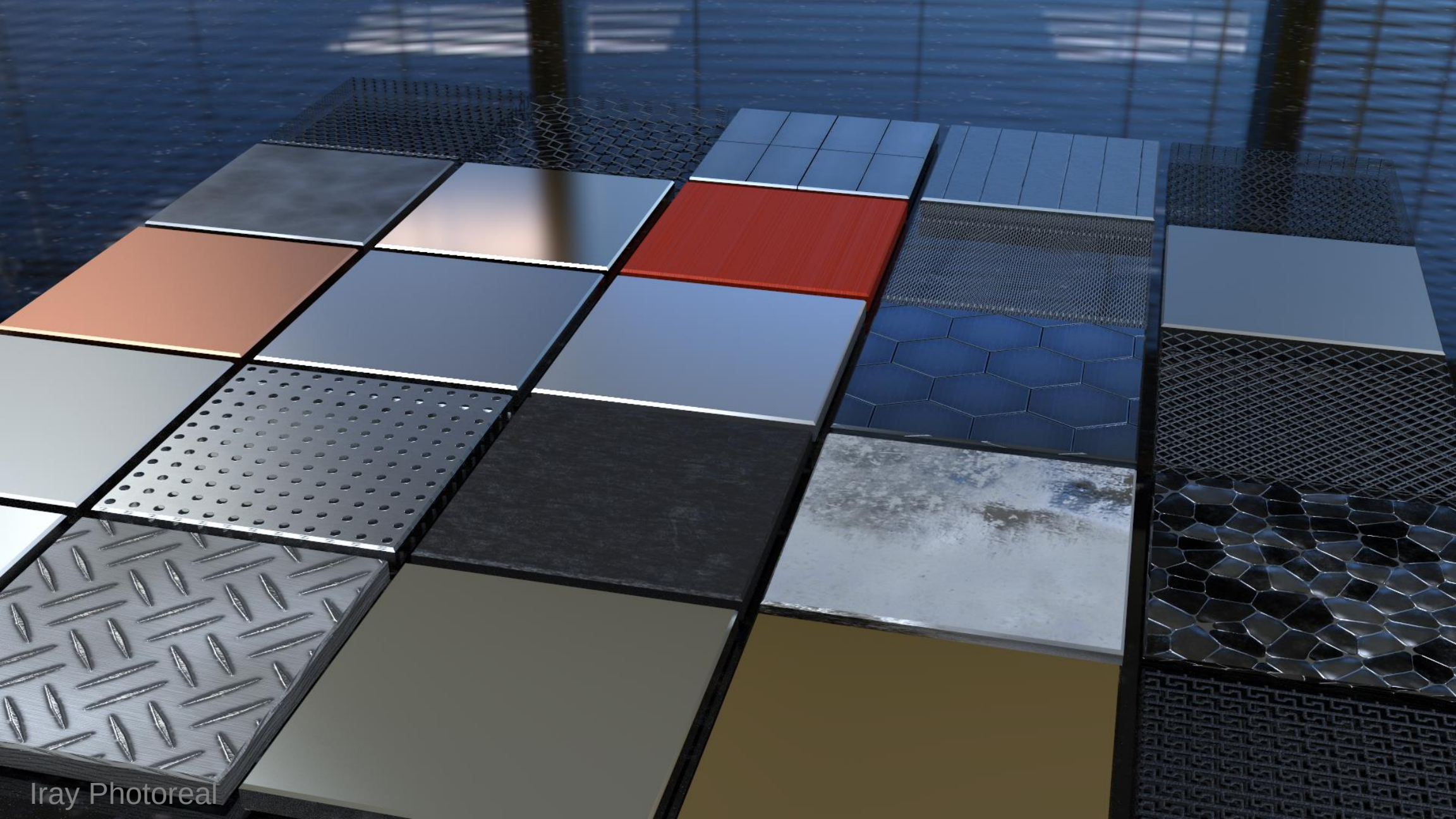
NVIDIA® MDL

The **NVIDIA Material Definition Language (MDL)**

is technology developed by NVIDIA

to define **physically-based** materials

for physically-based rendering solutions.


Iray Photoreal


Iray Photoreal


B TZ 2012
D
Iray Photoreal

courtesy Harley Davidson

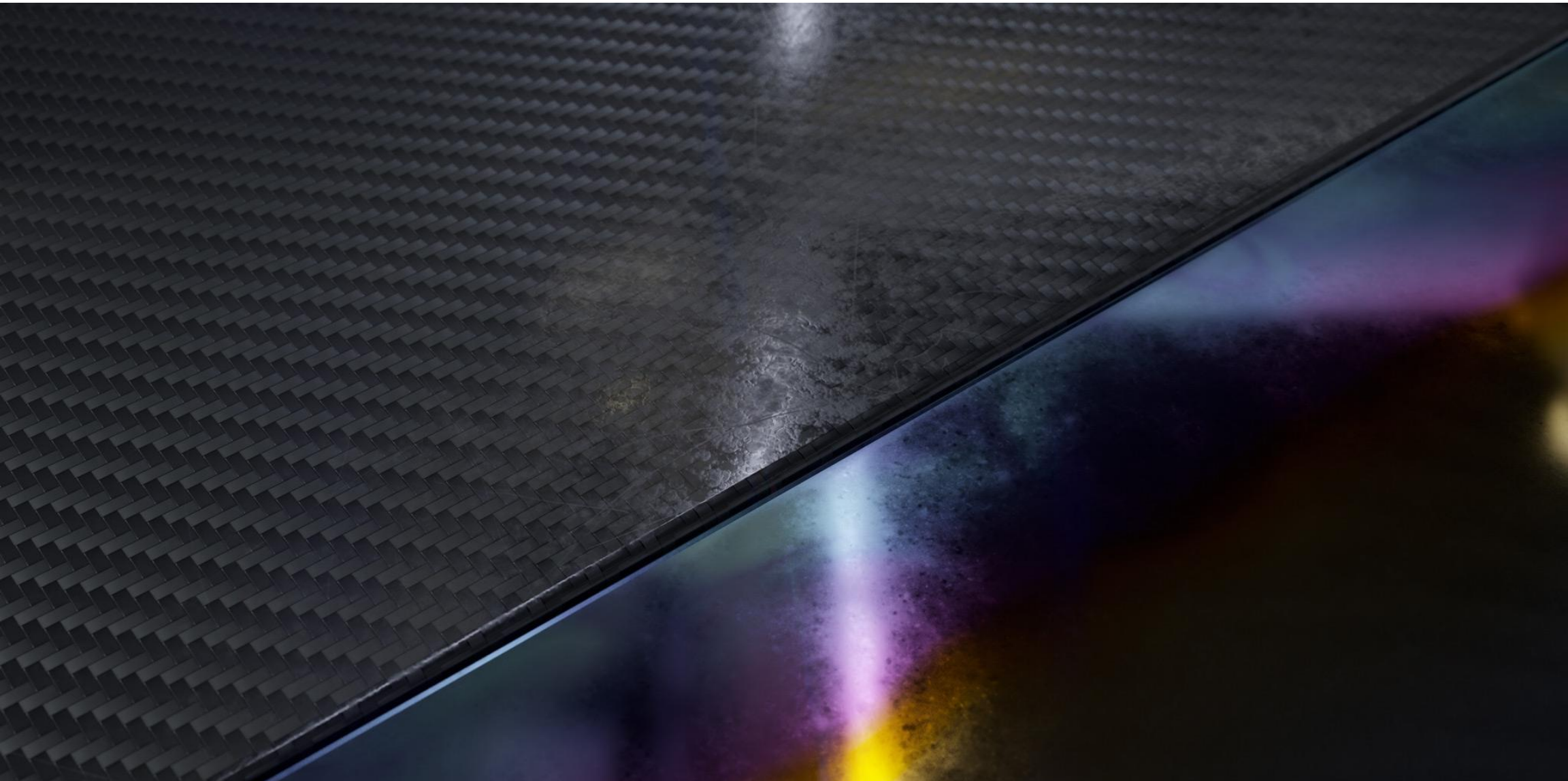
Photographer: Simon Puschmann/Rendering and Retouching: [zerone]


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


Iray Photoreal

Mark Foreman Malachite with Chrysocolla

SUBSTANCE DESIGNER



Substance Designer

Matching the Appearance of a Single Material Within Different Rendering Techniques

One Scene for Different Renderers

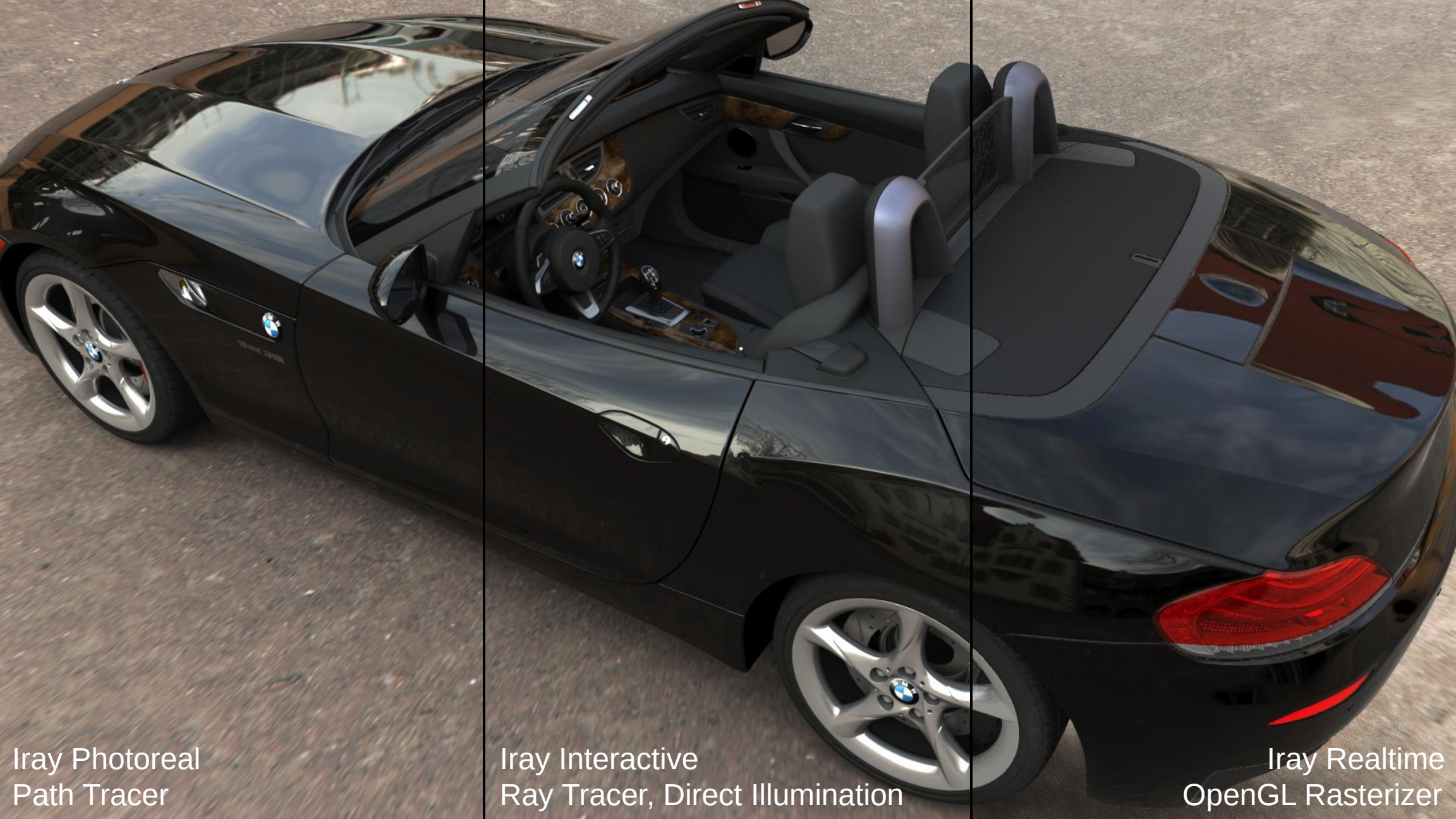
Realtime Rasterizer

Interactive Raytracer

Pathtracer

Share scene and MDL materials for a **consistent look**

Switching renderers with no scene modifications


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

Material Definition

- Glossy reflection
- Transparency
- Translucency

Material Implementation

- Light loops / trace N rays
- OIT / ray-continuation
- Ray marching

NVIDIA MDL

Renderer

Procedural Programming Language

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

Declarative Material Definition

- Glossy reflection
- Transparency
- Translucency

Rasterizer

Light loops / OIT

Raytracer

Trace N rays

Pathtracer

Ray-marching

NVIDIA MDL

Procedural Programming Language → **Declarative Material Definition**

Renderer

Rasterizer

Light loops / OIT

Raytracer

Trace N rays

Pathtracer

Ray-marching

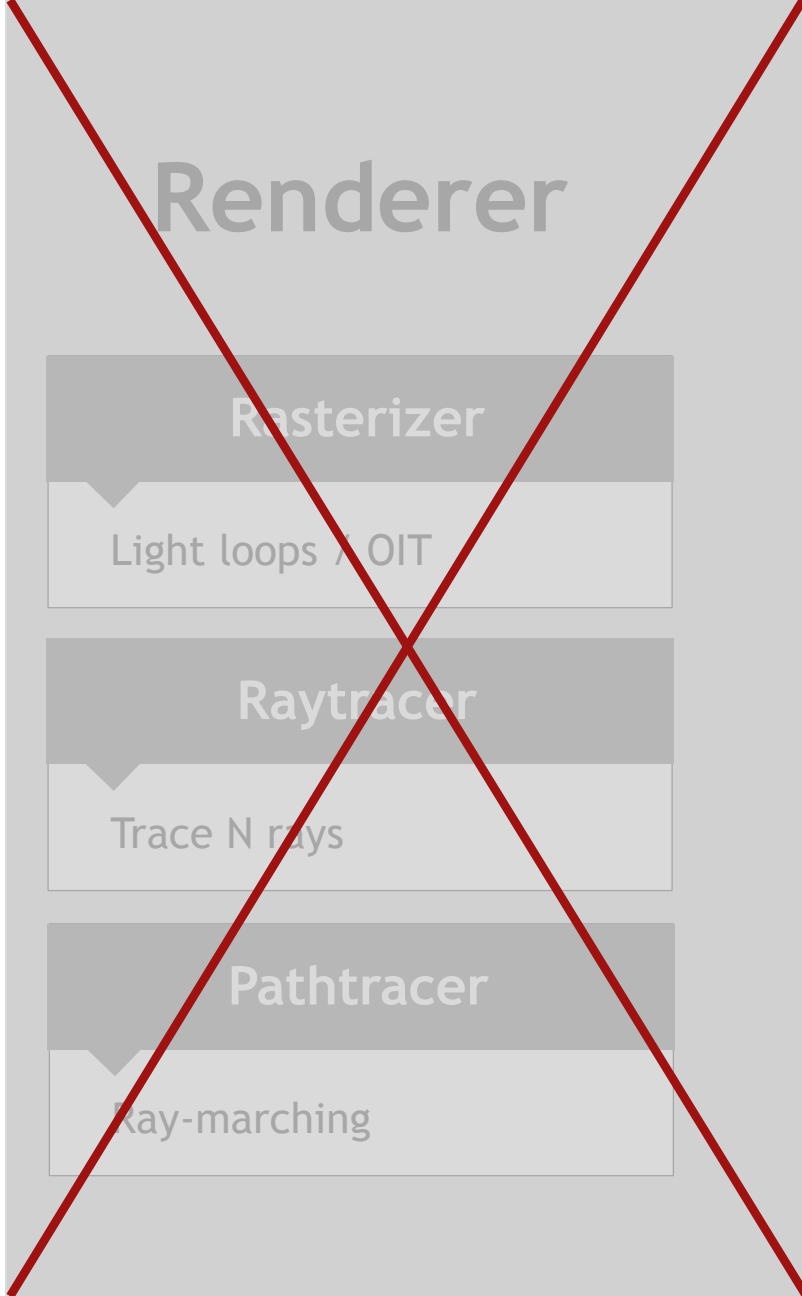
NVIDIA MDL

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

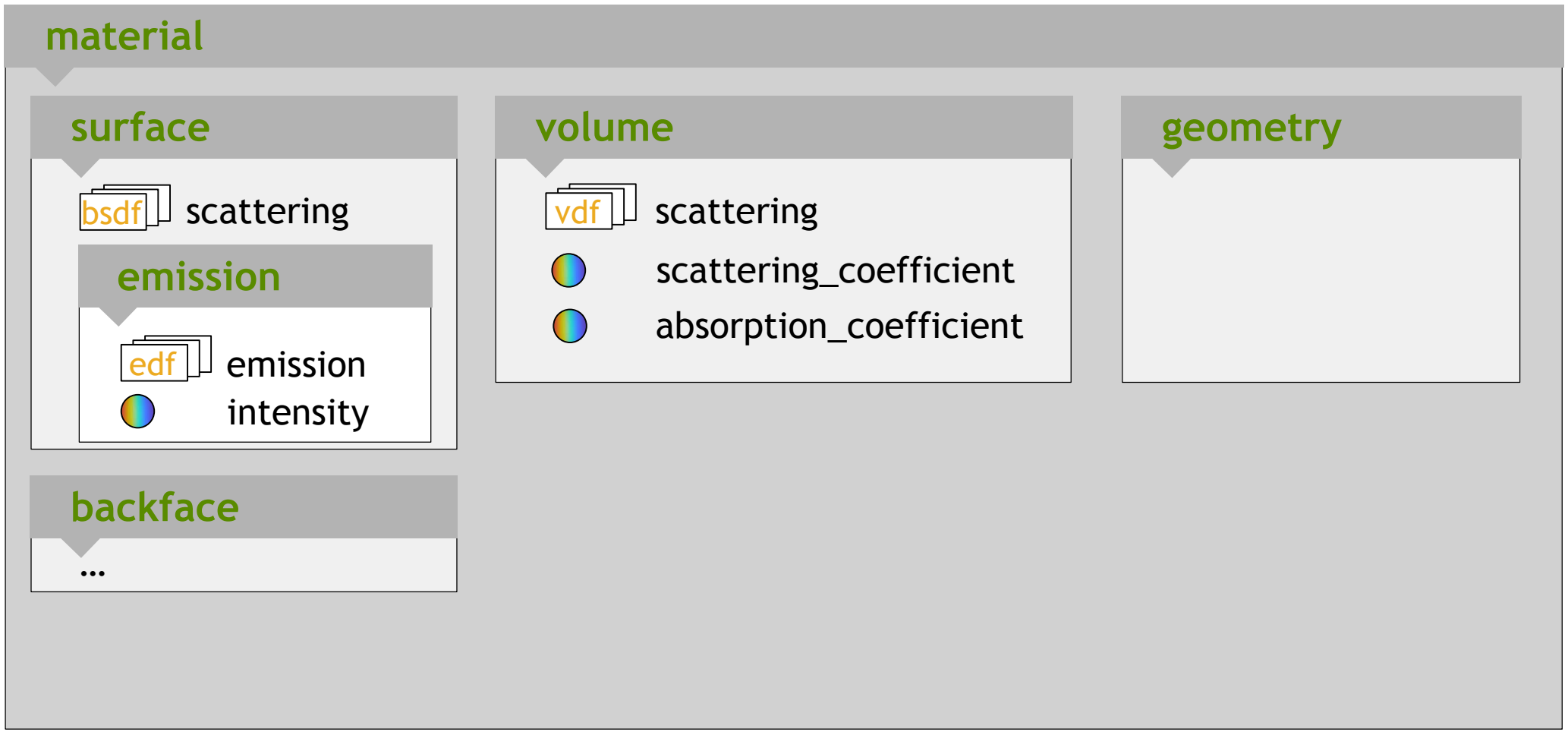


MDL Material Model

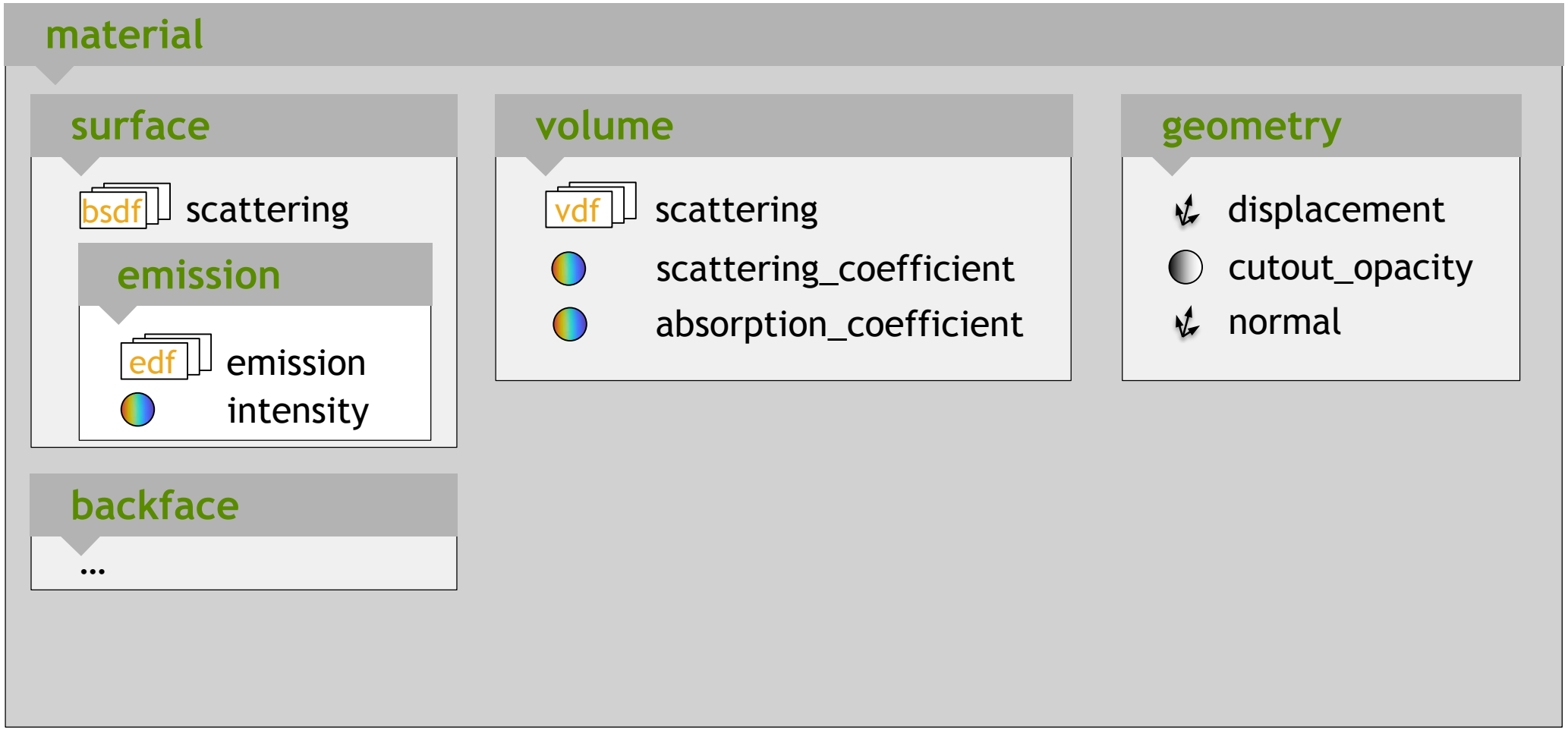
MDL Material Model

MDL Material Model

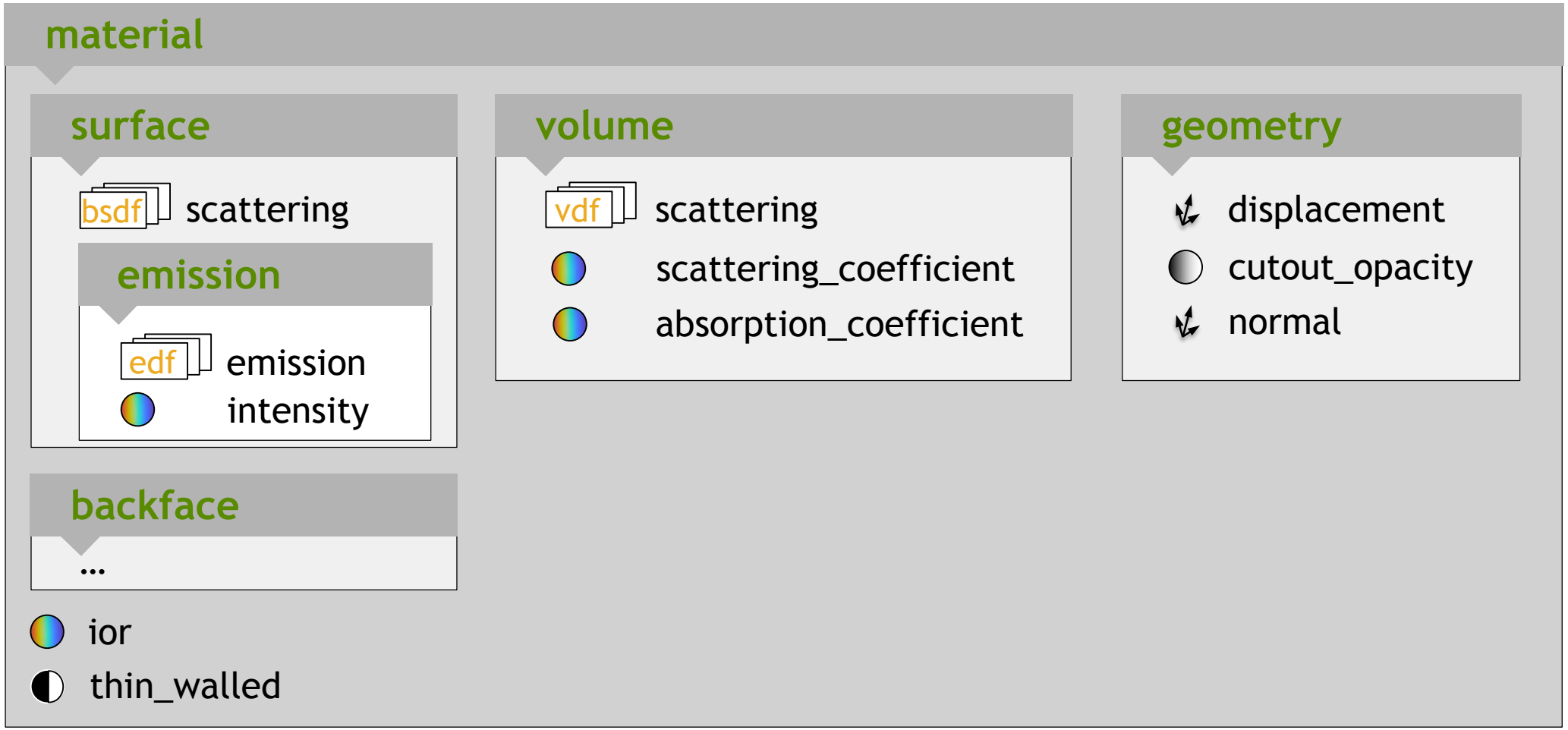
MDL Material Model



MDL Material Model



MDL Material Model



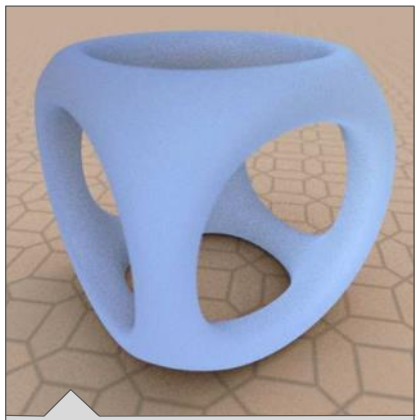
MDL Elemental Distribution Functions

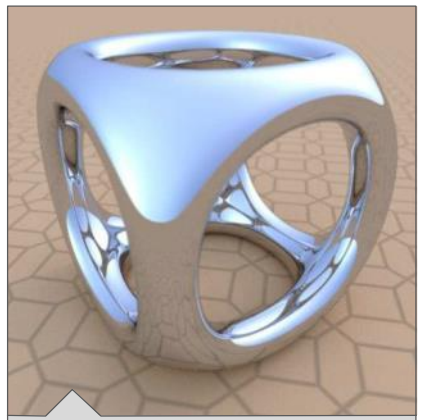
Bidirectional
Scattering
Distribution
Functions

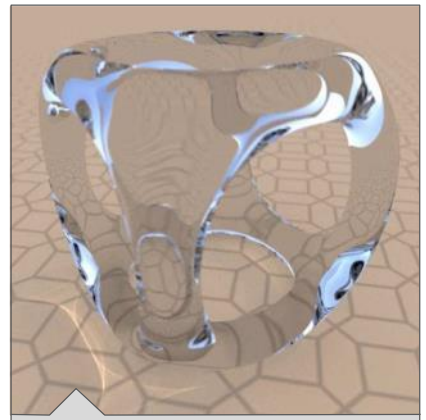

Diffuse Reflection


Diffuse Transmission


Glossy (various)


Backscatter Glossy


Specular Reflection

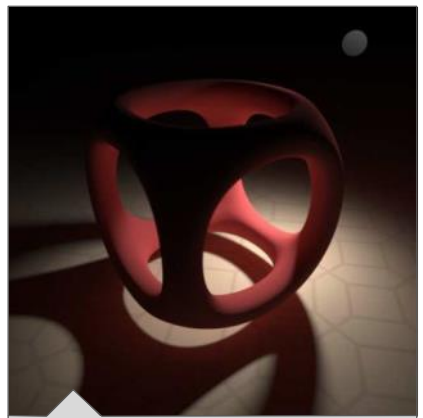

Spec. Refl.+Transm.


Measured BSDF

MDL Elemental Distribution Functions

Emissive
Distribution
Functions

Diffuse



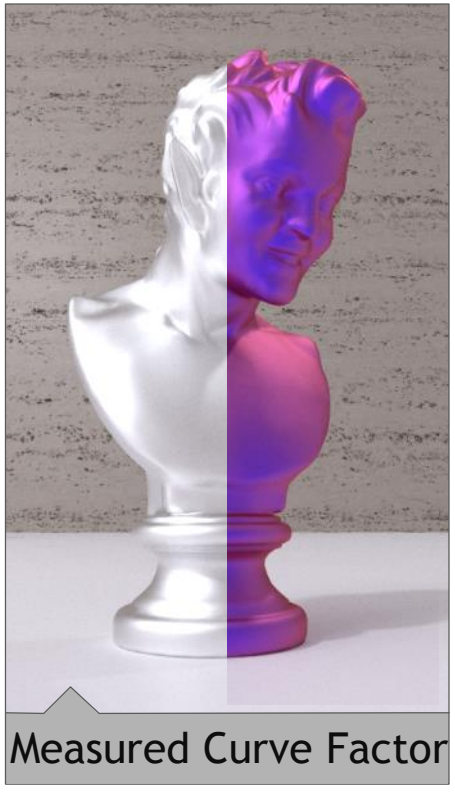
Spot

IES Profile

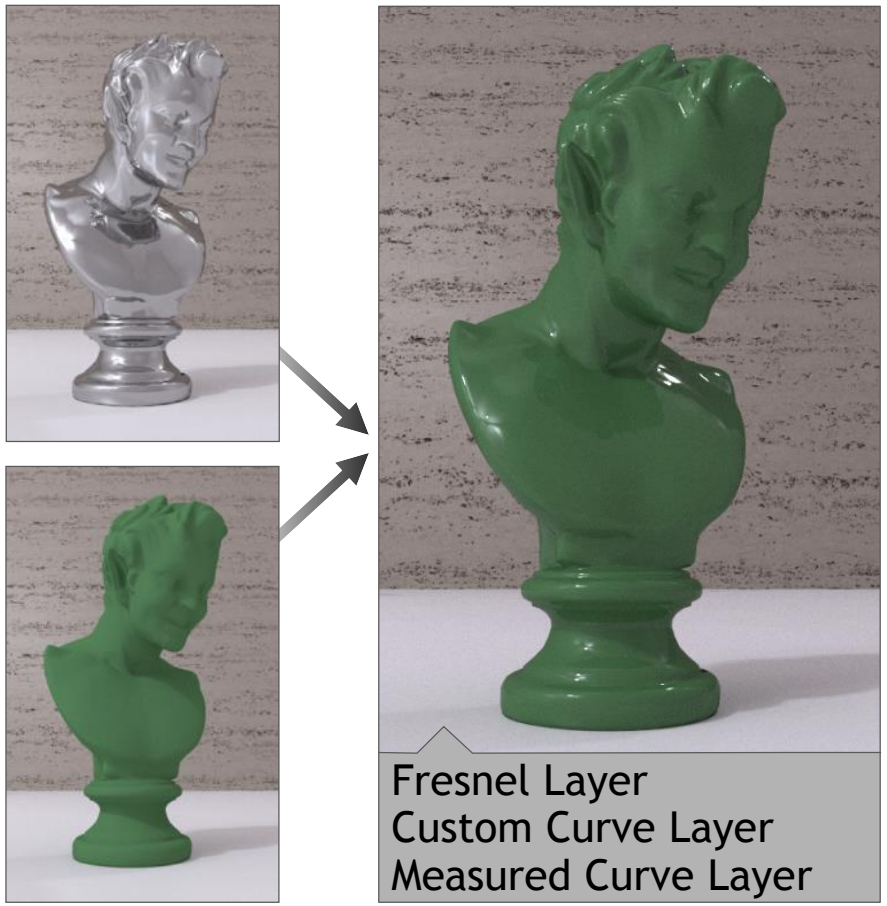
Volume
Distribution
Functions

Henyey-Greenstein

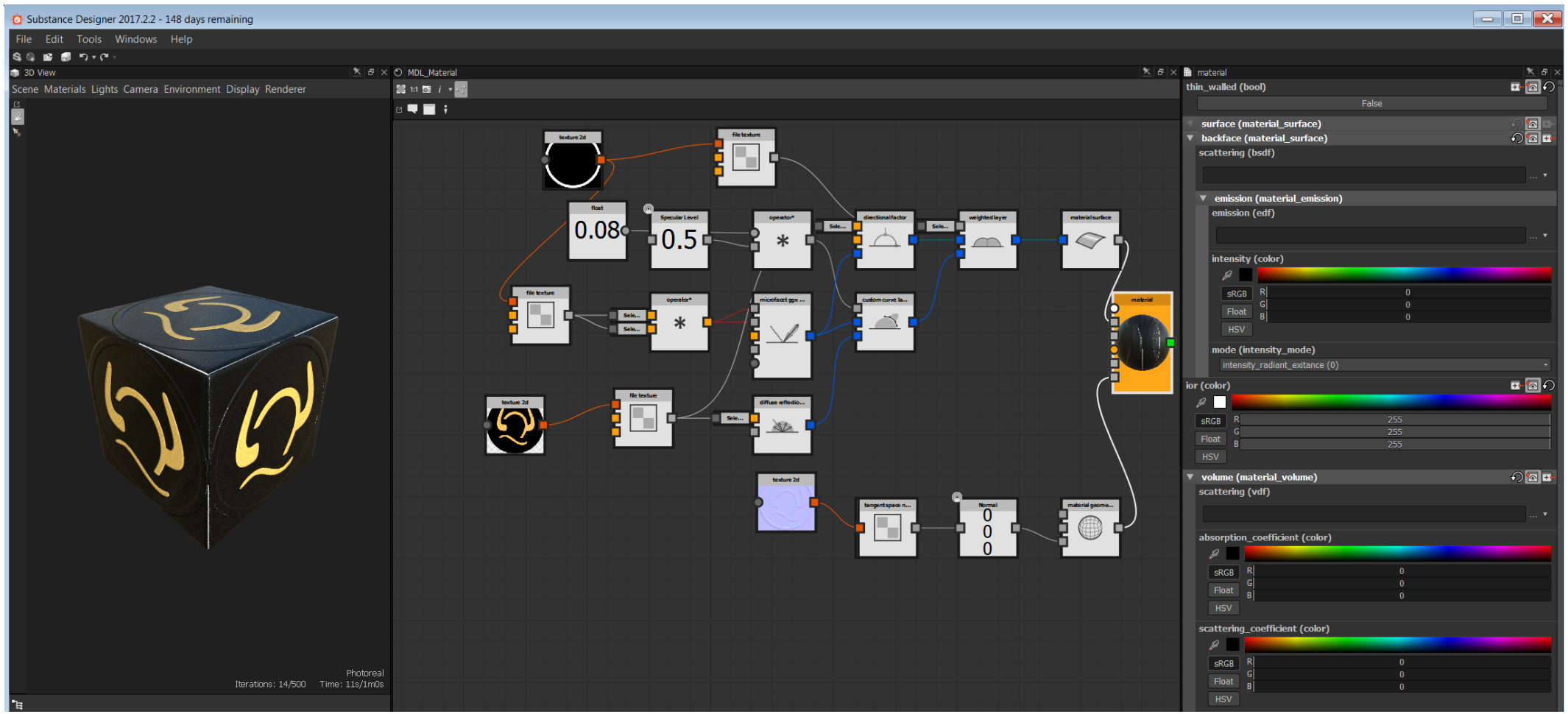
MDL Distribution Function Modifiers



MDL Distribution Functions Combiners



MDL Layered Material Example



Defining Physically-based Materials With Source Code

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();
```

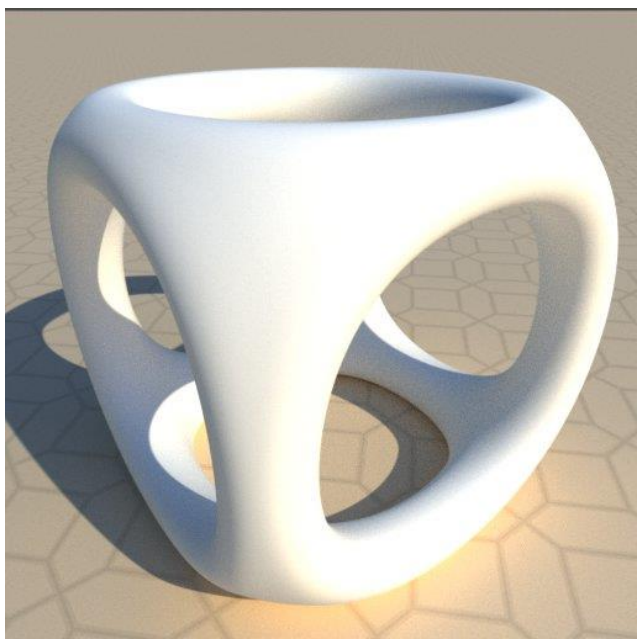
Defining a Material Using MDL

Creating new materials

```
material name    ( material-parameters )
    = material   ( material-arguments );
```

Defining a Material Using MDL

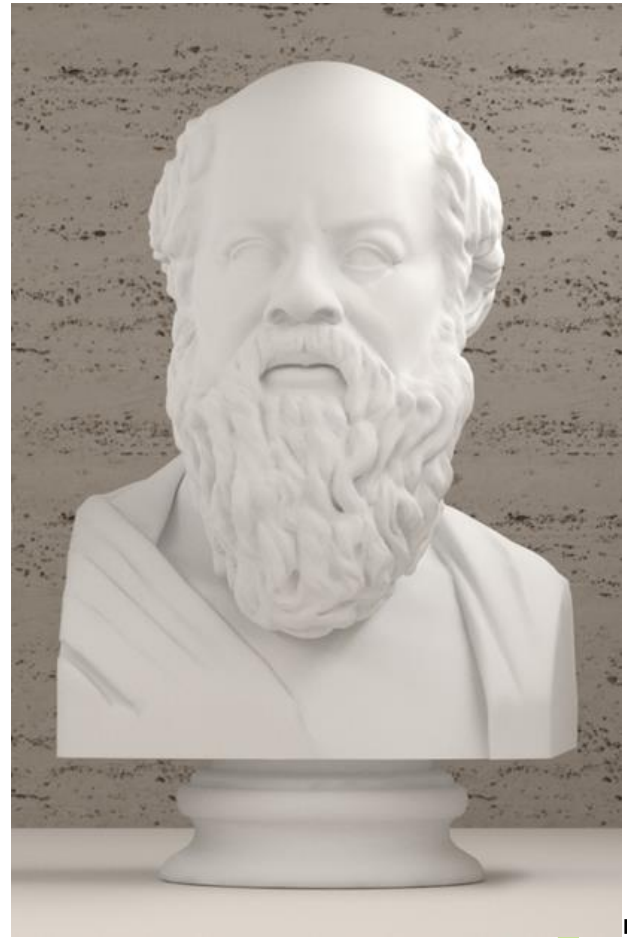
```
material plaster( )
    = material(
        surface: material_surface(
            scattering: df::diffuse_reflection_bsdf()
        )
    );
```



Defining a Material Using MDL

New materials can have parameters

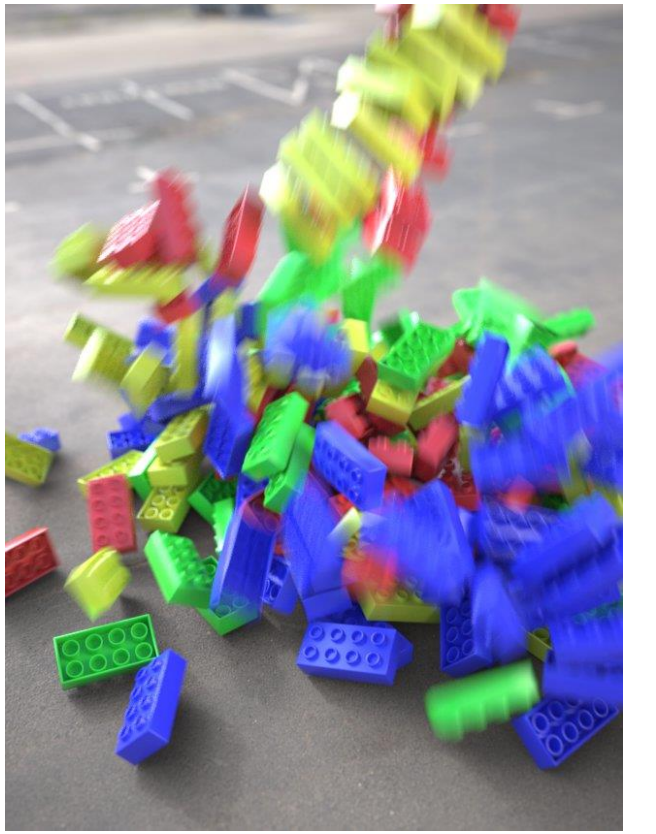
```
material plaster ( color plaster_color = color(.7))
    = material(
        surface: material_surface (
            scattering: df::diffuse_reflection_bsdf (
                tint: plaster_color
            )
        )
    );
```



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 )
        )
    )
);
```

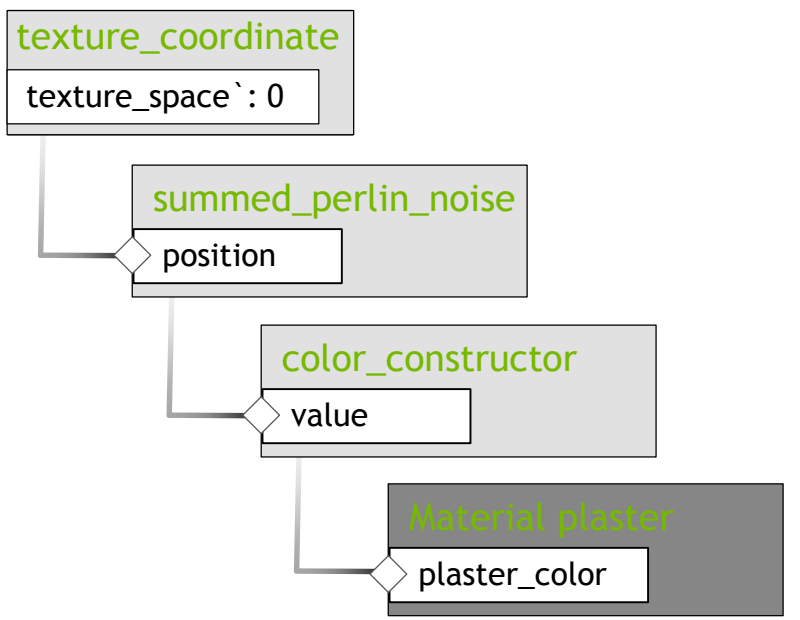


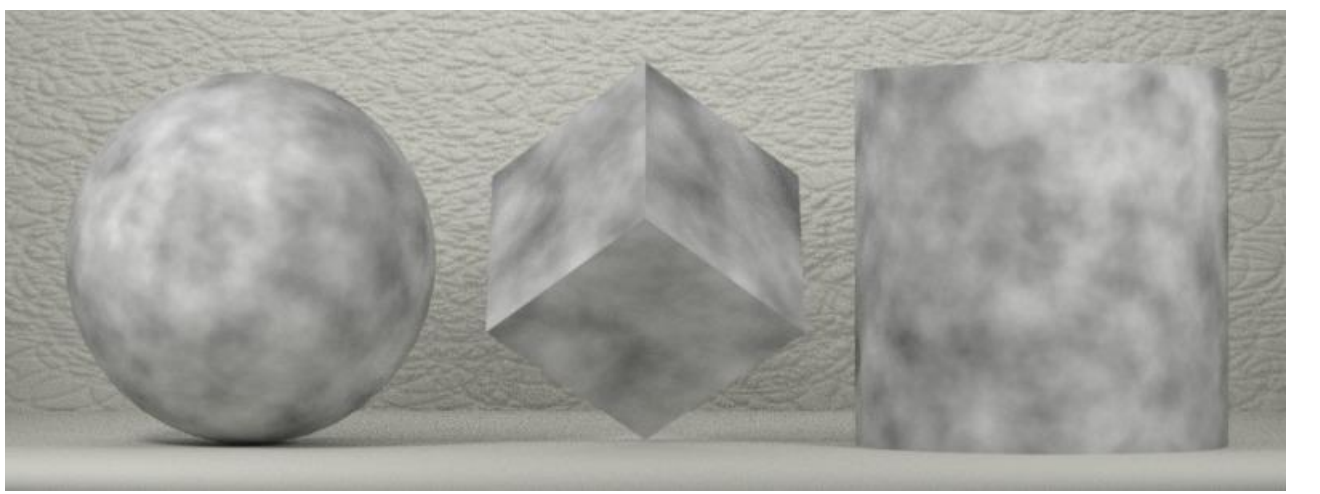
MDL Procedural Programming Language

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

Functions allow control flow like loops, switches, conditionals

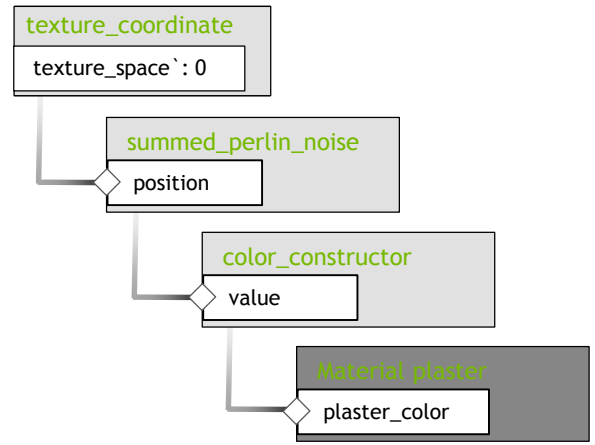
```
float summed_perlin_noise (
     float3 point,
     int level_count=4,
     float level_scale=0.5,
     float point_scale=2.0,
     bool turbulence=false)
{
     float scale = 0.5, noise_sum = 0.0;
     float3 level_point = point;
     for (int i = 0; i < level_count; i++)
     {
          float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

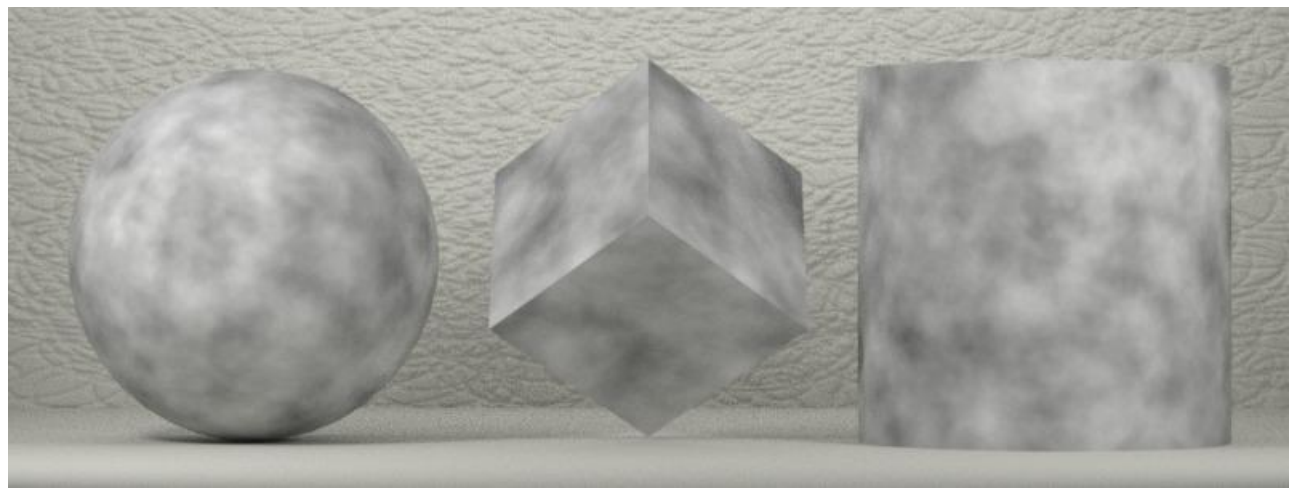
MDL Handbook

Defining a Function Using MDL

Call graph of functions substitute shader graphs

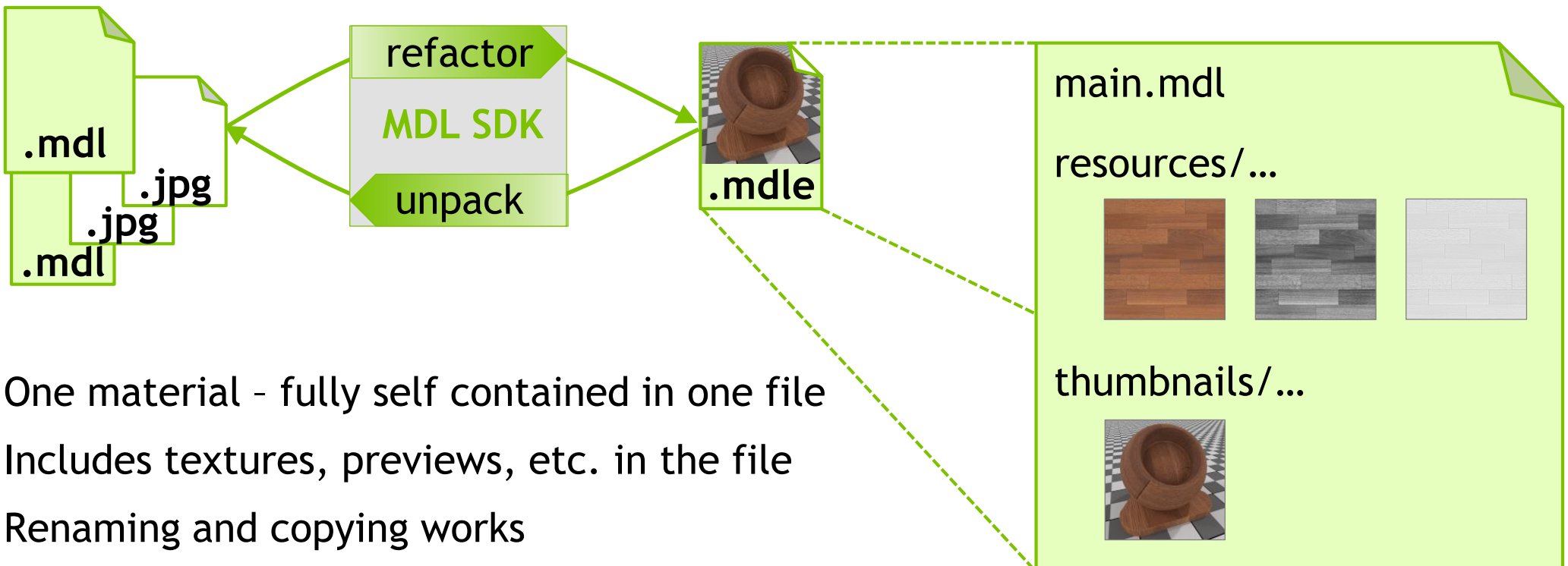
```
material perlin_noise_material()
= plaster(
    plaster_color: color(
        summed_perlin_noise(
            point: state::texture_coordinate(0)
        )
    )
)
```





MDL 1.5

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

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

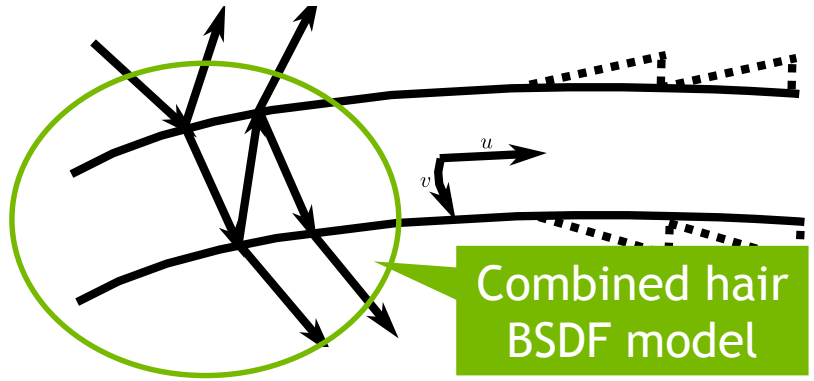
C:\Users\%USERNAME%\Documents\mdl\	MDL search path
nvidia\vMaterials\fr.xlf	French vMaterial package XLIFF file
nvidia\vMaterials\AEC\Glass\Mirror_fr.xlf	French Mirror module XLIFF file

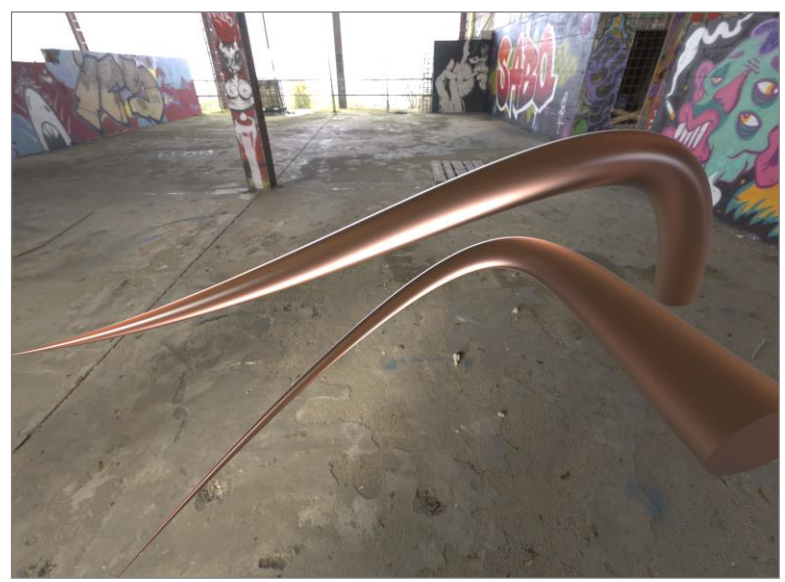
MDL 1.5

Hair shading

```
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;
};
```

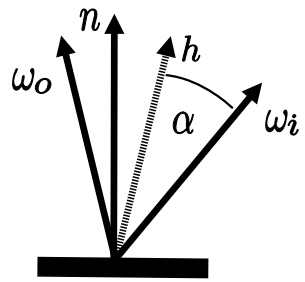


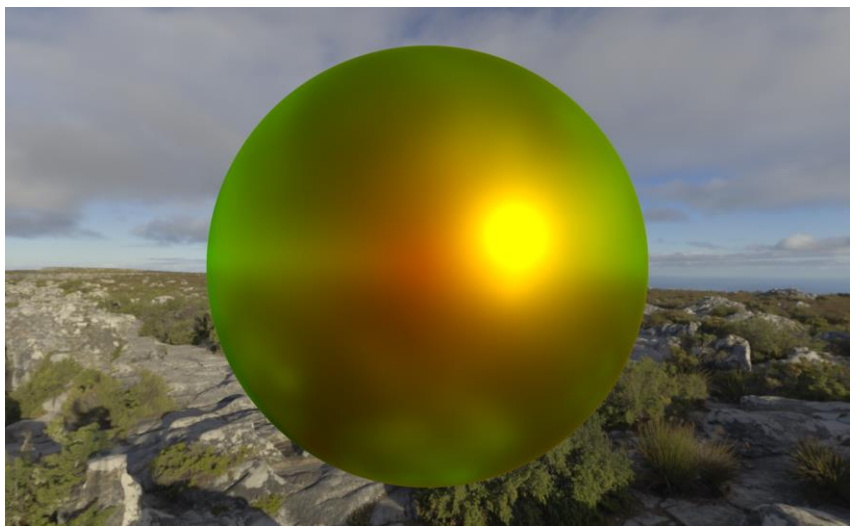


MDL 1.5

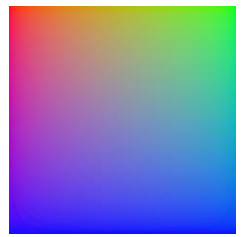
Microfacet coloring to support flip-flop car paints and more

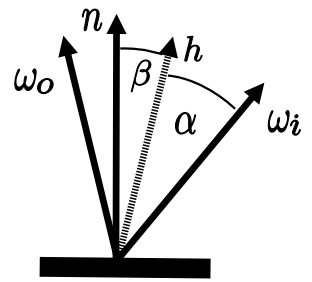
1D measured curve (MDL >=1.4)

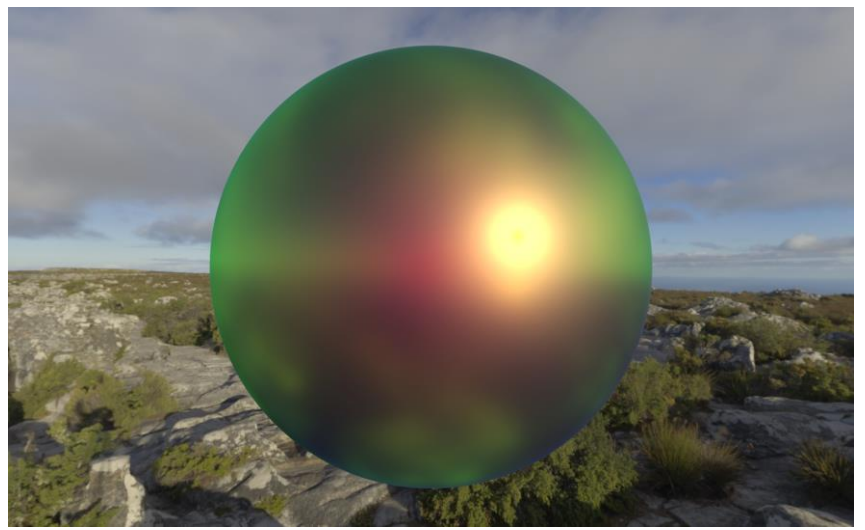




2D measured curve (new in MDL 1.5)

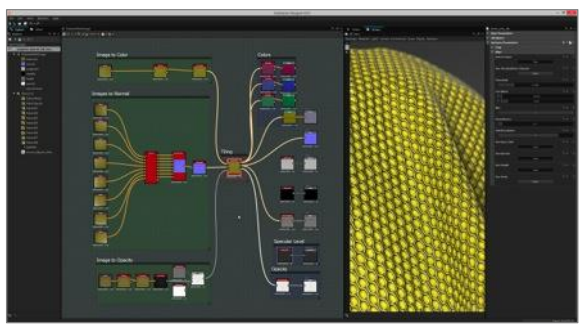






Additional MDL Benefits

Measured Materials

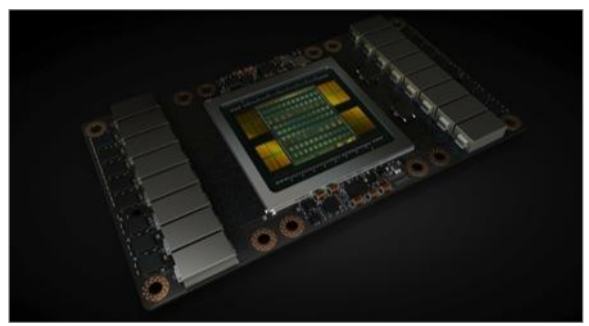


Spatially Varying BRDF

AxF from X-Rite

Measure Isotropic BSDF

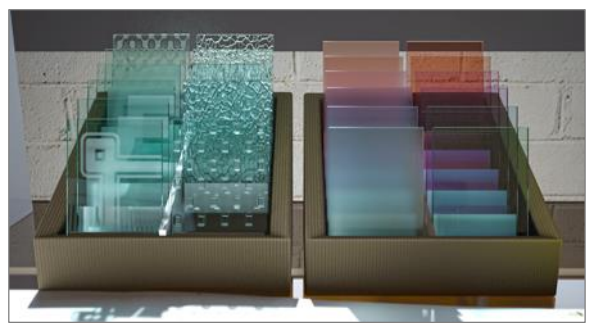
Designed for Parallelism



Little data dependencies

Side-effect free functions

Material Catalogs

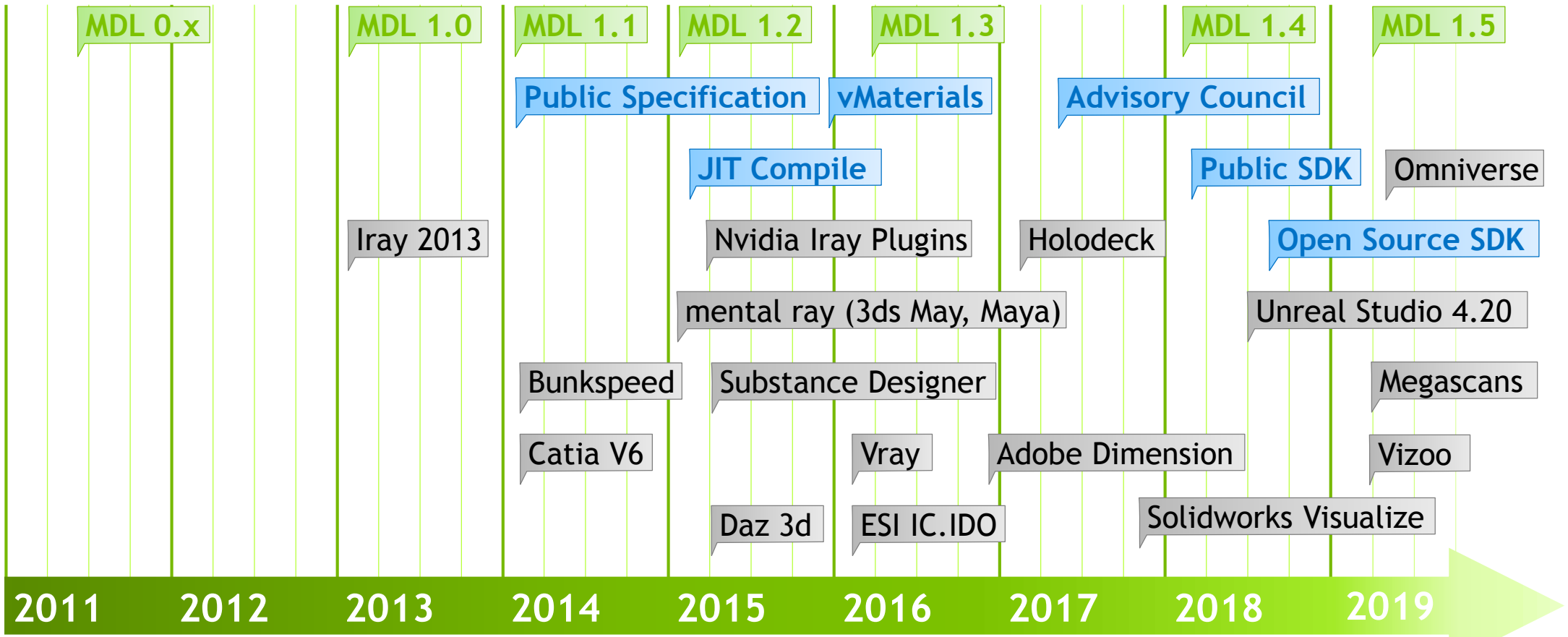


Modules and packages

Archives

MDL Ecosystem

MDL - Past, Present and Future



MDL Advisory Council

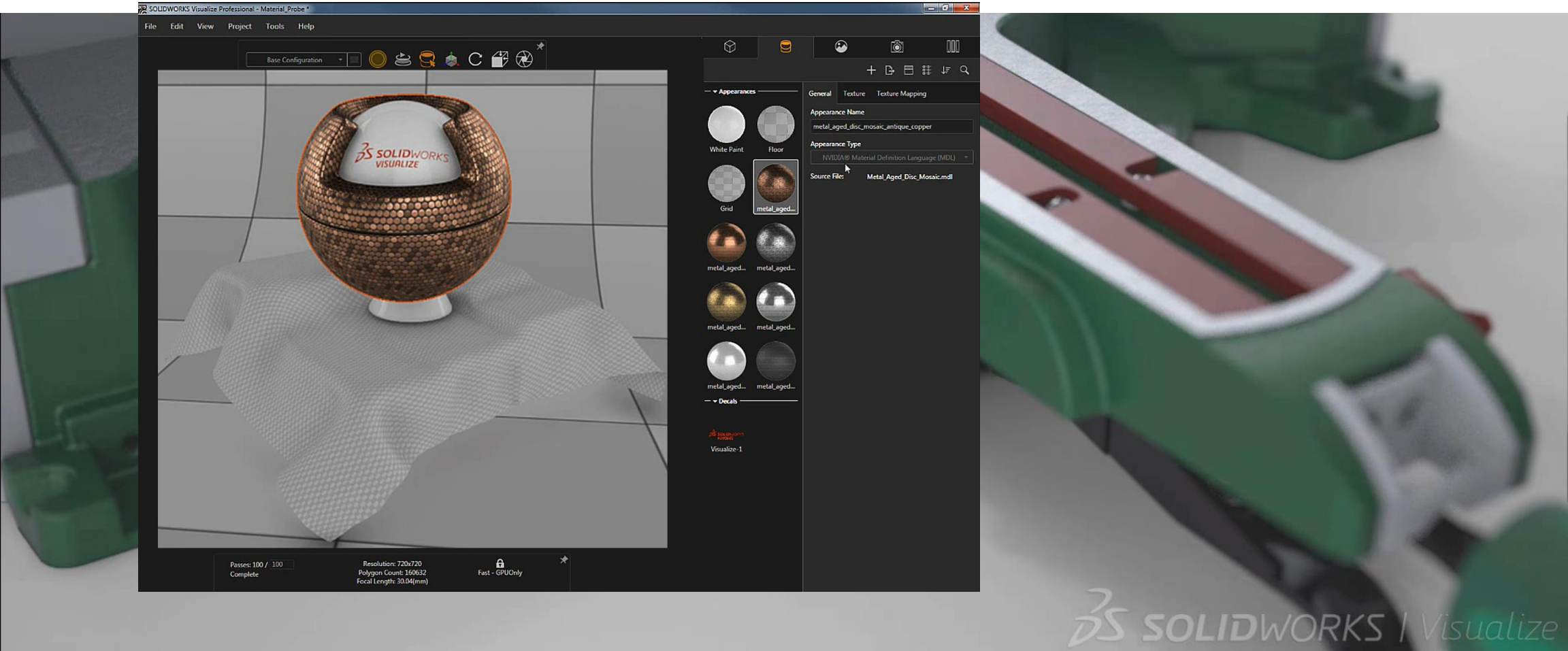
Companies sharing our vision of MDL

Joint direction of MDL and the MDL eco system

Include expertise other companies have gained in the field and with MDL

SOLIDWORKS Visualize

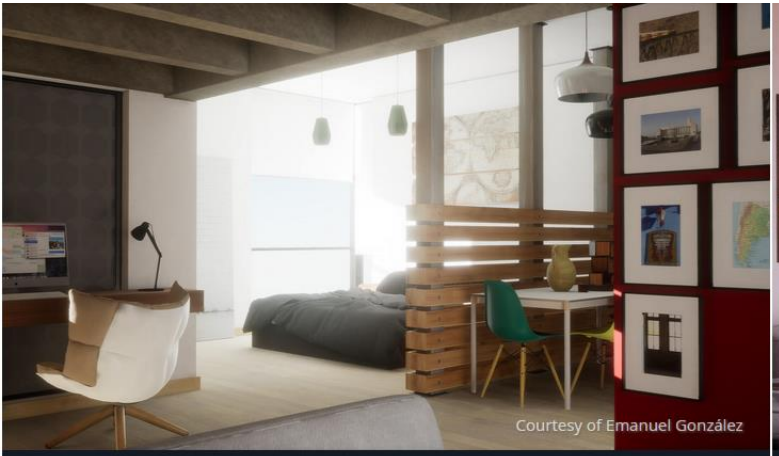
MDL import since 10/2018, tweaking + viewport preview coming



Epic Unreal Studio

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

MDL support through DATASMITH



UNREAL EDITOR

Unreal Studio includes access to Unreal Editor, a powerful real-time tool for creating photorealistic scenes and immersive AR and VR experiences.

Learn more

DATASMITH

Datasmith is the workflow toolkit in Unreal Studio that enables you to seamlessly import your data into Unreal Editor.

Learn more

LEARNING

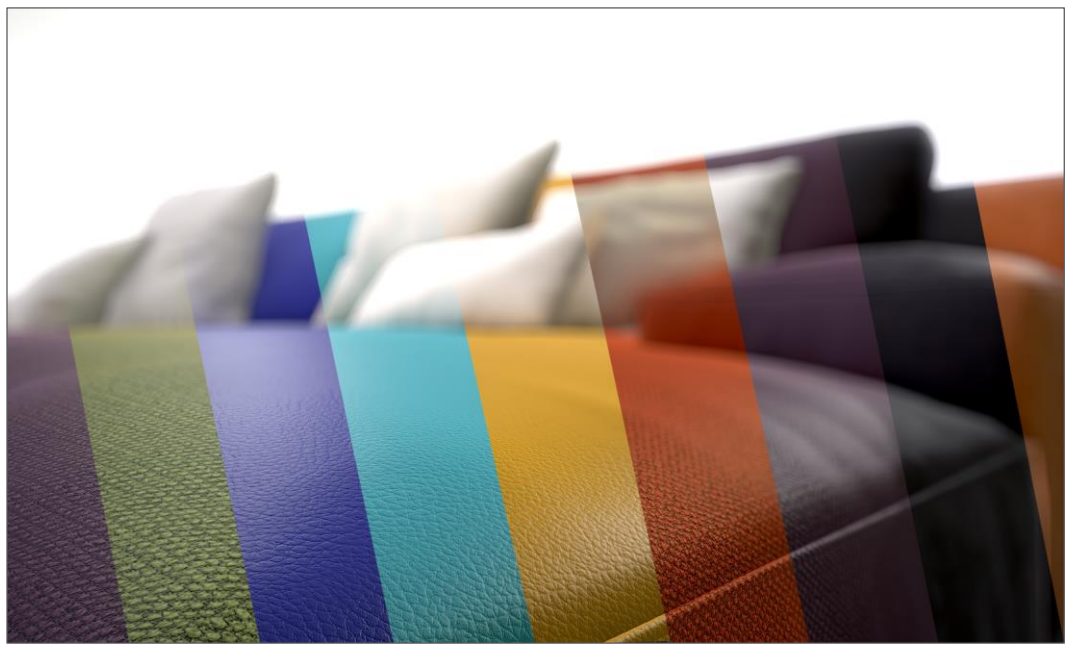
Enhance your skills and knowledge through a library of Unreal Engine and Datasmith video tutorials.

Learn more

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."

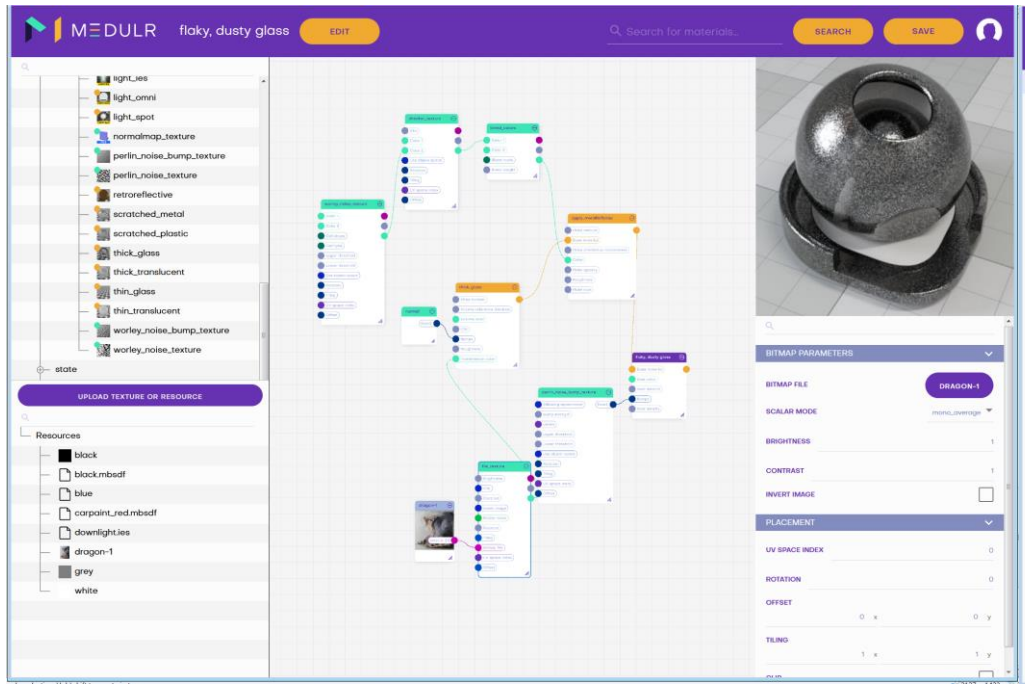


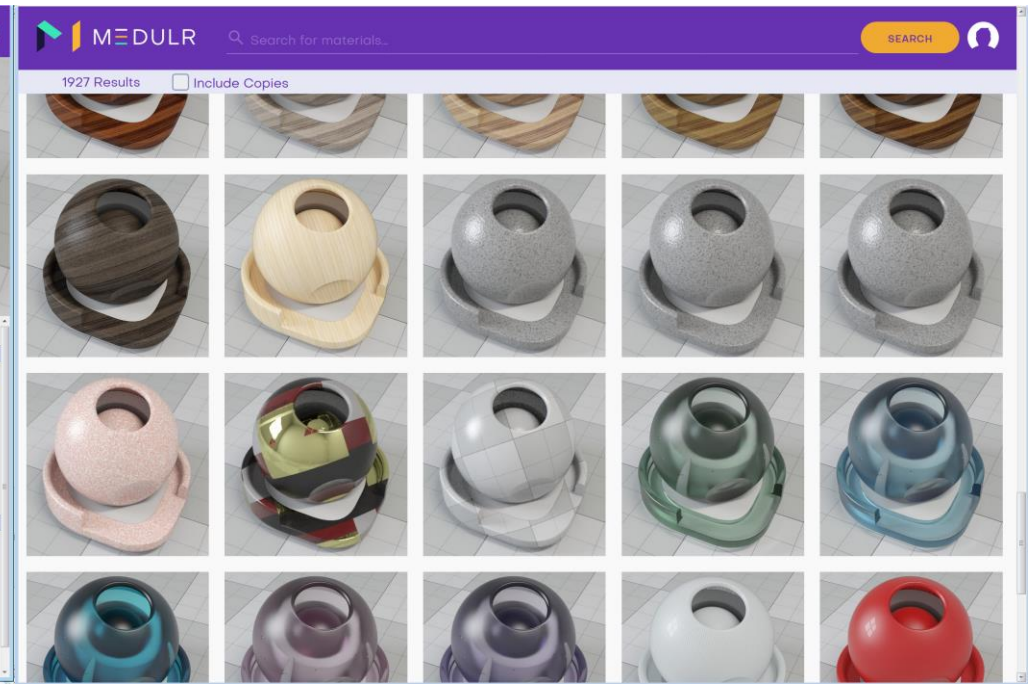
www.vizoo3d.com

MEDULR

Online MDL editor and material library

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

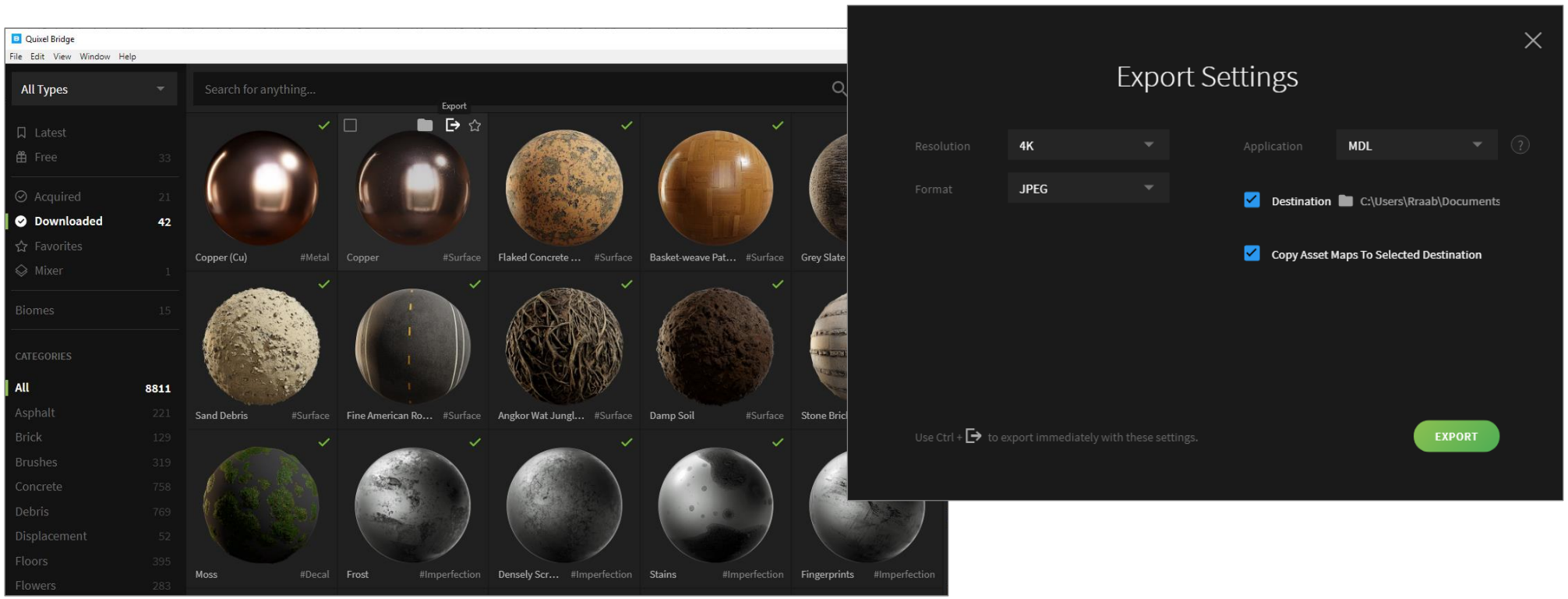




www.medulr.com

Quixel Megascans

“Incredible scans and tools for creatives.”



quixel.com

OMNIVERSE

CLIENT SDK	SERVER	FRAMEWORK	PHYSICS / AI	RTX RENDERER

OMNIVERSE Kit

Connection to Omniverse

Local / On Prem / Cloud

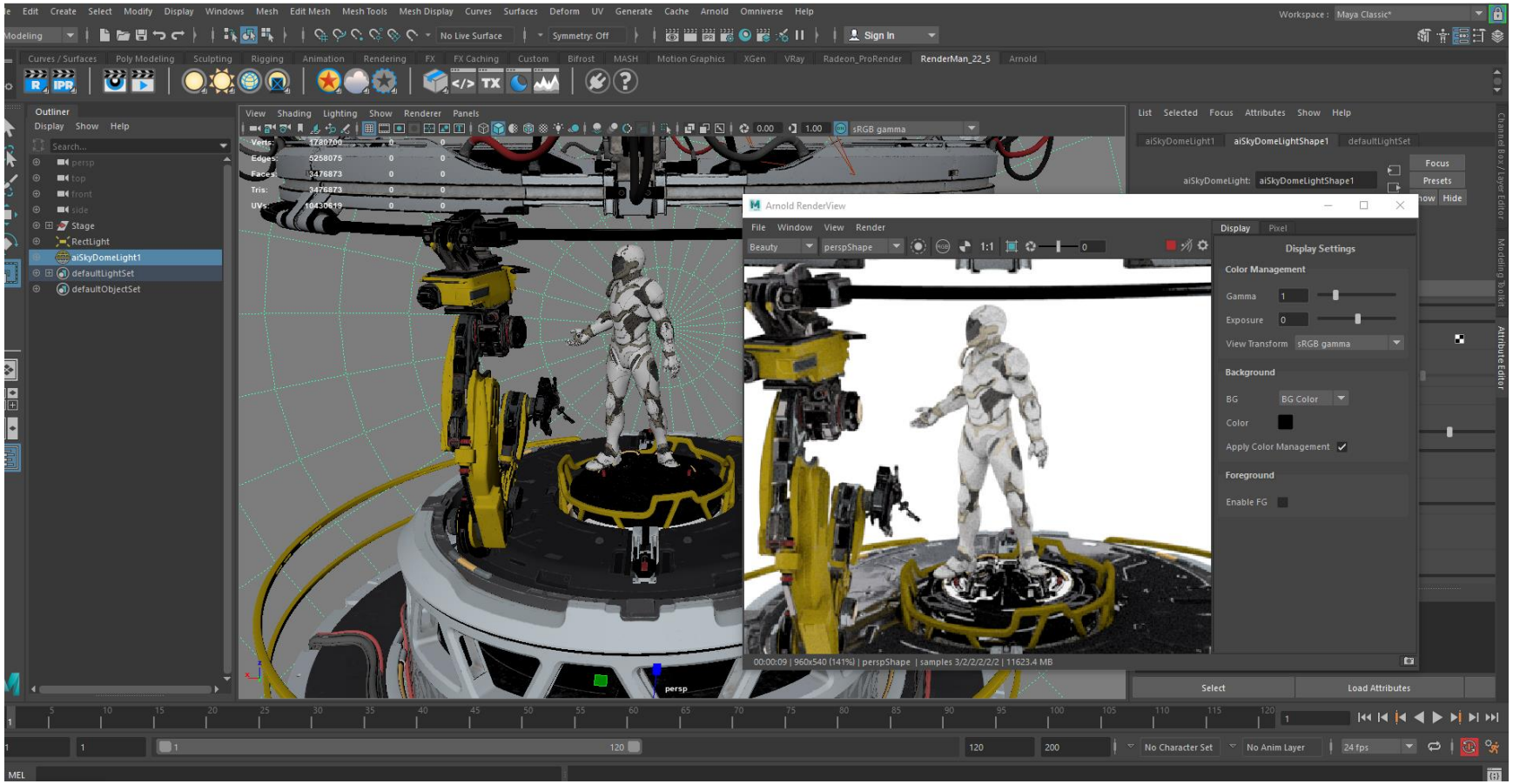
Viewer / Editor / Microservices

Physics / Animation / Behavior

Realtime / Accurate / MDL

POC MDL in Arnold

Rendering Omniverse content in Maya and Omniverse Kit



MDL Ecosystem

Libraries/Scan

NVIDIA VMaterials

Substance Source

Adobe Stock

Vizoo xTex

Quixel Megascans

Creation

Substance Designer

medulr

Rendering

NVIDIA Iray

Chaosgroup Vray

UE4

Omniverse RTX

Applications inc Renderers

Siemens NX11

Solidworks Vis.

Dassault Catia V6

Iray DCC Plugins

Other Iray Prod's

Daz 3D Studio

Adobe Dimensions

Vray Max/Maya

Unreal Studio

NVIDIA Omniverse

Focus on Material Exchange

Freely choose where to author material content

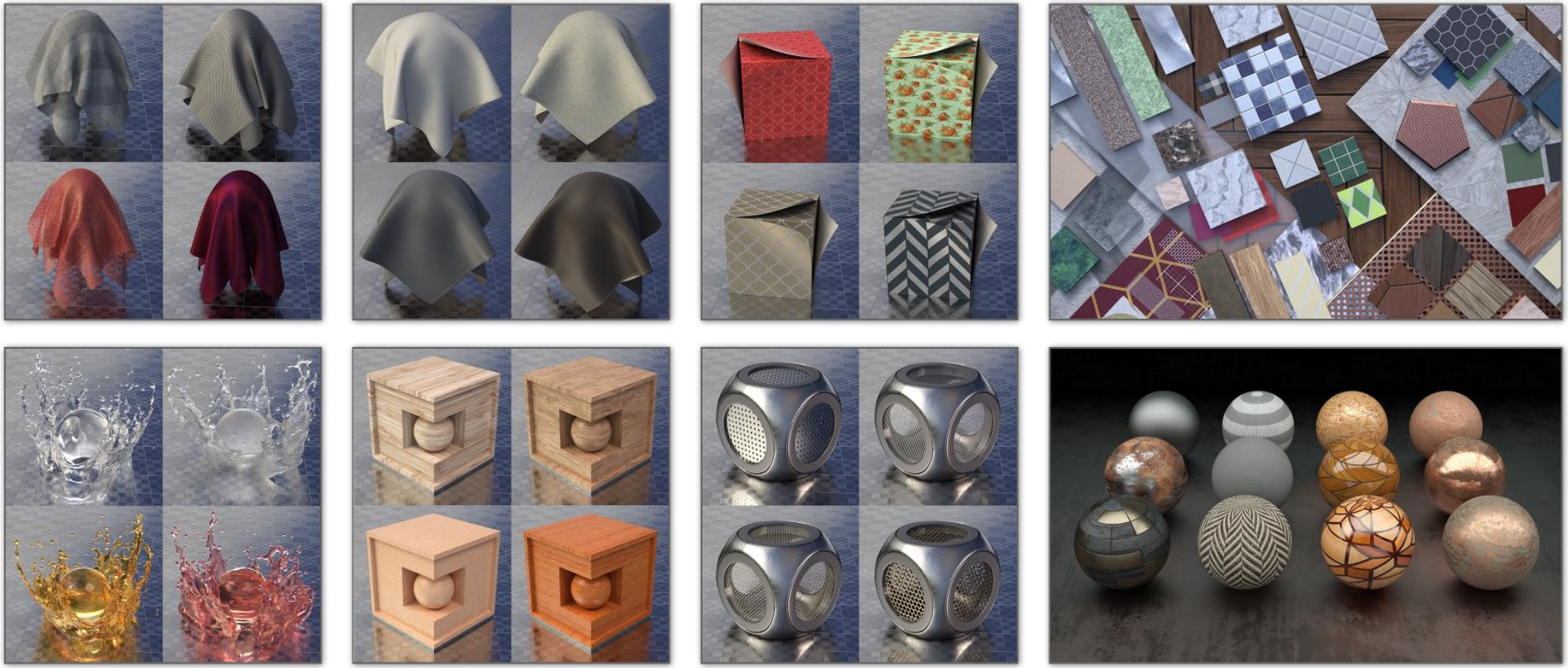
create

modify

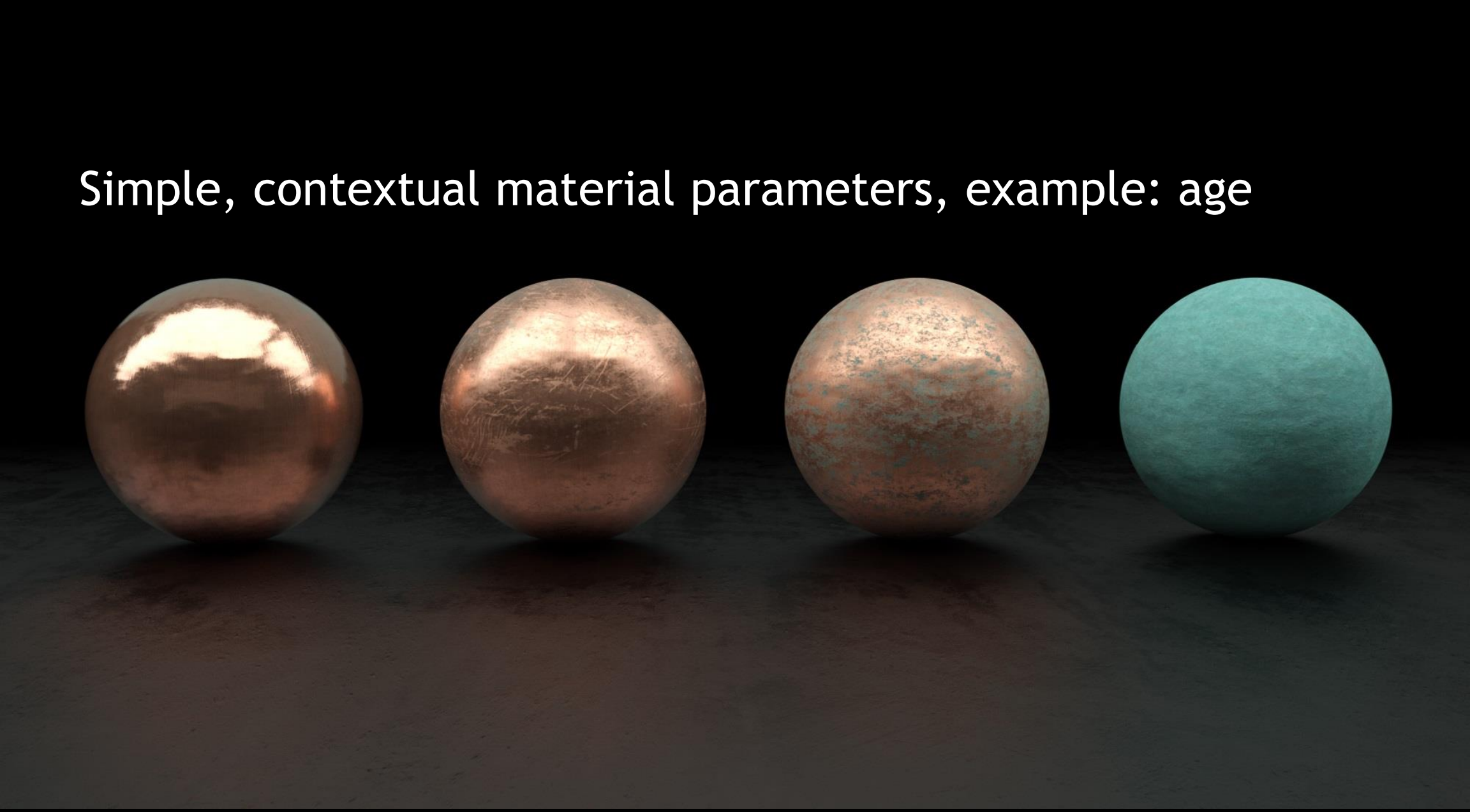
consume

NVIDIA vMaterials 1.6 - Just released

~1700 MDL materials verified for accuracy - FREE TO USE



Simple, contextual material parameters, example: age



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

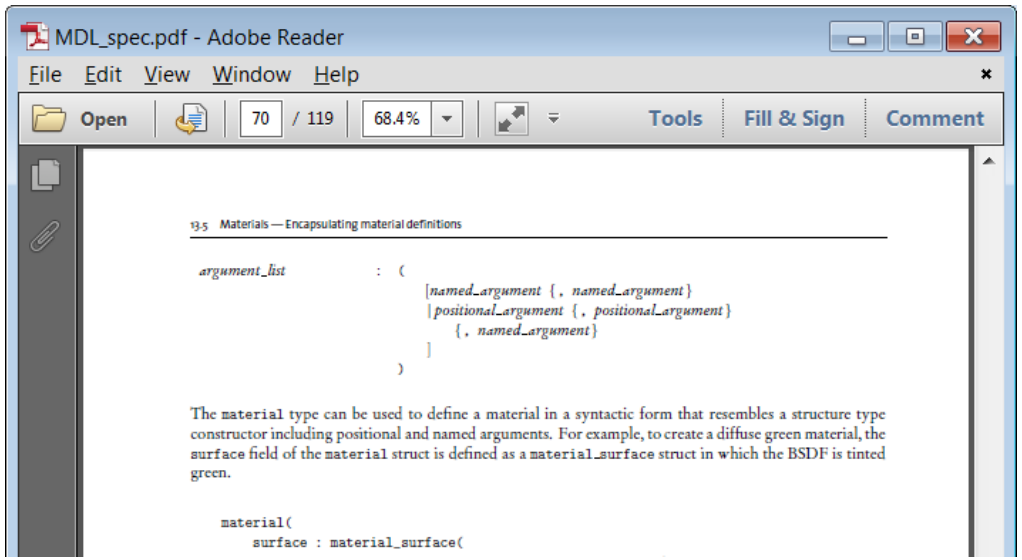
Published under the NVIDIA Designworks License and …

Write Your Own Compiler

MDL Specification

Language specification document
Free to use

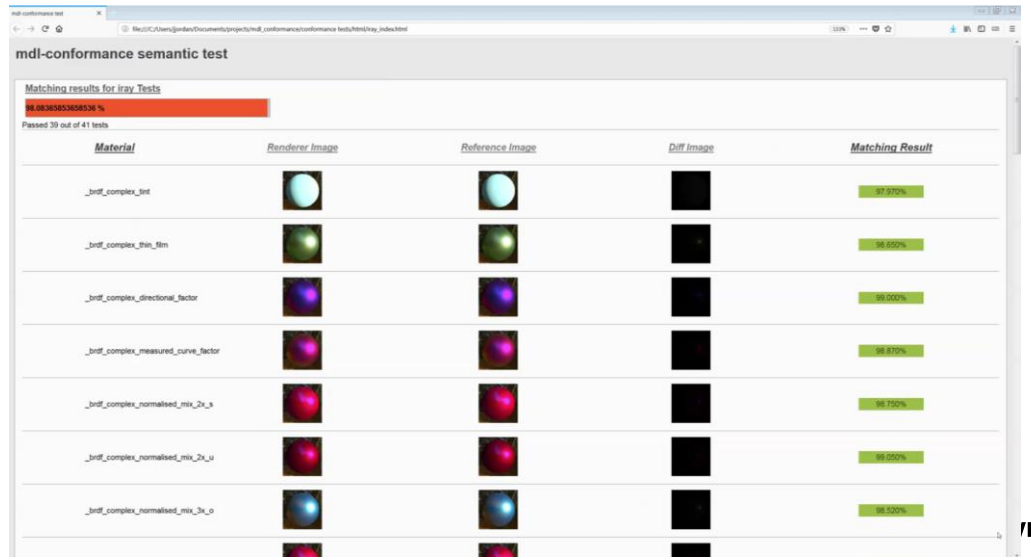
http://www.nvidia.com/mdl/



MDL conformance test suite

Syntactic conformance tests
Semantic conformance tests

Available on request



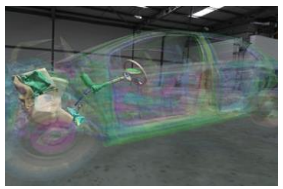
NVIDIA. DESIGNWORKS™

RENDERING


Iray SDK

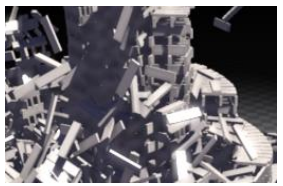

OptiX SDK


MDL SDK


NV Pro Pipeline

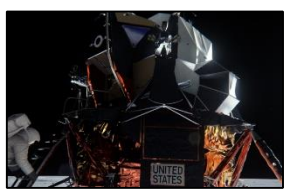

vMaterials

PHYSICS


PhysX

VOXELS


GVDB Voxels


VXGI

VIDEO


GPUDirect for Video

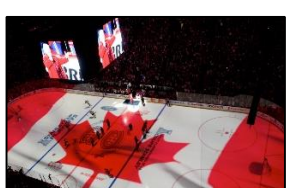

Video Codec SDK

MANAGEMENT


GRID SW MGMT SDK


NVAPI/NVWMI

DISPLAY


Multi-Display


Capture SDK


Warp and Blend

https://developer.nvidia.com/designworks

MDL SDK 2019

Features

MDL 1.5

DB for MDL definitions

DAG view on materials
several compilation modes

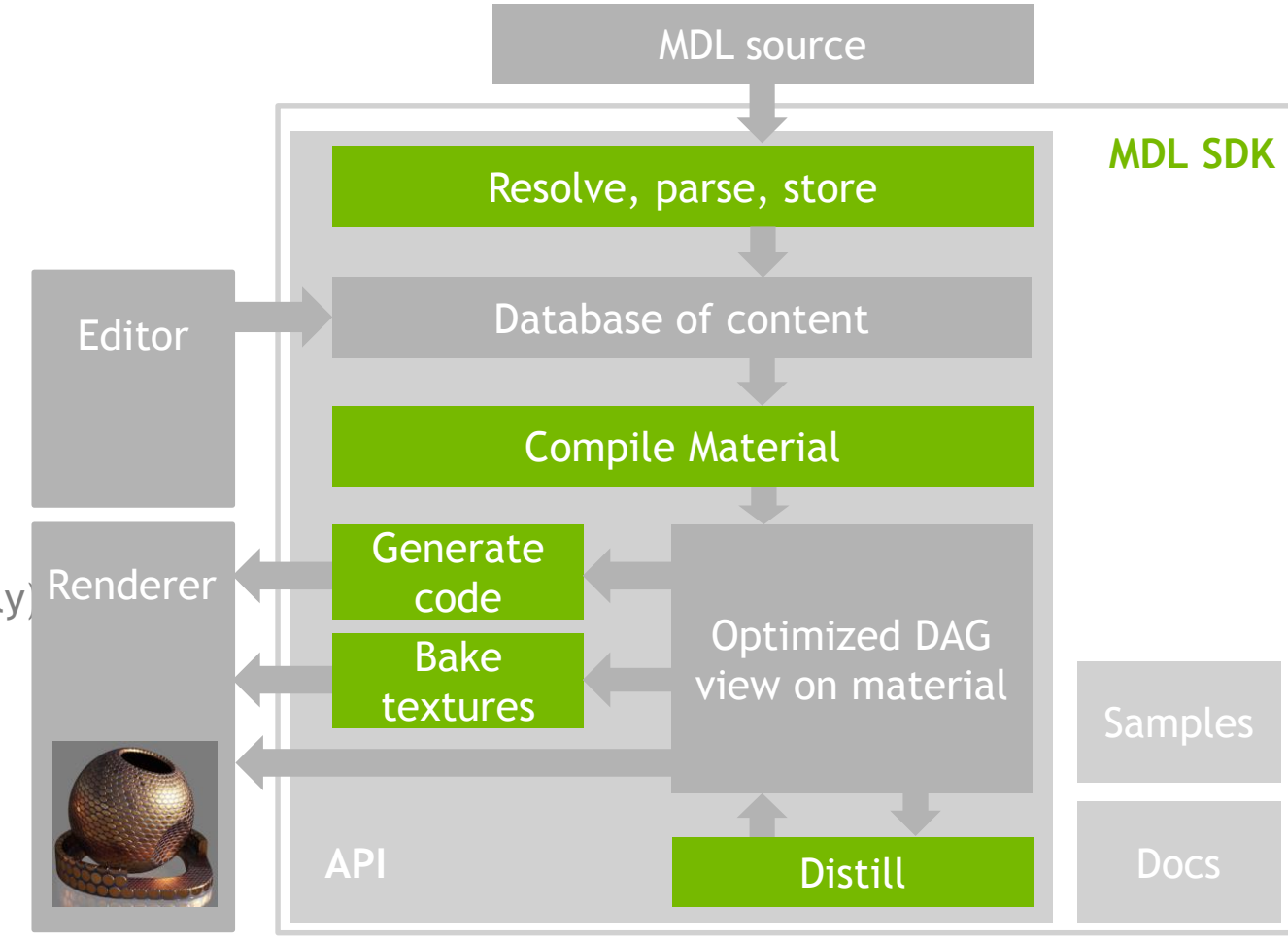
MDL editing

Code generators
PTX, LLVM IR, x86, HLSL, GLSL (fcts. only)

Distiller and texture baker

Code examples

Documentation and tutorials



MDL and RTX

Materials tricky for todays 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

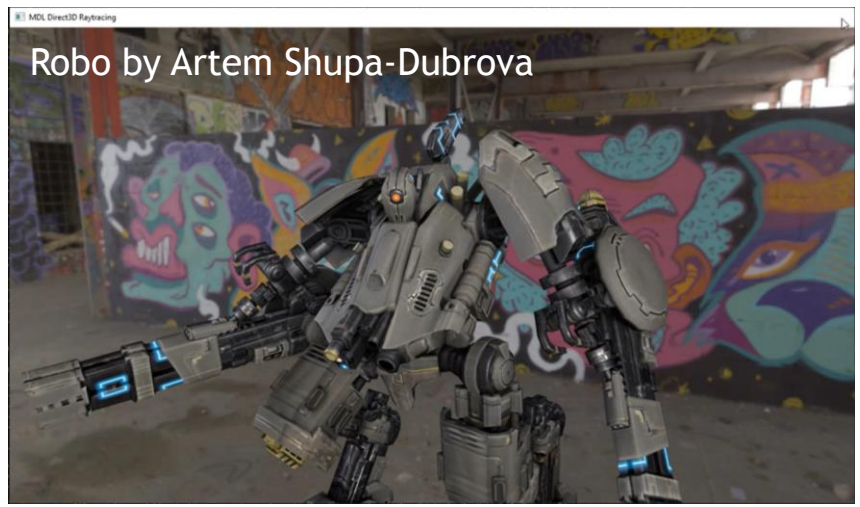
Microsoft DXR

- HLSL back-end since MDL SDK 2019.0.1
- example path tracer in the SDK with glTF support

NVIDIA OptiX

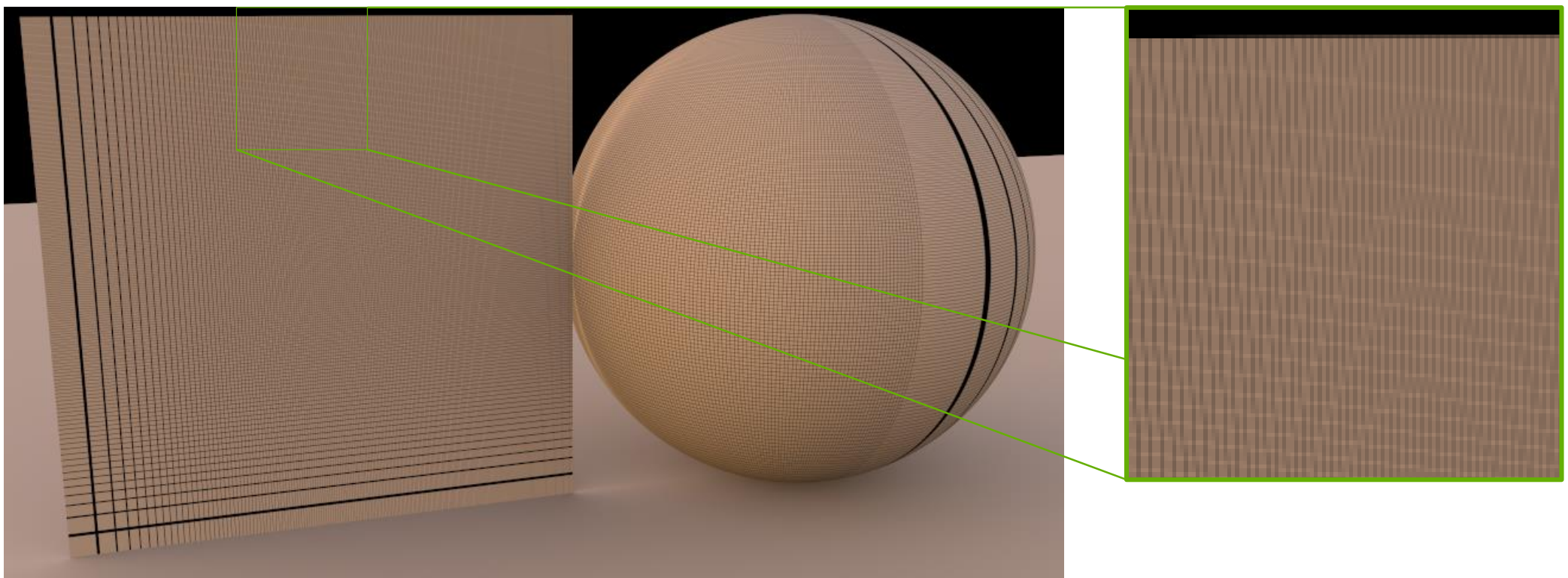
- PTX back-end since MDL SDK 2018.1
  example program available as part of Optix 5.1 & 6

Integrating MDL with an RTX based renderer is simple!



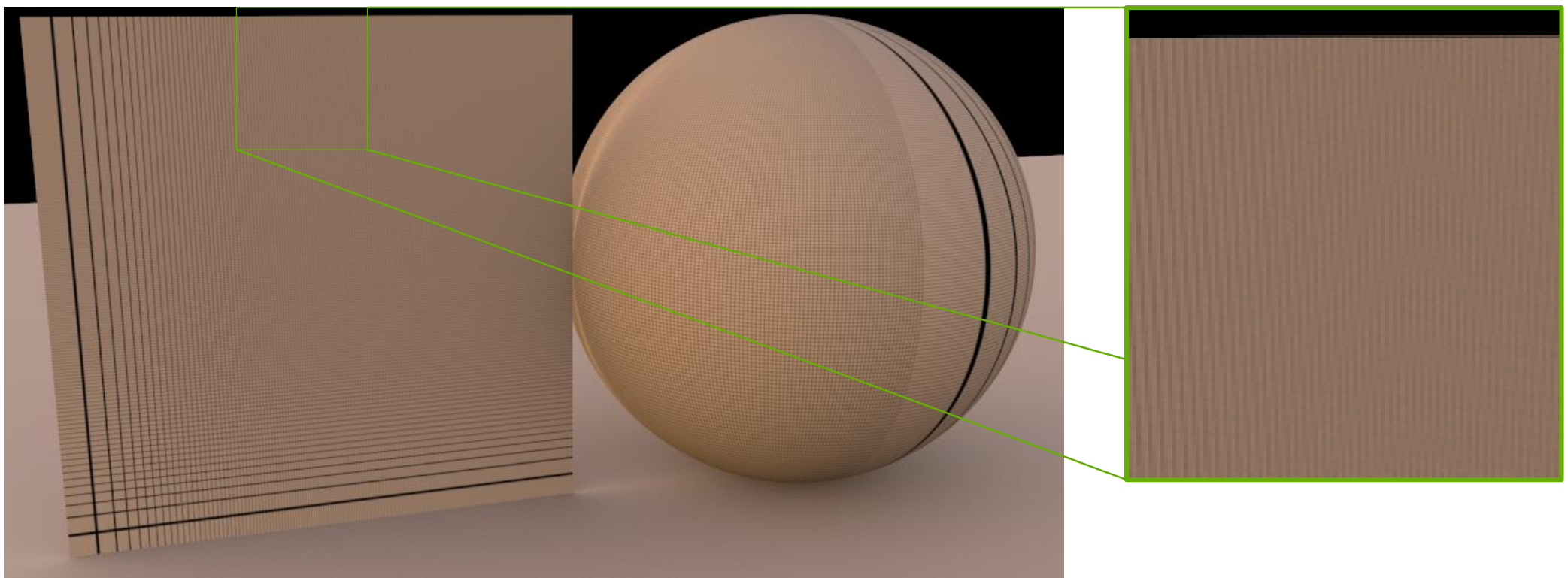
Automatic Derivatives for Texture Filtering

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

Distillation →

Fixed Material Model

Simple BSDF structure

One texture per parameter

Distillation to Fixed Material Model

MDL Material

Complex BSDF layering

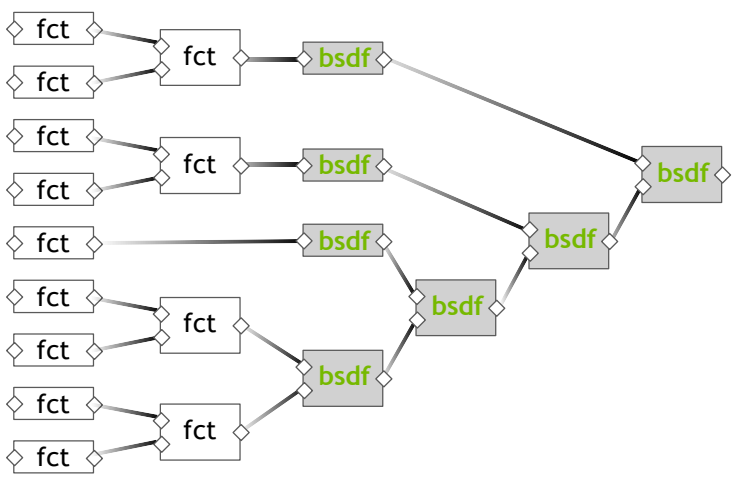
Complex procedurals

Distillation →

Fixed Material Model

Simple BSDF structure

One texture per parameter



Distillation to Fixed Material Model

MDL Material

Complex BSDF layering

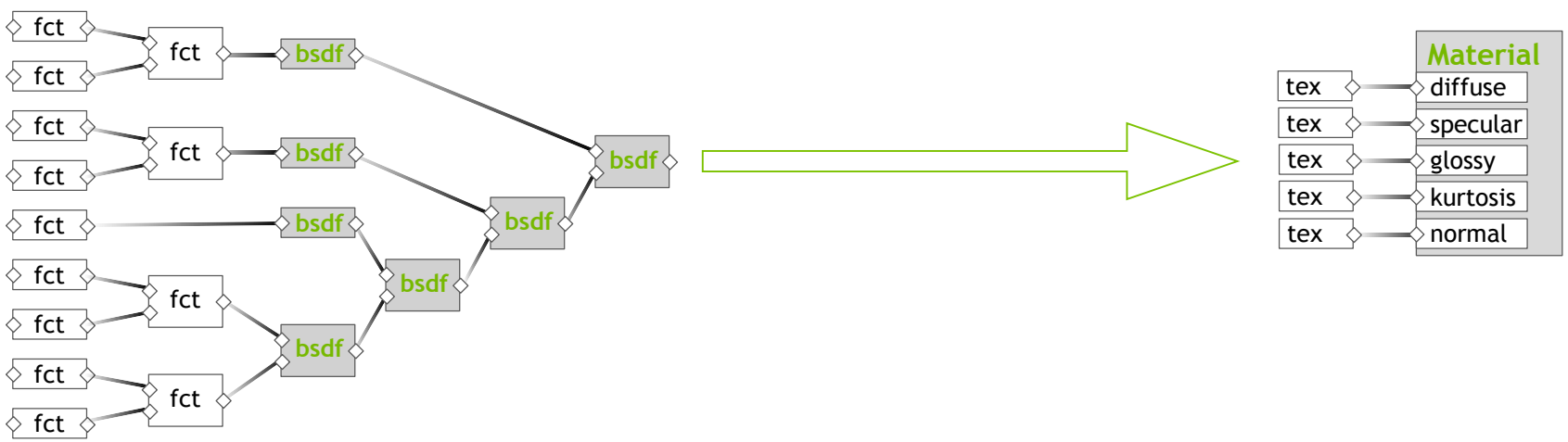
Complex procedurals

Distillation

Fixed Material Model

Simple BSDF structure

One texture per parameter



Distillation to Fixed Material Model

MDL Material

Complex BSDF layering

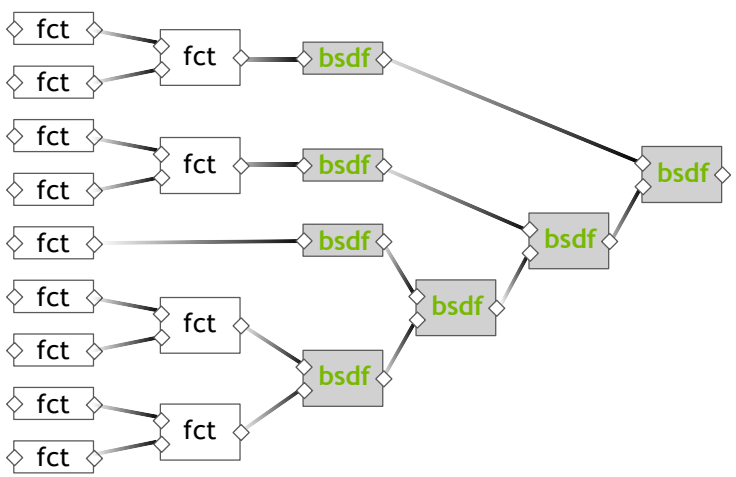
Complex procedurals

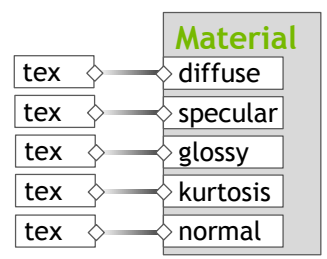
Distillation →

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

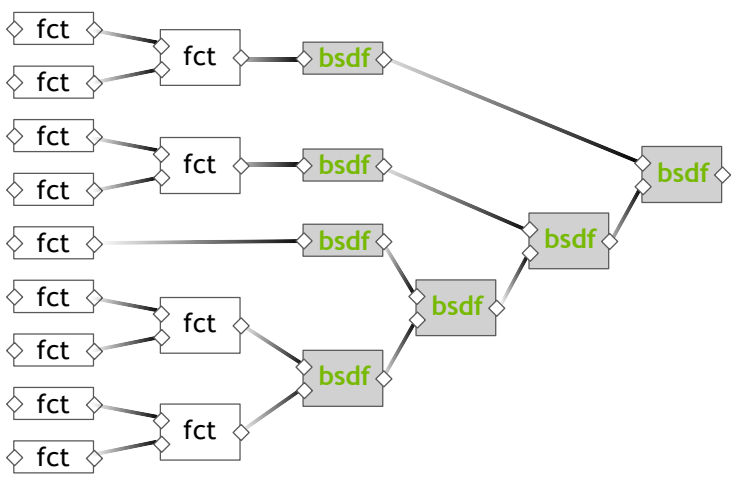
Complex procedurals

Distillation →

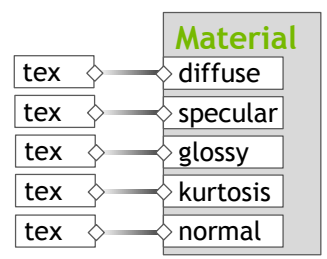
Fixed Material Model

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

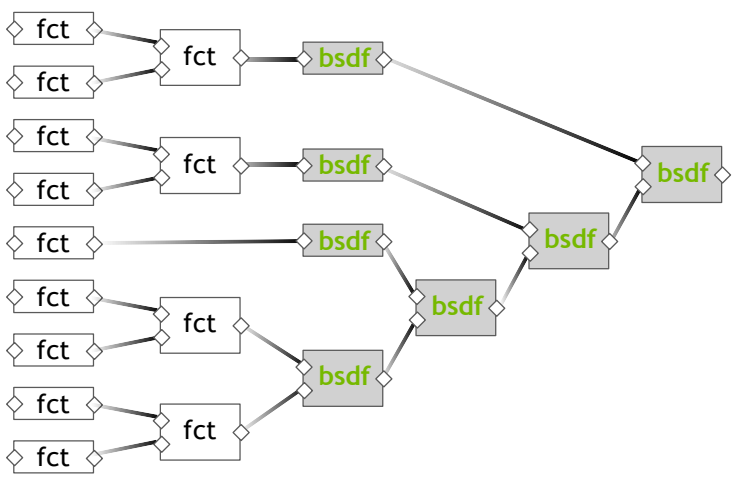
Complex procedurals

Distillation →

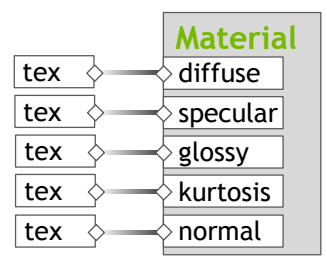
Fixed Material Model

Simple BSDF structure

One texture per parameter



Fast projection of material instances: Realtime editing

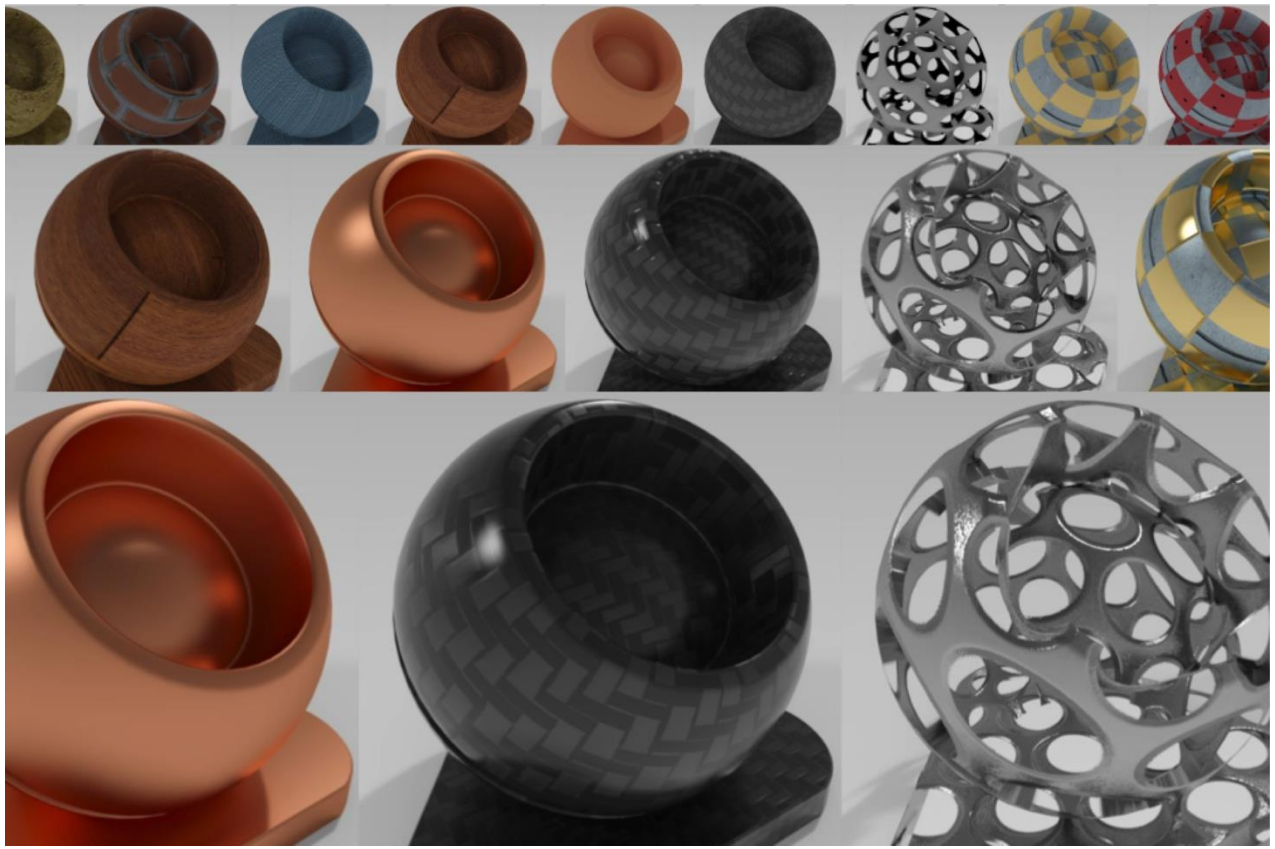


Approximate render result: Some materials will look quite different

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

Distillation to Fixed Material Model

Results on vMaterials



diffuse-only

Fresnel(glossy, diffuse)

original

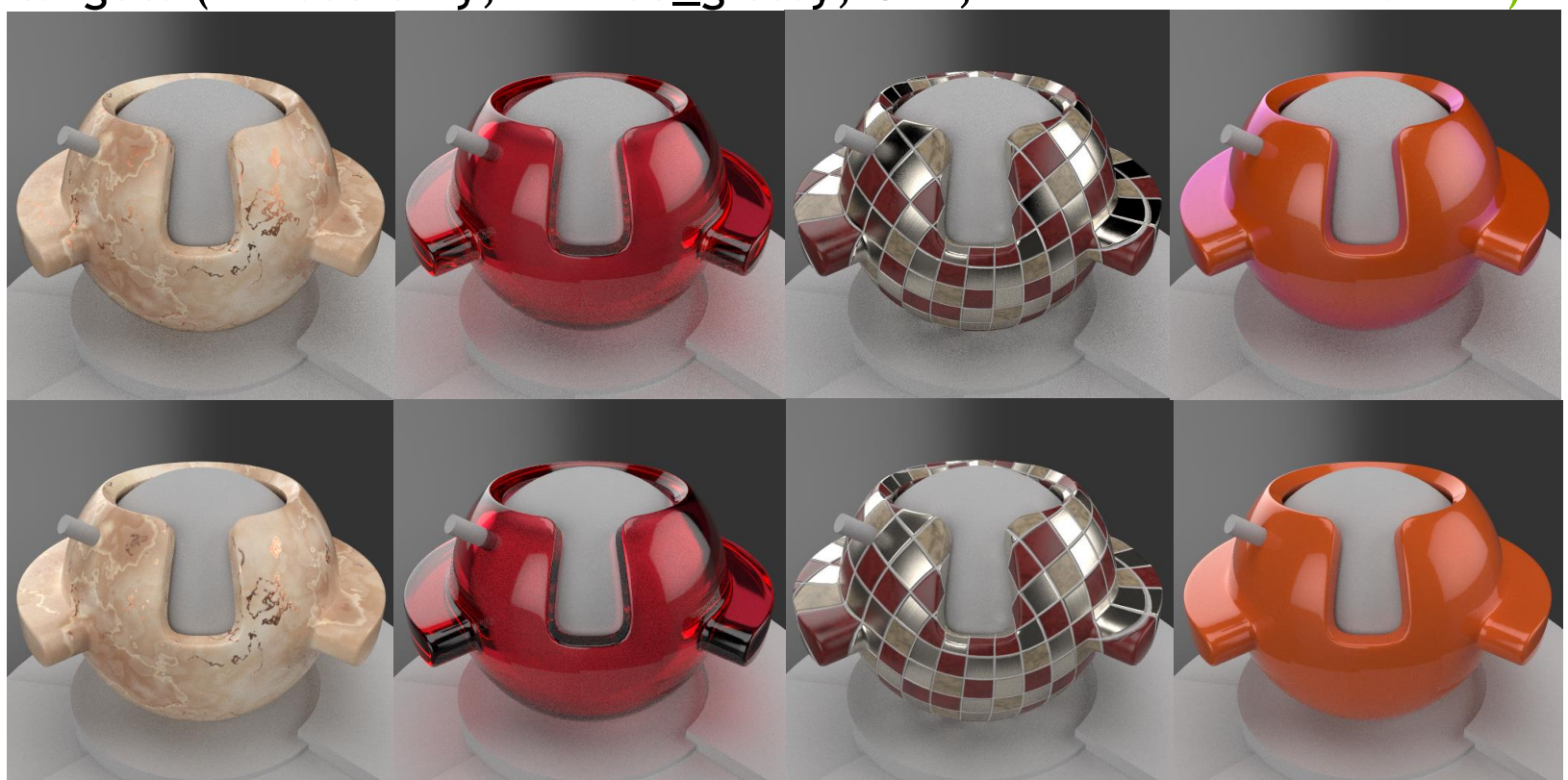
MDL Distilling

Released as part of Iray/MDL SDK

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

Original:
Iray MDL

Projection:
Dassault Stellar
with Enterprise
PBR



First Aniversary
of the MDL SDK open source release

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

6 releases shipped since SIGGRAPH 2018

HLSL backend and example just added

BSD 3-clause license

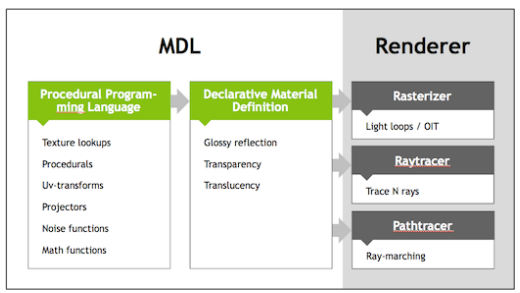
Full MDL SDK

- Added MDL Core API
- Excluding
  - MDL Distilling and texture baking
  - GLSL compiler back-end
- Includes MDL Core Definitions and more

Feature image courtesy of Adobe, created by art director Vladimir Petkovic.

MDL Takeaways

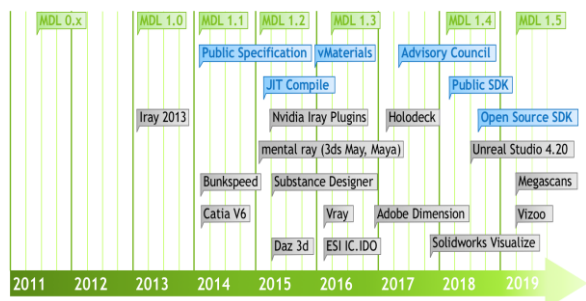
What is MDL



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 rendering 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*
- *Handbook*
- *Language Specification*

GTC On-Demand

on-demand-gtc.gputechconf.com

MDL@SIGGRAPH

Mon 10 AM Room 501AB	*Talk: Sharing Physically Based Materials Between Renderers with MDL*
Wed 12:30PM Room 503	*BoF: Material Definition Language (MDL): Application Independent PBR Materials*
Tue 11:00AM Room 503	*Talk:NVIDIA Omniverse: An Open, USD Based Collaboration Platform for Constructing and Simulating Virtual Worlds*
Tue -Thu Exhibition	*Omniverse Demos at the NVIDIA booth*