Sharing Physically Based Materials Between Renderers with MDL

Jan Jordan Software Product Manager MDL

Lutz Kettner Director Advanced Rendering and Materials

March 18, GTC San Jose 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

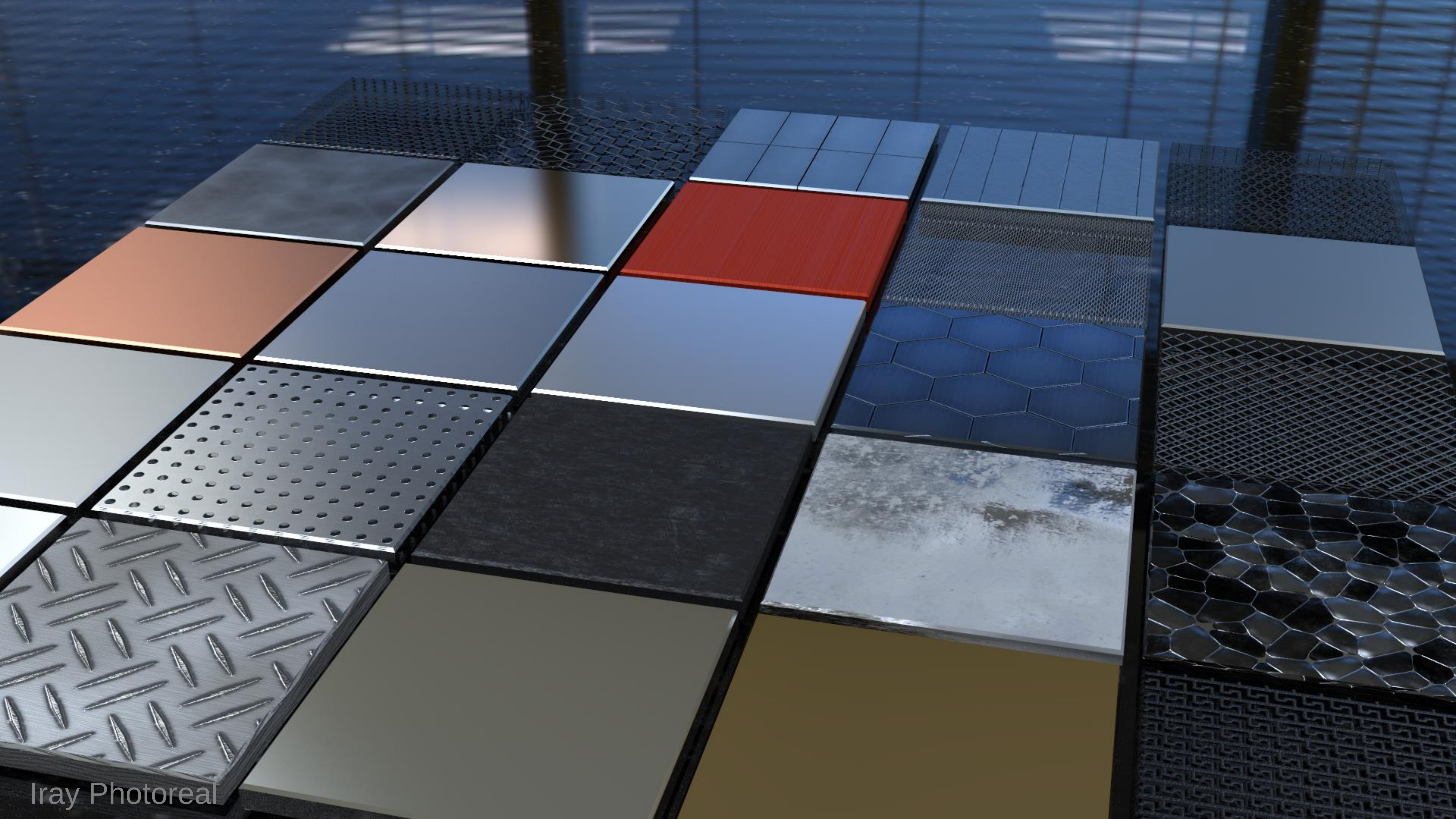
NVIDIA. MDL

The NVIDIA Material Definition Language (MDL)

is technology developed by NVIDIA

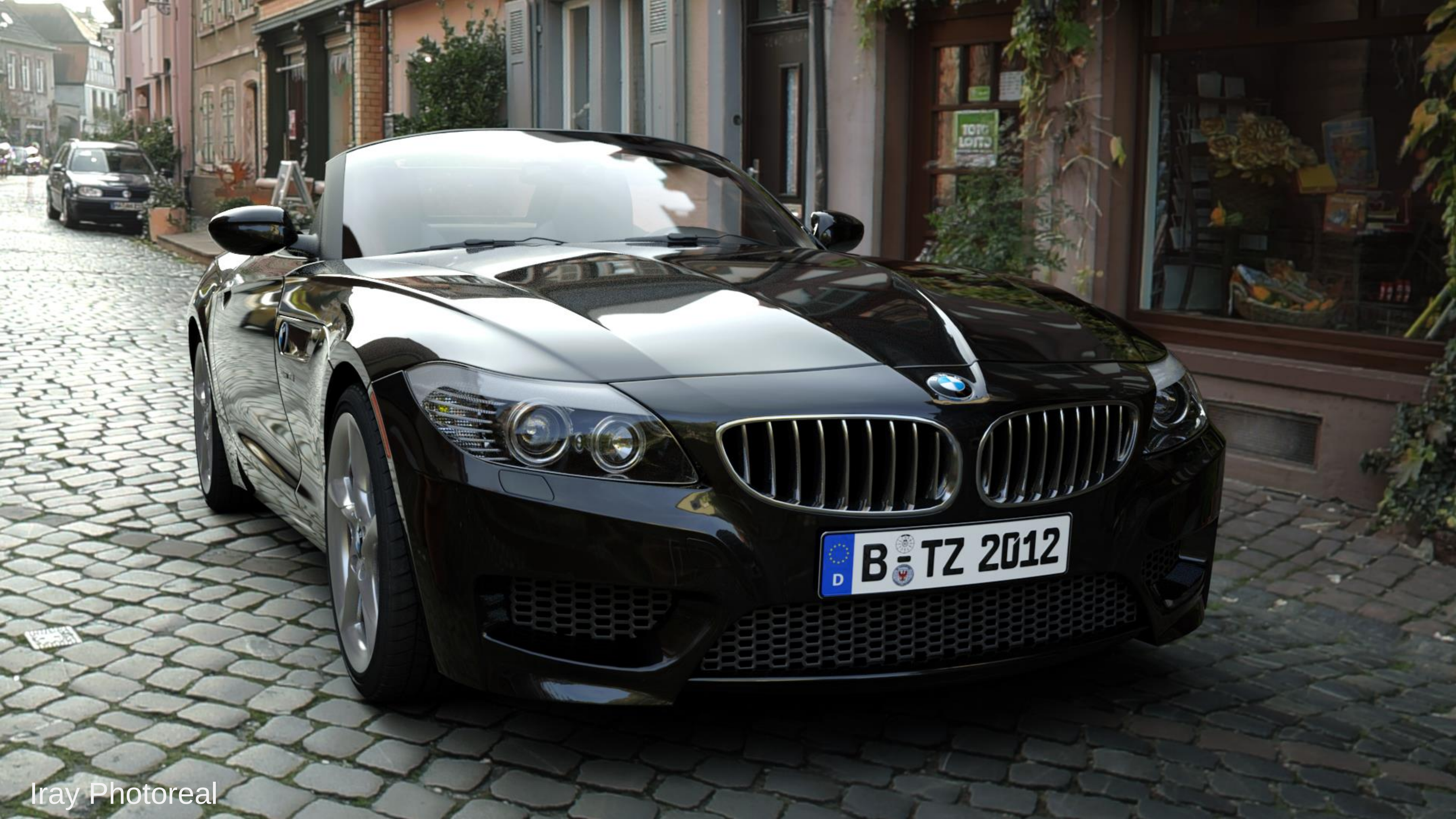
to define physically-based materials

for physically-based rendering solutions.



Iray Photoreal

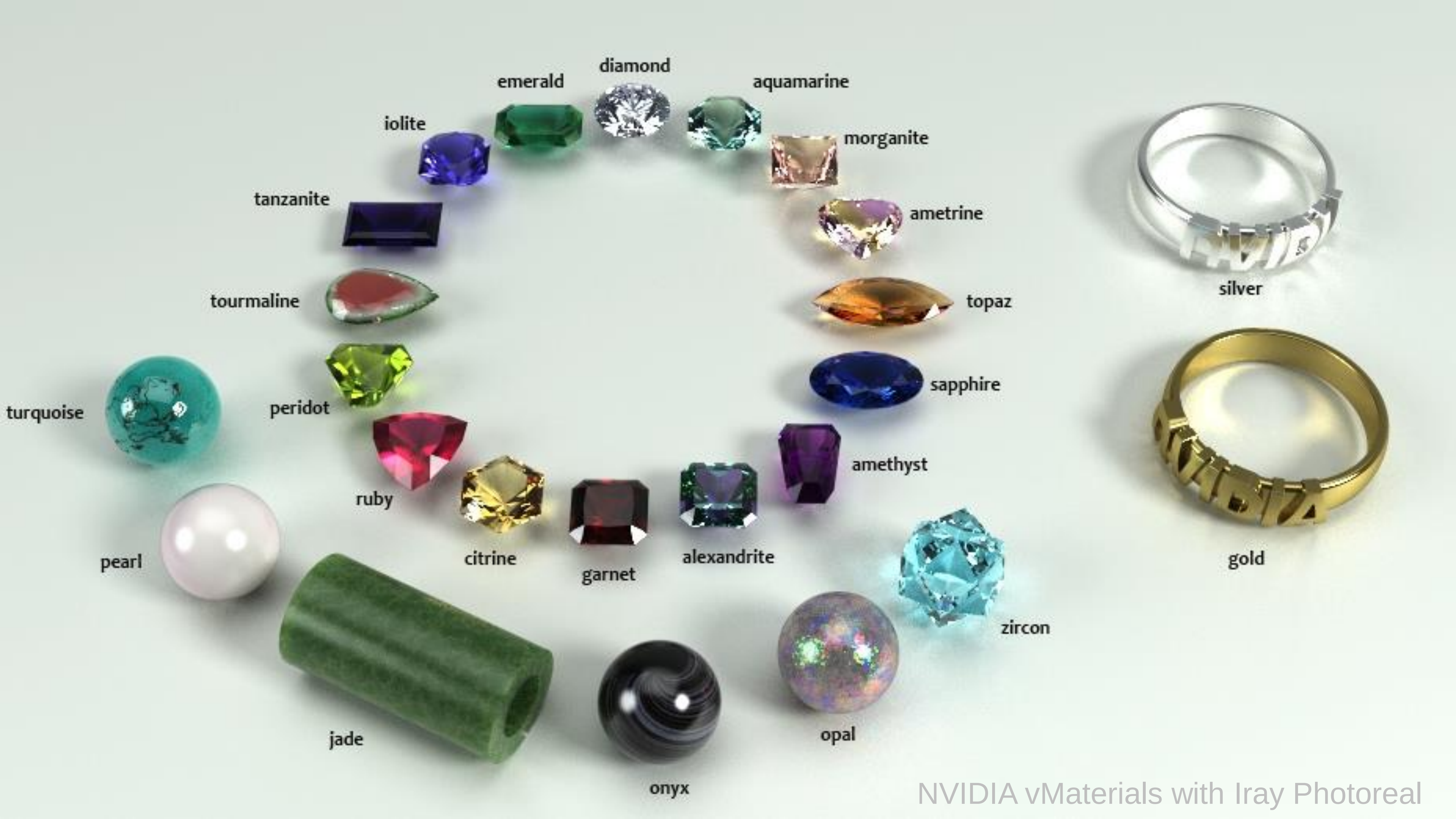
Iray Photoreal



Iray Photoreal

courtesy Harley Davidson

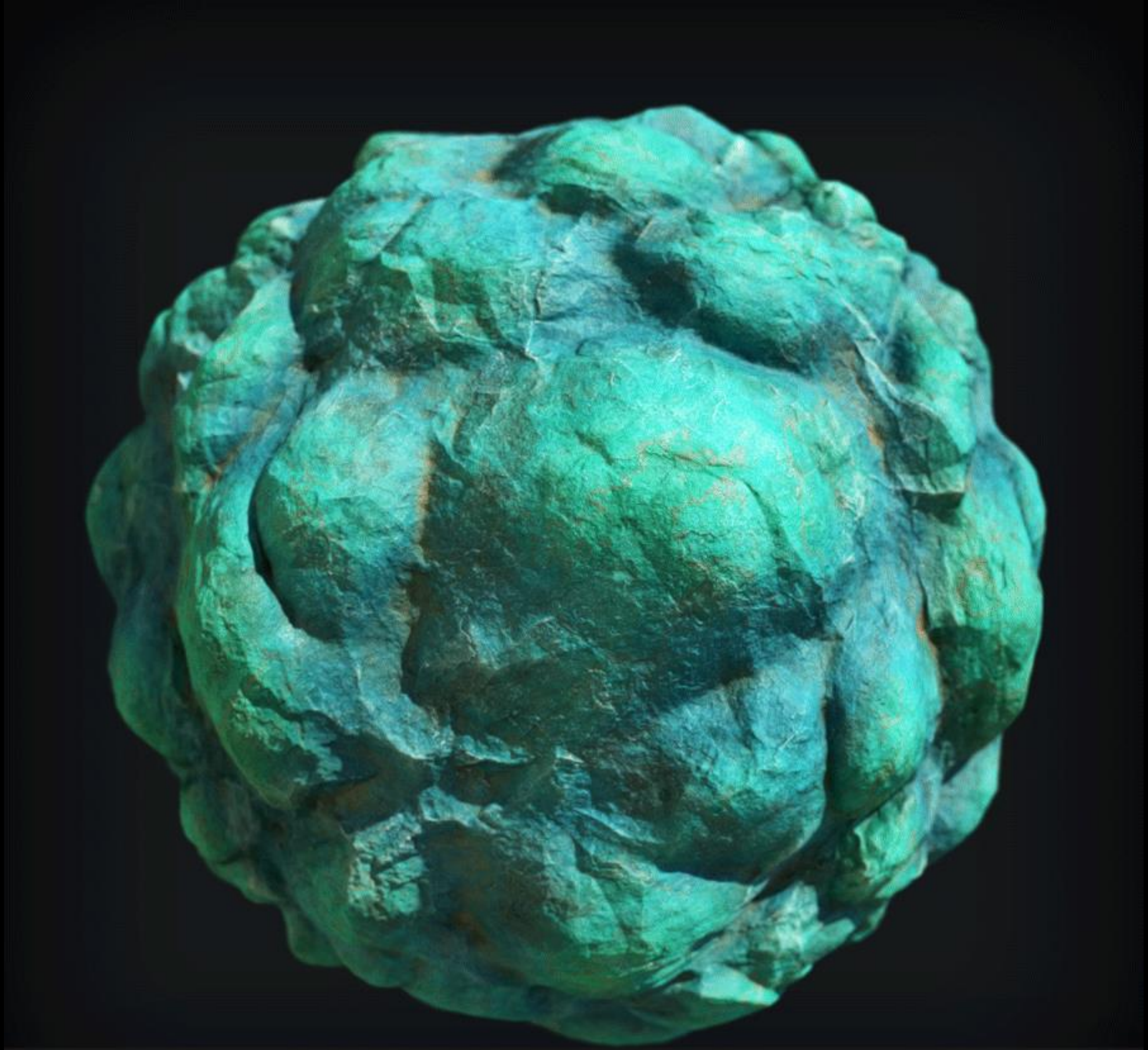
Photographer: Simon Puschmann/Rendering and Retouching: [zerone]



NVIDIA vMaterials with Iray Photoreal

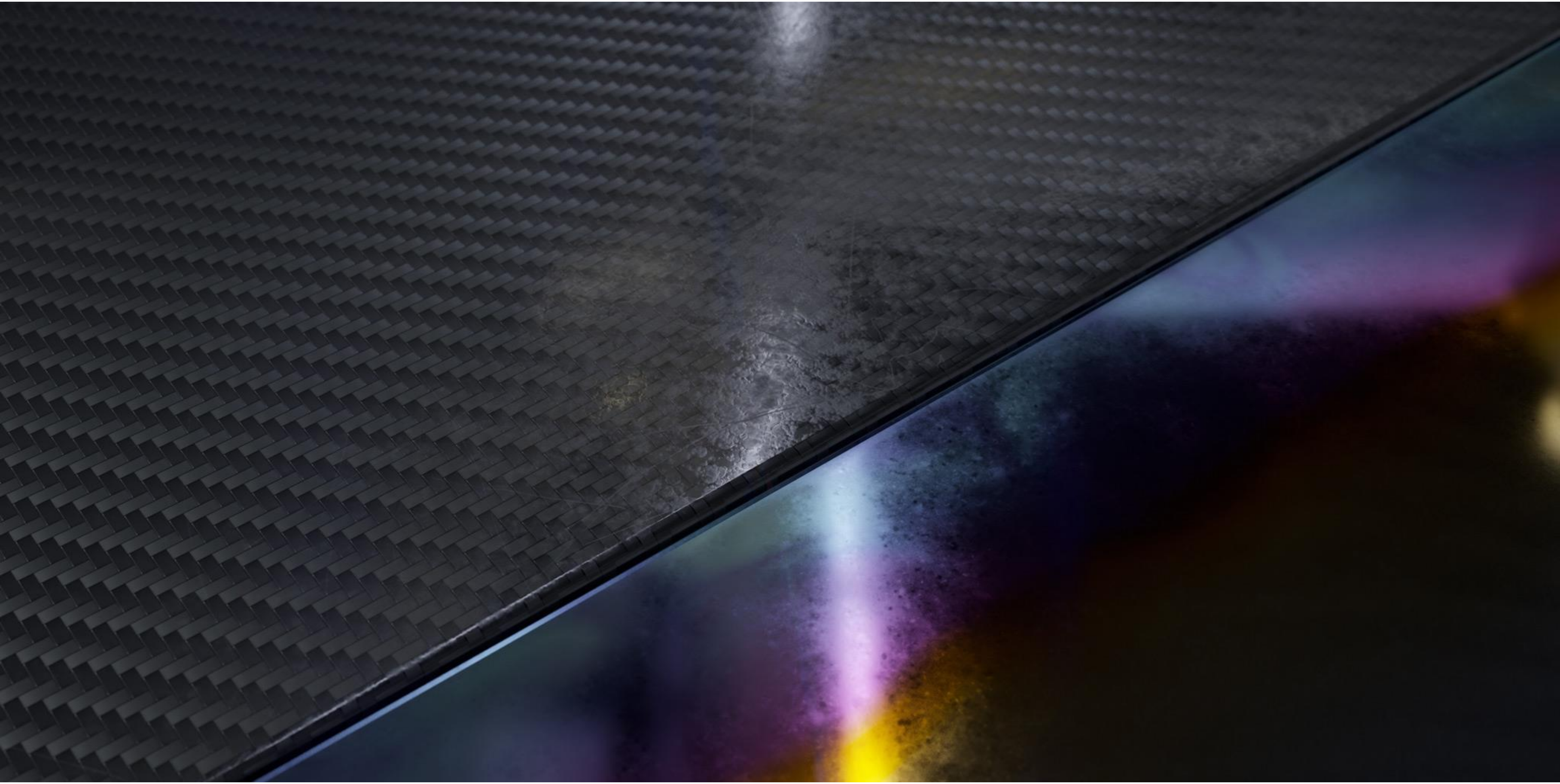
NVIDIA Iray

Iray Photoreal



Mark Foreman Malachite with Chrysocolla

SUBSTANCE DESIGNER

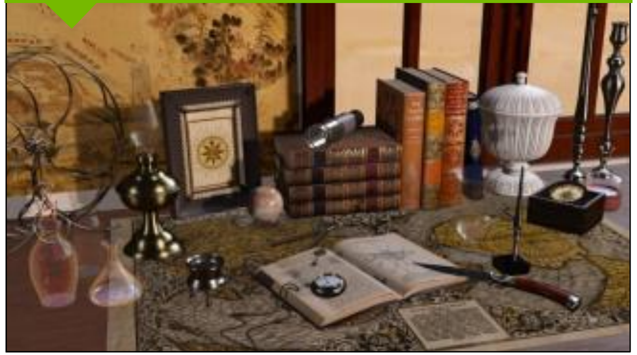


Substance Designer

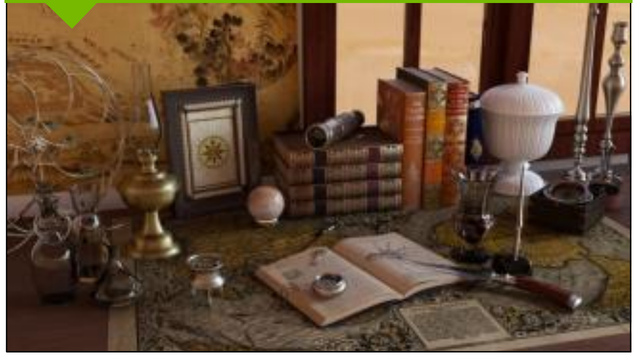
Matching the Appearance of a Single Material Within Different Rendering Techniques

One Scene for Different Renderers

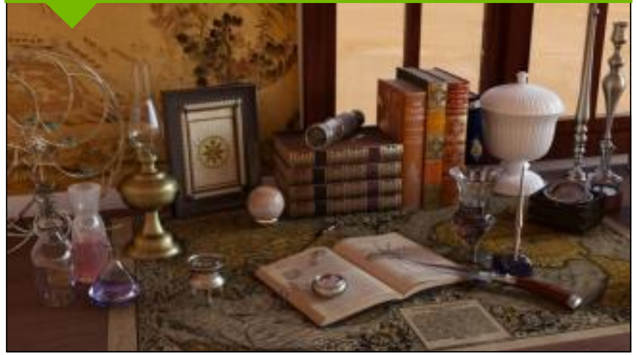
Realtime Rasterizer



Interactive Raytracer

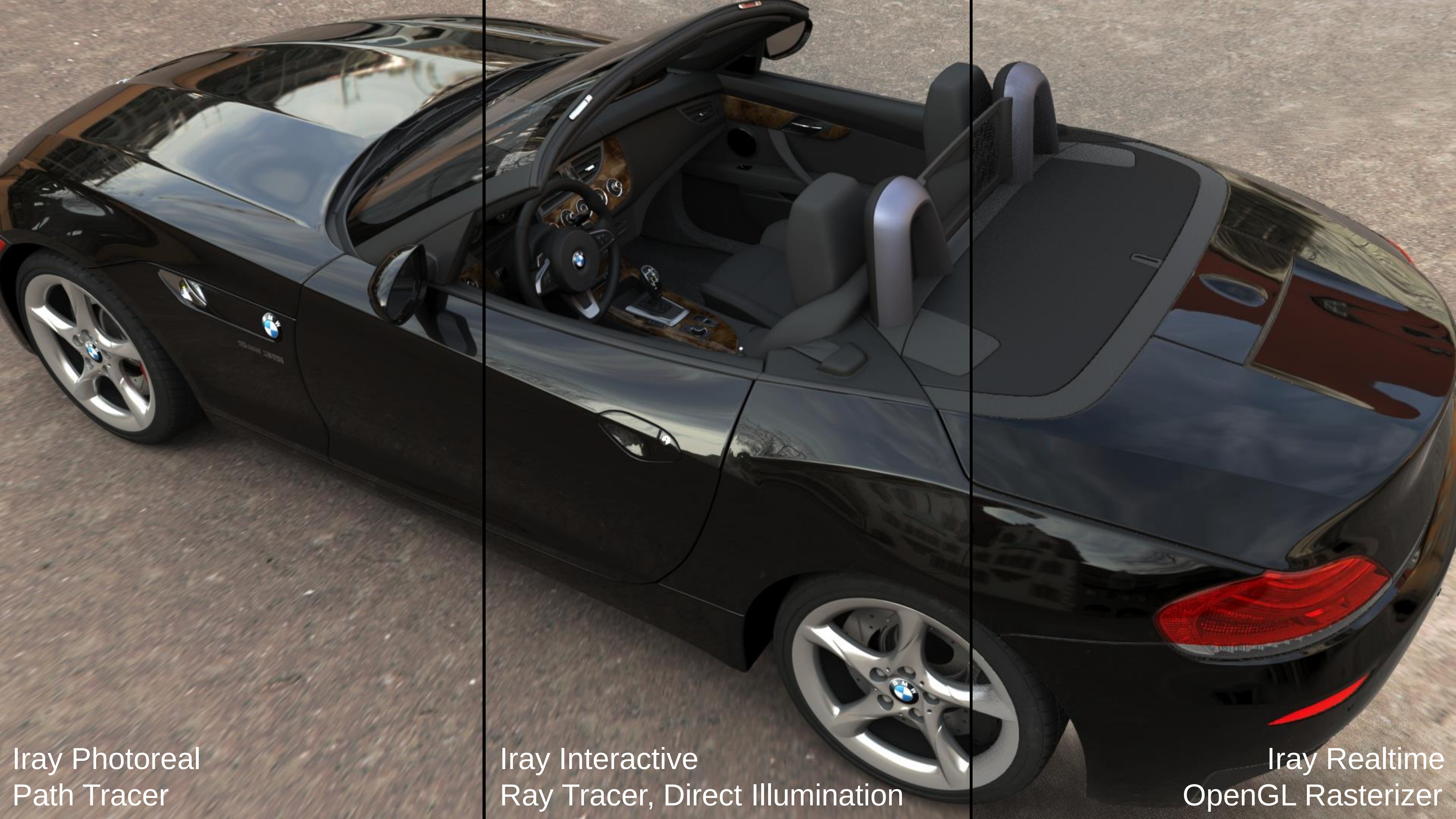


Pathtracer



Share scene and MDL materials for a consistent look

Switching renderers with no scene modifications

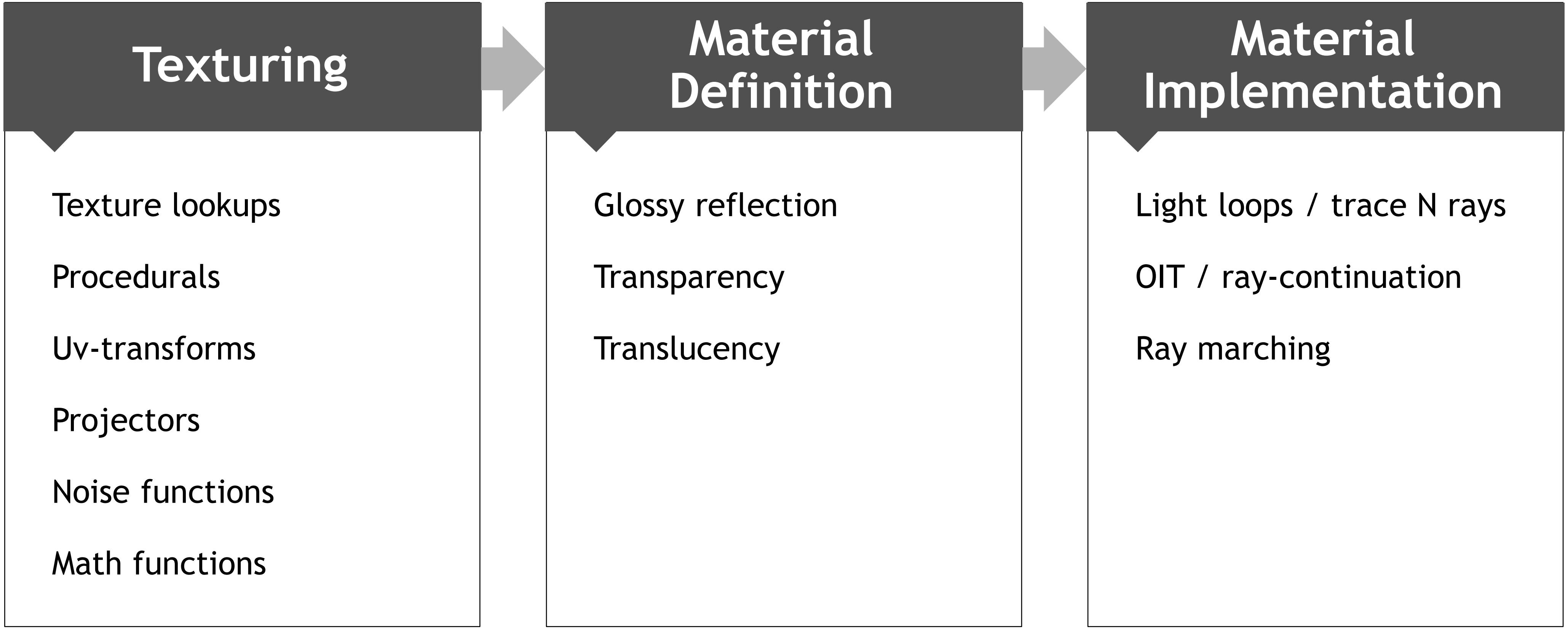


Iray Photoreal
Path Tracer

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

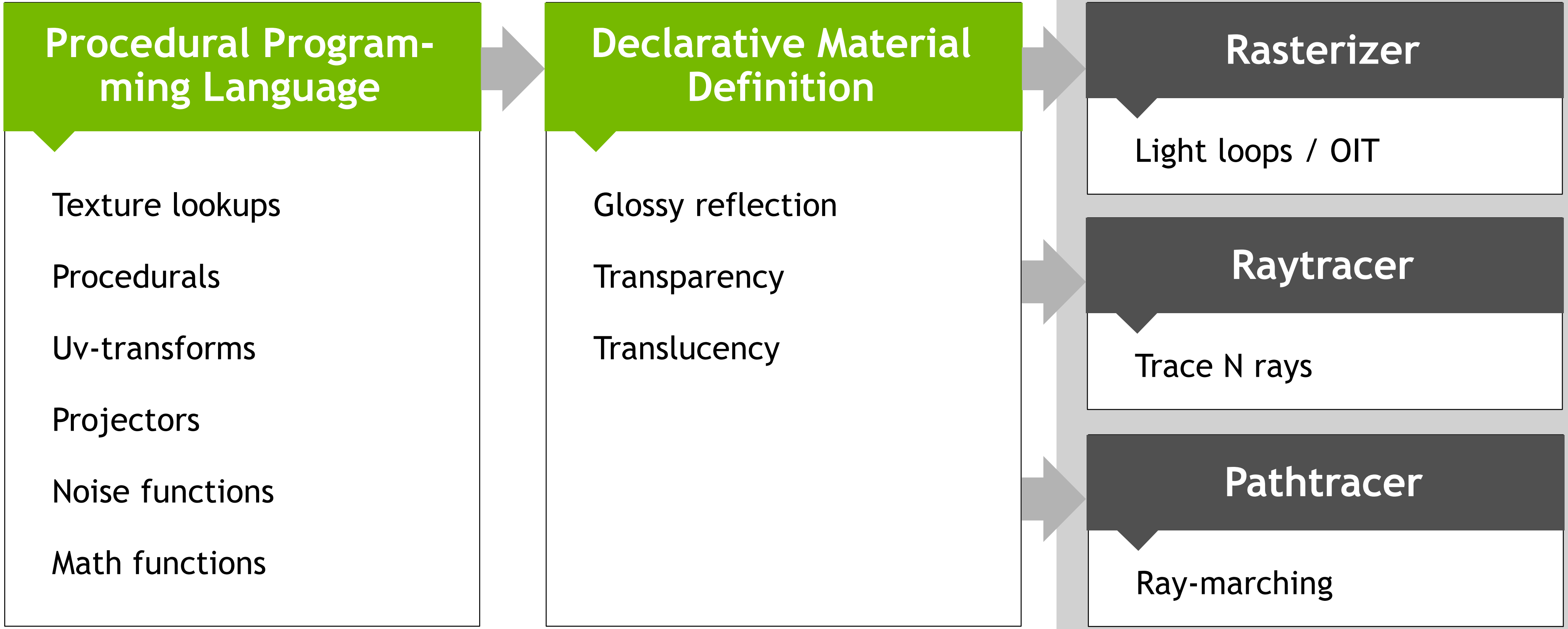
Light loops / trace N rays

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

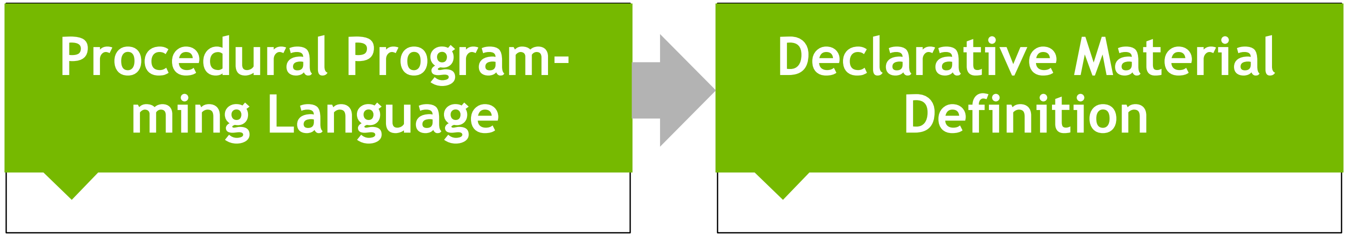
Raytracer

Trace N rays

Pathtracer

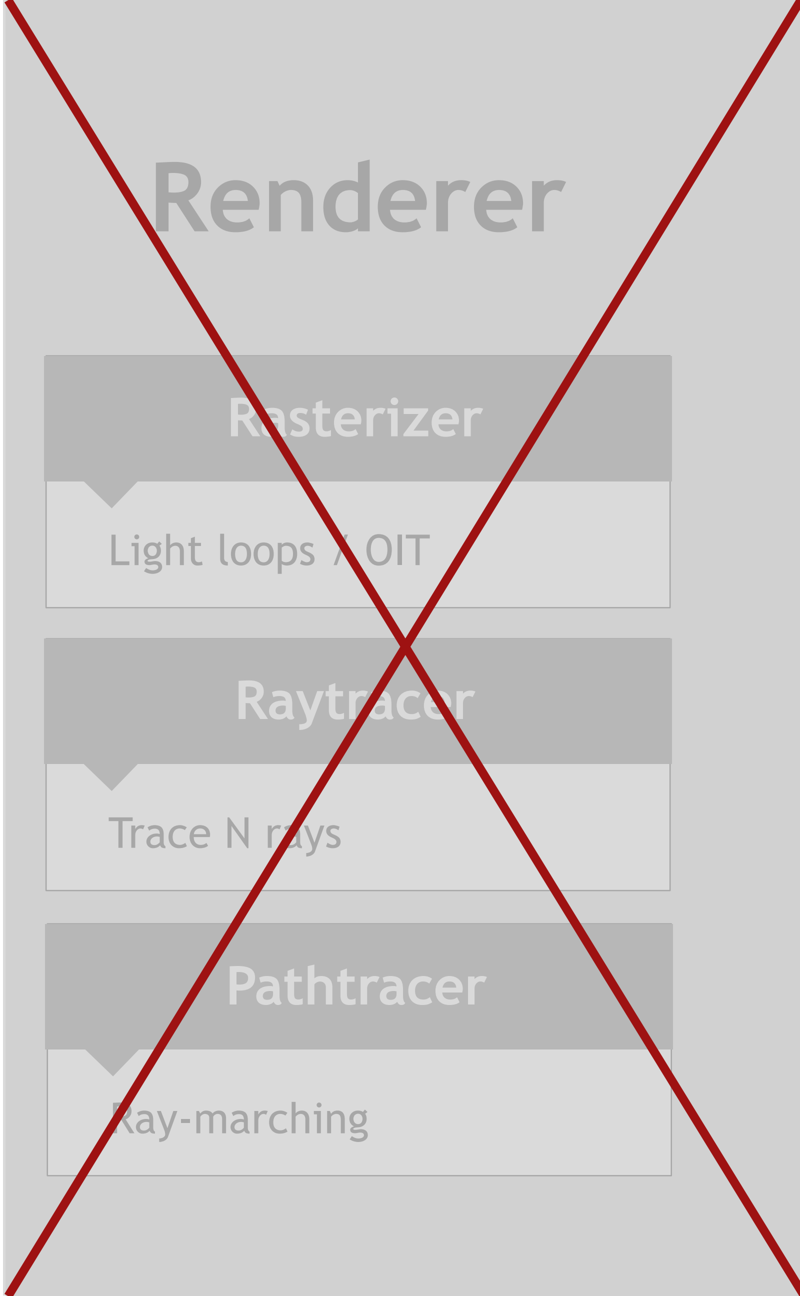
Ray-marching

NVIDIA MDL



Procedural Programming Language

Declarative Material Definition



~~Renderer~~

~~Rasterizer~~

~~Light loops / OIT~~

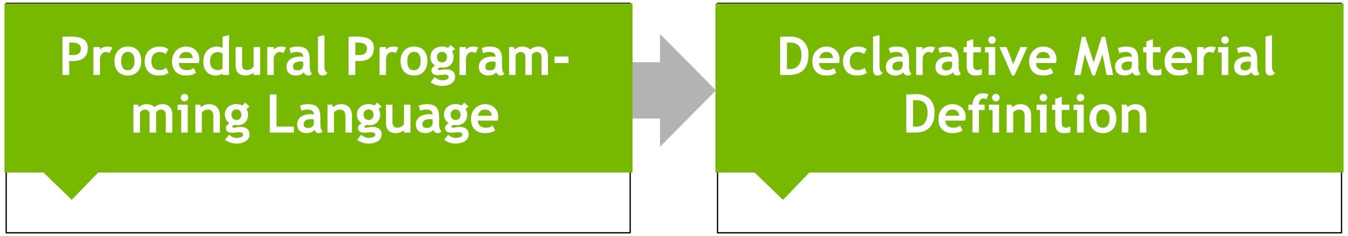
~~Raytracer~~

~~Trace N rays~~

~~Pathtracer~~

~~Ray-marching~~

NVIDIA® MDL

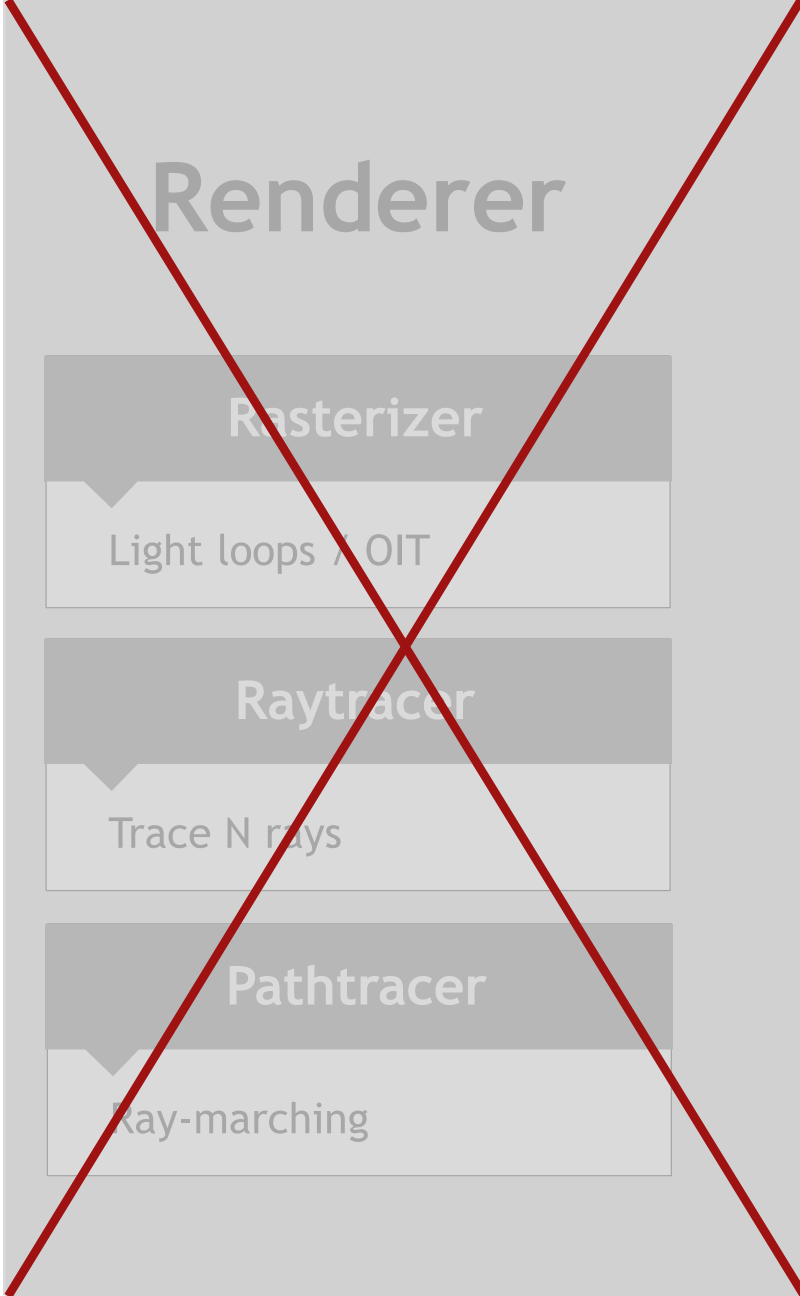


Procedural Programming Language → Declarative Material Definition

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



~~Renderer~~

~~Rasterizer~~

~~Light loops / OIT~~

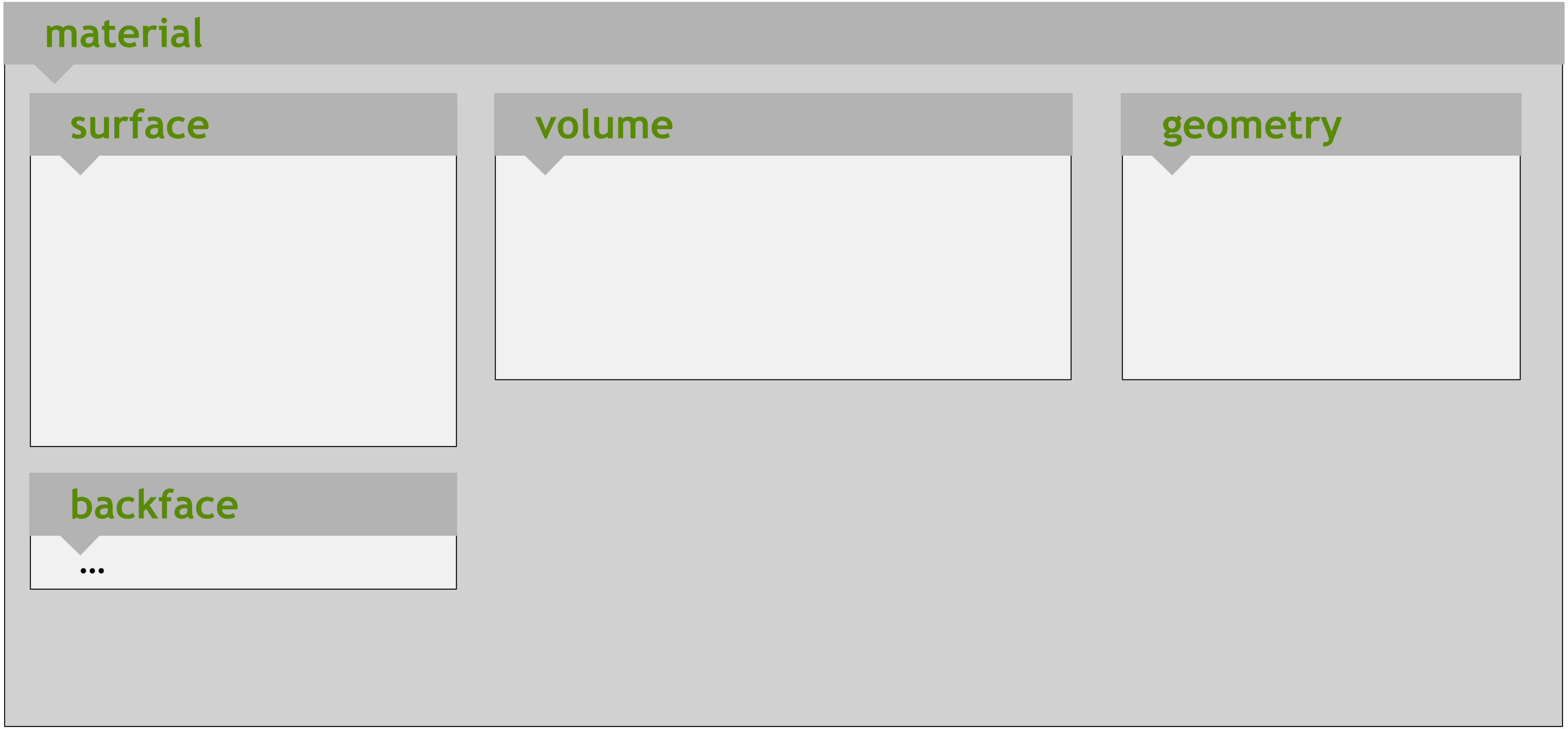
~~Raytracer~~

~~Trace N rays~~

~~Pathtracer~~

~~Ray-marching~~

MDL Material Model



material

surface

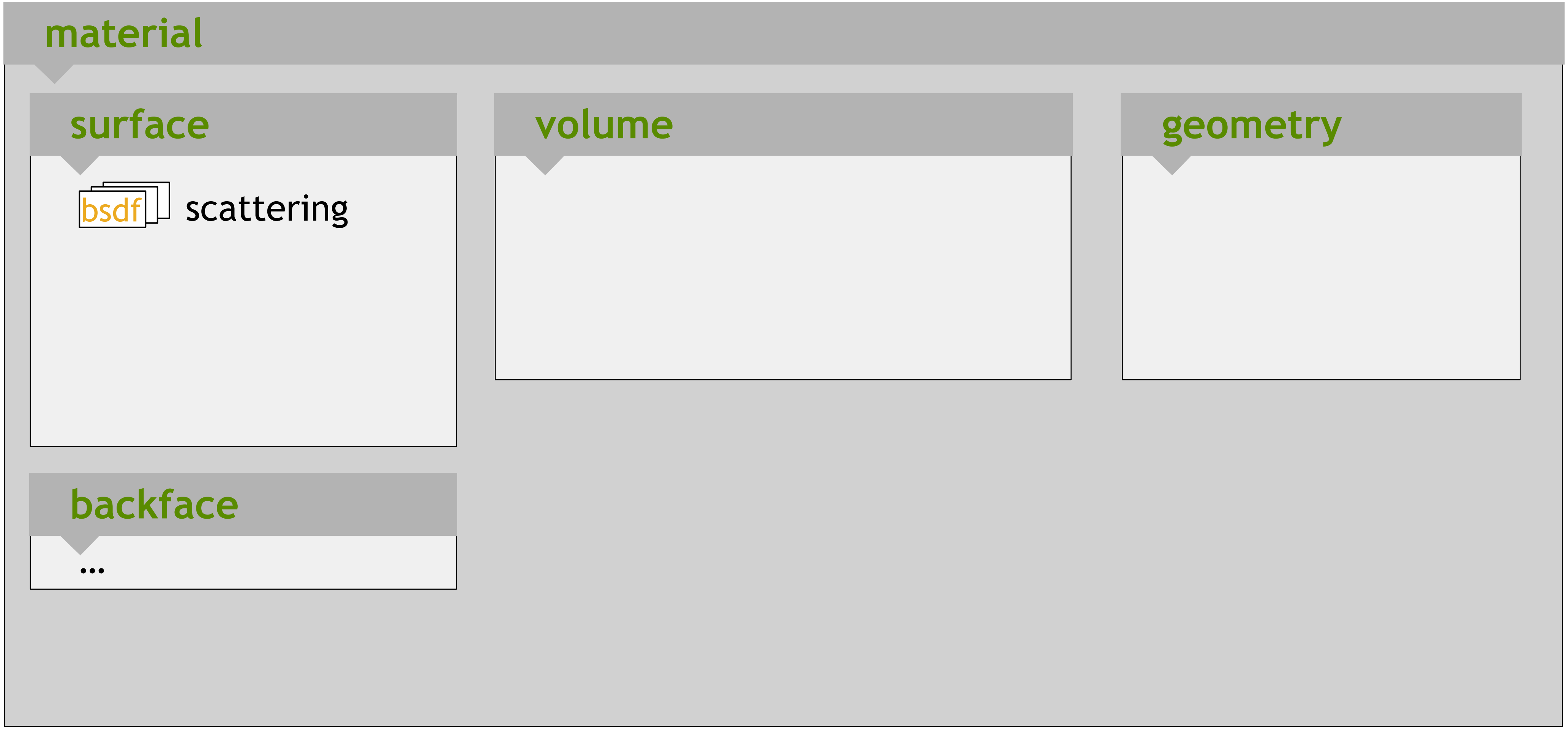
volume

geometry

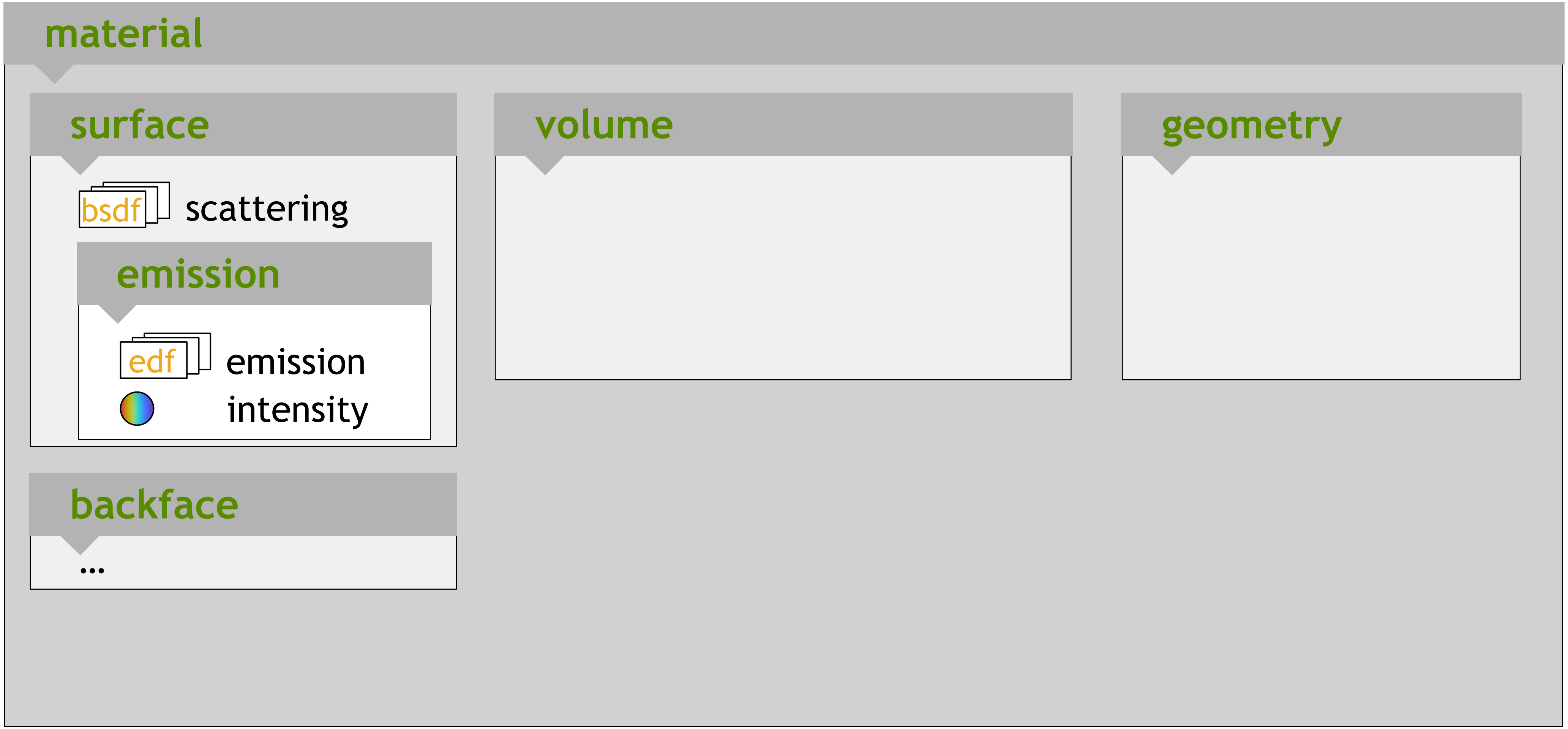
backface

...

MDL Material Model



MDL Material Model



material

surface

bsdf scattering

emission

edf emission

intensity

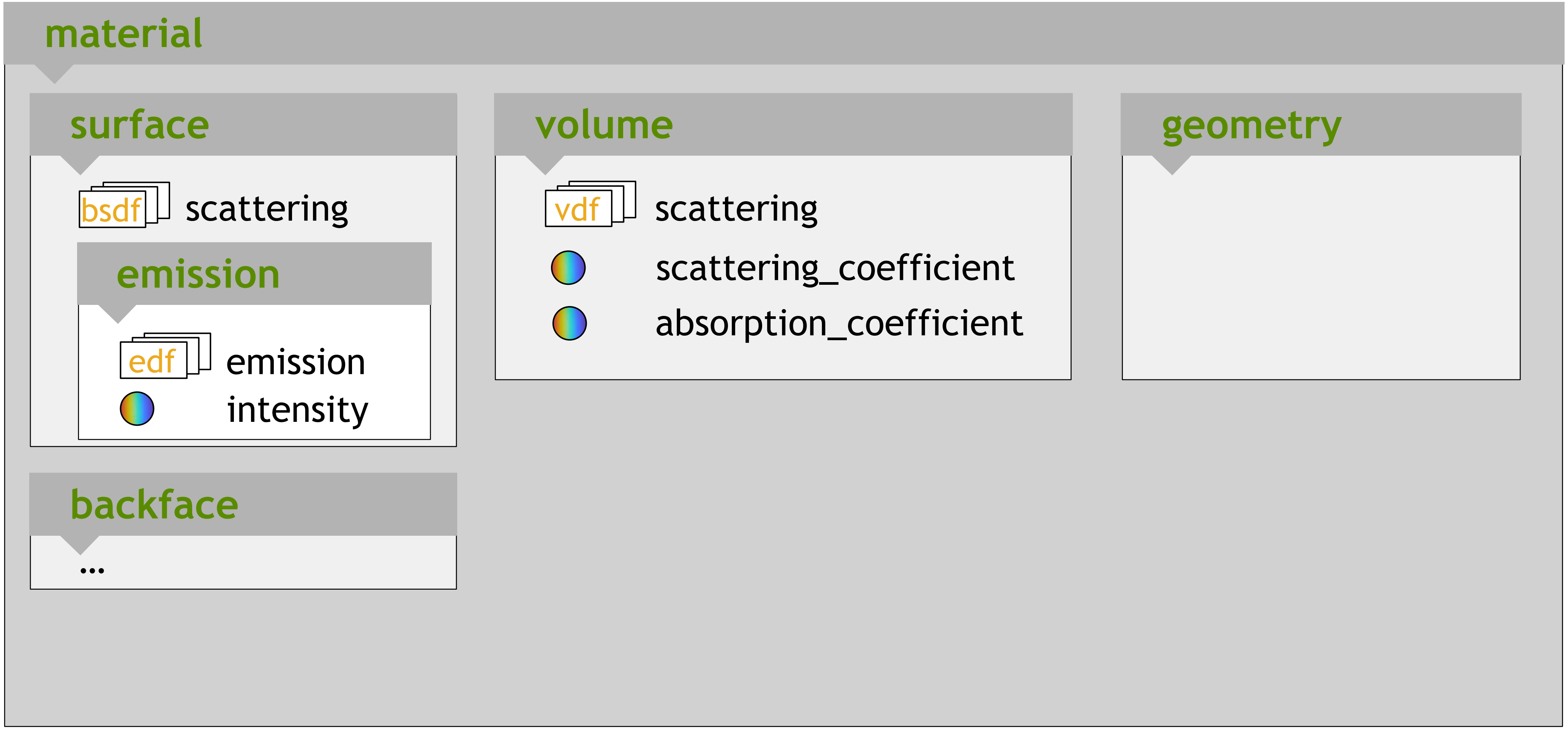
backface

...

volume

geometry

MDL Material Model



material

surface

bsdf scattering

emission

edf emission

● intensity

backface

...

volume

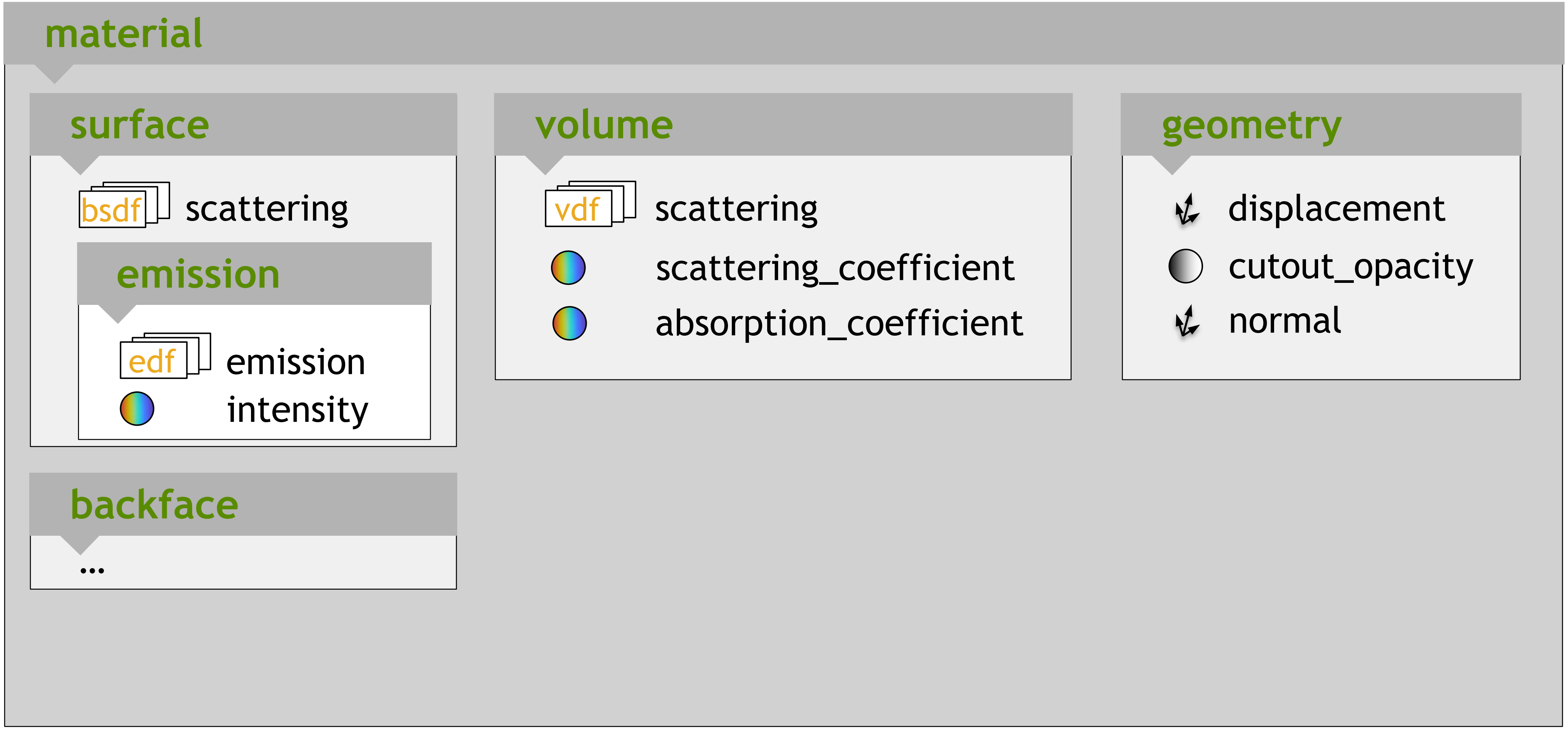
vdf scattering

● scattering_coefficient

● absorption_coefficient

geometry

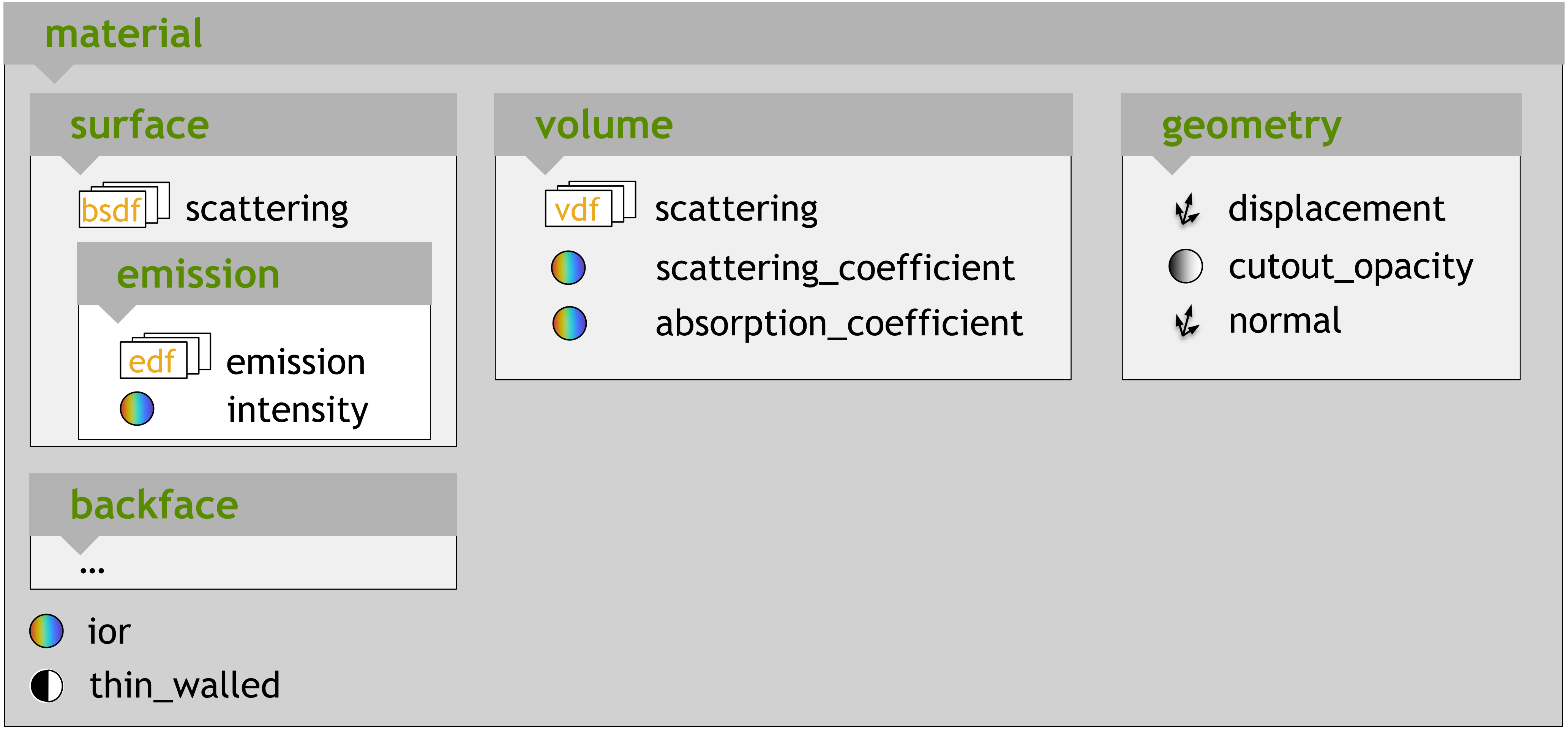
MDL Material Model



material

- surface
  - bsdf scattering
  - emission
    - edf emission
    - intensity
- backface
  - ...
- volume
  - vdf scattering
  - scattering_coefficient
  - absorption_coefficient
- geometry
  - displacement
  - cutout_opacity
  - normal

MDL Material Model



material

surface

bsdf scattering

emission

edf emission

intensity

backface

...

ior

thin_walled

volume

vdf scattering

scattering_coefficient

absorption_coefficient

geometry

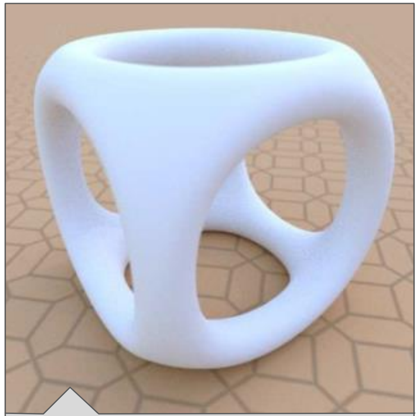
displacement

cutout_opacity

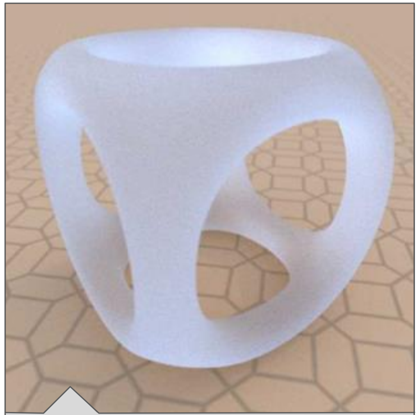
normal

MDL Elemental Distribution Functions

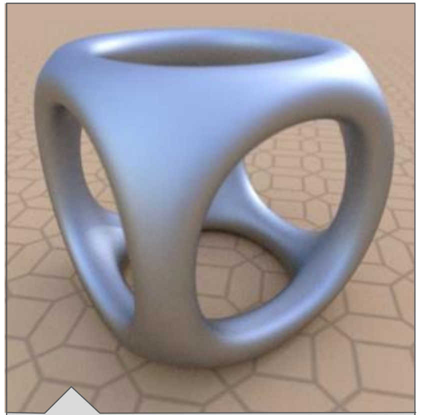
Bidirectional
Scattering
Distribution
Functions



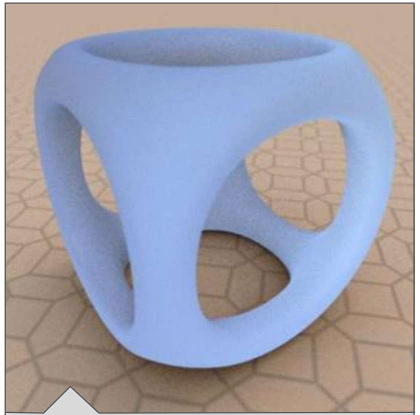
Diffuse Reflection



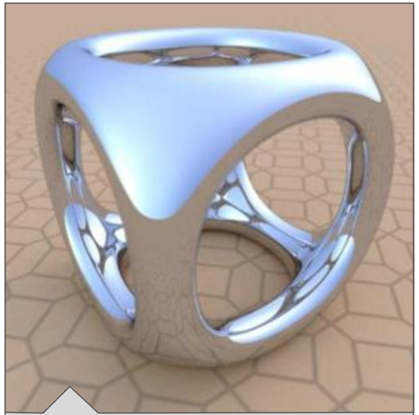
Diffuse Transmission



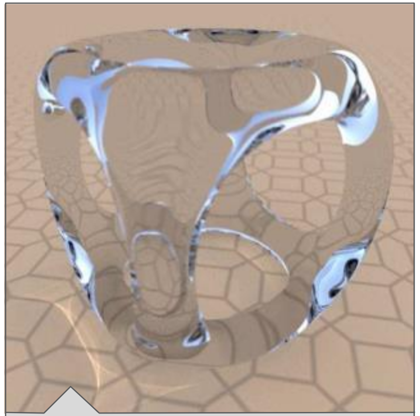
Glossy (various)



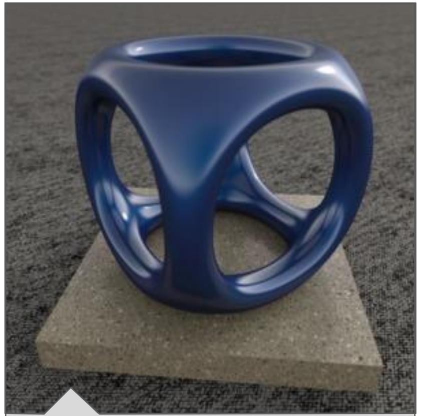
Backscatter Glossy



Specular Reflection



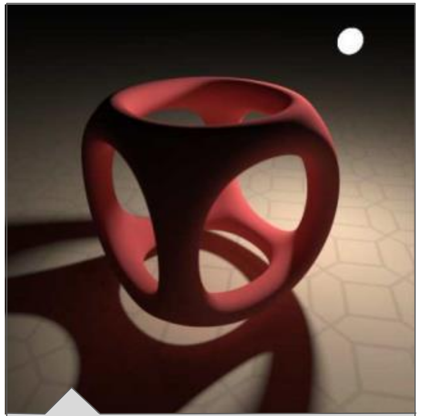
Spec. Refl.+Transm.



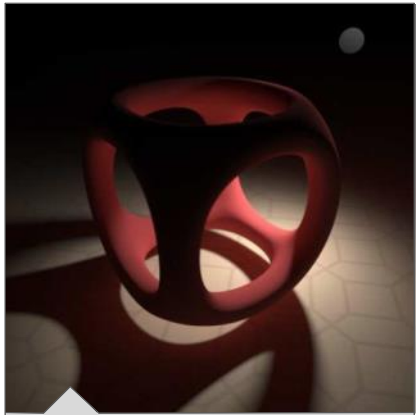
Measured BSDF

MDL Elemental Distribution Functions

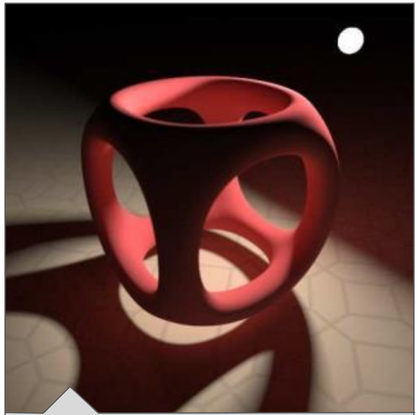
Emissive
Distribution
Functions



Diffuse

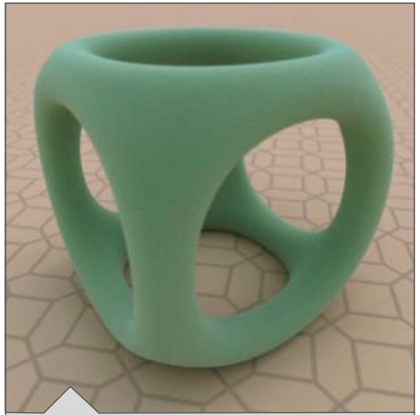


Spot



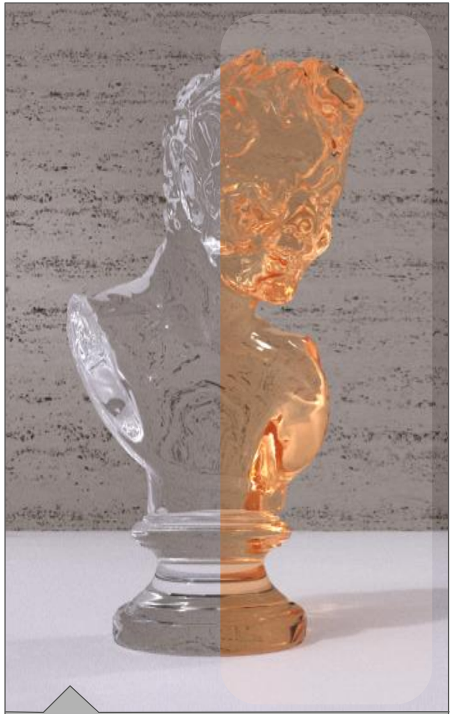
IES Profile

Volume
Distribution
Functions

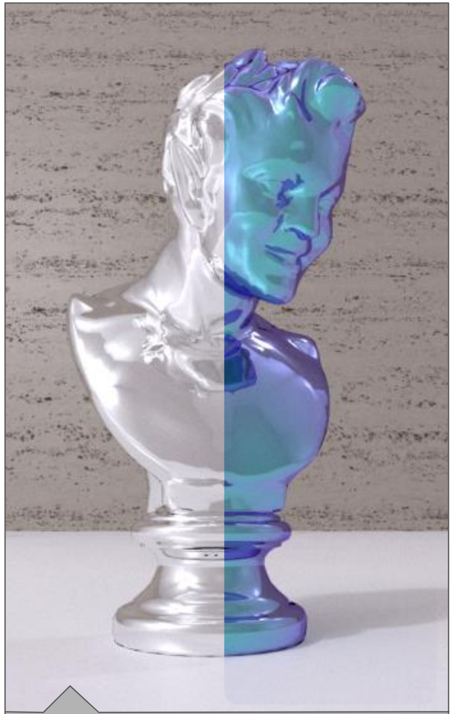


Henyey-Greenstein

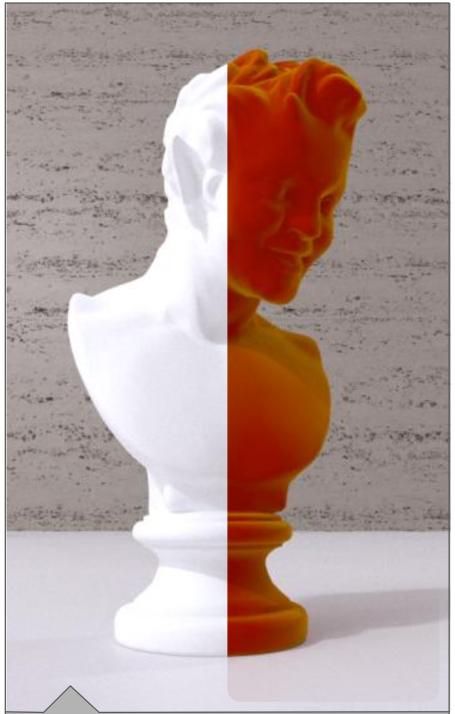
MDL Distribution Function Modifiers



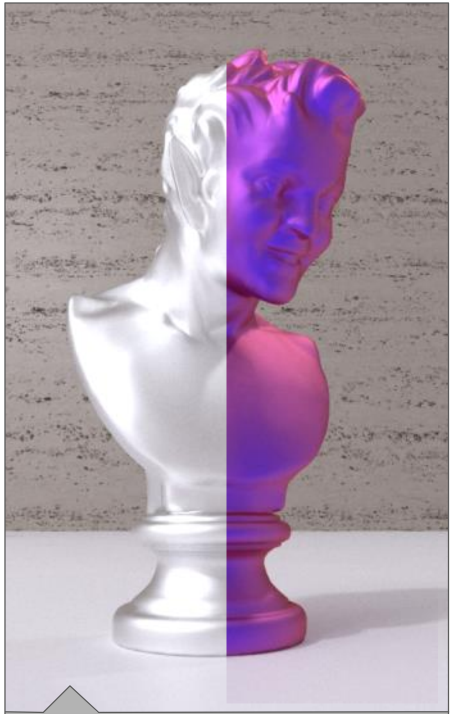
Tint



Thin Film

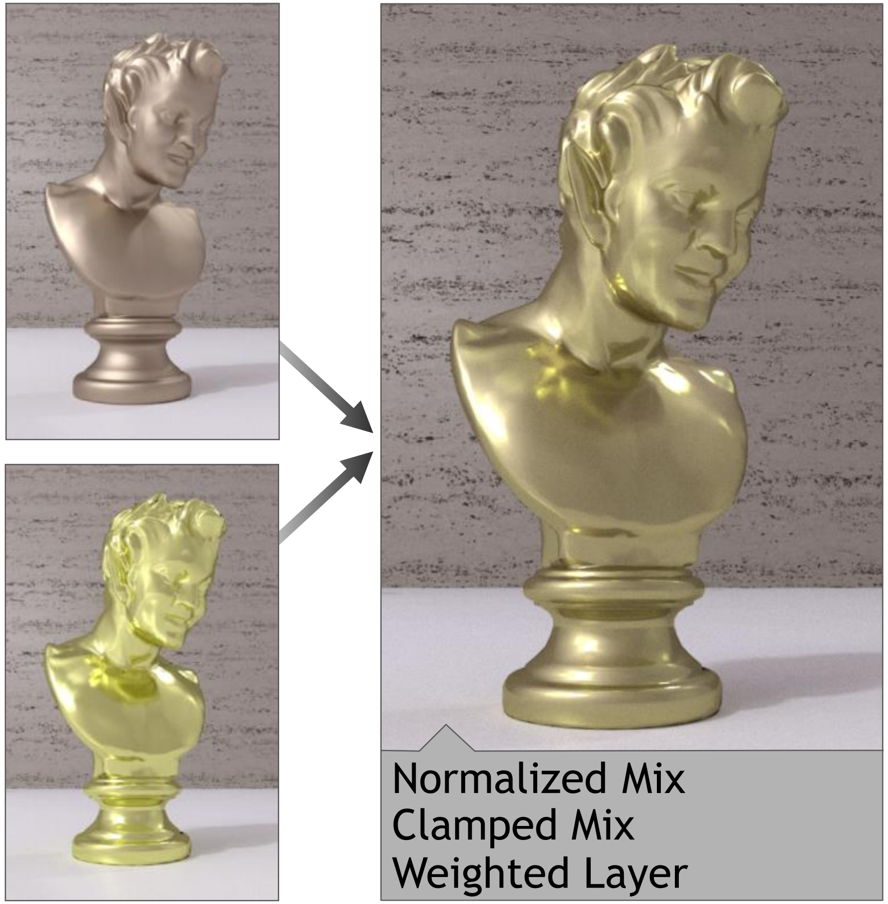


Directional Factor

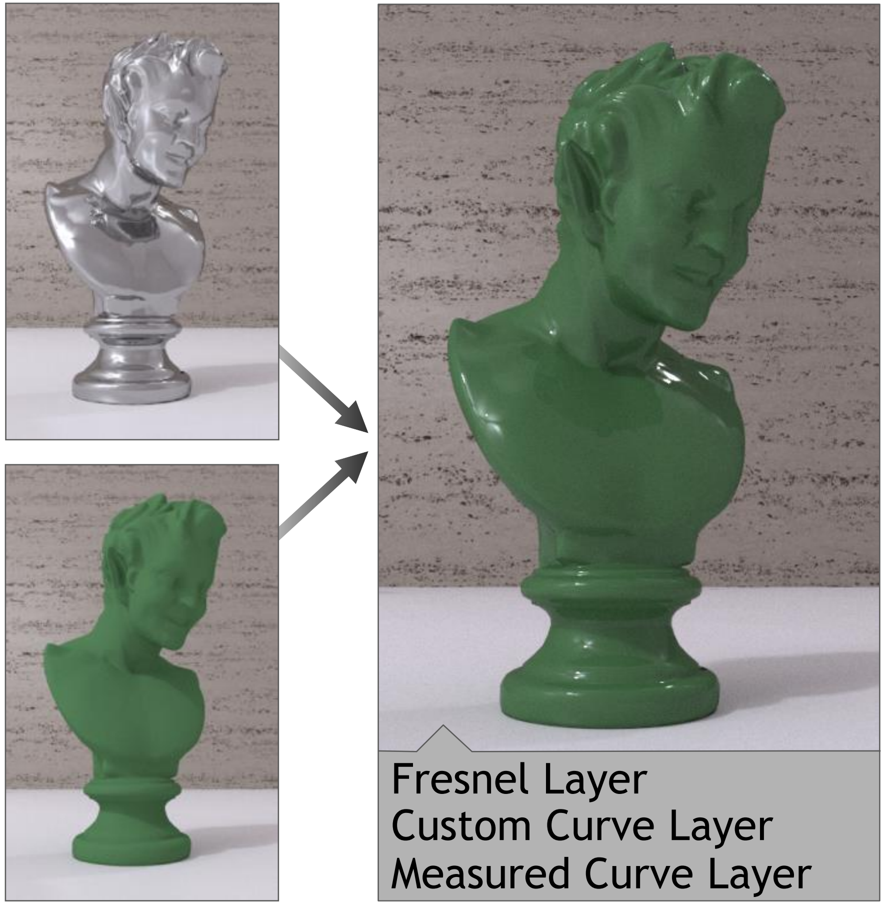


Measured Curve Factor

MDL Distribution Functions Combiners

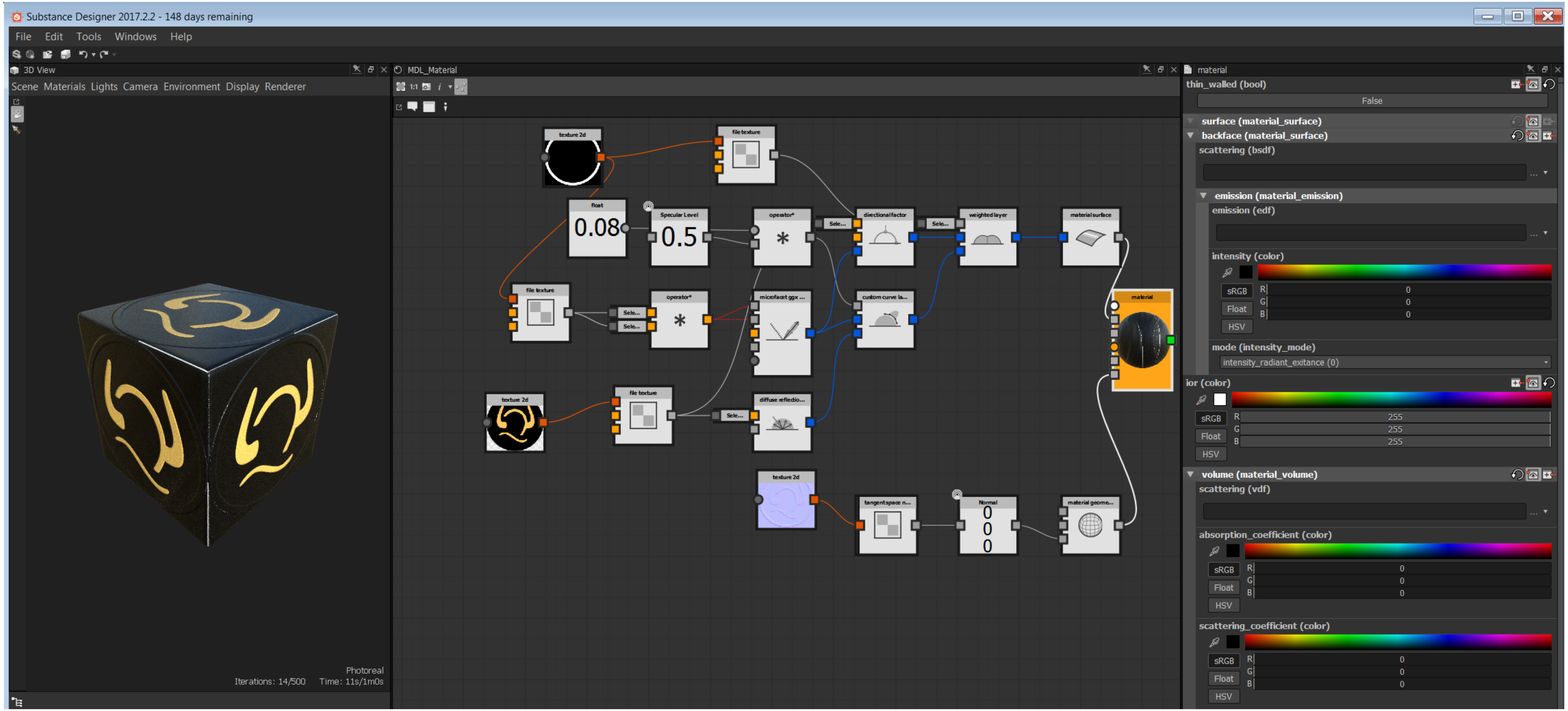


Normalized Mix
Clamped Mix
Weighted Layer



Fresnel Layer
Custom Curve Layer
Measured Curve Layer

MDL Layered Material Example



thin_walled (bool)
False

surface (material_surface)

backface (material_surface)

scattering (bsdf)

emission (material_emission)

emission (edf)

intensity (color)

sRGB R 0
Float G 0
HSV B 0

mode (intensity_mode)
intensity_radiant_exitance (0)

ior (color)

sRGB R 255
Float G 255
HSV B 255

volume (material_volume)

scattering (vdf)

absorption_coefficient (color)

sRGB R 0
Float G 0
HSV B 0

scattering_coefficient (color)

sRGB R 0
Float G 0
HSV B 0

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

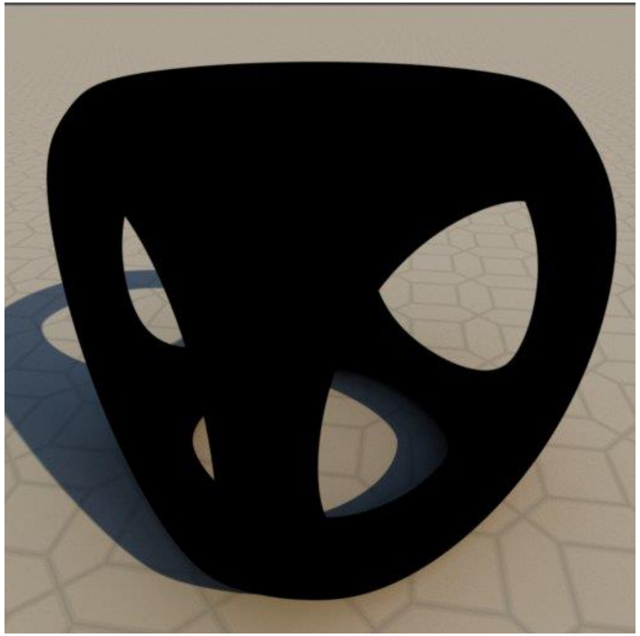
Material struct is already fully defined

```
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 );
```

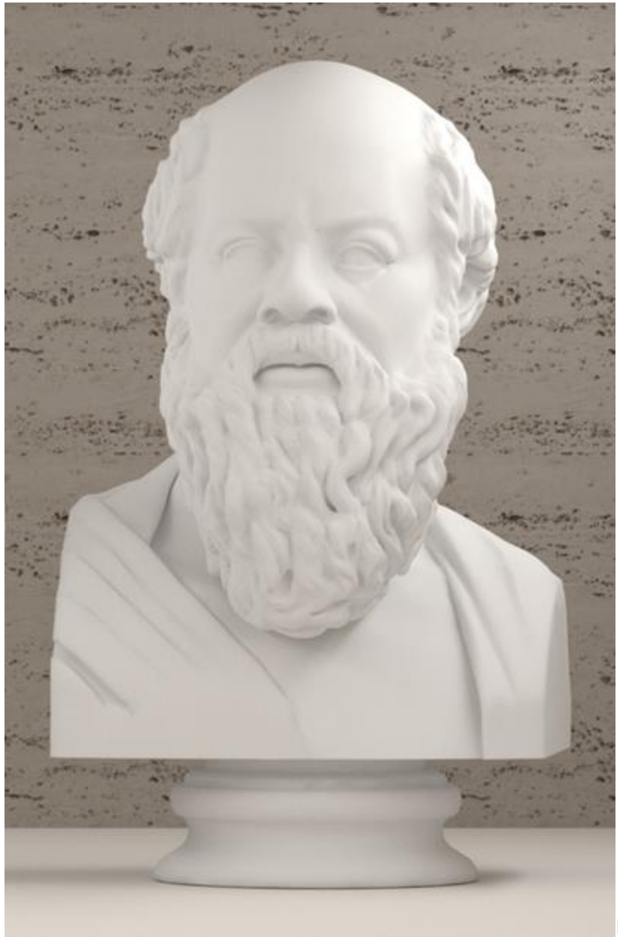
Defining a Material Using MDL

```
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

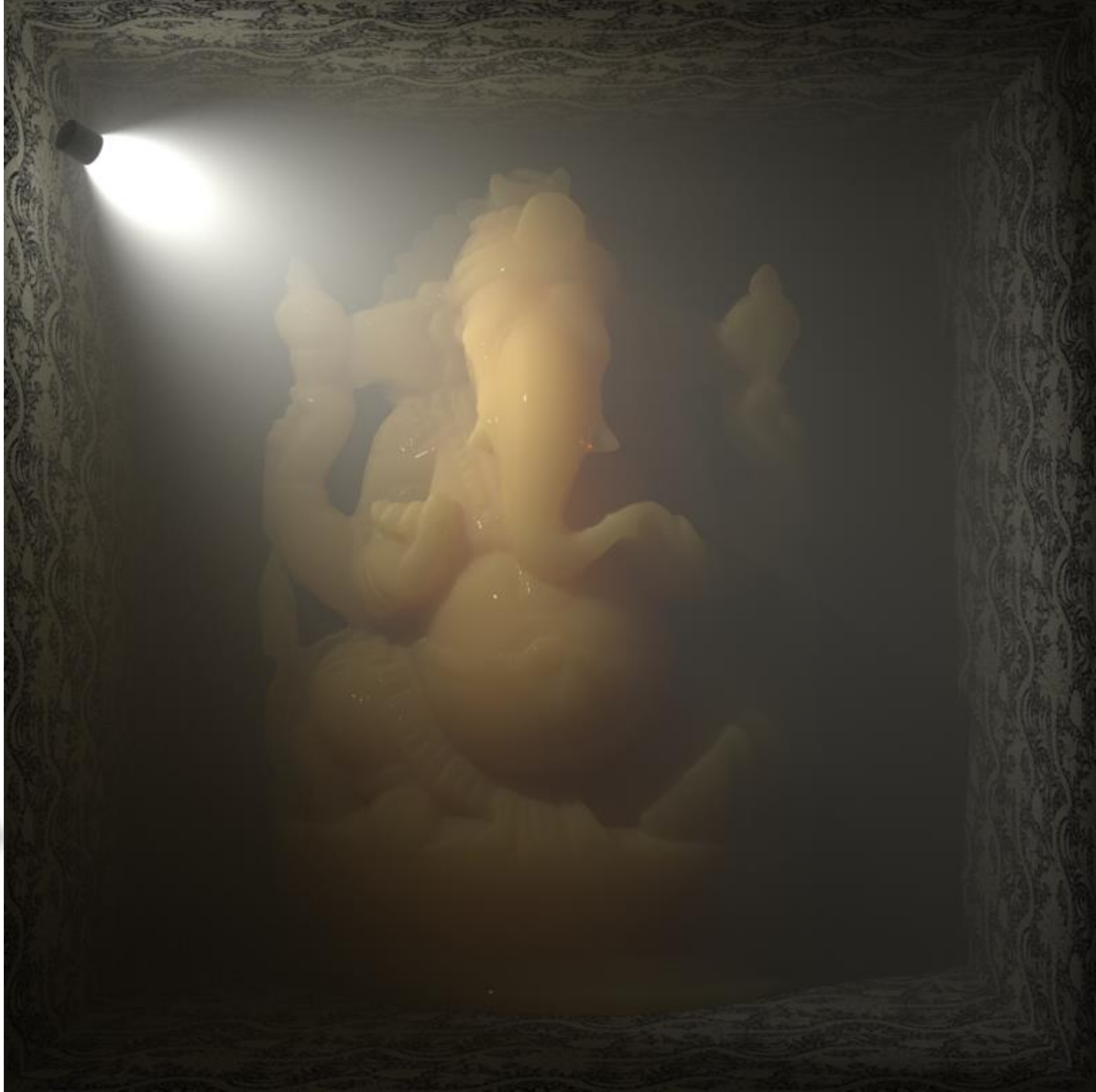
```
material plastic(
     color diffuse_color = color(.15,0.4,0.0),
     float roughness = 0.05
) = material(
     surface: material_surface(
          scattering: df::fresnel_layer (
               ior: color(1.5),
               layer: df::simple_glossy_bsdf (
                    roughness_u: glossy_roughness
               ),
               base: df::diffuse_reflection_bsdf (
                    tint: diffuse_color )
          )
     )
);
```

MDL Handbook

www.mdlhandbook.com

January 10th update: more on procedural texturing and displacement

Upcoming: advanced volumes

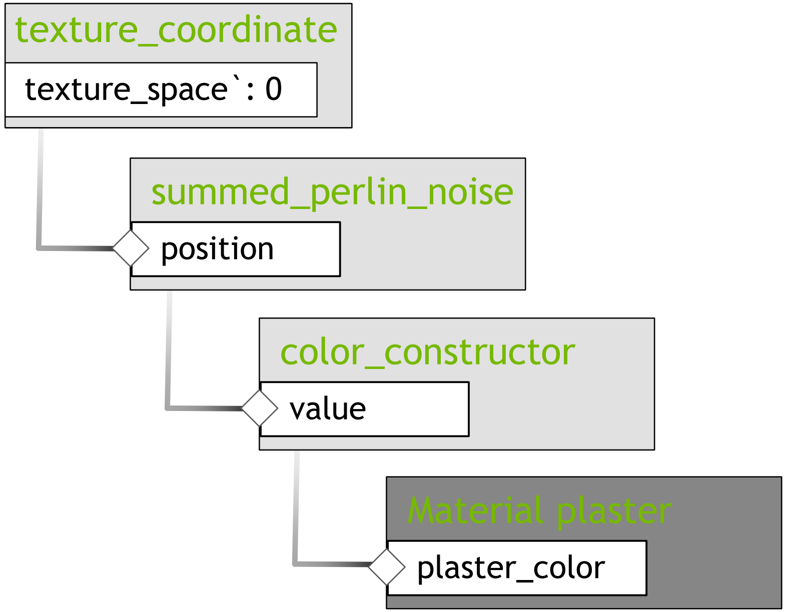


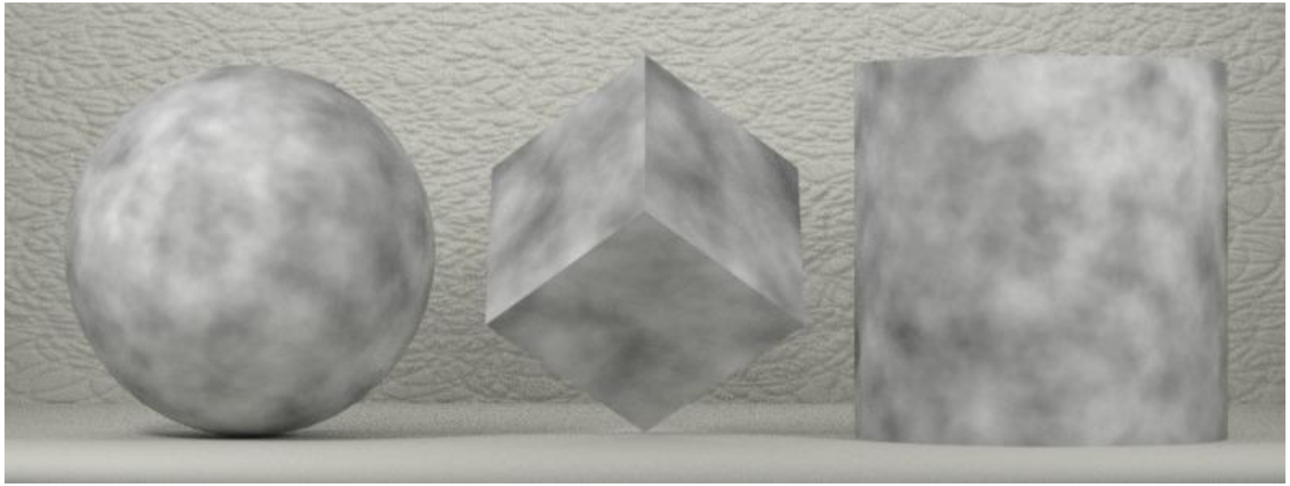
MDL Procedural Programming Language

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

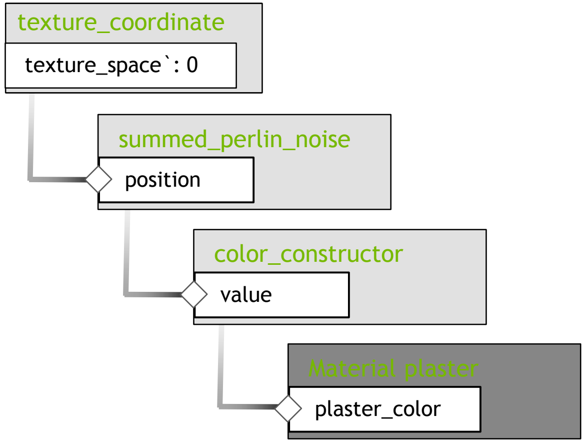
```
float summed_perlin_noise (
     float3 point,
     int level_count=4,
     float level_scale=0.5,
     float point_scale=2.0,
     bool turbulence=false)
{
     float scale = 0.5, noise_sum = 0.0;
     float3 level_point = point;
     for (int i = 0; i < level_count; i++)
     {
          float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

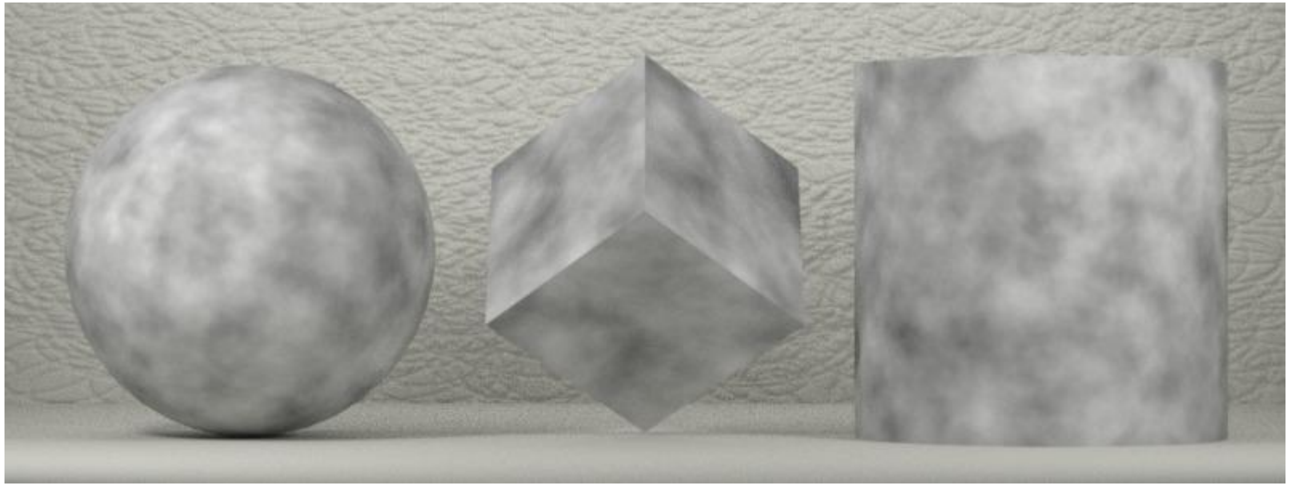
← MDL Handbook

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 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

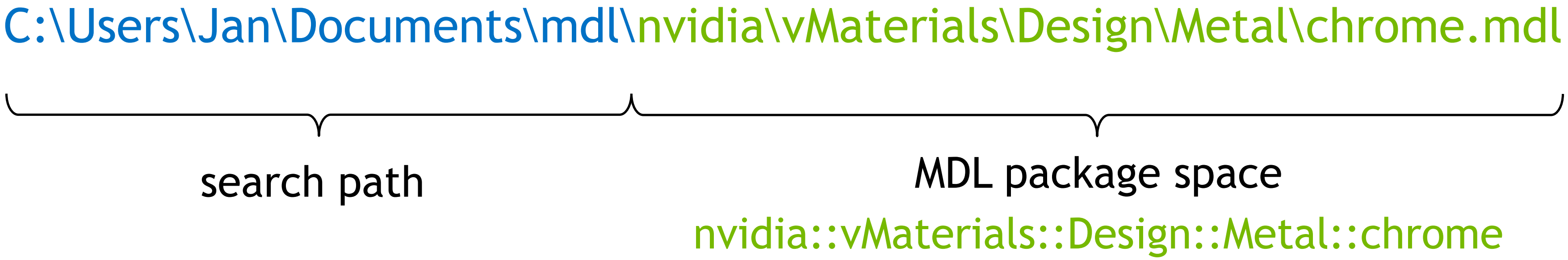
MDL Module System

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

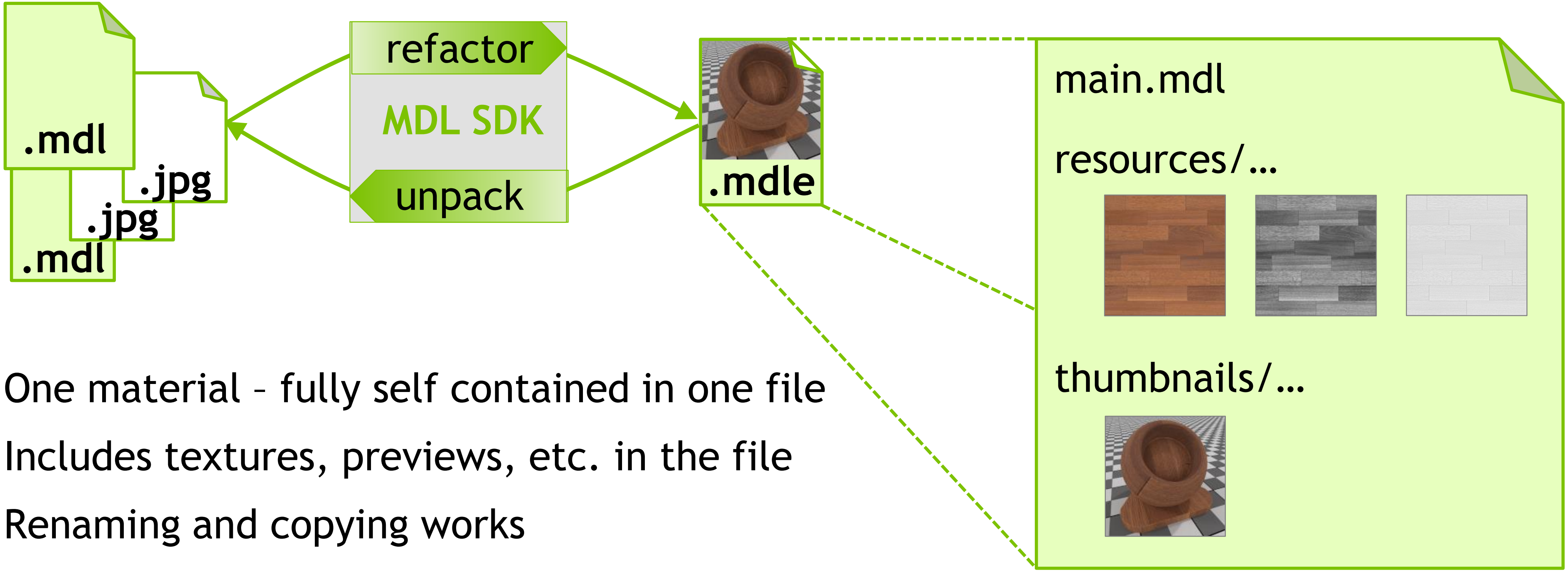
search path

MDL package space

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

MDL 1.5 Preview

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 Preview

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

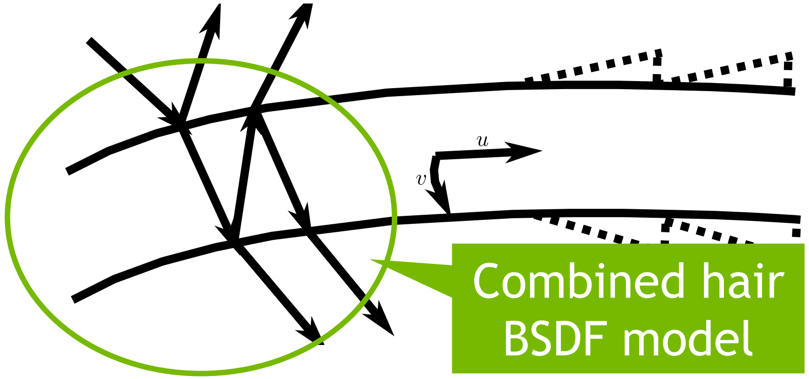
C:\Users\%USERNAME%\Documents\mdl\	MDL search path
nvidia\vMaterials\fr.xlf	French vMaterial package XLIFF file
nvidia\vMaterials\AEC\Glass\Mirror_fr.xlf	French Mirror module XLIFF file

MDL 1.5 Preview

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;
};
```

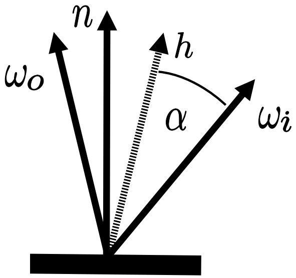


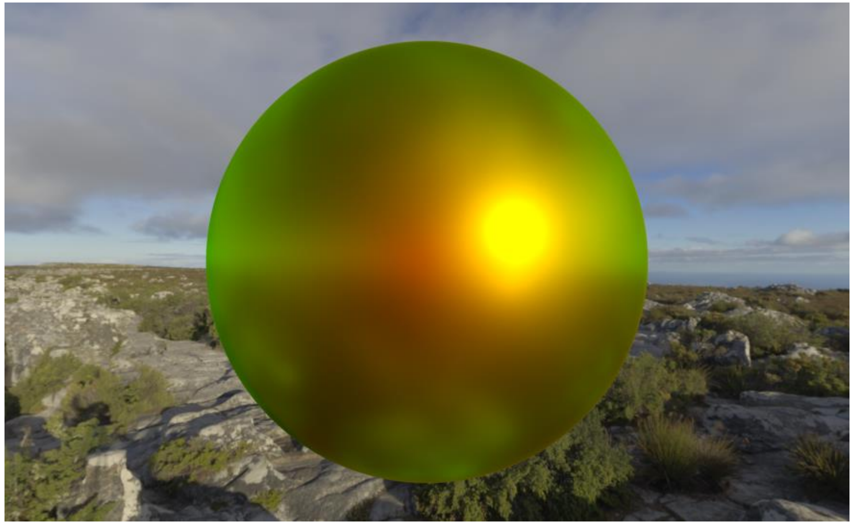
Combined hair BSDF model

MDL 1.5 Preview

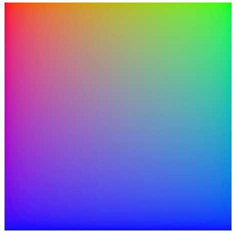
Microfacet coloring to support flip-flop car paints and more

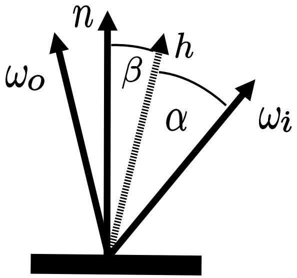
1D measured curve (MDL >=1.4)

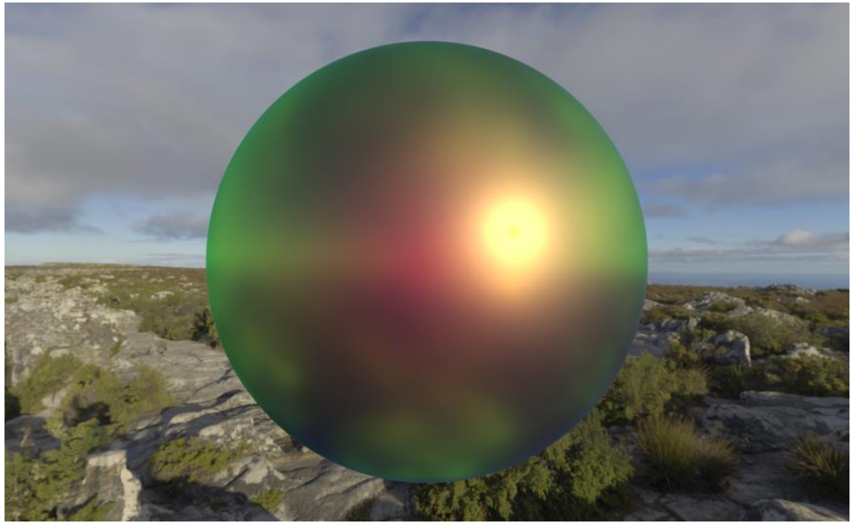




2D measured curve (new in MDL 1.5)

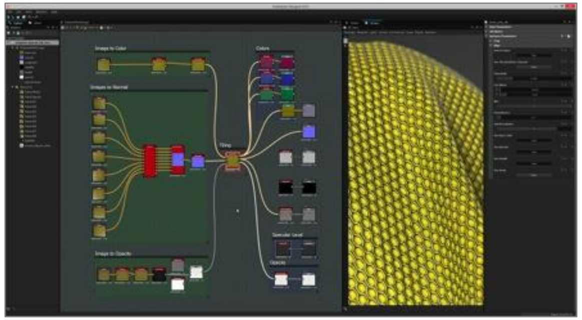






Additional MDL Benefits

Measured Materials

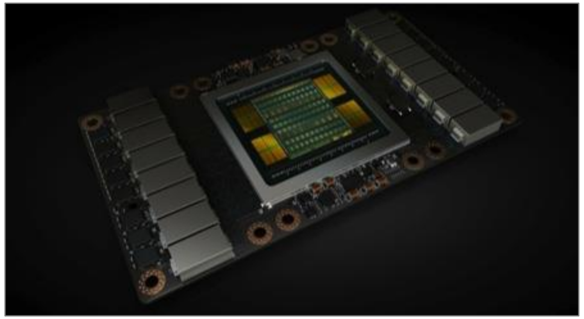


Spatially Varying BRDF

AxF from X-Rite

Measure Isotropic BSDF

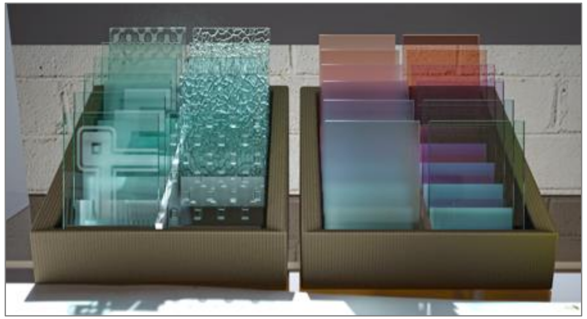
Designed for Parallelism



Little data dependencies

Side-effect free functions

Material Catalogs

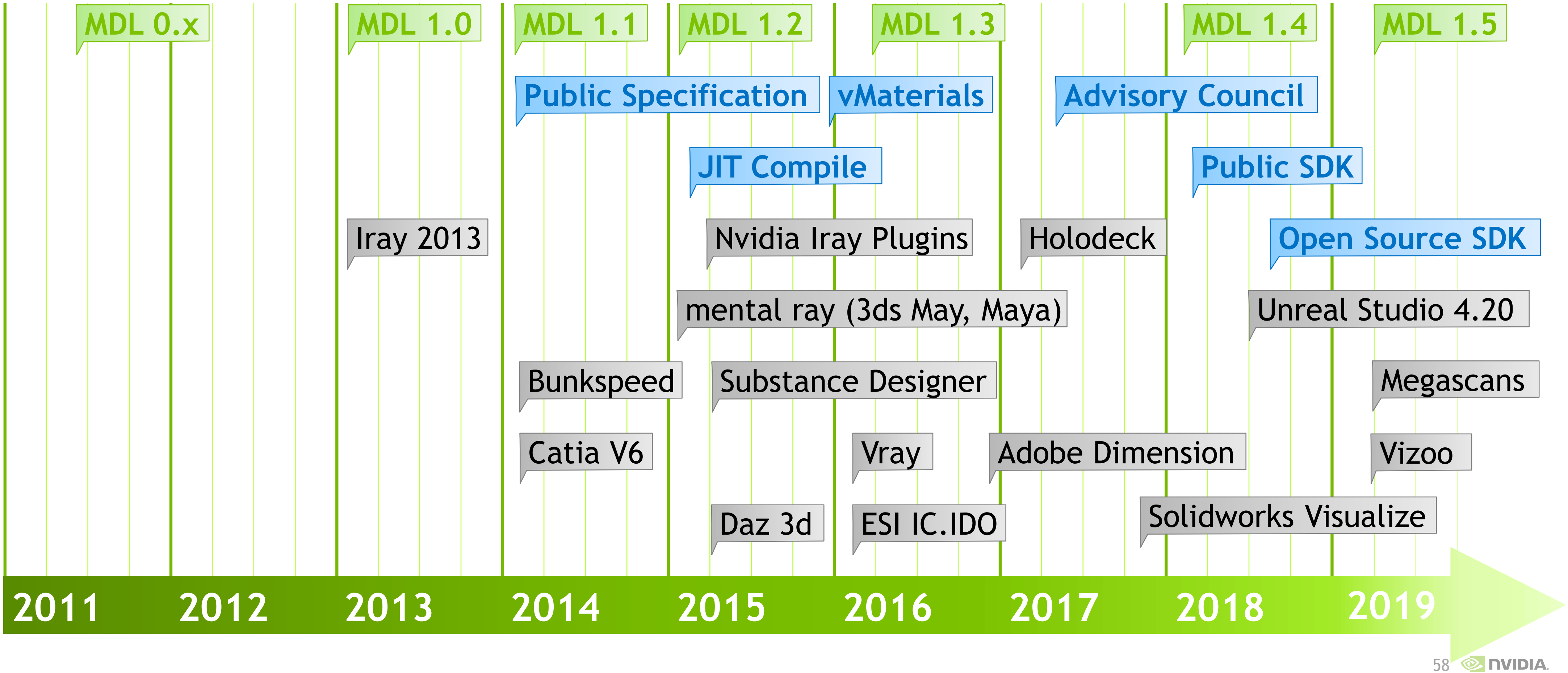


Modules and packages

Archives

MDL Ecosystem

MDL – Past, Present and Future



MDL 0.x

MDL 1.0

MDL 1.1

MDL 1.2

MDL 1.3

MDL 1.4

MDL 1.5

Public Specification

vMaterials

Advisory Council

JIT Compile

Public SDK

Iray 2013

Nvidia Iray Plugins

Holodeck

Open Source SDK

mental ray (3ds May, Maya)

Unreal Studio 4.20

Bunkspeed

Substance Designer

Megascans

Catia V6

Vray

Adobe Dimension

Vizoo

Daz 3d

ESI IC.IDO

Solidworks Visualize

2011

2012

2013

2014

2015

2016

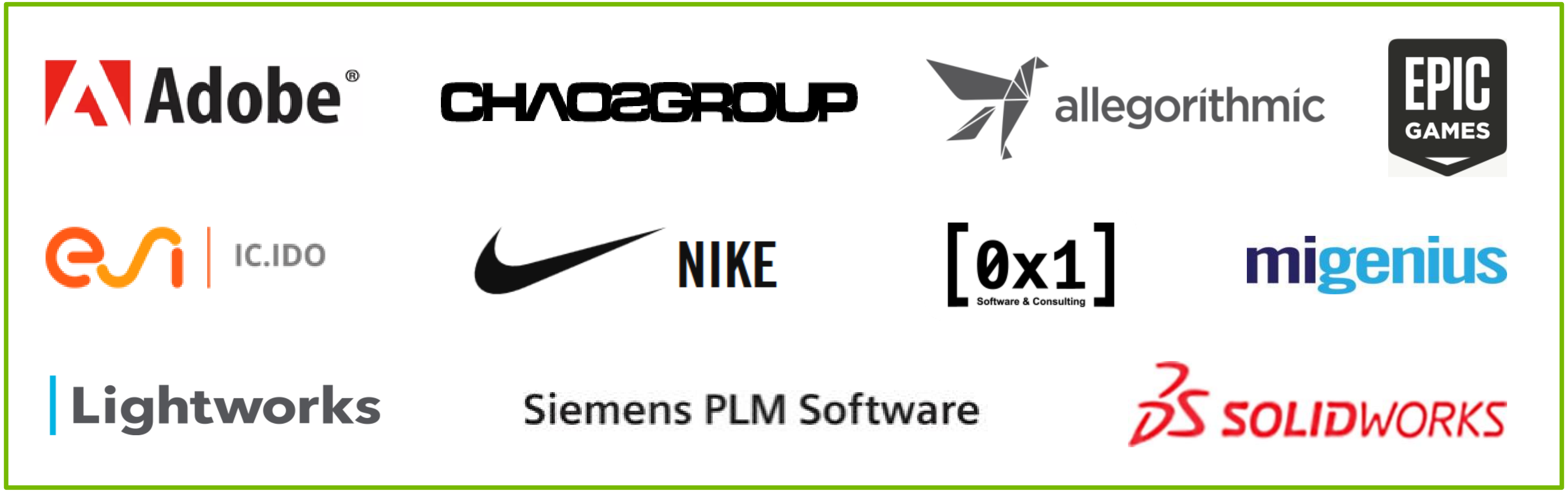
2017

2018

2019

MDL Advisory Council

Companies sharing our vision of MDL



Joint direction of MDL and the MDL eco system

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

NVIDIA Iray

Iray 2019 roadmap

Iray RTX 2019

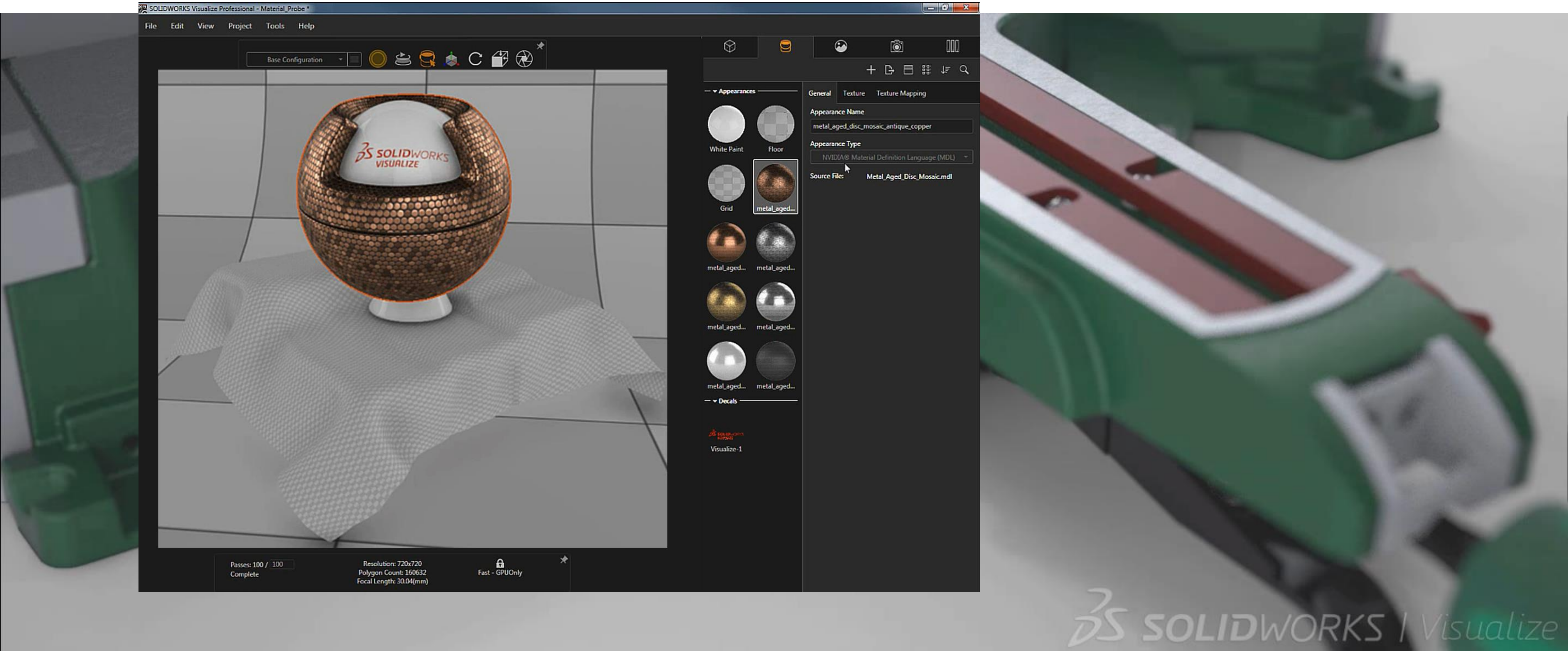
- Release in May
- RTX support, up to 5 times speedup!
- MDL 1.5 support for
  - MDLE
  - localization
  - 2d measured curve

Rendering and Retouching: [zerone]

Rendering and Retouching: [zerone]

SOLIDWORKS Visualize

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



Appearance Name: metal_aged_disc_mosaic_antique_copper

Appearance Type: NVIDIA® Material Definition Language (MDL)

Source File: Metal_Aged_Disc_Mosaic.mdl

Passes: 100 / 100
Complete

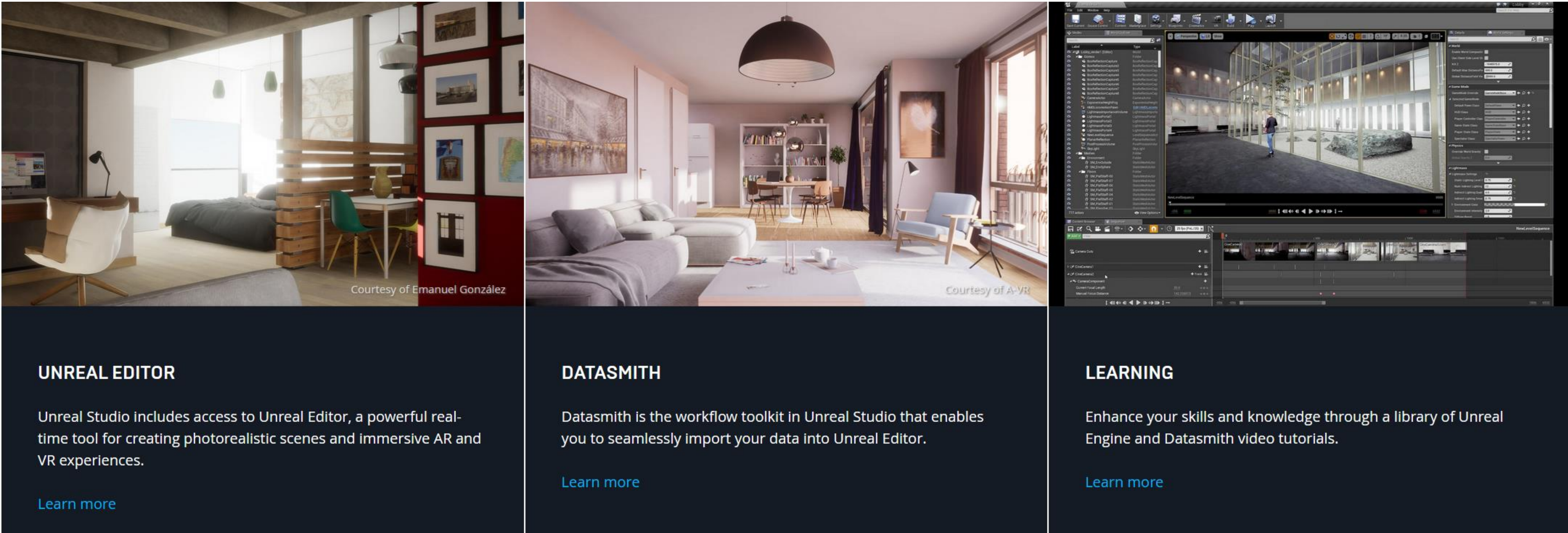
Resolution: 720x720
Polygon Count: 160632
Focal Length: 30.04(mm)

Fast - GPUOnly

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

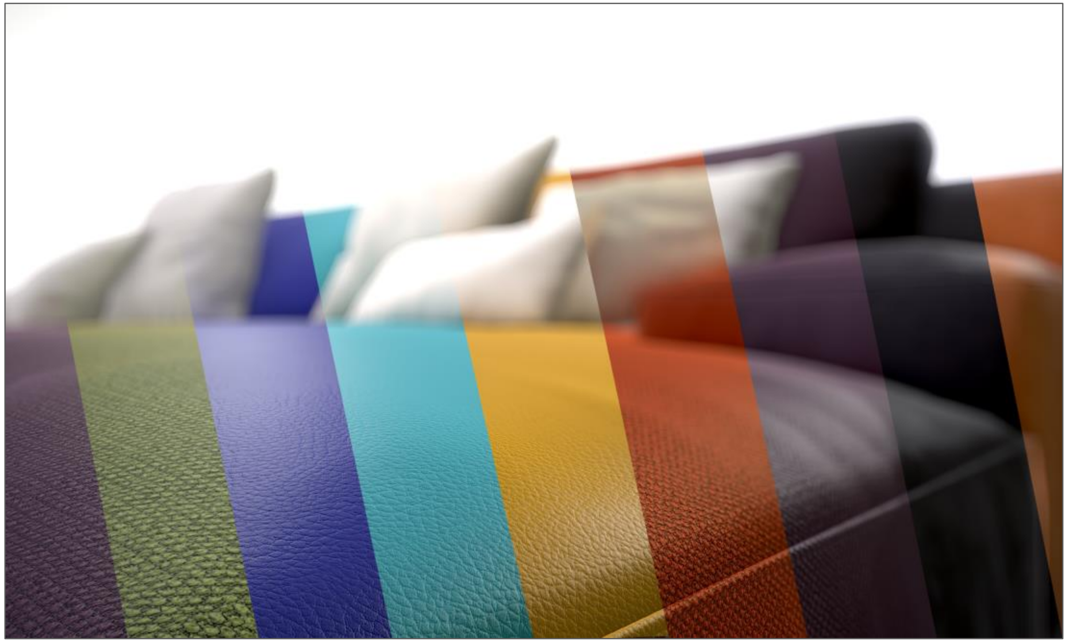
Enhance your skills and knowledge through a library of Unreal Engine and Datasmith video tutorials.

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.“

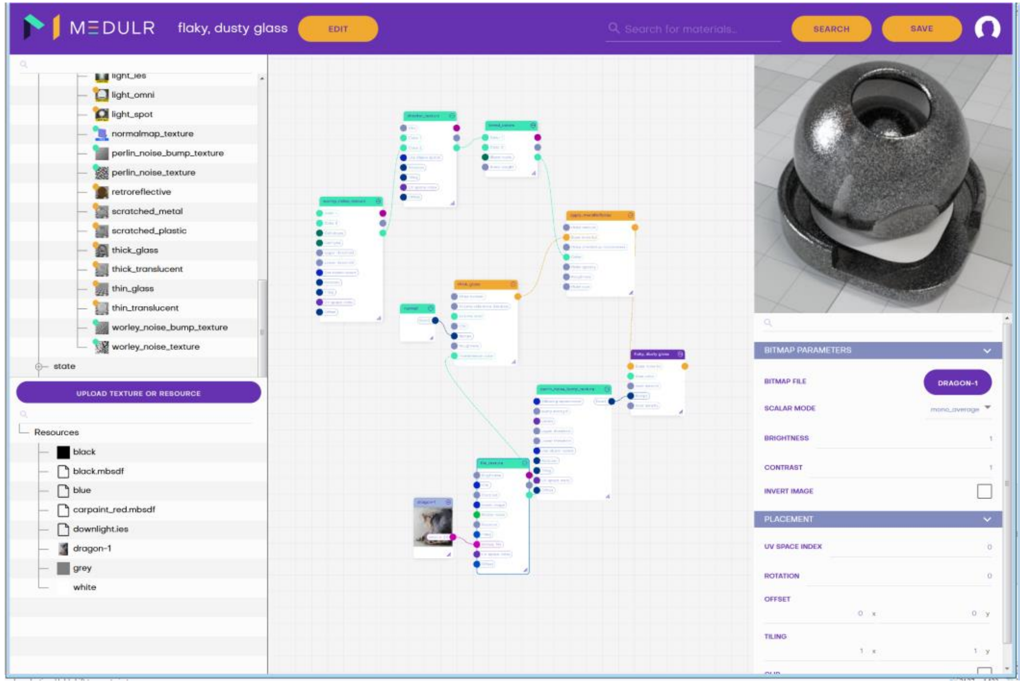


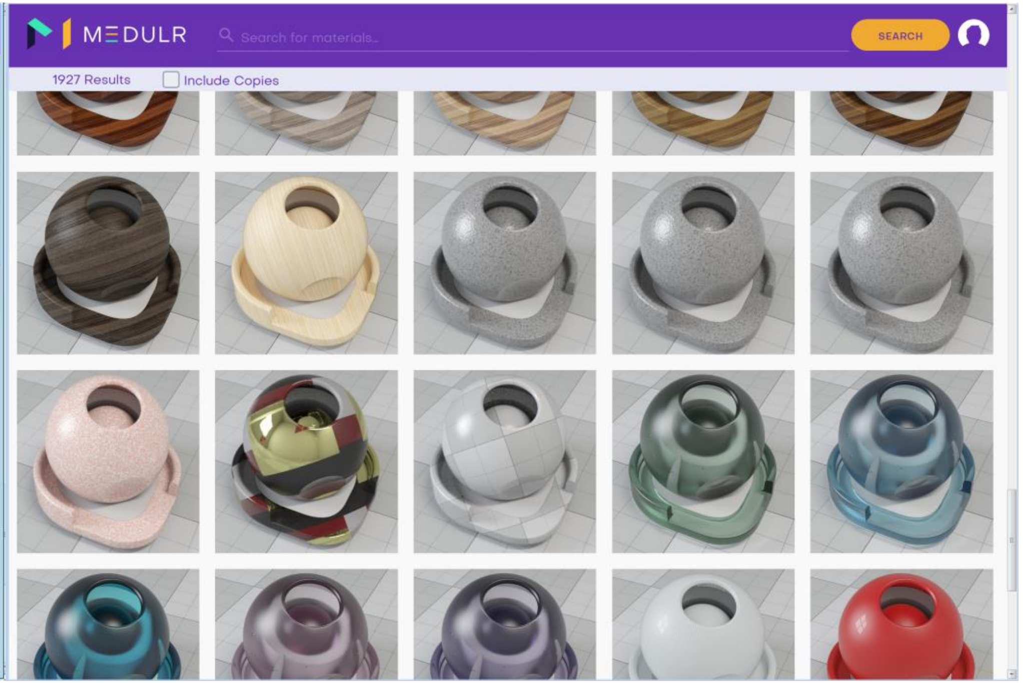
www.vizoo3d.com

MEDULR

Online MDL editor and material library preparing for opening

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

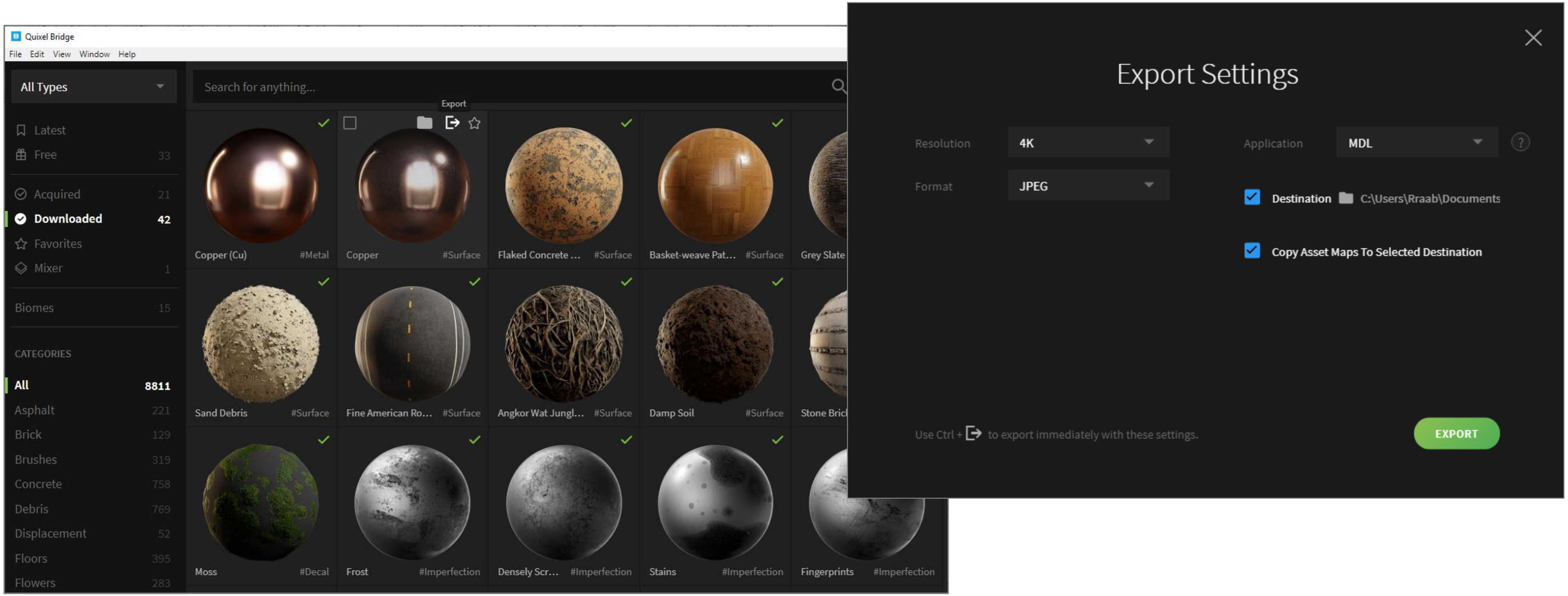




www.medulr.com

Quixel Megascans

“Incredible scans and tools for creatives.”



Export Settings

Resolution: 4K

Format: JPEG

Application: MDL

☑ Destination C:\Users\Rraab\Documents

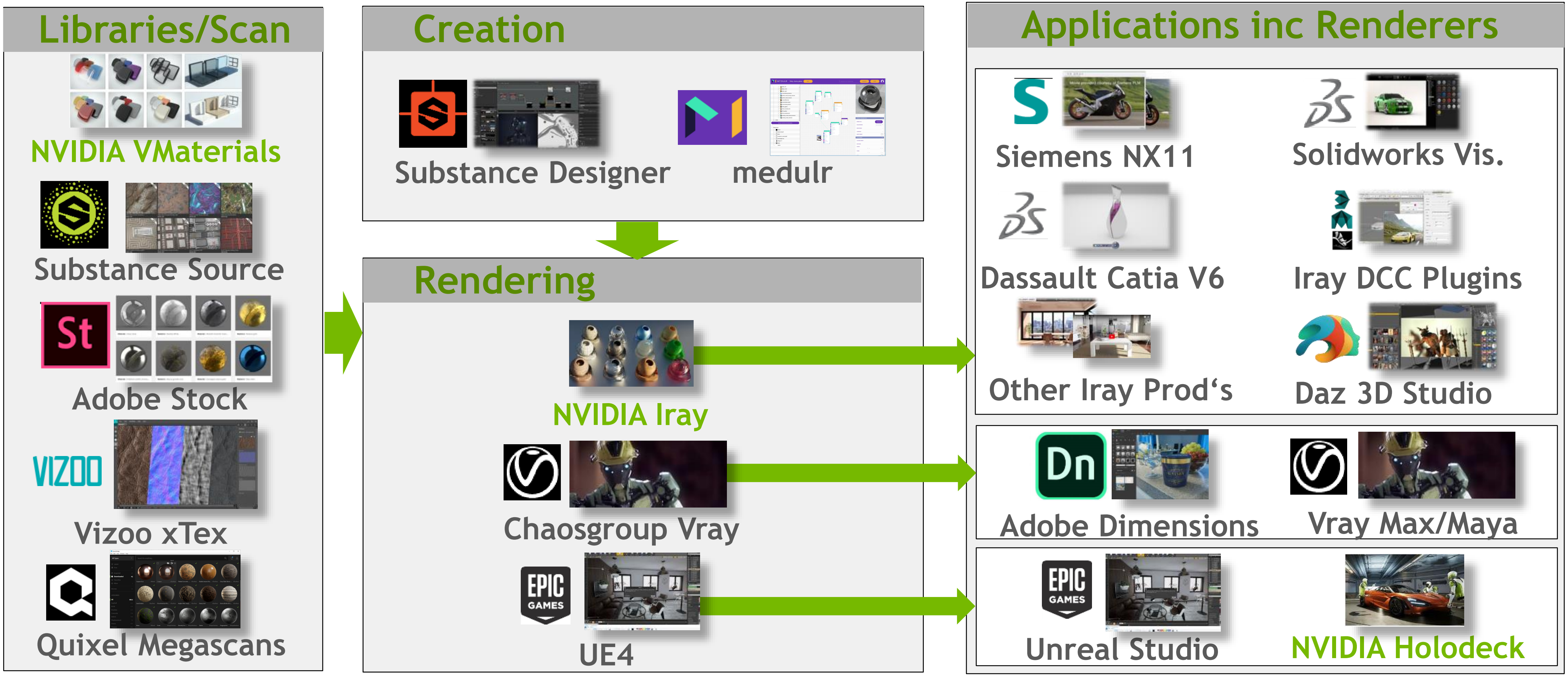
☑ Copy Asset Maps To Selected Destination

Use Ctrl + ⇨ to export immediately with these settings.

EXPORT

quixel.com

MDL Ecosystem



Libraries/Scan

NVIDIA VMaterials

Substance Source

Adobe Stock

Vizoo xTex

Quixel Megascans

Creation

Substance Designer

medulr

Rendering

NVIDIA Iray

Chaosgroup Vray

UE4

Applications inc Renderers

Siemens NX11

Solidworks Vis.

Dassault Catia V6

Iray DCC Plugins

Other Iray Prod's

Daz 3D Studio

Adobe Dimensions

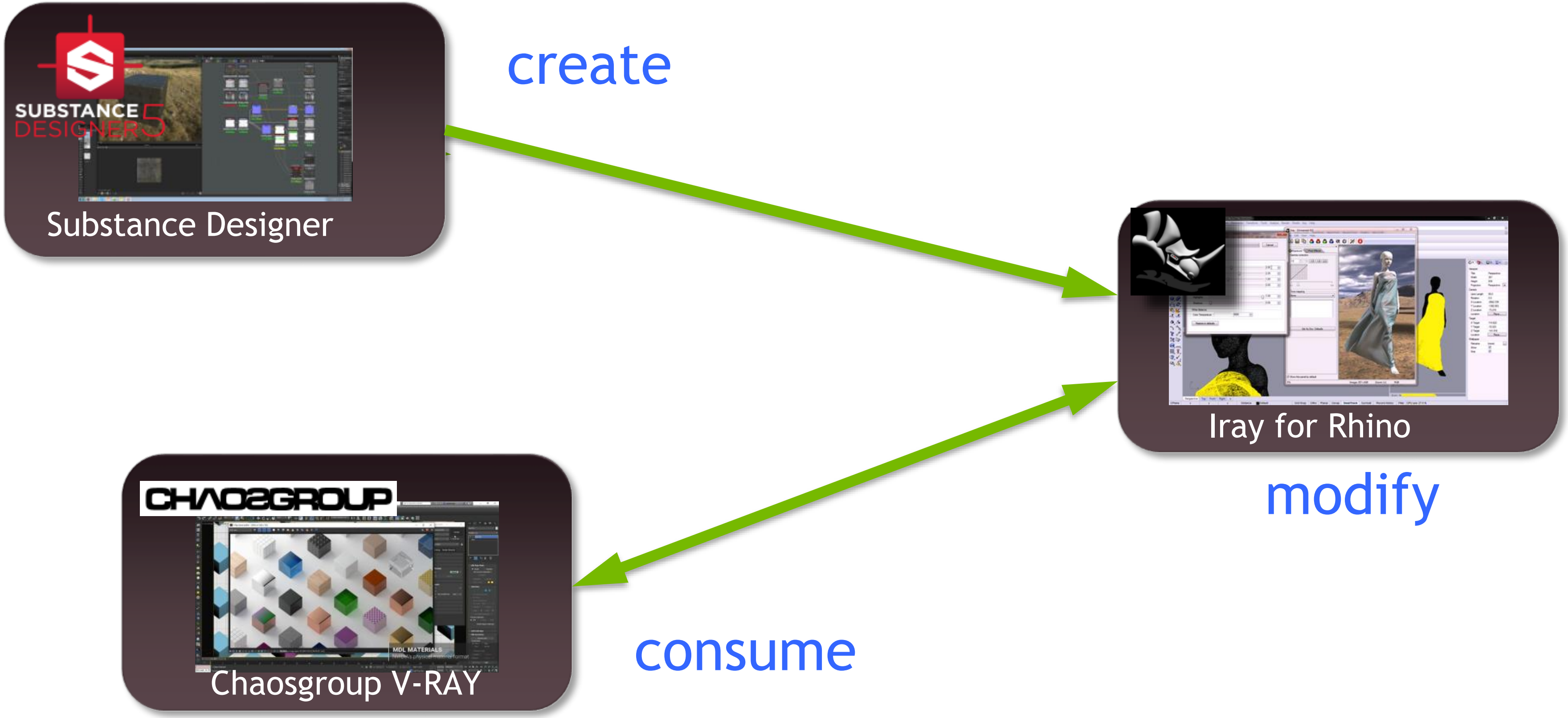
Vray Max/Maya

Unreal Studio

NVIDIA Holodeck

Focus on Material Exchange

Freely choose where to author material content



Substance Designer

create

Iray for Rhino

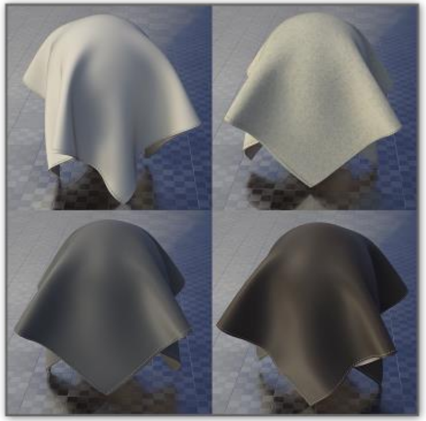
modify

Chaosgroup V-RAY

consume

NVIDIA vMaterials 1.6 – SIGGRAPH 2019

~1700 MDL materials verified for accuracy - FREE TO USE



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

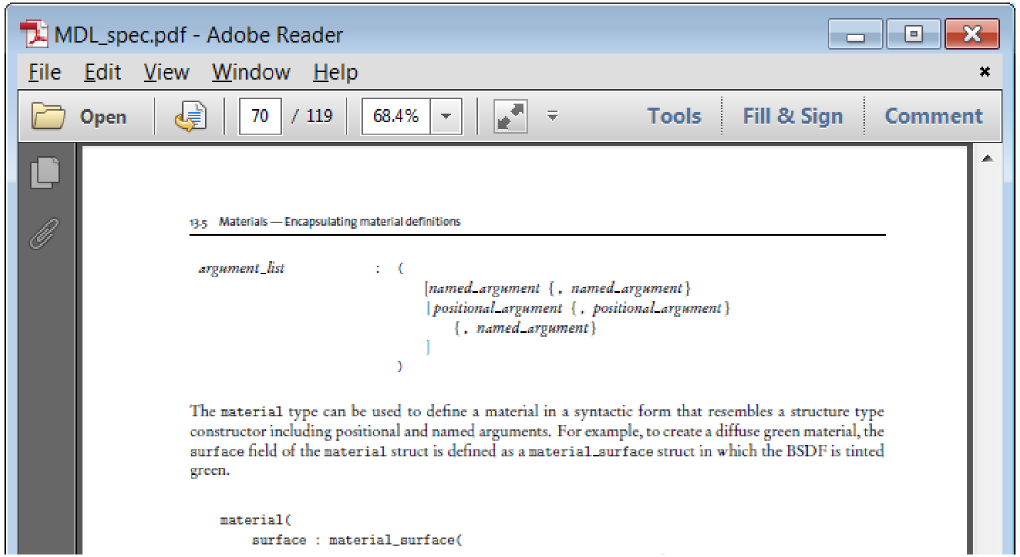
Language specification document
Free to use

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

MDL conformance test suite

Syntactic conformance tests
Semantic conformance tests

Available on request

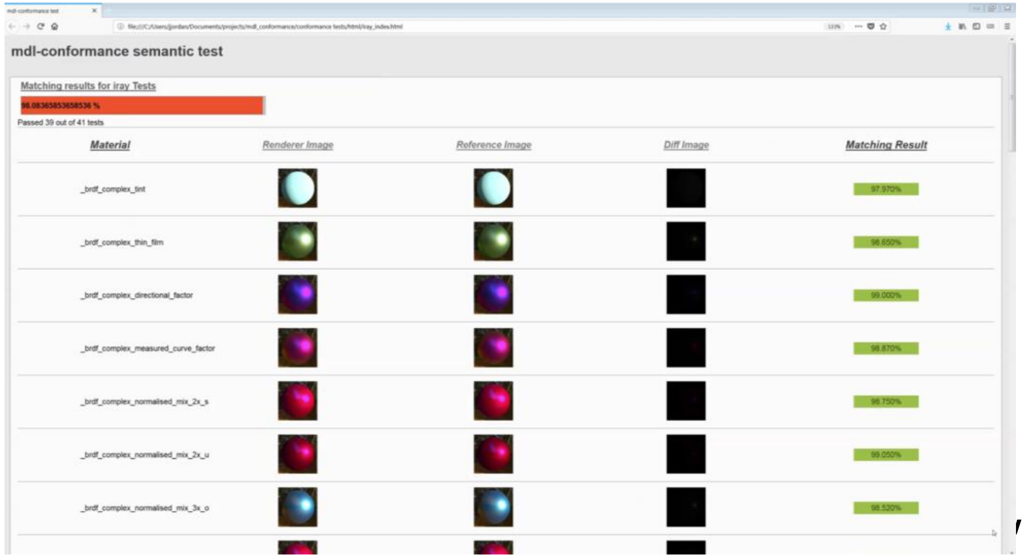


13.5 Materials — Encapsulating material definitions

```
argument_list       :  (
                          [named_argument {, named_argument}
                          | positional_argument {, positional_argument}
                             {, named_argument}
                          ]
                       )
```

The `material` type can be used to define a material in a syntactic form that resembles a structure type constructor including positional and named arguments. For example, to create a diffuse green material, the `surface` field of the `material` struct is defined as a `material_surface` struct in which the BSDF is tinted green.

```
    material(
        surface : material_surface(
```


mdl-conformance test
file:///C:/Users/jjordan/Documents/projects/mdl_conformance/conformance tests/html/iray_index.html

mdl-conformance semantic test

Matching results for iray Tests

98.08365853658536 %

Passed 39 out of 41 tests

Material	Renderer Image	Reference Image	Diff Image	Matching Result
_brdf_complex_tint				97.970%
_brdf_complex_thin_film				98.650%
_brdf_complex_directional_factor				99.000%
_brdf_complex_measured_curve_factor				98.870%
_brdf_complex_normalised_mix_2x_s				98.750%
_brdf_complex_normalised_mix_2x_u				99.050%
_brdf_complex_normalised_mix_3x_o				98.520%

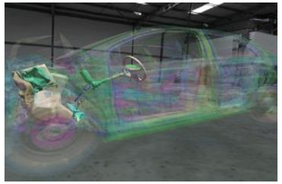
NVIDIA. DESIGNWORKS

RENDERING

Iray SDK

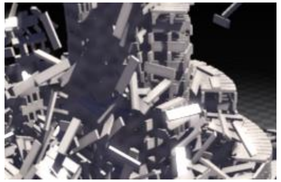
OptiX SDK

MDL SDK


NV Pro Pipeline

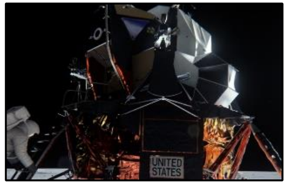
vMaterials

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 (.0.1)

Features

MDL 1.4 (1.5 feature previews)

DB for MDL definitions

DAG view on materials
several compilation modes

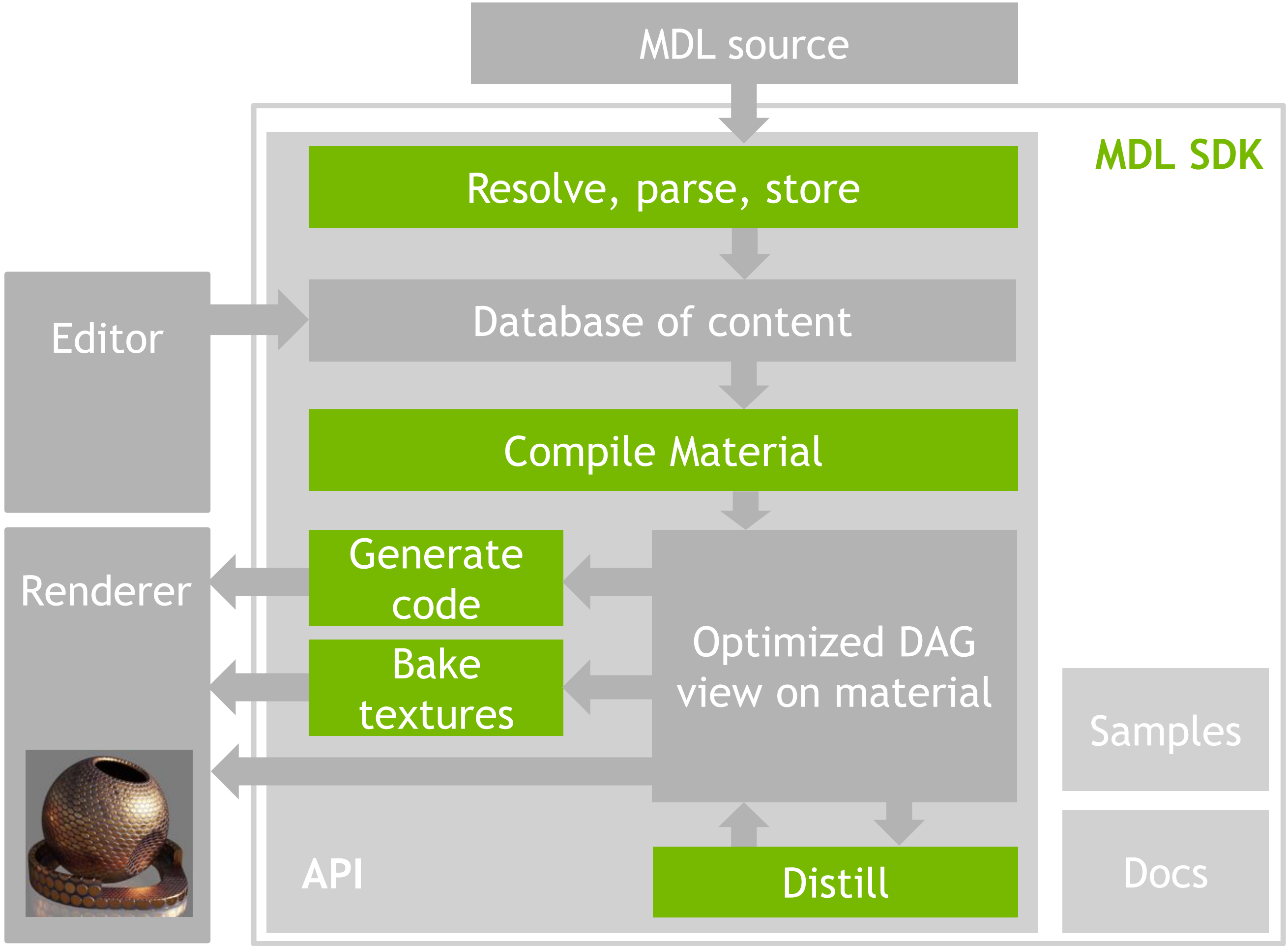
MDL editing

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

Distiller and texture baker

Samples

Documentation and tutorials



MDL SDK 2019 - What is New

Features

Preview of MDL 1.5 features:

- Localisation
- MDL encapsulated format

Improved BSDF reference implementation (libbsdf) including measured brdf and emissive distribution functions

Additional distilling mode (transmissive PBR)

HLSL backend (2019.0.1)

Automatic derivatives for texture filtering

Open source release available on Github

- Includes exclusive MDL core compiler API

More samples

- Updated CUDA sample for transmissive materials
- (CPU rendering sample)?
- MDL browser sample

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

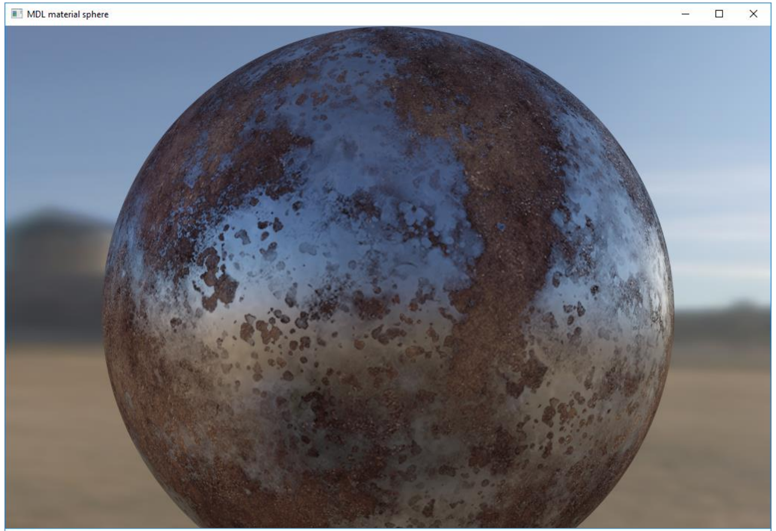
Microsoft DXR

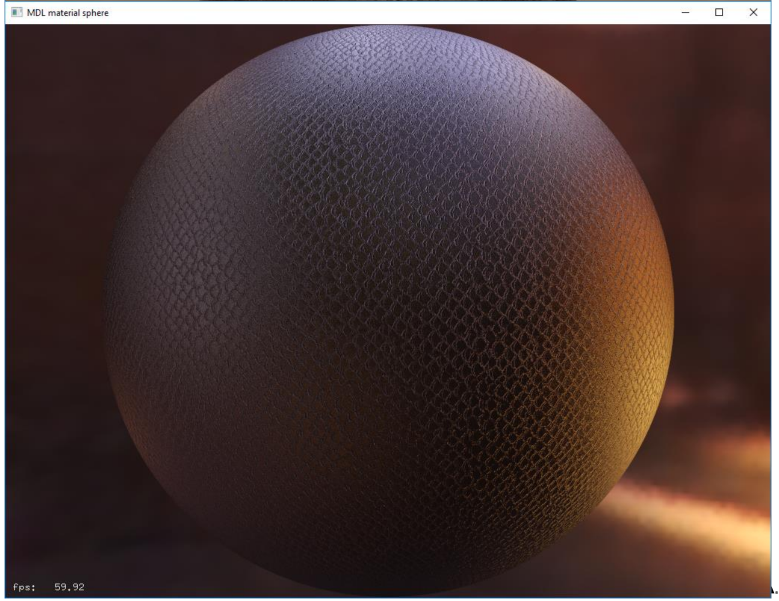
- HLSL back-end with MDL SDK 2019.0.1
  and sample path tracer in the SDK

NVIDIA OptiX

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

Integrating MDL with an RTX based renderer is simple!

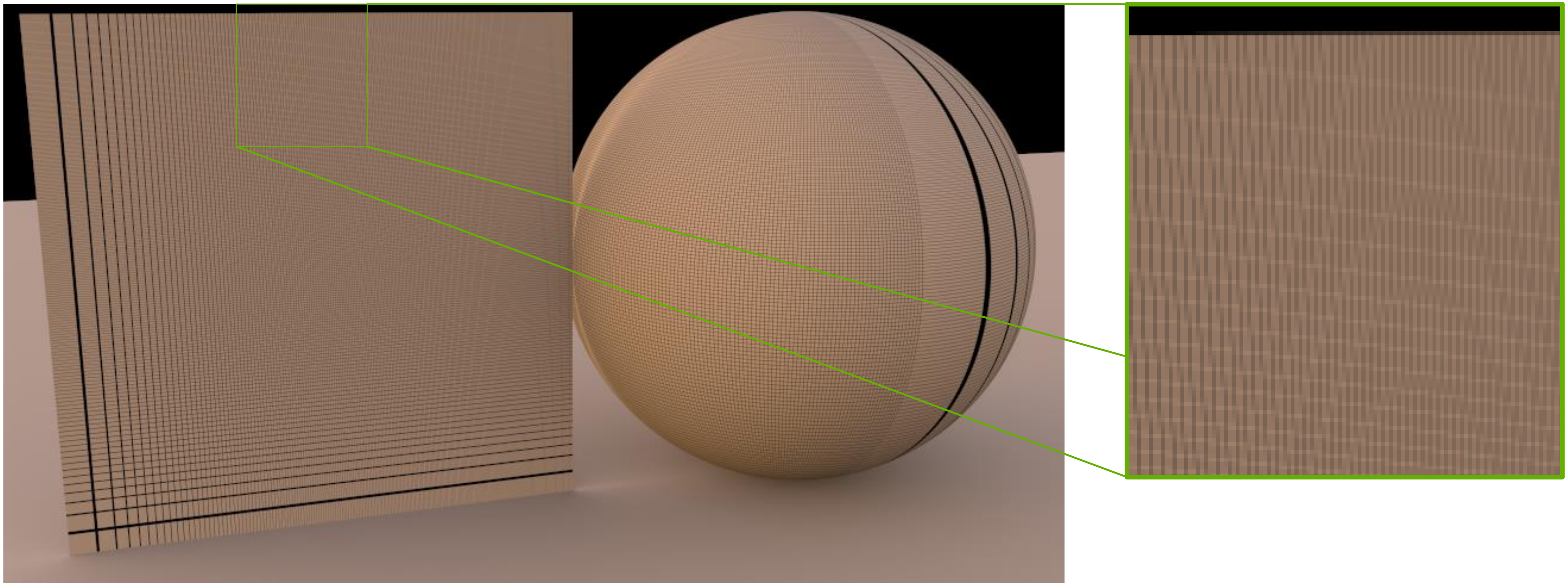




```
fps:   59.92
```

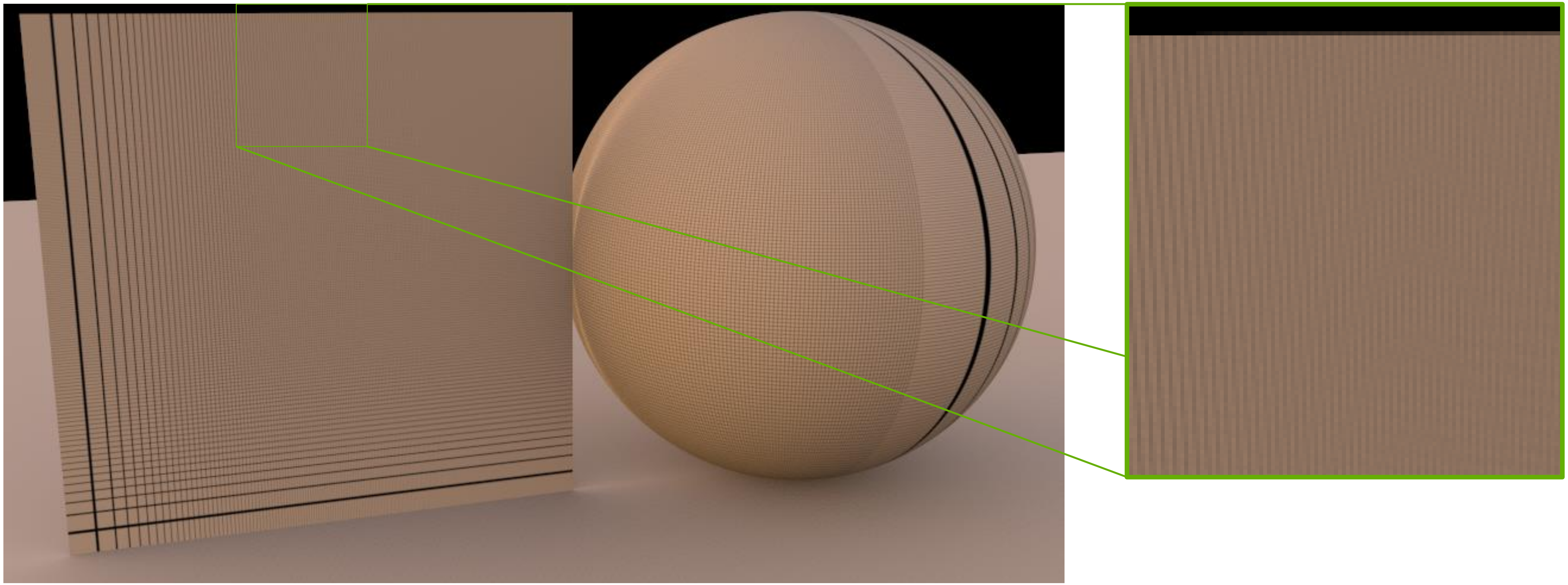
Automatic Derivatives for Texture Filtering

OptiX sample renderer integration: Derivatives off



Automatic Derivatives for Texture Filtering

OptiX sample renderer integration: Derivatives on



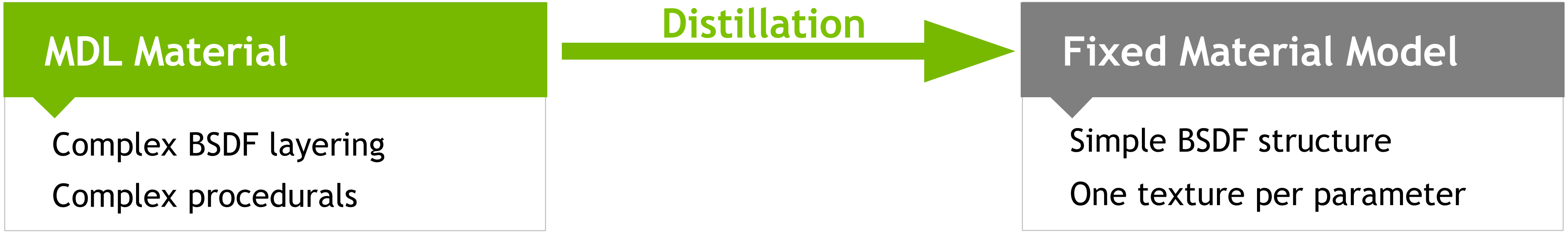
MDL in Realtime Rendering

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

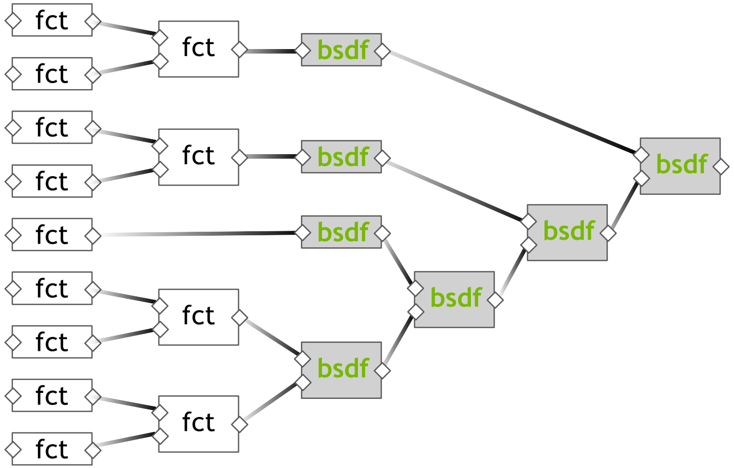
Complex procedurals

Distillation →

Fixed Material Model

Simple BSDF structure

One texture per parameter



Distillation to Fixed Material Model

MDL Material

Complex BSDF layering

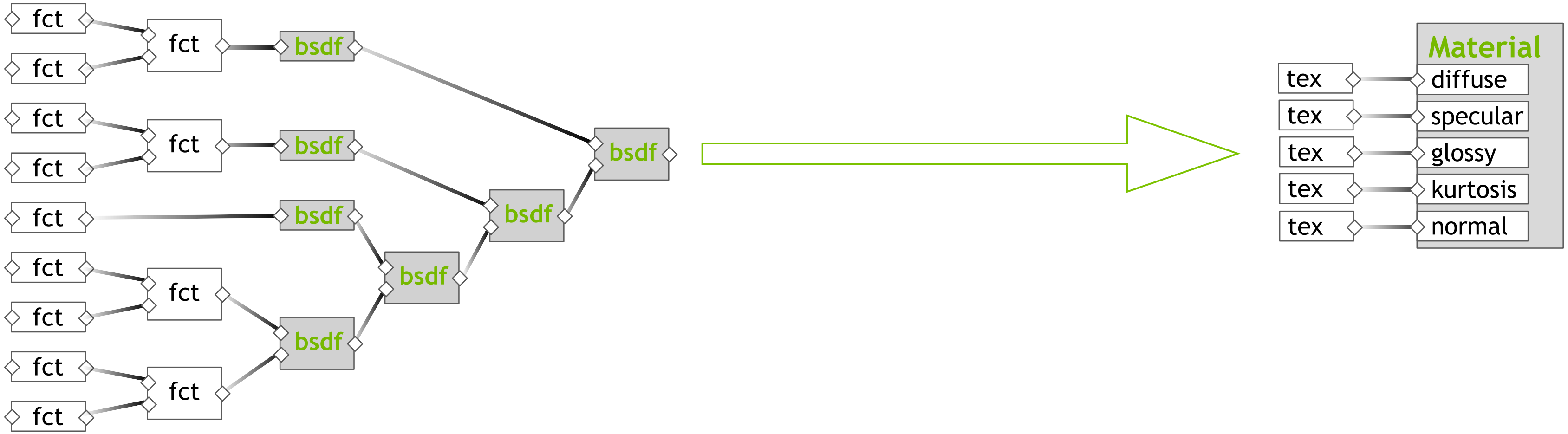
Complex procedurals

Distillation

Fixed Material Model

Simple BSDF structure

One texture per parameter



Distillation to Fixed Material Model

MDL Material

Complex BSDF layering

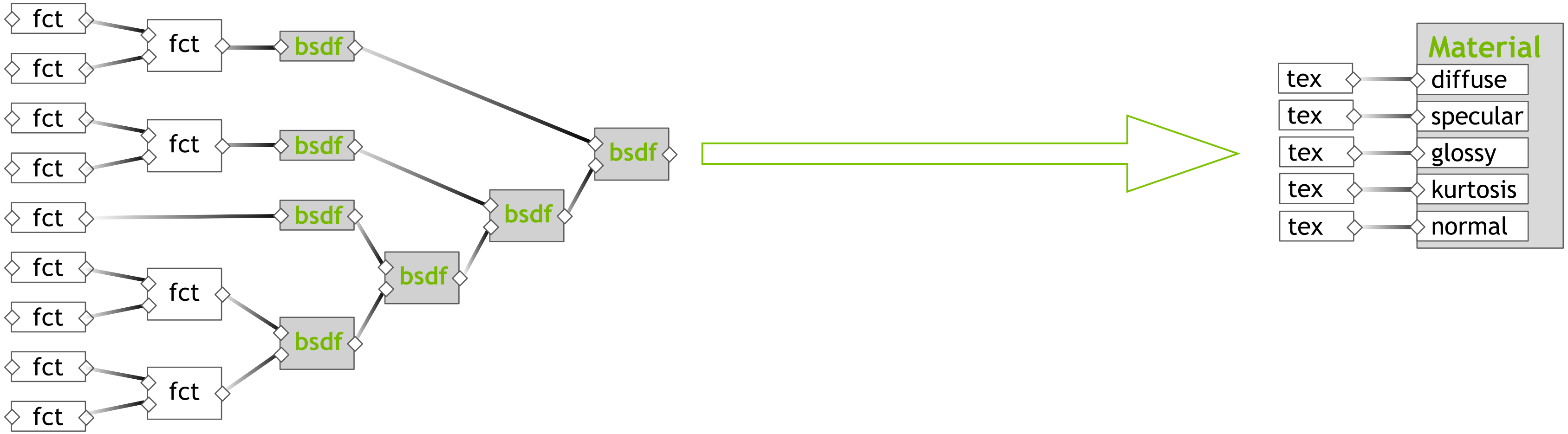
Complex procedurals

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

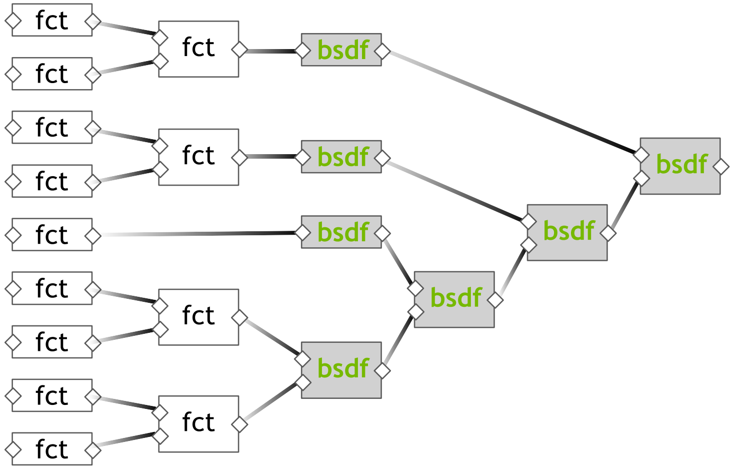
Complex procedurals

Distillation

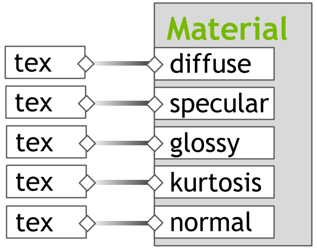
Fixed Material Model

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

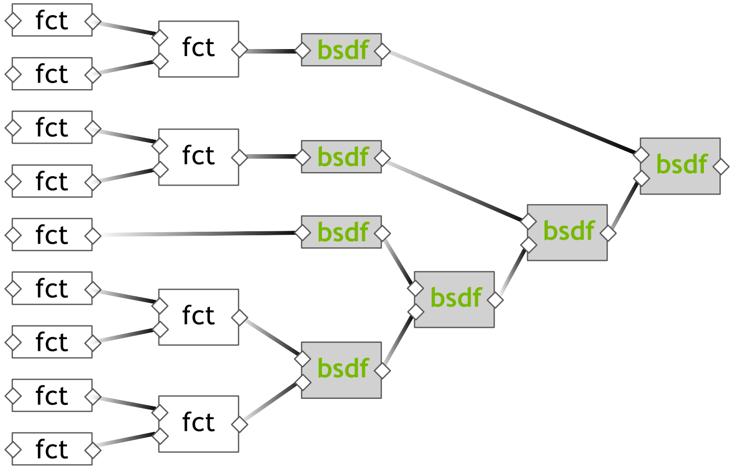
Complex procedurals

Distillation

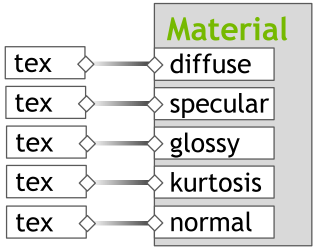
Fixed Material Model

Simple BSDF structure

One texture per parameter



Fast projection of material instances: Realtime editing

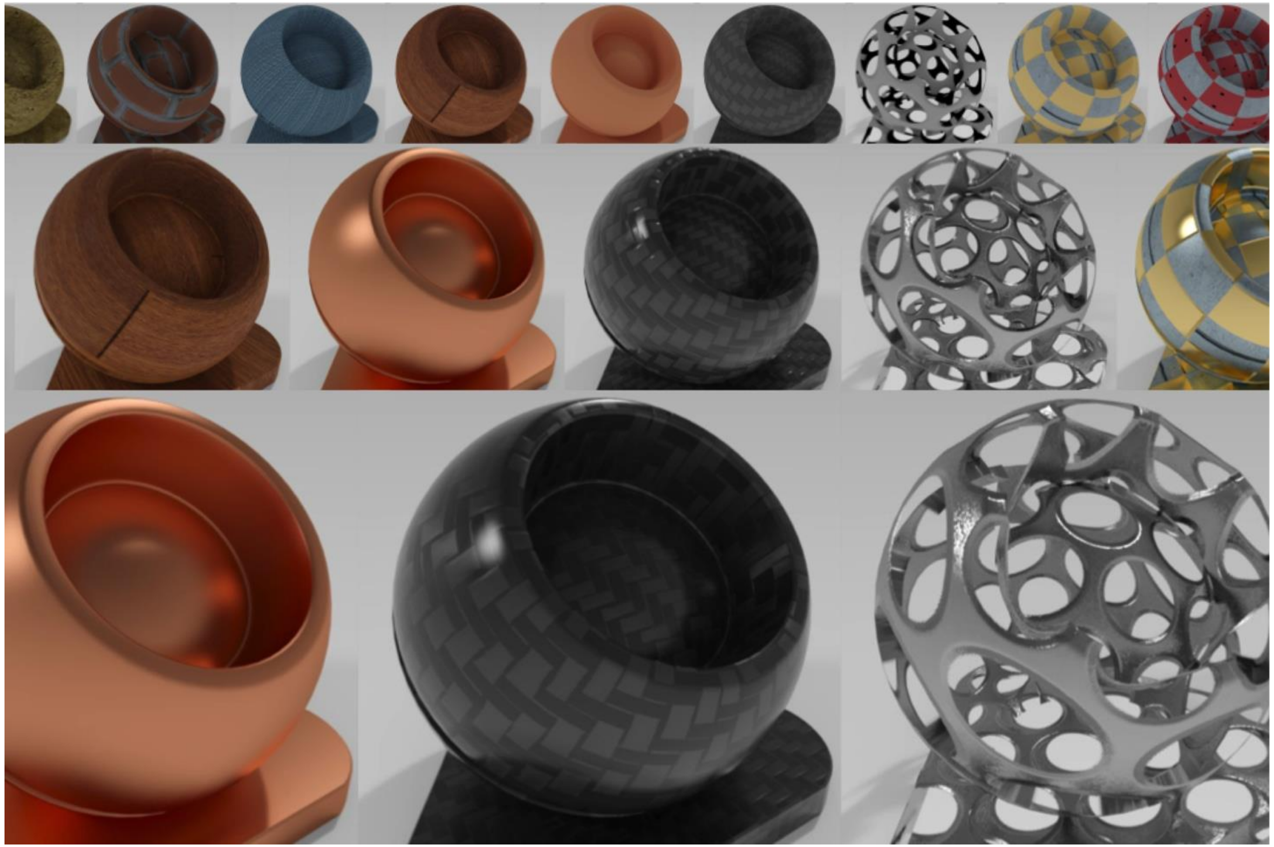


Approximate render result: Some materials will look quite different

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

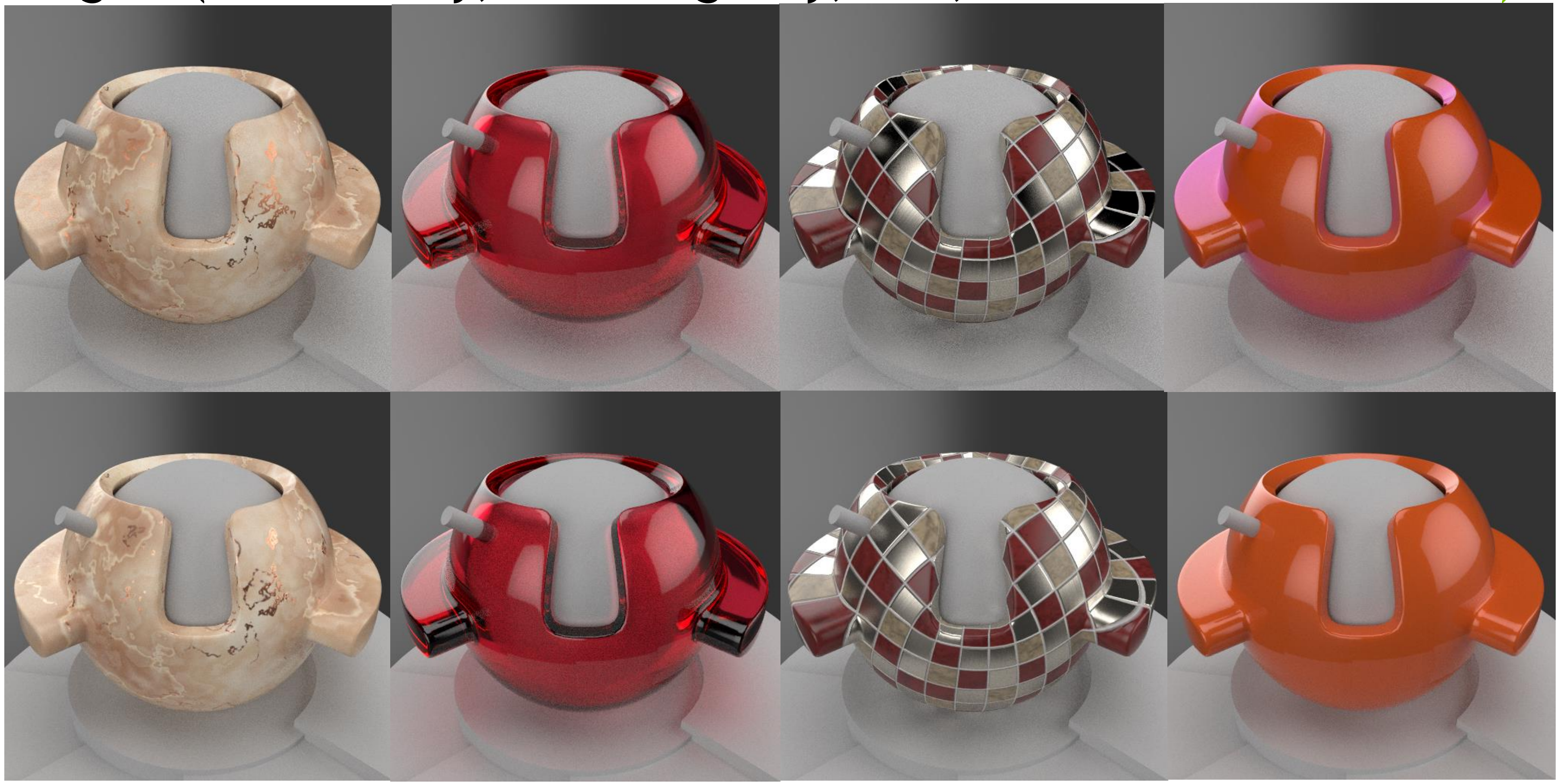
Fresnel(glossy, diffuse)

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

May the Source Be with You

NVIDIA Open Sourced 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

4 releases shipped since SIGGRAPH 2018

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

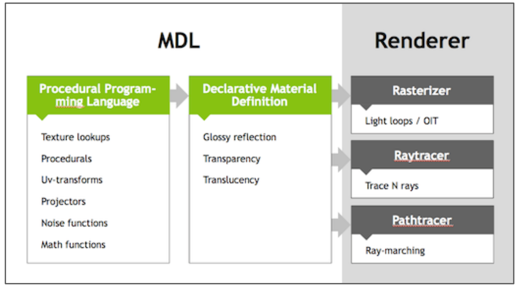
MDL Core API

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

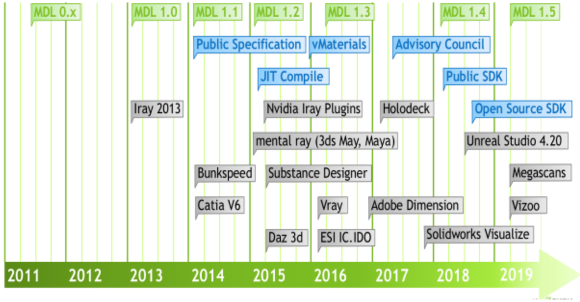
What is MDL



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*
- *Handbook*
- *Language Specification*

GTC On-Demand

on-demand-gtc.gputechconf.com

MDL@GTC

Mon 9 AM SJCC 230B	*Sharing Physically Based Materials Between Renderers with MDL*
Mon 10 AM SJCC 230B	*Integrating the NVIDIA Material Definition Language MDL in Your Application*
Mon 11 AM Hilton Hotel Almaden 2	*A New PBR Material Serving Mobile, Web, Real-Time Engines and Ray Tracing*
Tue 9 AM Hilton Hotel Almaden 2	*Multi-Platform Photo-Real Rendering: Utilizing NVIDIA'S MDL and Allegorithmic's Substance Suite for Product Imaging*
Thu 10 AM SJCC 230C	*Real-Time Ray Tracing with MDL Materials*