/* ACADEMY SOFTWARE FOUNDATION

open Source days '26

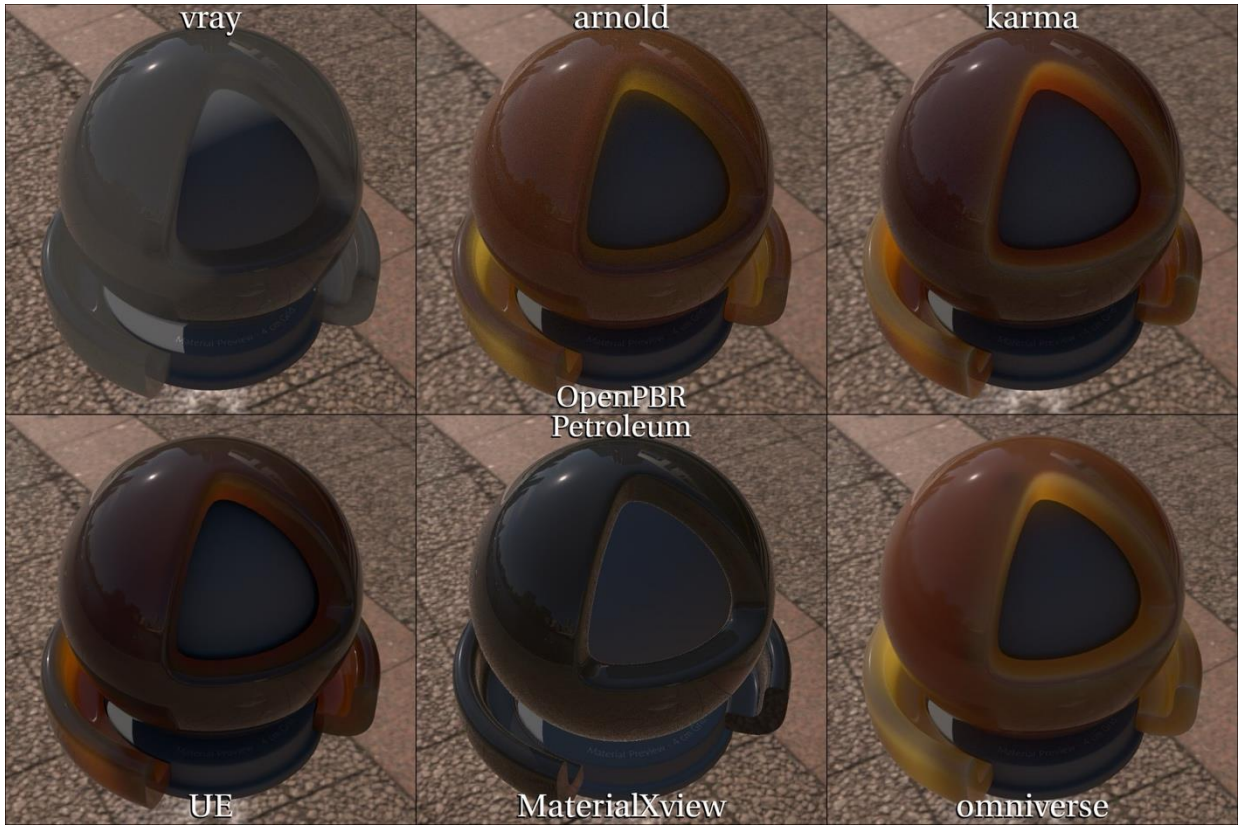
Typhoon: A Reference Renderer for OpenUSD Interoperability

Masuo Suzuki
NVIDIA

Anders Langlands
NVIDIA

Why Final Images Still Drift

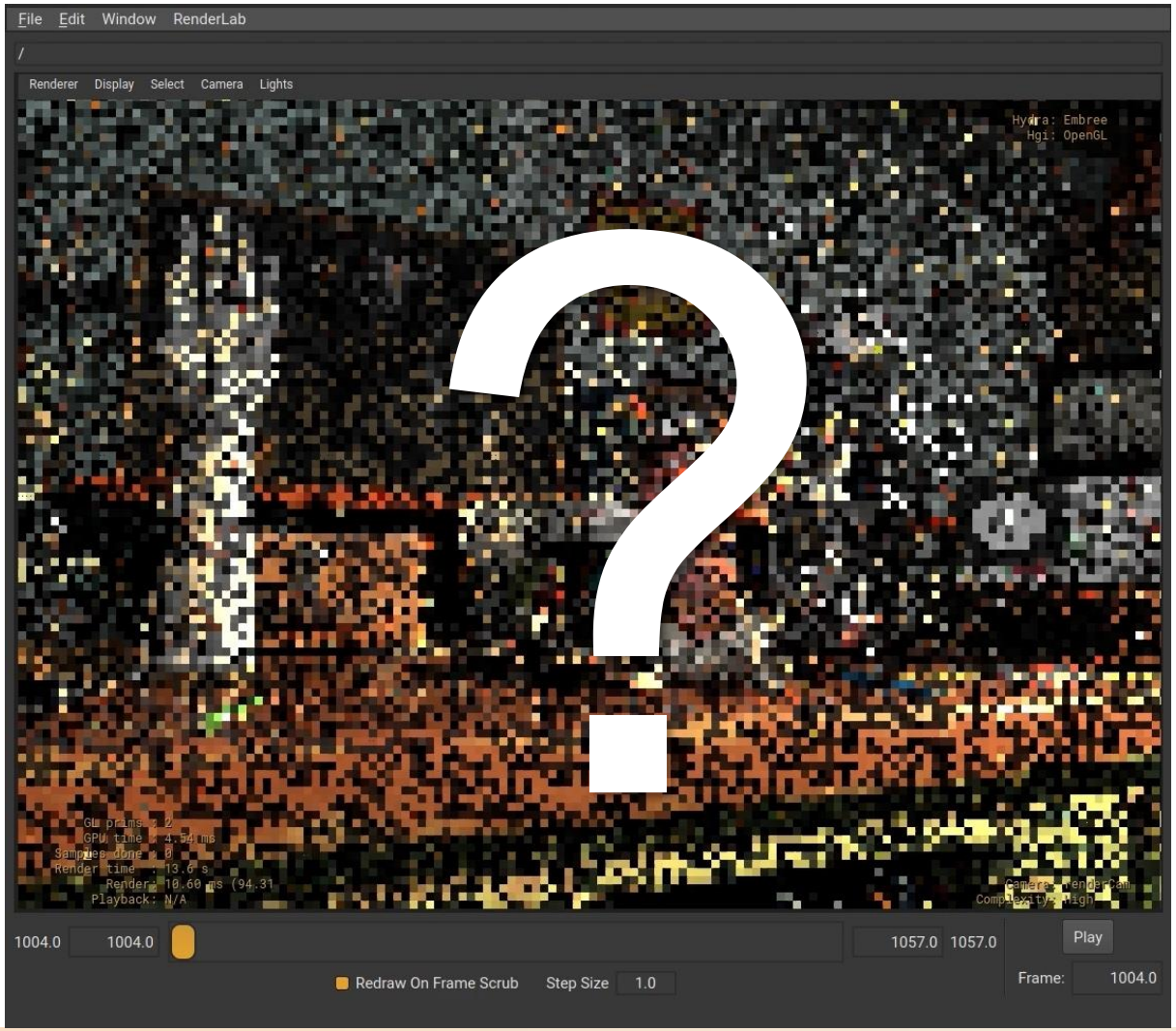
- The same USD / MaterialX asset produces different final images across DCCs and renderers.
- Drift occurs in light transport, material evaluation, lights, and cameras.
- A shared reference makes interpretation ambiguity visible and actionable.



DF Talk [https://note.com/dftalk/n/n55483de2bf3e]

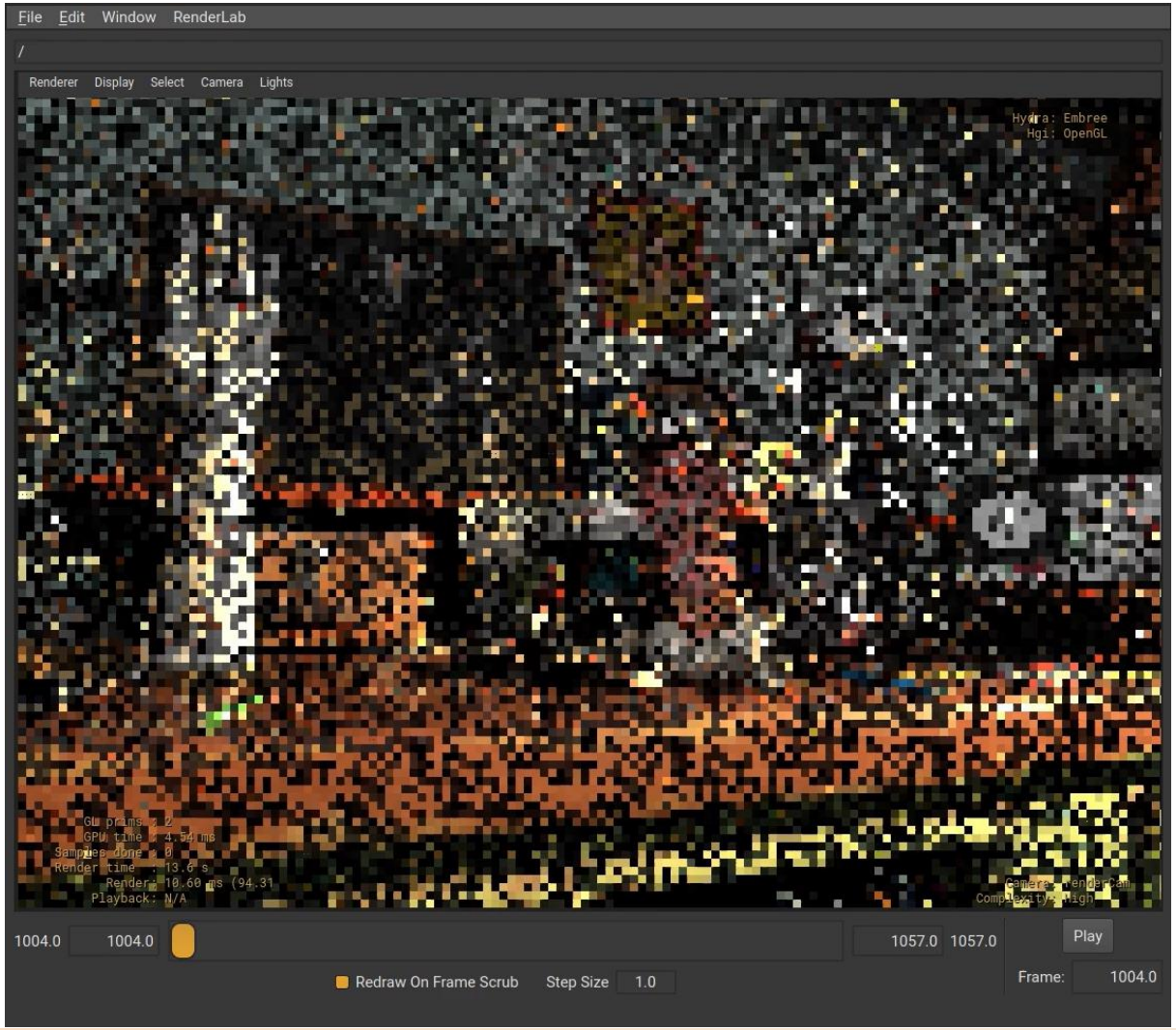
We Need a Reference Renderer

- We need a **reference path tracer**
- But that's far, far too much work



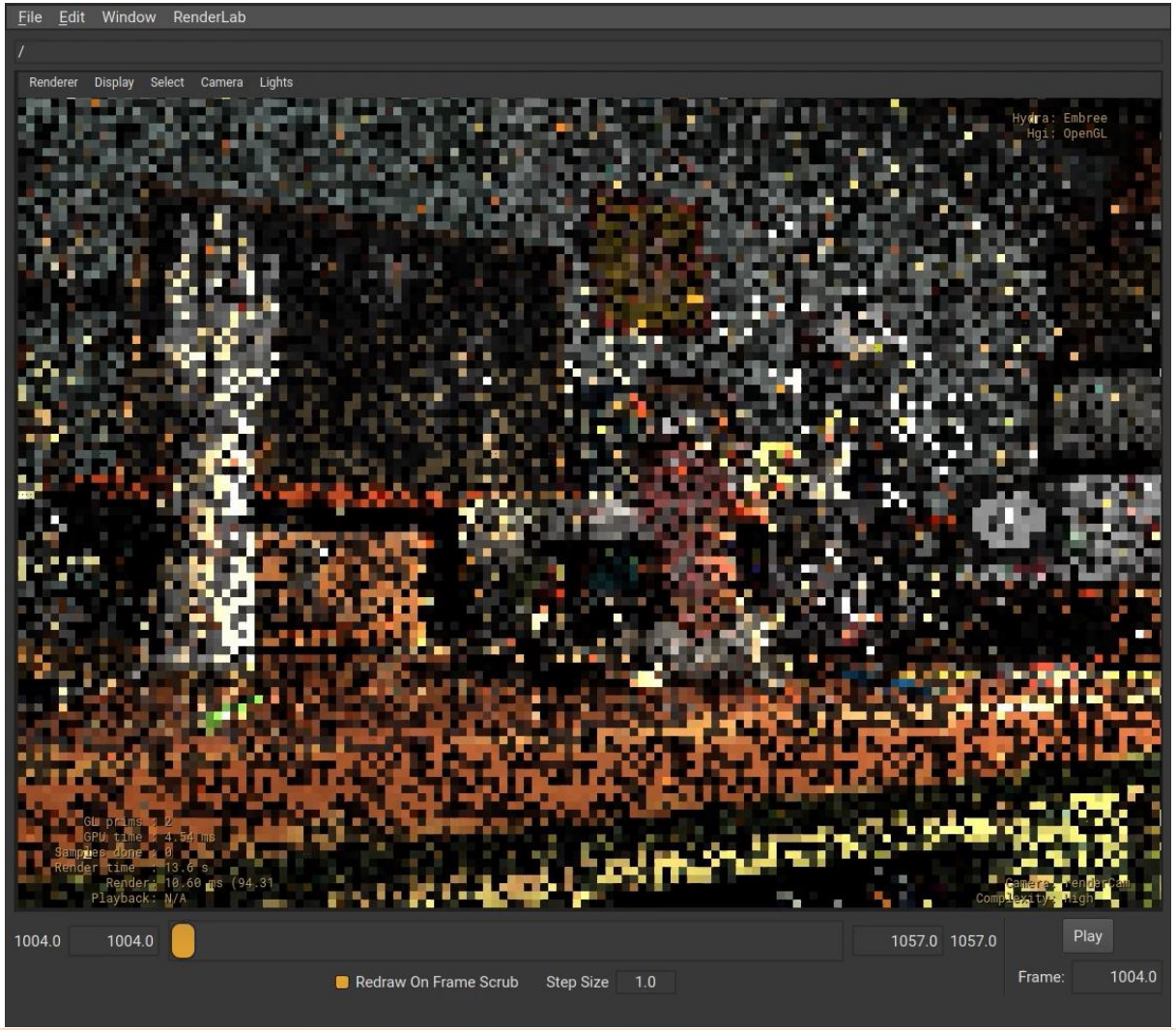
We Need a Reference Renderer

- We need a **reference path tracer**
- But that's far, far too much work
- **…or is it?**



We Need a Reference Renderer

- We need a **reference path tracer**
- But that's far, far too much work
- **…or is it?**
- **Maybe not – but specification, validation and reference is more important than ever**



The Proposal

/* ACADEMY SOFTWARE FOUNDATION

open Source days '26

- Extend hdEmbree into **Typhoon**
- Propose Typhoon as the reference renderer for OpenUSD
- Use brute-force full path tracing as the final-image baseline
- Ideally: release as part of OpenUSD

From hdEmbree to Typhoon

- hdEmbree was a minimal delegate
- UsdLux reference added direct lighting
- Typhoon adds full path tracing and material coverage while staying readable.

What Typhoon Is

- Full path tracing renderer focused on consistent reference behavior.
- Brute-force baseline for final-image light transport.
- C++ MaterialX implementation independent of the existing shader generator.
- Readable implementation for validation, testing, and community proposals.
- CPU-only renderer — runs on Linux, macOS, and Windows, CI, without GPU dependencies.

What Typhoon Is Not

- **Not** a replacement for real-time viewport renderers
- **Not** a production renderer
- **ALL** software has trade-offs – Typhoon's are specifically designed for development of OpenUSD standards

Built on Open-Source Foundations

/* ACADEMY SOFTWARE FOUNDATION

open Source days '26

intel EMBREE

OpenQMC

OpenUSD

OIIO

OSL

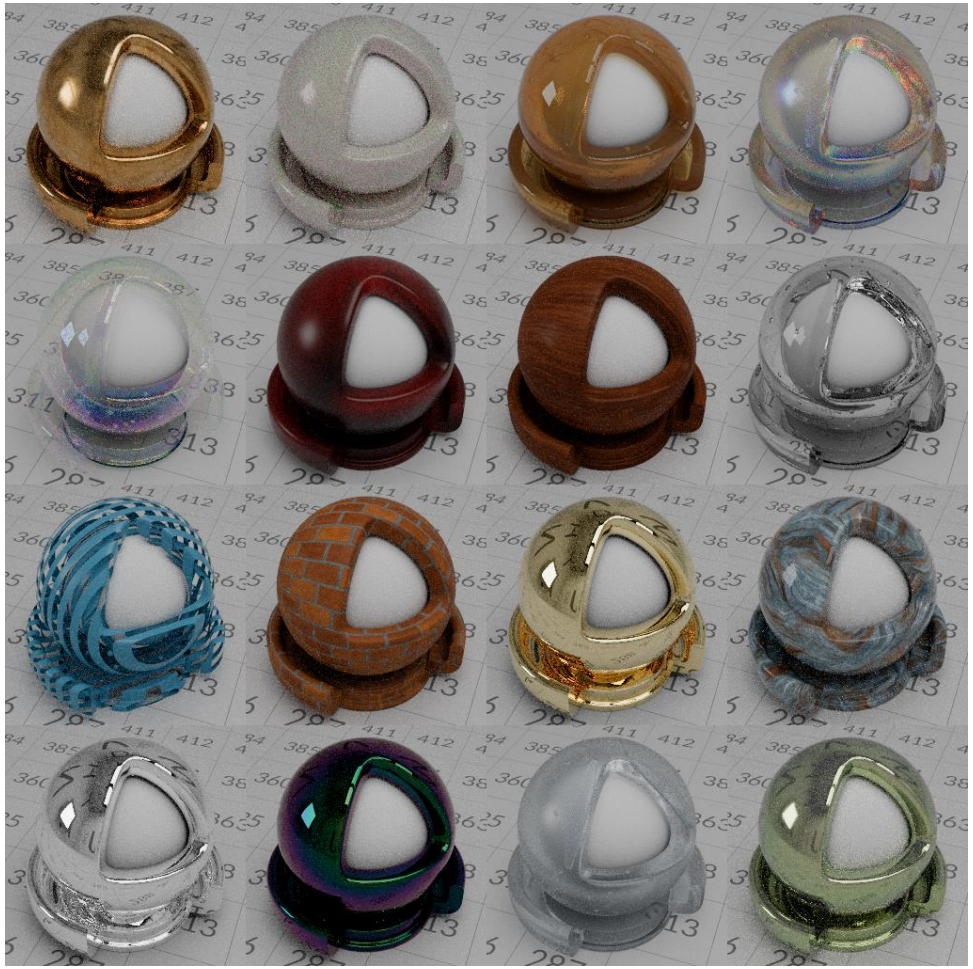
(BSDL)

openpbr

MaterialX Nodes

- Does not use MaterialX shader generator
- UsdShade only
- Implements all standard nodes*

**haven't gotten to hair or displacement or flake yet, but everything else is done*

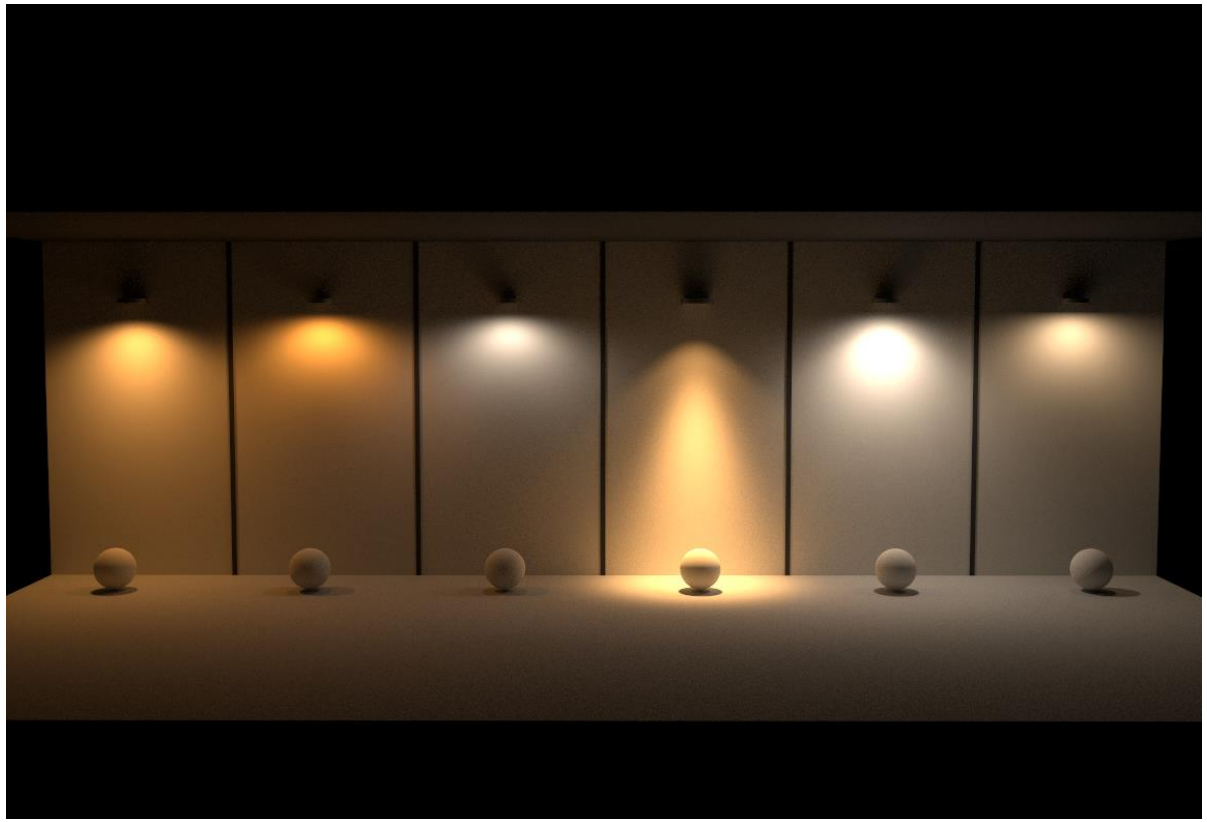


OpenPBR Support

- Implements MaterialX material models including OpenPBR.
- Includes Typhoon's native OpenPBR implementation.
- Also integrates the Adobe OpenPBR implementation.
- Covers features required for modern material validation.

Beyond Surface Shading

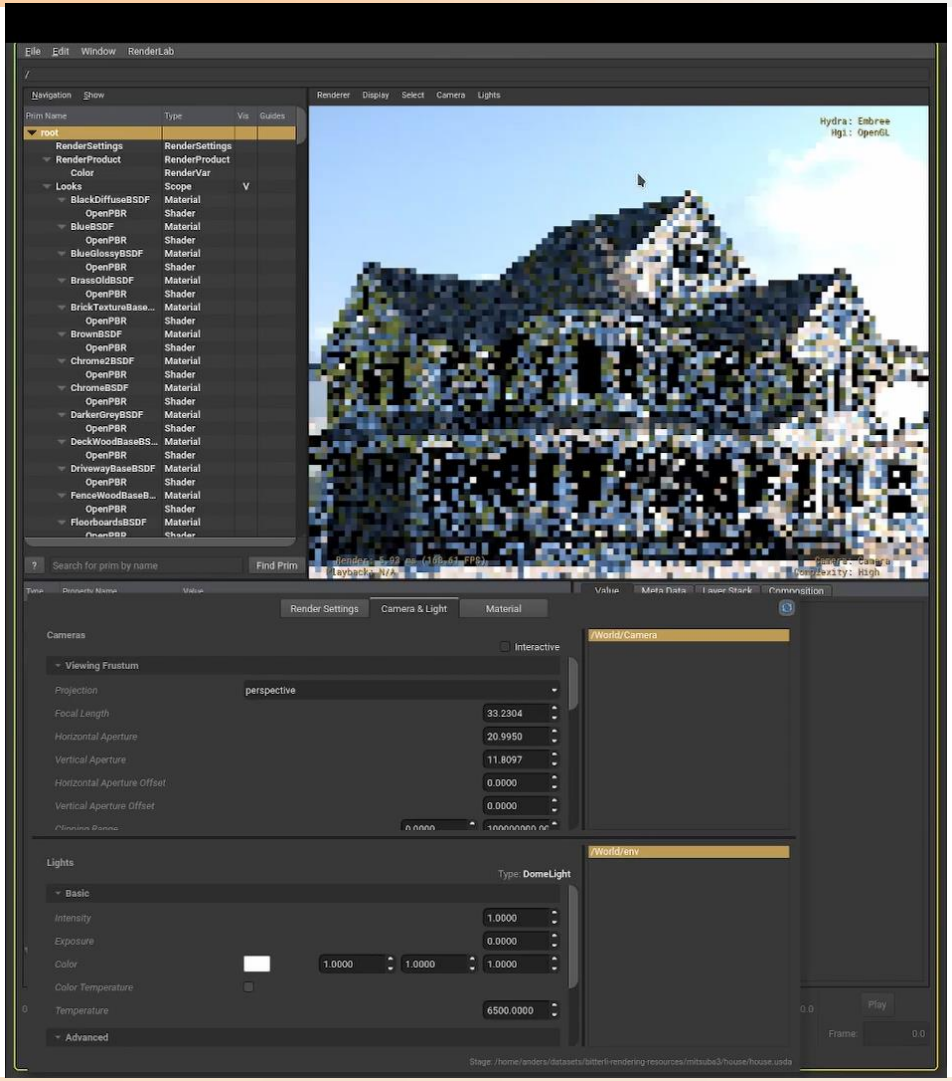
- Random-walk SSS, volumetrics
- Lights, cameras (UsdLux spec)
- Full path tracing exposes interactions invisible to rasterized or local models.
- Goal: consistent interpretation of the full USD asset.



Physical Lighting Implementation

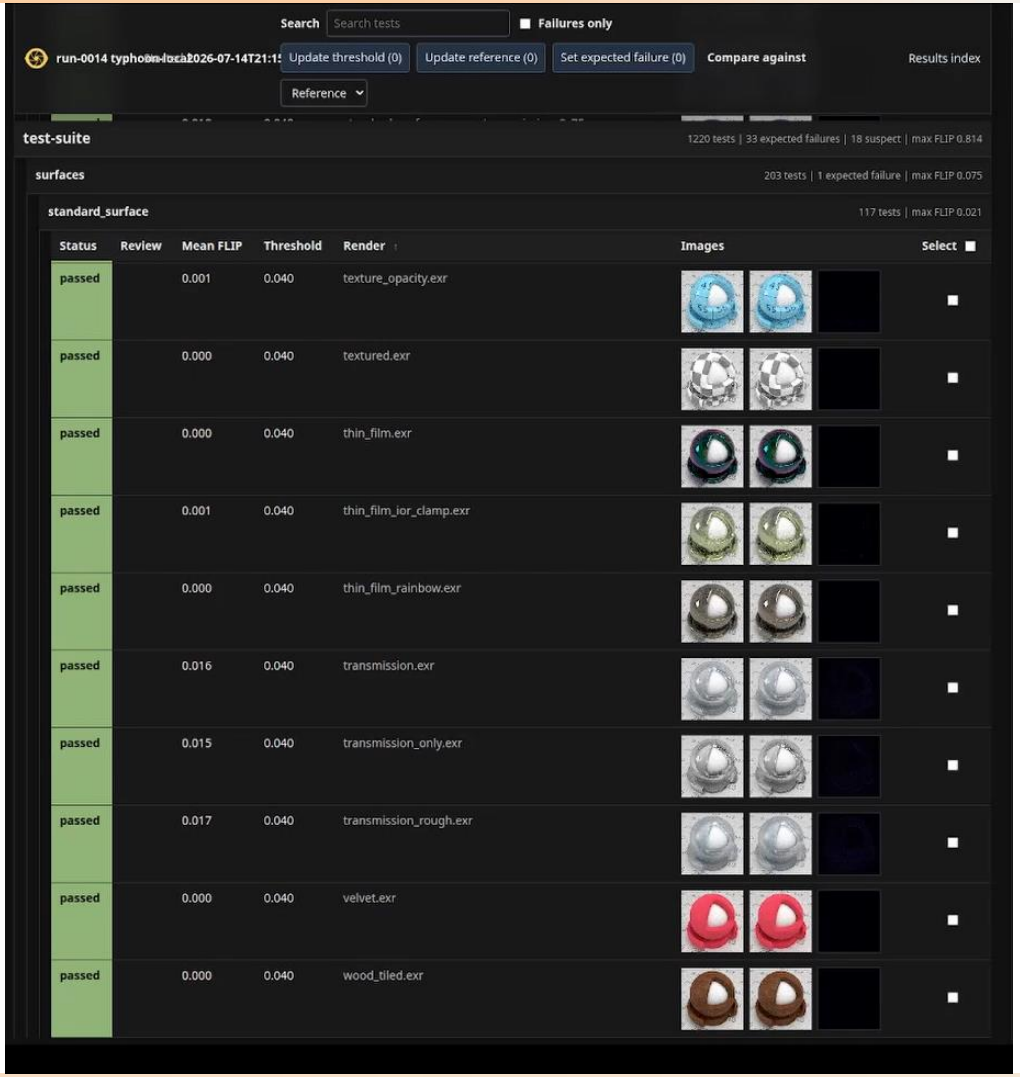
- Specify lights in photometric units
- Implementation in Hydra
- Typhoon gives us an example of how to use it and what it looks like

```
Le = HdLight::ComputePhysicalScalingFactor(delegate, id)
                * color * intensity * pow(2, exposure);
```



AOUSD Materials Specification

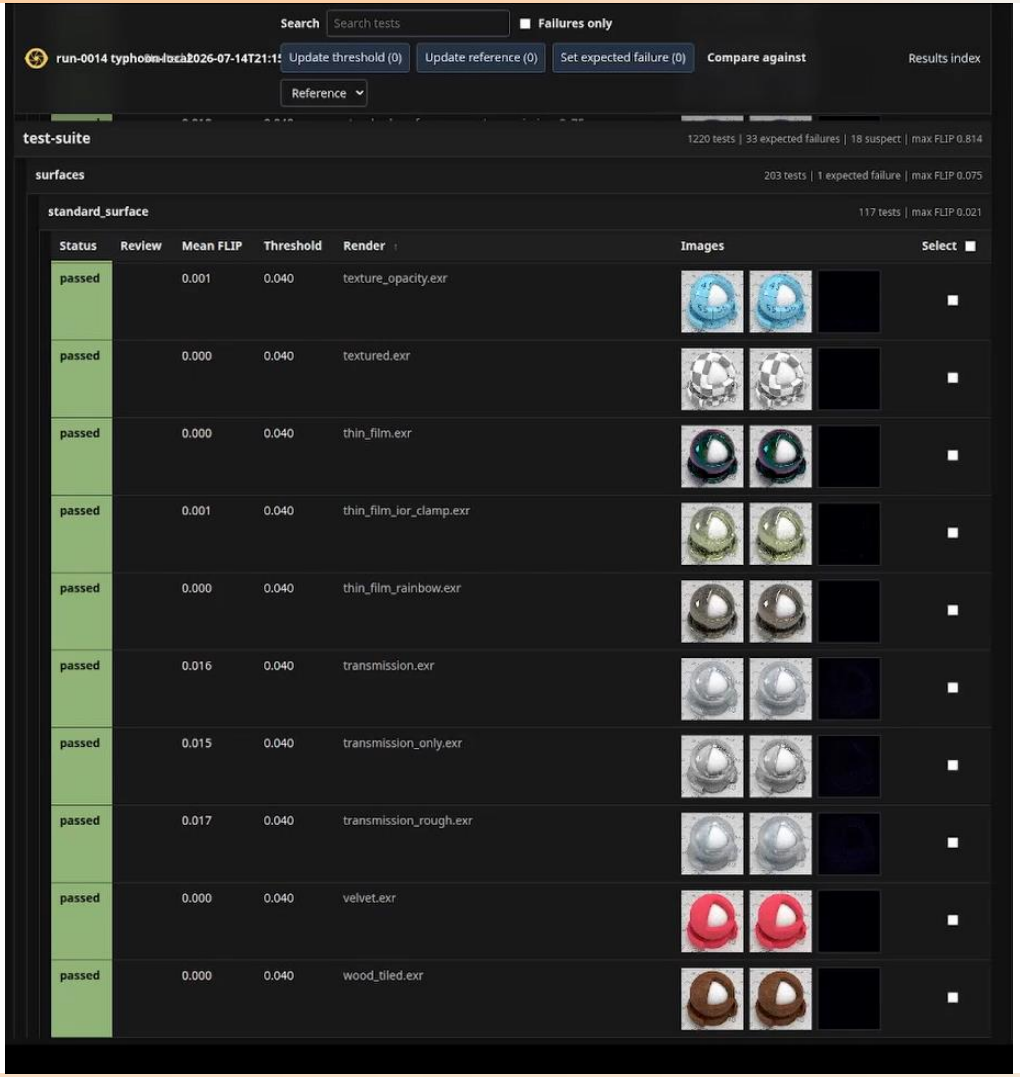
- Reference implementation
- Conformance test suite
- >1200 tests and counting
- Big thanks to Ben Houston's MaterialX Fidelity tests



AOUSD Materials Specification

- Reference implementation
- Conformance test suite
- >1200 tests and counting

```
pixi add goldeye
pixi run goldeneye init
pixi run pytest
```



Trying Typhoon for Yourself

- Just run:

```
pixi exec                                                    \
 --spec openusd-typhoon                                      \
 --channel https://conda.anaconda.org/anderslanglands        \
 usdview ALab/entry.usda
```

Trying Typhoon for Yourself

- Just run:

```
pixi exec                                                    \
 --spec openusd-typhoon                                      \
 --channel https://conda.anaconda.org/anderslanglands        \
 usdview ALab/entry.usda
```

- ~50k SLOC
  - ~8k "Renderer proper" (Integrators, core, lights, sampling etc)
  - ~4k Delegate
  - ~35k Materials (node implementations, graph etc)

Takeaways

- Typhoon is a reference path tracer for OpenUSD
- Typhoon is **native USD** – no behaviour other than what we specify, no customer other than the USD community
- Single focal point for community reference implementations
- Easy-to-use USD testing framework to evaluate implementations against reference

Takeaways

- Typhoon is a reference path tracer for OpenUSD
- Typhoon is **native USD** – no behaviour other than what we specify, no customer other than the USD community
- Single focal point for community reference implementations
- Easy-to-use USD testing framework to evaluate implementations against reference
- **Reference implementation is an accelerator**
- **Validation is the value**

/* ACADEMY SOFTWARE FOUNDATION

open source days '26