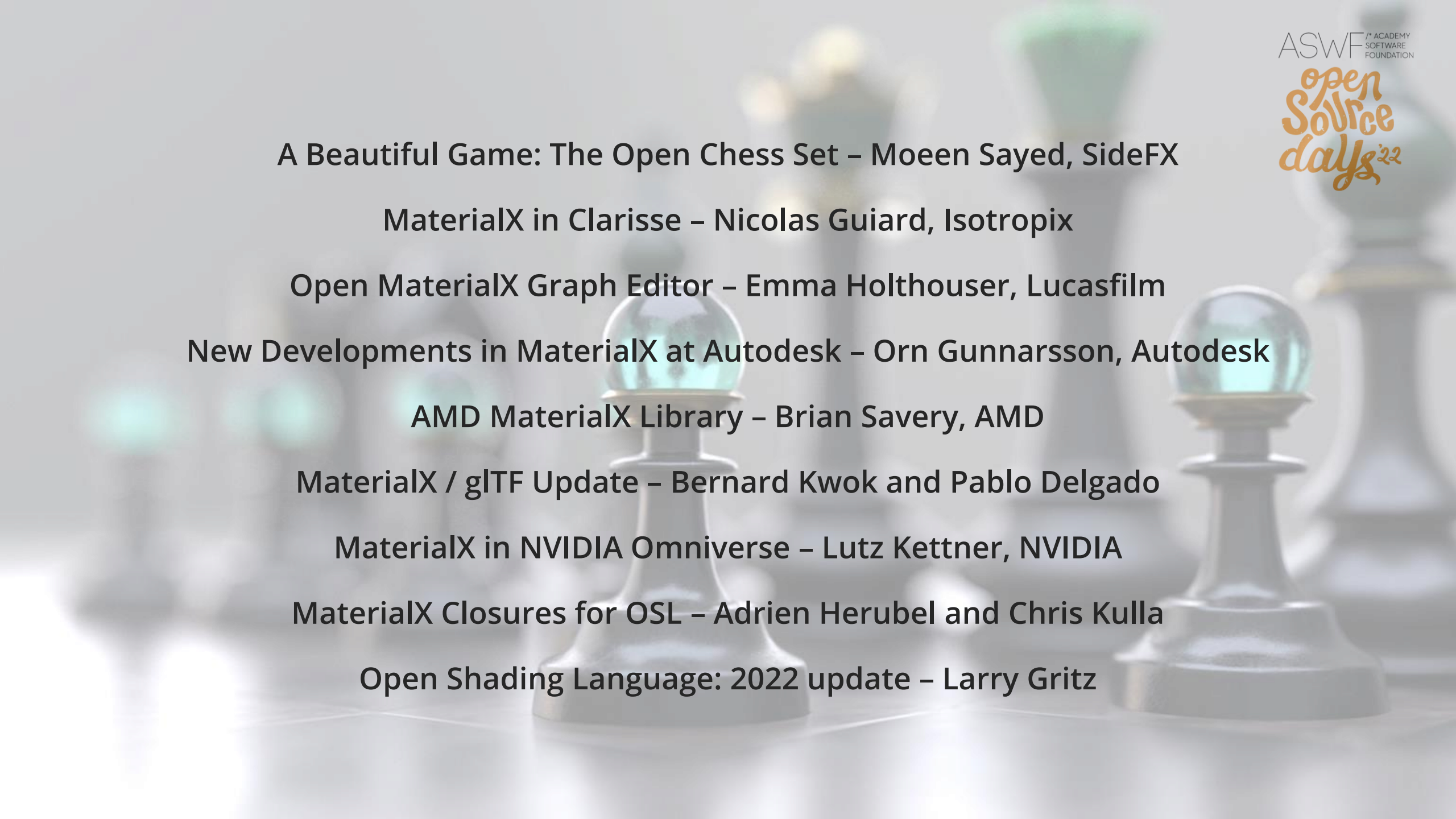
ASWF /* ACADEMY SOFTWARE FOUNDATION

open Source days '22

New Developments in MaterialX and OSL

Jonathan Stone, Lucasfilm Advanced Development Group



A Beautiful Game: The Open Chess Set – Moeen Sayed, SideFX

MaterialX in Clarisse – Nicolas Guiard, Isotropix

Open MaterialX Graph Editor – Emma Holthouser, Lucasfilm

New Developments in MaterialX at Autodesk – Orn Gunnarsson, Autodesk

AMD MaterialX Library – Brian Savery, AMD

MaterialX / glTF Update – Bernard Kwok and Pablo Delgado

MaterialX in NVIDIA Omniverse – Lutz Kettner, NVIDIA

MaterialX Closures for OSL – Adrien Herubel and Chris Kulla

Open Shading Language: 2022 update – Larry Gritz

ASWF /* ACADEMY SOFTWARE FOUNDATION

open source days '22

A Beautiful Game: The Open Chess Set

Moeen Sayed, VFX Artist, NineBetween for SideFX


The
Open
Chess Set

"A Beautiful Game" Karma Learning Material

- Originally a learning resource by SideFX, titled "A Beautiful Game"
- USD compliant set of assets created by Mujtaba Sayed
- Used to explore various shaders within Solaris
- Explored rendering workflows within Solaris
- MaterialX Shaders rendered in Karma CPU and XPU

The Open Chess Set

- Due to public availability, Open-Sourcing was the next logical step
- Tutorial USD assets used to create Open-Source MaterialX Chess Set
- Chris Rydalch converted the materials and extracted the geometry from the original files
- Chris, with Mark Elendt, managed the process of contributing to the MaterialX repository
- Pablo Delgado created a compiled USD asset.
- Jonathan Stone fixed any issues and completed the conversion to .mtlx and .gltf

Benefits of The Open Chess Set

- Acts as a shared, consistent testing asset which can allow comparison between renderers.
- Covers a range of shading features such as transmission, metallics and SSS all within a single scene.
- It encourages the experimentation of 3rd party renderers within the Solaris context which allows Houdini users further flexibility in exploration.
- With conformity to Open Standards, it is accessible for personal and professional research and development.
- Acts as a unified anchor-point for discussion and exchange acting almost as a benchmark between renderers for MaterialX.

Closing Thoughts

- The Open Chess Set has been proposed for inclusion in the new **Digital Production Examples Library** (DPEL), which will be presented in the DPEL segment later today.
- Thanks to Chris Rydalch, Mark Elendt and Pablo Delgado for their efforts on the Open Chess Set
- Special thanks to Rafal Jaroszkiewicz and Jonathan Stone for their assistance with information for this presentation

MaterialX in Clarisse

Nicolas Guiard, Head of RnD, *Isotropix*

What is Clarisse?

- Specialized DCC for set-dressing/lookdev/lighting/rendering
- Used in VFX and Animation
- Big studios to freelancers
- Feature films, TV shows, commercials

Example: Dune (2021)

- 1000+ shots rendered with Clarisse renderer
- Best VFX Oscar 2022
  Kudos to DNEG

Why is MaterialX the next big thing?

- We, software vendors, suck! ;)
  - No Ultimate solution
    - Users forced to juggle between multiple packages/renderers
- Different technologies/approaches to consider:
  - CPU, GPU, Hybrid CPU+GPU (Apple)
  - Realtime/offline renderers
- What do users care about?
  - Easy asset exchange between applications

Today's solutions

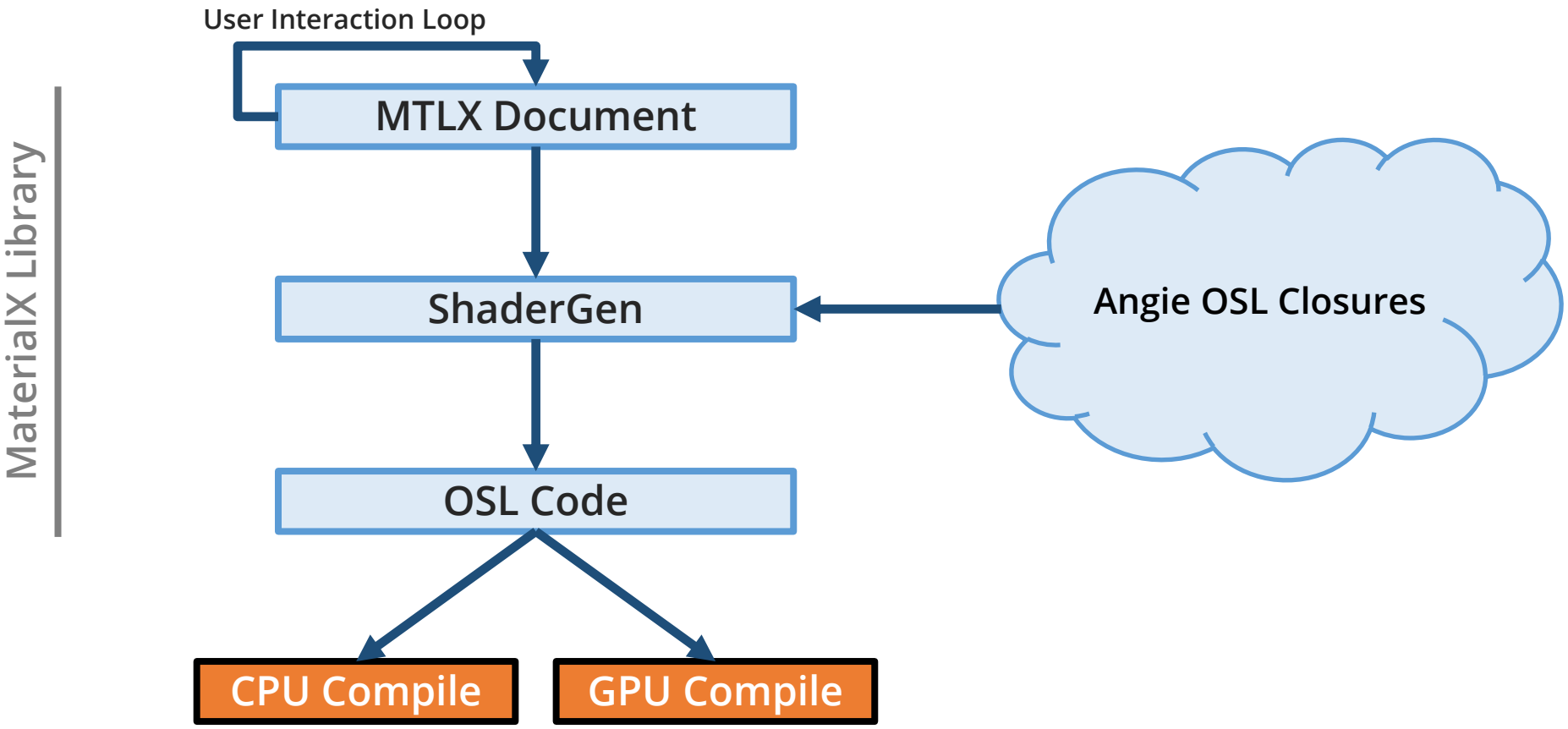
- Alembic
  - Reliable exchange of geometries, animations, cameras
    - Alembic is industry standard and USD wide-scale adoption is at the corner
- OpenVDB
  - Reliable exchange of volumetric datasets
    - OpenVDB is industry standard
- What about looks?
  - MaterialX is the missing link to offer render-agnostic looks exchange

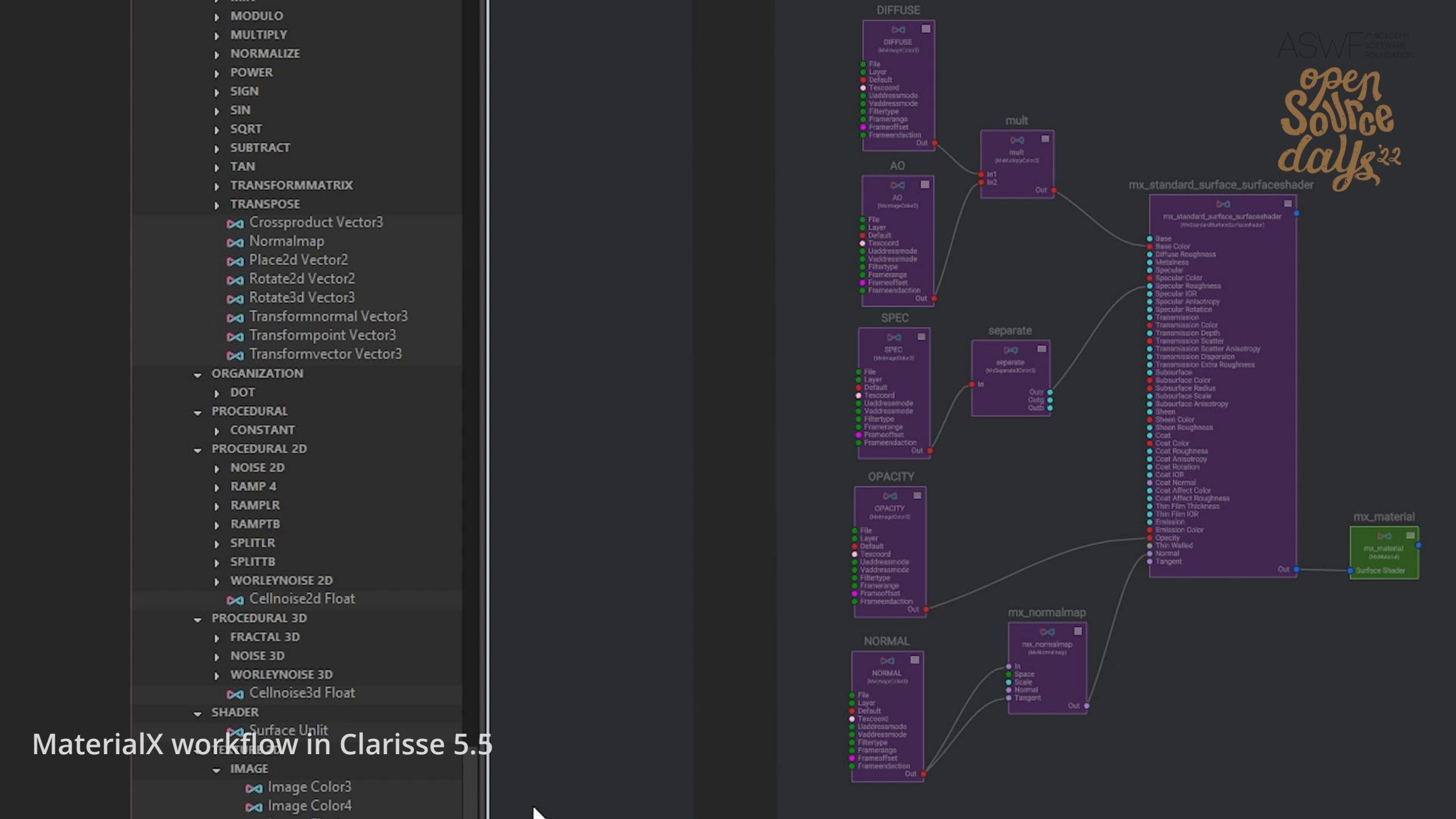
The 3D Rendering Jungle

MaterialX support in Clarisse

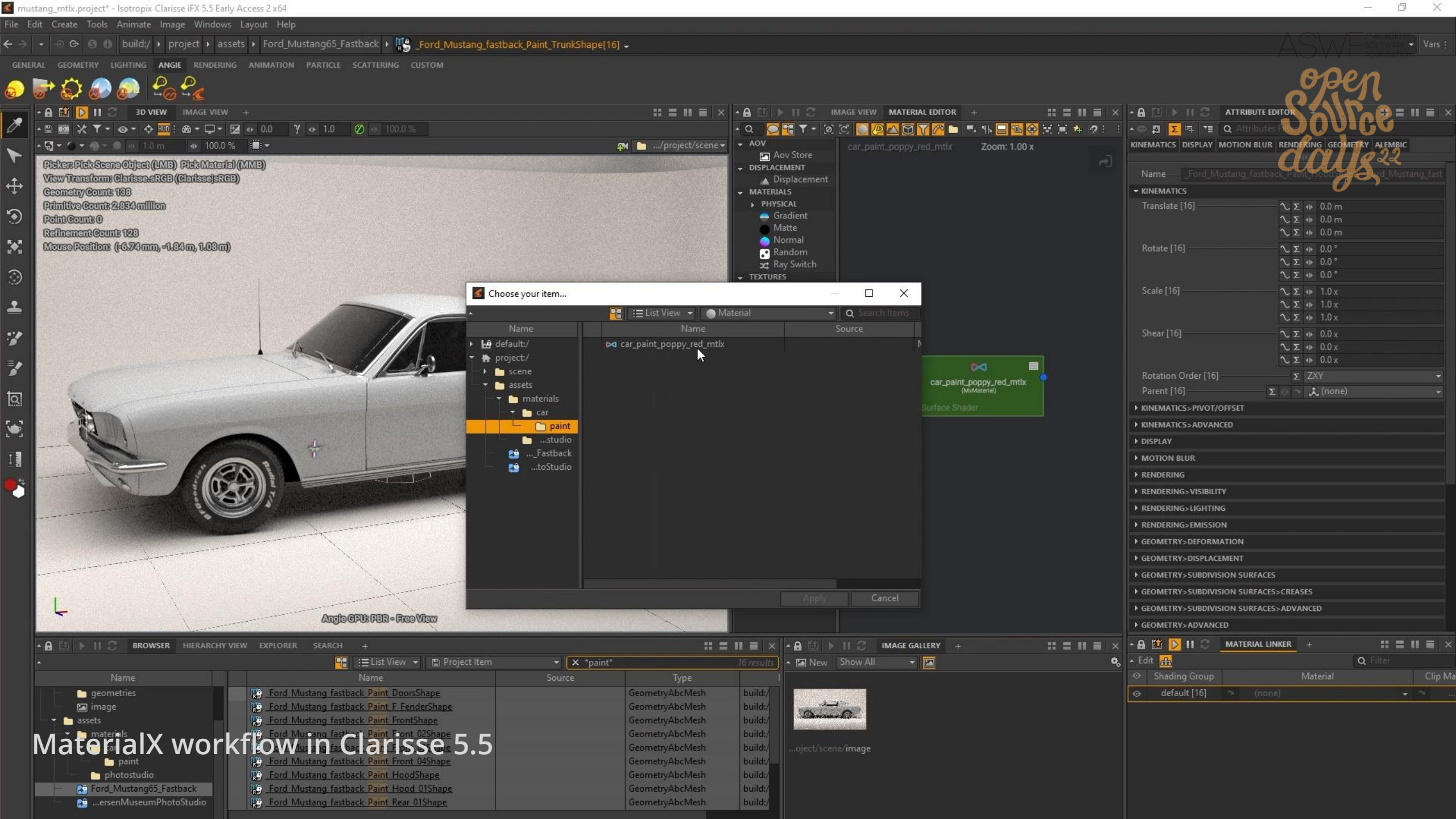
- Available in Clarisse 5.5
  - Early Access available since December 2021
- Rendered exclusively with Angie
  - Nextgen OSL based CPU+GPU renderer of Clarisse
- MaterialX looks can be accessed in Python/C++
  - Useful for pipeline or for third party renderers in Clarisse

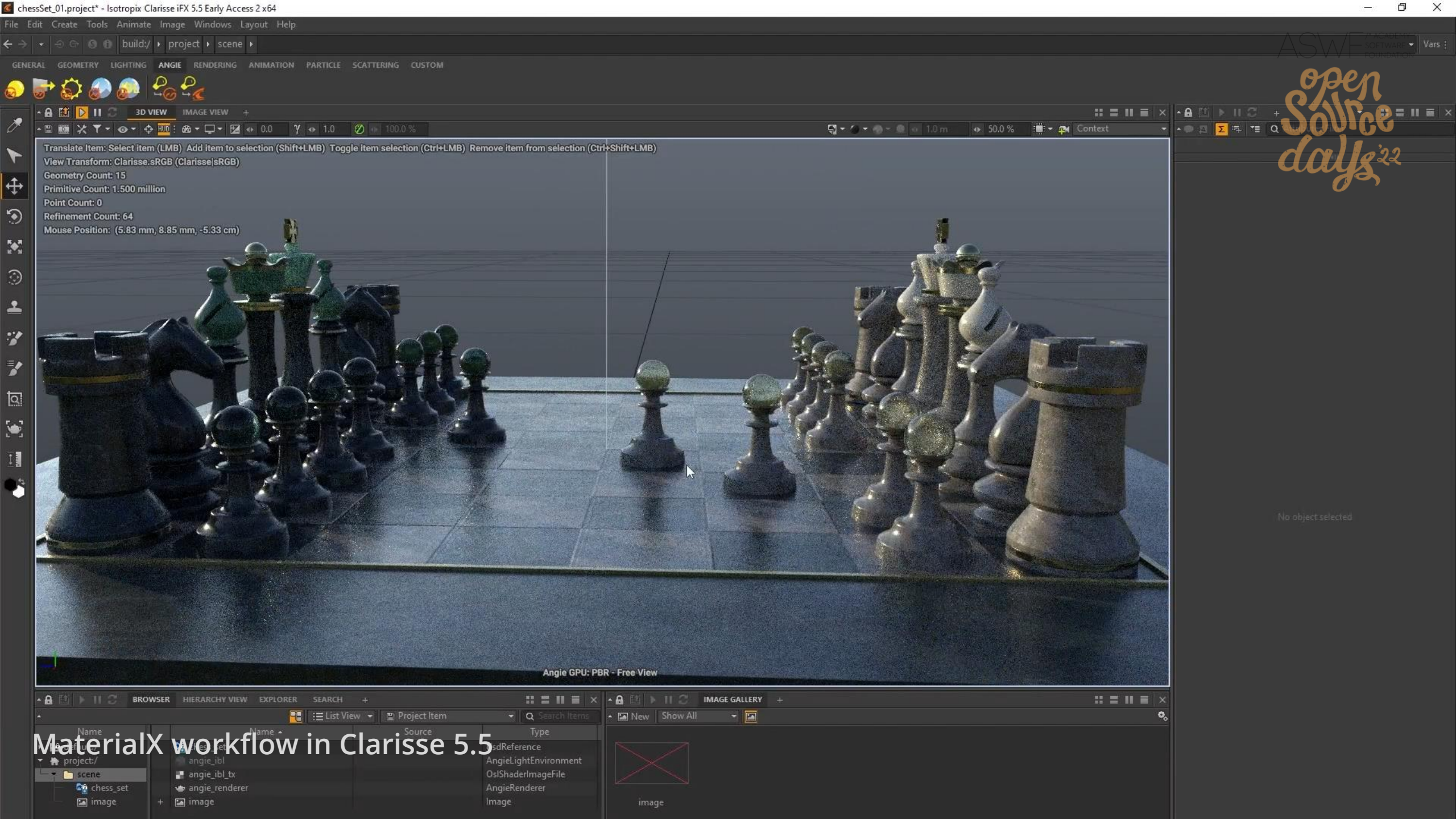
MaterialX in Clarisse Overview

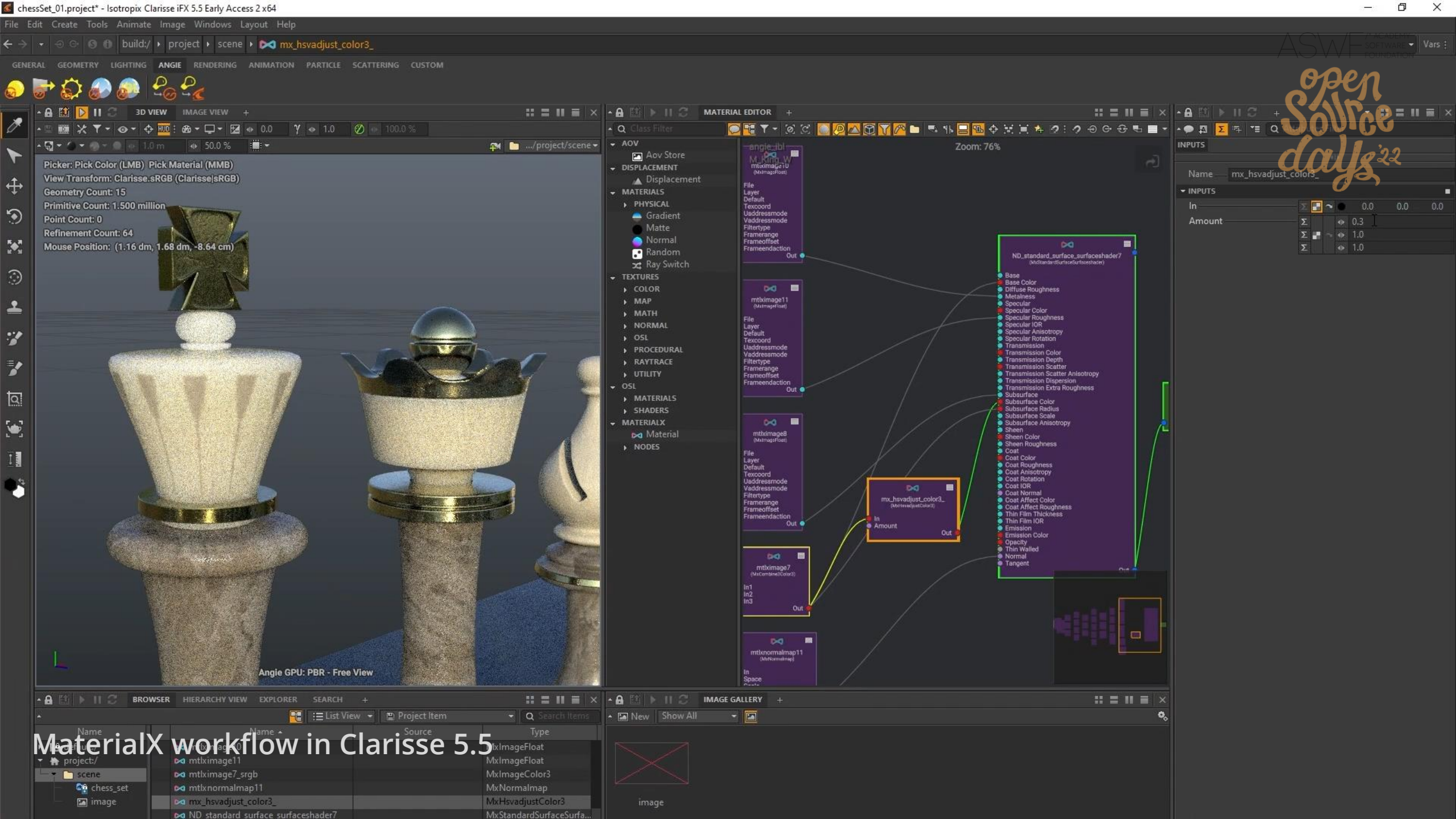




MaterialX workflow in Clarisse 5.5


mustang_mtlx.project* - Isotropix Clarisse iFX 5.5 Early Access 2 x64
File Edit Create Tools Animate Image Windows Layout Help
build:/ project assets Ford_Mustang65_Fastback _Ford_Mustang_fastback_Paint_TrunkShape[16]
GENERAL GEOMETRY LIGHTING ANGIE RENDERING ANIMATION PARTICLE SCATTERING CUSTOM
3D VIEW IMAGE VIEW
Picker: Pick Scene Object (LMB) Pick Material (MMB)
View Transform: Clarisse sRGB (ClarisseJsRGB)
Geometry Count: 138
Primitive Count: 2.834 million
Point Count: 0
Refinement Count: 128
Mouse Position: (-6.74 mm, -1.84 m, 1.08 m)
Angie GPU: PBR - Free View
IMAGE VIEW MATERIAL EDITOR
AOV
Aov Store
DISPLACEMENT
Displacement
MATERIALS
PHYSICAL
Gradient
Matte
Normal
Random
Ray Switch
TEXTURES
car_paint_poppy_red_mtlx
Zoom: 1.00 x
car_paint_poppy_red_mtlx
(MxMaterial)
Surface Shader
Choose your item...
List View
Material
Search Items
Name
Source
default:/
project:/
scene
assets
materials
car
paint
...studio
..._Fastback
...toStudio
car_paint_poppy_red_mtlx
Apply
Cancel
ATTRIBUTE EDITOR
KINEMATICS DISPLAY MOTION BLUR RENDERING GEOMETRY ALEMBIC
Name
KINEMATICS
Translate [16]
Rotate [16]
Scale [16]
Shear [16]
Rotation Order [16]
ZXY
Parent [16]
(none)
KINEMATICS>PIVOT/OFFSET
KINEMATICS>ADVANCED
DISPLAY
MOTION BLUR
RENDERING
RENDERING>VISIBILITY
RENDERING>LIGHTING
RENDERING>EMISSION
GEOMETRY>DEFORMATION
GEOMETRY>DISPLACEMENT
GEOMETRY>SUBDIVISION SURFACES
GEOMETRY>SUBDIVISION SURFACES>CREASES
GEOMETRY>SUBDIVISION SURFACES>ADVANCED
GEOMETRY>ADVANCED
BROWSER HIERARCHY VIEW EXPLORER SEARCH
List View
Project Item
paint
16 results
Name
Source
Type
geometries
image
assets
materials
paint
photostudio
Ford_Mustang65_Fastback
...ersenMuseumPhotoStudio
_Ford_Mustang_fastback_Paint_DoorsShape
_Ford_Mustang_fastback_Paint_F_FenderShape
_Ford_Mustang_fastback_Paint_FrontShape
_Ford_Mustang_fastback_Paint_Front_02Shape
_Ford_Mustang_fastback_Paint_Front_04Shape
_Ford_Mustang_fastback_Paint_HoodShape
_Ford_Mustang_fastback_Paint_Hood_01Shape
_Ford_Mustang_fastback_Paint_Rear_01Shape
GeometryAbcMesh
build:/
IMAGE GALLERY
New
Show All
...oject/scene/image
MATERIAL LINKER
Edit
Filter
Shading Group
Material
default [16]
(none)
MaterialX workflow in Clarisse 5.5


chessSet_01.project* - Isotropix Clarisse iFX 5.5 Early Access 2 x64
File Edit Create Tools Animate Image Windows Layout Help
build:/ project scene
GENERAL GEOMETRY LIGHTING ANGIE RENDERING ANIMATION PARTICLE SCATTERING CUSTOM
3D VIEW IMAGE VIEW
Translate Item: Select item (LMB) Add item to selection (Shift+LMB) Toggle item selection (Ctrl+LMB) Remove item from selection (Ctrl+Shift+LMB)
View Transform: Clarisse.sRGB (Clarisse|sRGB)
Geometry Count: 15
Primitive Count: 1.500 million
Point Count: 0
Refinement Count: 64
Mouse Position: (5.83 mm, 8.85 mm, -5.33 cm)
Angie GPU: PBR - Free View
BROWSER HIERARCHY VIEW EXPLORER SEARCH
IMAGE GALLERY
No object selected
project:/
scene
chess_set
image
angie_ibl
angie_ibl_tx
angie_renderer
AngieLightEnvironment
OslShaderImageFile
AngieRenderer
Image
MaterialX workflow in Clarisse 5.5


chessSet_01.project* - Isotropix Clarisse iFX 5.5 Early Access 2 x64
File Edit Create Tools Animate Image Windows Layout Help
build:/ project scene mx_hsvadjust_color3_
GENERAL GEOMETRY LIGHTING ANGIE RENDERING ANIMATION PARTICLE SCATTERING CUSTOM
3D VIEW IMAGE VIEW
Picker: Pick Color (LMB) Pick Material (MMB)
View Transform: Clarisse.sRGB (Clarisse|sRGB)
Geometry Count: 15
Primitive Count: 1.500 million
Point Count: 0
Refinement Count: 64
Mouse Position: (1.16 dm, 1.68 dm, -8.64 cm)
Angie GPU: PBR - Free View
MATERIAL EDITOR
Zoom: 76%
AOV
Aov Store
DISPLACEMENT
Displacement
MATERIALS
PHYSICAL
Gradient
Matte
Normal
Random
Ray Switch
TEXTURES
COLOR
MAP
MATH
NORMAL
OSL
PROCEDURAL
RAYTRACE
UTILITY
OSL
MATERIALS
SHADERS
MATERIALX
Material
NODES
mtlximage10
mtlximage11
mtlximage8
mtlximage7
mtlxnormalmap11
mx_hsvadjust_color3_
ND_standard_surface_surfaceshader7
Base
Base Color
Diffuse Roughness
Metalness
Specular
Specular Color
Specular Roughness
Specular IOR
Specular Anisotropy
Specular Rotation
Transmission
Transmission Color
Transmission Depth
Transmission Scatter
Transmission Scatter Anisotropy
Transmission Dispersion
Transmission Extra Roughness
Subsurface
Subsurface Color
Subsurface Radius
Subsurface Scale
Subsurface Anisotropy
Sheen
Sheen Color
Sheen Roughness
Coat
Coat Color
Coat Roughness
Coat Anisotropy
Coat Rotation
Coat IOR
Coat Normal
Coat Affect Color
Coat Affect Roughness
Thin Film Thickness
Thin Film IOR
Emission
Emission Color
Opacity
Thin Walled
Normal
Tangent
INPUTS
Name mx_hsvadjust_color3_
In
Amount
0.3
1.0
1.0
BROWSER HIERARCHY VIEW EXPLORER SEARCH
List View
Project Item
Search Items
Name
Source
Type
project:/
scene
chess_set
image
mtlximage10 MxImageFloat
mtlximage11 MxImageFloat
mtlximage7_srgb MxImageColor3
mtlxnormalmap11 MxNormalmap
mx_hsvadjust_color3_ MxHsvadjustColor3
ND_standard_surface_surfaceshader7 MxStandardSurfaceSurfa...
IMAGE GALLERY
New
Show All
image
MaterialX workflow in Clarisse 5.5
ASWF
open source days '22

ASWF /* ACADEMY SOFTWARE FOUNDATION

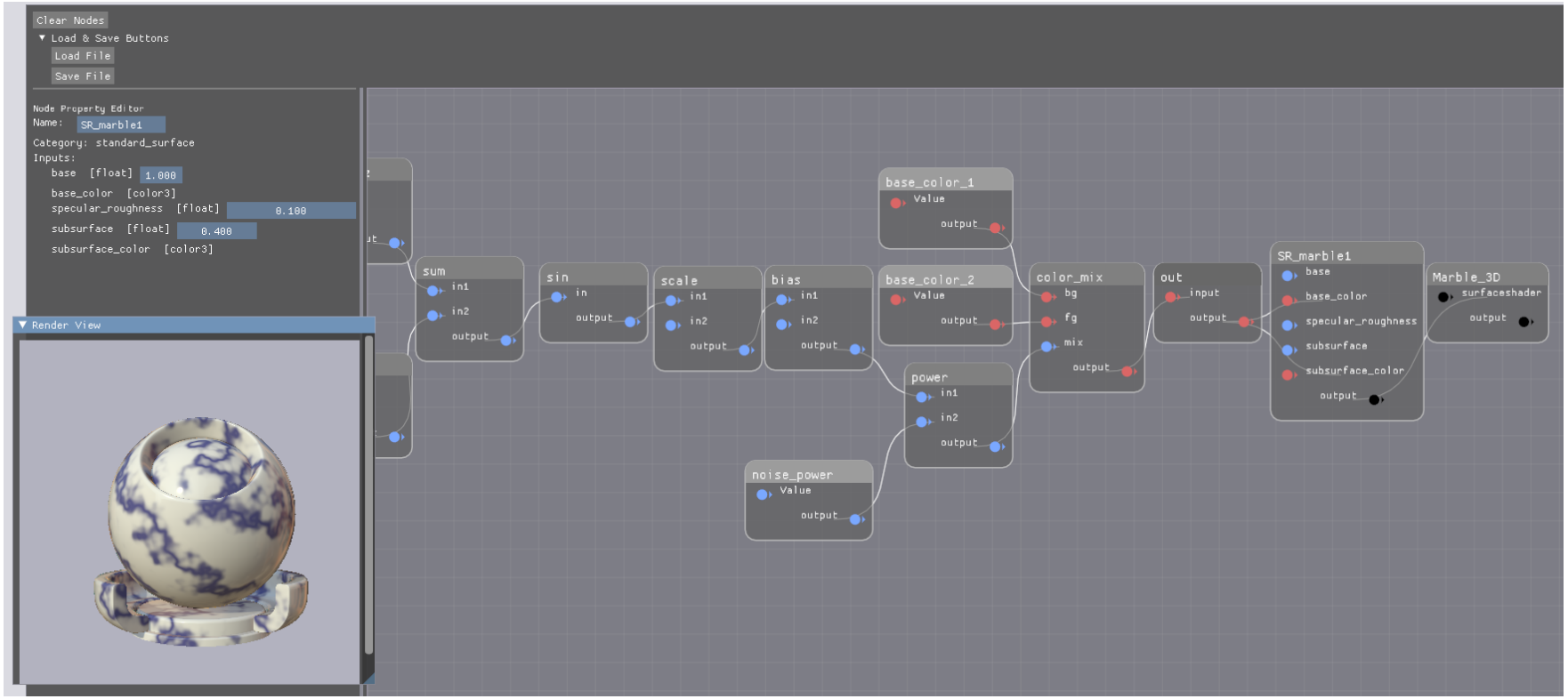
open source days '22

Open MaterialX Graph Editor

Emma Holthouser, Rendering Engineer Intern, Lucasfilm

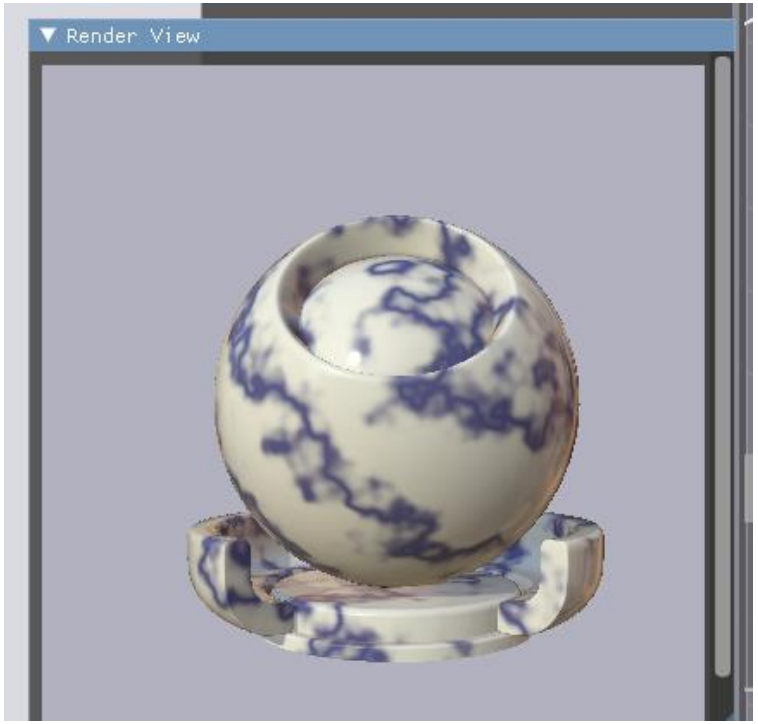
Motivation

- Requests from ILM artists and members of the MaterialX community
- Allow for an Open-Source way to edit MaterialX Graphs



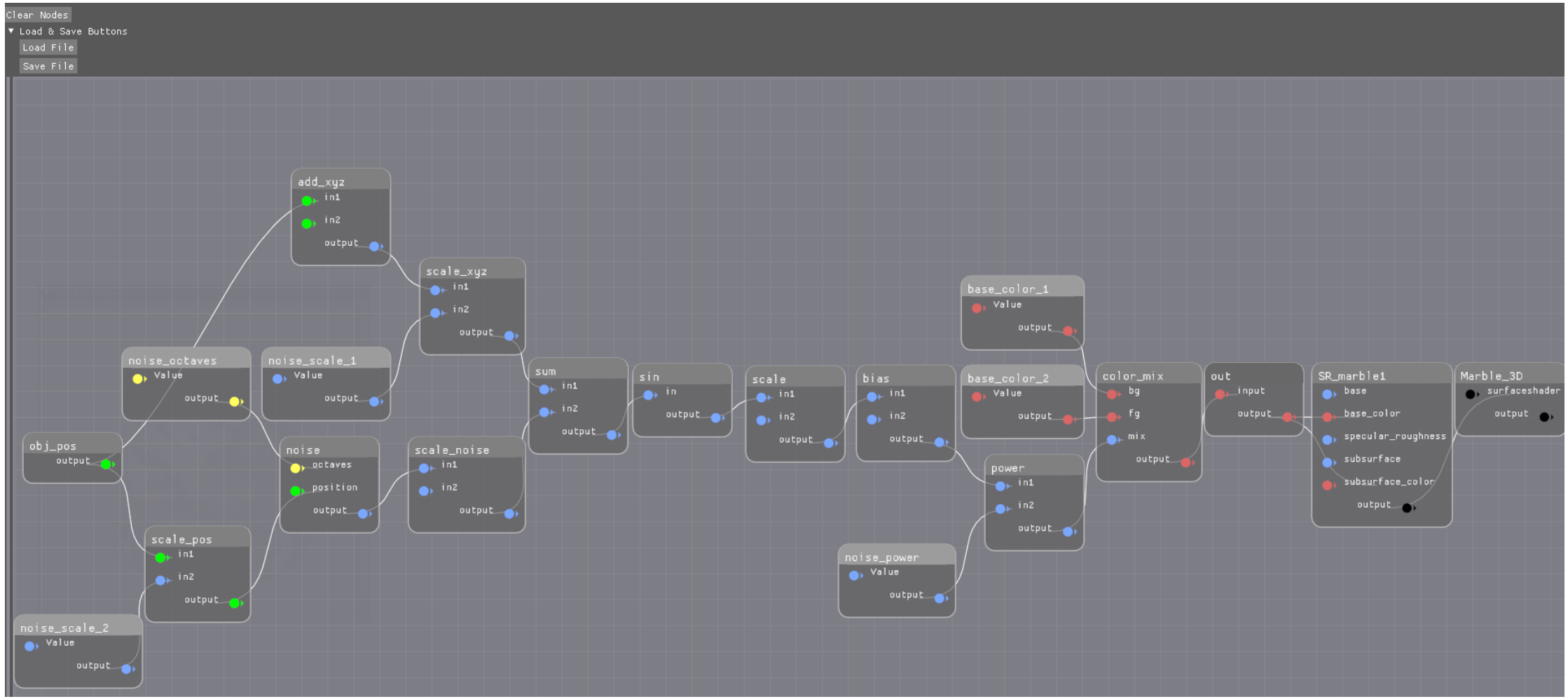
Dependencies

- ImGui for Ui[1]
- Node Editor for ImGui[2]
- MaterialXRender[3] for Rendering



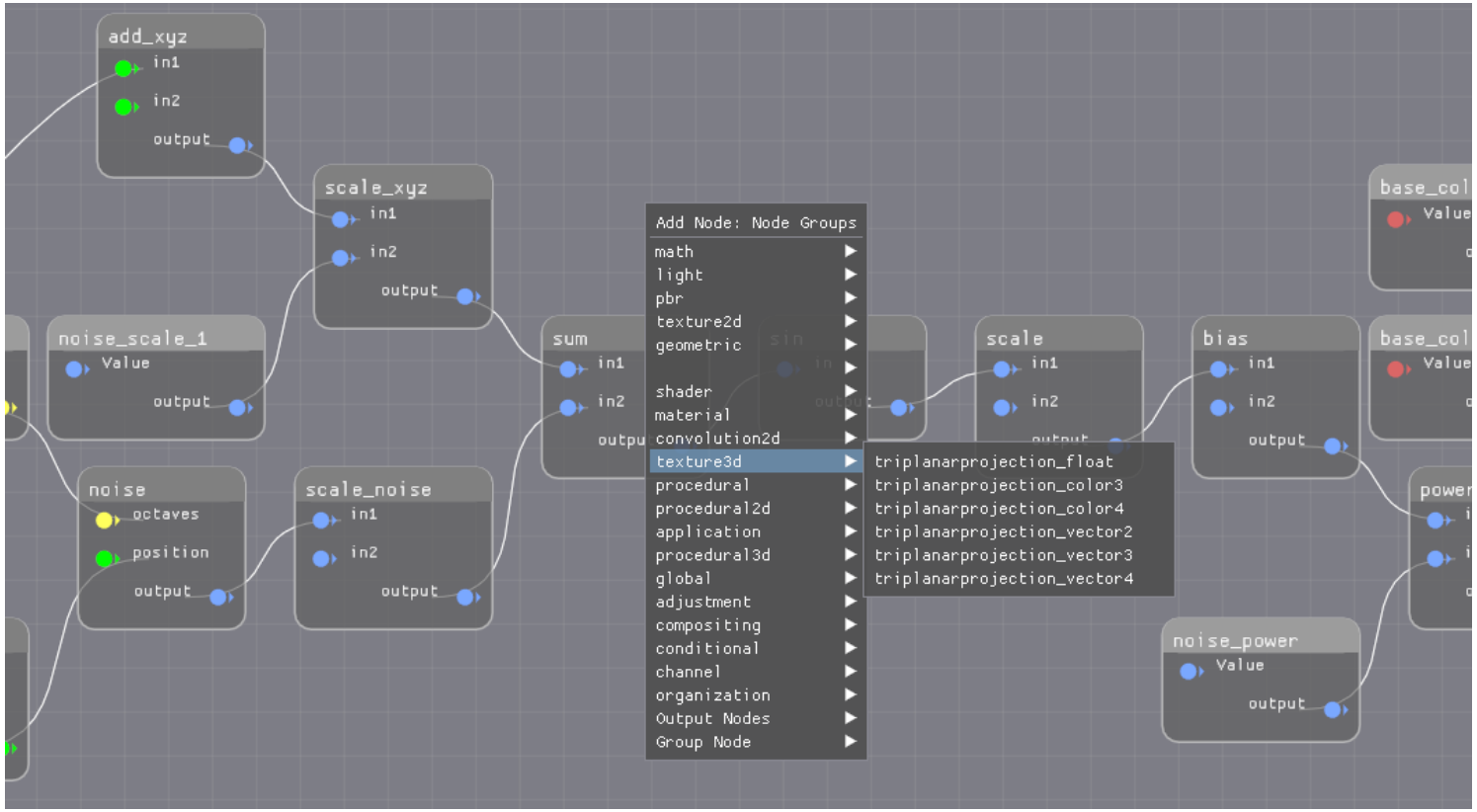
[1]https://github.com/ocornut/imgui
[2]https://github.com/thedmd/imgui-node-editor
[3]https://github.com/AcademySoftwareFoundation/MaterialX/tree/main/source/MaterialXRender

Current Features - Load MaterialX File



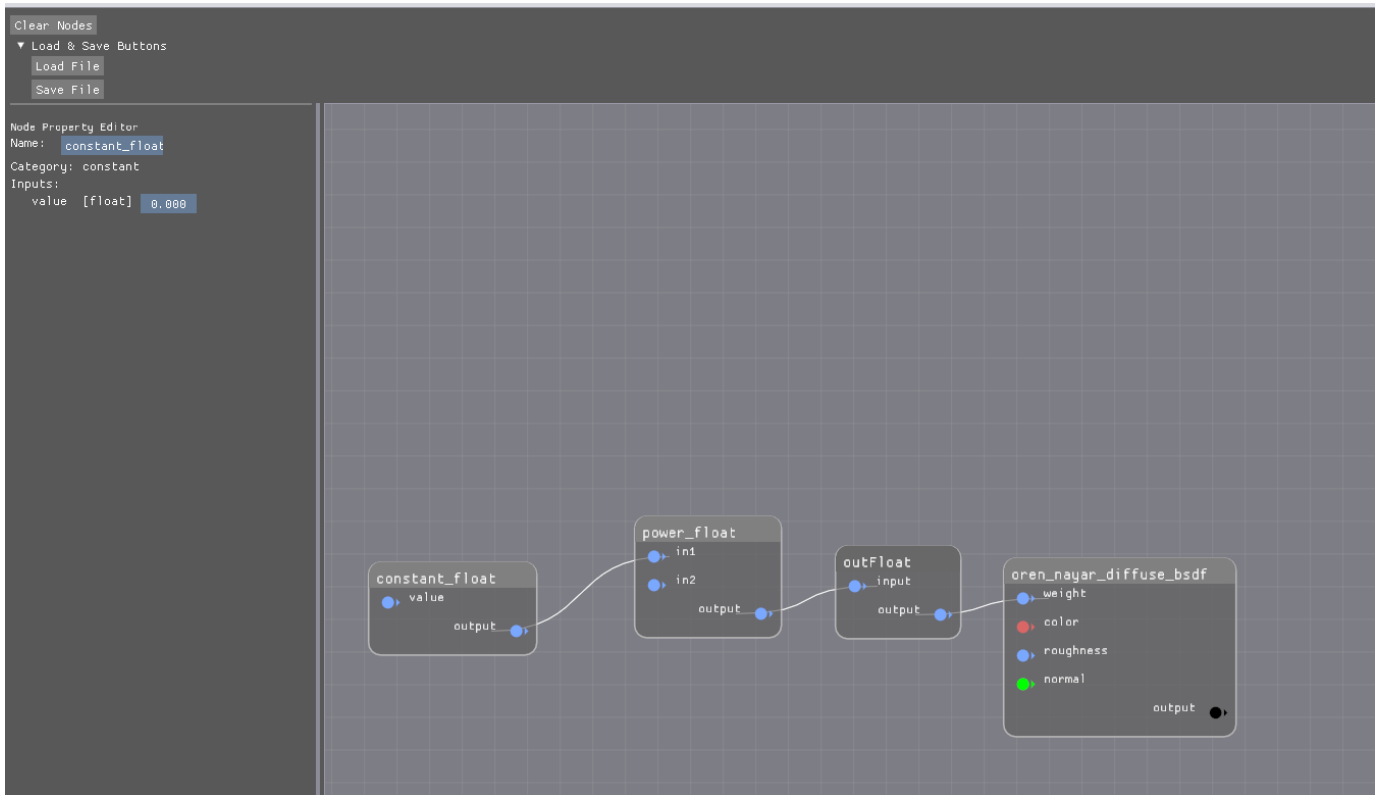
Current Features – Add Node

- Update existing graphs

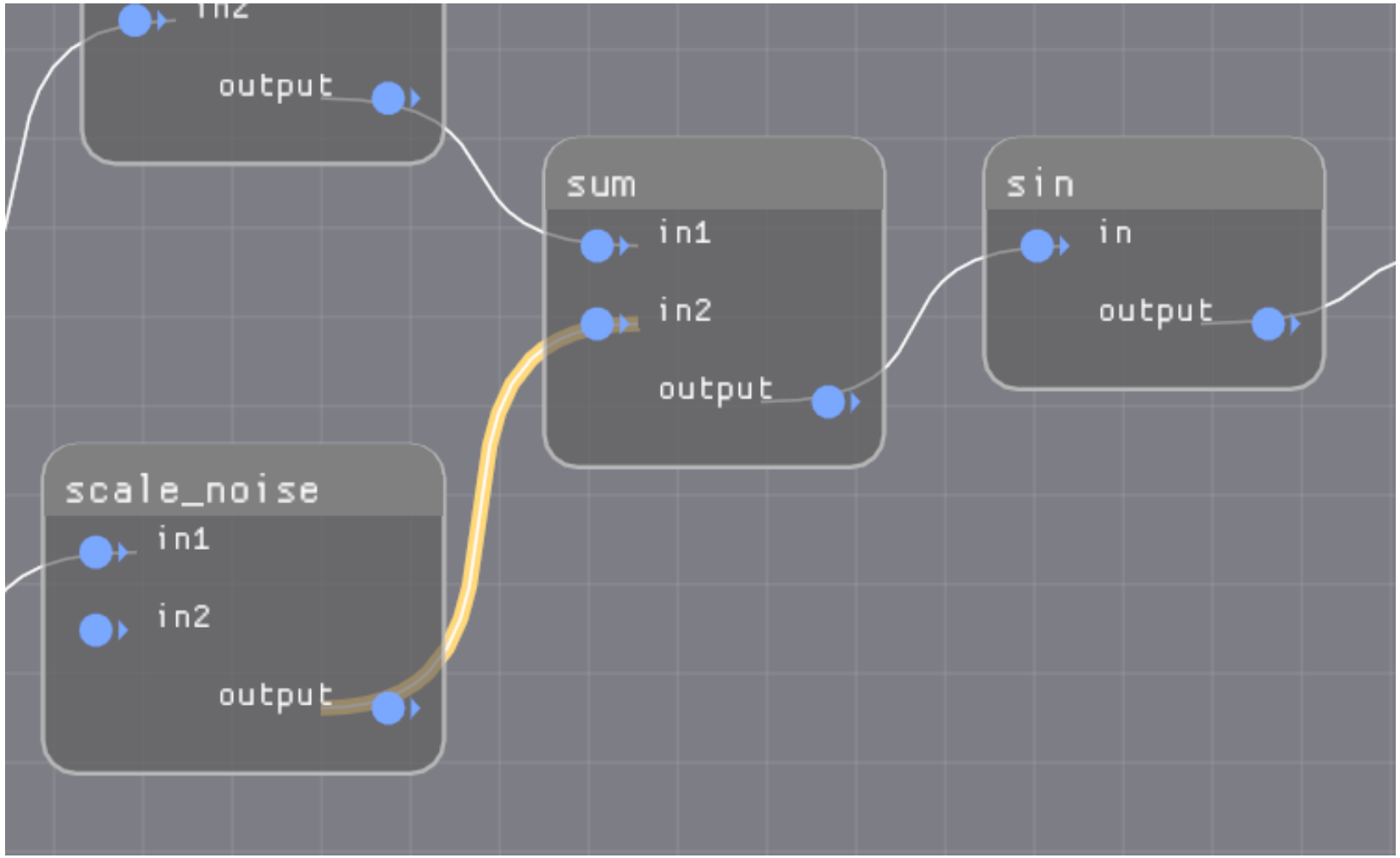


Current Features – Create & Save

- Start from scratch
- Save as .mtlx

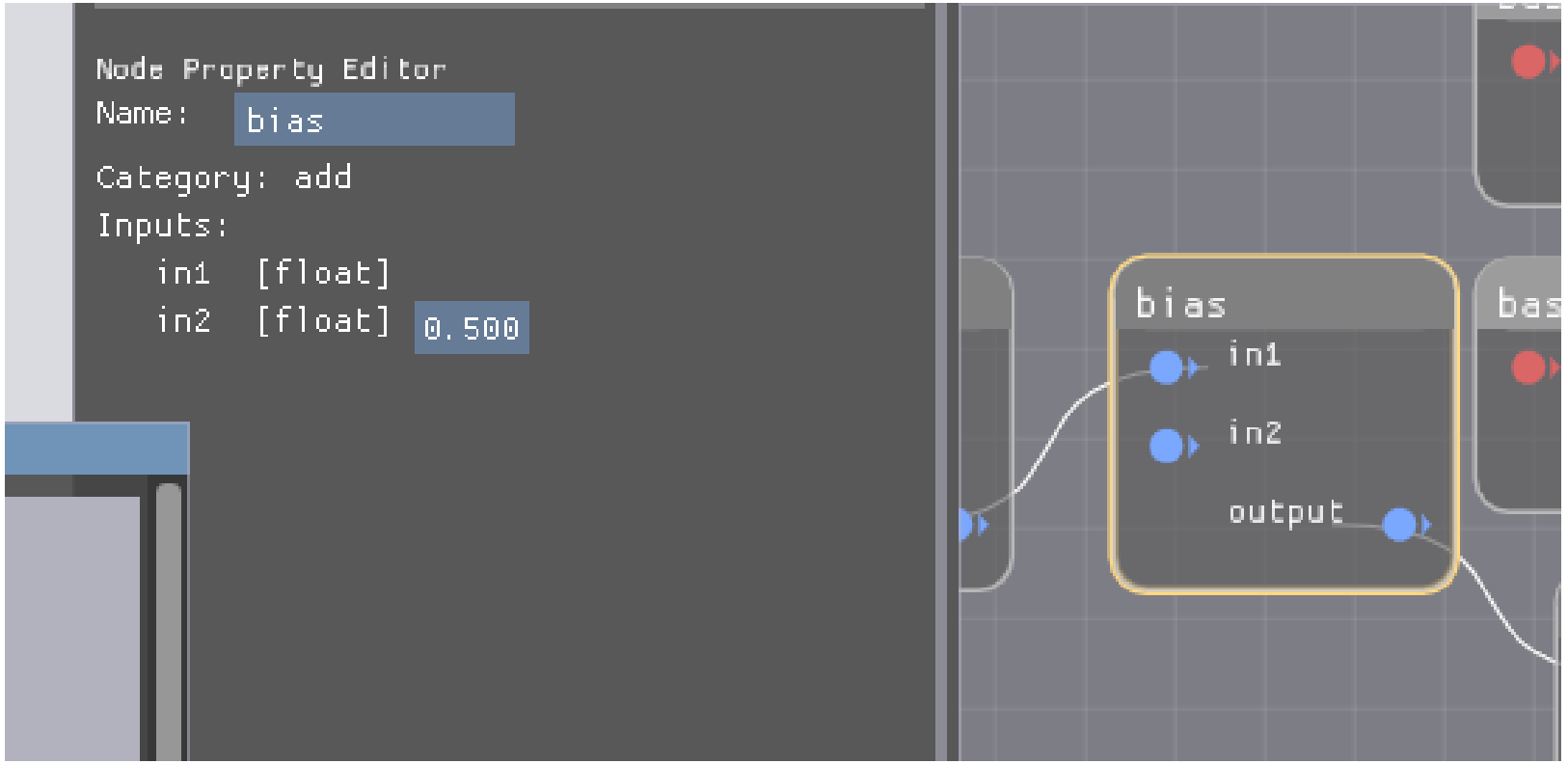


Current Features – Connect Nodes

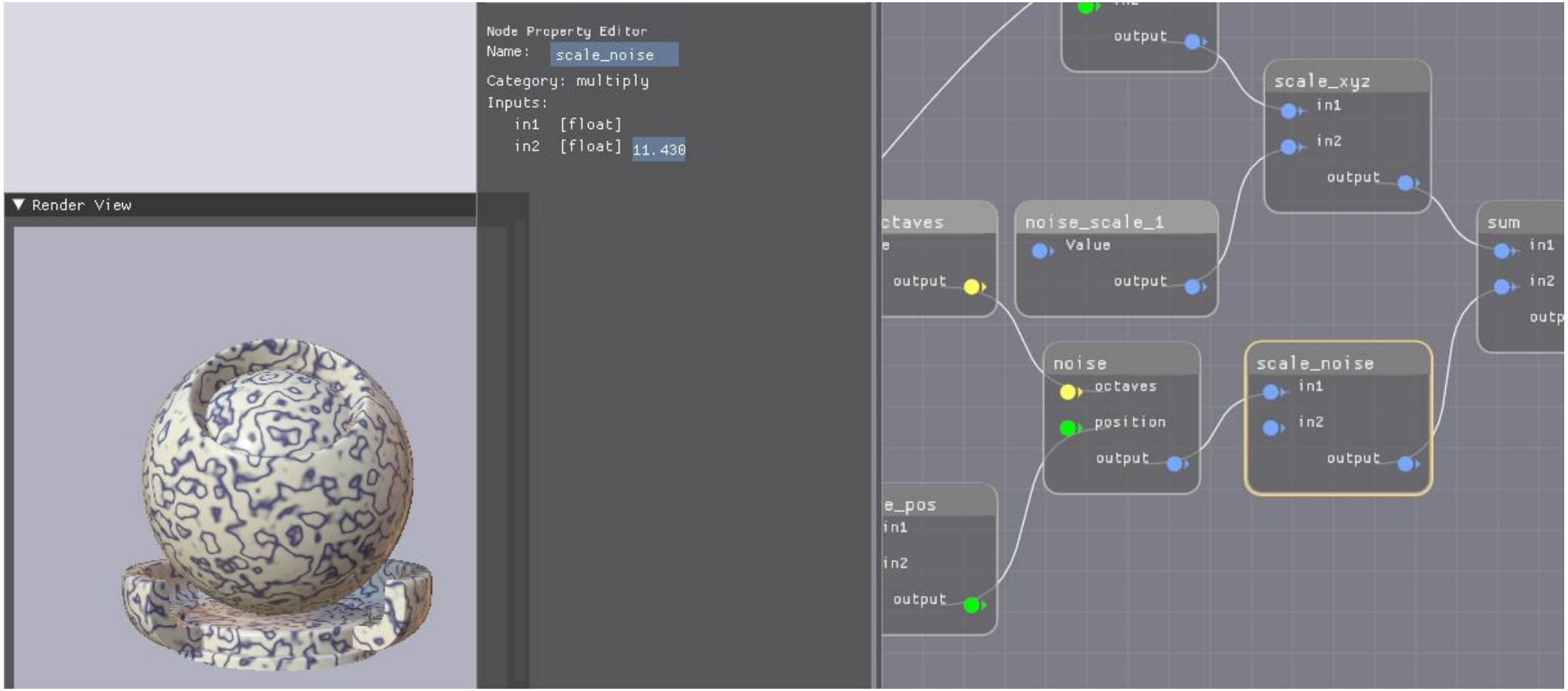


Current Features – Node Property Editor

- Change Node Name
- Edit Parameters



Current Features – Interactive Rendering



Future Features

- Thumbnail Render in Property Editor
- Hierarchical Traversal of Node Graphs
- Open Source Release (planned for Fall 2022)

Thanks to

Jonathan Stone

Eoghan Cunneen

Roger Cordes

André Mazzone

Dave Meny

Maggie Perlman

Magnus Pettersson

Rasmus Bonnedal

New Developments in MaterialX at Autodesk

Safe Harbor Statement

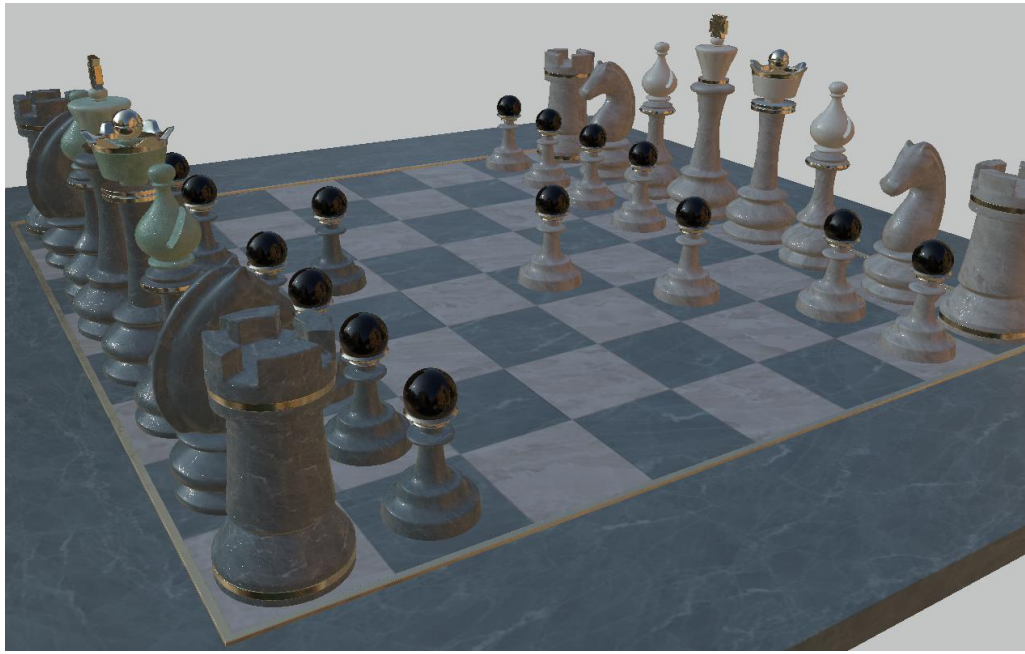
The presentations during this event may contain forward-looking statements about our outlook, future results and related assumptions, total addressable markets, acquisitions, products and product capabilities, and strategies. These statements reflect our best judgment based on currently known factors. Actual events or results could differ materially. Please refer to our SEC filings, including our most recent Form 10-K and Form 10-Q filings available at www.sec.gov, for important risks and other factors that may cause our actual results to differ from those in our forward-looking statements.

The forward-looking statements made in these presentations are being made as of the time and date of their live presentation. If these presentations are reviewed after the time and date of their live presentation, even if subsequently made available by us, on our website or otherwise, these presentations may not contain current or accurate information. We disclaim any obligation to update or revise any forward-looking statements.

Statements regarding planned or future development efforts for our products and services are not intended to be a promise or guarantee of future availability of products, services, or features but merely reflect our current plans and based on factors currently known to us. Purchasing decisions should not be made based upon reliance on these statements.

Contributions

- MaterialX Web
  - Introduced at ASWF OSD 2021.
  - Merged in early 2022 and is now hosted live on MaterialX ASWF.
- Vulkan ShaderGen
  - MaterialX::VkShaderGenerator
    - Based on Glsl ShaderGen
    - Generates Vulkan-compliant GLSL instead of SPIRV.
    - No dependency on Khronos tooling in MaterialX.
  - Shaders are validated using glslangValidator.
  - Plan to bring Vulkan rendering for testing and viewing to MaterialX.
    - Seeking help from Vulkan experts & contributors!



MaterialX Vulkan ShaderGen example with limited transmission support (under development).

Scene credit: Chessboard by Moeen Sayed and Mujtaba Sayed for SideFX.

Contributions

- Modernized the CMake code to generate downstream configuration files
  - Allows dynamic library use on the Windows platform

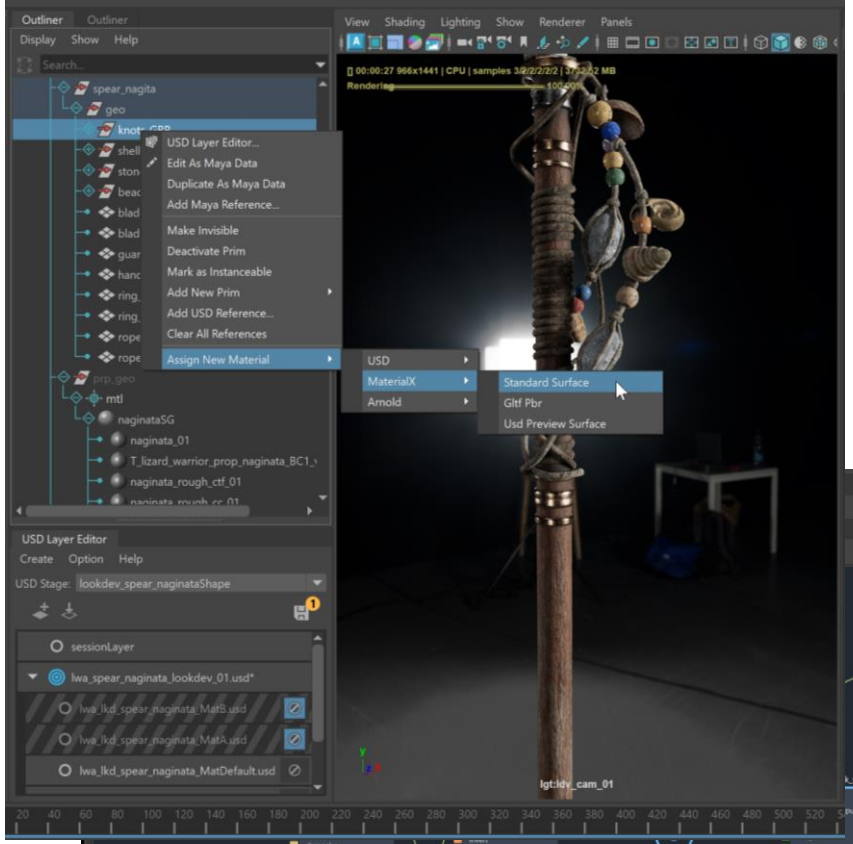
- Kept USD repository up-to-date with MaterialX releases and provided shader registry patches in order to
  - Improve support for boolean and color4 parameters
  - Support MaterialX metadata

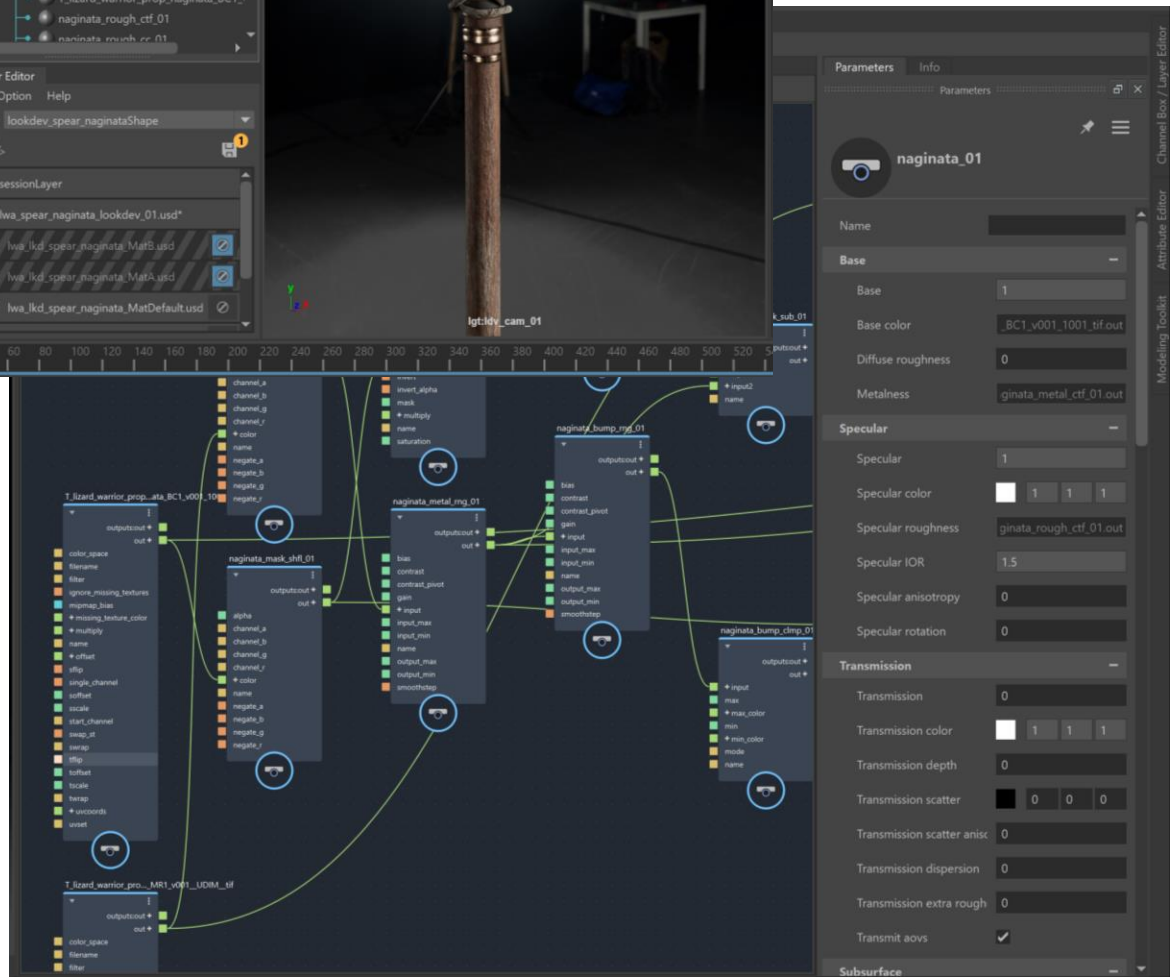
MaterialX in MayaUSD

- Supports MaterialX 1.38.4
- FIS lighting with Maya 2023.2 dome lights
- Tangent generator and color management nodes added when required
- Export/import/render Maya utility nodes and component level connections
  - place2dTexture
  - LookdevKit:colorCorrect

LookdevX – Material Editor

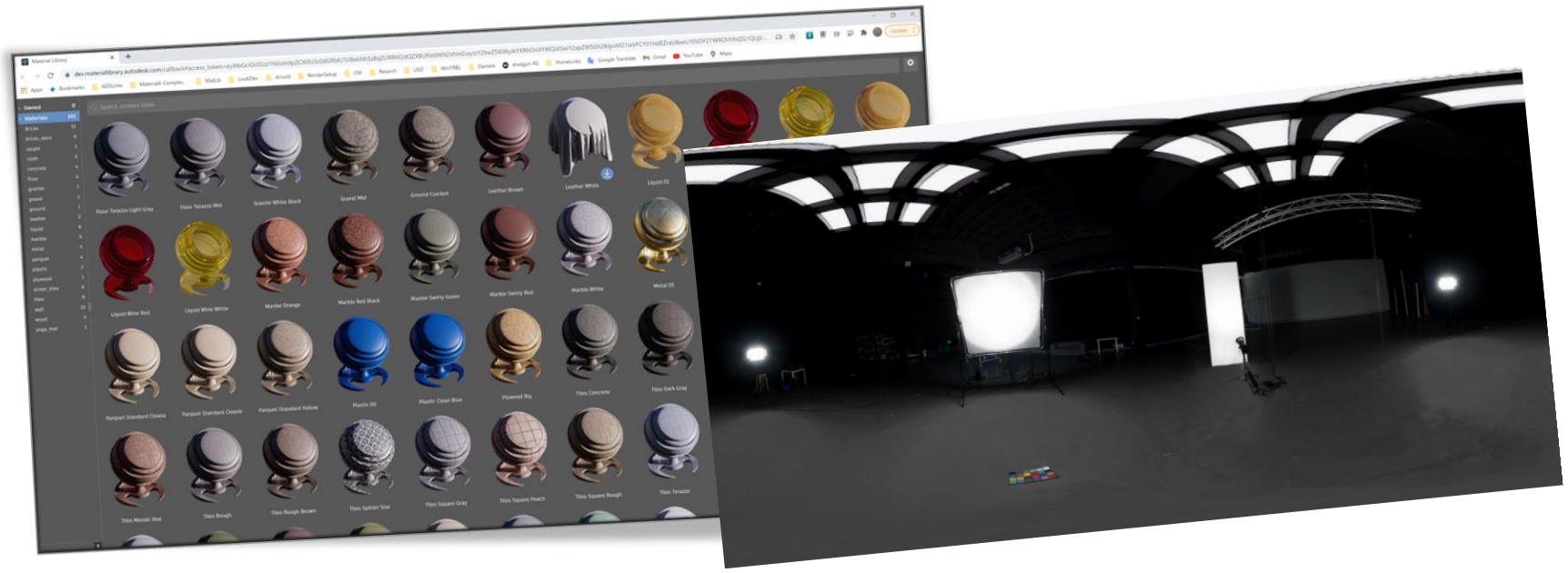
- Portable material workflows
  - MaterialX nodegraphs
  - MaterialX and Arnold nodegraphs encoded in USD
  - Can support different DCCs and renderers
- Artist Friendly
  - Fully featured graph management
  - Troubleshooting & Graph toolsets
  - Authoring & Publishing workflows
- Available soon in the Maya beta program
  - MaterialX and Arnold nodegraphs using Maya USD workflows





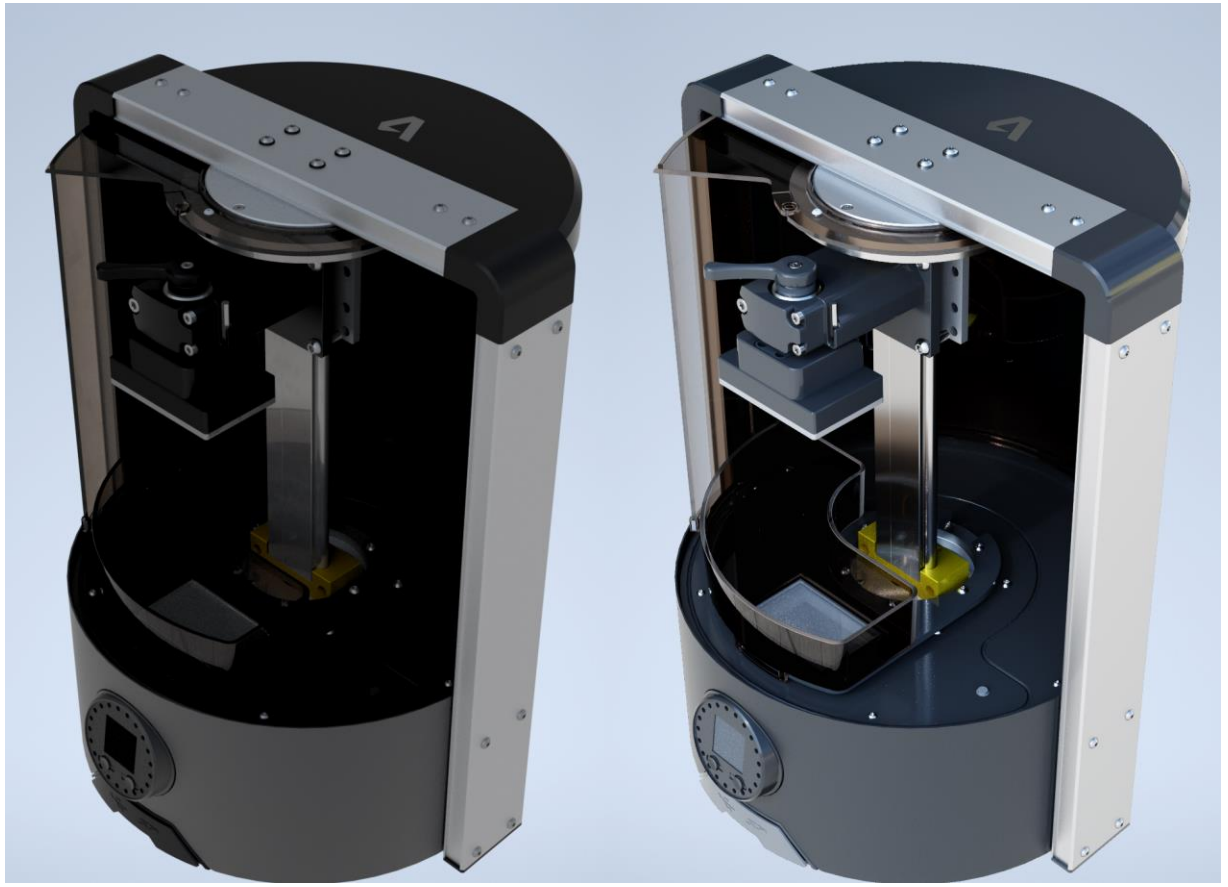
Autodesk Material Library

- Standard material library will help artists get up and running quickly
- Calibrated HDR and real-world lights



MaterialX in Autodesk Inventor

- Manufacturing Materials
  - Legacy materials to Standard Surface
  - Heavy use of procedurals, units and height maps.
- Improvements in appearance quality

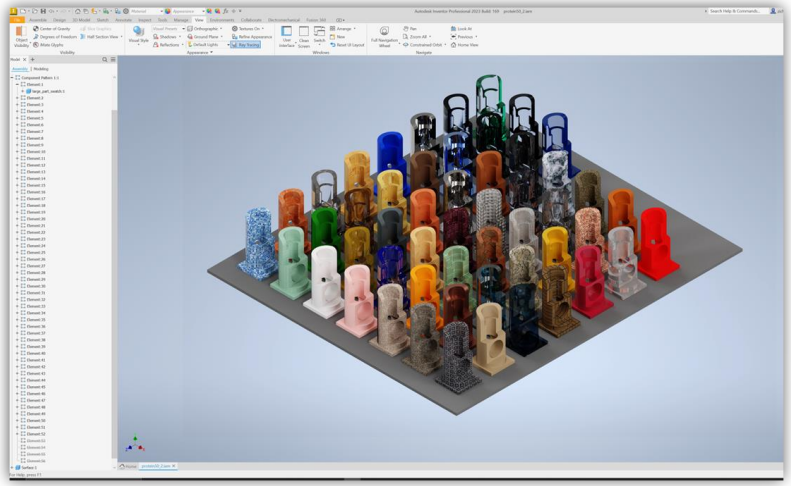


Model: Autodesk Ember 3D Printer.
Legacy materials (left) and Standard Surface (right)

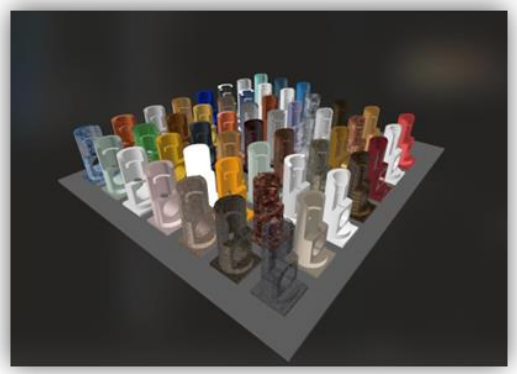
Material Translation using MaterialX

- Use Case: Export data for viewing.
  - Apple AR – USDZ
  - Web Viewers – glTF

- Shader Translation Graphs
  - Introduced in MaterialX 1.38.4
  - Convert from Standard Surface to usdPreviewSurface and glTF-pbr.
  - See ASWF OSD 2021 slides for details.
    https://www.materialx.org/assets/ASWF_OSD2021_MaterialX_slides_final.pdf



Autodesk Inventor 2023
GPU Ray Tracing



Exported to glTF (top)
and USDZ (bottom)

Model Credit: Test scene courtesy Roberto Ziche (Autodesk)

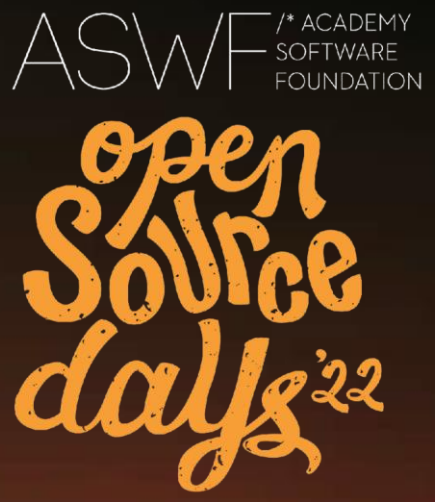
MaterialX in Arnold

- MaterialX supported since Arnold 5.1 (2018)
  - API and tools to exchange materials between DCCs in production
  - Arnold support both OSL code generation and Arnold native nodes in MaterialX documents

- Sneak peek: USD/MaterialX support
  - Compatible with Maya USD, LookdevX, and Houdini Solaris
  - Works in the Hydra render delegate and the Arnold procedural
  - New API in Arnold to generate inline OSL from a MaterialX node reference


AUTODESK Arnold

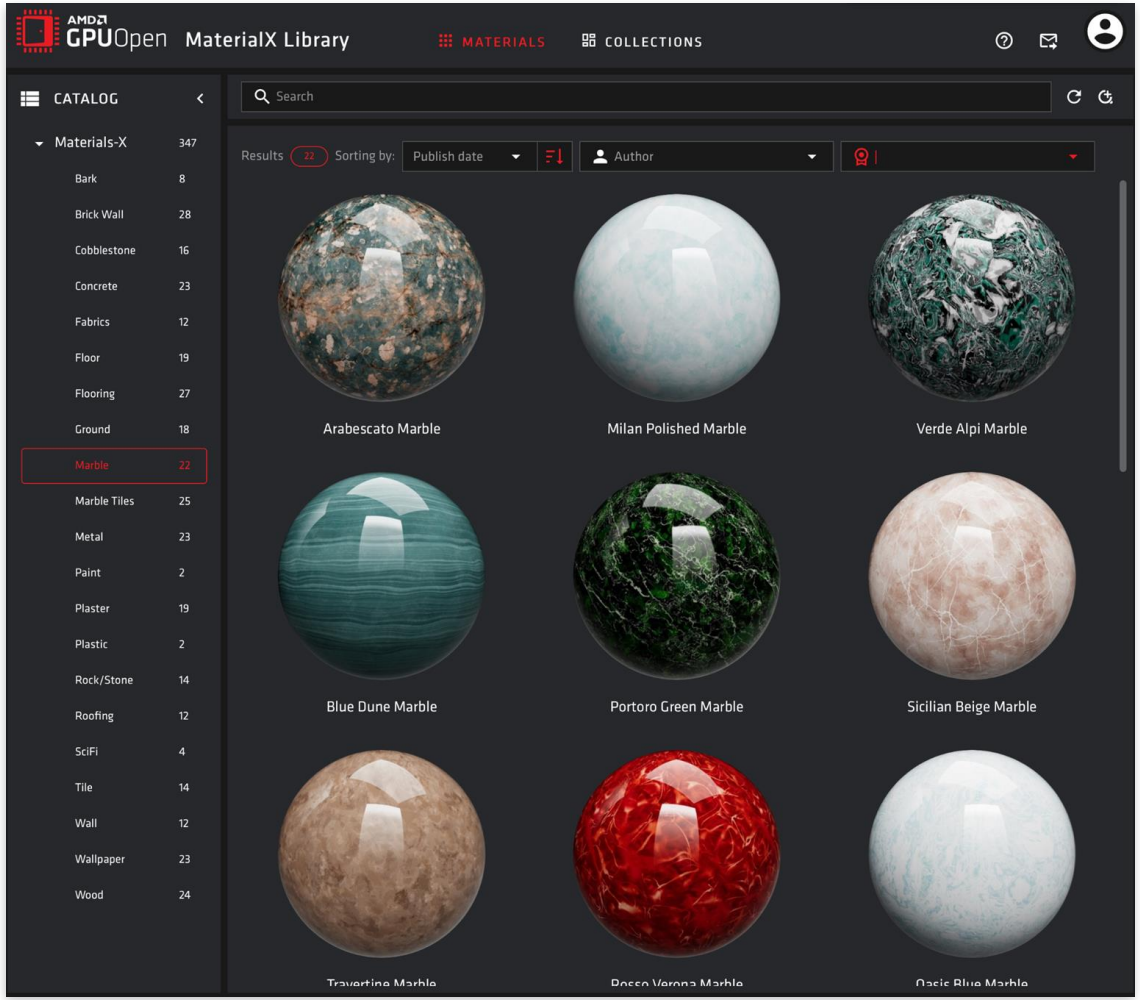

AUTODESK Arnold



AMD MaterialX Library (and more!)

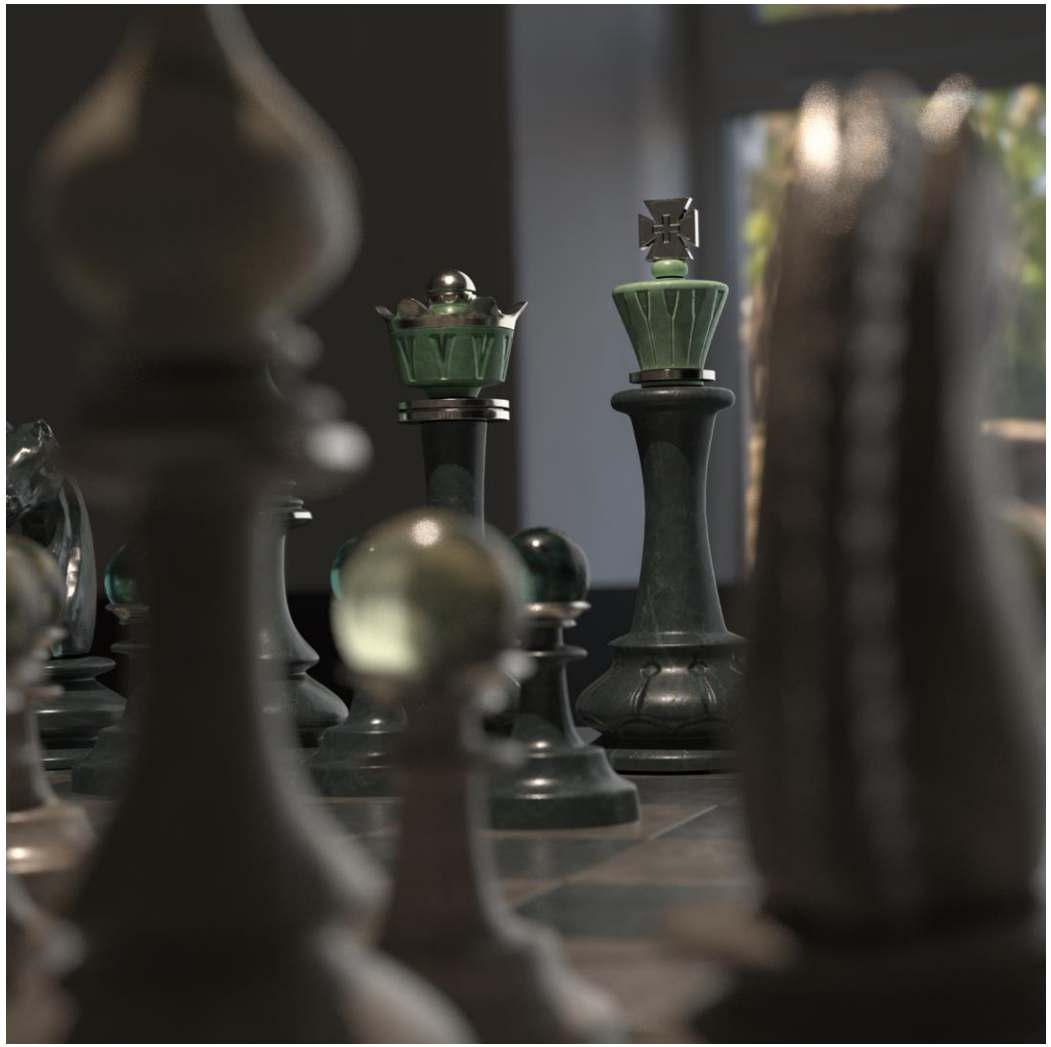
Free MaterialX Library on GPUOpen.com

- 347 permissive licensed materials… and growing
- Mostly PBR map style, some procedural
- REST API for integration
- Users contributions encouraged



GPU rendering of MaterialX with Radeon ProRender

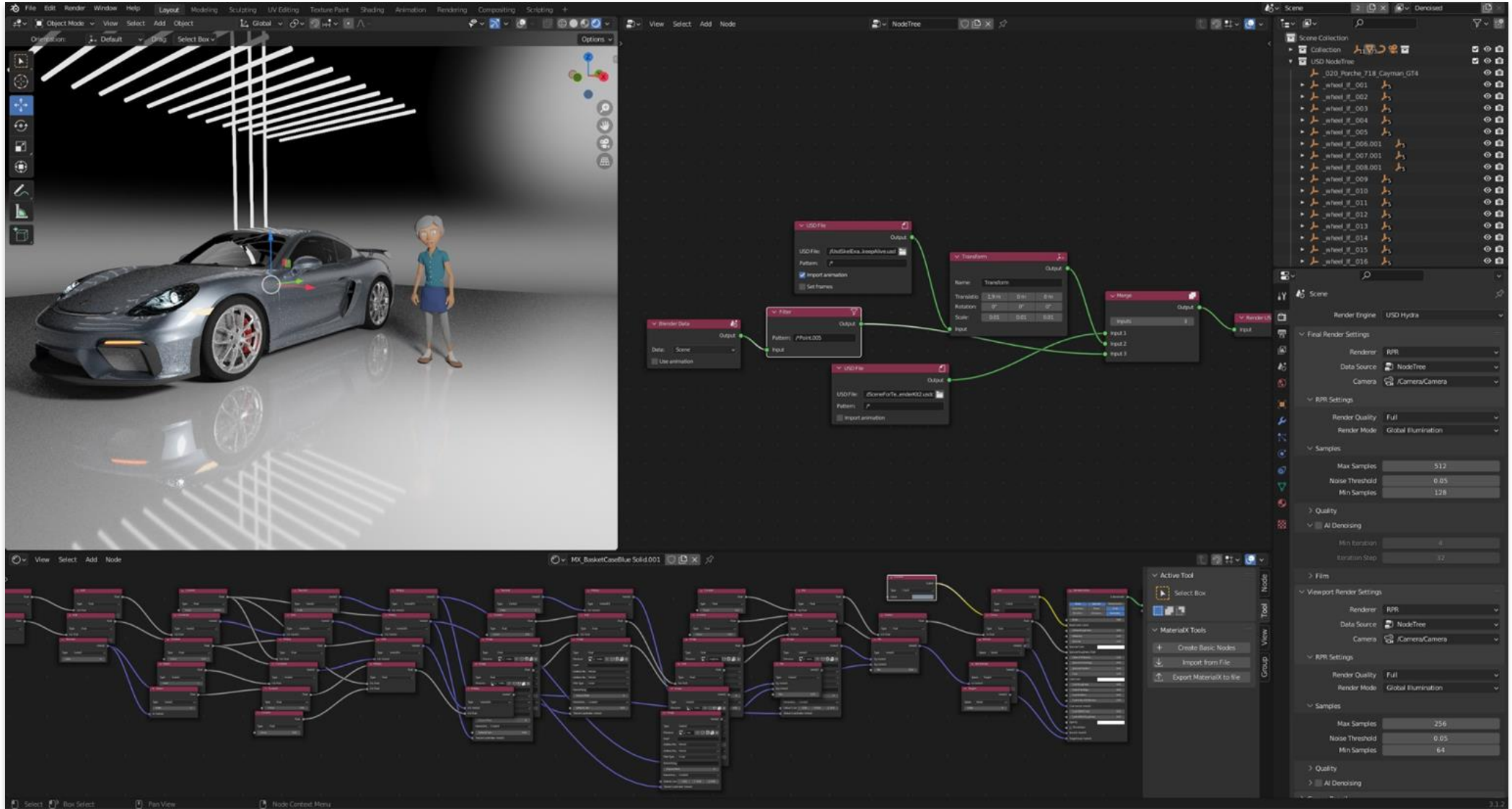
- Three rendering backends: Vulkan, OpenCL 1.2, Metal
- CPU/GPU rendering on AMD, and competitors.
- MaterialX 1.38
- Closed Source but very permissive licensing



Plugins and Blender USD / MaterialX

- Radeon ProRender focused on USD / MaterialX workflow
- Plugins for Autodesk Inventor, Houdini, Maya, Blender…
- Blender plugin with MaterialX and USD editing, working to integrate fully into Blender source.

Blender MaterialX and USD Workflow



MaterialX / glTF Update

Bernard Kwok (Khronos/MaterialX) and Pablo Delgado (Enscape)

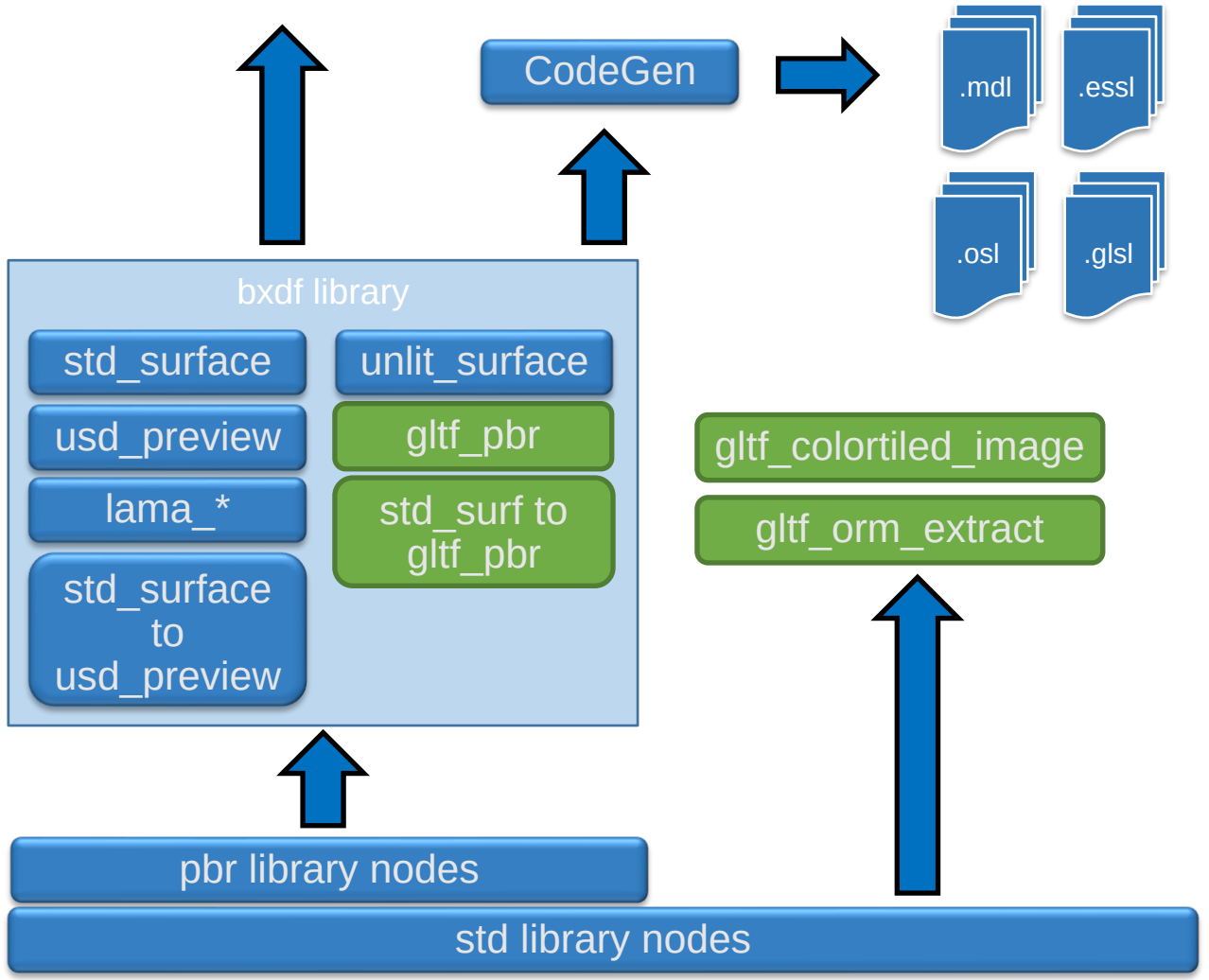
Outline

- glTF
  - glTF Materials in the MaterialX Ecosystem
  - glTF Interoperability
- Web Updates
  - Assets and Visualization
  - Rendering and Validation
- What's Next

Chessboard assets by Moeen Sayed and Mujtaba Sayed for SideFX.

Introducing glTF to MaterialX

- MaterialX v1.38.4
- New **gltf_pbr** shading model definition available (from Tobias Häußler)
- Building blocks: standardized node libraries
- In progress:
  - Shader translation graphs
  - Pattern definitions
    - Colored image
    - Image channel extract and combine.

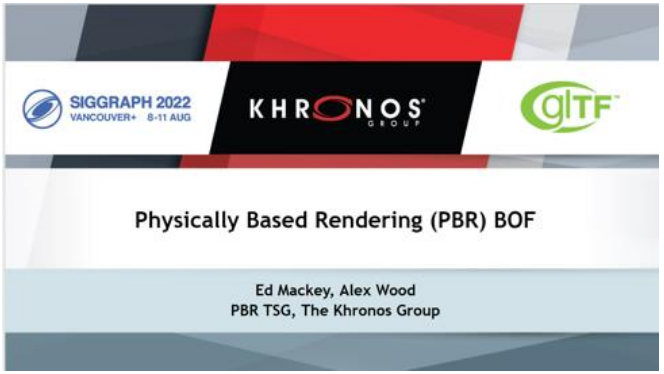


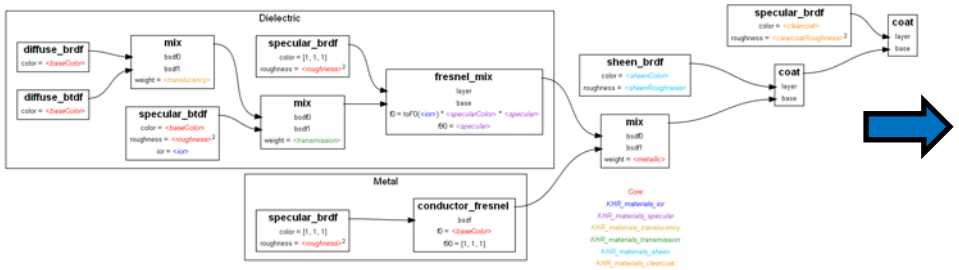
The glTF PBR Shading Model

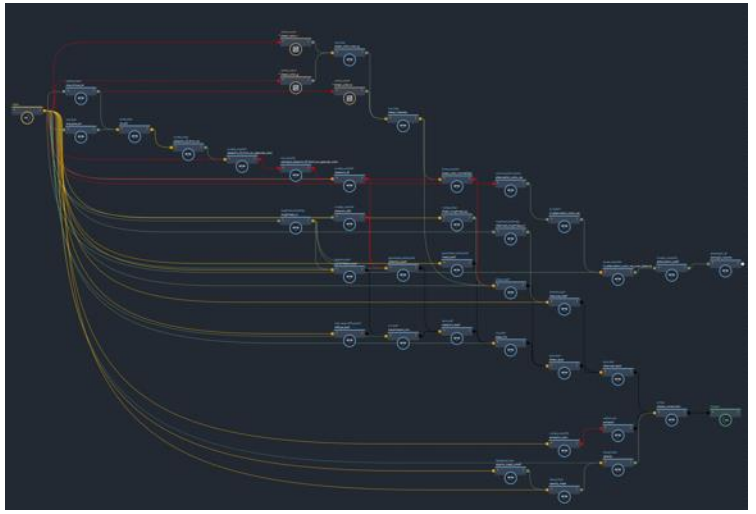
- KHR transmission, specular, ior, sheen, clearcoat, volume, emissive strength support.
- Iridescence, diffuse transmission to come.

- Help drive pbr improvements (e.g., thin-walled material support)

- Details: Khronos Siggraph 2022 BOF







Left: PBR Layering (Courtesy Tobias Häußler). Middle: MaterialX nodegraph,(Courtesy Nicolas Savva, Autodesk). Right: definition and UI (from MaterialXView). Chessboard assets by Moeen Sayed and Mujtaba Sayed for SideFX.

glTF PBR Support

- Use definition like any other shading model
  - Version, limits, UI hints metadata
  - Arbitrary nodegraph inclusion
  - Definition creation
  - Materials and assignments

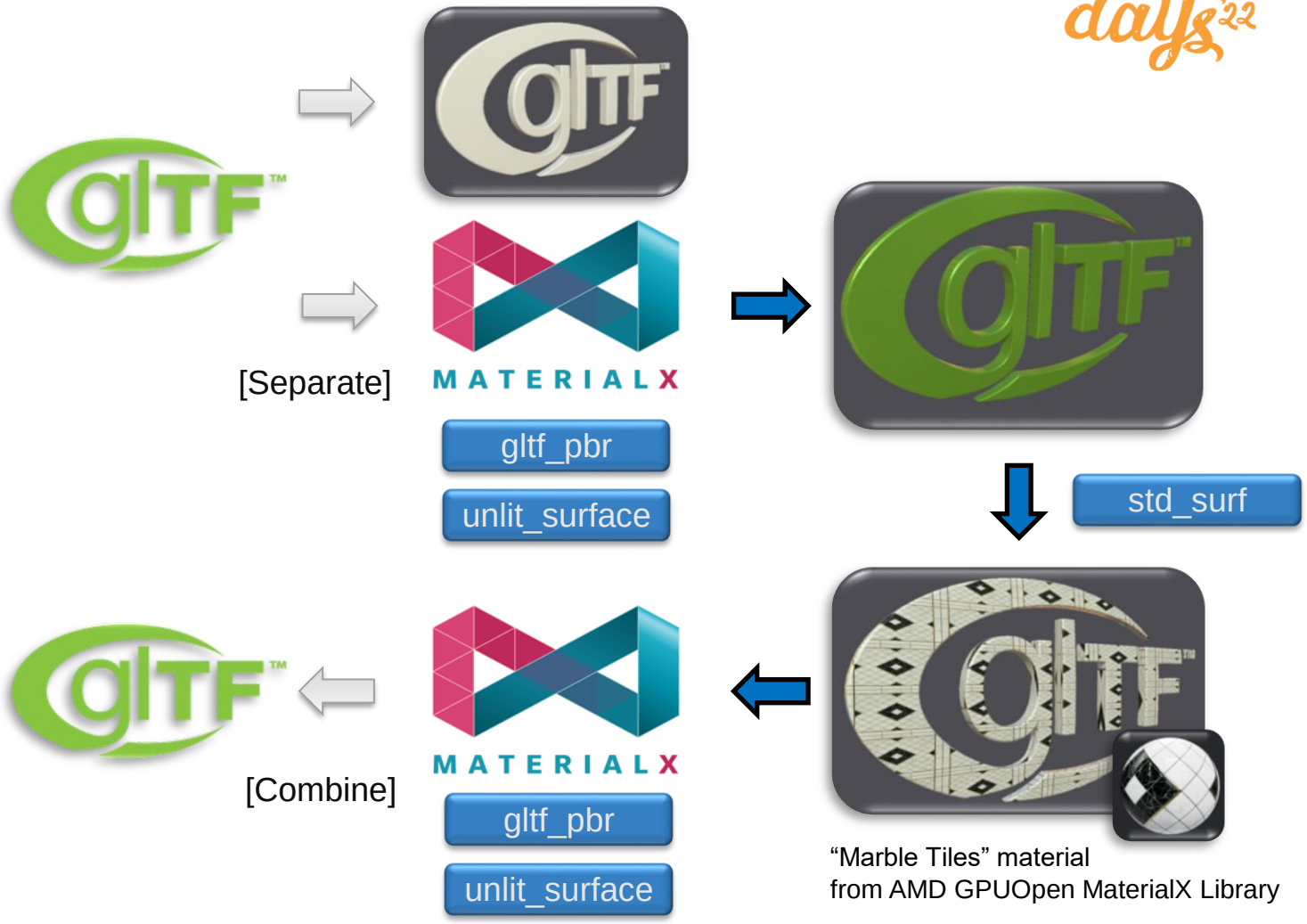
“King “ from chessboard assets by Moeen Sayed and Mujtaba Sayed for SideFX.

- Multiple backend shader generators (GLSL, ESSL, OSL, MDL etc.)

“boombox” sample model from Khronos. MDL sample viewer from NVIDIA. MaterialX example courtesy Ashwin Bhat, Autodesk)

MaterialX Workflows

- MaterialX workflows: authoring / lookdev / rendering
- *Separation of* materials, assignments, bindings
- Optional glTF translation
- Considerations:
  - Value resolves
  - UV Transform spaces
  - Path conventions
  - Binary resources
  - Units, color management



"Marble Tiles" material
from AMD GPUOpen MaterialX Library

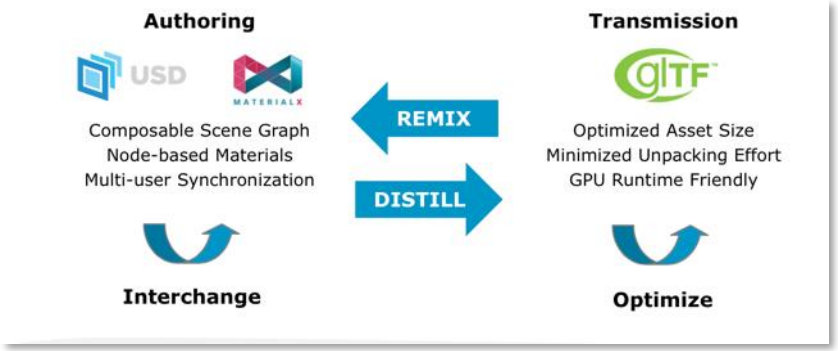
Logos from Khronos (2D and 3D), and MaterialX (2D)

Remapping / Distillation Workflow

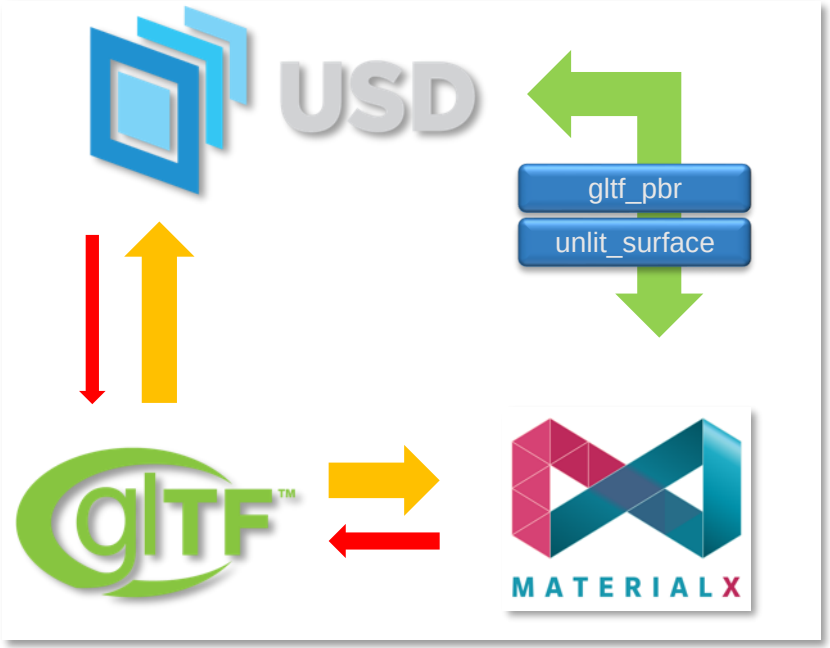
Rendered using: MaterialXView (left 3 courtesy Christian Robles, Autodesk) and Dassault Enterprise PBR Sample Renderer Demo Viewer (right-most). Model and textures for “King” from chessboard assets by Moeen Sayed and Mujtaba Sayed for SideFX.

USD / MaterialX / glTF Ecosystem

- **gltf_pbr** and unlit available in USD 22.08
- glTF's role, responsibilities, and workflows under "Metaverse" umbrella of discussions
- For authoring: USD / MaterialX *interoperability*
- For transmission: glTF *interoperation*

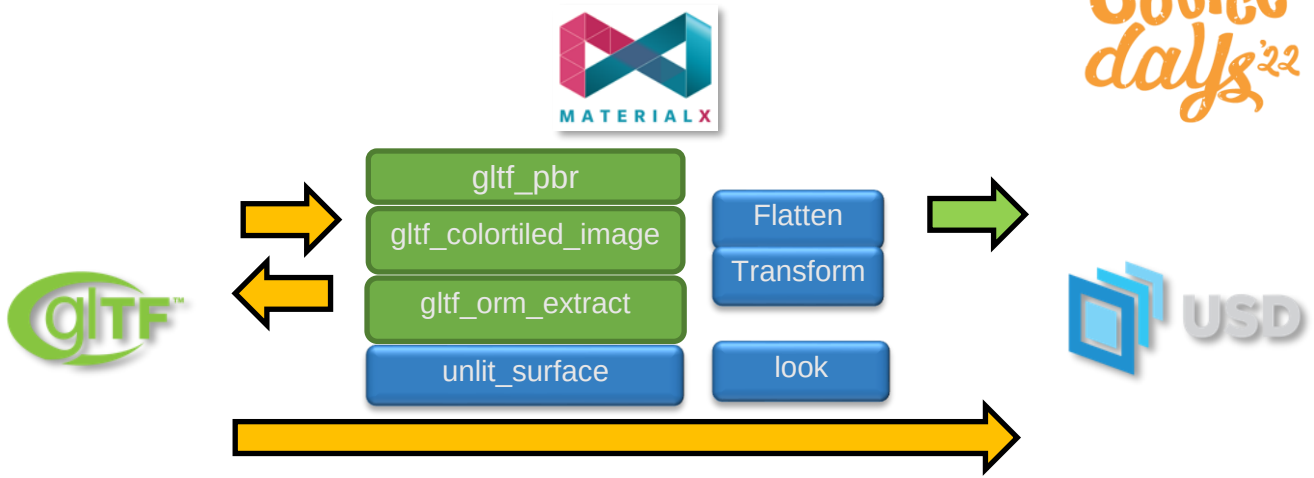


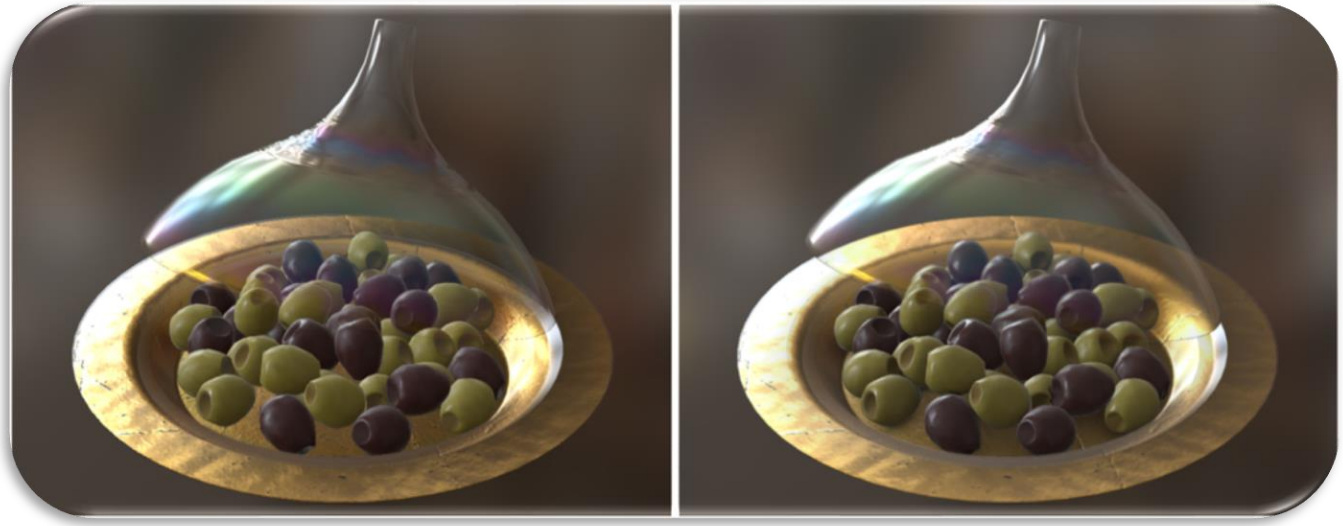
From: 2022 Metaverse Standards Forum



glTF Interoperation

- MaterialX gltf_pbr to glTF import / export in progress. (Khronos fork, guc)
  - Material variants, unlit to come
- guc: glTF to USD+MaterialX converter via MaterialX graph creation or direct to USD.
- Standardize target MaterialX graph and utilities in Khronos or core MaterialX





USD+MaterialX in hdStorm | glTF Sample Viewer

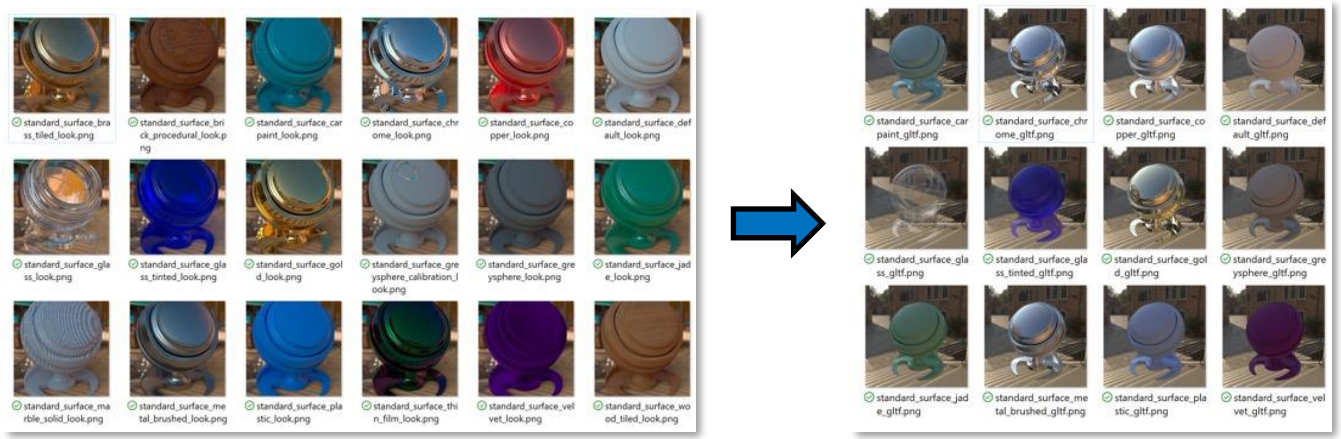
Wayfair's Iridescent Dish with Olives asset (CC BY). Example courtesy Pablo Delgado.

Shader Generation and Rendering Validation

- *glslangValidator* for ESSL and Vulkan shader validation

- Extend test assets (e.g. [glTF Sample Library]or [Cesium Examples] etc.)

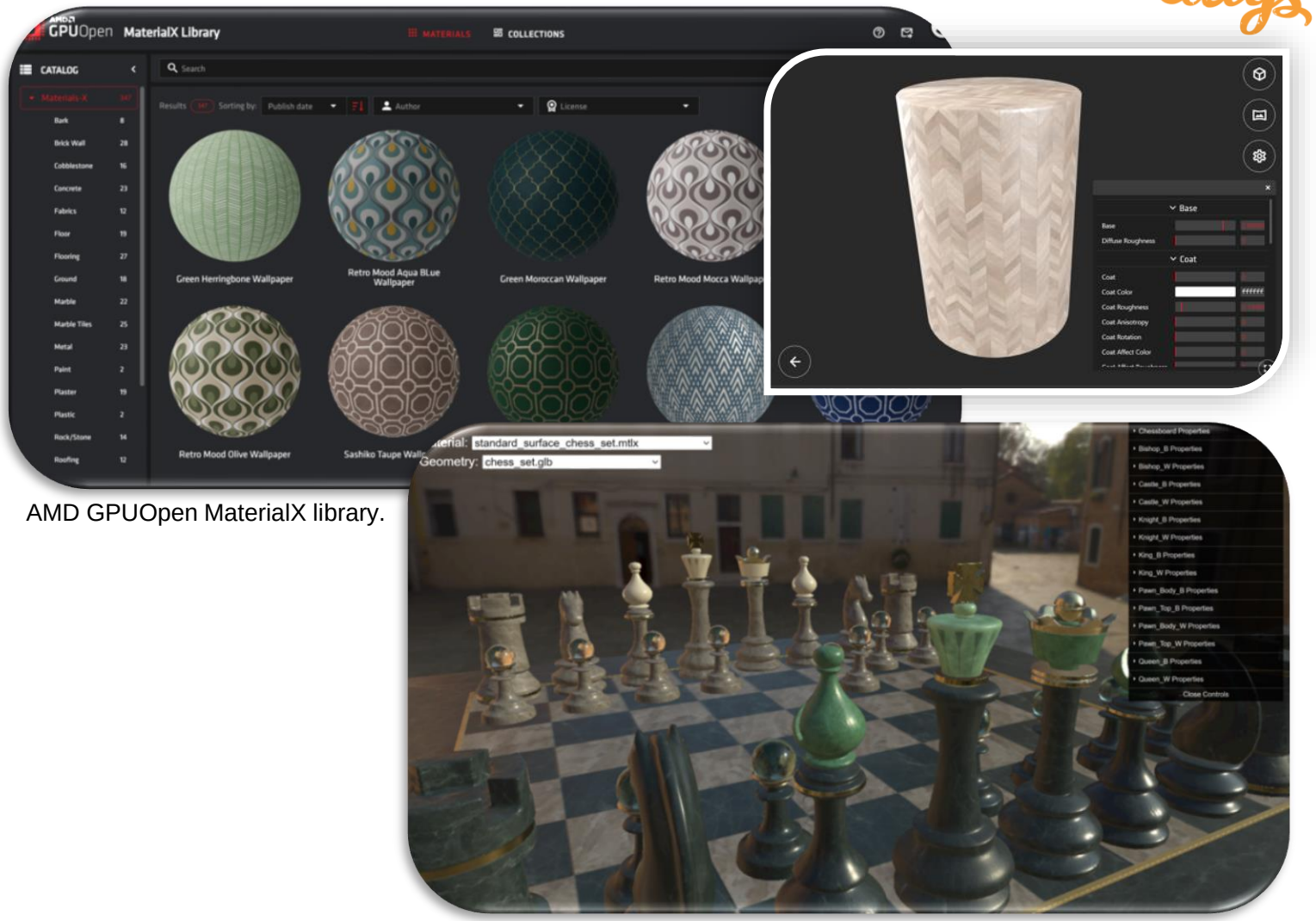
- Open Source Reference Rasterizers and Path tracers
  - glTF for OSL “testrender”



MaterialX example materials rendered using std surface (left) and gltf_pbr (right) to test conversion process. (Images courtesy Christian Robles, Autodesk)

Assets and Visualization

- Desktop Viewer
  - glTF geometry support
- MaterialX Web Viewer
  - Editability and Performance
  - Assignment support
- AMD GPUOpen MaterialX Web Viewer
  - gltf_pbr materials
- Packaging: npm registry



AMD GPUOpen MaterialX library.

Chessboard assets by Moeen Sayed and Mujtaba Sayed for SideFX.

Summary and Future

- glTF material model in MaterialX ecosystem
  - Current and future extensions
- Complete export workflow
  - Translation graphs, baking of procedural patterns
- Validation via open-source reference path tracers
  - Desktop: OSL “testrender”
  - Web: Dassault Renderer, other?
- Improve real-time performance / configurability
  - Shader generation variants
  - Stream requirements
- Extend transmission formats
  - Standard shading models and shader node graph support ?
  - Meta data: Color management, real world units ?

Chessboard assets by Moeen Sayed and Mujtaba Sayed for SideFX.

Acknowledgements

Alex Wood: AGI

Ashwin Bhat: Autodesk

Brian Savory: AMD

Christian Robles: Autodesk

Doug Smythe: ILM

Ed Mackey: AGI

Emmett Lalish: Google

Eric Chadwick: Wayfair

Jonathan Stone: LucasFilm

Lutz Kettner: NVIDIA

Magnus Petterson: IKEA

Moeen Sayed: Side Effects

Mujtaba Sayed: Side Effects

Nicolas Savva: Autodesk

Niklas Harrysson: Lumiere

Pablo Delgado: Enscape

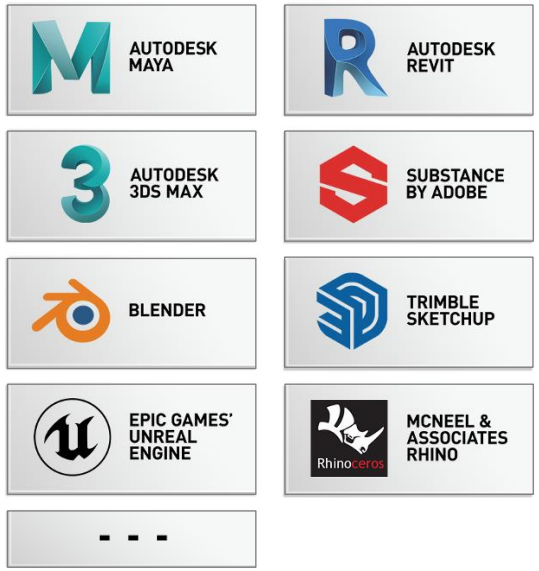
Tobias Häußler: Dassault Systèmes

MaterialX in NVIDIA Omniverse

Lutz Kettner
Director Rendering and Materials
NVIDIA

NVIDIA Omniverse

Enabling workflows

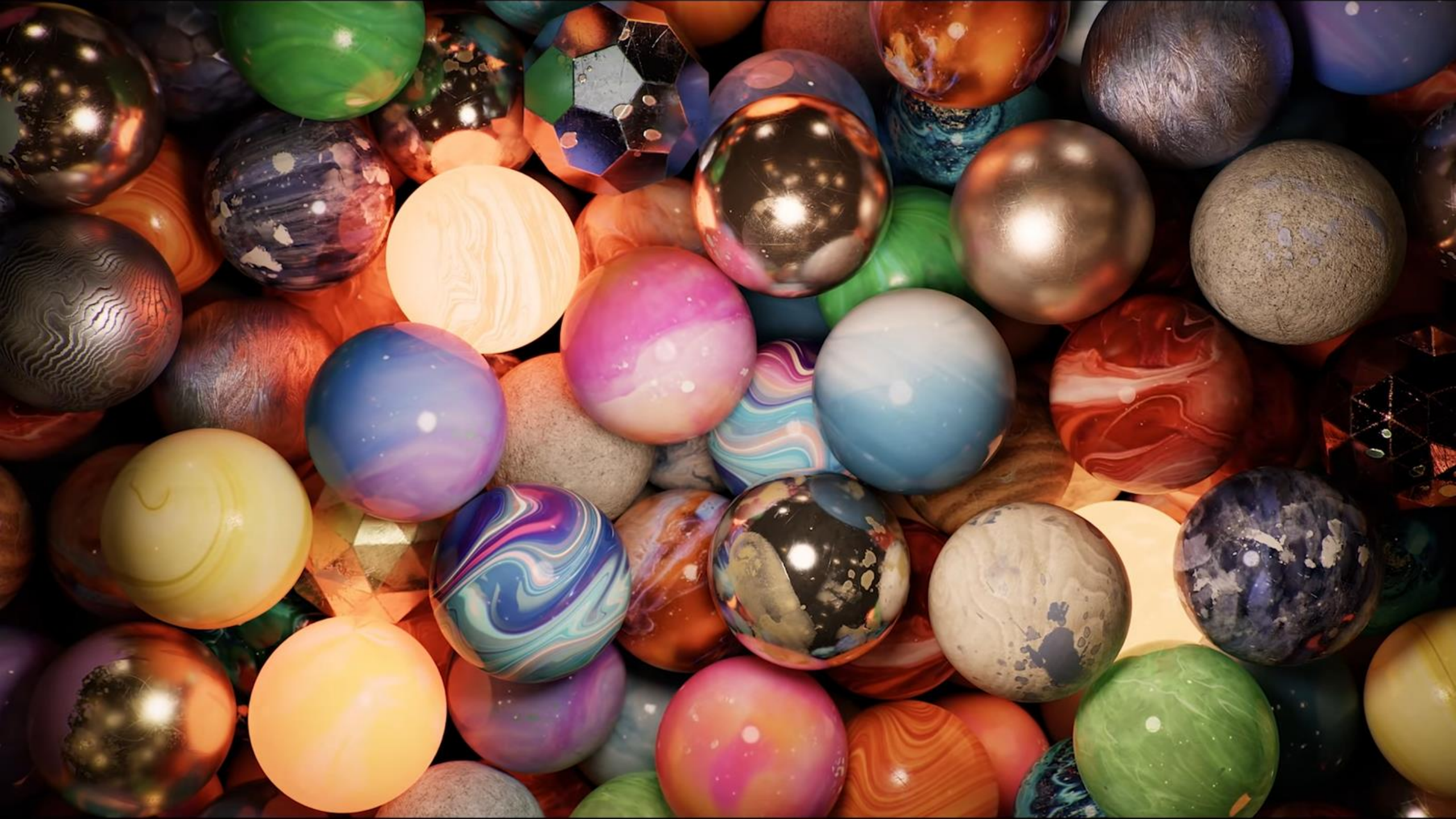


Top Industry Tools

Realtime Collaboration

Open Standards


Parker and Sons
GOUACHE
Pantaleoni
Acrylic Gouache
Lemon Green Hue
20.5ml
Parker and Sons
Titanium White
GOUACHE



MDL SDK
Open Source Release

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

23 releases shipped since SIGGRAPH 2018

BSD 3-clause license

Full MDL SDK

+ MDL Core API
+ MDL Core Definitions and more

Contributions welcome, standard CLA available

Omniverse Core Materials are Open Source too

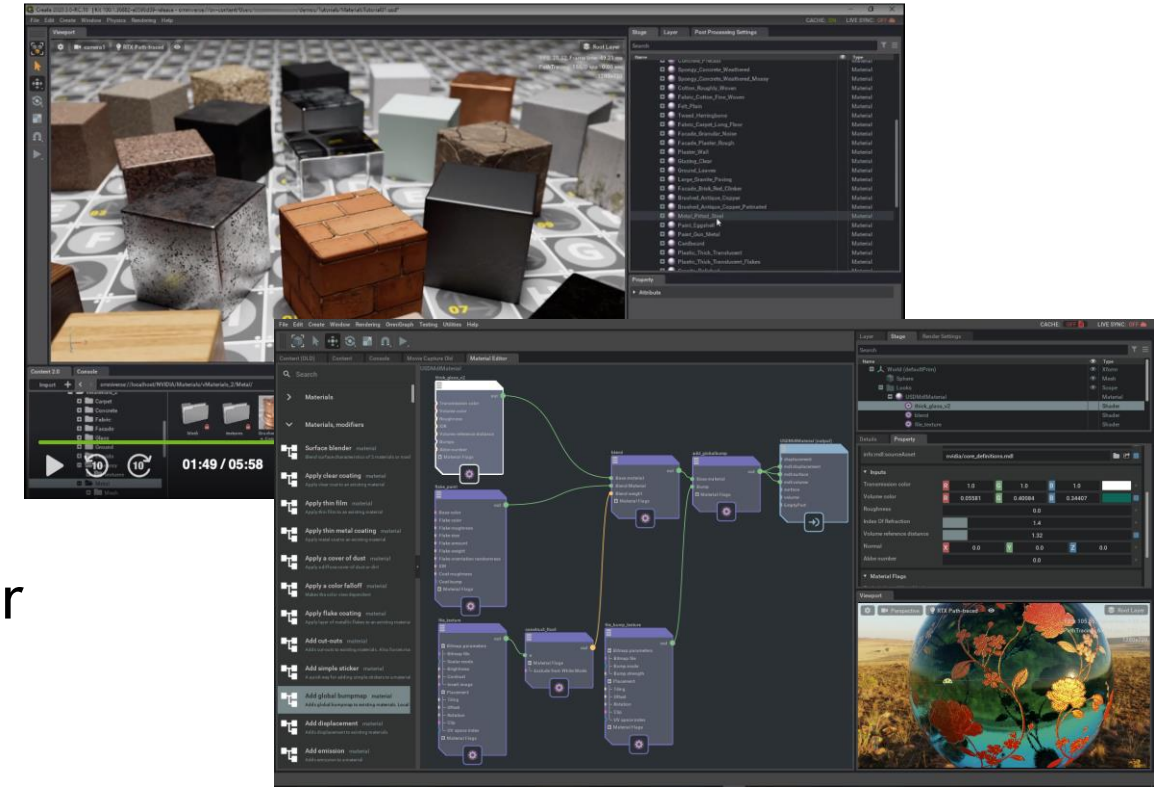
Shipping with Omniverse applications

BSD 3-clause license

OmniPbr family

OmniSurface family

Core definitions in the material graph editor



https://docs.omniverse.nvidia.com/prod_materials-and-rendering/prod_materials-and-rendering/materials.html


MATERIALX

+


NVIDIA
MDL

MaterialX and ShaderGen

MaterialX

An open standard for network-based CG object looks
Originally developed by Lucasfilm

https://www.materialx.org/

MaterialX Physically-Based Shading Nodes

Data types, nodes, and node graphs for layered physically-based shading

https://www.materialx.org/assets/MaterialX.v1.38.PBRSpec.pdf

ShaderGen

Transforms the agnostic MaterialX descriptions into executable code for a specific renderer
Contribution by Autodesk

https://github.com/materialx/MaterialX/blob/master/documents/DeveloperGuide/ShaderGeneration.md

MaterialXGenMdl

Library for MDL Code Generation

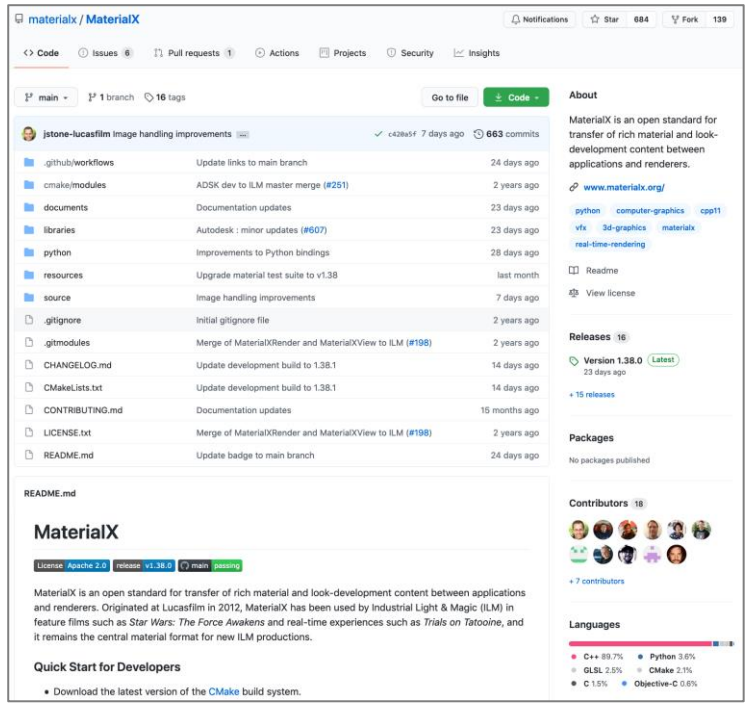
Official part of MaterialX 1.38 repository

Open Source Release

https://github.com/materialx/MaterialX

Joint development from

AUTODESK & NVIDIA



Kit Dev App 104.0+mr14860.130097.5dae3064.teamcity - C:\Users\jjordan\Documents_usd\mtlx\lucasfilms\scene3.usd*

File Edit Create Window Rendering Help

CACHE: ON LIVE SYNC: OFF

Viewport

Perspective RTX Path-traced

Root Layer

FPS: 109.20, Frame time: 9.16 ms
NVIDIA Quadro RTX 8000: 4.1 GiB used, 42.7 GiB available
Host Memory: 16.5 GiB used, 15.1 GiB available
PathTracing: 64/64 spp : 3.54 sec
1280x720

Stage Layer Render Settings

Search

Name	Type
DomeLight	DomeLight
Cube	Mesh
scene_01	Xform
Sphere	Mesh
Looks	Scope
DomeLight	DomeLight
scene_02	Xform
Sphere	Mesh
Looks	Scope
DomeLight	DomeLight
scene_03	Xform
Sphere	Mesh
Looks	Scope
DomeLight	DomeLight
scene_04	Xform
scene_05	Xform
Sphere	Mesh
Looks	Scope
Sphere	Scope
Materials	
MaterialX_Graph	Material
DomeLight	DomeLight

Property

Add MaterialX_Graph

Prim Path /World/scene_05/Looks/Sphere/Materials/MaterialX_Graph

Instanceable

Material and Shader

Inputs

inputs:base	0.8
inputs:diffuse_roughness	0.0
inputs:specular	1.0
inputs:specular_color	R 1.0 G 1.0 B 1.0
inputs:specular_IOR	1.5
inputs:specular_anisotropy	0.0
inputs:specular_rotation	0.0
inputs:transmission	0.0
inputs:transmission_color	R 1.0 G 1.0 B 1.0
inputs:transmission_depth	0.0
inputs:transmission_scatter	R 0.0 G 0.0 B 0.0
inputs:transmission_scatter_anisotropy	0.0
inputs:transmission_dispersi	0.0

Console Content

Import C:/Users/jjordan/Documents/_usd/mtlx/sandstone/

Search

- A_Beautiful_Game
- CarPaint_SingleCol
  - textures
- lucasfilms
  - BB8_Standard_S
  - R2D2_Standard_
  - sandstone
  - shaderball_checker
  - standard_surface
  - TH_Church_Bricks
  - TH_Medieval_Wooc
  - TH_planks_brown_1

textures goegap.hdr scene.usda TH_Sandstone_Cra....mtlx TH_Sandstone_Cra...b.zip

Checkpoints

Location does not support Checkpoints.

MaterialX import in Omniverse

Based on usdMtlx plugin in the USD SDK

Two supported representations in USD

1. A reference with an asset path to a MaterialX file

```
references = @./R2D2_Standard_Surface/R2D2_Standard_Surface.mtlx@</MaterialX>
```

2. Flattened to a set of UsdShade nodes

```
uniform token info:id = "ND_swizzle_vector2_float"
```

Based on MaterialX v1.38

MaterialX import in Omniverse

Based on usdMtlx plugin in the USD SDK

Import re-creates a MaterialX document from USD, supports both representations

Uses MaterialXGenMdl to create a corresponding MDL material stored on disc

MDL material is referenced in a single new UsdShade node with sourceType “mdl”

```
def Material ”sample“ {
  token outputs:mdl:surface.connect = </sample/flex_material.outputs:out>
  ...
  def Shader "flex_material“ {
    uniform token info:implementationSource = "sourceAsset"
    uniform asset info:mdl:sourceAsset = @nvidia/core_definitions.mdl@
    uniform token info:mdl:sourceAsset:subIdentifier = "flex_material“
    token outputs:out
    ...
  }
}
```

https://developer.nvidia.com/usd/MDLschema

MaterialX import in Omniverse

Based on usdMtlx plugin in the USD SDK

Details of the workflow TBD

- We don’t want to change a USD file just because we opened it, the initial conversion must be transitory, until we start editing it
- Integration into the material graph editor, e.g., two contexts, mtlx and mdl

Expected availability in Q4 2022


Chessboard assets by Moeen Sayed and Mujtaba Sayed for SideFX.


Chessboard assets by Moeen Sayed and Mujtaba Sayed for SideFX.


Chessboard assets by Moeen Sayed and Mujtaba Sayed for SideFX.

MaterialXGenMdl

Next steps

Very complete open-source reference implementation of the MaterialX standard nodes

Upgrading from MDL 1.6 to MDL 1.7 completes outstanding limitations

- full layering support for <sheen_bsdf>
- emissive volumes
- arbitrary weights with <add_bsdf>
- non-uniform weights with <mix_edf>

<subsurface_bsdf> node implementation (done!)

See Also

Material Workflows in NVIDIA Omniverse

Lutz Kettner, Director Rendering and Materials, NVIDIA

Francis Liu, Sr. Product Manager for Materials and Rendering for NVIDIA Omniverse, NVIDIA

Thursday 9:00 AM – 9:50 AM

Conference Center West Level 1

Room 121/122

Acknowledgements

Kai Roehmer

Derek Haase

Charles Anderson

Francis Liu

Jan Jordan

ASWF /* ACADEMY SOFTWARE FOUNDATION

open source days '22

MaterialX Closures for OSL

Adrien Herubel, Sr Manager, Autodesk/Arnold
Chris Kulla, Principal Rendering Programmer, Epic Games,

Presentation outline

- Why introduce a new set of closures ?
- Closure description and OSL testrender implementation status
- Ubershader example
- Next steps

Why introduce a new set of closures ?

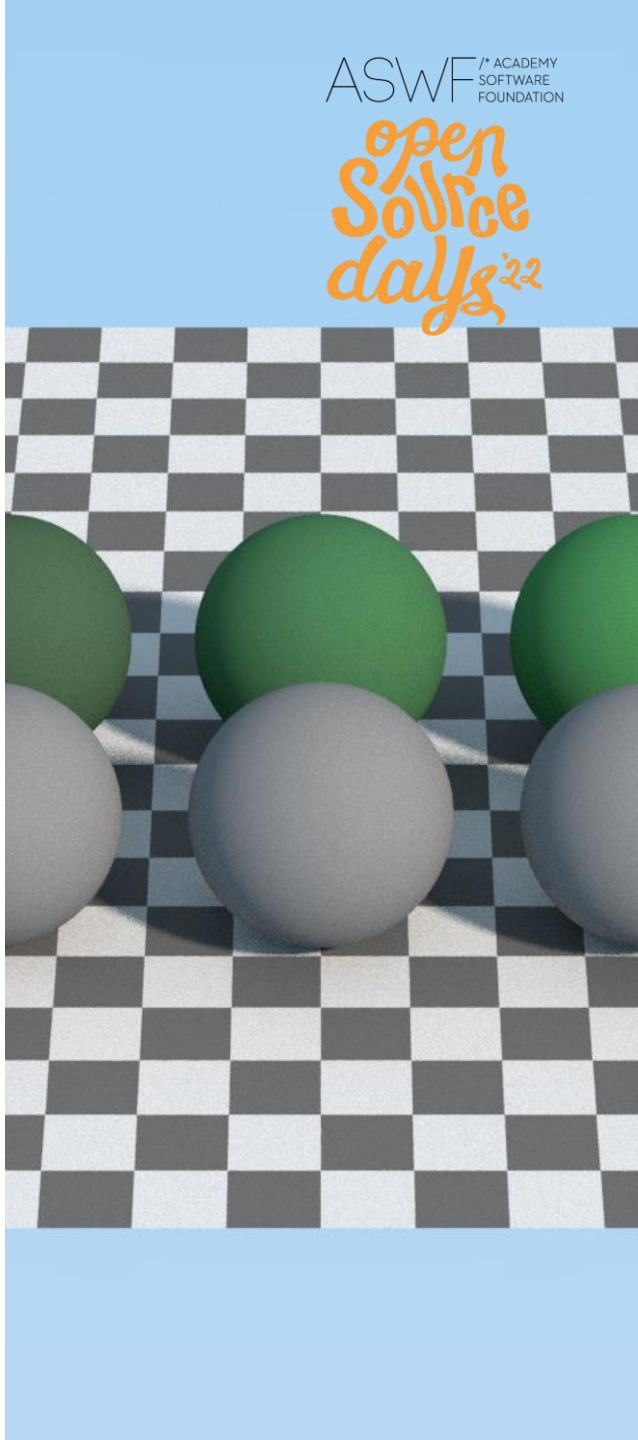
- Existing predefined closures are not all relevant, and loosely implemented in OSL-enabled renderers
- MaterialX generates OSL/GLSL
  - The OSL code generator should rely on a widely supported and fully featured set of closures
- MaterialX and OSL developers worked on a common set of closures capable of representing modern materials

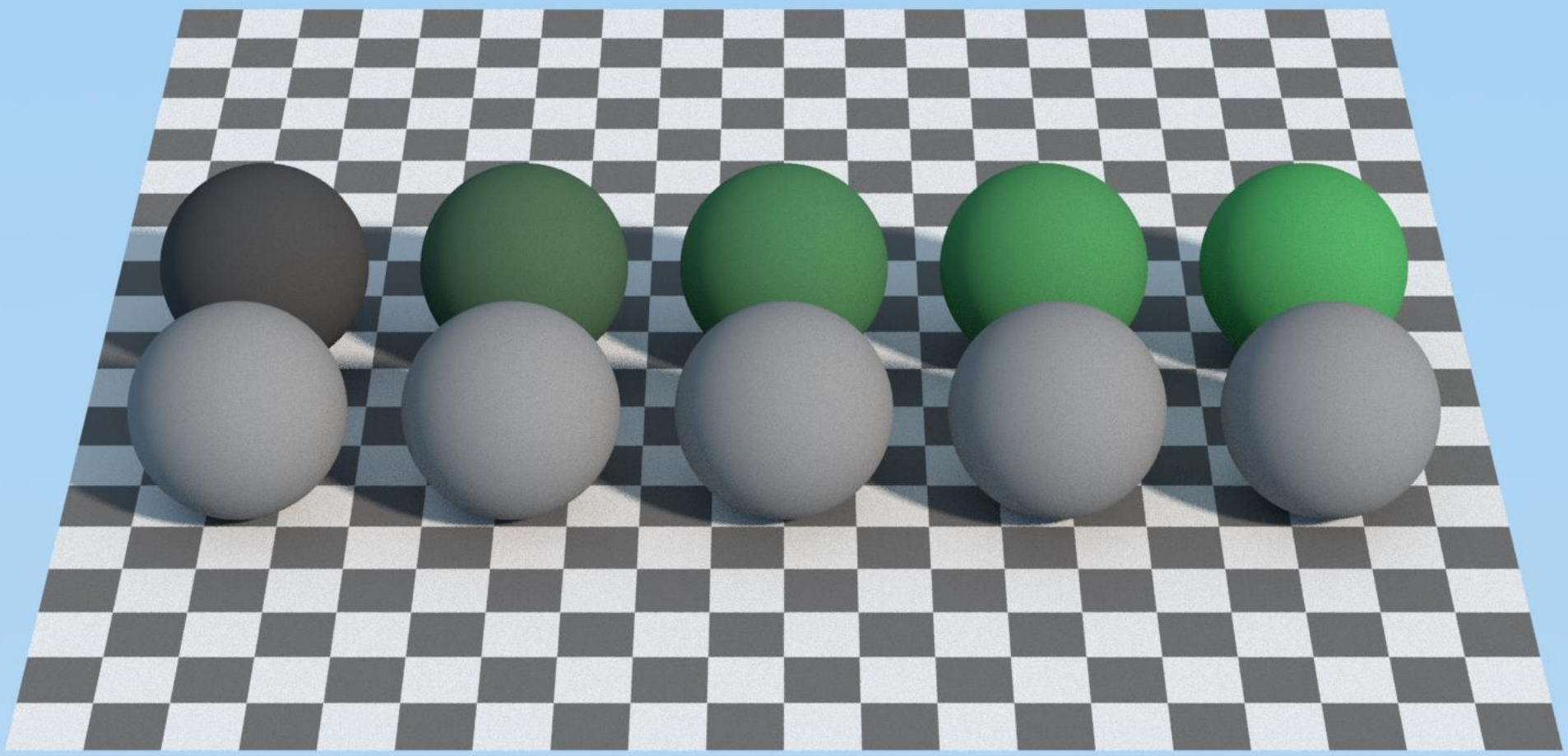
Project status

- The new set of closures is specified and included in OSL as a header
  - See MaterialX#614 and OpenShadingLanguage#1371 for the full discussion
- Ongoing work to provide implementations for all closures in OSL testrender
  - OSL#1533 OSL#1536 OSL#1537 OSL#1538 OSL#1539 OSL#1541 OSL#1542 OSL#1543 OSL#1547

Diffuse BSDFs

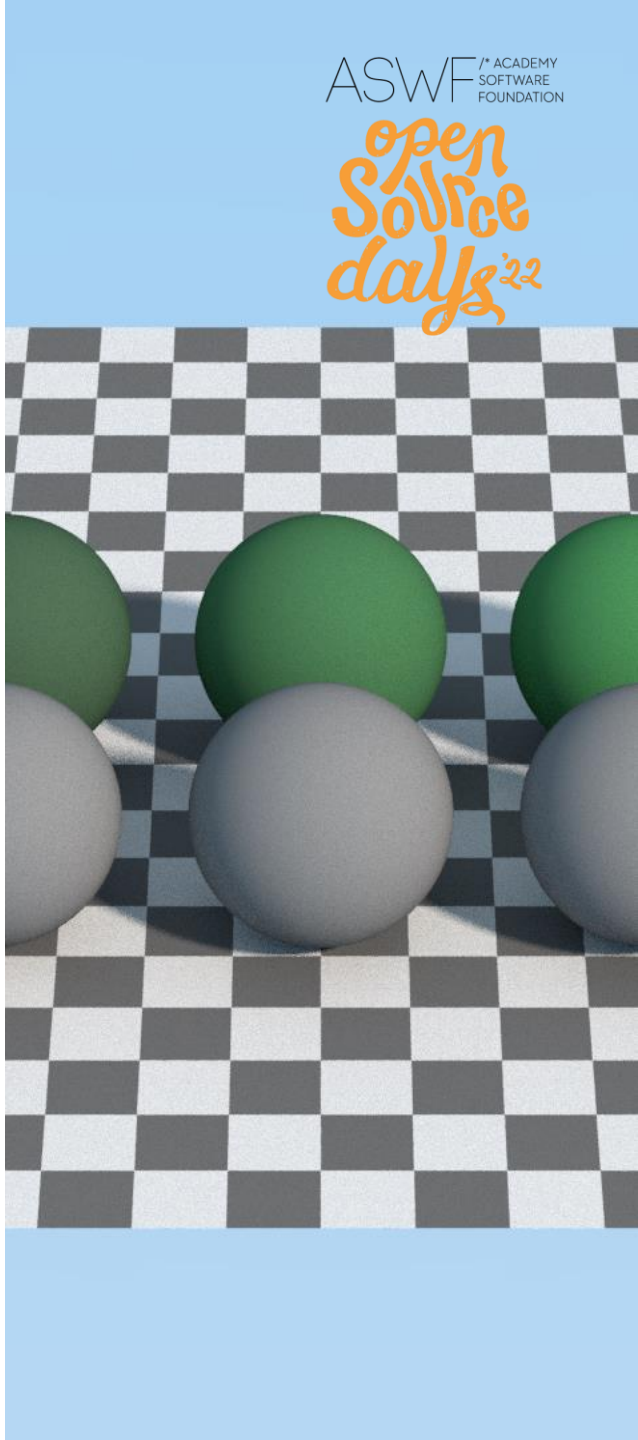
- Oren-Nayar
  - Diffuse reflection BSDF based on the Oren-Nayar reflectance model
  - **oren_nayar_diffuse_bsdf**(*normal* N, *color* albedo, *float* roughness)
  - Implemented in testrender OSL#1547

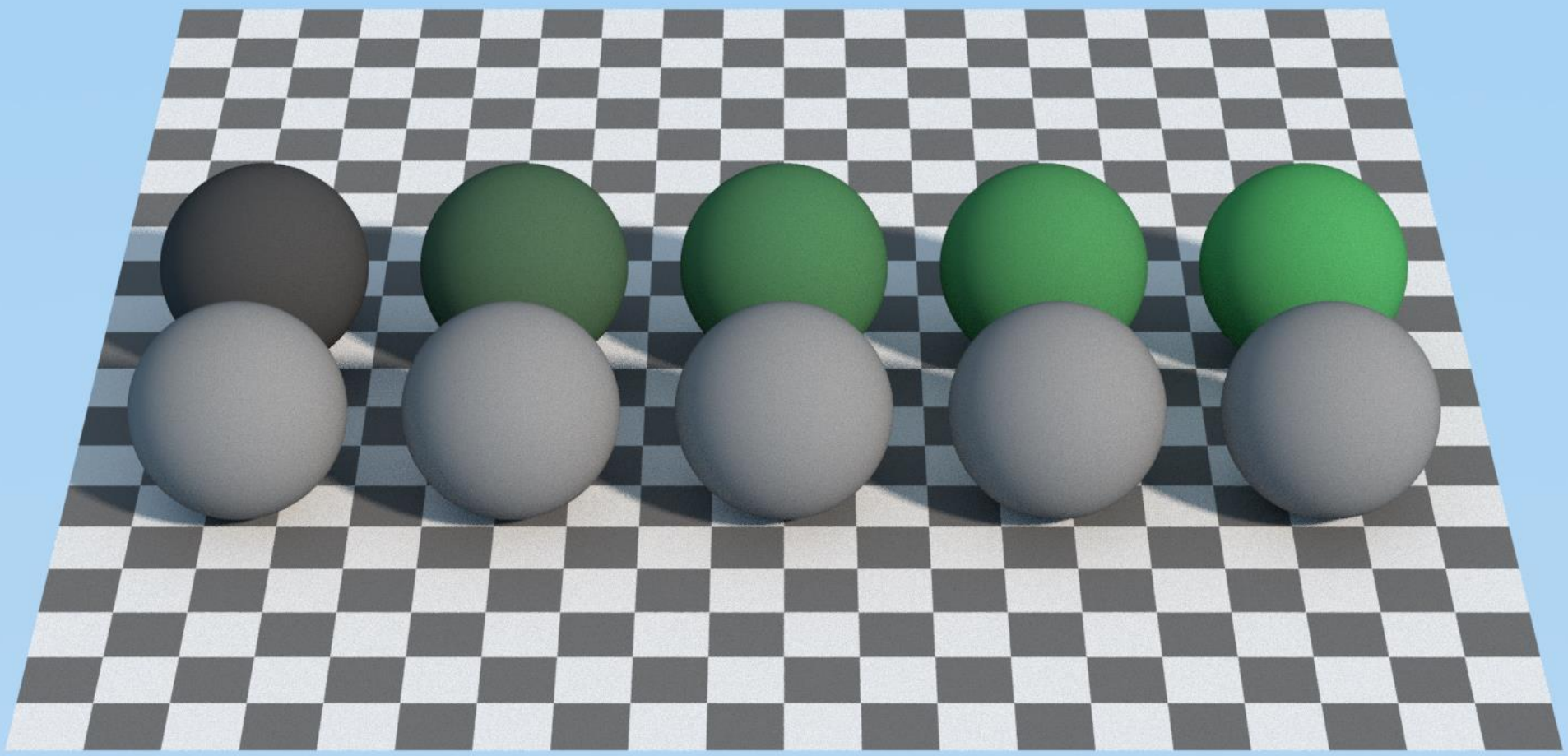




Diffuse BSDFs

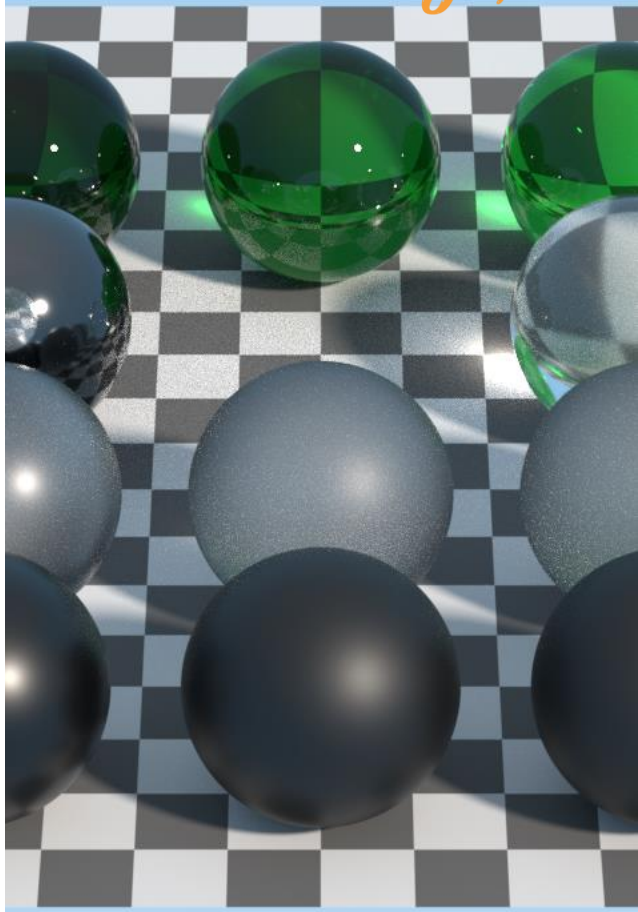
- Burley diffuse
  - Diffuse reflection BSDF based on the corresponding component of the Disney Principled shading model
  - **`burley_diffuse_bsdf`**(*`normal N, color albedo, float roughness`*)
  - Implemented in testrender OSL#1536

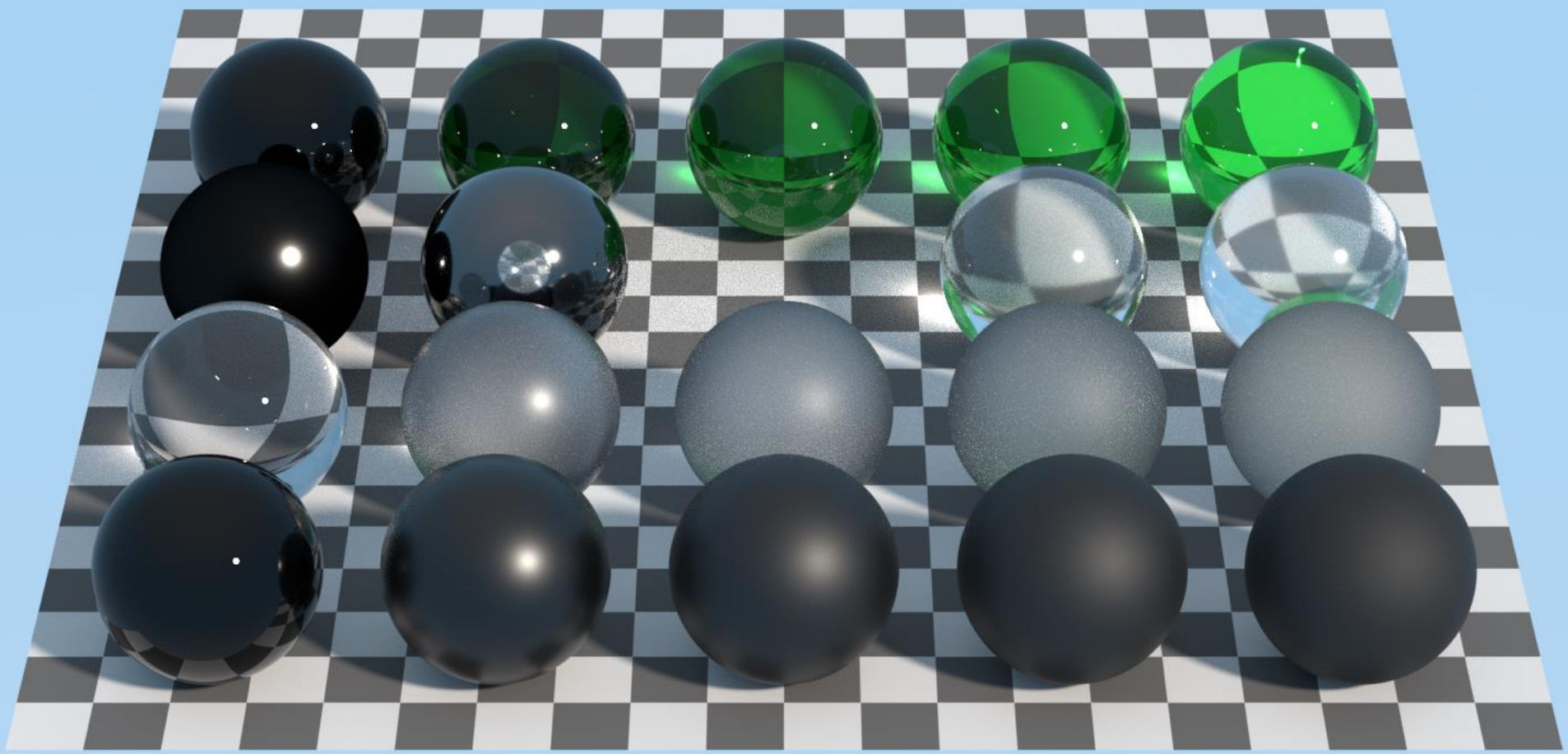




Microfacet BSDFs

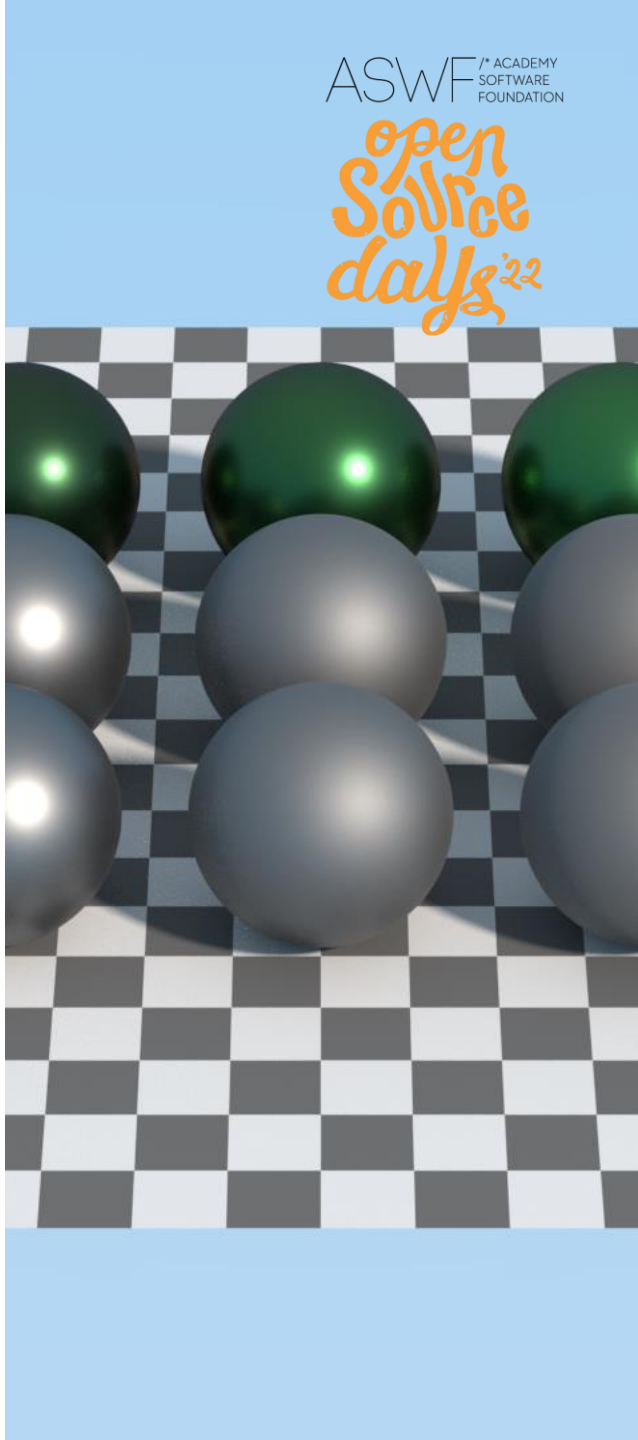
- Dielectric
  - Reflection and/or transmission BSDF based on a microfacet reflectance model and a Fresnel curve for dielectrics
  - **dielectric_bsdf**(*normal* N, *vector* U, *color* reflection_tint, *color* transmission_tint, *float* roughness_x, *float* roughness_y, *float* ior, *string* distribution)
  - Implemented in testrender OSL#1541

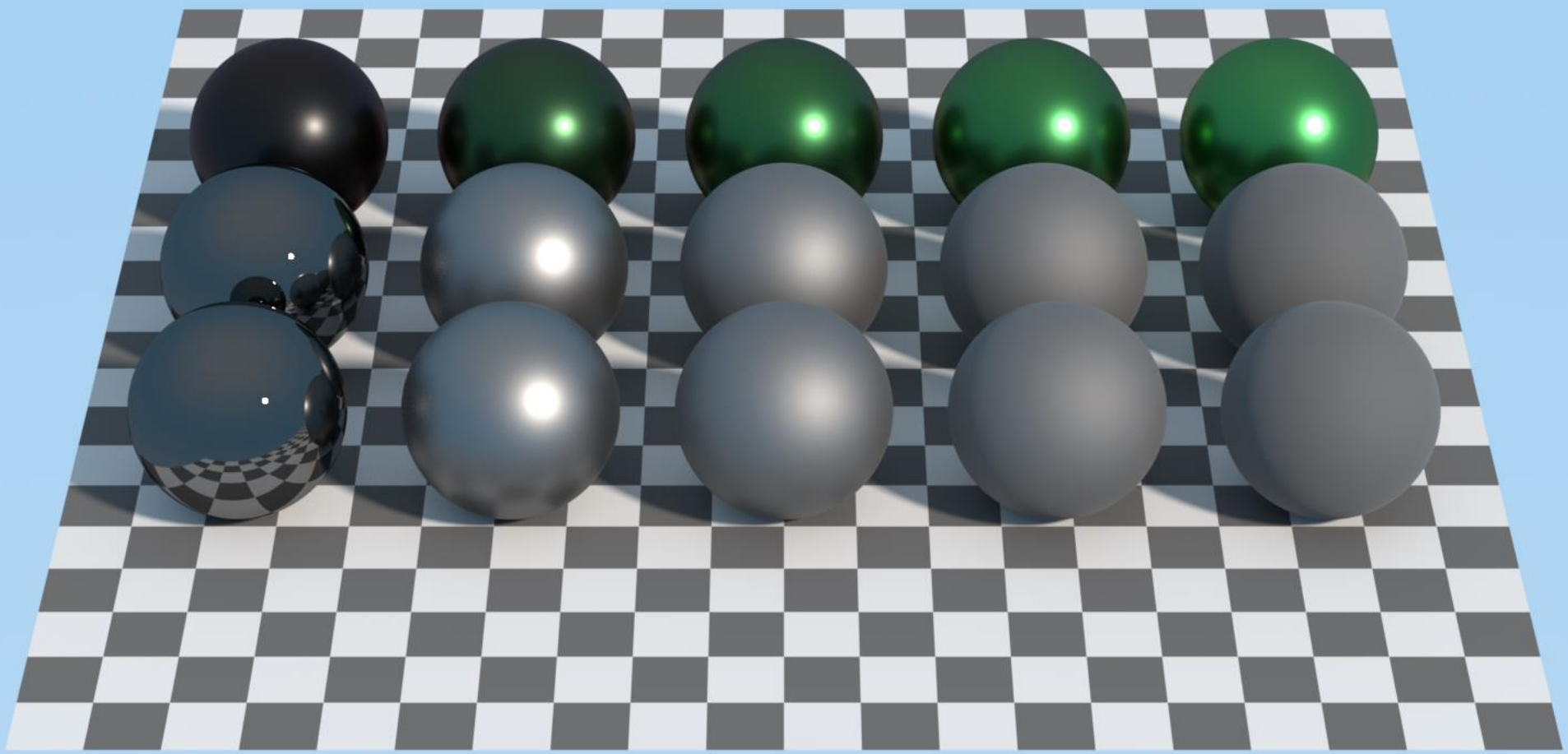




Microfacet BSDFs

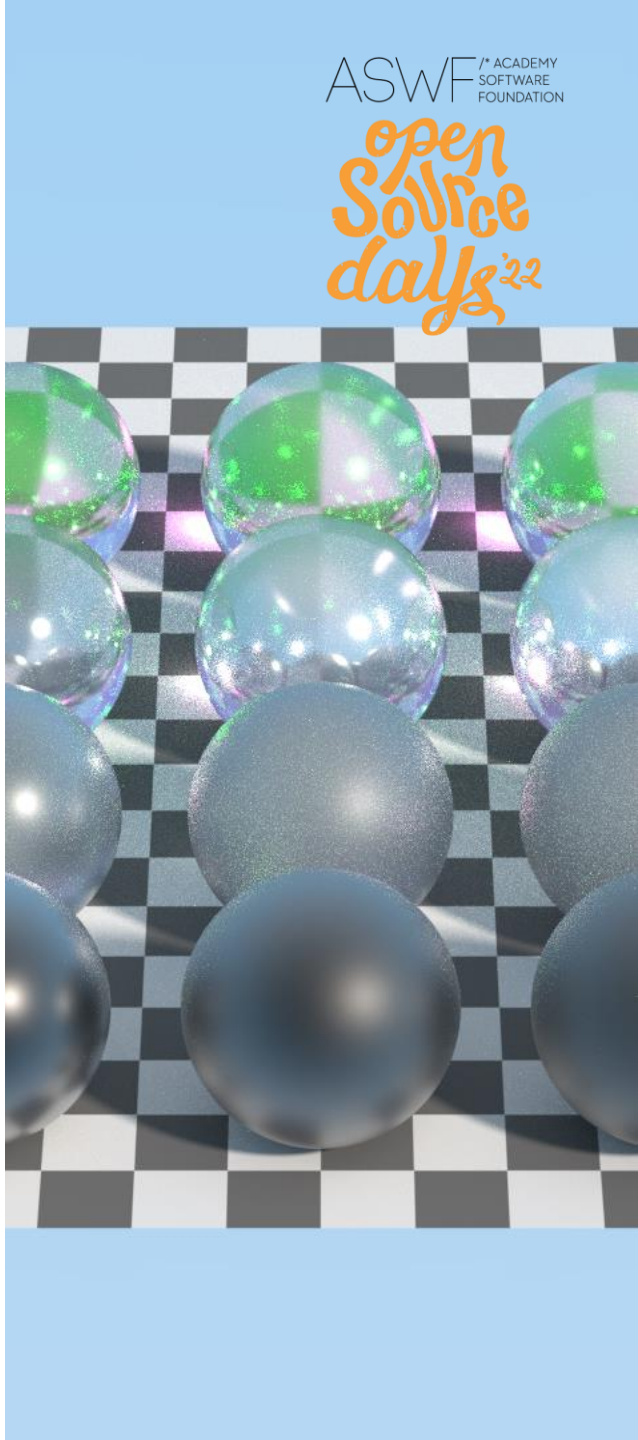
- Conductor
  - Reflection BSDF based on a microfacet reflectance model, it uses a Fresnel curve with complex refraction index for conductors/metals
  - **`conductor_bsdf`**(*`normal N, vector U, float roughness_x, float roughness_y, color ior, color extinction, string distribution`*)
  - Implemented in testrender OSL#1541

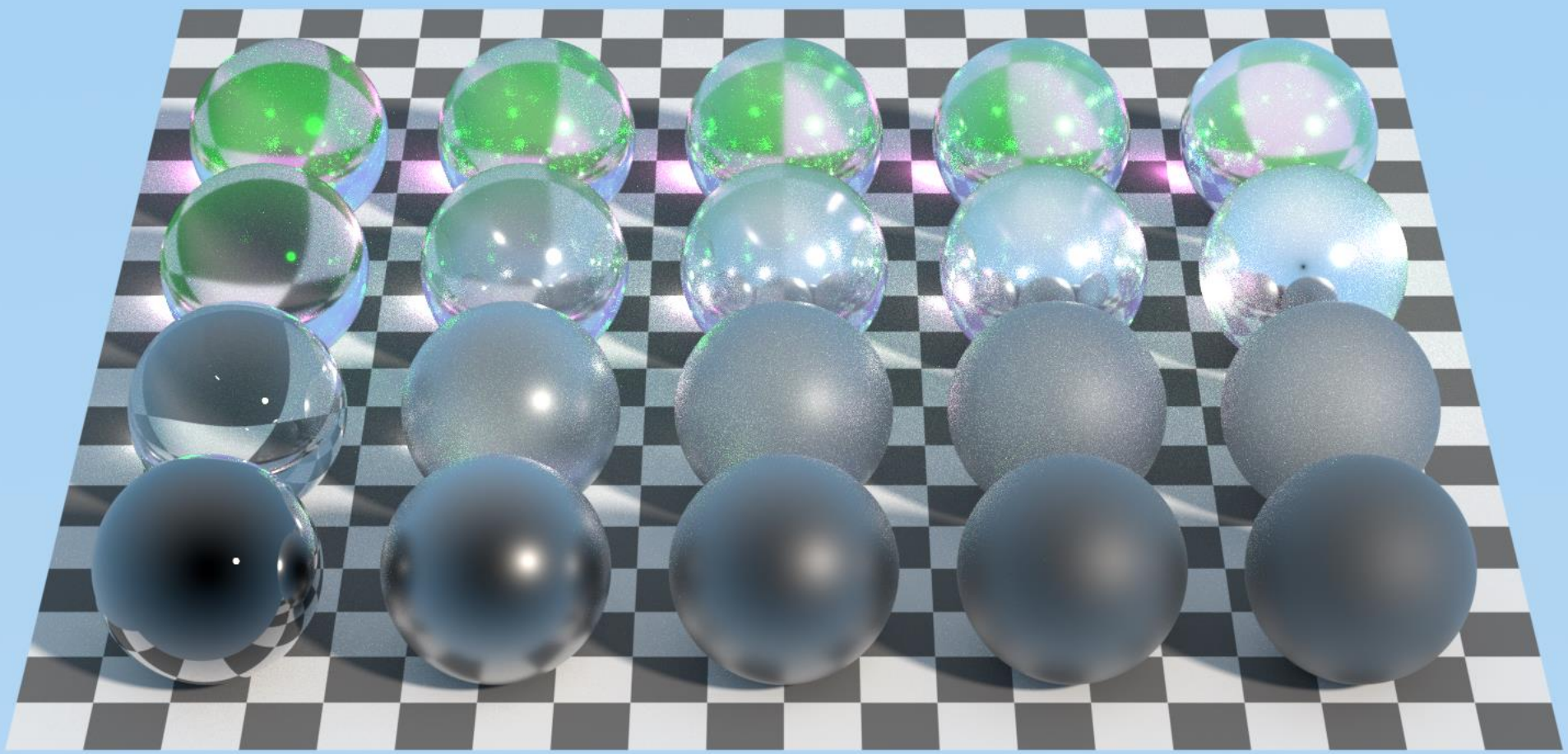




Microfacet BSDFs

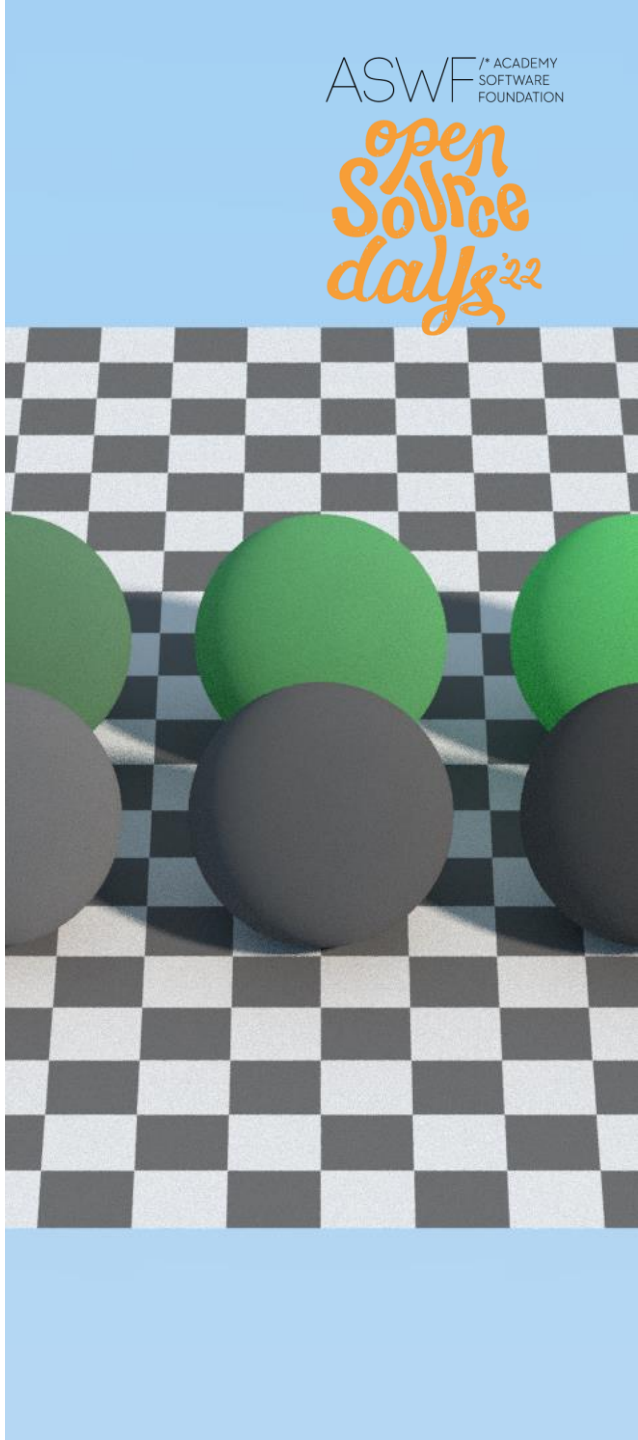
- Generalized Schlick
  - Reflection and/or transmission BSDF based on a microfacet reflectance model and a generalized Schlick Fresnel curve.
  - **generalized_schlick_bsdf**(*normal* N, *vector* U, *color* reflection_tint, *color* transmission_tint, *float* roughness_x, *float* roughness_y, *color* f0, *color* f90, *float* exponent, *string* distribution)
  - Implemented in testrender OSL#1541

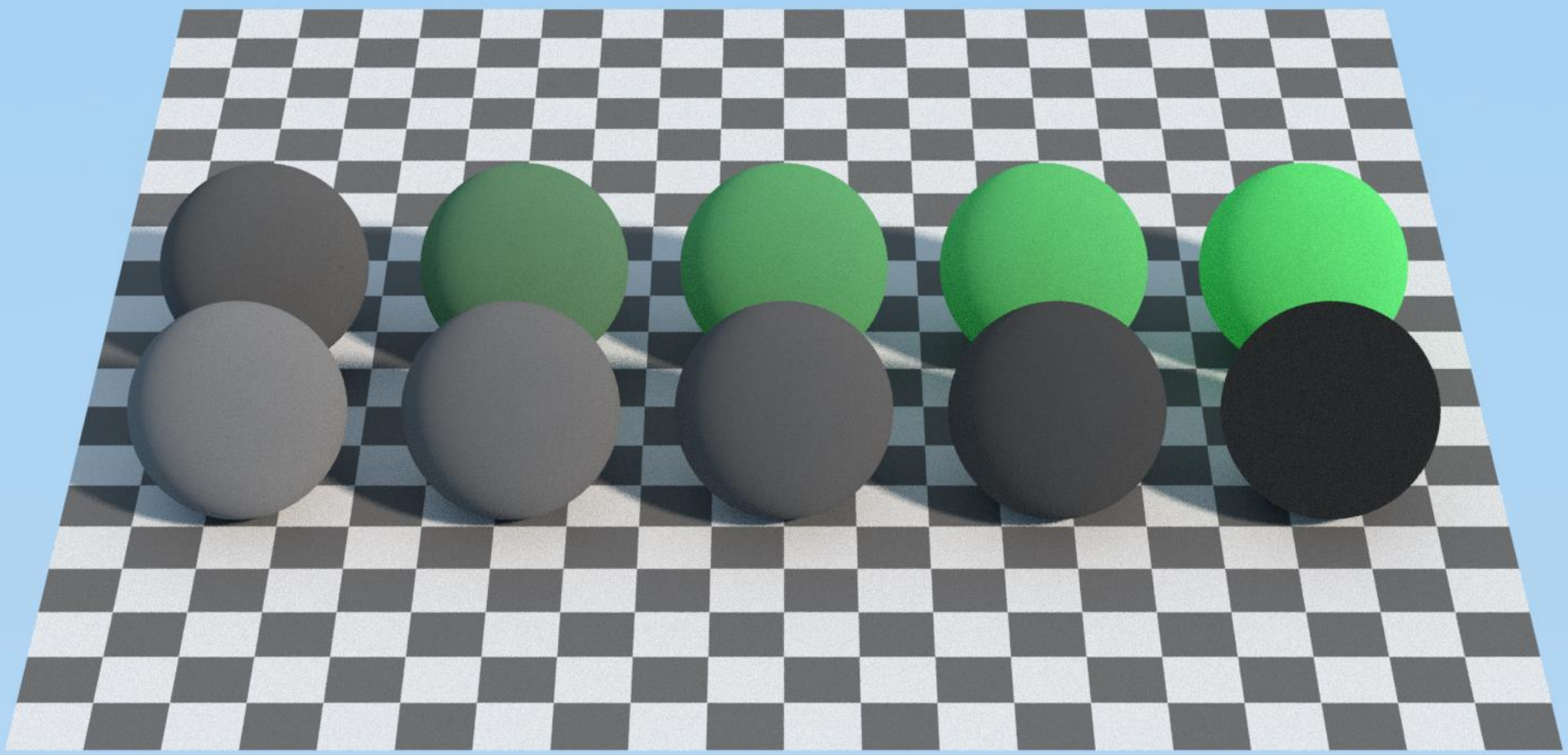




Diffuse transmission

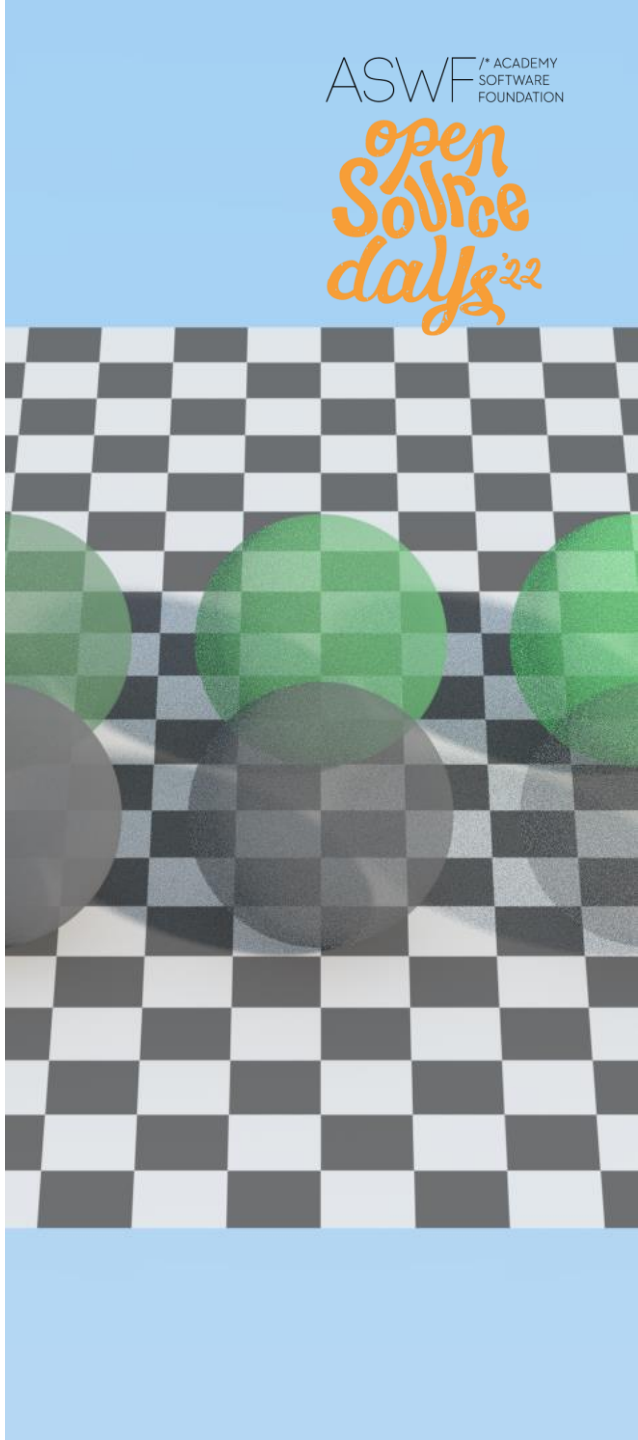
- Translucent
  - translucent (diffuse transmission) BSDF based on the Lambert reflectance model
  - **`translucent_bsdf`**(*`normal N, color albedo`*)
  - Implemented in testrender OSL#1547

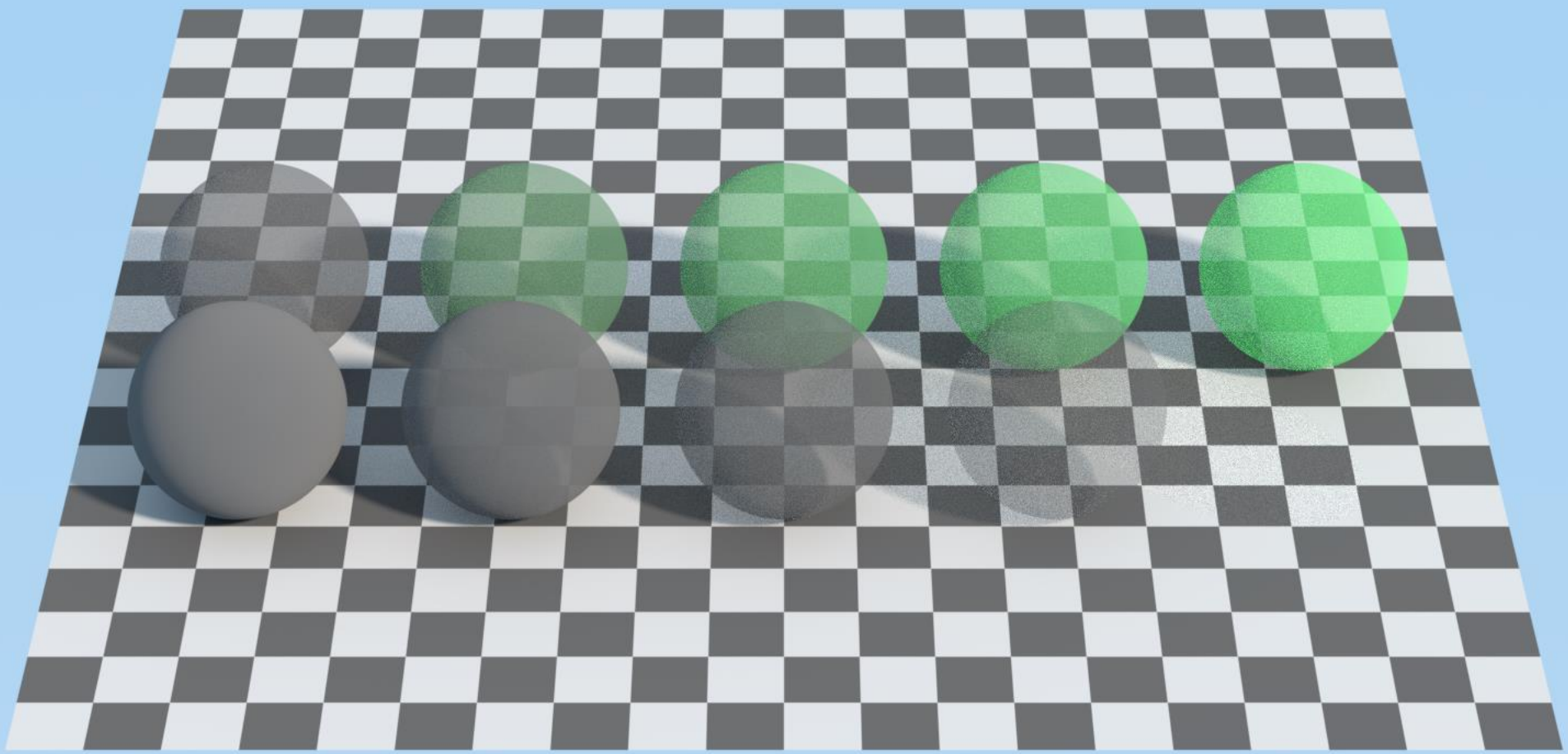




Transparency

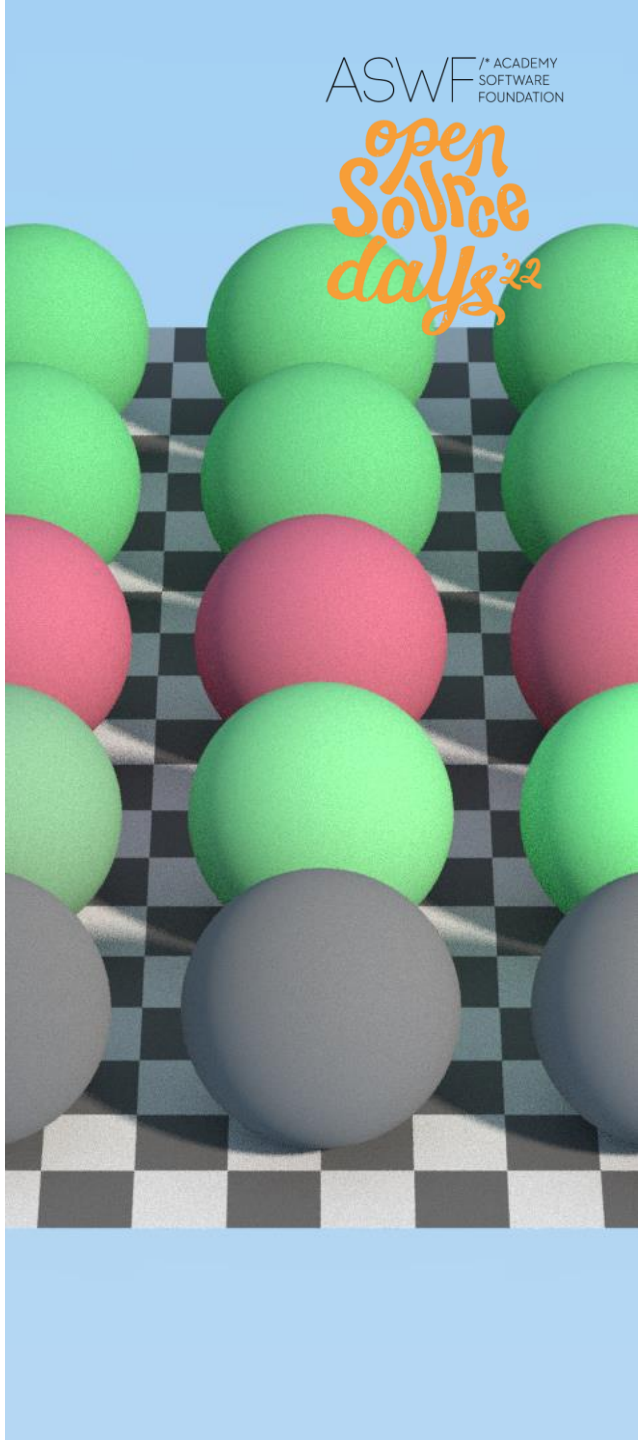
- Transparent
  - Straight transmission through a surface
  - **`transparent_bsdf`**()
  - Implemented in testrender OSL#1547





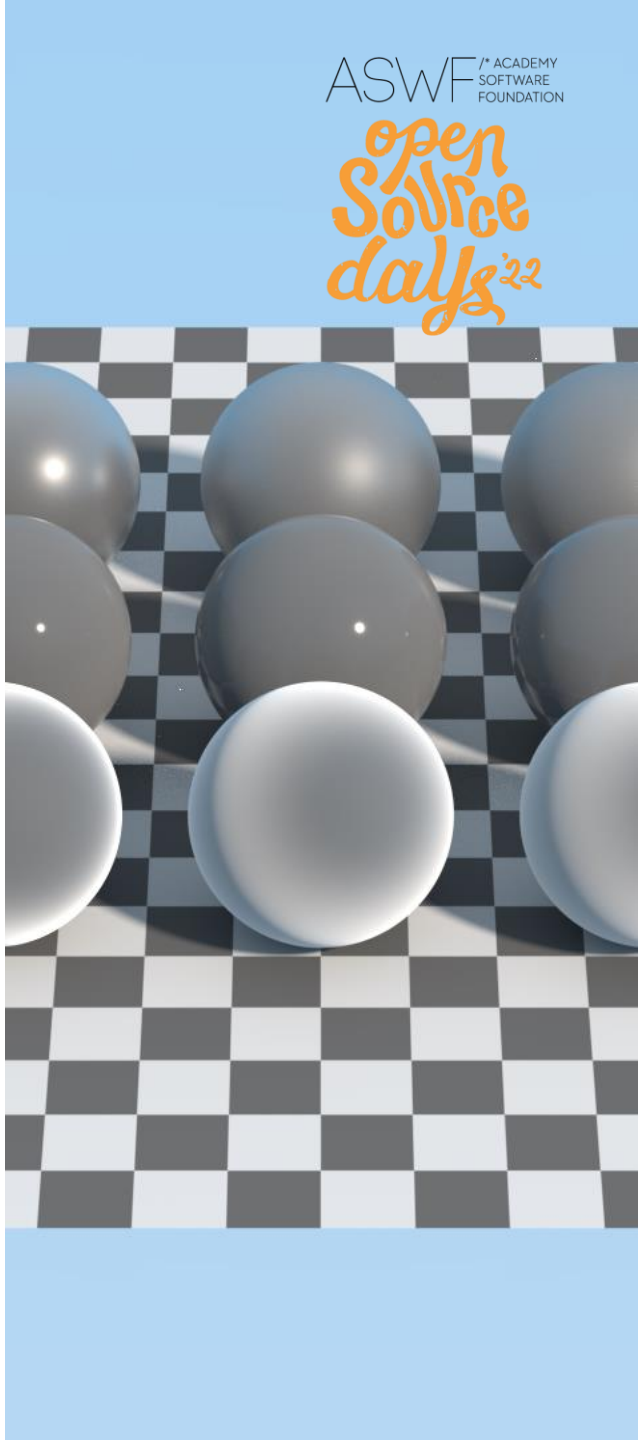
Subsurface scattering BSSRDF

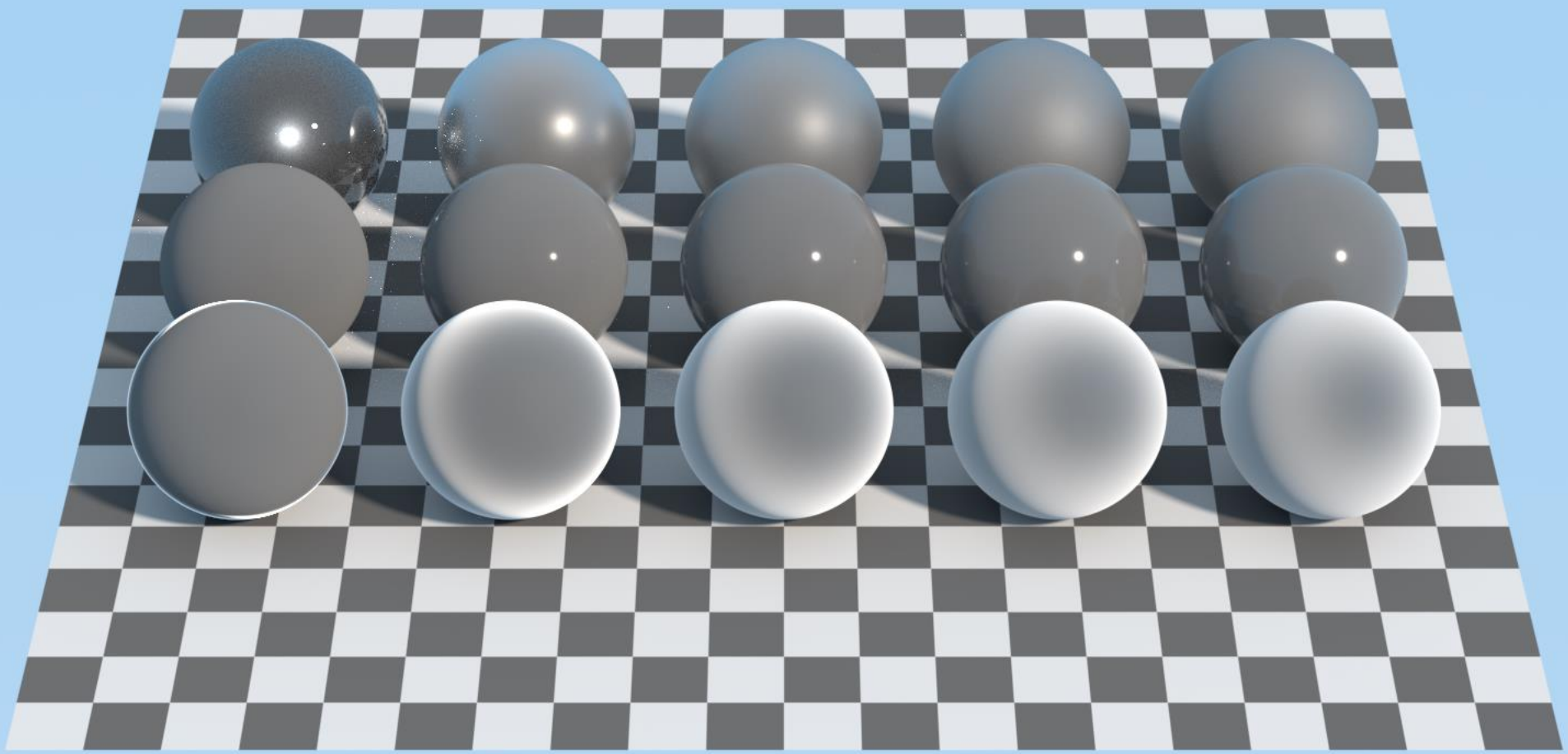
- Subsurface
  - BSSRDF for subsurface scattering within a homogeneous medium
  - **`subsurface_bssrdf`**(*`normal N, color albedo, float transmission_depth, color transmission_color, float anisotropy`*)
  - No support in testrender
    - Translated as a diffuse BSDF



Back scattering microfacet

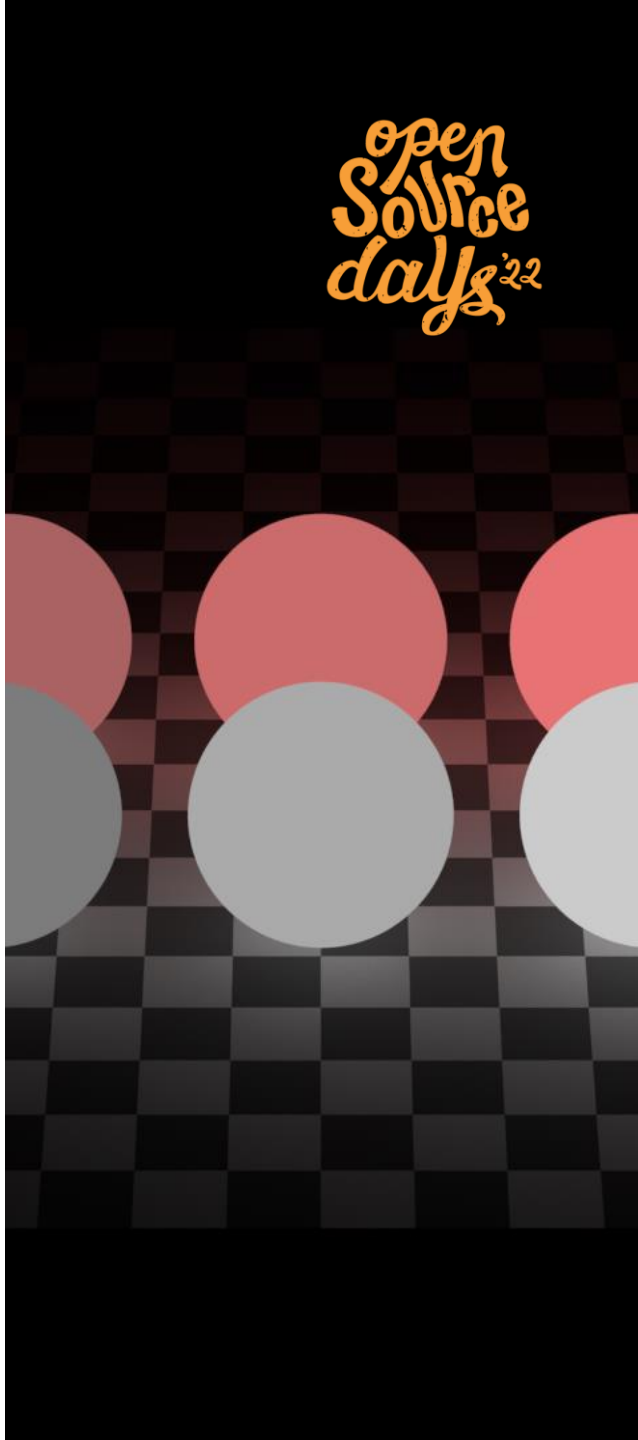
- Sheen
  - Microfacet BSDF for the back-scattering properties of cloth-like materials.
  - **`sheen_bsdf`**(*`normal N, color albedo, float roughness`*)
  - Implemented in testrender OSL#1537

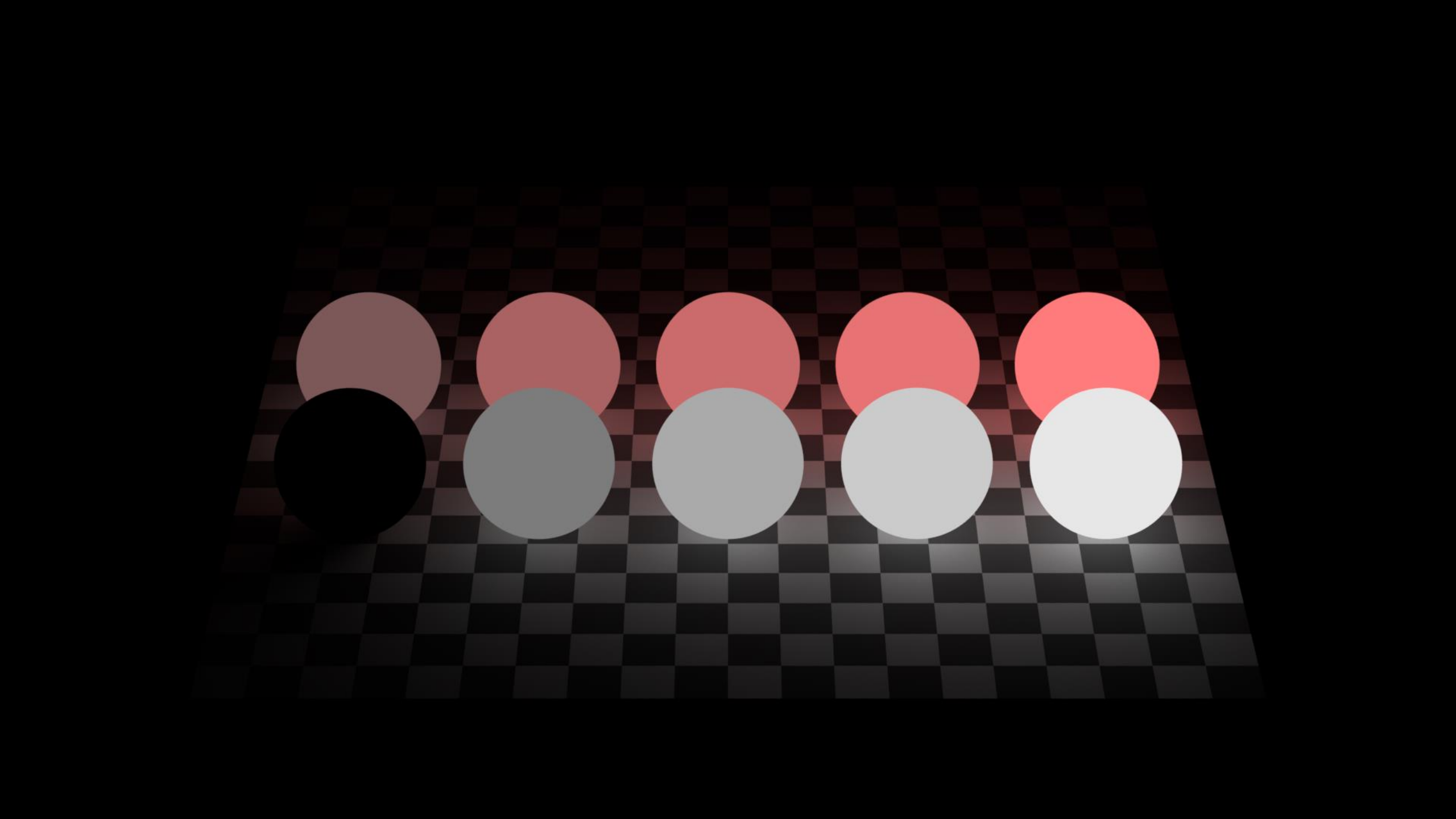




Emission EDF

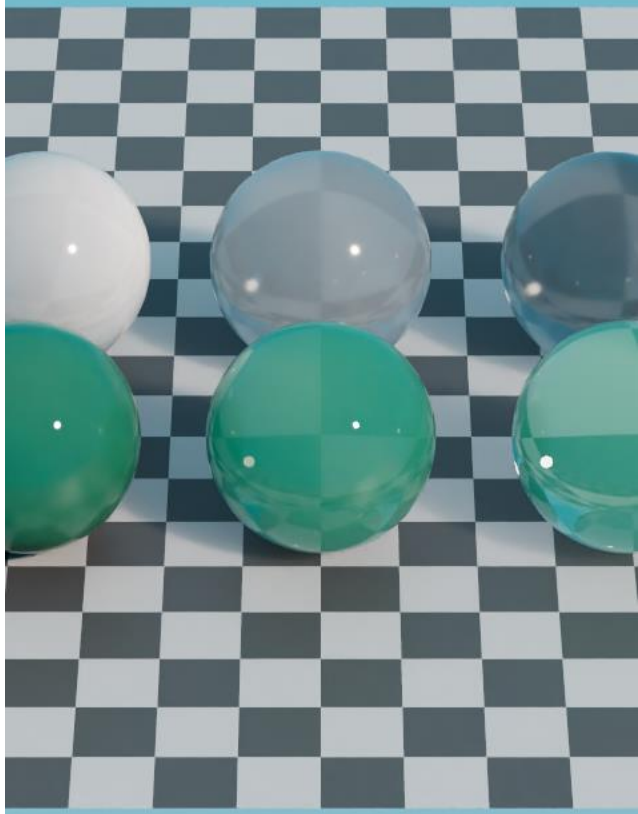
- Uniform
  - EDF emitting light uniformly in all directions
  - **`uniform_edf`**(*`color emittance`*)
  - Implemented in testrender [OSL#1547]





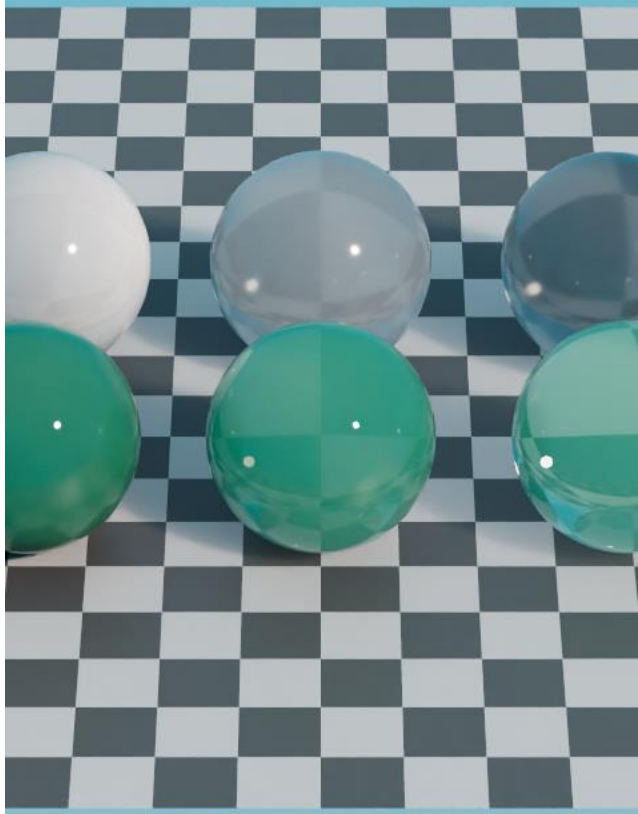
Volume closures

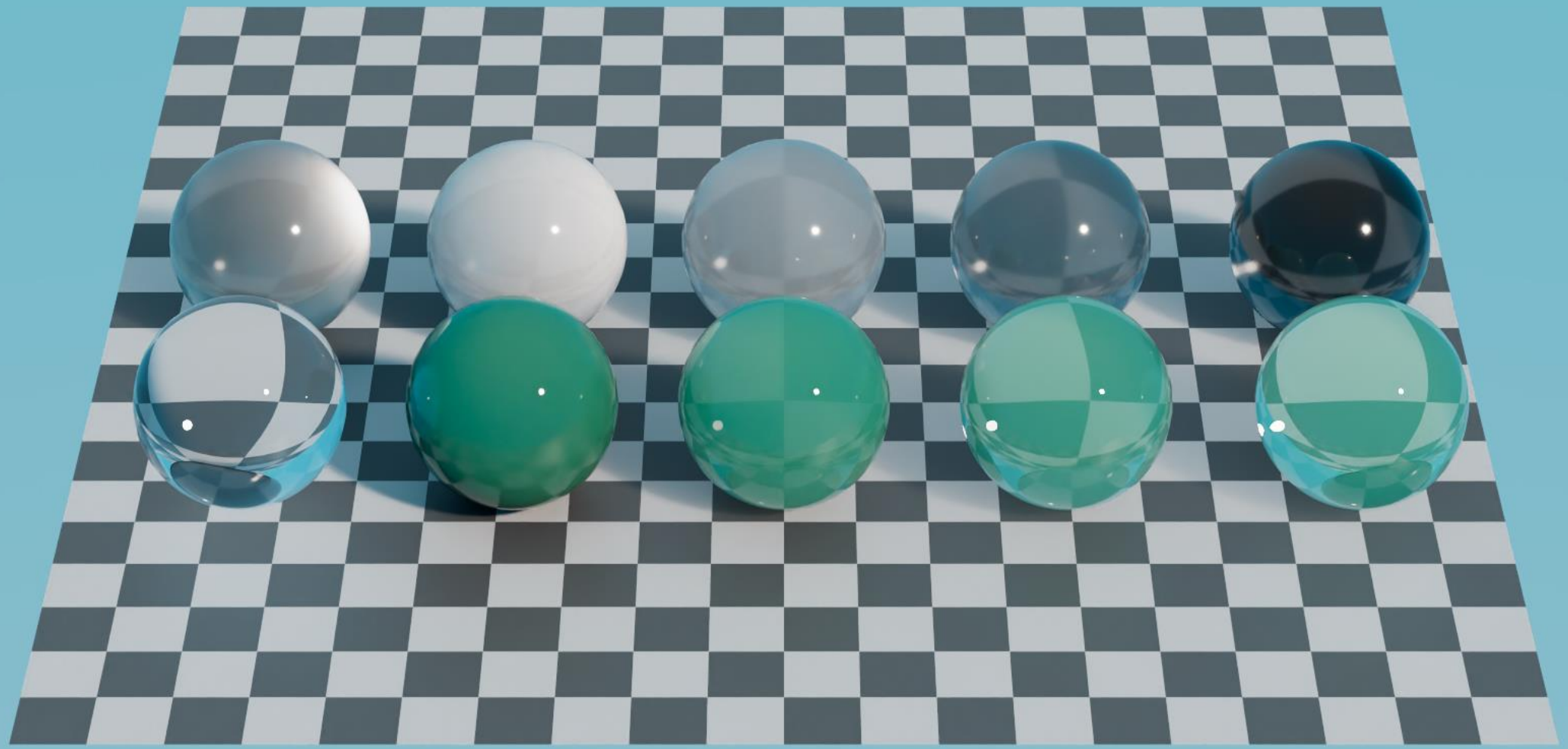
- Anisotropic
  - VDF scattering light for a general participating medium, based on the Henyey-Greenstein phase function
  - **`anisotropic_vdf`**`(`*`color`* `albedo`, *`color`* `extinction`, *`float`* `anisotropy`)
  - Implemented as extinction only [OSL#1547]



Volume closures

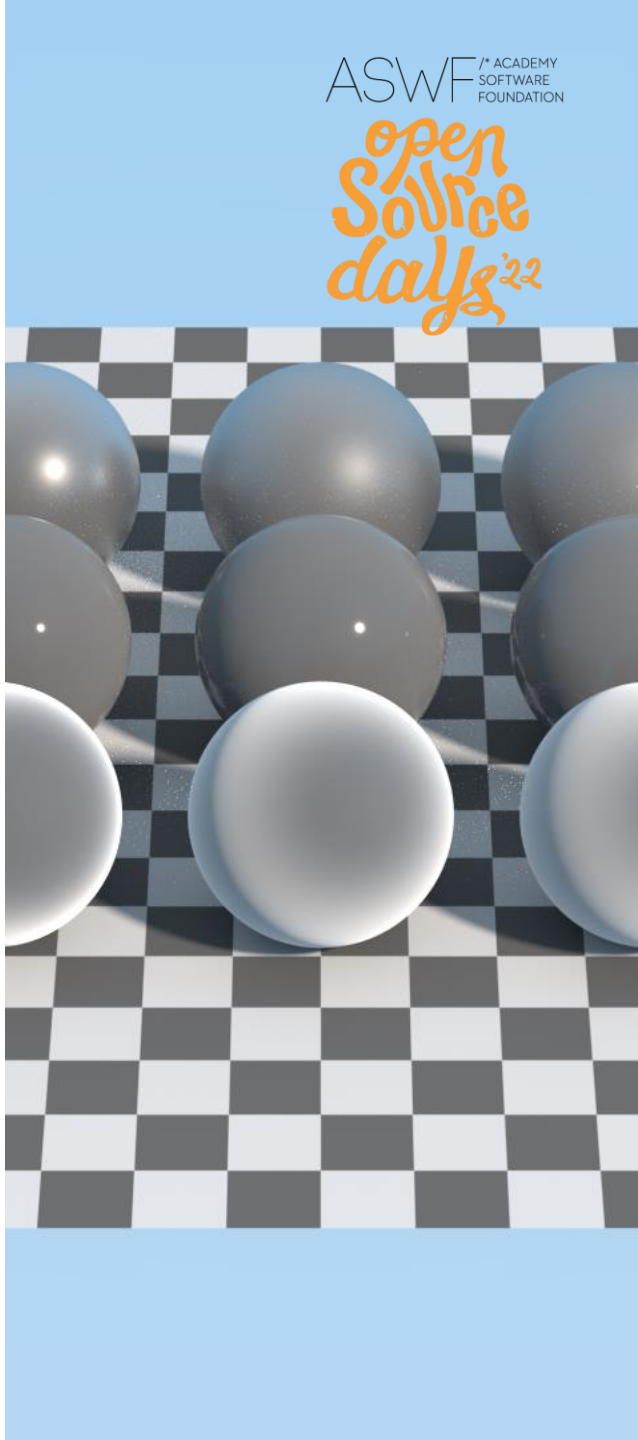
- Medium
  - VDF for light passing through a dielectric homogenous medium, such as glass or liquids
  - `medium_vdf(color albedo, float transmission_depth, color transmission_color, float anisotropy, float ior, int priority)`
  - Implemented as extinction only OSL#1547

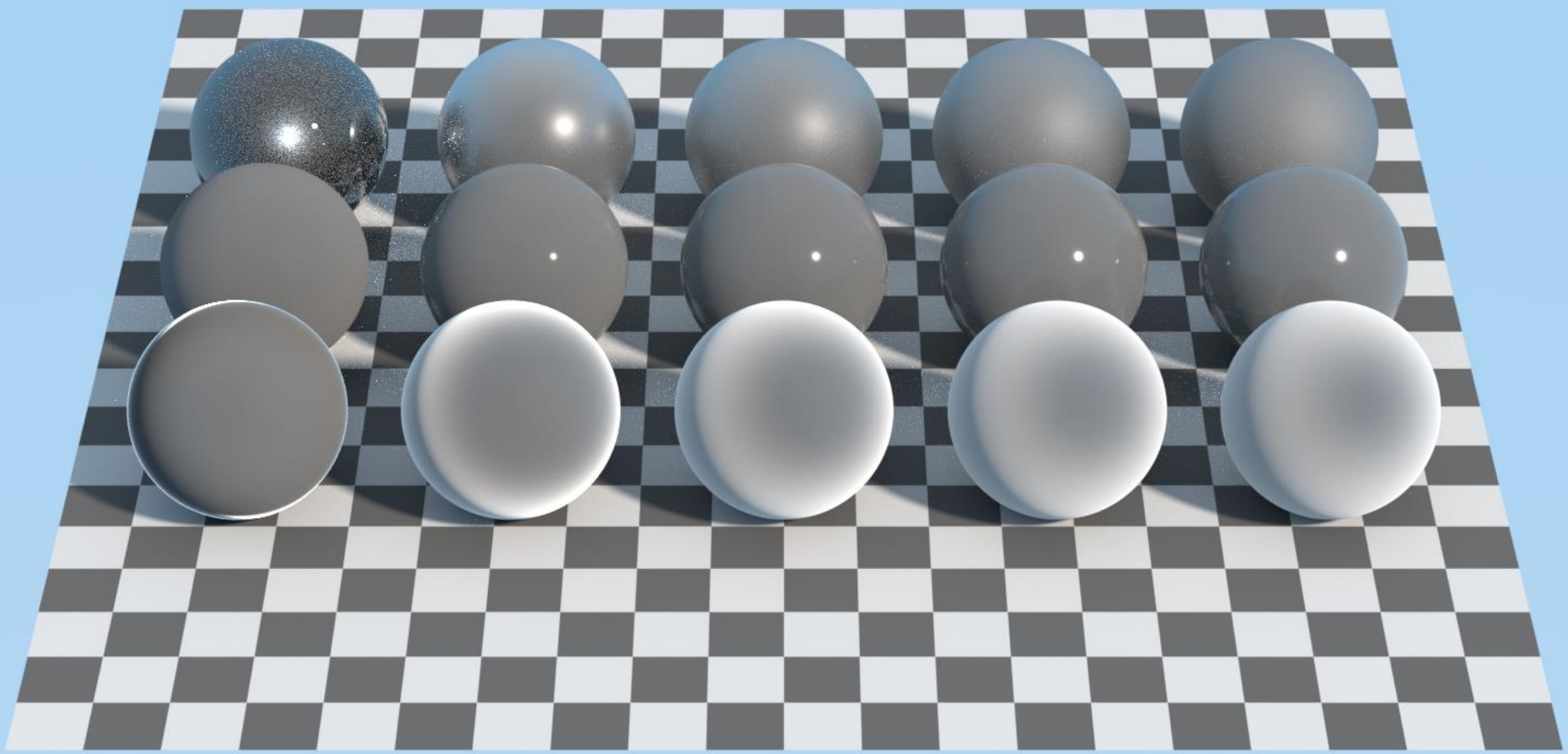




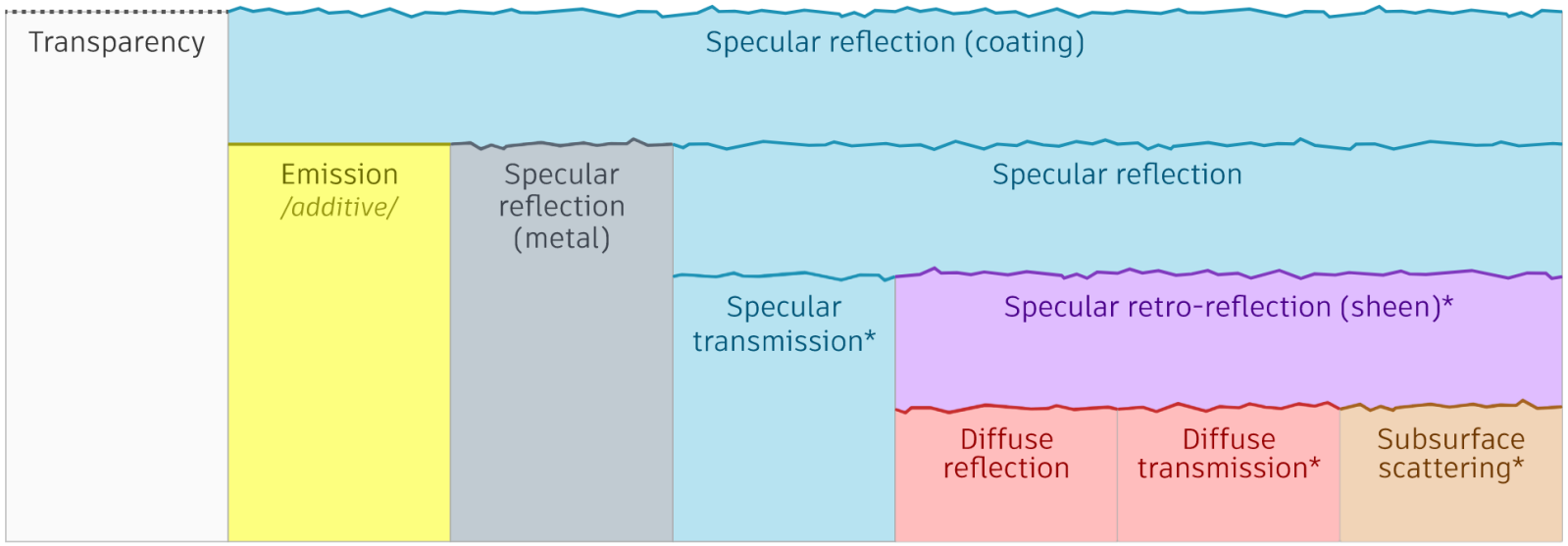
Layering closures

- Layer
  - Vertically layer a layerable BSDF such as dielectric_bsdf, generalized_schlick_bsdf or sheen_bsdf over a BSDF or VDF
  - **`layer`**`(`*`closure color`* `top`, *`closure color`* `base`)
  - Implemented in testrender OSL#1538
    - Missing support for layering VDFs

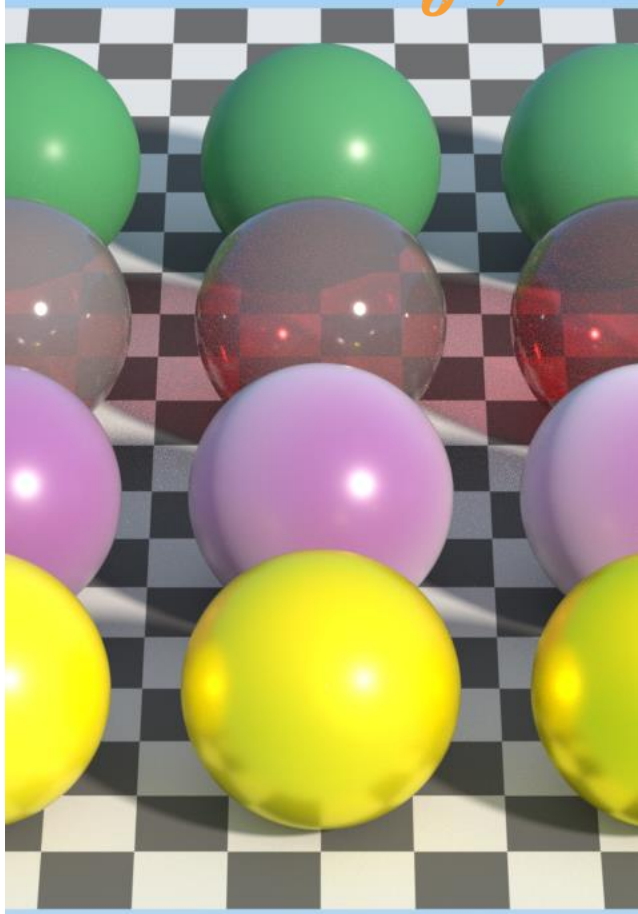




Implementing an ubershader using standard closures

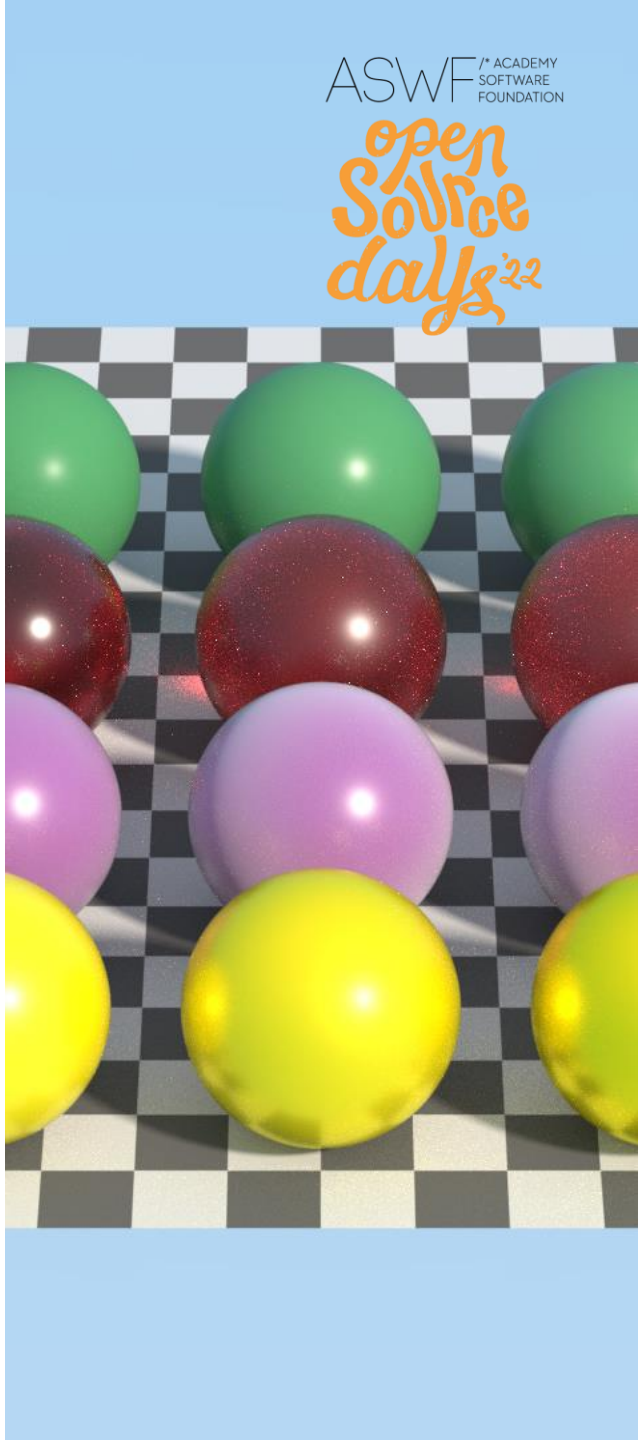


Source : Autodesk Standard Surface whitepaper - https://autodesk.github.io/standard-surface/



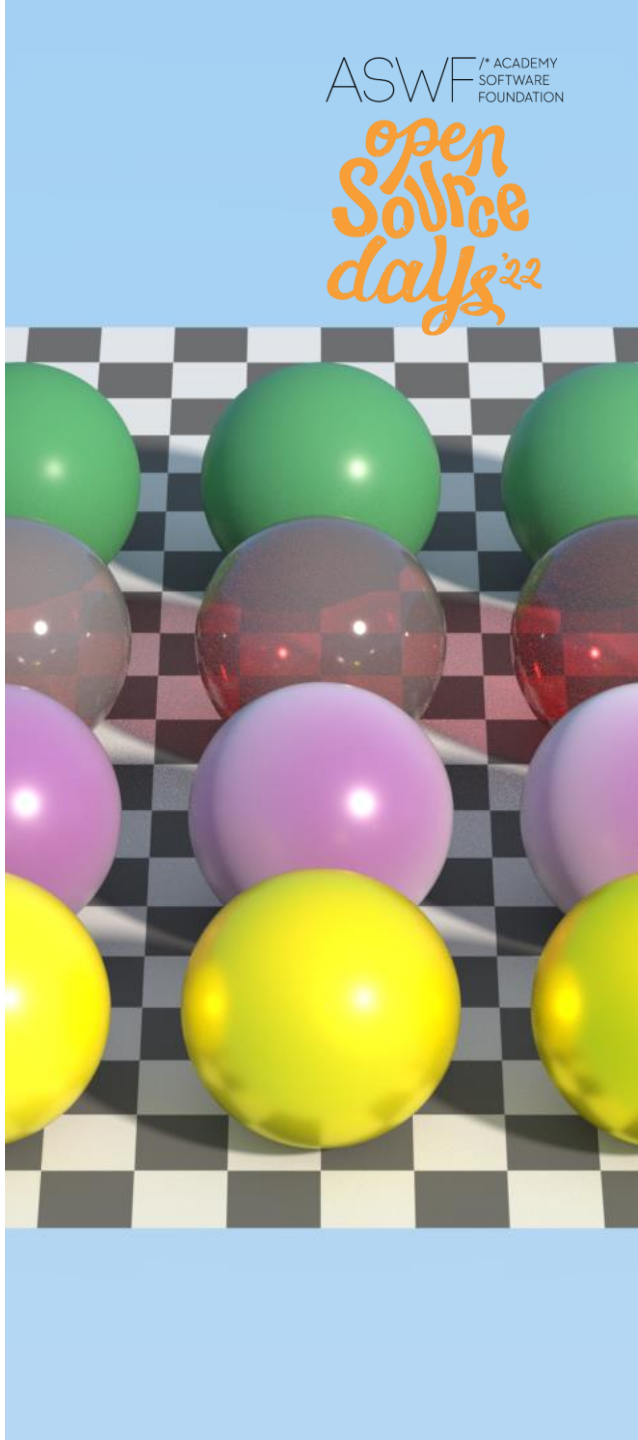
Implementing an ubershader using standard closures

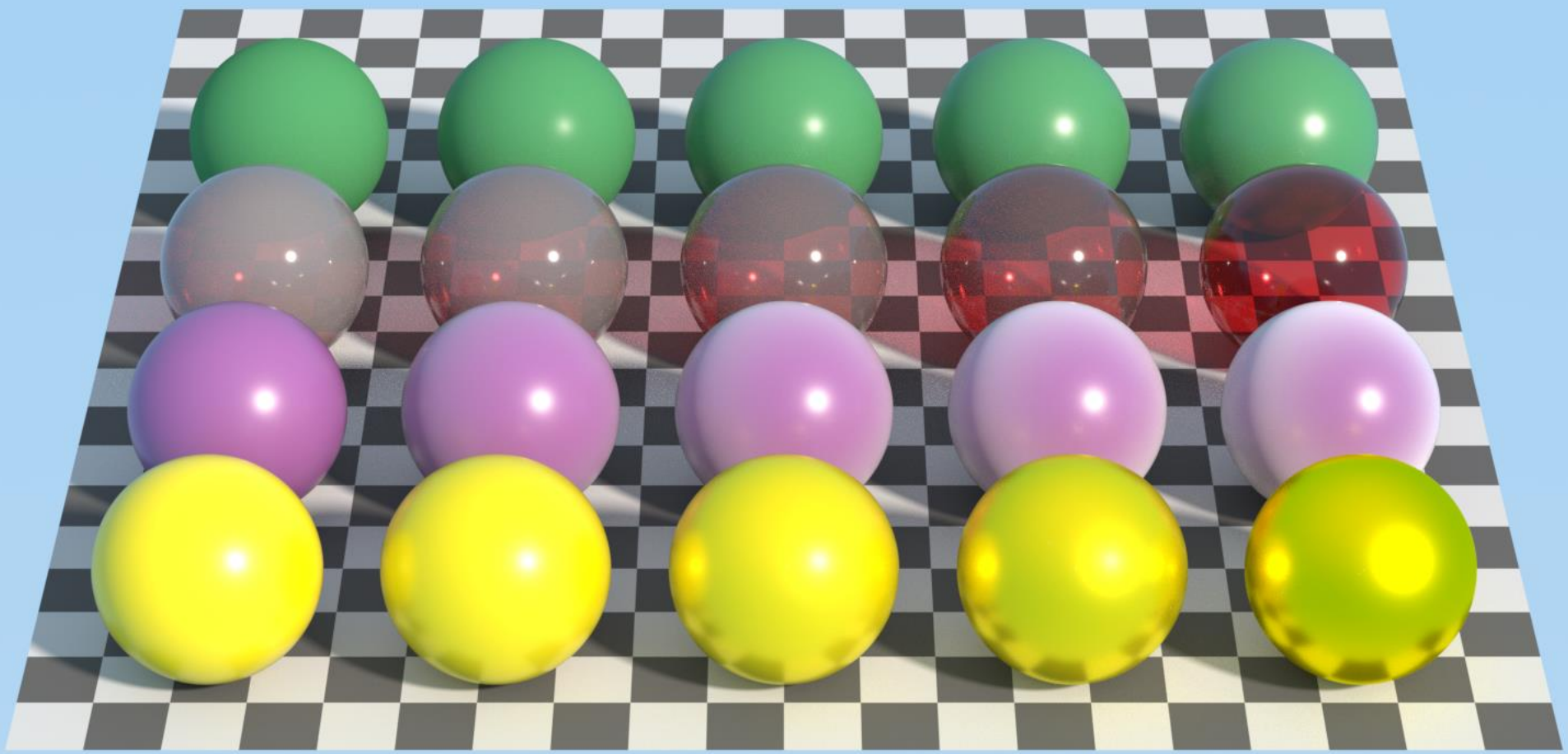
```
shader standard_surface(
     float base = .8,
     color base_color = color(1),
     float diffuse_roughness = 0,
     float specular = 1,
     color specular_color = color(1),
     float specular_roughness = .1,
     float specular_IOR = 1.52,
     float specular_anisotropy = 0,
     float specular_rotation = 0,
     float metalness = 0,
     float transmission = 0,
     color transmission_color = color(1),
     float transmission_depth = 0,
     float transmission_scatter = 0,
     float transmission_scatter_anisotropy = 0,
     float transmission_dispersion = 0,
     float subsurface = 0,
     color subsurface_color = color(1),
     color subsurface_radius = color(1),
     float subsurface_scale = 1,
     float subsurface_anisotropy = 0,
     float sheen = 0,
     color sheen_color = color(1),
     float sheen_roughness = 0.3,
     int thin_walled = 0 [[ string widget = "boolean"]],
     normal input_normal = N,
     vector tangent = dPdu,
     float coat = 0,
     color coat_color = color(1),
     float coat_roughness = .1,
     float coat_IOR = 1.5,
     normal coat_normal = N,
     float coat_affect_color = 0,
     float coat_affect_roughness = 0,
     float thin_film_thickness = 0,
     float thin_film_IOR = 1.5,
     float emission_w = 0,
     color emission_color = color(1),
     color opacity = color(1),
     output closure color standard_surface_closures = 0)
{
    ...
}
```



Implementing an ubershader using standard closures

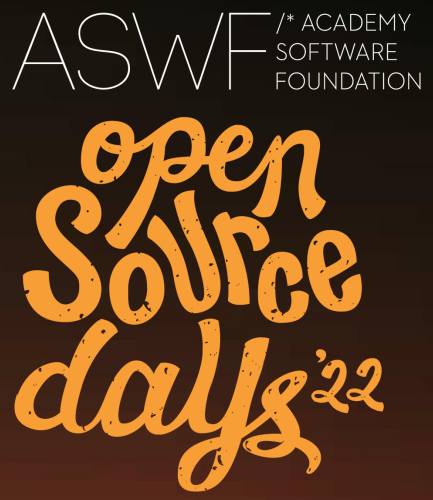
```
Ci = (color(1) - opacity) * transparent_bsdf() + // transparency
     opacity *
     (emission_w * uniform_edf(emission_color) + // emission
     // coat
     layer(coat * dielectric_bsdf (N, 0, coat_color, 0, coat_roughness, coat_roughness, coat_IOR, "ggx"),
         // specular_reflection
         mix(layer(dielectric_bsdf (N, 0, specular_color, 0, specular_roughness, specular_roughness, specular_IOR, "ggx"),
                     // sheen
                     mix(layer(sheen * sheen_bsdf(N, sheen_color, sheen_roughness),
                             mix(base * oren_nayar_diffuse_bsdf(N, base_color, diffuse_roughness),
                                 subsurface_bssrdf(N, subsurface_color, subsurface_radius[0], subsurface_color, subsurface_anisotropy),
                                 subsurface_weight)),
                         // specular transmission
                         dielectric_bsdf (N, 0, 0, transmission_color, specular_roughness, specular_roughness, specular_IOR, "ggx"),
                         transmission)),
             // metal
             conductor_bsdf(N, 0, specular_roughness, specular_roughness, conductor_ior, conductor_extinction, "ggx"),
             metalness))
     );
```





Next steps

- Implement all hard surface closures and parameters in testrender
- Expand the capabilities of testrender to implement SSS closures, volume closures ?
  - OSL testrender needs to be kept simple so a full volume integrator is not currently planned
  - We could go for a simple approximation
- Call to renderer writers to implement the new set of closures
- Implement support in the MaterialX OSL generator
  - WIP PR MaterialX#1039



Open Shading Language – 2022 update

Larry Gritz
Sony Pictures Imageworks

What is OSL?

- De facto standard shading language for production path tracing
- Language spec + libraries
- Found in: Arnold, RenderMan, 3dsMax, 3Delight, Clarisse, V-Ray, OTOY Octane and Brigade, Redshift, Blender/Cycles, …
- Academy Sci-Tech Award in 2017
- Originated at Sony Pictures Imageworks, now ASWF project
- Open source: http://openshadinglanguage.org

OSL 1.12 release

- A lot of studios work from 'main', but we haven't had a new official stable release branch for a long time.
- 1.12 is in beta NOW, intend to call it "release" by Sept 1 (-ish).
- Major new features:
  - Batch shading via SIMD
  - GPU / OptiX
  - Synchronize standard material closures with MaterialX

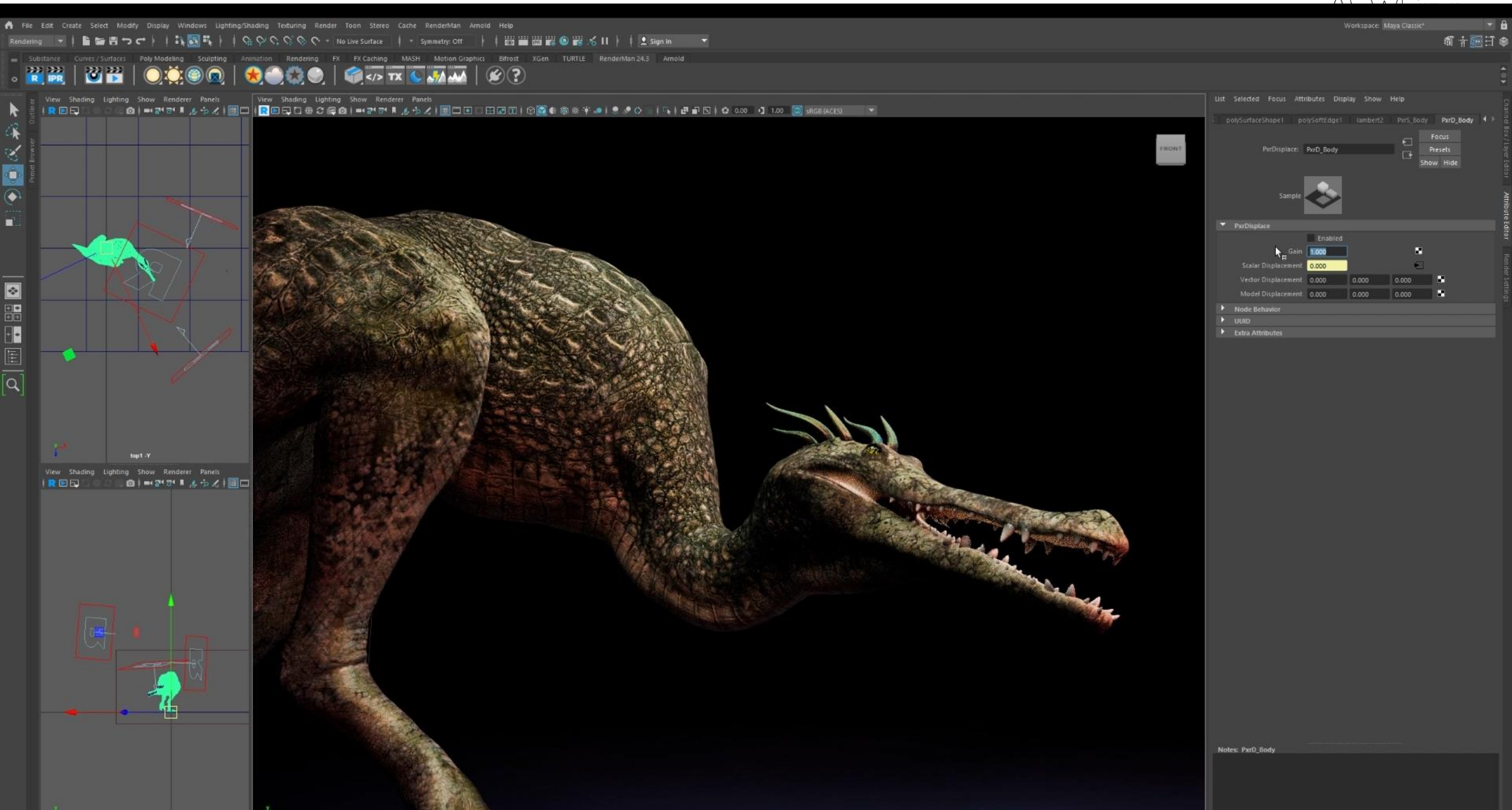
Batch shading

- Reminder: usually, we execute shader at a single point (ray)

- Work by Intel + Pixar (in use now by RenderMan)
- Shade in batches of 8 or 16 points at the same time
- Accelerated by Intel AVX-2 or AVX-512 SIMD instructions
- All shades in the batch must be the same material / shader group
- OSL language features are fully supported

GPU / OptiX shading

- GPU ray tracing on NVIDIA
- Work by NVIDIA + Sony Imageworks (in use now by Arnold, RenderMan XPU, Isotropix Angie)
- For use primarily with OptiX toolkit (though pure Cuda also works)
- Most of the language works, enough for procedural patterns, etc. Still more to go to reach 100% feature support.
- This will grow to full feature parity in the roadmap for the next release
- Demo video


File Edit Create Select Modify Display Windows Lighting/Shading Texturing Render Toon Stereo Cache RenderMan Arnold Help
Workspace: Maya Classic*
Rendering
No Live Surface
Symmetry: Off
Sign In
Substance Curves / Surfaces Poly Modeling Sculpting Animation Rendering FX FX Caching MASH Motion Graphics Bifrost XGen TURTLE RenderMan 24.3 Arnold
View Shading Lighting Show Renderer Panels
FRONT
top1 -Y
List Selected Focus Attributes Display Show Help
polySurfaceShape1 polySoftEdge1 lambert2 PxrS_Body PxrD_Body
PxrDisplace: PxrD_Body
Focus
Presets
Show Hide
Sample
PxrDisplace
Enabled
Gain 1.000
Scalar Displacement 0.000
Vector Displacement 0.000 0.000 0.000
Model Displacement 0.000 0.000 0.000
Node Behavior
UUID
Extra Attributes
Notes: PxrD_Body

Roadmap for the next year

- OptiX back end → full feature parity
- RendererServices → "free function" callbacks as LLVM bitcode
- First class support for vector2, vector4, and {color+alpha} type
- Re-evaluation of parameters and their upstream nodes
- A core C-like back end that can be customized to translate to other languages such as GLSL

Help wanted!

- This roadmap is ambitious, need to improve bus/retirement factor
- Please get involved
- If your studio or product is heavily relying on OSL, please consider dedicating some engineering time to helping to improve it
- It doesn't have to be a huge engineering task
- Usual comms:
  - Open biweekly TSC meetings: https://www.aswf.io/meeting-calendar/
  - Dev mail list: https://lists.aswf.io/g/osl-dev
  - Slack: https://slack.aswf.io/ (#openshadinglanguage)


ASWF
/* ACADEMY SOFTWARE FOUNDATION
open source days '22
Thank You!
Questions?