# Sharing Physically Based Materials Between Renderers with MDL

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

October 10, GTC Europe 2018

# 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

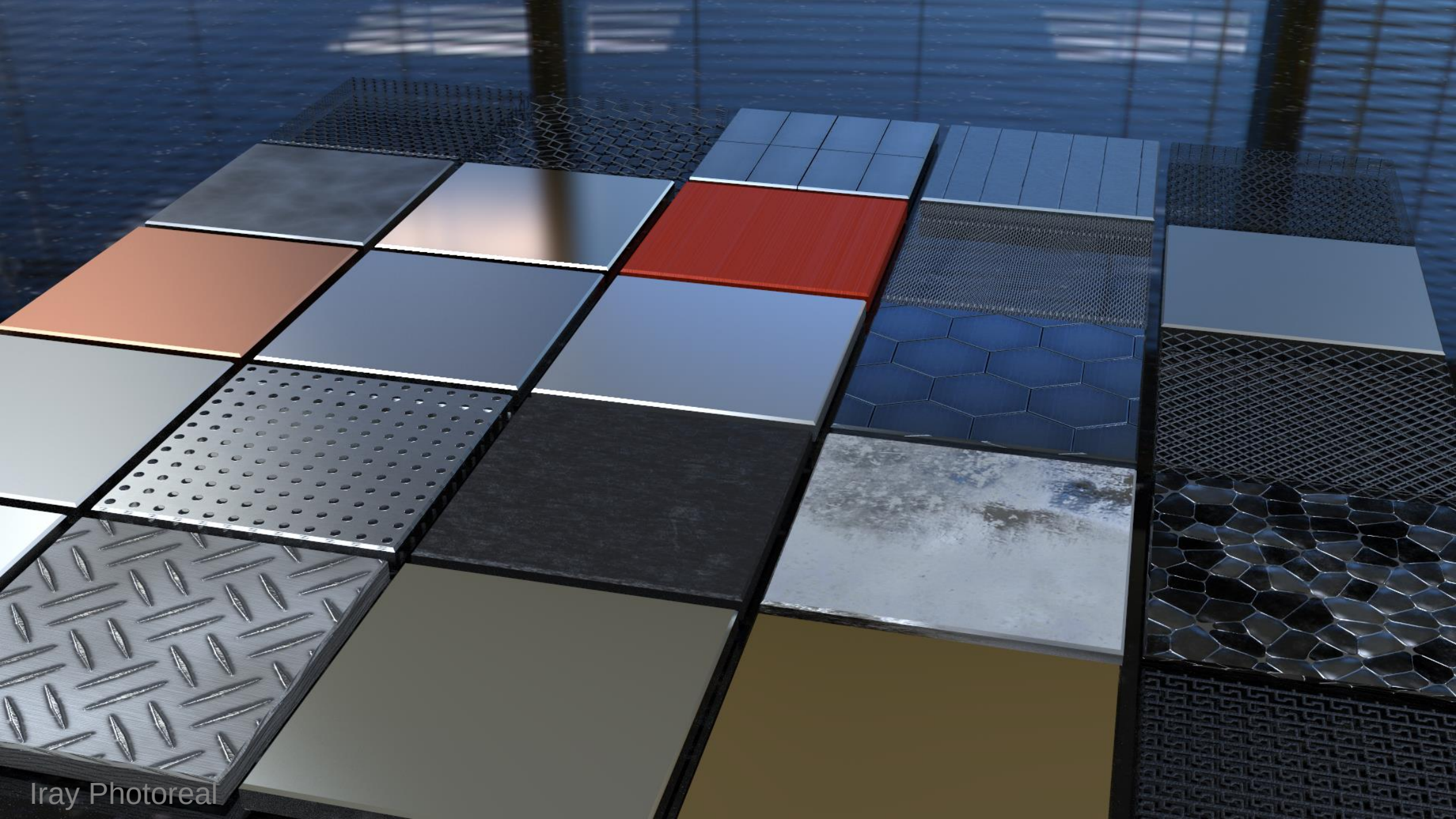
# NVIDIA® MDL

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

# Matching the Appearance of a Single Material Within Different Rendering Techniques

# One Scene for Different Renderers

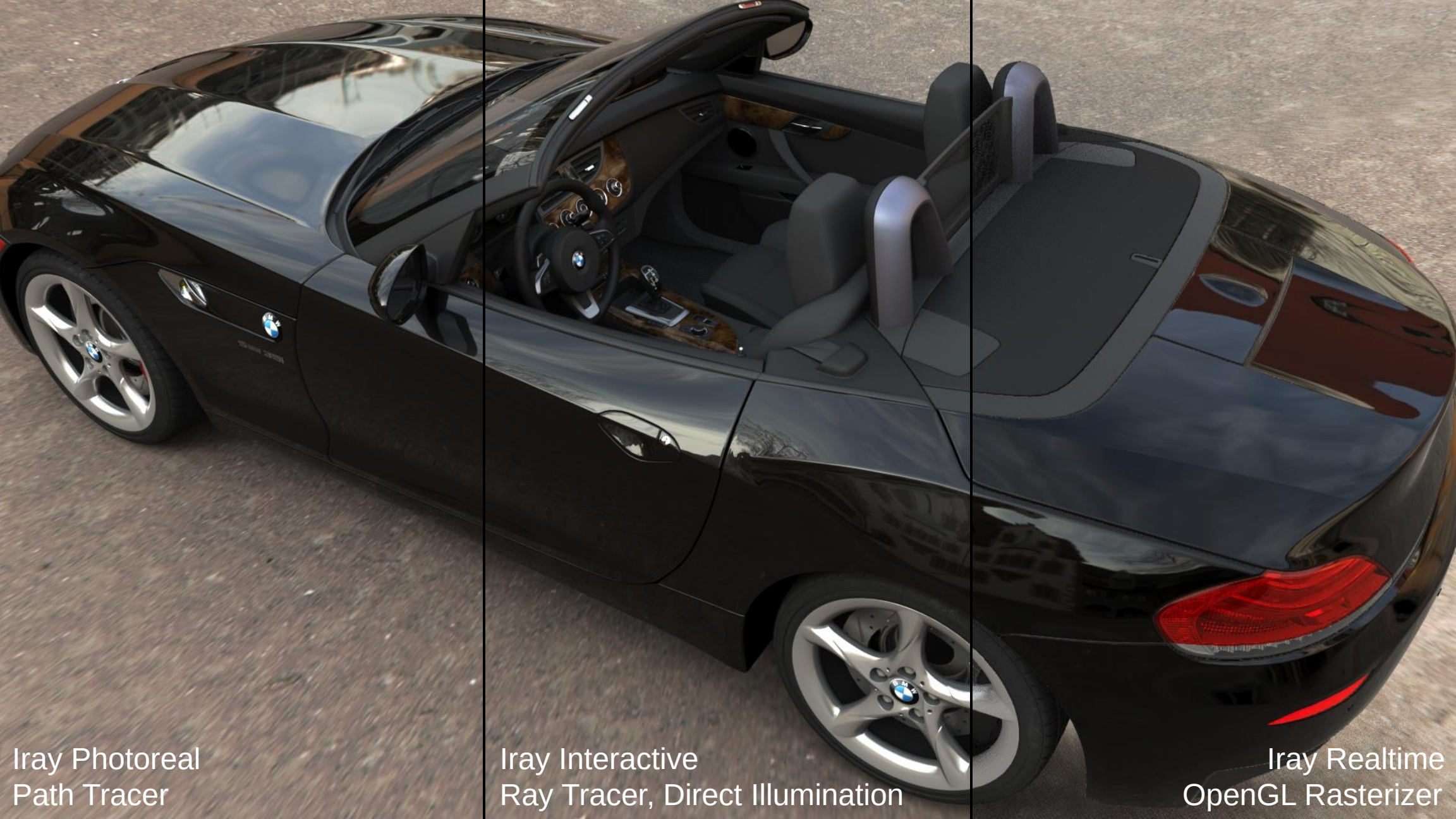
**Realtime Rasterizer**

**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

## 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

# NVIDIA MDL

**Procedural Programming Language** → **Declarative Material Definition**

## Renderer

### Rasterizer

Light loops / OIT

### Raytracer

Trace N rays

### Pathtracer

Ray-marching

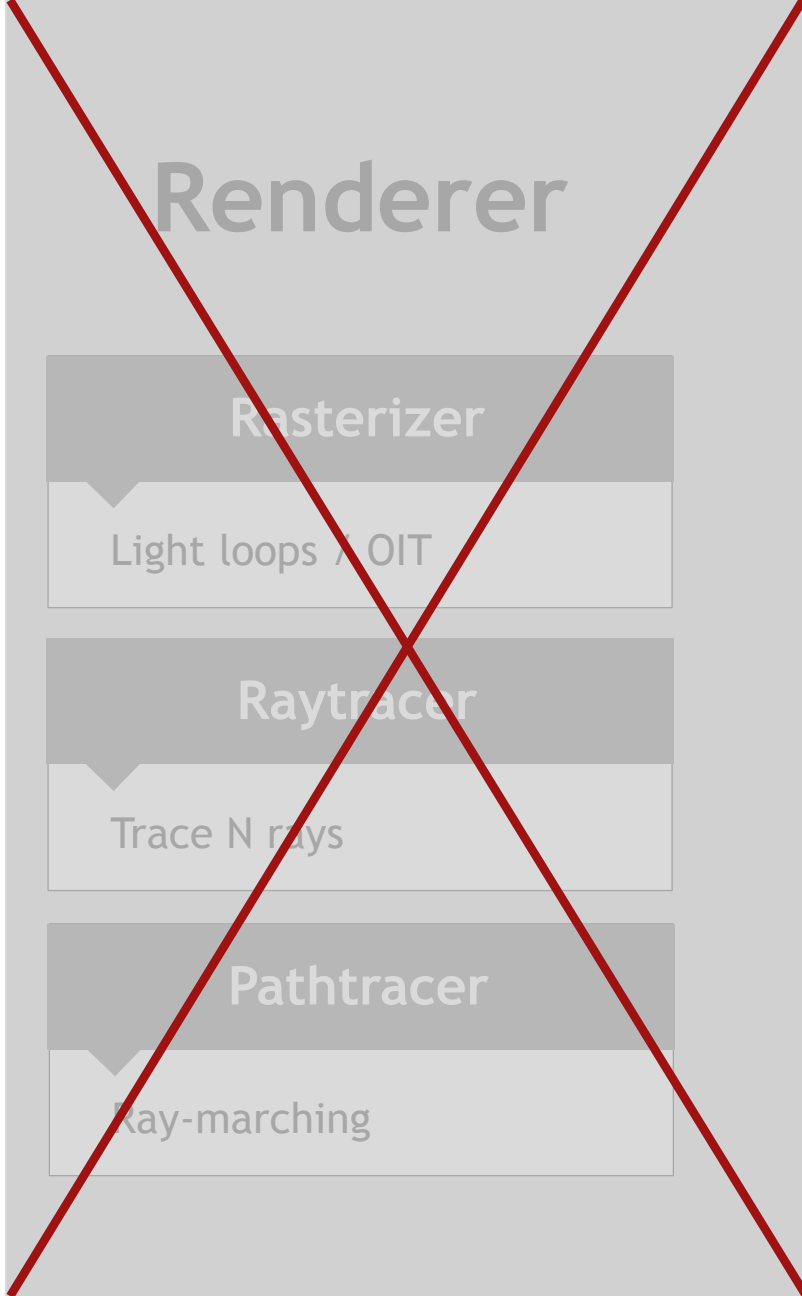
# NVIDIA MDL

**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

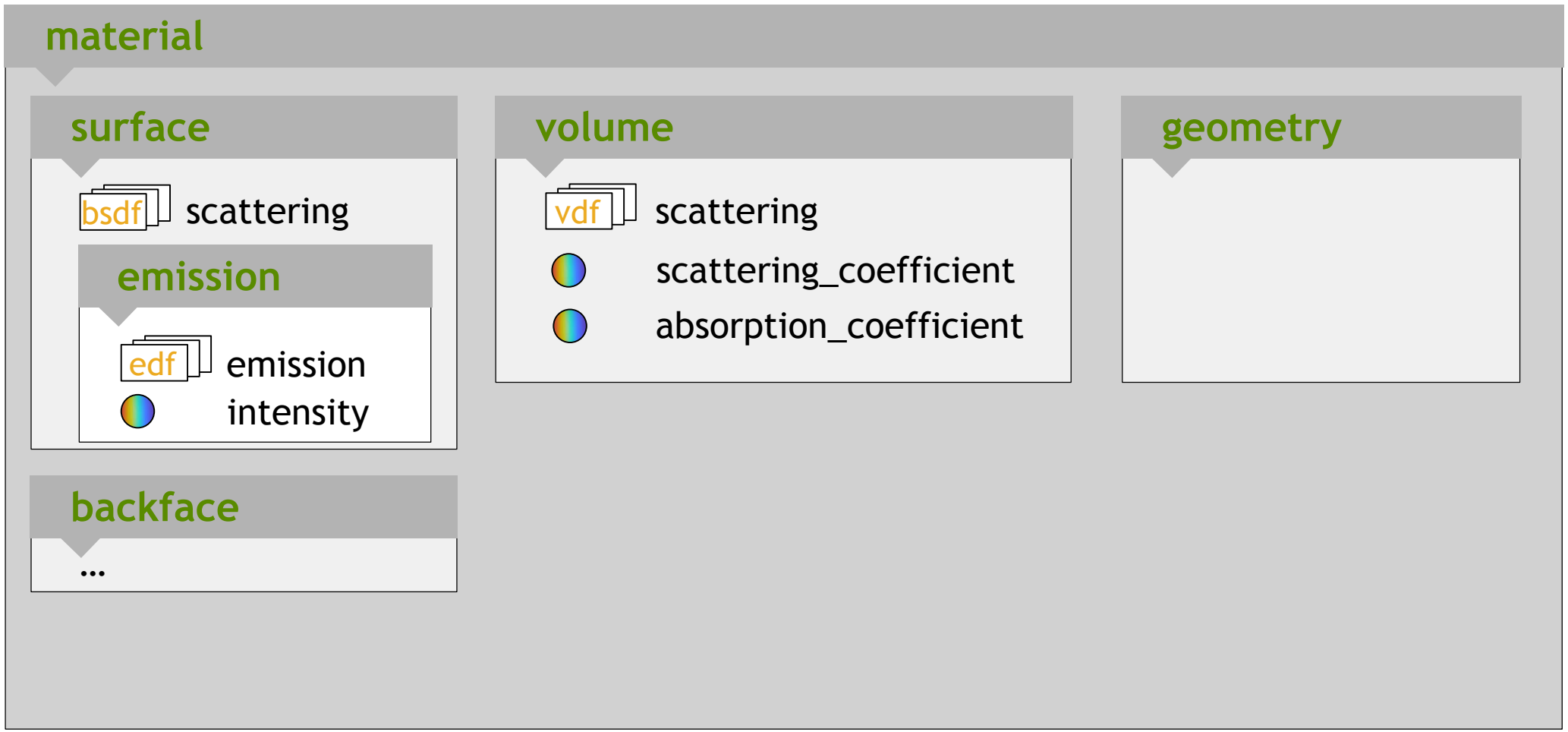


# MDL Material Model

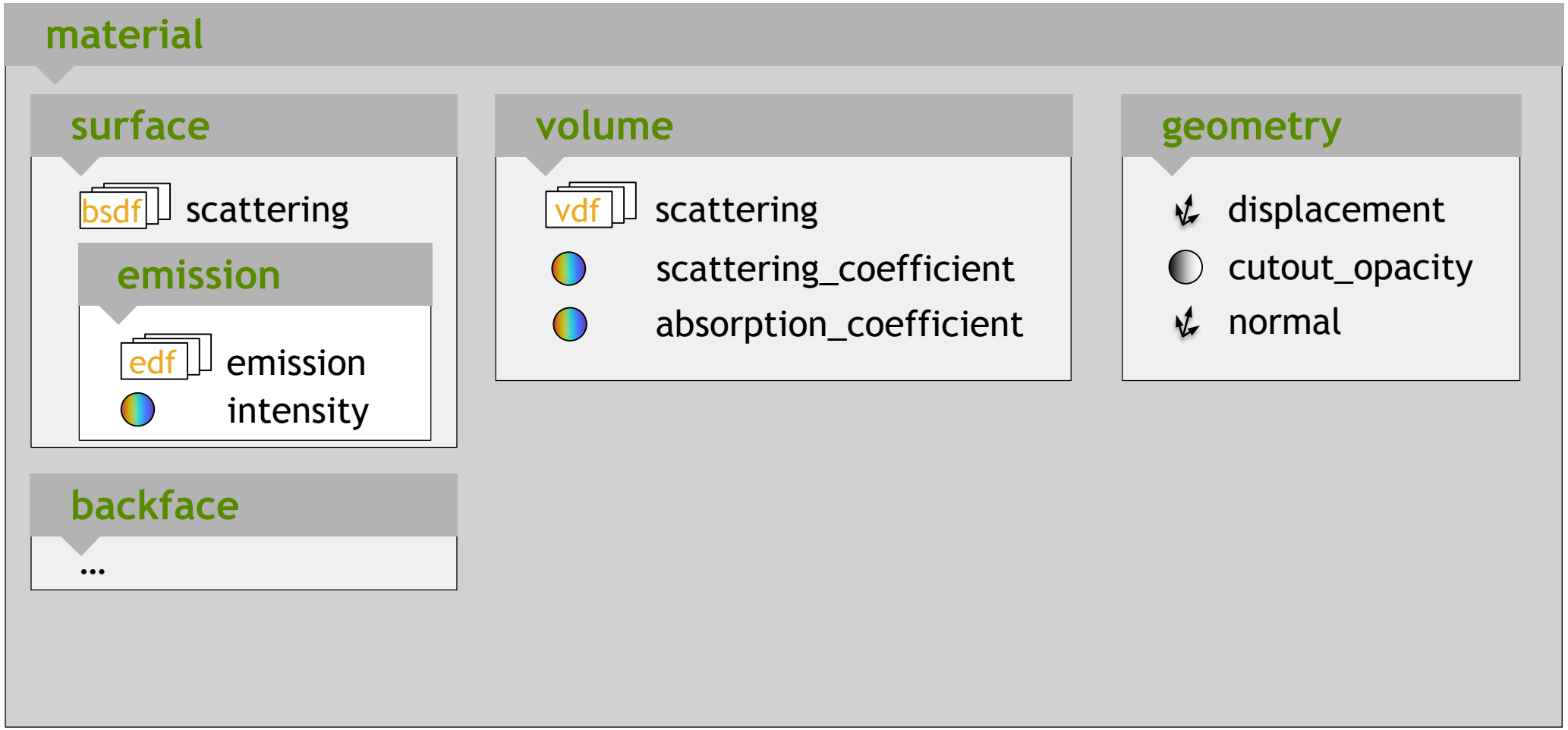
# MDL Material Model

# MDL Material Model

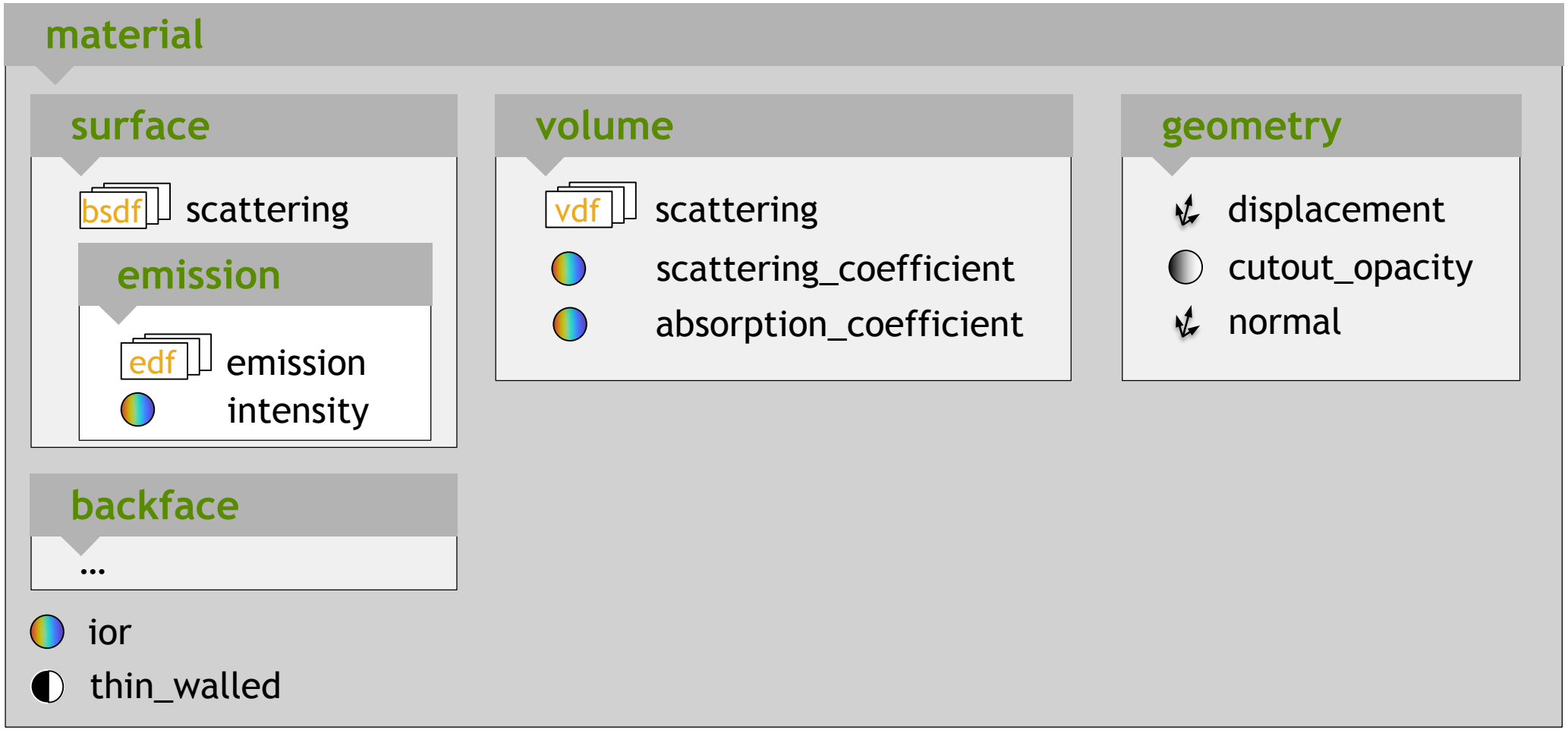
# MDL Material Model



# MDL Material Model



# MDL Material Model



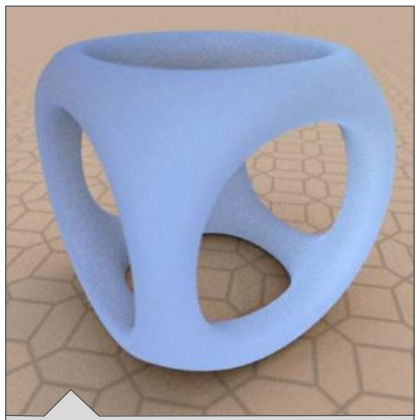
# MDL Elemental Distribution Functions

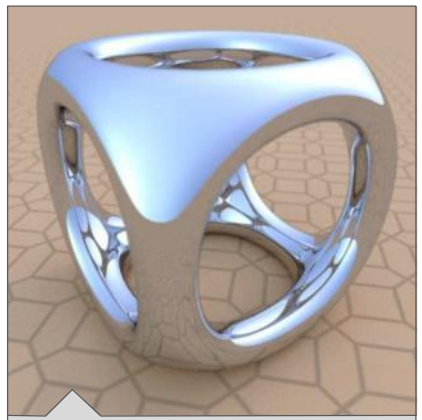
Bidirectional
Scattering
Distribution
Functions

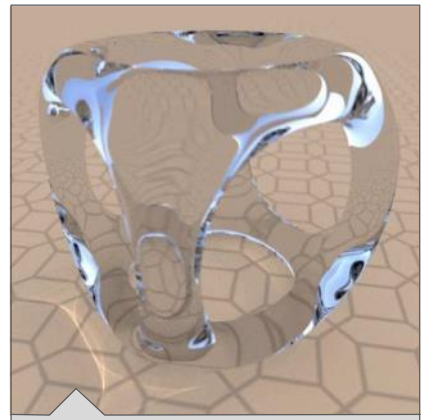

Diffuse Reflection


Diffuse Transmission


Glossy (various)


Backscatter Glossy


Specular Reflection

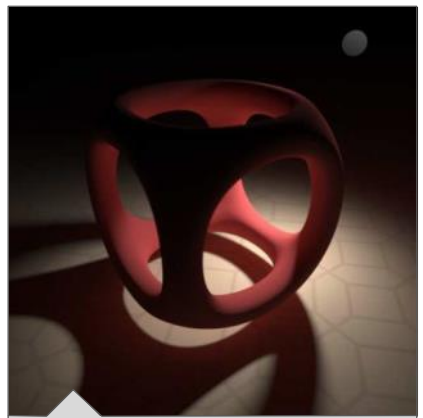

Spec. Refl.+Transm.


Measured BSDF

# MDL Elemental Distribution Functions

Emissive
Distribution
Functions

Diffuse



Spot

IES Profile

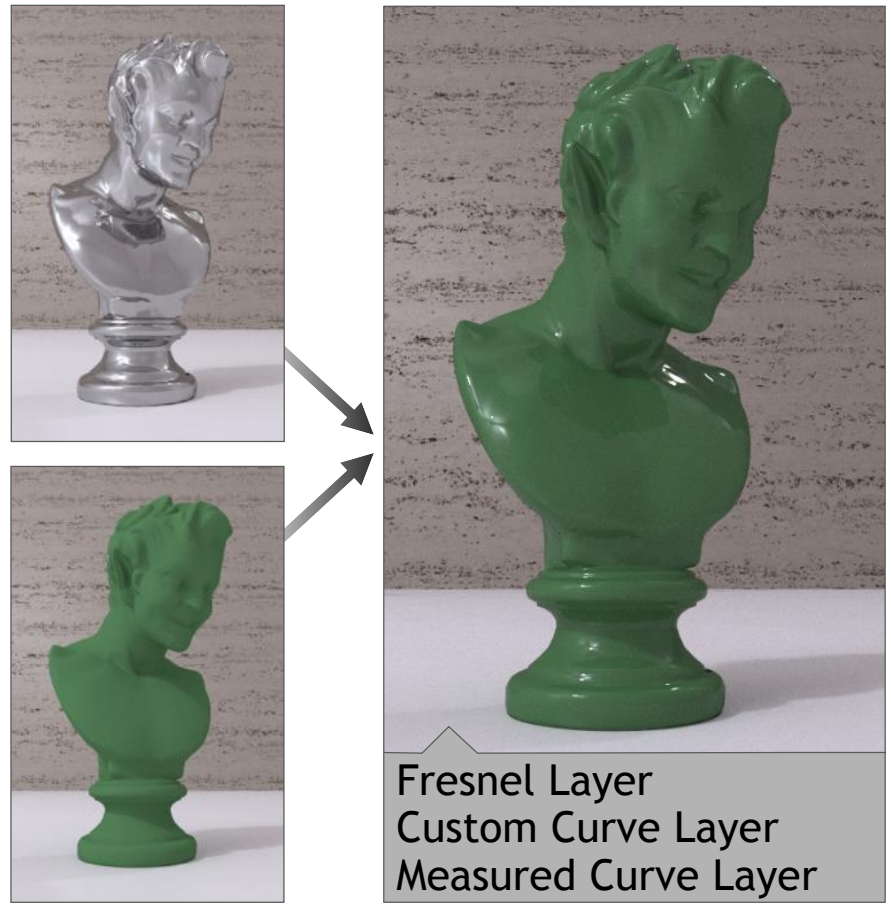
Volume
Distribution
Functions

Henyey-Greenstein

# MDL Distribution Function Modifiers

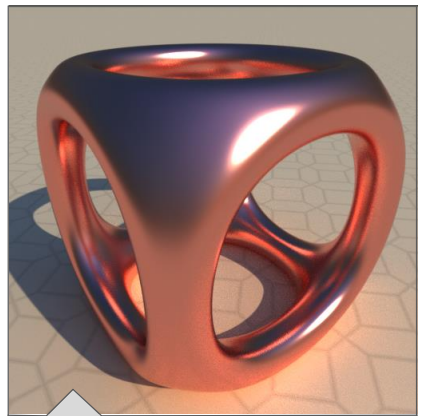



# MDL Distribution Functions Combiners

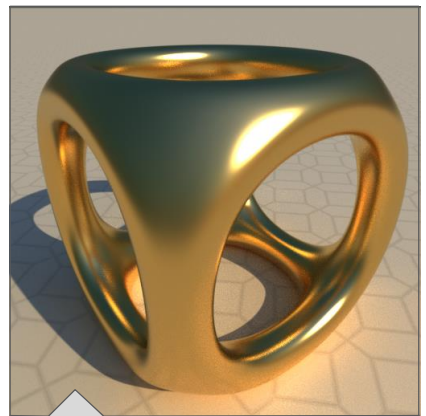



# MDL 1.4: New BSDF

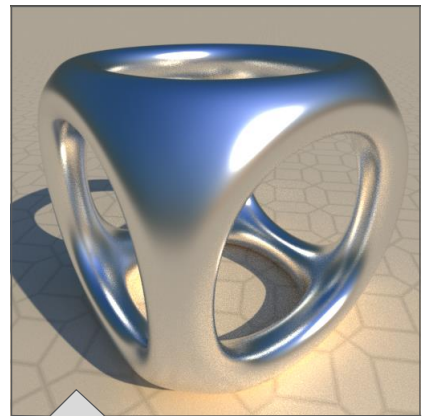
Modifier:
Complex ior factor



Copper

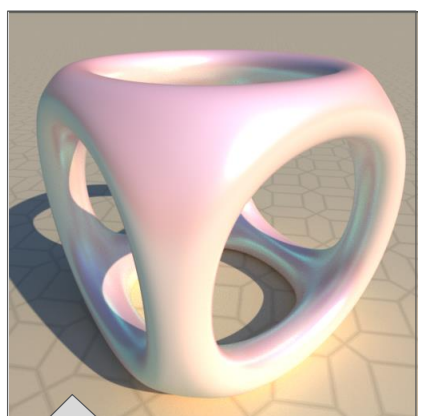


Gold



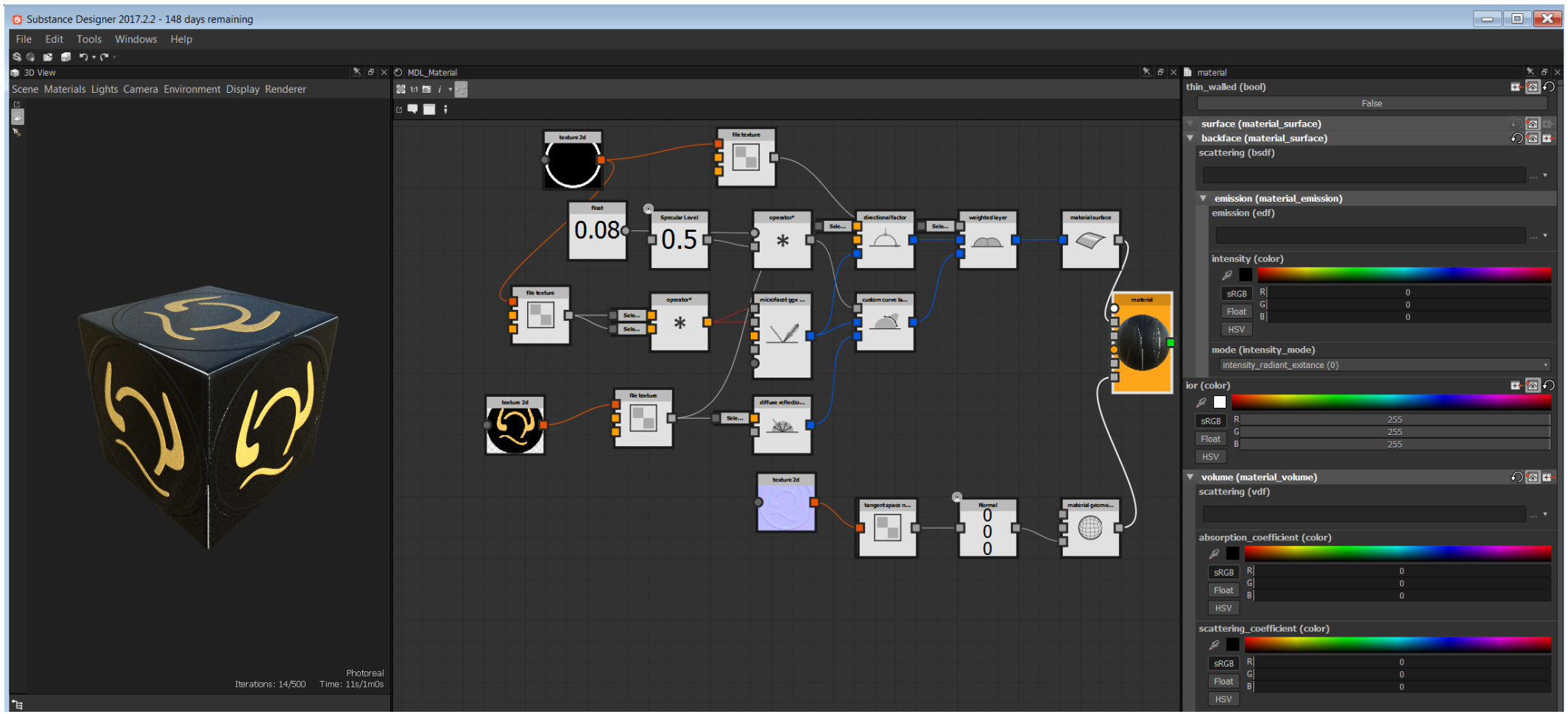
Silver

Combiners:
All weights can be color now



Custom curve layer

# 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();
```

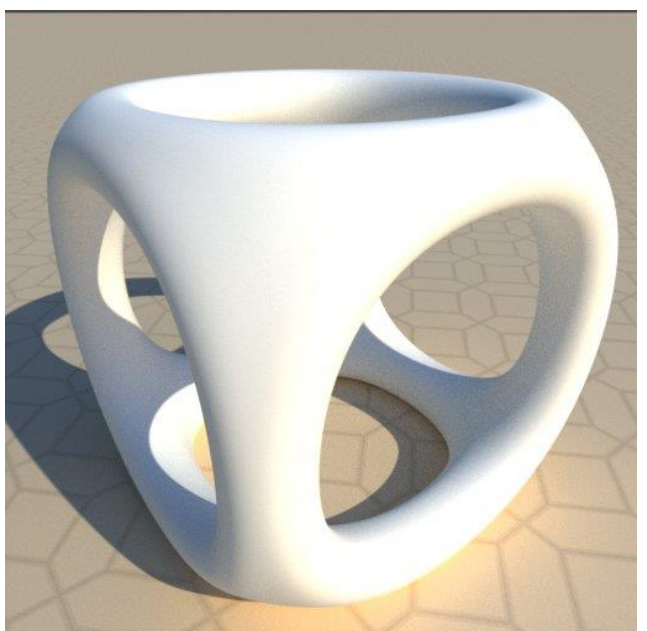
# Defining a Material Using MDL

Creating new materials

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

# Defining a Material Using MDL

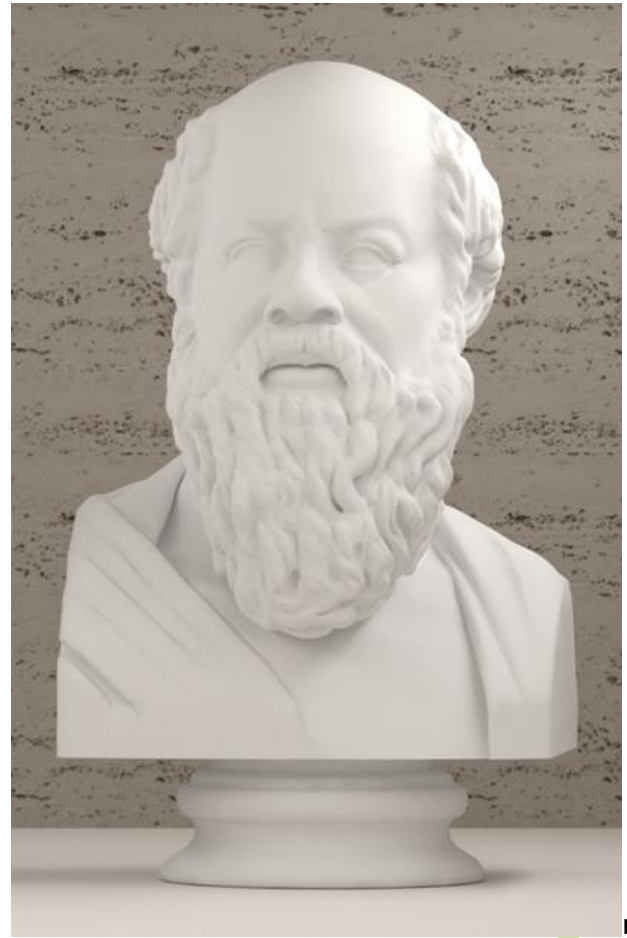
```
material plaster( )
    = material(
        surface: material_surface(
            scattering: df::diffuse_reflection_bsdf()
        )
    );
```



# Defining a Material Using MDL

New materials can have parameters

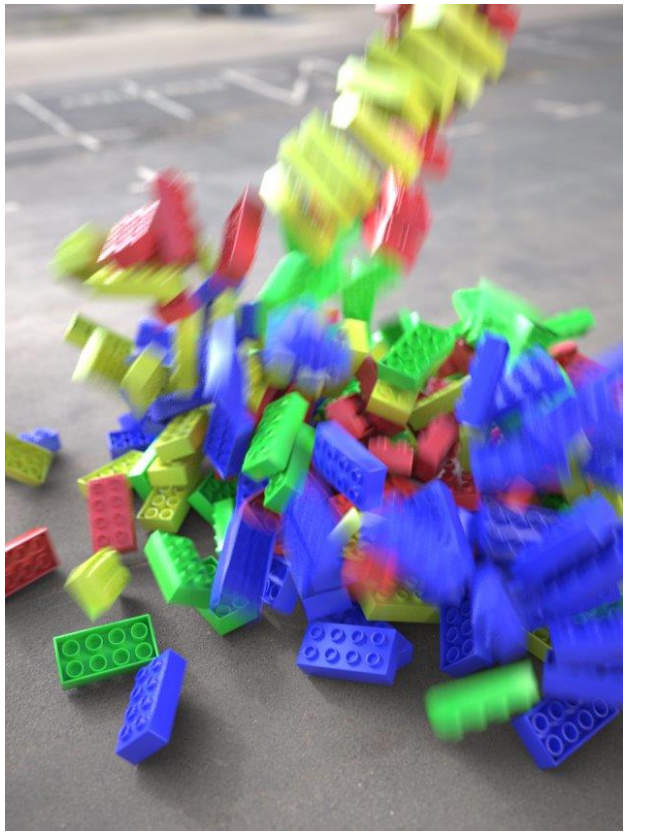
```
material plaster ( color plaster_color = color(.7))
    = material(
        surface: material_surface (
            scattering: df::diffuse_reflection_bsdf (
                tint: plaster_color
            )
        )
    );
```



# Defining a Material Using MDL

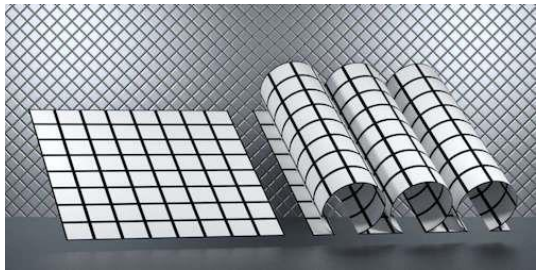
Create complex materials by layering

```
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 Handbook

www.mdlhandbook.com



Added displacement since 2017

## Cloth example

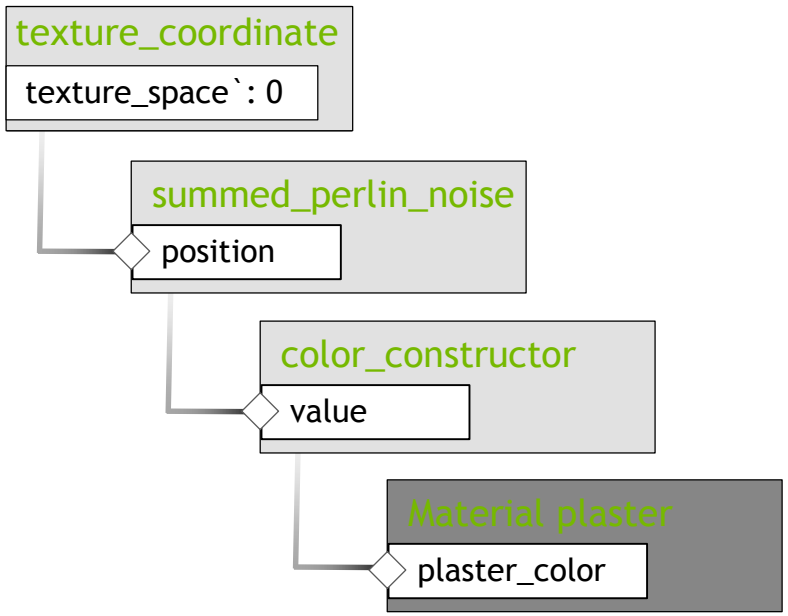
4 anisotropic glossy highlights + translucency

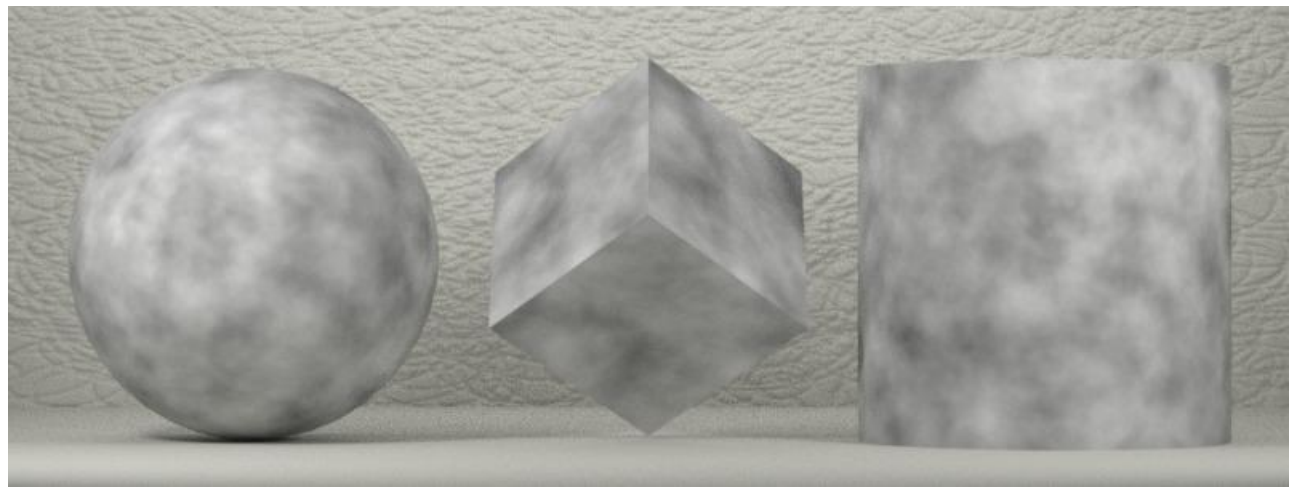
# MDL Procedural Programming Language

## C-like language for function definitions

Function results feed into material and function parameters

“Shader graphs” are equivalent to function call graphs



# 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

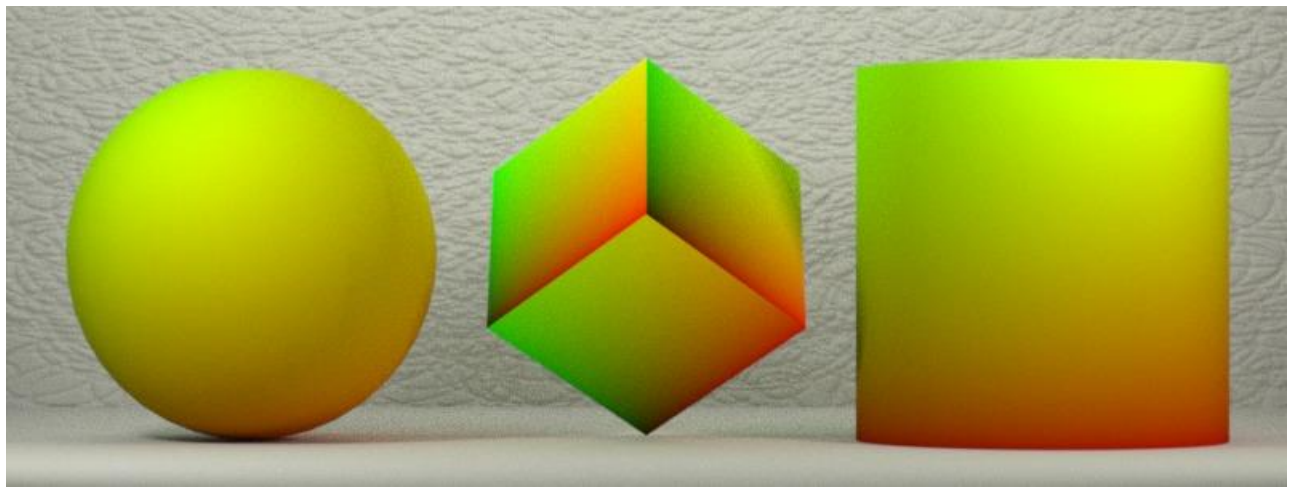
```
color uv_as_color()
{
	return color( state::texture_coordinate(0) );
}
```

# 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

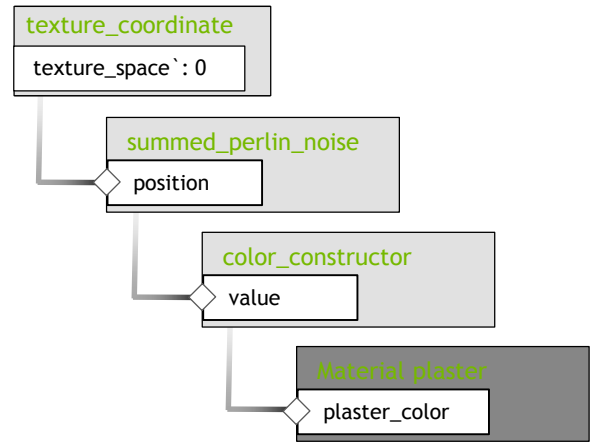
```
float summed_perlin_noise (
     float3 point,
     int level_count=4,
     float level_scale=0.5,
     float point_scale=2.0,
     bool turbulence=false)
{
     float scale = 0.5, noise_sum = 0.0;
     float3 level_point = point;
     for (int i = 0; i < level_count; i++)
     {
          float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

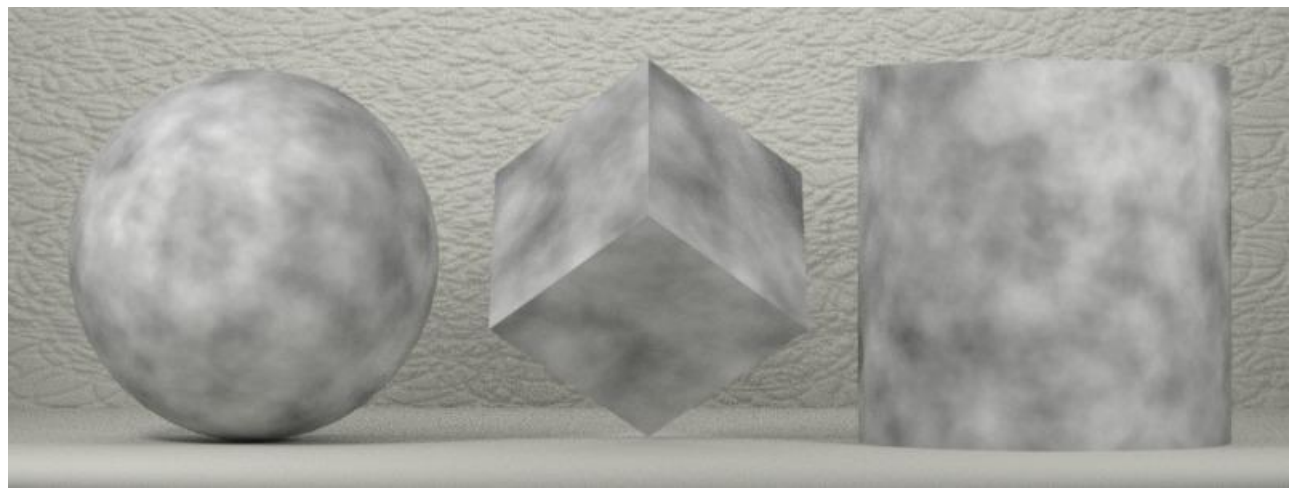
MDL Handbook

# 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 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

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

We use search paths to resolve imports

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

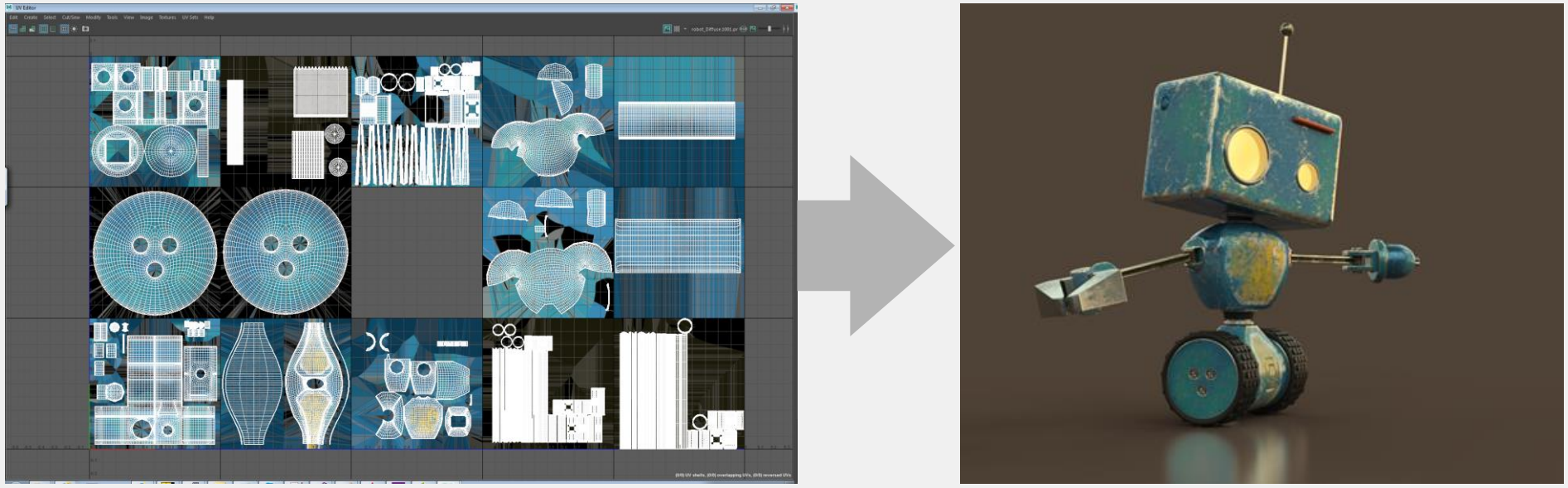
search path

MDL package space

nvidia::vMaterials::Design::Metal::chrome

# UDIM and uv-tiles

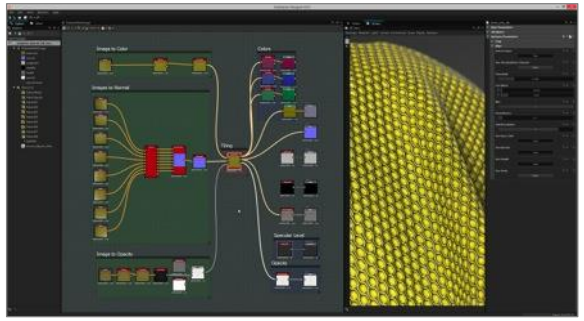
## New in MDL 1.4



UDIM texture layout in Autodesk Maya, rendering in Iray

# Additional MDL Benefits

## Measured Materials

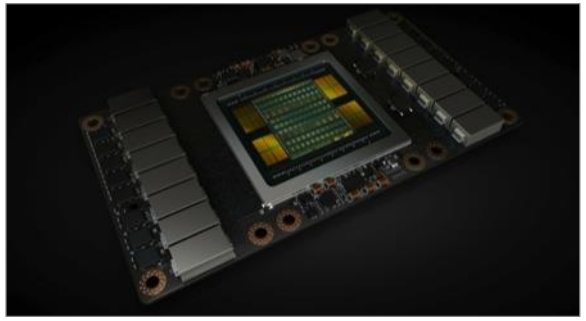


Spatially Varying BRDF

AxF from X-Rite

Measure Isotropic BSDF

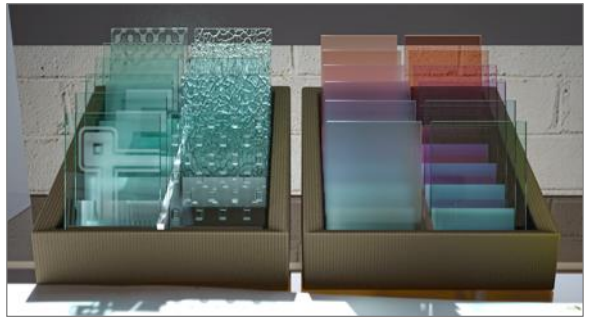
## Designed for Parallelism



Little data dependencies

Side-effect free functions

## Material Catalogs

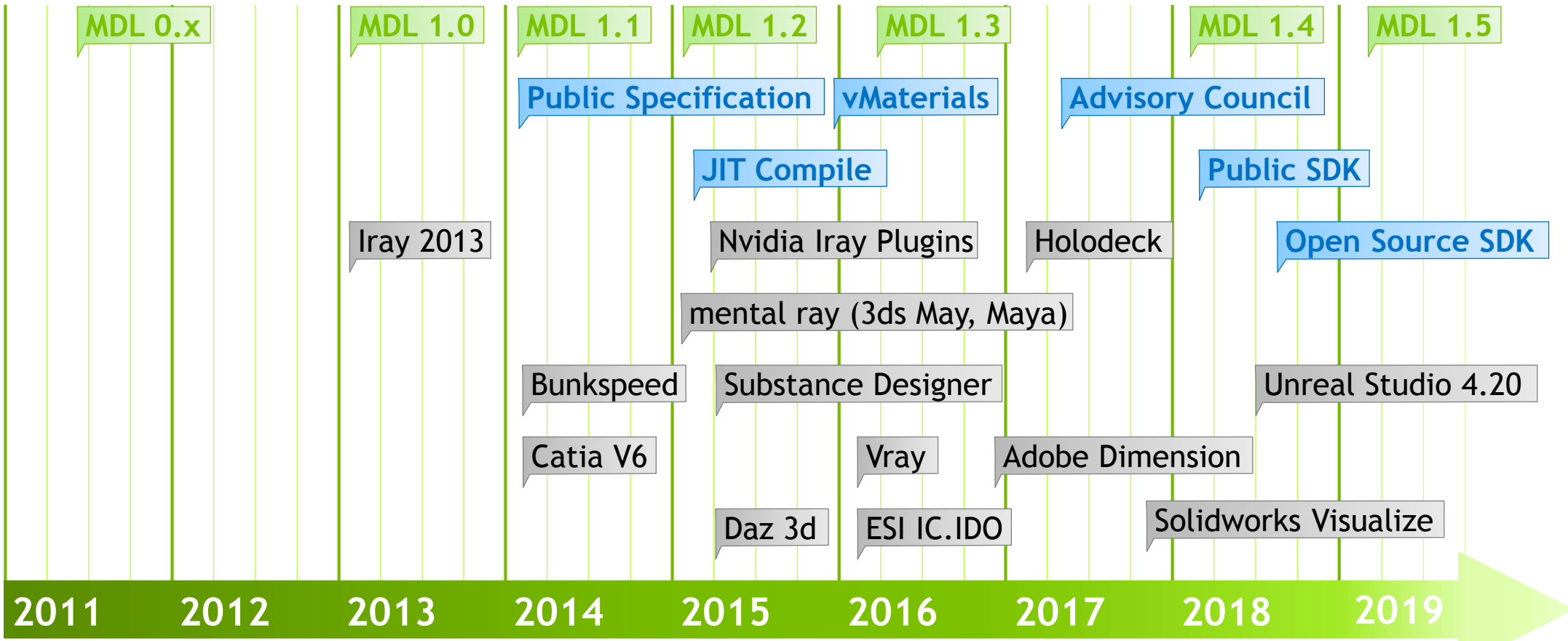


Modules and packages

Archives

# 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

# MDL Ecosystem

## Creation

Substance Designer

## Libraries

NVIDIA VMaterials

Substance Source

Adobe Stock

## Rendering

Chaosgroup Vray

UE4

NVIDIA Iray

## Applications integrating Renderers

Adobe Dimensions

Vray Max/Maya

Unreal Studio

NVIDIA Holodeck

Siemens NX11

Solidworks Vis.

Dassault Catia V6

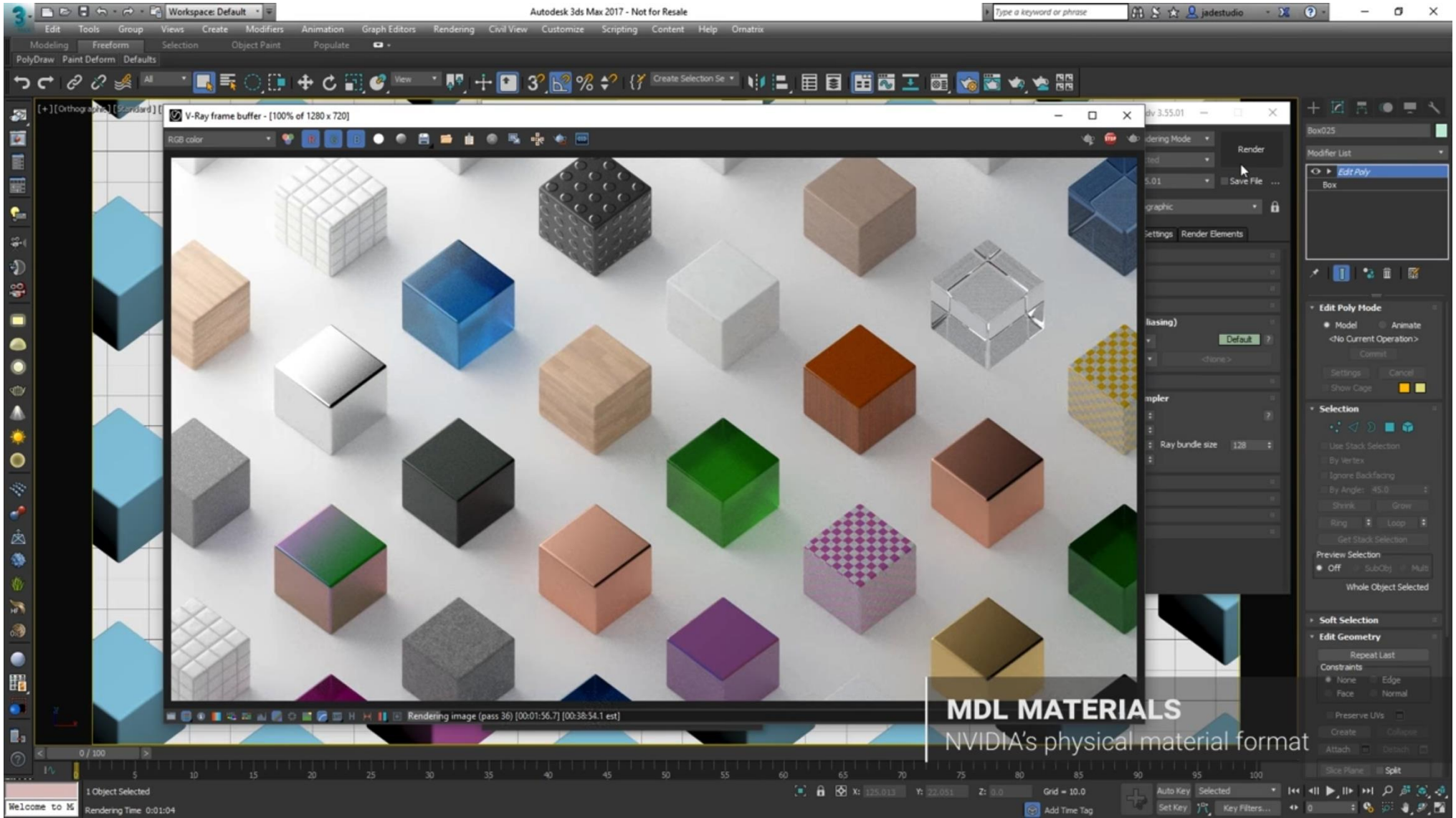
Iray DCC Plugins

Other Iray Products

Patchwork 3D

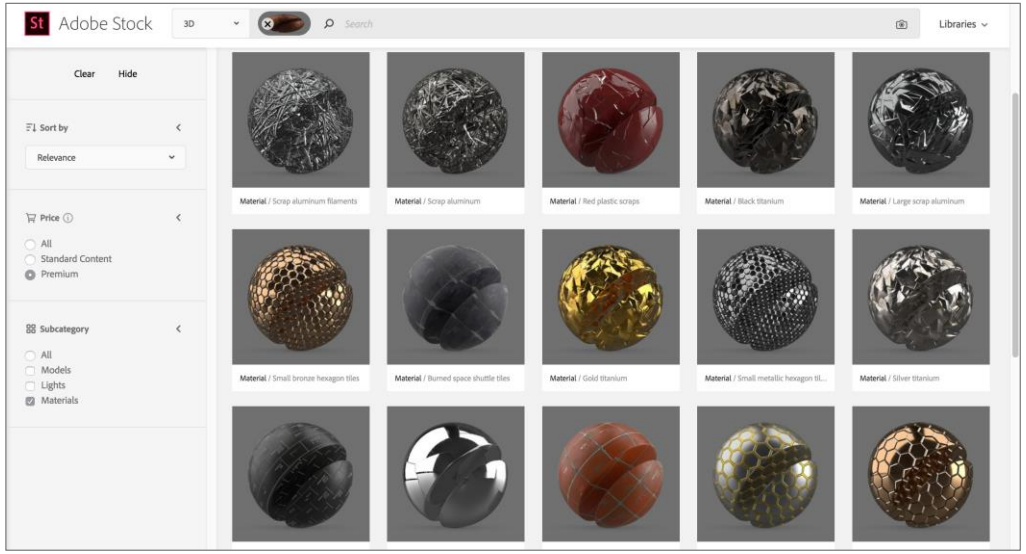
Daz 3D Studio

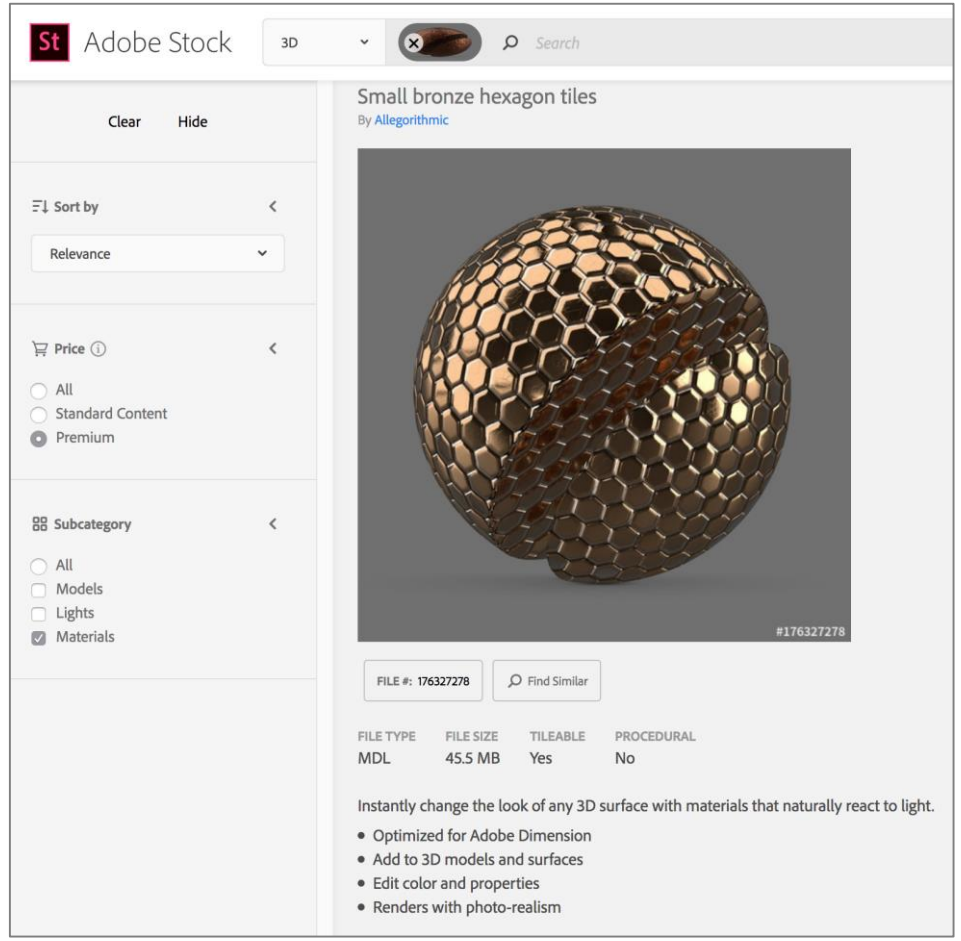
# MDL in VRAY


Autodesk 3ds Max 2017 - Not for Resale
V-Ray frame buffer - [100% of 1280 x 720]
Rendering image (pass 36) [00:01:56.7] [00:38:54.1 est]
MDL MATERIALS
NVIDIA's physical material format

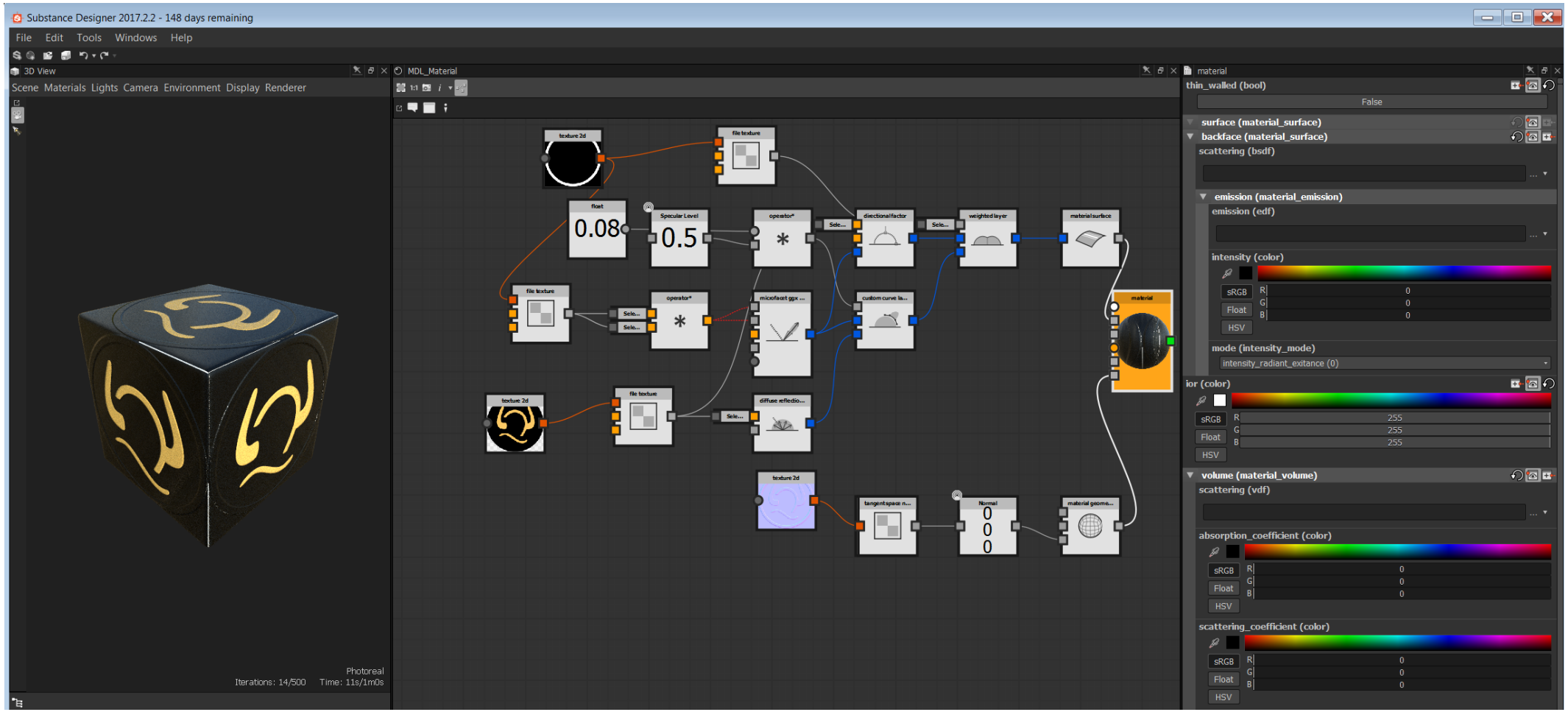
# MDL Adobe Dimension and Adobe Stock

http://www.adobe.com/products/dimension.html

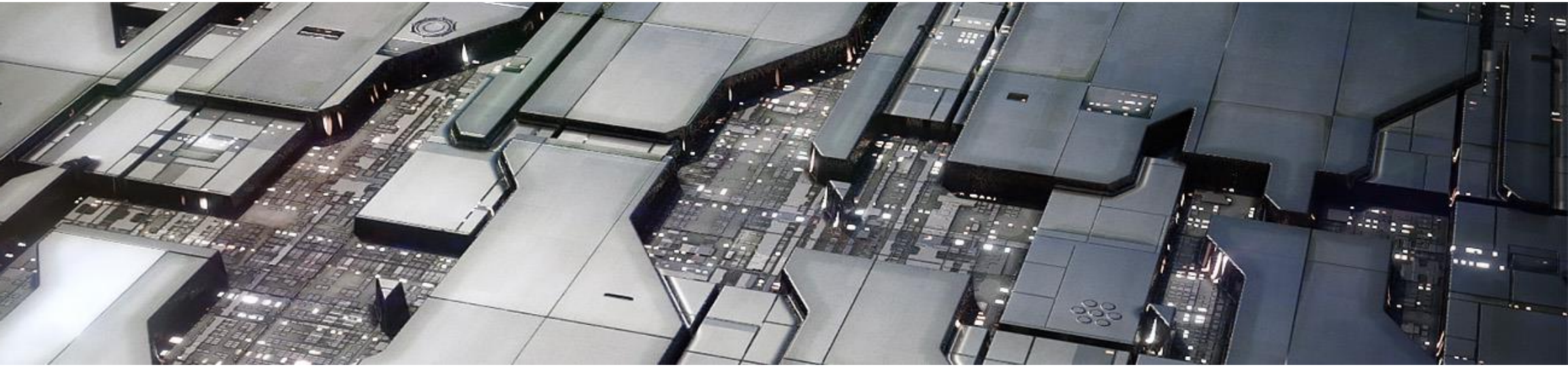


# MDL in Substance Designer



# MDL in Substance Designer

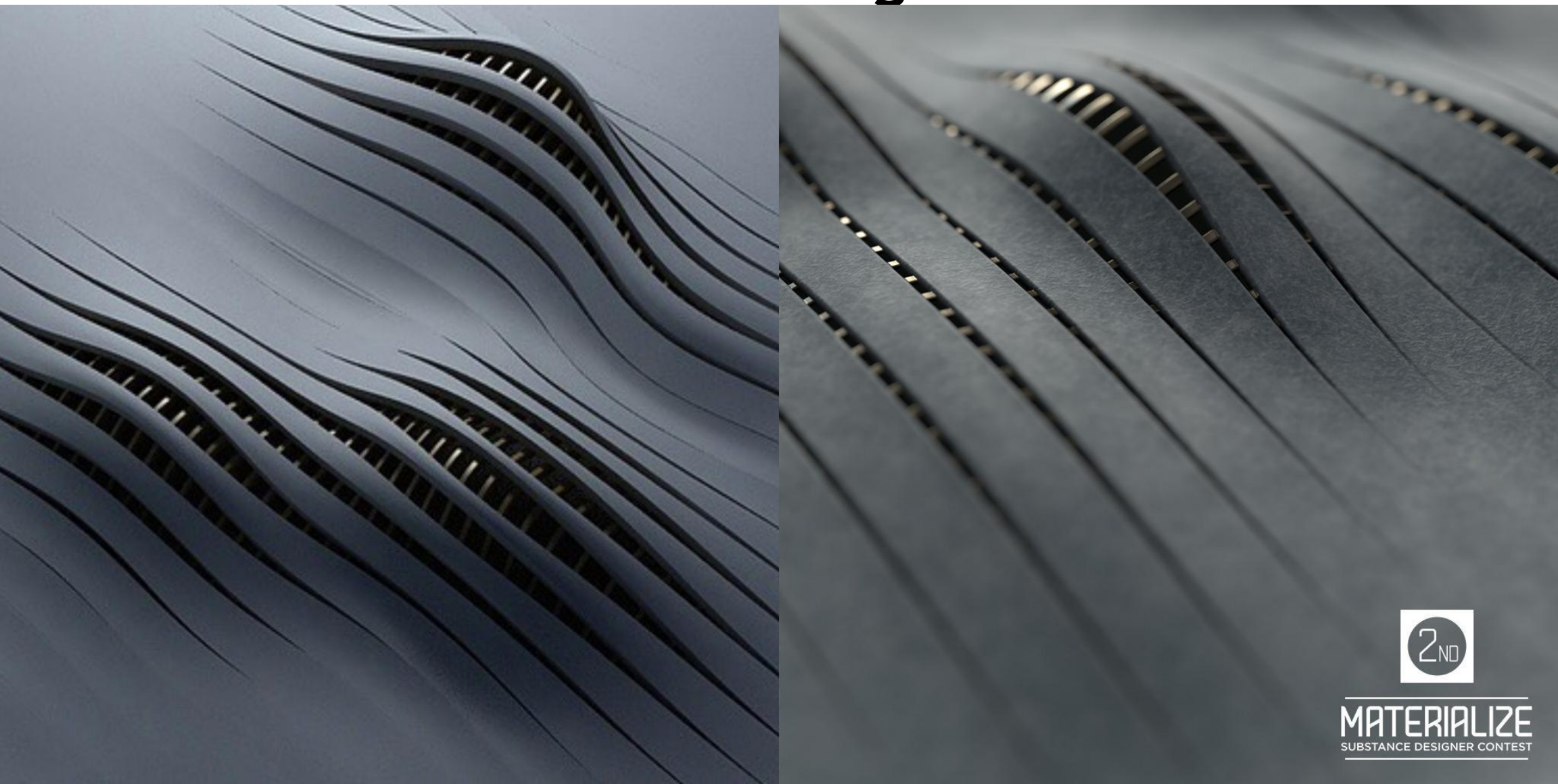


# MDL in Substance Designer

Mark Foreman Malachite with Chrysocolla

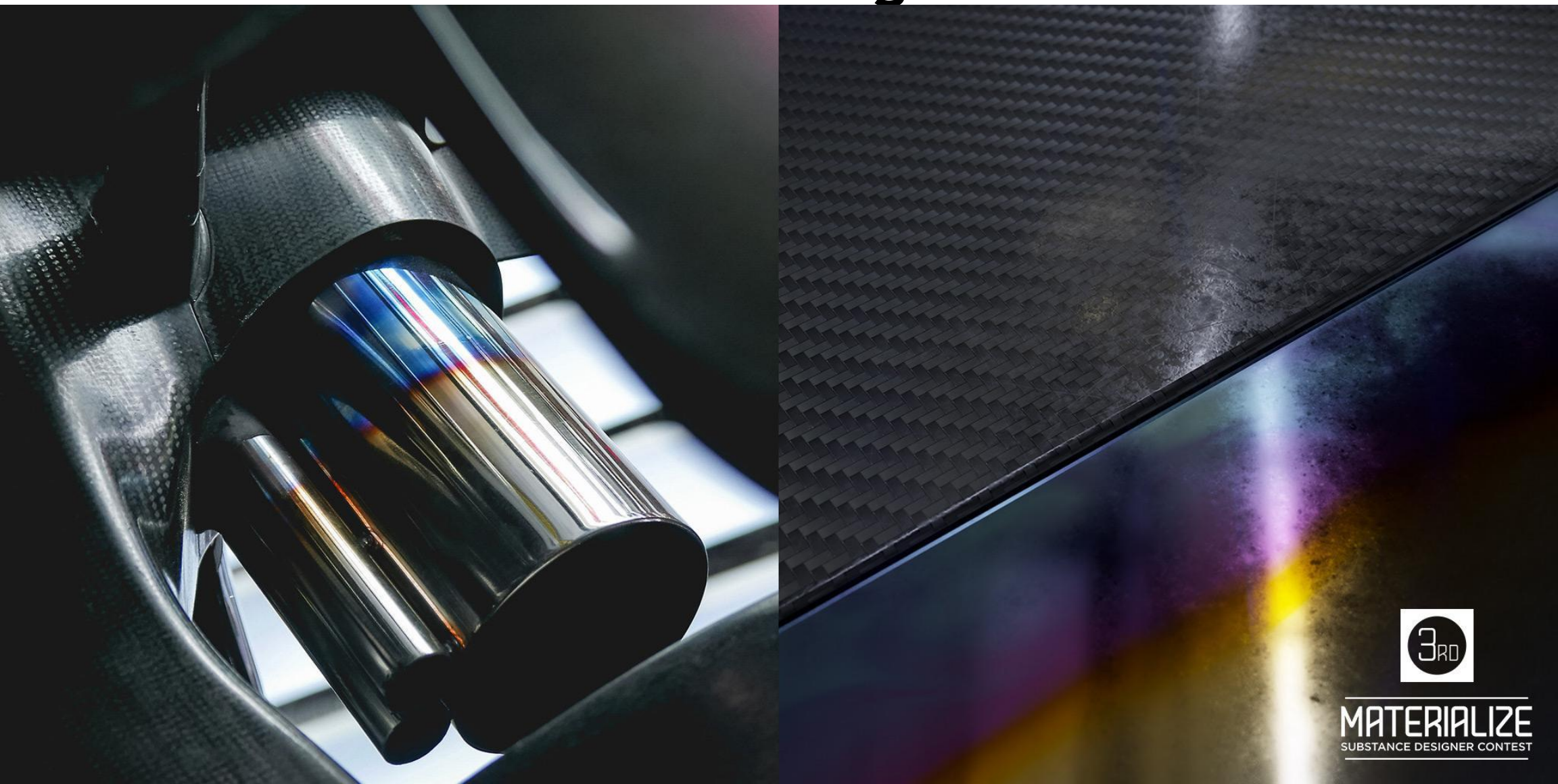
SUBSTANCE DESIGNER

# MDL in Substance Designer

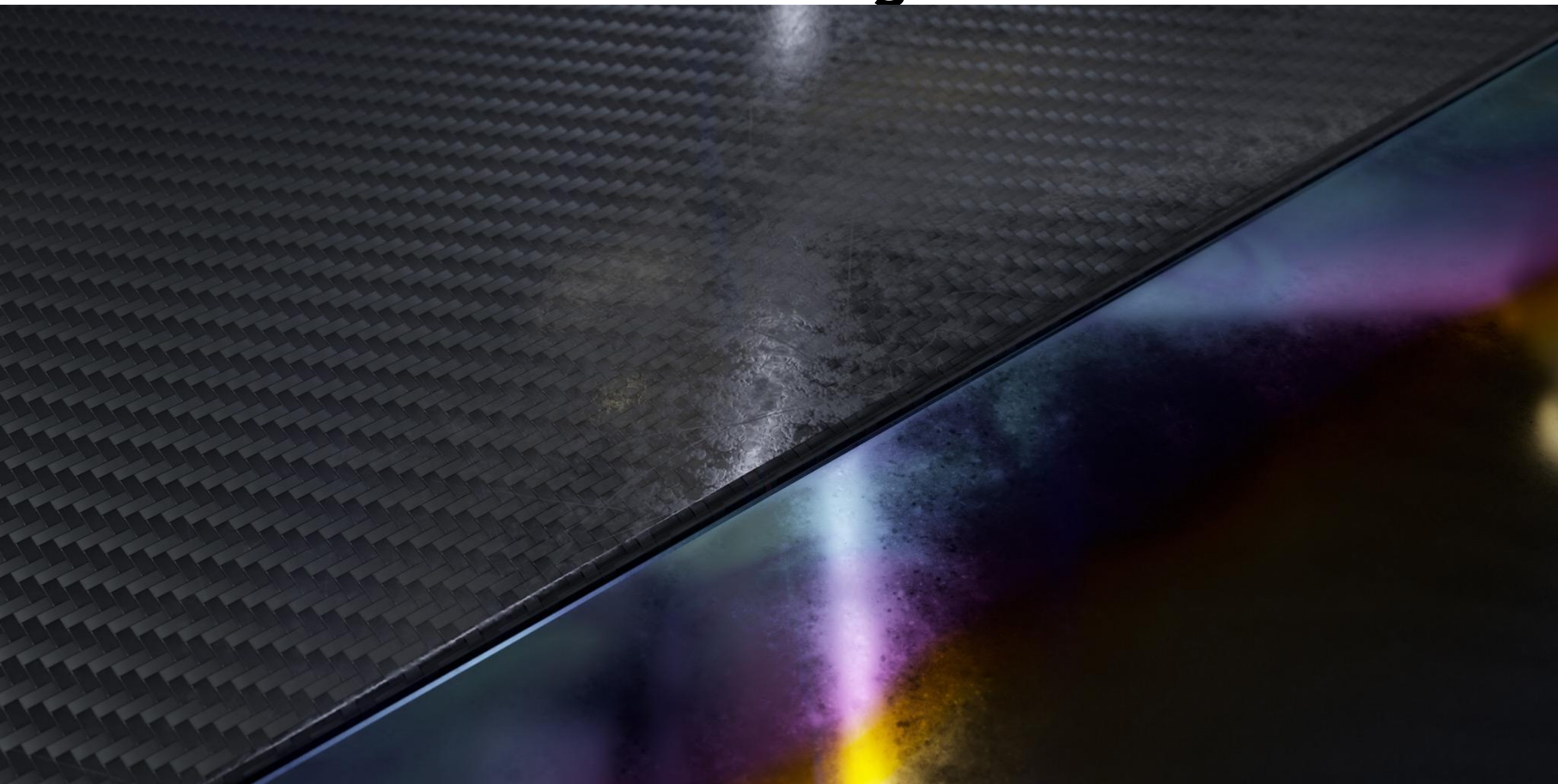


# MDL in Substance Designer

# MDL in Substance Designer



# MDL in Substance Designer



# Focus on Material Exchange

## Freely choose where to author material content


create

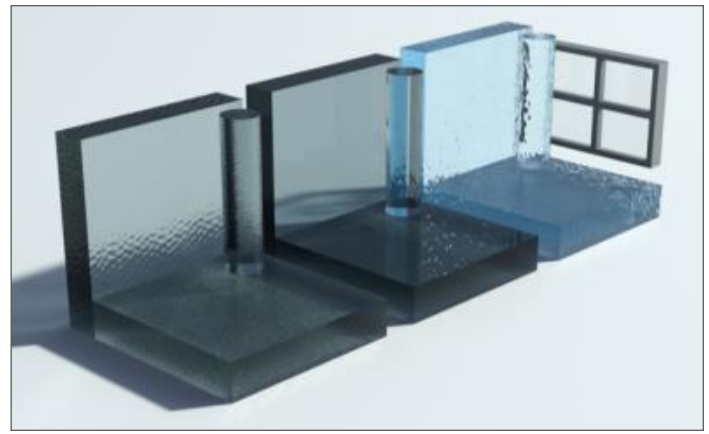

modify


consume

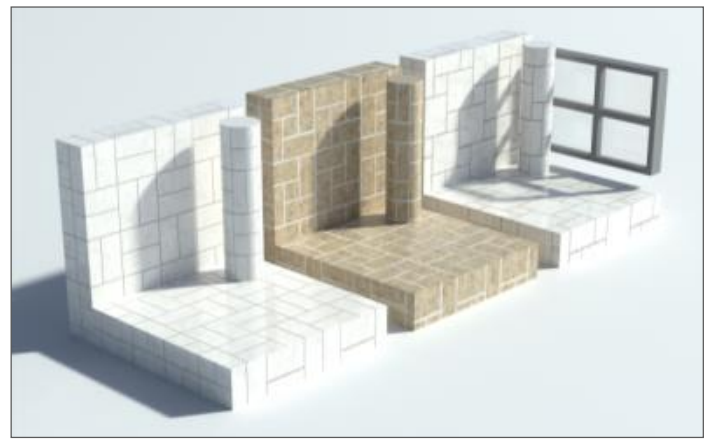
# NVIDIA vMaterials 1.5

## ~1600 MDL materials verified for accuracy - FREE TO USE



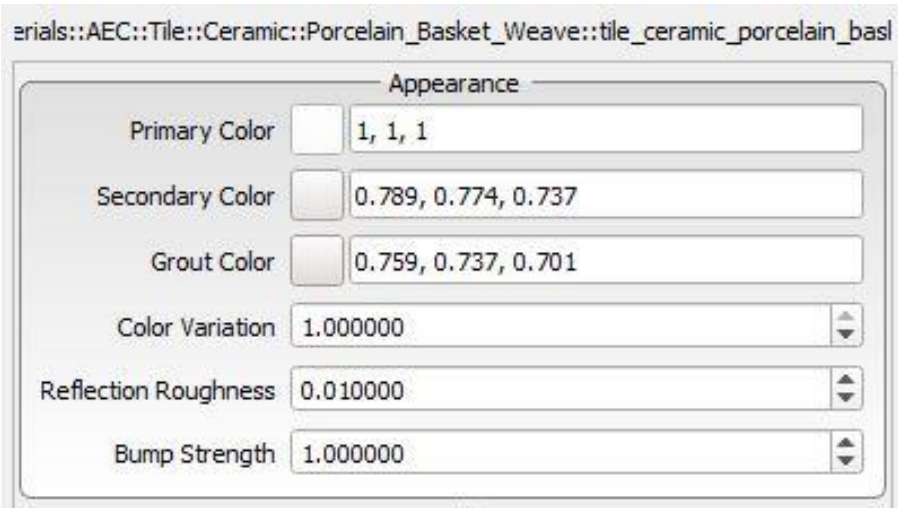



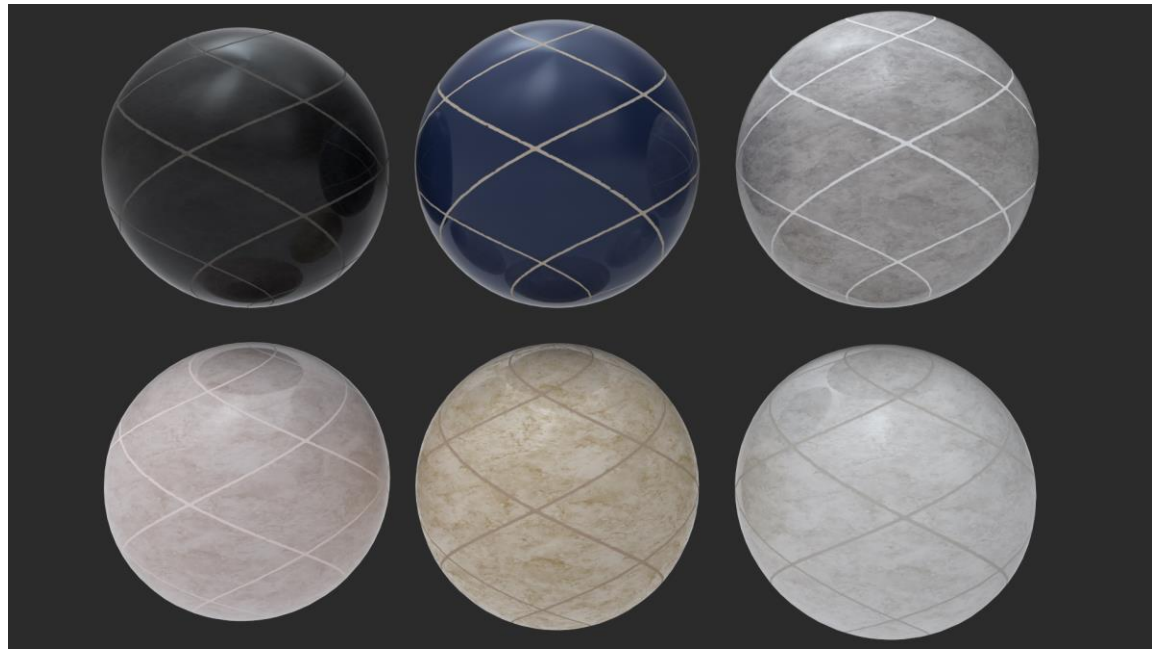




# NVIDIA vMaterials 1.5

## 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

## Use the MDL SDK

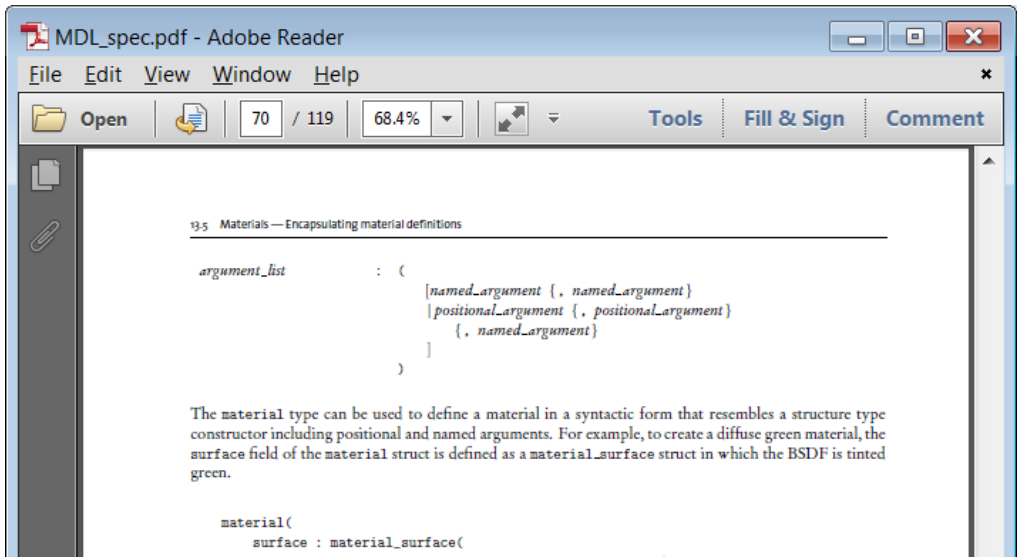
Published under the NVIDIA Designworks License and …

# Write Your Own Compiler

## MDL Specification

Language specification document
Free to use

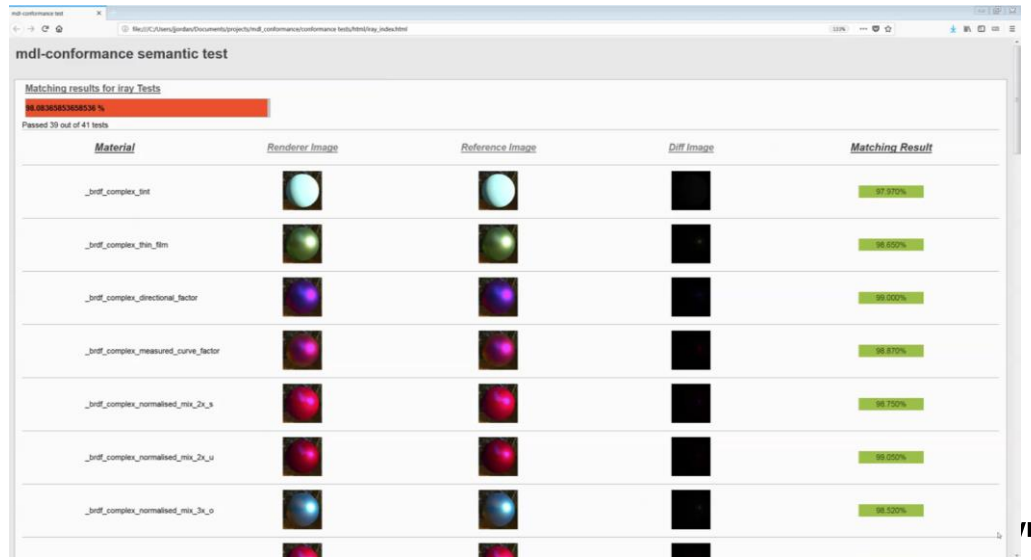
http://www.nvidia.com/mdl/




## MDL conformance test suite

Syntactic conformance tests
Semantic conformance tests

Available on request



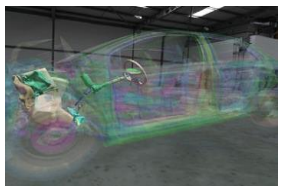
# NVIDIA. DESIGNWORKS™

## RENDERING


Iray SDK

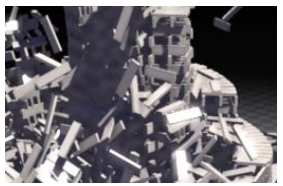

OptiX SDK


MDL SDK


NV Pro Pipeline

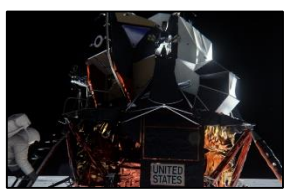

vMaterials

## PHYSICS


PhysX

## VOXELS


GVDB Voxels


VXGI

## VIDEO


GPUDirect for Video

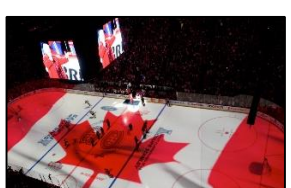

Video Codec SDK

## MANAGEMENT


GRID SW MGMT SDK


NVAPI/NVWMI

## DISPLAY


Multi-Display


Capture SDK


Warp and Blend

**https://developer.nvidia.com/designworks**

# MDL SDK 2018.1.1

## Features

MDL 1.4

DB for MDL definitions

DAG view on materials
several compilation modes

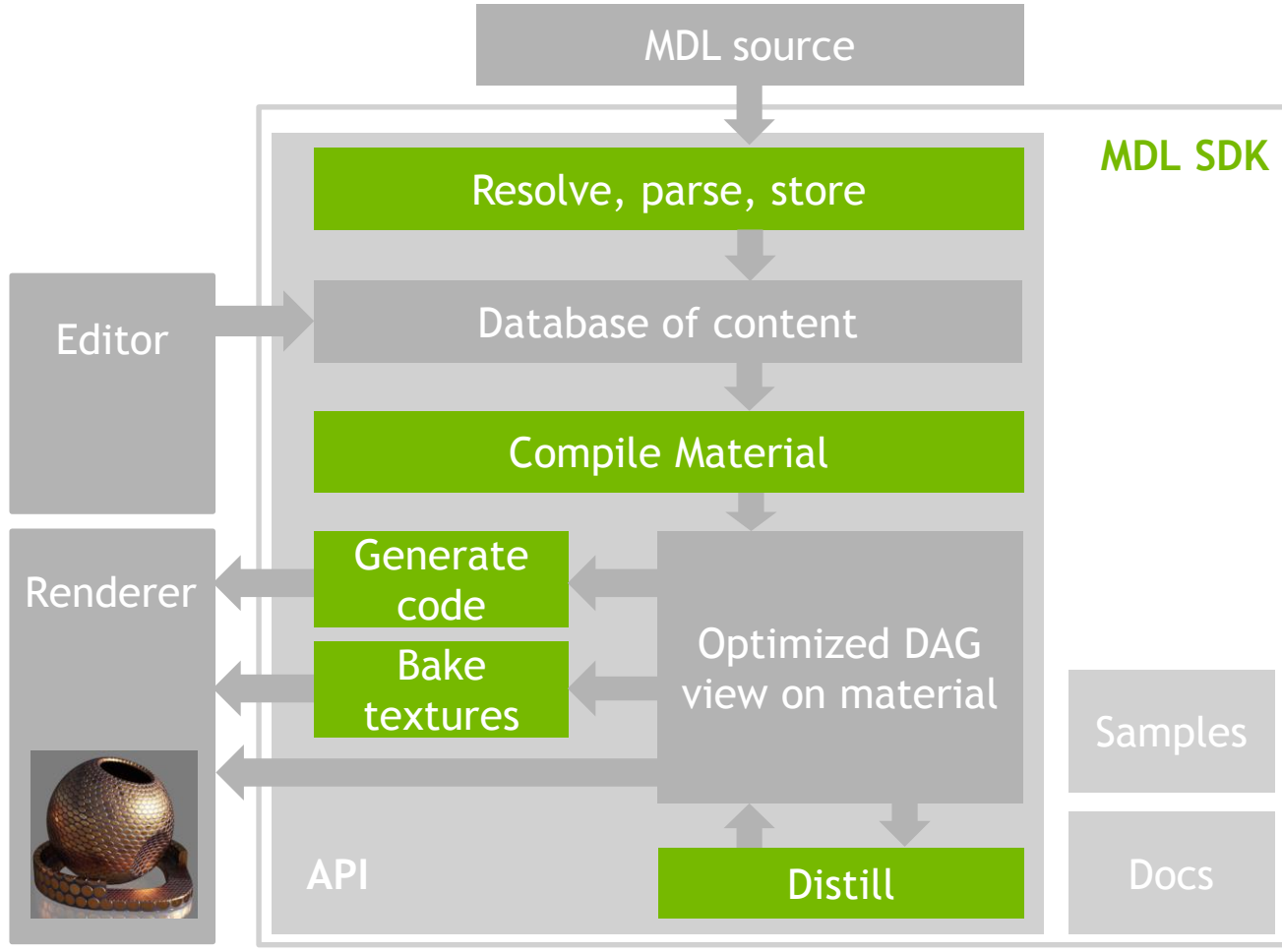
MDL editing

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

Distiller and texture baker

Samples

Documentation and tutorials



# MDL SDK 2018 - 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

MDL archive access accelerated

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

Samples reorganized

CMake build for samples

# 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

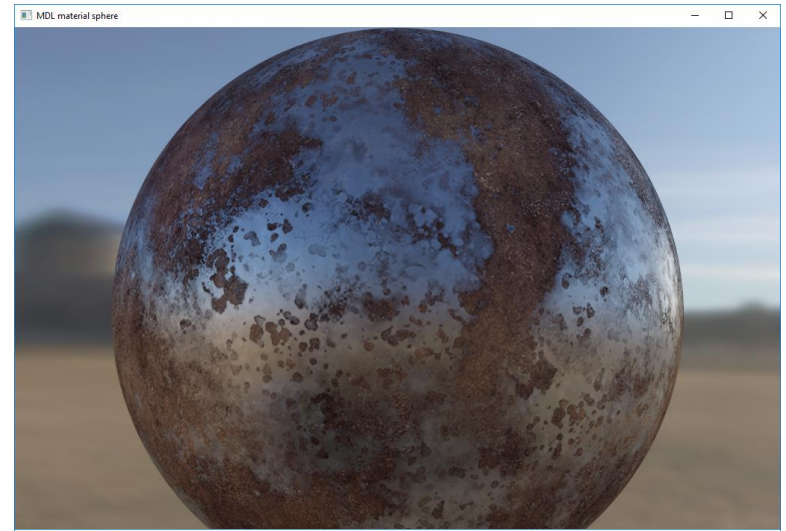
## The MDL SDK directly generates code for use in RTX enabled renderer

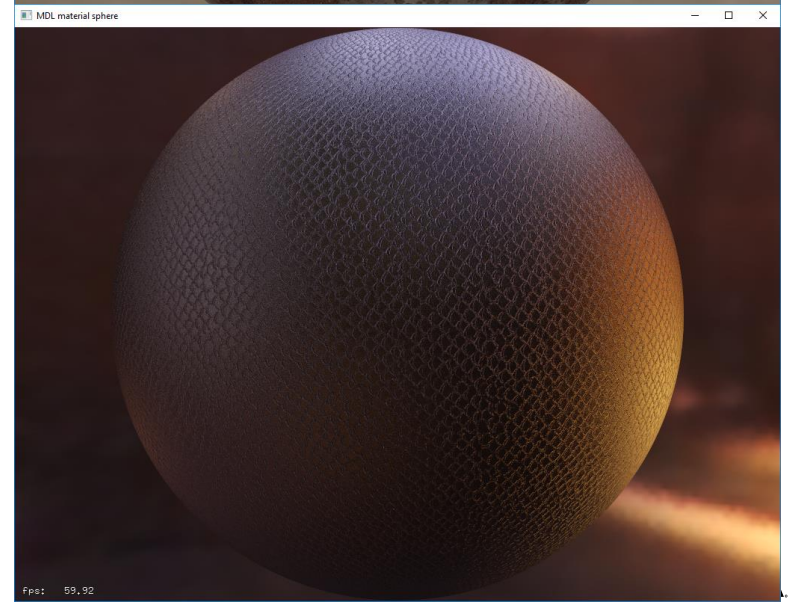
MDL SDK generates PTX material code suitable to be called by OptiX and used with RTX

- available since MDL SDK 2018.1
- Sample program available as part of Optix 5.1

MDL SDK will generate HLSL material code suitable to be used in an DXR based renderer (upcoming feature)

Integrating MDL with an RTX based renderer is simple!



# 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

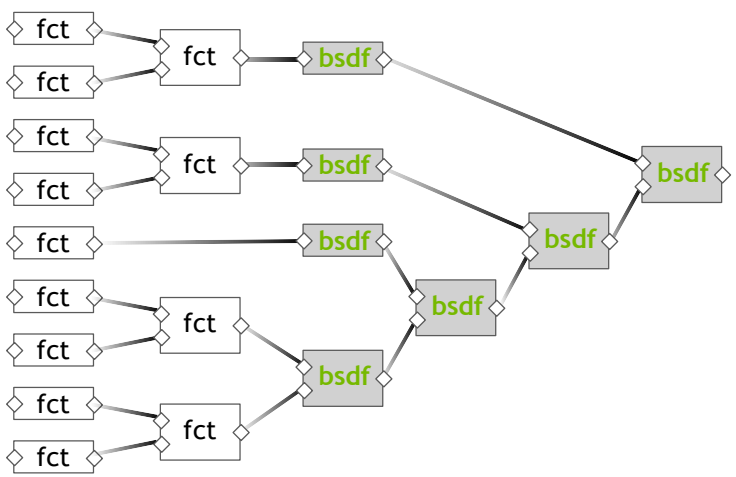
Complex procedurals

Distillation →

**Fixed Material Model**

Simple BSDF structure

One texture per parameter



# Distillation to Fixed Material Model

**MDL Material**

Complex BSDF layering

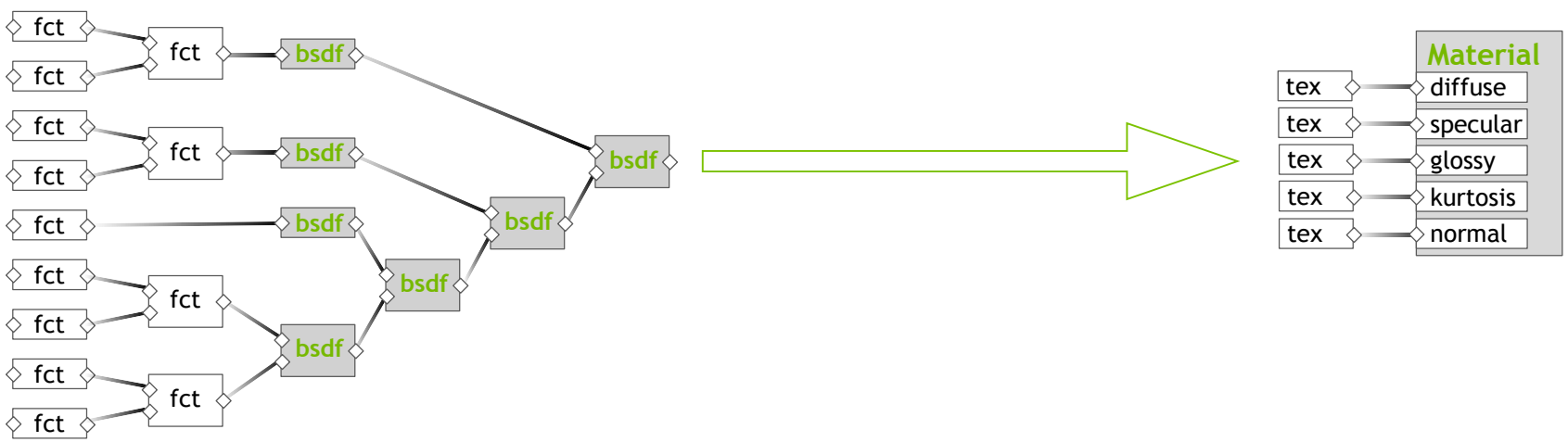
Complex procedurals

Distillation →

**Fixed Material Model**

Simple BSDF structure

One texture per parameter



# Distillation to Fixed Material Model

**MDL Material**

Complex BSDF layering

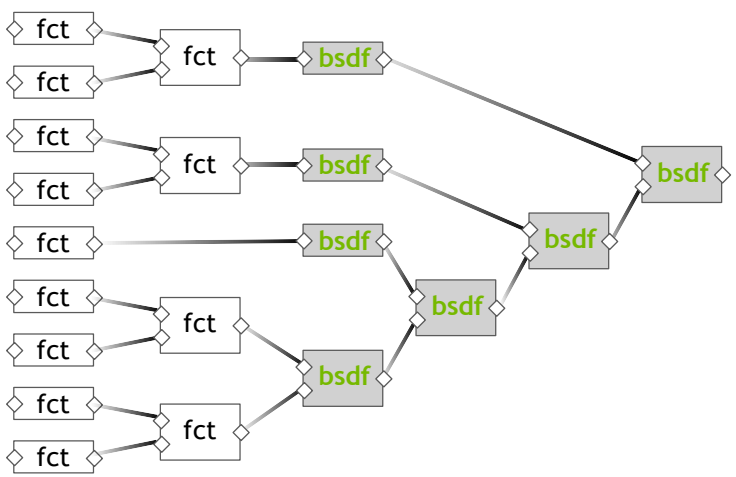
Complex procedurals

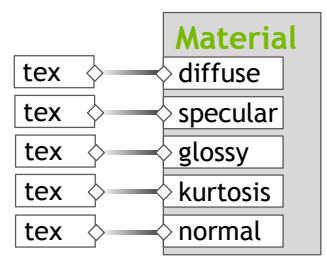
Distillation →

**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

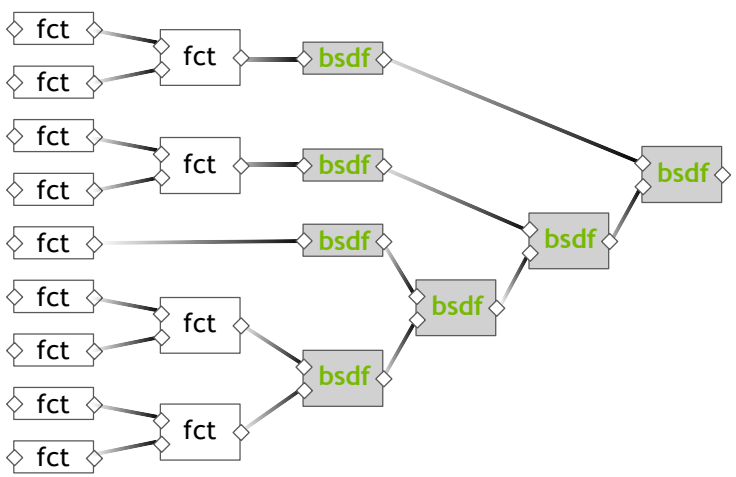
Complex procedurals

Distillation →

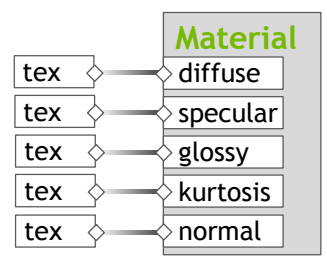
**Fixed Material Model**

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

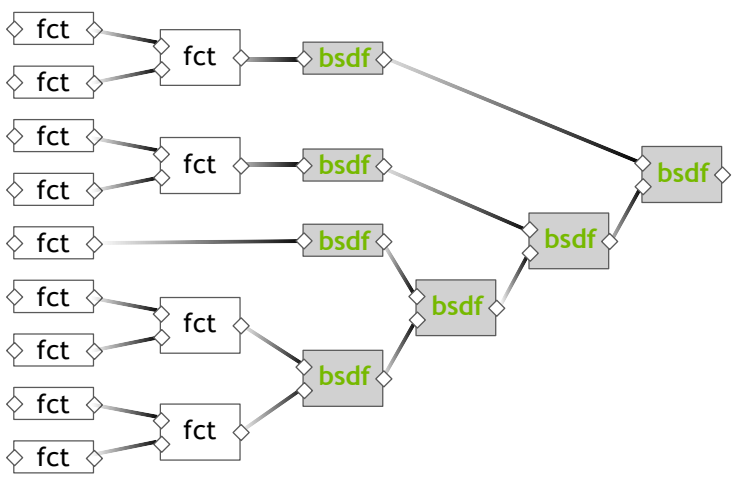
Complex procedurals

Distillation →

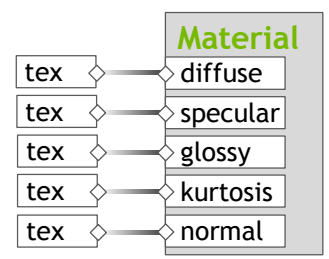
**Fixed Material Model**

Simple BSDF structure

One texture per parameter




Fast projection of material instances: Realtime editing

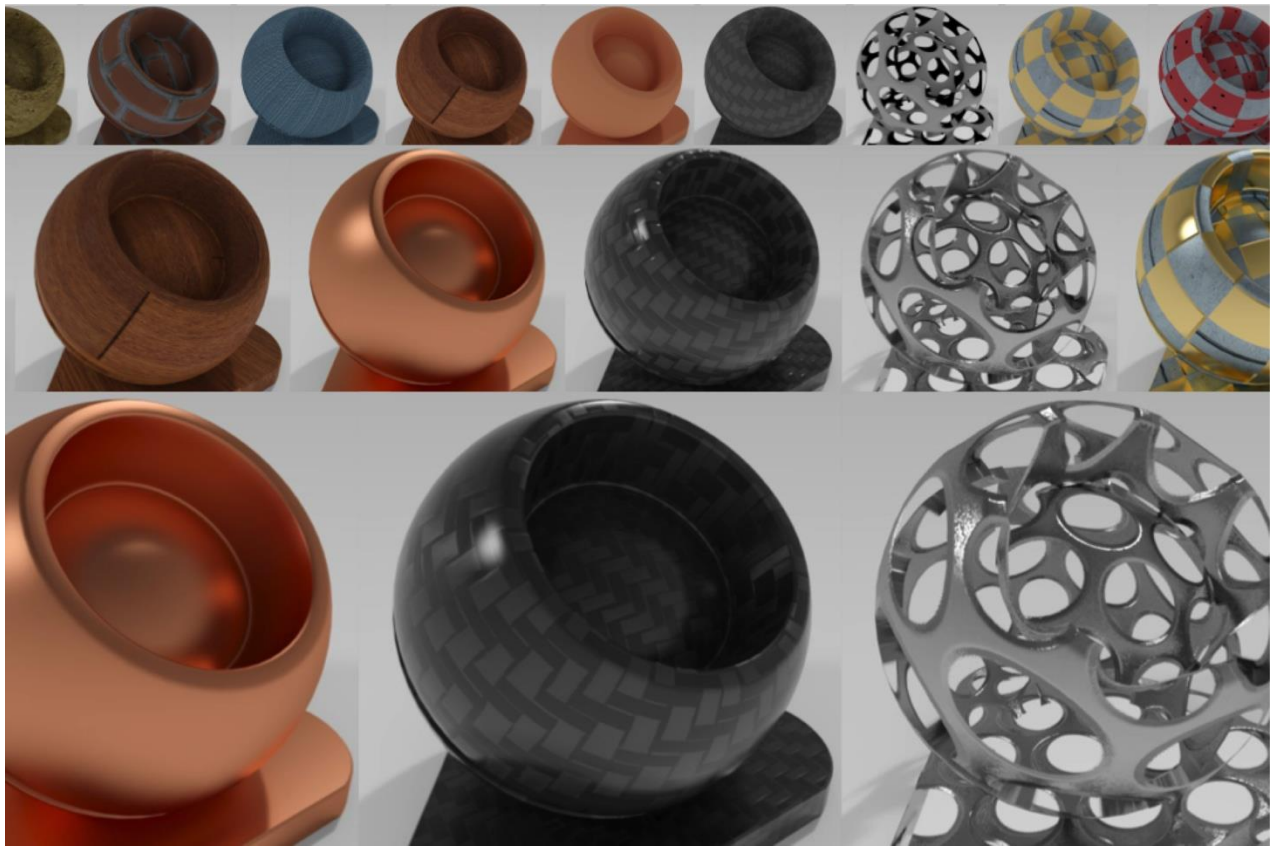



Approximate render result: Some materials will look quite different

**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

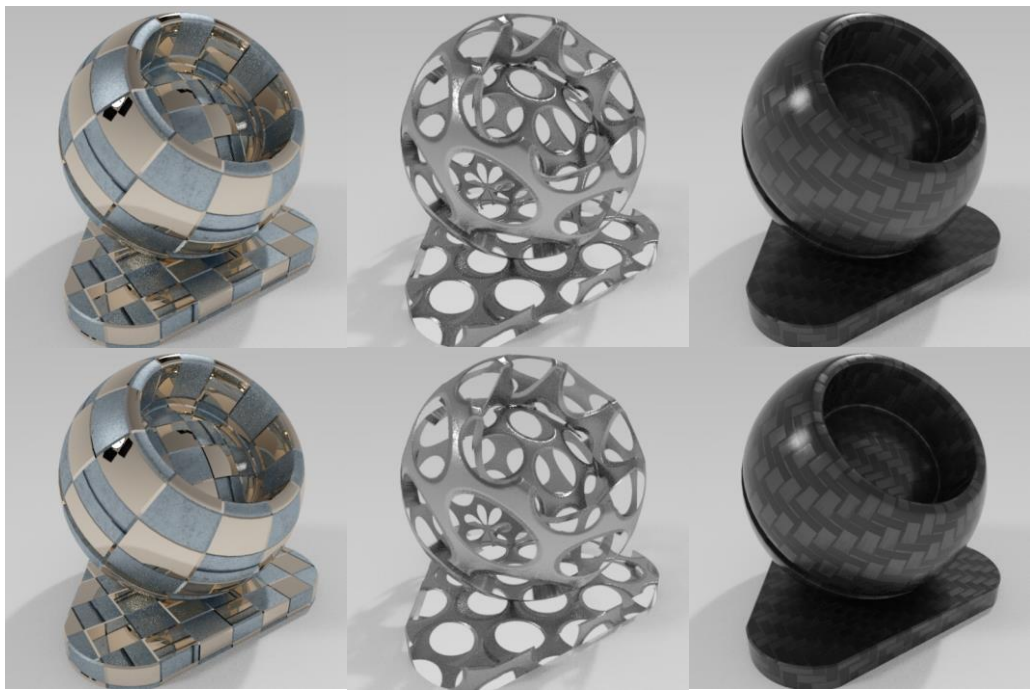
## Released as part of Iray/MDL SDK

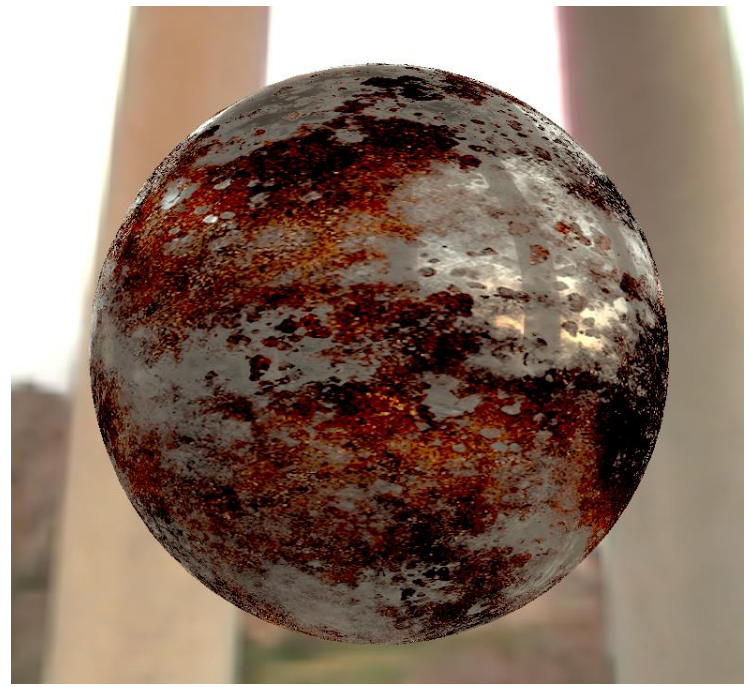
Example: UE4 target with clearcoat and transparency through alpha

GLSL rendering sample using Distilling and baking

MDL

UE4





# May the Source Be with You


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

# May the Source Be with You

## NVIDIA Open Sources the MDL SDK

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


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

# May the Source Be with You

## NVIDIA Open Sources 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


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

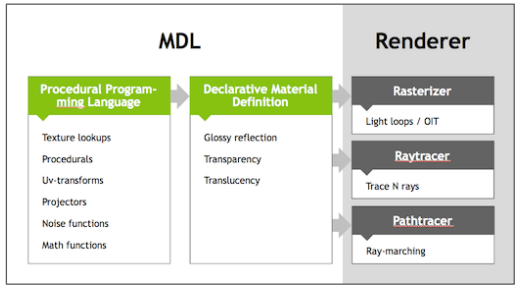
# MDL Core API

## A Lower-level Compiler API in the MDL SDK

| MDL SDK API | MDL Core API |
| --- | --- |
| Higher-level API for easy integration | API close to the compiler |
| Reference counted interfaces | Objects managed in arenas |
| Mutable objects | Immutable objects |
| In-memory store | Stateless compiler |
| Texture and resource importer | Callbacks |

# 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 Backend 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*
*Material Definition Language ▫ Handbook*
*NVIDIA Material Definition Language ▫ Language Specification*

## MDL@GTC On-Demand

https://on-demand-gtc.gputechconf.com/