

MaterialX and OpenPBR Virtual Town Hall





MaterialX Project Updates 2026 – Jonathan Stone (Lucasfilm, ASWF)

OpenPBR Project Updates 2026 – Adrien Herubel (Autodesk, ASWF)

The MaterialX Stirling – Leif Pedersen (Pixar)

The MaterialX Teapot and Lion – Andrea Weidlich (NVIDIA)

MaterialX and OpenPBR in Houdini 22 – Chris Rydalch (SideFX)

The MaterialX Playground – João Vítor Silva

MaterialX and OpenPBR in OctaneRender – Arnon Marcus (OTOY)

OpenPBR and MaterialX in Blender – Sebastian Herholz (Blender), Owen O'Malley

OpenPBR Volume – Chris Rydalch (SideFX), Adrien Herubel (Autodesk)

MaterialX Project Updates 2026

Jonathan Stone – Lucasfilm, ASWF



MaterialX v1.39.5

- Released in May of 2026, and now integrated into OpenUSD 26.08
- The largest set of contributors in the history of the MaterialX project
- Adding support for Slang shader generation, flake nodes, retro-reflective BSDFs, graph editor improvements, and much more



MaterialX Scattering Equations

- Contributing formal mathematics for the Physically Based Shading Nodes
- A collaboration between the MaterialX project and the Alliance for OpenUSD Materials Working Group
- Enabling a new wave of open path tracers for MaterialX and OpenUSD content



MaterialX Stirling

- A new production-quality example asset from Pixar
- Built entirely from MaterialX nodes, leveraging new features in v1.39.5
- More details in the upcoming talk from Leif Pedersen



MaterialX Teapot and Lion

- Two new production-quality example assets from NVIDIA
- Built entirely from MaterialX nodes, and approved by the ASWF DPEL
- More details in the upcoming talk from Andrea Weidlich



ASWF Dev Days 2026

- Our next ASWF Dev Days event will be on September 17th
- A chance for new developers to receive guidance on a one-day contribution to an ASWF project
- Register at <https://www.aswf.io/dev-days/>



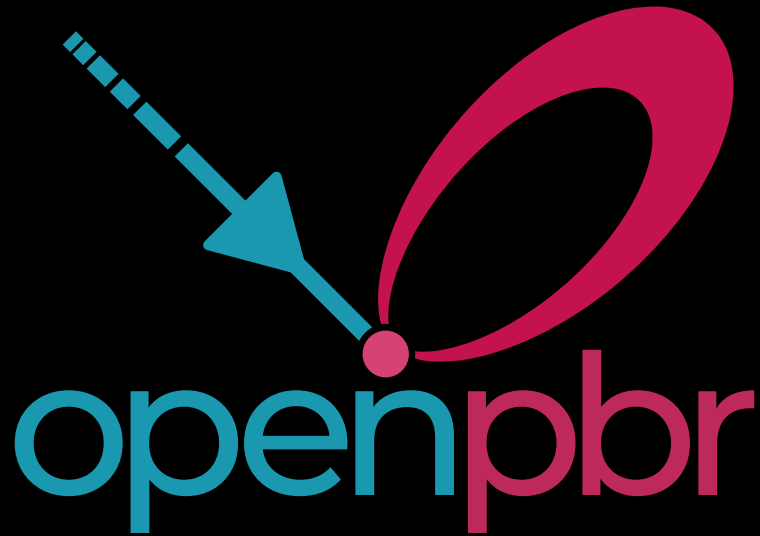
ASWF

/* ACADEMY
SOFTWARE
FOUNDATION



Adrien Herubel
Director, Software Engineering
Autodesk

14/09/2026



Project Update

MaterialX Virtual Town Hall 2026

Updates

- Contributions
 - 34 PRs merged in fix and feature branches
- OpenPBR 1.1.1 released in MaterialX 1.39.5
 - Nonbreaking wording improvements
 - MaterialX implementation fixes
- OpenPBR 1.2 Feature Complete



Industry adoption

Adobe

 **UNREAL ENGINE**

 **octane**render

 **AUTODESK**

 **REDSHIFT**

 **blender**[®]

 **NVIDIA OMNIVERSE**[™]

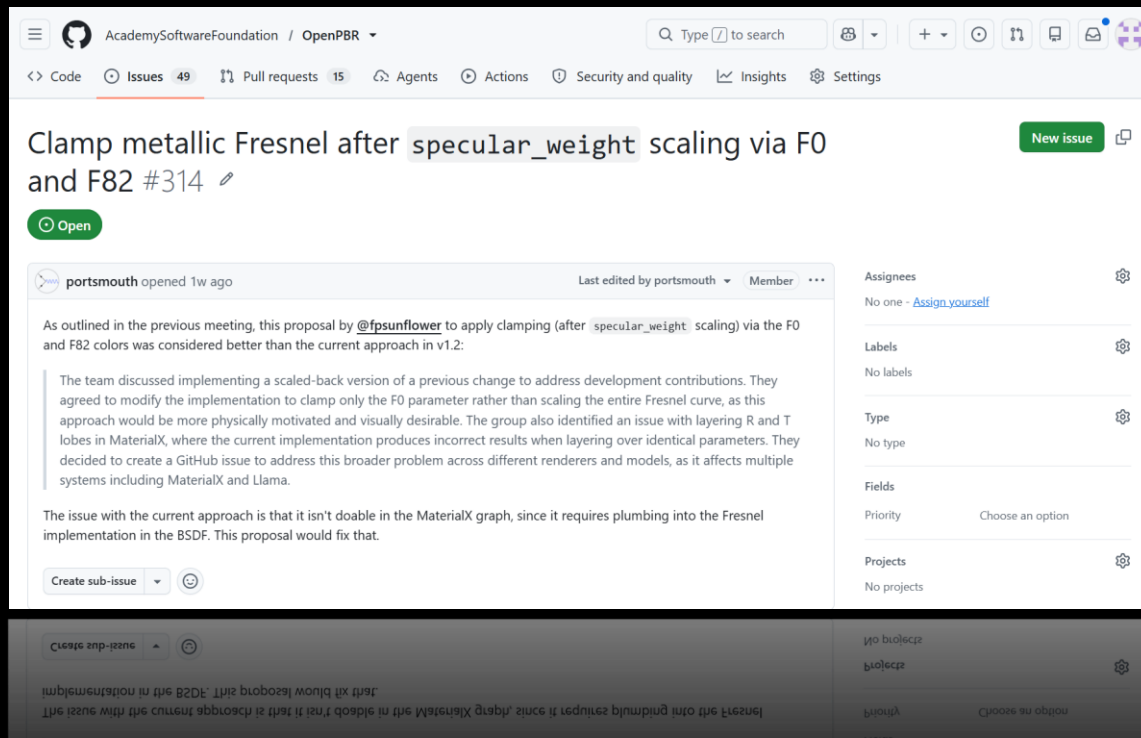


 **chaos V-Ray**

 **SideFX**[®]

Correctness and Validation

- New implementers help clarify the specification



The screenshot shows a GitHub issue page for the repository 'AcademySoftwareFoundation / OpenPBR'. The issue title is 'Clamp metallic Fresnel after specular_weight scaling via F0 and F82 #314'. It is marked as 'Open' and was created by 'portsmouth' one week ago. The issue description discusses a proposal to apply clamping to the F0 parameter instead of the entire Fresnel curve, citing a previous meeting and the need to address incorrect results in MaterialX and Llama renderers. The right sidebar shows the issue's metadata, including assignees, labels, type, and fields.

AcademySoftwareFoundation / OpenPBR

Search: Type [7] to search

Code Issues 49 Pull requests 15 Agents Actions Security and quality Insights Settings

Clamp metallic Fresnel after specular_weight scaling via F0 and F82 #314

New issue

Open

portsmouth opened 1w ago Last edited by portsmouth Member

As outlined in the previous meeting, this proposal by @fpsunflower to apply clamping (after specular_weight scaling) via the F0 and F82 colors was considered better than the current approach in v1.2:

The team discussed implementing a scaled-back version of a previous change to address development contributions. They agreed to modify the implementation to clamp only the F0 parameter rather than scaling the entire Fresnel curve, as this approach would be more physically motivated and visually desirable. The group also identified an issue with layering R and T lobes in MaterialX, where the current implementation produces incorrect results when layering over identical parameters. They decided to create a GitHub issue to address this broader problem across different renderers and models, as it affects multiple systems including MaterialX and Llama.

The issue with the current approach is that it isn't doable in the MaterialX graph, since it requires plumbing into the Fresnel implementation in the BSDF. This proposal would fix that.

Create sub-issue

Assignees: No one - Assign yourself

Labels: No labels

Type: No type

Fields: Priority: Choose an option

Projects: No projects

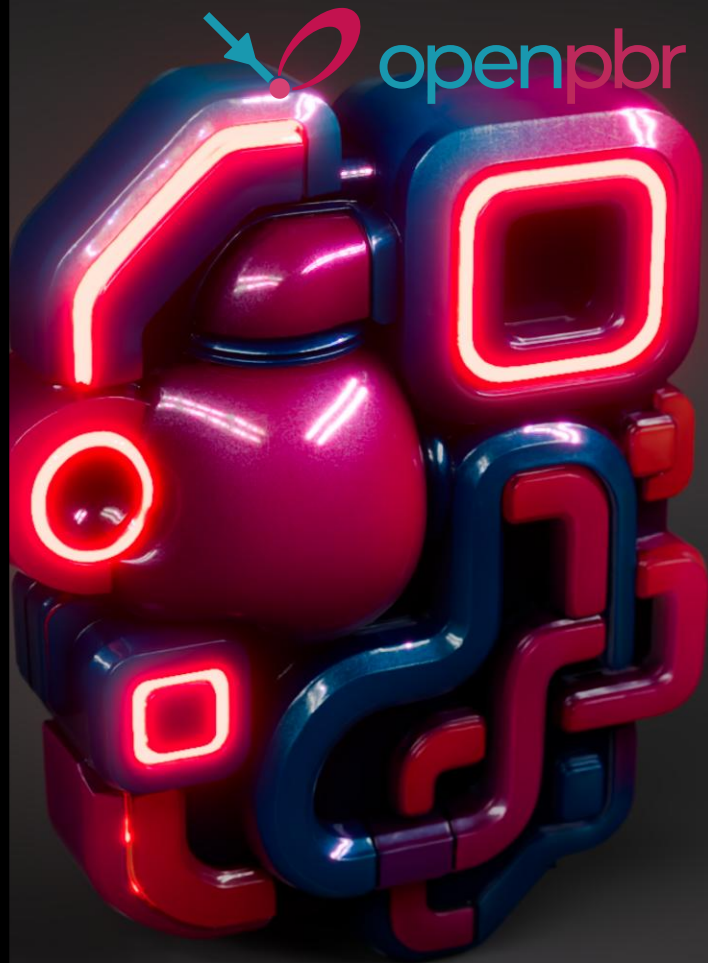
Correctness and Validation



- Validation and reference implementations
 - NVIDIA typhoon
 - Reference pathtracer for USD, MaterialX
 - Blender OpenPBR testsuite
 - Multirenderer validation
- Make it easy for implementers to check correctness
- Ensure implementations are converging

OpenPBR 1.2 is Feature Complete

- Specular retro-reflectivity
- Specular Haze
- Emission weight
 - Better parameter symmetry
- Unbound specular weight
 - Boost reflectivity without breaking physicality
- Decoupled specular IOR
 - Adjust reflectivity without affecting refraction direction
- Transmission scatter as volume albedo
 - Simplified parametrization



Retro Reflectivity

- New contribution
- *The Mininal Retroreflective Microfacet Model*
- Collaboration between Autodesk, NVIDIA, Intel
- In MaterialX 1.38.6
- Single parameter

The Minimal Retroreflective Microfacet Model

Jamie Portsmouth  Autodesk
Matthias Raab  NVIDIA
Laurent Belcour  Intel
Francis Liu  NVIDIA



Figure 1. Renderings of our MRM model (bottom row) obtained by an almost trivial modification of the equivalent roughness GGX microfacet model (top row). The lighting in all cases is aligned with the view direction, exhibiting a strong retroreflective peak in the MRM model.

Abstract

We present the Minimal Retroreflective Microfacet (MRM) model, which turns any existing microfacet BSDF into a physically plausible retroreflective one by a single substitution: replacing the view direction with its reflection about the surface normal before evaluating the standard model. Based on the previously published back-vector formulation, MRM requires only minimal code changes and has been adopted in the OpenPBR and MaterialX material standards. We prove reciprocity and energy conservation under the assumption of a reflection-symmetric normal distribution function (NDF), which holds for all commonly used distributions, and validate the model against measured retroreflective material data.





Specular Haze

- Represent smudged reflective surface
- Two parameters
 - specular_haze
 - specular_haze_spread
- Easy to use dual specular model



Specular Haze



What is next ?

- OpenPBR 1.3
 - Better Thin-walled translucency ?
 - A Biophysical Approach to Foliage Rendering: Time-Dependent BSDF for Aging Leaves - EGSR26
 - Glint NDFs ?
 - [Evaluating and Sampling Glinty NDFs in Constant Time](#)
- Beyond surfaces

How to contribute

- GitHub
 - Issues and PRs
 - <https://github.com/AcademySoftwareFoundation/OpenPBR>
- ASWF Slack
 - #openpbr
- ASWF Zoom OpenPBR meeting
 - Bi-weekly alternating with the MaterialX meeting



Cars 2 © Disney/Pixar



MATERIALX

Stirling

Leif Pedersen | Pixar

A Collaboration between
Lucasfilm and Pixar to bring
native MaterialX support to
RenderMan XPU.

Model was modified by Andrew Schmidt from
an initiative by Dylan Sisson. MaterialX and
Lama conversion by Leif Pedersen.



Cars 2 © Disney/Pixar

Motivations And Scope

- Pixar Production asset. Modified version of Finn McMissile, from *Cars 2*.
- Share a portable asset using OpenUSD and MaterialX.
- Validation of pixel parity feature set for both CPU and GPU.
- Shared development environment helps develop a robust and forward thinking code base.

Motivations and Scope

Shared Goals for MaterialX across Disney Studios

- Final renders for film and streaming
- Accurate artist previews across DCC tools
- Real-time experiences and LED walls
- Digital backlots and asset persistence
- Asset interchange with partner studios

MaterialX in RenderMan

- Lucasfilm and Pixar both rely upon RenderMan for final renders
- MaterialX BSDFs in RenderMan would complete the picture

Pixar and Lucasfilm Collaboration

- Introduction of RenderMan XPU opened a window of opportunity



Cross-Disney Studio Collaboration Pixar & Lucasfilm



Shipped!



PIXAR



LUCASFILM
Ltd

Tech At-A-Glance

Production-Proven Foundation

- Built on the existing MaterialX Lama implementation in XPU.
- Reuses shared RenderMan scattering classes and functions.
- Proven in production at ILM.

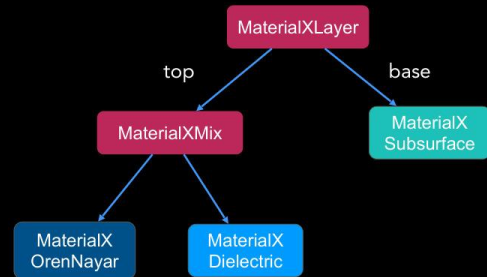
Unified CPU/GPU Execution and Pixel Parity

- OSL emits data into a preallocated closure pool.
- C++/CUDA kernels implement BSDF sampling and evaluation.
- The same kernels compile for CPU and GPU.

Engine-Controlled DAG Traversal

- Combiner traversal is centralized in the engine.
- No recursive execution inside combiner plugins.
- Predictable, GPU-friendly evaluation of arbitrary networks.

Atomic MaterialX





Lighting

Rect and Dome
Lights

No UDIMs

Maximum
portability

Cameras

5 Cameras with
depth of field

Materials

Individual .mtlx files
for easy edits

Atomic MaterialX
shader graphs

MaterialX Stirling. Cars © Disney/Pixar



Stirling 2.0

New Tech, New Standards

Portable USD

Drag and Drop in
all 3D apps

Validate mtlx

Check bsdf and
pattern support

Contribute

Share a variant
for your renderer





github.com/leif3d/Stirling

Stirling Private

Watch 0 Fork 1 Star 0

main 1 Branch 0 Tags

Go to file

Add file > Code >

leif3d Update README.md 6d47940 · last week 13 Commits

- geo Add Stirling Project files via Git LFS last week
- images/RenderMan Add Stirling Project files via Git LFS last week
- materials Sync repo: remove unnecessary macos hidden files last week
- textures Sync repo: remove unnecessary macos hidden files last week
- usd Sync repo: remove unnecessary macos hidden files last week
- .gitattributes Add Stirling Project files via Git LFS last week
- .gitignore Add .gitignore to prevent hidden system files last week
- LICENSE.md Update LICENSE.md last week
- README.md Update README.md last week

README License

Stirling

Stirling is an official Pixar asset derived from the Finn McMissile character in Pixar's Cars 2 and designed to showcase MaterialX. The asset was authored using MaterialX 1.39 base building blocks and does not use monolithic materials such as Standard Surface and OpenPBR.

Stirling comes in two openUSD variants:

- Real Time: PNG raster format. Textures use a traditional sRGB color space with a maximum resolution of 2K. Typical in web or game engines.
- RenderMan: TEX mipmapped format. Textures use a OpenColorIO Aces color space with a maximum resolution of 4K. Typical in feature animation and visual effects.

Stirling running in Houdini 22 and RenderMan 28:

Distribution

Non-Commercial, Pixar Asset License

Real Time	Offline
Polys - High Res Mesh	SubD - Low Res Mesh
PNG Raster Textures	TEX Mipmap Textures
sRGB Color Space	OCIO Aces Color Space
2K Max Resolution	4K Max Resolution
Games Web Mobile	Production Rendering

Oh... and some easter eggs!

Showcasing some cool features



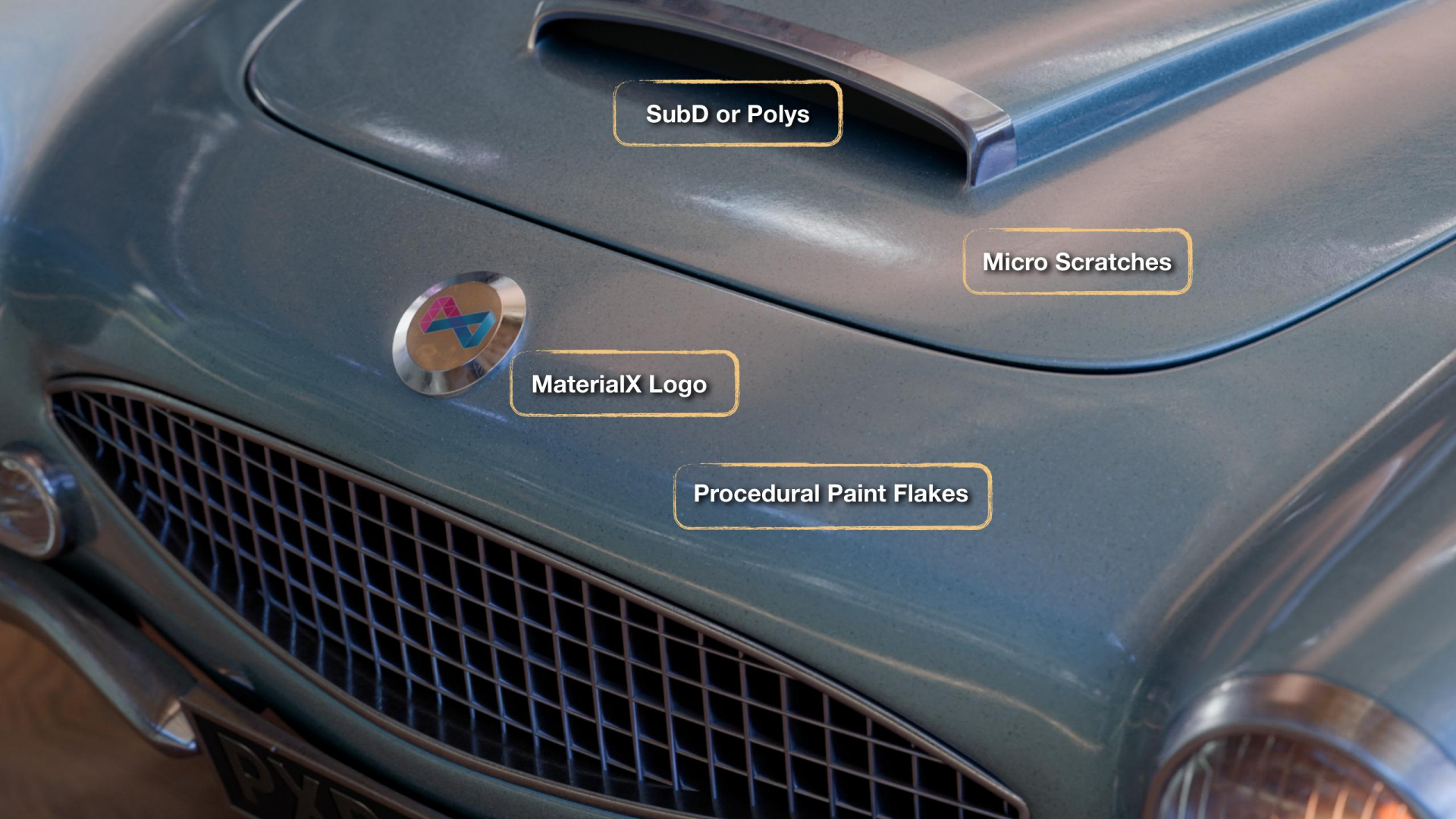
A close-up photograph of the rear of a dark green classic car. The car's body is highly reflective, showing highlights from the environment. On the left, a portion of a large, multi-lens taillight is visible, with red and amber sections. In the center, a black license plate with a silver border displays the text 'PXR A113' in a bold, sans-serif font. Above the license plate is a small, horizontal chrome trim piece. To the right of the license plate, a circular chrome badge features a blue OpenUSD logo. Two small, round red taillights are positioned on either side of the license plate. At the bottom of the frame, two circular exhaust tips are visible. Two yellow-bordered text boxes with black backgrounds are overlaid on the image: one in the upper right pointing to the OpenUSD logo, and one in the lower right pointing to the license plate.

OpenUSD Logo

Tag - A113

Number of the classroom at CalArts where many legendary animators studied



A close-up shot of the front of a teal-colored car. The image focuses on the hood, which features a chrome handle and a circular logo. The car's body is covered in fine, procedural paint flakes, giving it a realistic, weathered appearance. The lighting is soft, highlighting the metallic textures and the fine details of the paint.

SubD or Polys

Micro Scratches

MaterialX Logo

Procedural Paint Flakes





Cornell Box

Legendary test scene

Utah Teapot

Legendary geo primitive

Odometer

Year RiSpec was published





Tires

RenderMan Trivia
Displacement + Bump
Hex Tiling Dirt

Special Thanks

- Fran Gonzalez | Pixar
- Jonathan Stone | Lucasfilm
- Arnon Marcus | OTOY
- ASWF - Academy Software Foundation
- Pixar's RenderMan Team! 🍵

In Depth ASWF Talk

YOUTUBE
[watch?v=CMGajlM3_sQ](https://www.youtube.com/watch?v=CMGajlM3_sQ)





MaterialX Teapot and MaterialX Lion Release

Andrea Weidlich, Principal Researcher | MaterialX and OpenPBR Virtual
Townhall/Sept 15th, 2026



Origins

Designed as research assets for neural materials



Neural Appearance Project

Real-Time Neural Appearance Models

TIZIAN ZELTNER*, NVIDIA, Switzerland
FABRICE ROUSSELLE*, NVIDIA, Switzerland
ANDREA WEIDLICH*, NVIDIA, Canada
PETRIK CLARBERG*, NVIDIA, Sweden
JAN NOVAK*, NVIDIA, Czech Republic
BENEDIKT BITTERLI*, NVIDIA, USA
ALEX EVANS, NVIDIA, United Kingdom
TOMÁŠ DAVIDOVIČ, NVIDIA, Czech Republic
SIMON KALLWEIT, NVIDIA, Switzerland
AARON LEFHOFF, NVIDIA, USA



Fig. 1. Close-up renderings of a Tesseract with our neural BDEF. Our model learns the intricate details and complex multi-layered material behavior of the scene. Algorithms, materials, and data are responsible for the realism of the object while being faster to evaluate than traditional non-neural models of similar complexity. The system we present allows us to include such high-fidelity objects in real-time renderers in a scalable way.

We present a complete system for real-time rendering of scenes with complex appearance previously reserved for offline use. This is achieved with a combination of algorithmic and system-level innovations.

Our appearance model utilizes learned hierarchical features that are interpreted using neural decoders, which produce reflectance values and importance-sampled directions. To best utilize the modeling capacity of the decoders, we equip the decoders with two grapheme priors. The first prior—transformations of directions into learned shading features—facilitates accurate reconstruction of anisotropic effects. The second prior—a stochastic sampling distribution—allows the neural decoder to perform importance sampling efficiently. The resulting appearance model supports anisotropic sampling and level-of-detail rendering, and allows highly detailed layered material graphs into a compact unified neural representation.

*Equal contribution. Order determined by a rock-paper-scissors tournament.

© 2024 Copyright held by the author(s).

This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive version of the work was published in ACM Transactions on Graphics, <https://doi.org/10.1145/3691077>.

By exposing hardware accelerated tensor operations to ray tracing shaders, we show that it is possible to utilize and exercise the neural decoders efficiently inside a real-time path tracer. We analyze scalability with increasing number of neural materials and propose to improve performance using code optimized for coherent and divergent execution. Our neural material shaders can be over an order of magnitude faster than non-neural layered materials. This opens up the door for using like-quality visuals in real-time applications such as games and live previews.

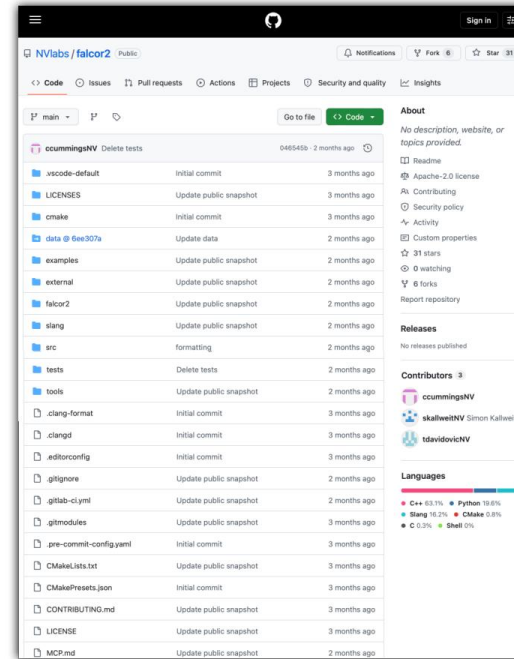
CCS Concepts • Computing methodologies • Reflection modeling.

Additional Key Words and Phrases: appearance models, neural networks, real-time rendering.

ACM Reference Format:

Tizian Zeltner, Fabrice Roussele, Andrea Weidlich, Petrik Clarberg, Jan Novak, Benedikt Bitterli, Alex Evans, Tomáš Davidovič, Simon Kallweit, and Aaron Lefhoff. 2024. Real-Time Neural Appearance Models. *ACM Trans. Graph.* 43, 3, Article 33 (June 2024), 17 pages. <https://doi.org/10.1145/3691077>

ACM Trans. Graph., Vol. 43, No. 3, Article 33. Publication date: June 2024.



Falcor 2

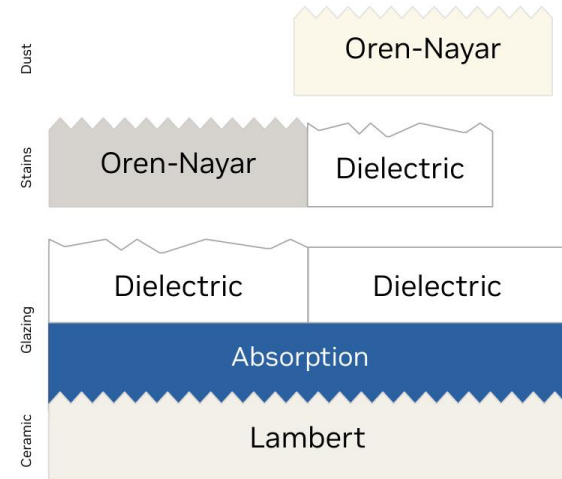
Technical Specification

- USD assets with .mtlx attached
- MaterialX 1.38
- Authored in LookdevX
- Minimum resolution of 4k per texture + UDIMs
- Approximately 1 GB per asset
- Assembled from individual BSDFs
 - BSDFs combined with layer and mix operators
 - Textures in sRGB
 - No proprietary nodes
- Download: [github.com/
DigitalProductionExampleLibrary/
MaterialXTeapotLion](https://github.com/DigitalProductionExampleLibrary/MaterialXTeapotLion)



MaterialX Teapot Asset

84 4k textures on body (14 UDIMS)



MaterialX Teapot Asset

Eight 8k + one 4k textures on handle



Grease

Conductor

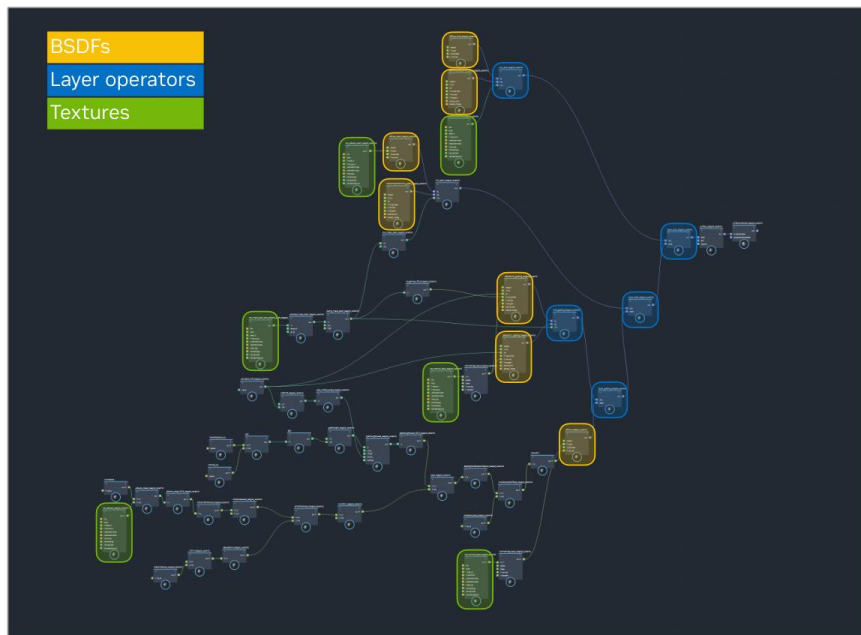
Dirt

Oren-Nayar

Metal

Conductor

Shadergraph Teapot



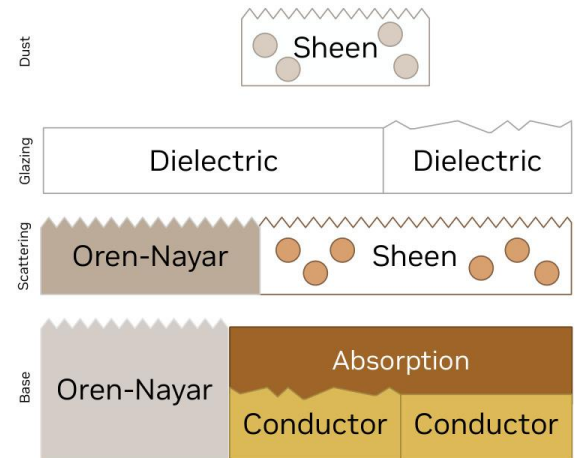
Ceramic



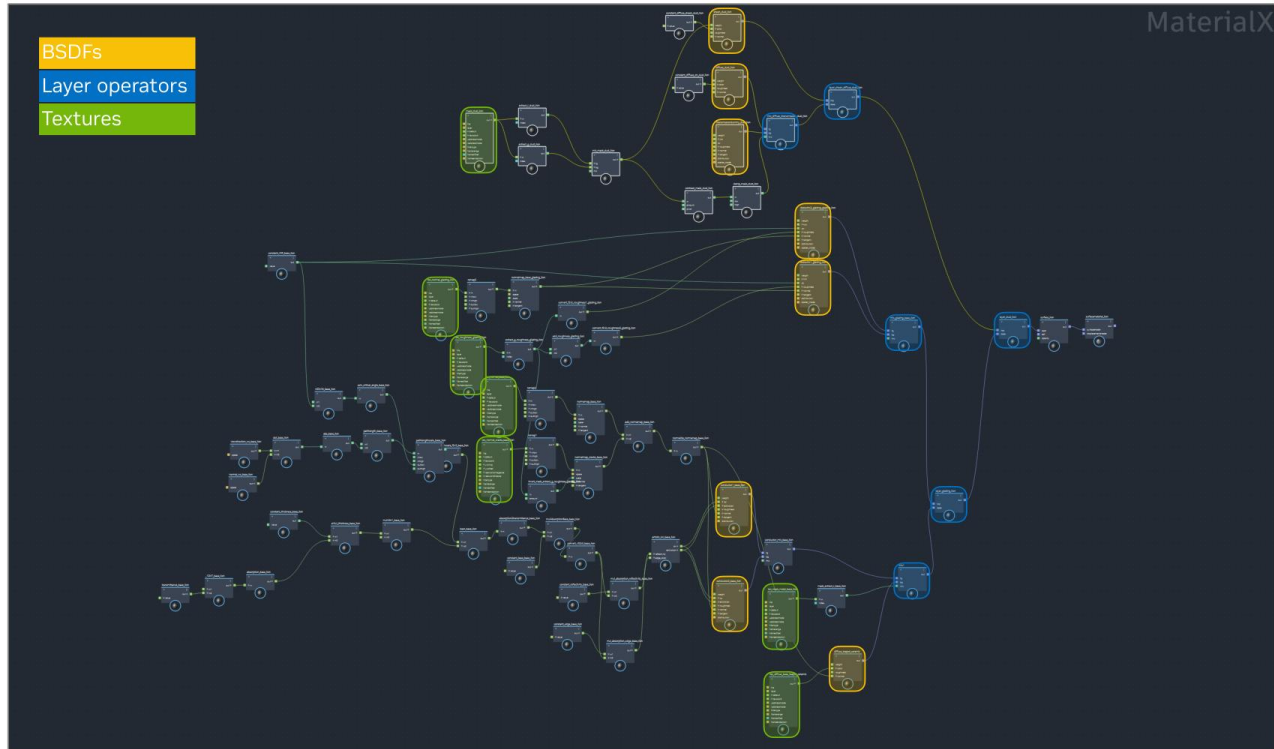
Handle

MaterialX Lion Asset

42 4k textures (6 UDIMS)



Shadergraph Lion



Known Issues

- MaterialX UDIM issue with Storm
 - Maya 2026/2027 Viewport, Hydra GL: No UDIM support
- Arnold: Layering issue (fixed in an upcoming version)
- Houdini 19/21 Karma (XPU): Arbitrary nodes not fully supported

- Confirmed working
 - Houdini 22 Karma CPU
 - Falcor 2: <https://github.com/NVlabs/falcor2>
 - MaterialX Playground
 - Octane OTOY

Upcoming MaterialX library from NVIDIA

2982 MaterialX OpenPBR materials

The screenshot displays the NVIDIA MaterialX library interface. At the top, navigation tabs include PHYSICAL, PHYSICAL AI SIMREADY MATERIALS, GALLERY, EXTENSIONS, HOW TO USE, and LICENSING. A status bar in the top right corner shows '2965 rendered', '0 failed', and '2982 total' as of '2026-08-16 09:36'. Below the navigation, a header indicates '1361 materials' and provides options to 'SELECT ALL IN VIEW', 'CLEAR SELECTION', and 'SORT' by 'Category'. A 'Library update' section on the left lists: 'Releasing a new update to our PhysicalAi-SimReady Material Library', 'The new library will release on Hugging Face', and 'Include 1.39.3, 1.39.4 and 1.39.5 compatible materials.' The main area features a grid of material thumbnails. Two thumbnails are enlarged in the center: 'LOOK - SHADER BALL' (a wooden sphere with a white core) and 'PHYSICAL SCALE - 1 M PLANE' (a wood tile pattern). To the right of these is a detailed panel for 'Wood Tiles_Walnut_Herringbone'. This panel includes a 'DOWNLOAD THIS MATERIAL' button, a 'KEY' field with the material name, 'MDL SOURCE', 'ASSET' (Wood), 'RENDER' statistics (mean luma 0.4666 - 4.4s), 'SIMREADY' version (MaterialX 1.39.3 - OpenPBR 1.1), 'ATTRIBUTES' (Textured, Anisotropic, Normal Map), 'SIMREADY TAGS' (wood, walnut, master, tiles, tied, pattern, decorative, multimaterial, organic, natural, coated, infinite tiling, herringbone, architecture, interior), and 'NON-VISUAL' settings (BASE: wood, COATING: none, ATTR: none). At the bottom, a row of material thumbnails is labeled: Ruby, Sapphire, Tanzanite, Topaz, Tourmaline, Turquoise, and Turquoise_Apple_Green. A footer note states: 'Gallery + pop card are MaterialX / OpenPBR renders from the shipping catalog - not MDL-direct.'

Acknowledgements

- Alex Liu, Davide Di Giannantonio Potente, Zhelong Xu
- Tomáš Davidovič, Eric Enderton, Aaron Lefohn, Gabriele Leone, Kevin Margo,
- Matthew Low, Jonathan Stone and the entire DPEL TSC





MaterialX and OpenPBR in Houdini 22

Chris Rydalch | SideFX

Houdini  TM



Houdini 22 | Overview & Library

- ▶ MaterialX included since Houdini 19.0
- ▶ Solaris takes a “MaterialX-as-Spec” approach (i.e. no code-gen)
- ▶ Focused on USD-encoded materials
- ▶ Super-set of Karma nodes add functionality requested by artists
- ▶ MtlX Standard Surface has become the default material in Houdini





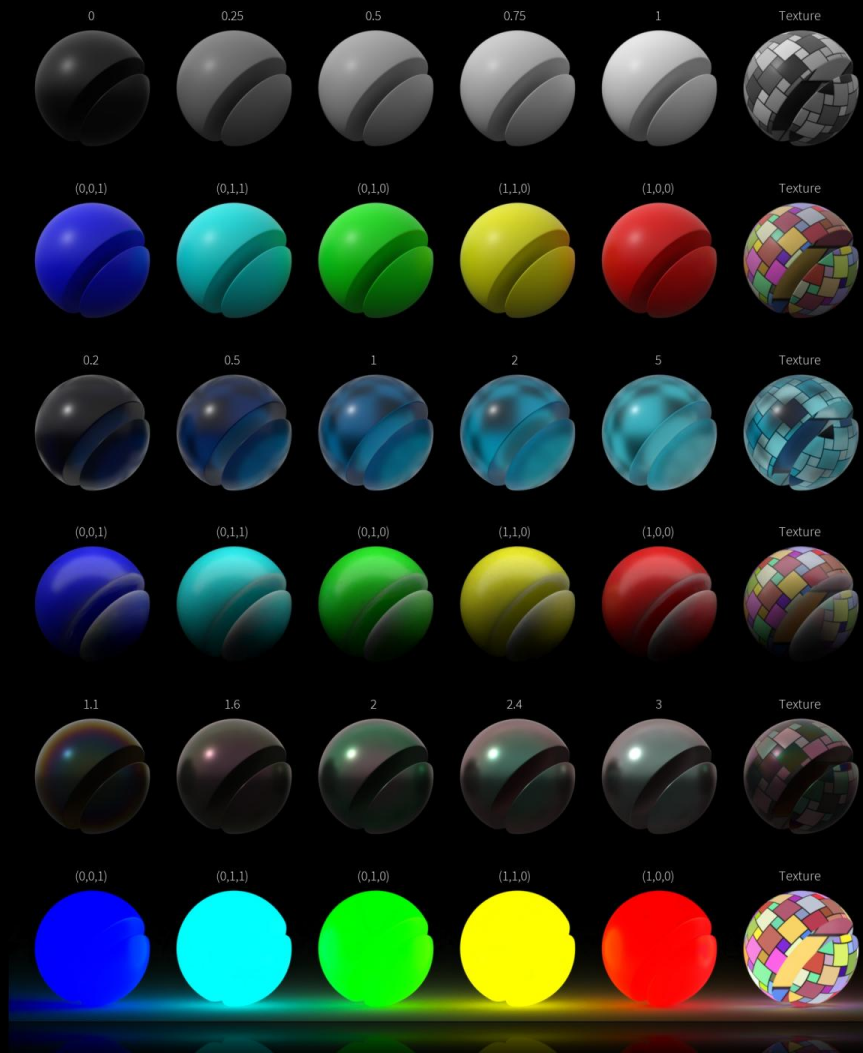
Houdini 22 | Overview & Library

- Houdini
 - H22 released July 15, 2026
 - H21 released August 26, 2025
- USD
 - 24.03 → 25.05 → 26.06
- MaterialX
 - 1.38.10 → 1.39.3 → 1.39.5

	USD	MtIX	VFX
H22	26.05	1.39.5	CY2026
H21	25.05	1.39.3	CY2025
H20.5	24.03	1.38.10	CY2024
H20	23.08	1.38.8	CY2023



- ▶ Introduced in Houdini 21
- ▶ Utilizes the implementation included with MaterialX 1.39.5
 - ▶ H22 uses v1.1.1
- ▶ Support added to KarmaCPU, KarmaXPU, and HoudiniVK



- Flakes
- GeomPropValueUniform
- Karma Inputs for Texture Maps





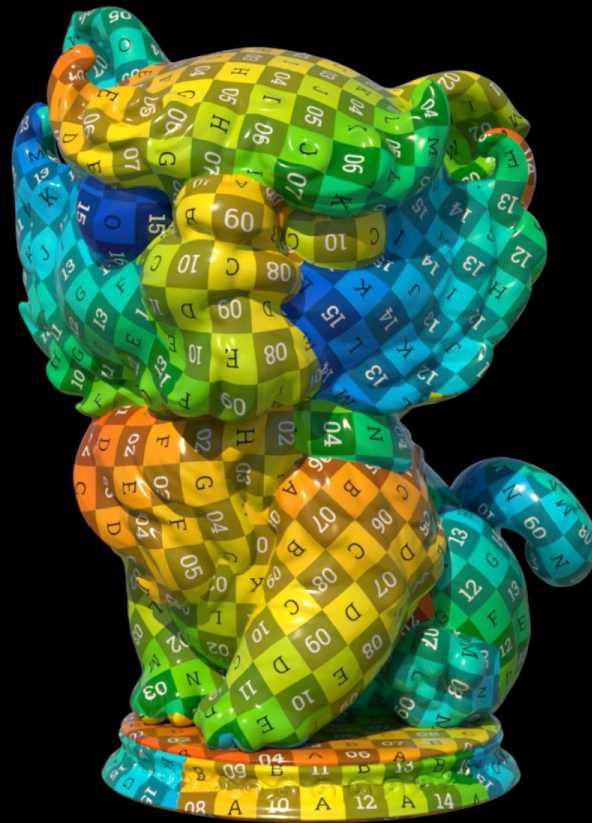
Houdini 22 | Overview & Library

- Flake Shaders
- GeomPropValueUniform
- Karma Inputs for Texture Maps



Houdini 22 | Overview & Library

- Flake Shaders
- GeomPropValueUniform
- Karma Inputs for Texture Maps

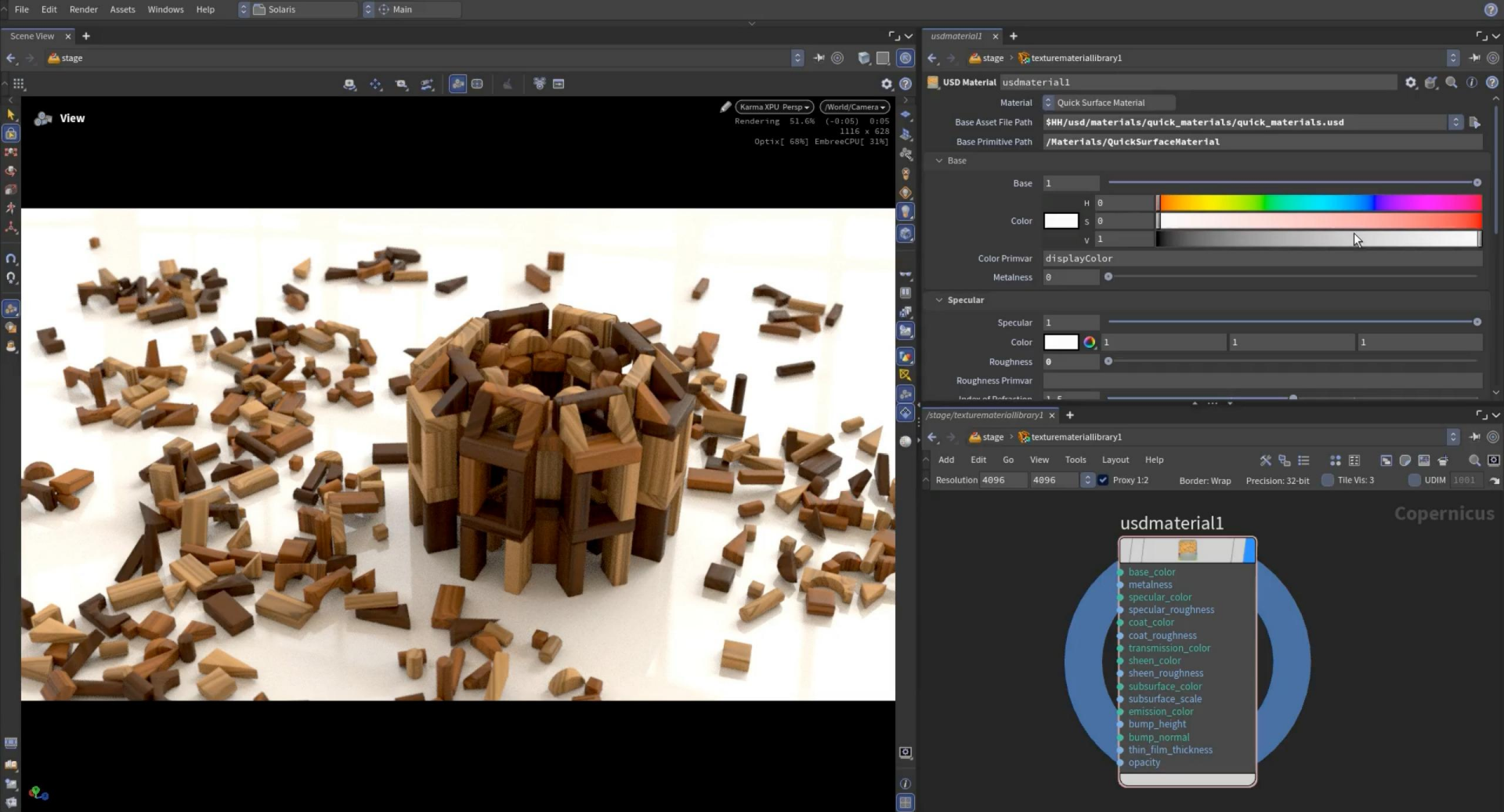




Houdini 22 | Quick Materials

- ▶ Introduced in Houdini 20.5
- ▶ Currently uses MtlX Standard Surface
- ▶ Core part of Texture Material Library workflows new in H22
- ▶ Improved Quick Material tooling and OpenPBR Surface flavor are in-progress





THANK YOU!

MaterialX Playground

An open-source project that aims to make MaterialX easier to use everywhere: open, view, build and learn MaterialX in your browser, your IDE, your desktop, or any HTML page.

JOÃO VÍTOR BUSCATTO SILVA · MATERIALX TOWN HALL 2026 · SEPTEMBER 15



What is the Playground?



Open it anywhere

Browser, IDE, desktop, or one tag on any HTML page. Nothing to install (unless you want to!).



For Newcomers

An intro to MaterialX, a gallery to explore materials, and (soon) tutorials.



True to the library

Rendering runs on the official MaterialX WASM bindings.

Four main tools, one engine



Viewer

Drop in a .mtlx or a zip with textures, or pick a preset. Your own HDR or EXR lighting and shader-ball scenes.

NEW: Load custom geometry, .obj, .glb or .gltf

NEW: Studio backdrop with shadows and environment-driven light.



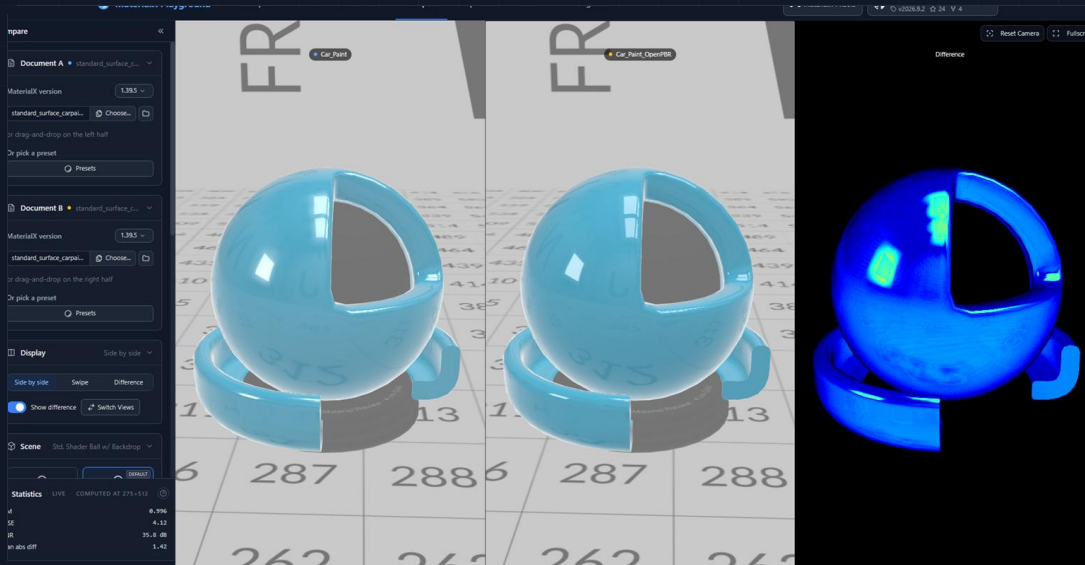
Four main tools, one engine



Compare

Two materials side by side or under a slider, synced cameras, per-pixel difference, live SSIM, RMSE and PSNR

NEW: Comparison across different MaterialX versions!



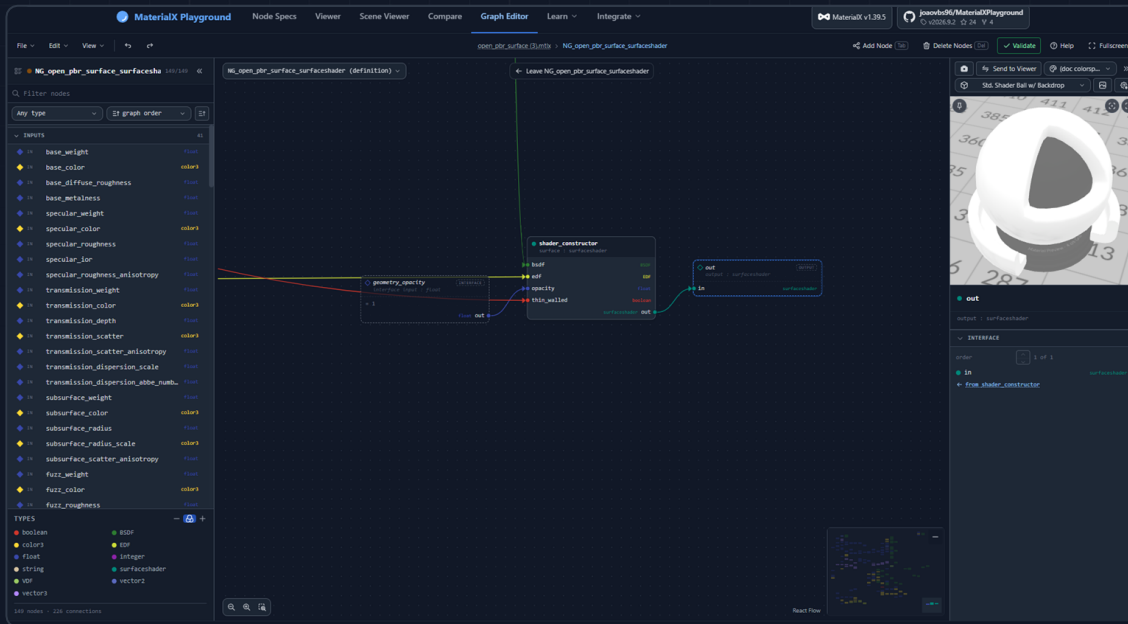
Four main tools, one engine



Graph Editor

Build spec-conformant graphs visually: nested nodegraphs, validation, autosave, session restore and .mtlx export.

NEW: Node Def/Node Graph creation/editing/viewing



Four main tools, one engine



Node Docs

Every standard node, documented from the spec, easily searchable, with a live interactive preview & color coded tables.

The screenshot displays the MaterialX Playground interface. On the left, the 'NODE LIBRARY' panel shows a search bar and a list of nodes under 'BXDF/TRANSLATION'. The 'open_pbr_surface_to_standard_surface' node is selected. The main area shows a 3D preview of a yellow sphere with a standard surface material. To the right, the 'Parameters' panel for this node is visible, showing various sliders and color pickers for properties like 'subsurface_weight', 'subsurface_color', 'subsurface_radius', 'subsurface_radius_scale', 'subsurface_scatter_anisotropy', 'fuzz_weight', 'fuzz_color', 'fuzz_roughness', 'coat_weight', 'coat_color', 'coat_roughness', and 'coat_roughness_anisotropy'. Below the parameters, a table lists the ports and their default values.

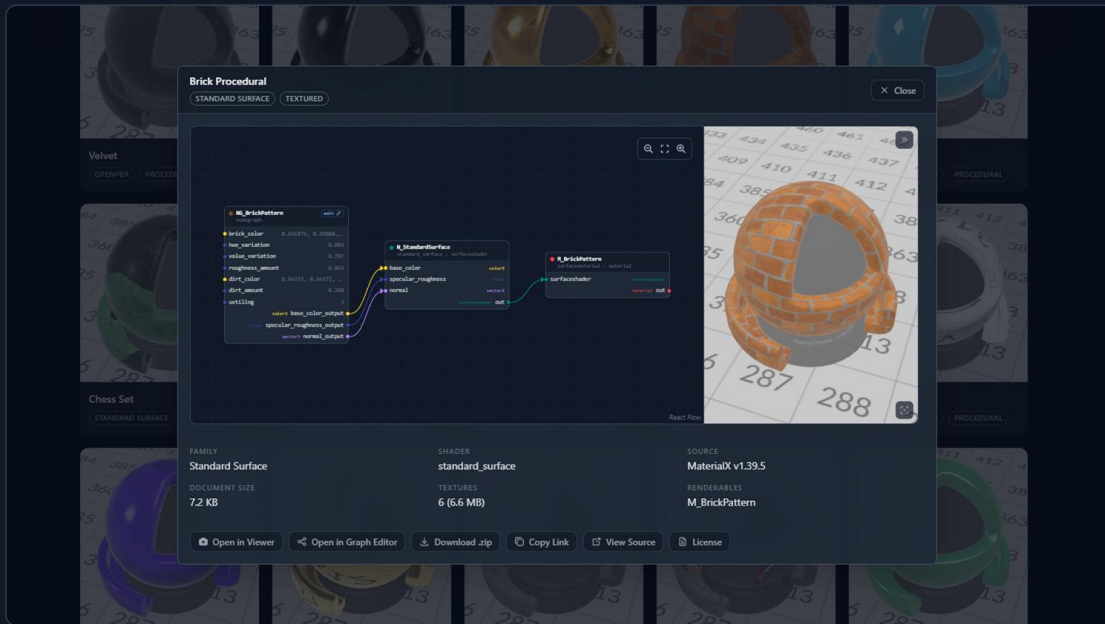
PORT	DESCRIPTION	TYPE	DEFAULT
base_weight		float	1.0
base_color		color3	0.8, 0.8, 0.8
base_diffuse_roughness		float	0.0

Explore example Materials



Material Gallery

- Browse the examples and showcase materials with search and filters.
- One click to a zip with textures.
- License, source and permalink per material.



USD Scene Loading



Open a USD stage

- Drop a folder with .usda, .usdc or .usdz plus its textures
- OpenUSD runs in the browser through WebAssembly
- Supports authored cameras
- Turntable GIF export



MaterialX on full scenes

- Referenced .mtlx materials render through the same engine as the Viewer
- Supports JPEG, PNG, EXR, HDR, TIFF, WebP and KTX2 textures
- UDIM Textures



MATERIALX TEAPOT

EXPERIMENTAL

USD Scene Loading



MATERIALX LION



MATERIALX STIRLING

USD Scene Loading



USD Scene Loading



Browser, IDE, desktop, any page



VS Code Extension

- Edit .mtlx with live preview, validation and hover docs.
- The same WASM engine, running locally in your editor.
- Installs from a .vsix on GitHub Releases.

EXPERIMENTAL



Desktop App

- An Electron shell around the same code base.
- Native menus and file access for pipeline folders.
- No browser limits on memory and file access.

EXPERIMENTAL

A `<model-viewer>` for MaterialX

```
1 <!doctype html>
2 <html>
3   <head>
4     <title>Hello MaterialX!</title>
5   </head>
6   <body>
7     <script src="https://joaovbs96.github.io/MaterialXPlayground/embed/mtlx-viewer.js"></script>
8     <materialx-viewer
9       src="https://raw.githubusercontent.com/AcademySoftwareFoundation/MaterialX/refs/heads/main/resources/MaterialX/Examples/OpenPbr/open_pbr_carpaint.mtlx">
10   </materialx-viewer>
11 </body>
12 </html>
```



Same engine, same fidelity

The official MaterialX WASM bindings, exactly as on the full site.

No build step, no framework

Docs, blogs, asset stores, internal wikis: paste and publish.

Focus on Performance

Lazy-loads and caps how many viewers run at once, so the page stays smooth.

Embed Builder

Writes the snippet for you on the site, with a live preview.

AVAILABLE NOW

Where this is going



Shipped

Material Gallery, custom geometry, color management, switchable 1.39.4 / 1.39.5 builds



In progress

Full release of Desktop app for Windows, macOS and Linux. Releasing VSCode extension on store. USD Scene Viewer.



Next

Learn: interactive tutorials served alongside the app. A graph-preview embed. Support for glTF files in the scene viewer. Support for ShadingLanguageX.



Exploring

Your suggestions: tell us what would help you most!

THANK YOU / QUESTIONS



Try it in your browser

Issues, ideas and pull requests are very welcome.

Open the Playground

View the source



joaovbs96.github.io/MaterialXPlayground



github.com/joaovbs96/MaterialXPlayground



[VS Code extension](#)



SCAN TO OPEN



MaterialX & OpenPBR
Octane Render®
Arnon Marcus



2026



Native Pattern Nodes Import (mtlx & USD)



MaterialX 1.39.4



2027

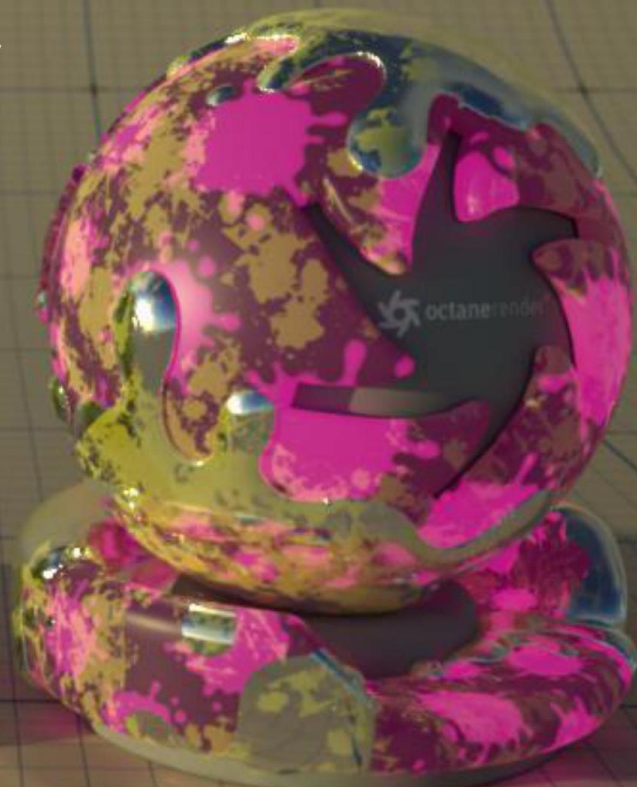
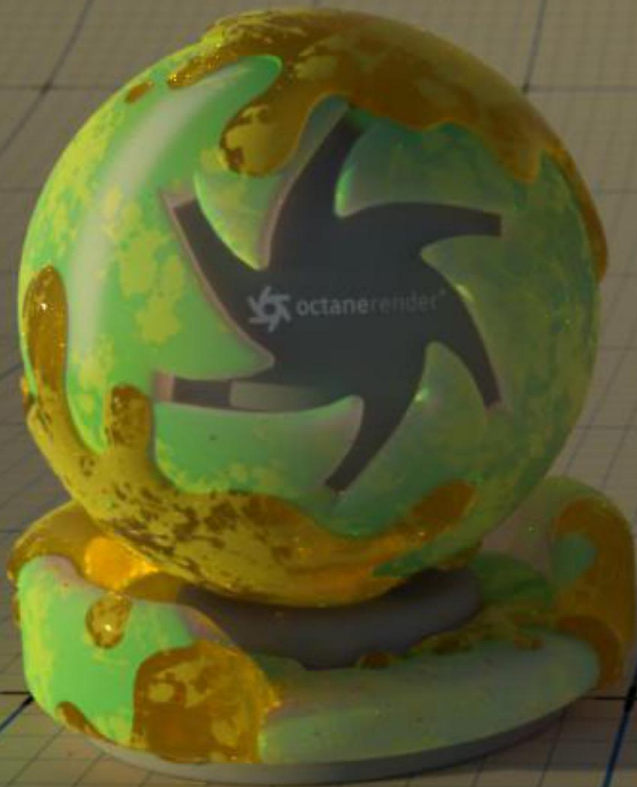


PBR Nodes & Flakes



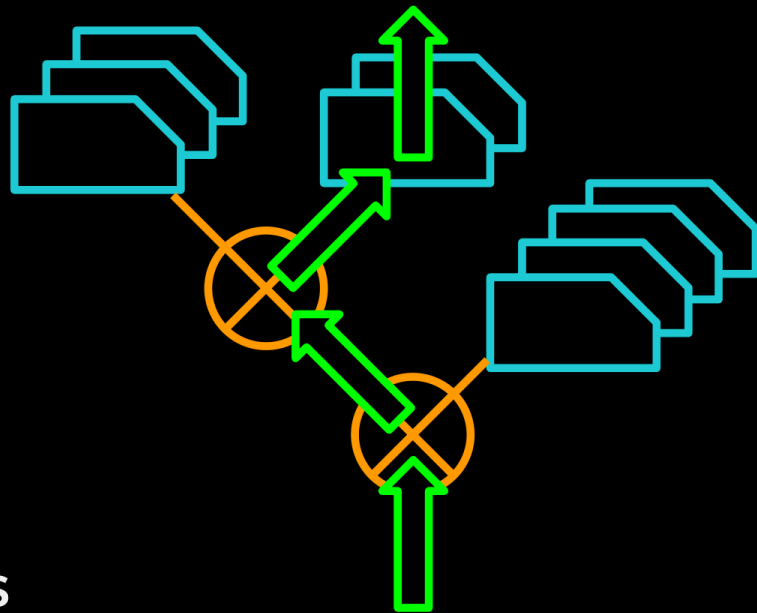
MaterialX 1.39.6

Multiply
Layer
Add
Mix

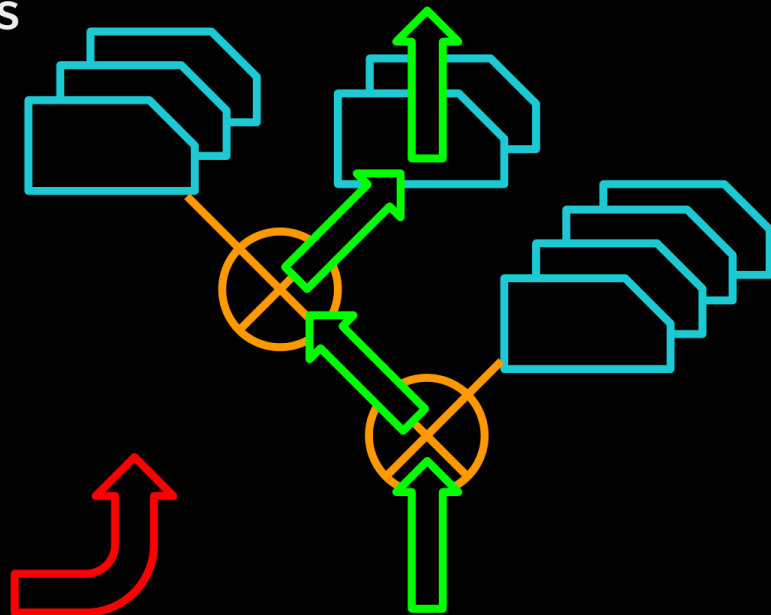
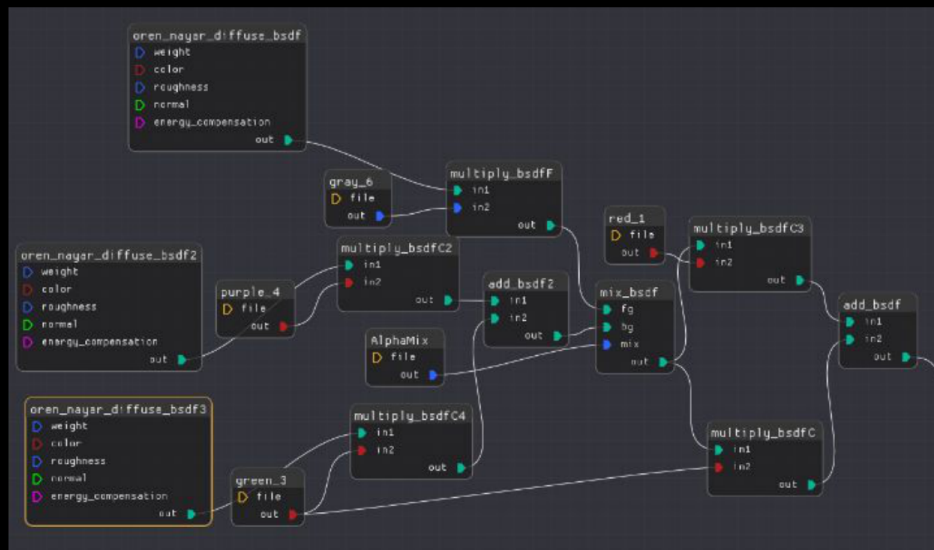


Performance > Flexibility :

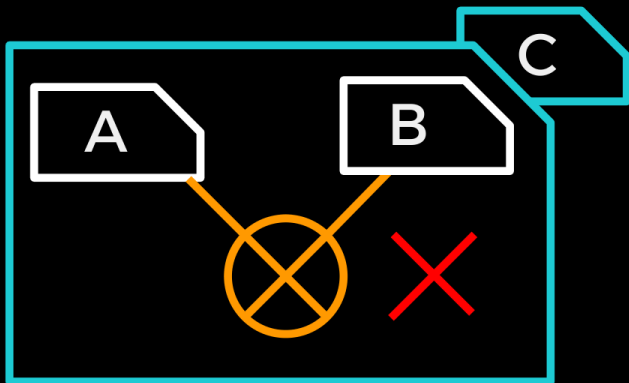
- No OSL Closures
- No *general* graph traversal
- Can *mix* entier materials
- Can *stack* layers linearly
- Can not *mix* layers
- Can not *stack* material mixes
- Can not *add* or *multiply* layers



BSDF DAG -> Mix Tree of BSDF stacks

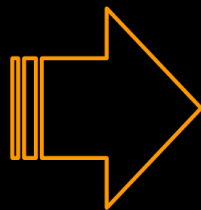
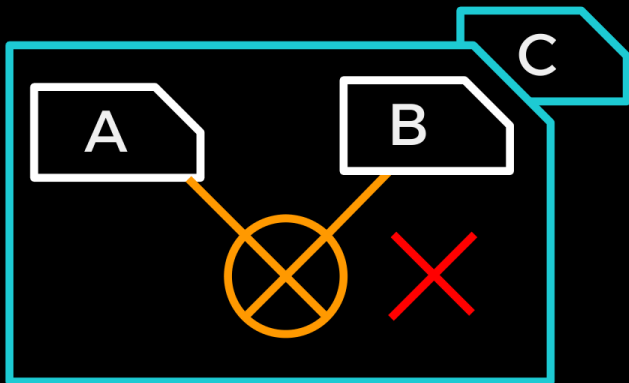


Layer(Mix(A, B, t), C)

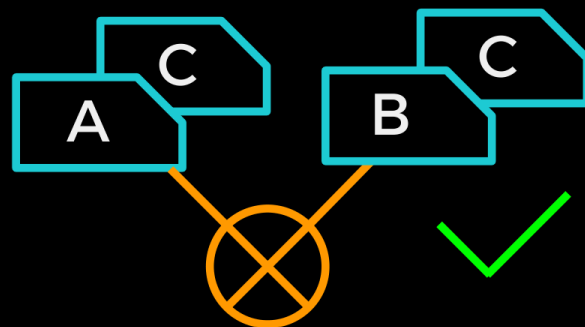


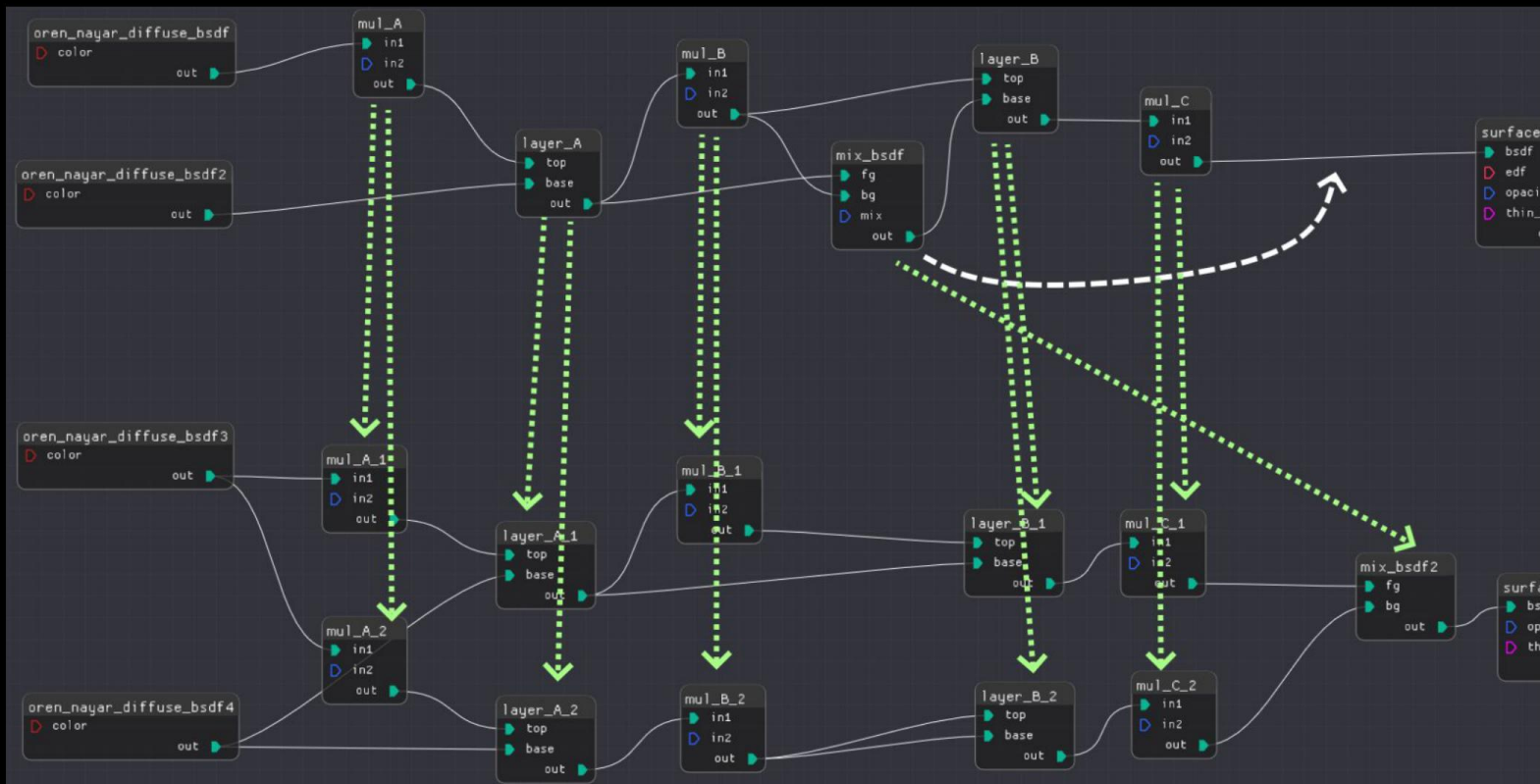
Mixing and Layering are associative.

Layer(Mix(A, B, t), C)



Mix(Layer(A, C),
Layer(B, C), t)





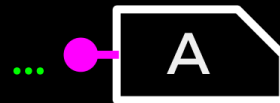
Mul(A, 3)



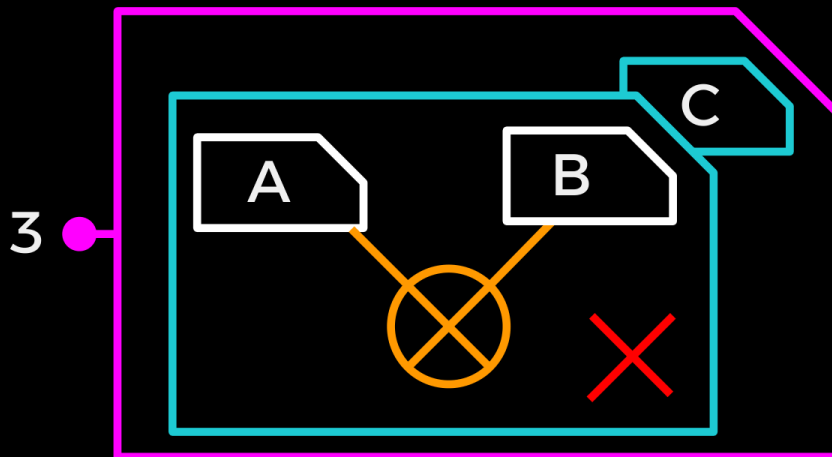
Mul(A, )



Mul(A, ...)

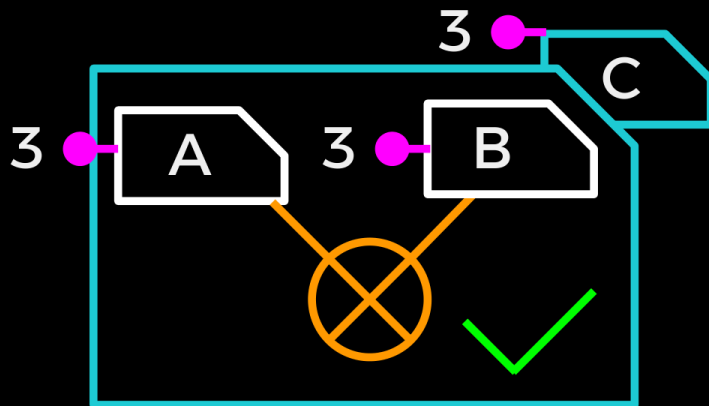


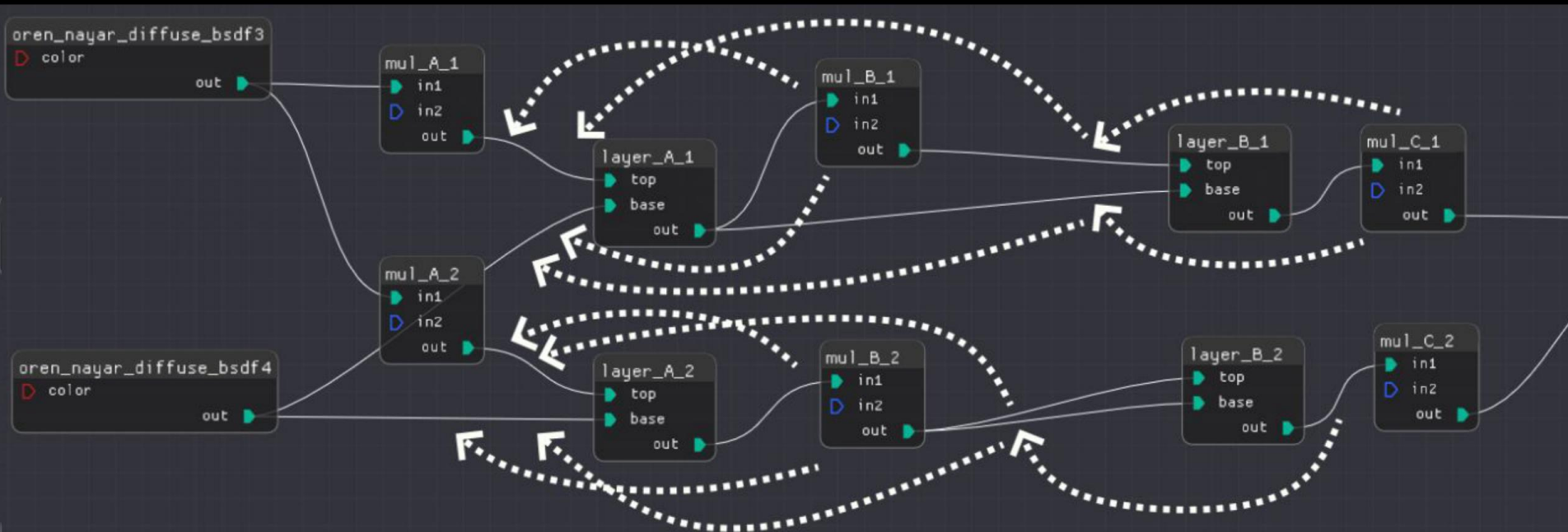
$\text{Mul}(\text{Layer}(\text{Mix}(A, B, t), C), 3)$

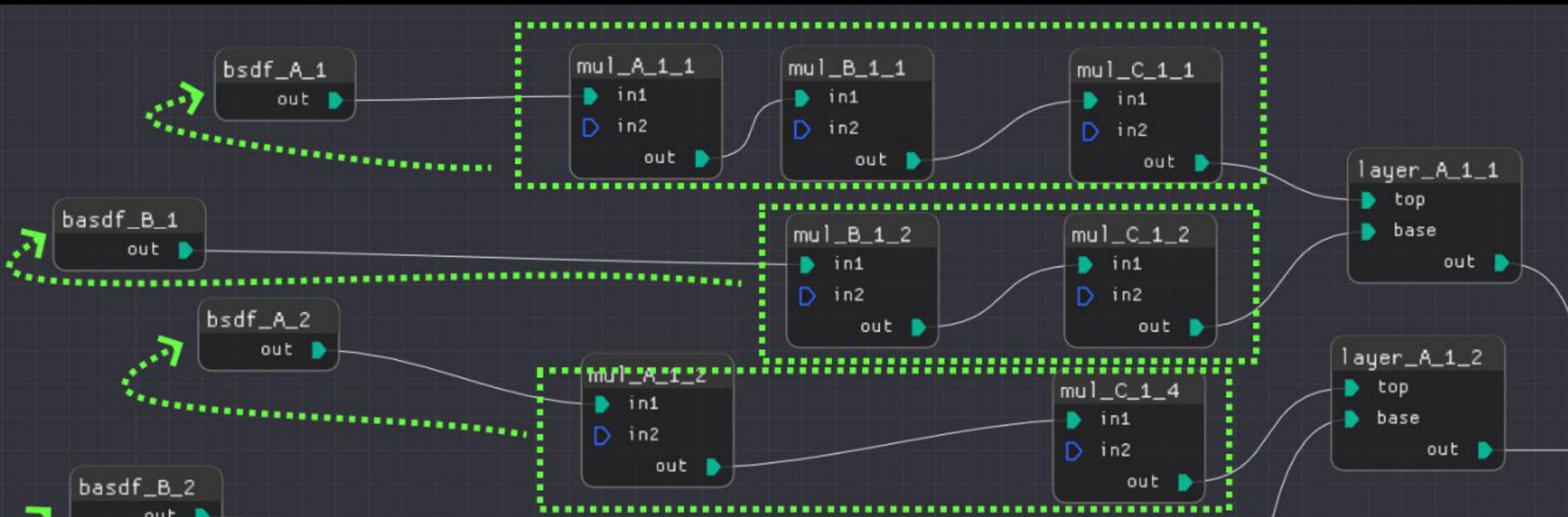


Multiplication is distributive with Mixing and Layering

Layer(Mix(Mul(A, 3), Mul(B, 3), t), Mul(C, 3))









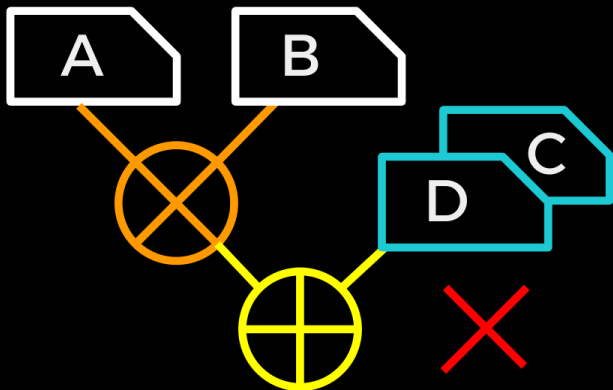
2027



BSDF Add

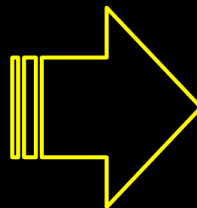
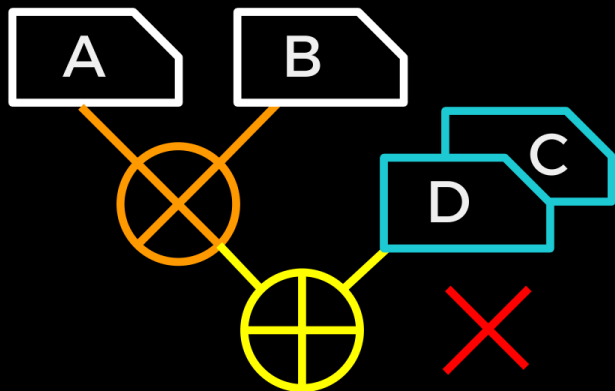


Add(Mix(A, B, t),
Layer(D, C))

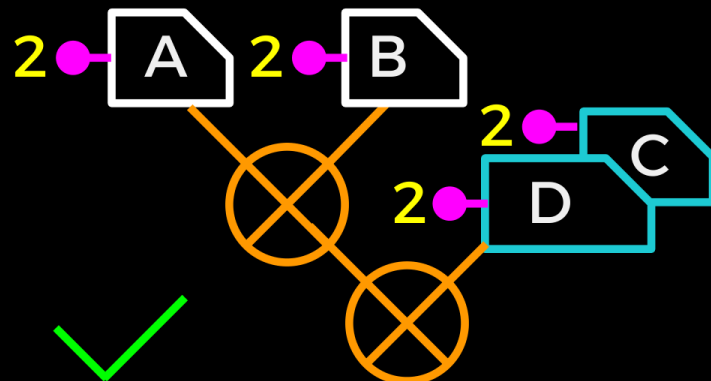


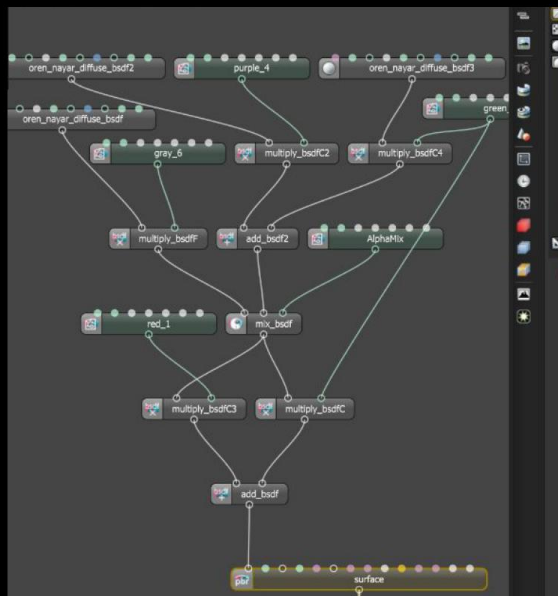
Add is a 50:50 Mix of the branches Multiplied by 2

Add(Mix(A, B, t),
Layer(D, C))



Mix(Mix(2A, 2B, t),
Layer(2D, 2C), 1/2)





- No OSL Closures
- Native Spectral Layers
- Coded to the Spec.
- Optimized for Octane
- Diligently Compared
- Closely Matching



MaterialX PBS Library

MaterialX includes a library of types and nodes for creating physically plausible materials and lights as described above. This section outlines the content of that library.

Data Types

- BSDF: Data type representing a Bidirectional Scattering Distribution Function.
- EDF: Data type representing an Emission Distribution Function.
- VDF: Data type representing a Volume Distribution Function.

The PBS nodes also make use of the following standard MaterialX types:

- surfaceshader: Data type representing a surface shader.
- lightshader: Data type representing a light shader.
- volumeshader: Data type representing a volume shader.
- displacementshader: Data type representing a displacement shader.

BSDF Nodes

Conductor Fresnel Equations

The reflectance F is the average of the s-polarized (R_s) polarized (R_p) Fresnel reflectances at a conductor interface

$$a^2 + b^2 = \sqrt{(\eta^2 - \kappa^2 - \sin^2\theta)^2 + 4\eta^2\kappa^2}$$

$$a = \sqrt{\max\left(0, \frac{1}{2}(a^2 + b^2 + \eta^2 - \kappa^2 - \sin^2\theta)\right)}$$

$$R_s = \frac{a^2 + b^2 - 2a \cdot \cos\theta + \cos^2\theta}{a^2 + b^2 + 2a \cdot \cos\theta + \cos^2\theta}$$

$$R_p = R_s \cdot \frac{\cos^2\theta(a^2 + b^2) - 2a \cdot \cos\theta \sin^2\theta + \sin^4\theta}{\cos^2\theta(a^2 + b^2) + 2a \cdot \cos\theta \sin^2\theta + \sin^4\theta}$$

$$F = \frac{1}{2}(R_s + R_p)$$

Generalized Schlick Equations

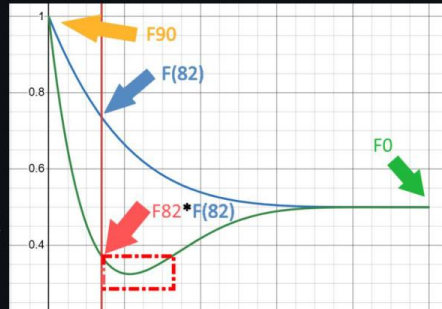
The Hoffman^[13] generalization of the Schlick Fresnel curve adds a controllable dip in the reflectance at

$\theta_{\max} = \arccos(1/7) \approx 81.79^\circ$, whose depth is set by the `color82` multiplier t :

$$\cos\theta_{\max} = \frac{1}{7}$$

$$a = \frac{[r_0 + (r_{90} - r_0)(1 - \cos\theta_{\max})^q](1 - t)}{\cos\theta_{\max}(1 - \cos\theta_{\max})^6}$$

$$F_\theta = r_0 + (r_{90} - r_0)(1 - \cos\theta)^q - a \cdot \cos\theta(1 - \cos\theta)^6$$



Oren Nayar



Oren Nayar (Energy Preserving)



Burley

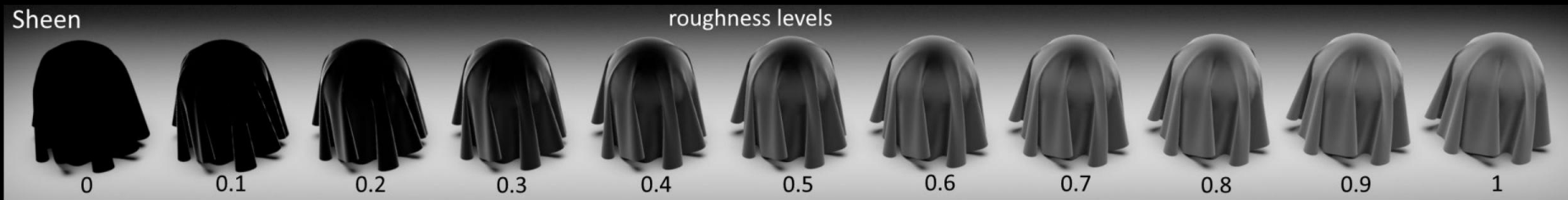


Single Scattering

Sheen

Conty Kulla

roughness levels

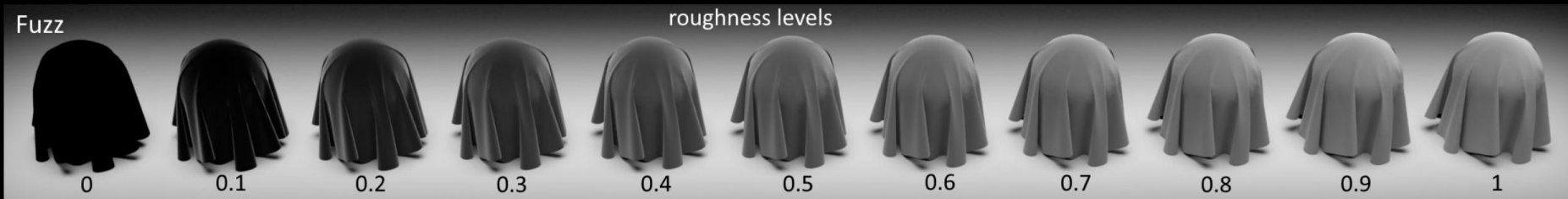


Multi Scattering

Fuzz

Zeltner

roughness levels



Translucent



High Saturation

Translucent



Medium Saturation

Translucent



Low Saturation



2027



Subsurface BSDF



Subsurface



Color

Subsurface



Radius

Subsurface



Anisotropy

-1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0

Tangent

Artistic IOR

Complex IOR

Dual Roughness



Thin Film

Retroreflective



Silver



Copper



Brass



Lead

Tangent

IOR

Scatter Modes

Dual Roughness



Thin Film

Retroreflective



Tangent

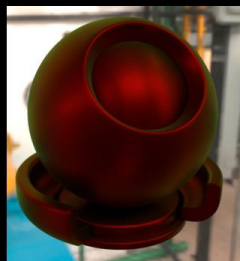
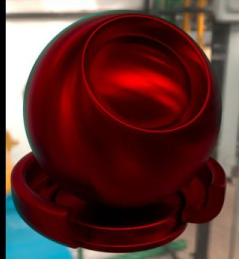
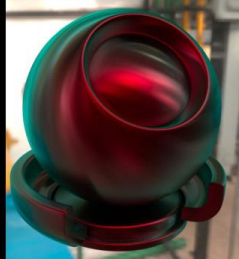
Exponent

Scatter Modes



Thin Film

Retroreflective





VDF Nodes



VDF

Absorption



VDF

Scattering

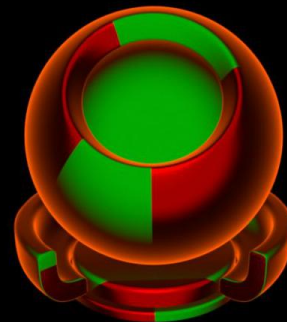
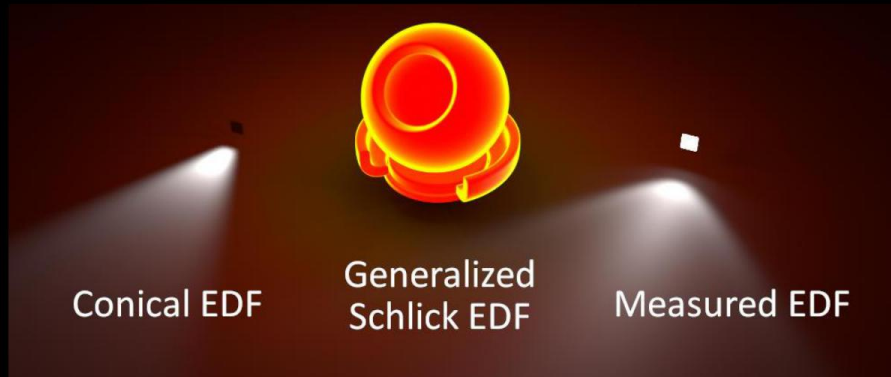


VDF

Anisotropy



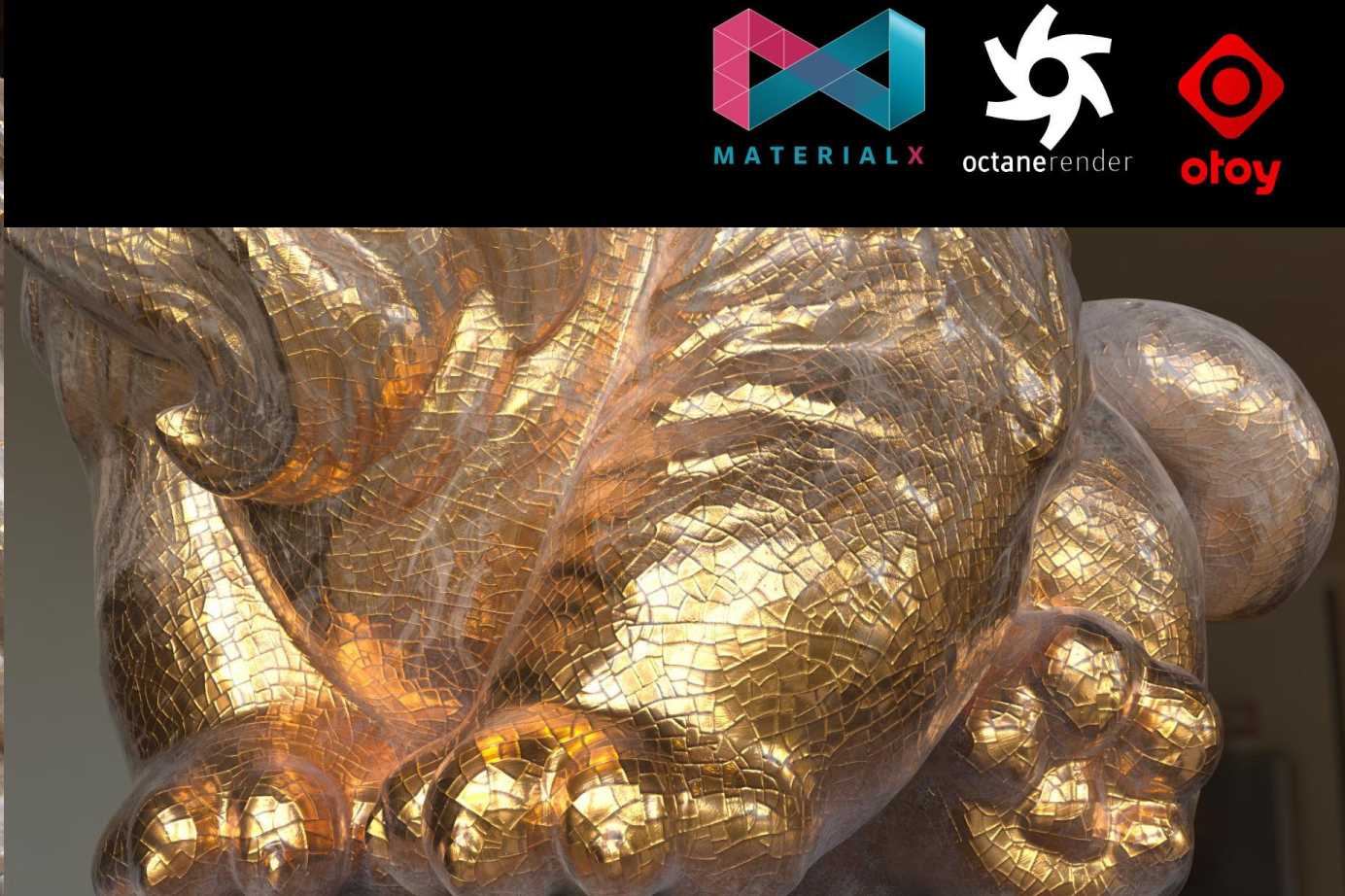
- No OSL Closures
- Native Pattern Nodes
- Surface (EDF) : Emission
- EDF : Color (Emission)
- Combiners : Color Math





2027

PBR Nodes



MATERIALX



octane**render**



otoy

2027

PBR Nodes

Real Time



2027

PBR Nodes





2026



Oren Nayar (EON)



Oren Nayar

Energy Preserving

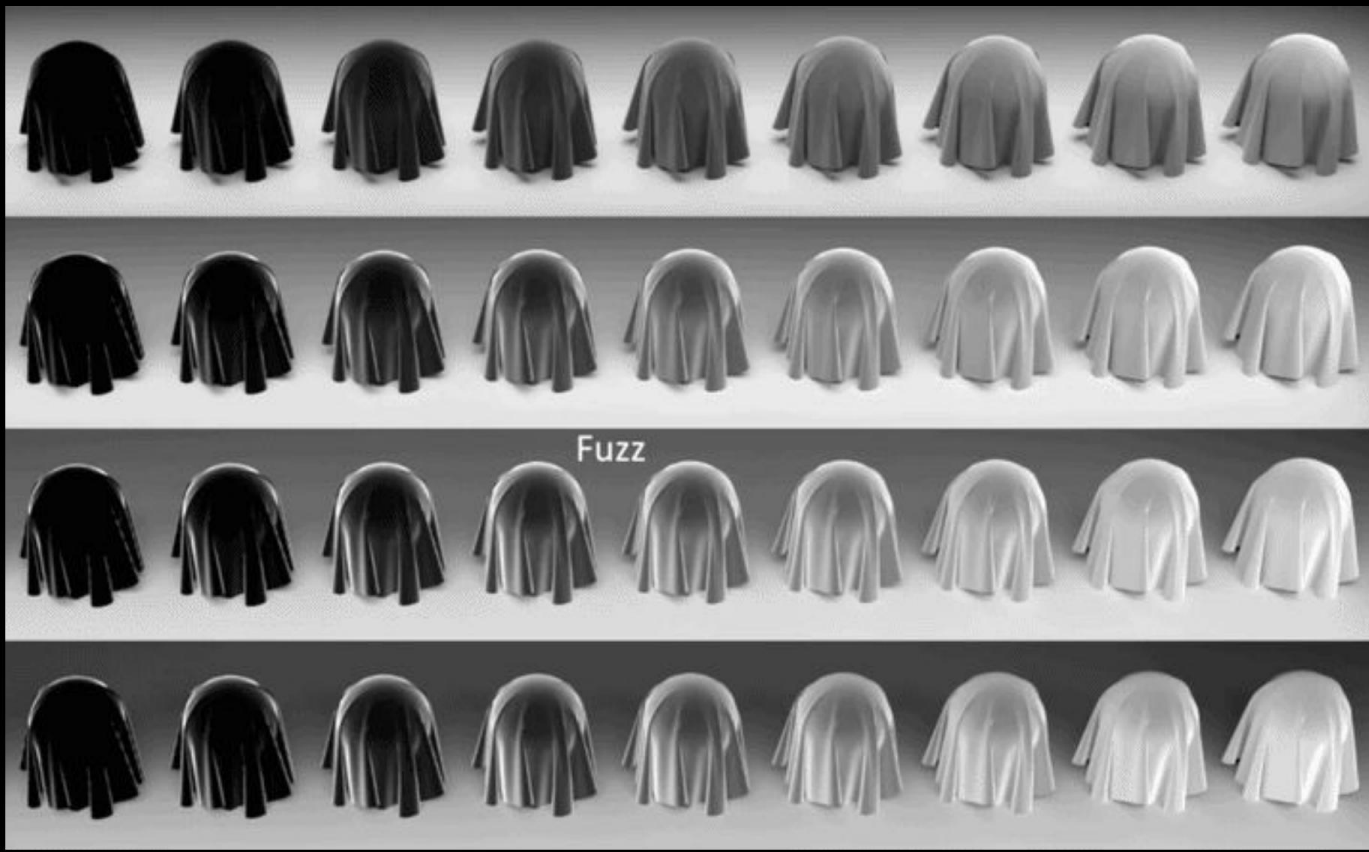




2026



Fuzz





2026



Coat Darkening





DefaultMaterial:OpenPBR surface material OpenPBR surface material

- ▶ Base Layer
 - ▼ Specular Layer
 - ▶ Specular weight: 1.000
 - ▶ Specular color: 
 - ▶ Specular roughness: 0.000
 - ▶ Specular haze: 0.000
 - ▶ Specular haze spread: 0.250
 - ▶ Specular IOR: 1.500
 - ▶ Specular anisotropy: 0.000
 - ▶ Specular retro-reflectivity: ☐
 - ▶ Transmission Layer
 - ▶ Subsurface
 - ▶ Medium
 - ▶ Coating Layer
 - ▶ Fuzz Layer
 - ▶ Emission Layer
 - ▶ Thin Film Layer
 - ▶ Geometry Properties



Material Properties Panel:

- Base Layer
 - Specular Layer
 - Specular weight: IfGreater IfGreater
 - Specular color: [Color Picker]
 - Specular roughness: 0.201
 - Specular haze: 0.000
 - Specular haze spread: 0.000
 - Specular IOR: 1.500
 - Specular anisotropy: 0.000
 - Specular retro-reflectivity: IfGreater IfGreater
 - Transmission Layer
 - Subsurface
 - Medium
 - Coating Layer
 - Fuzz Layer
 - Emission Layer
 - Thin Film Layer
 - Geometry Properties
- Custom AOV: None



2027



Real Time





octanerender

2027 Physical Optics

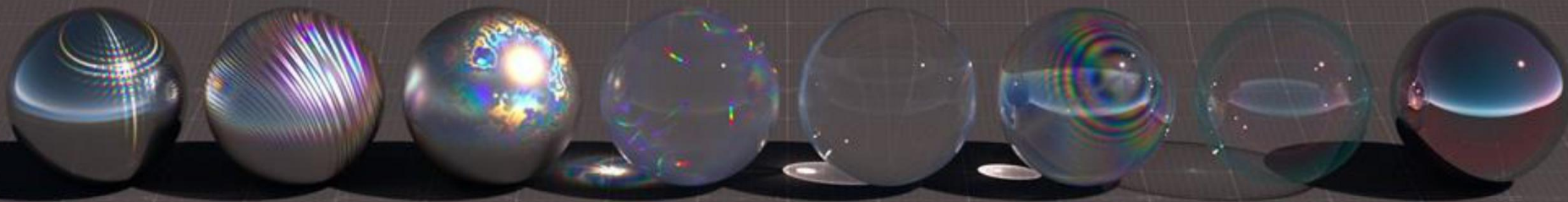


otoy



Double Refraction
Spectral Diffraction
Birefringence Interference

Crossover with MaterialX & OpenPBR

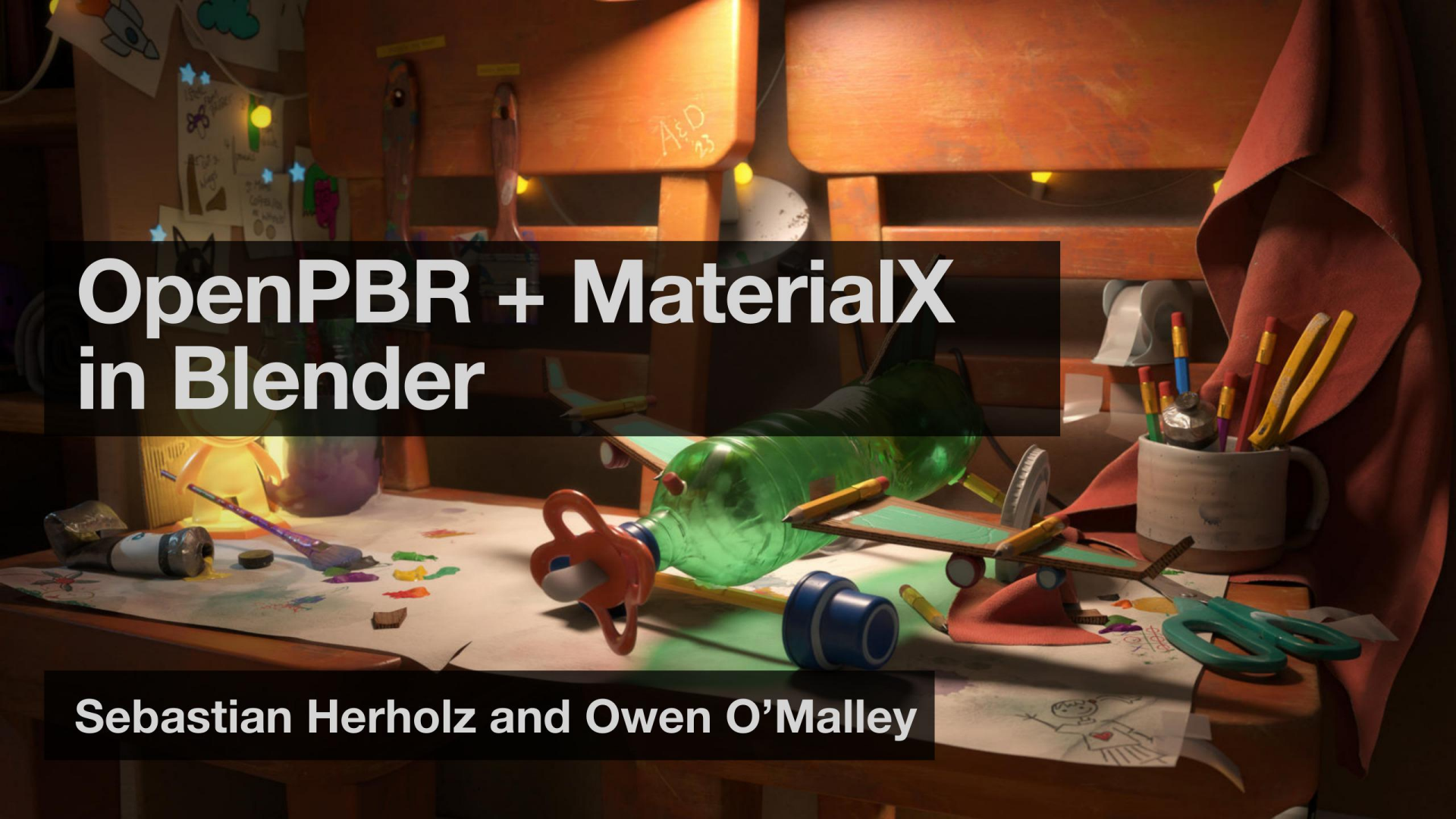




MaterialX & OpenPBR Octane Render®

Arnon Marcus

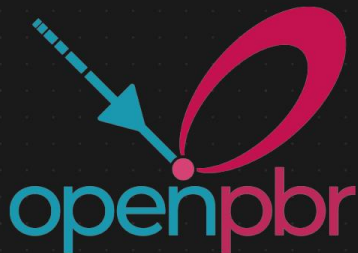
2026.1: <https://render.otoy.com/forum/viewtopic.php?t=85834>
2027.1: <https://render.otoy.com/forum/viewtopic.php?t=85747>



OpenPBR + MaterialX in Blender

Sebastian Herholz and Owen O'Malley

Blender's Journey to interchangeable Assets



- Material appearance

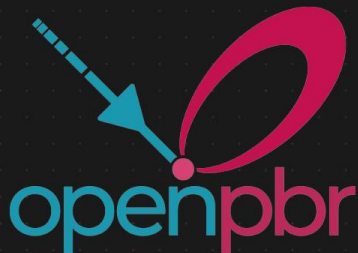


- Material interchange



- Full scene/asset representation

Blender's Journey to interchangeable Assets



- Material appearance



- Material interchange



- Full scene/asset representation

✓ OpenPBR is Blender's first step



OpenPBR Integration in Cycles

Cycles: Material/Shading Backends

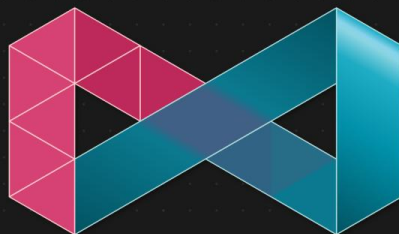
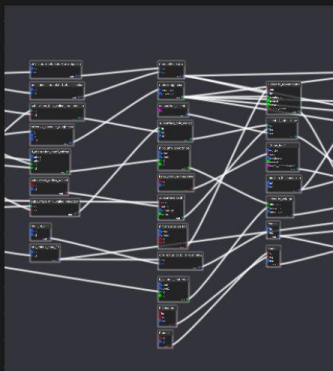
✓ Shader Virtual Machine (SVM)

- Supported on all backends:
CPU, GPU (CUDA/Optix, HIP,
Metal, oneAPI)
- Native representation of all
Blender shader nodes

✓ Open Shading Language (OSL)

- Supported on CPU and Optix
- All shader nodes reimplemented
in OSL
- Executing own OSL shader code

OSL: MaterialX CodeGen



MATERIALX
OSLCodeGen



```
586
587 BSDF_T fuzz_bsdf_out = fuzz_weight * sheen_bsdf(geometry_normal, fuzz_color, fuzz_roughness);
588 BSDF_T coat_bsdf_out = null_closure();
589 mx_dielectric_bsdf(coat_weight,
590                    color(1.000000, 1.000000, 1.000000),
591                    coat_ior,
592                    coat_roughness_vector_out,
593                    0.000000,
594                    1.500000,
595                    geometry_coat_normal,
596                    geometry_coat_tangent,
597                    "multi_ggx",
598                    "R",
599                    coat_bsdf_out);
600 BSDF_T metal_bsdf_tf_out = null_closure();
601 mx_generalized_schlick_bsdf(specular_weight,
602                             metal_reflectivity_out,
603                             metal_edgecolor_out,
604                             color(1.000000, 1.000000, 1.000000),
605                             5.000000,
606                             main_roughness_out,
607                             thin_film_thickness_nm_out,
608                             thin_film_ior,
609                             geometry_normal,
610                             geometry_tangent,
611                             "multi_ggx",
612                             "R",
613                             metal_bsdf_tf_out);
614
615 BSDF_T metal_bsdf_out = null_closure();
616 mx_generalized_schlick_bsdf(specular_weight,
617                             metal_reflectivity_out,
618                             metal_edgecolor_out,
619                             color(1.000000, 1.000000, 1.000000),
620                             5.000000,
621                             main_roughness_out,
622                             0.000000,
623                             1.500000,
624                             geometry_normal,
625                             geometry_tangent,
626                             "multi_ggx",
627                             "R",
628                             metal_bsdf_out);
```



✓ Converting the OpenPBR MaterialX graph to OSL code

SVM: Clean Room Integration



Weizhen Huang
(Blender)

✓ Word-by-word implementation following the OpenPBR v1.1.1 specification

Adobe vs SVM vs OSL



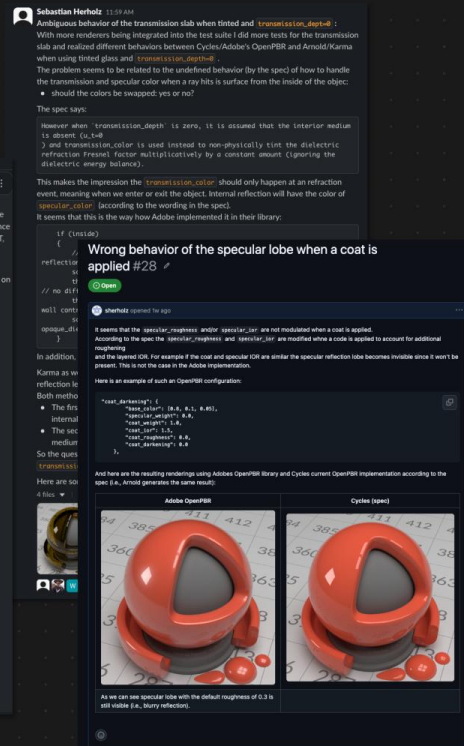
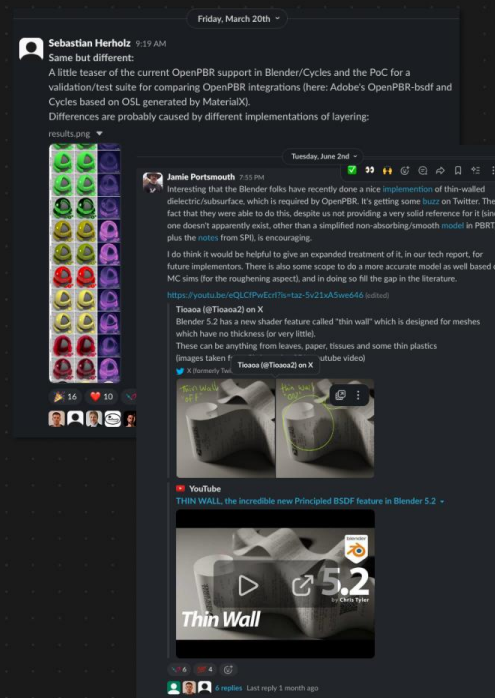
Adobe vs SVM vs OSL



Community Interactions

✓ Collaborative work unify OpenPBR appearances:

- Discussions (Slack)
- PRs



Validation Framework

- ✓Collection of OpenPBR parameter sets:

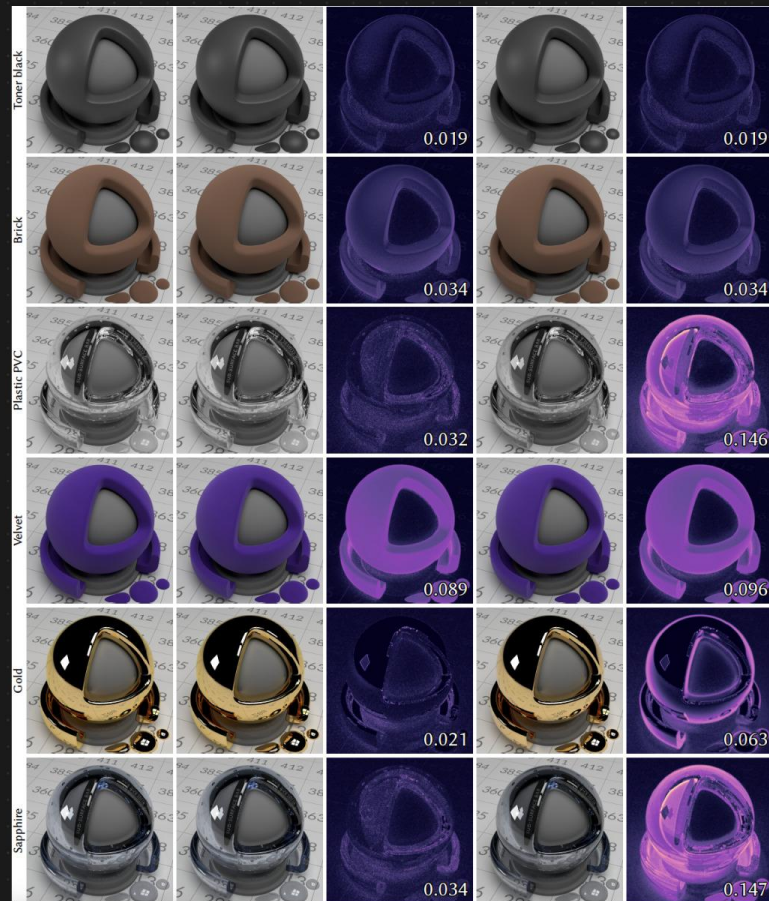
Rendered on the same scene
using calibrated lighting and color spaces

- ✓Curated rendering scripts to replicate results

Easy to expand for new renderers or parameter sets

- ✓Supported renderers:

Cycles, Adobe's Eclair, Arnold, Karma,
VRay, MoonRay (WIP), Glimpse (WIP)



Junkshop (Principled)



Junkshop (OpenPBR)





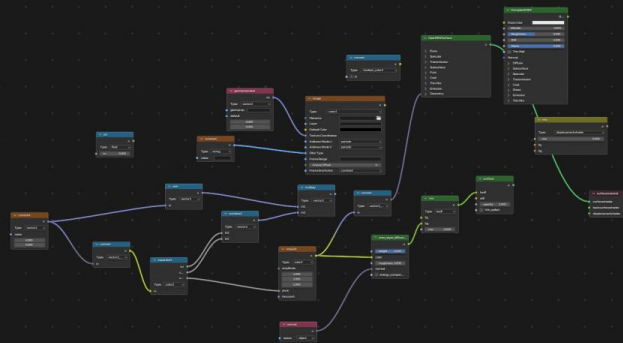
MaterialX (GSoC)

Student: Owen O'Malley

Node Representation



```
<materialx version="1.39" colorspace="lin_rec709">
<surfacematerial name="Test_Material" type="material" >
  <input name="surfaceshader" type="surfaceshader" nodename="surface" />
</surfacematerial>
<texcoord name="texcoord_vector2" type="vector2" />
<oren_nayar_diffuse_bsdf name="oren_nayar_diffuse_bsdf" type="BSDF" >
  <input name="color" type="color3" nodename="noise2d_color3" />
  <input name="roughness" type="float" value="0.1" />
  <input name="energy_compensation" type="boolean" value="true" />
</oren_nayar_diffuse_bsdf>
<surface name="surface" type="surfaceshader" >
  <input name="bsdf" type="BSDF" nodename="oren_nayar_diffuse_bsdf" />
</surface>
<noise2d name="noise2d_color3" type="color3" >
  <input name="texcoord" type="vector2" nodename="multiply_vector2FA" />
</noise2d>
<multiply name="multiply_vector2FA" type="vector2" >
  <input name="in1" type="vector2" nodename="texcoord_vector2" />
  <input name="in2" type="float" value="50" />
</multiply>
</materialx>
```



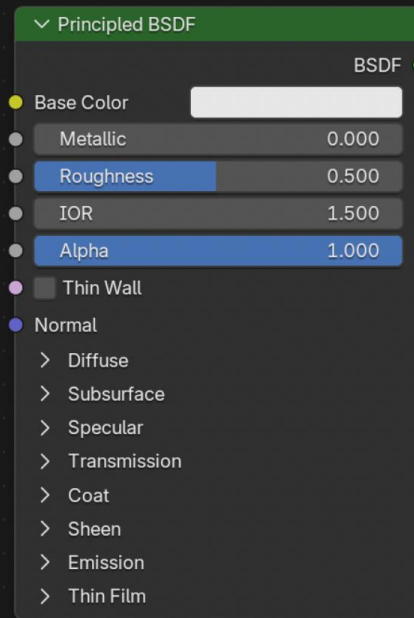
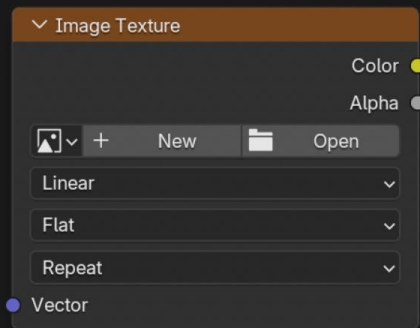
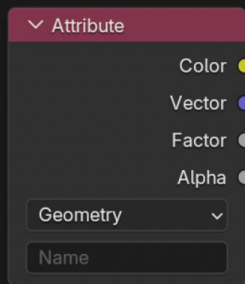
Import/Export

Shader Editor

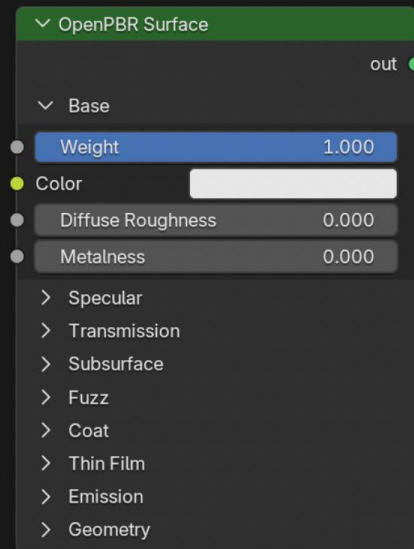
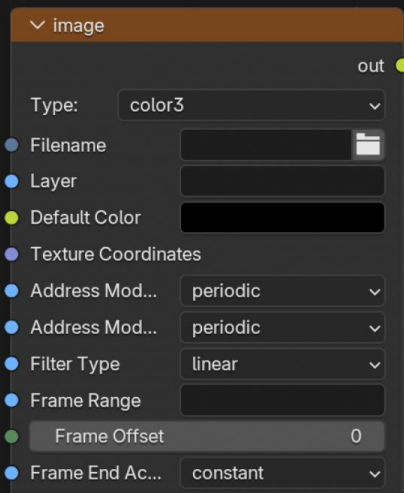
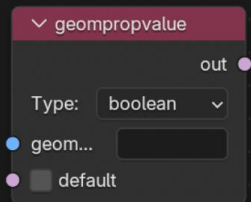
Node Representation

- One shader node compiled
- State determines everything
- Reads \$MATERIALX_SEARCH_PATH

Node Representation



Node Representation



Import/Export

- For now, *.mtlx only
- Supports name collision options
 - Add/Overwrite/Merge/Ignore
- Optional flattening of NodeDef/NodeGraph pairs



Rendering

- Use OSL codegen
- Generate bytecode per node



Outlook

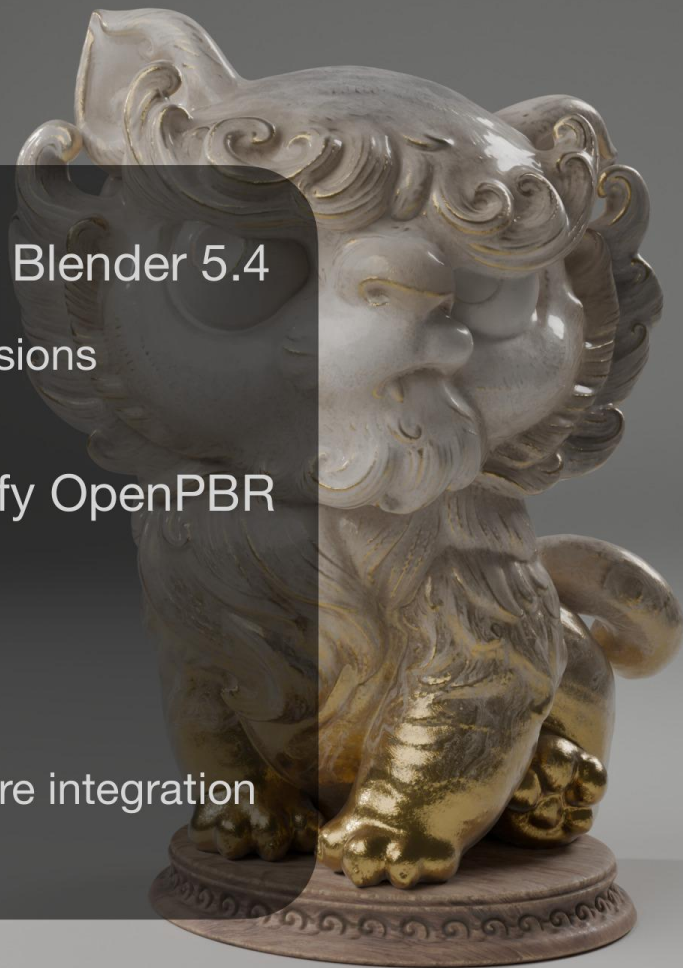
✓ Full OpenPBR v1.1.1 support planned for Blender 5.4

- Adding support for upcoming OpenPBR versions

✓ Collaborating with the community to unify OpenPBR appearance

✓ Continuing MaterialX GSoC project

Working towards an initial experimental feature integration



Thank You!



OpenPBR Volume

MaterialX Virtual Town Hall 2026

Adrien Herubel
Director, Software Engineering
Autodesk

14/09/2026

Chris Rydalch
Senior UX Designer
SideFX



OpenPBR Volume



×



- A new appearance model for volume assets
- Collaboration between Autodesk and SideFX
- Combining Houdini Pyro and Arnold Standard Volume
- Work in progress
- Early draft stage - now in private review, reach out!



Karma



Arnold

Parametrization

- Extinction
- Scattering
- Emission (s)
 - Multiple sets
 - Direct control
 - Physical Blackbody
 - Artistic Blackbody
- Ramps and masks



Next steps

- Public draft soon
- Working with the community to support the parametrization in USD and MaterialX

ASWF

/* ACADEMY
SOFTWARE
FOUNDATION

