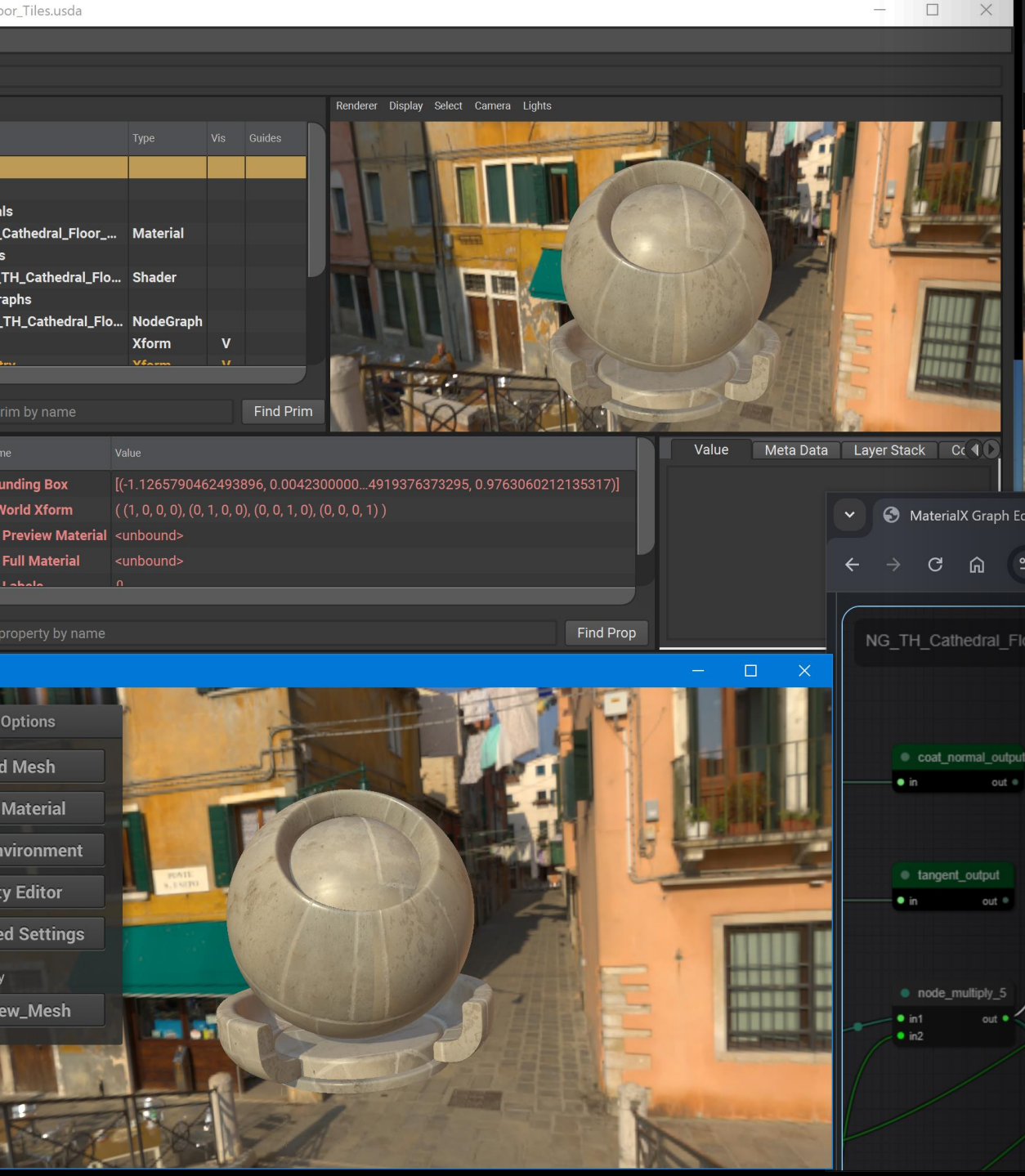
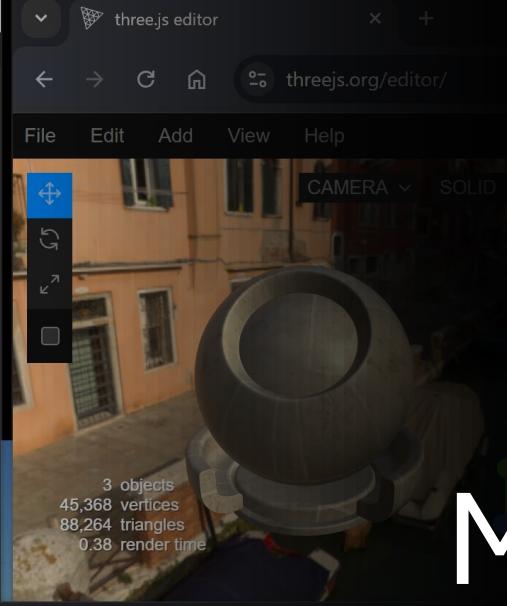
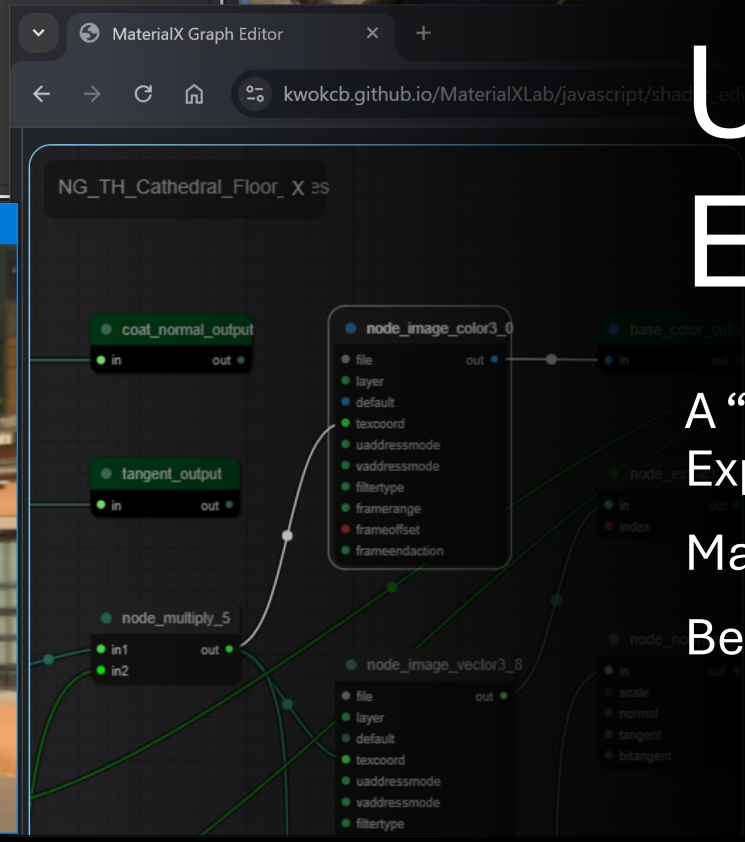






MaterialX / USD Equivalency

A “Render Test Suite Experiment”

March 6, 2025

Bernard Kwok

Overview

Target	"Validity"	Evaluation	Verification	Decorations
Building blocks for: MaterialX / USD equivalency test	MaterialX and USD consider different node graph configurations to be valid. Concept of "renderable" differs. Some graph configurations not representable.	Different graph configurations required to perform the same evaluation.	Single source, multiple validation tools. No way to confirm lossless interop (e.g. no USD to MaterialX)	Binding semantics and logic can differ or not be defined. E.g. handed-ness, matrix order, lighting, units

Result: "Hit and miss" to determine if a MaterialX graph can be supported in USD.

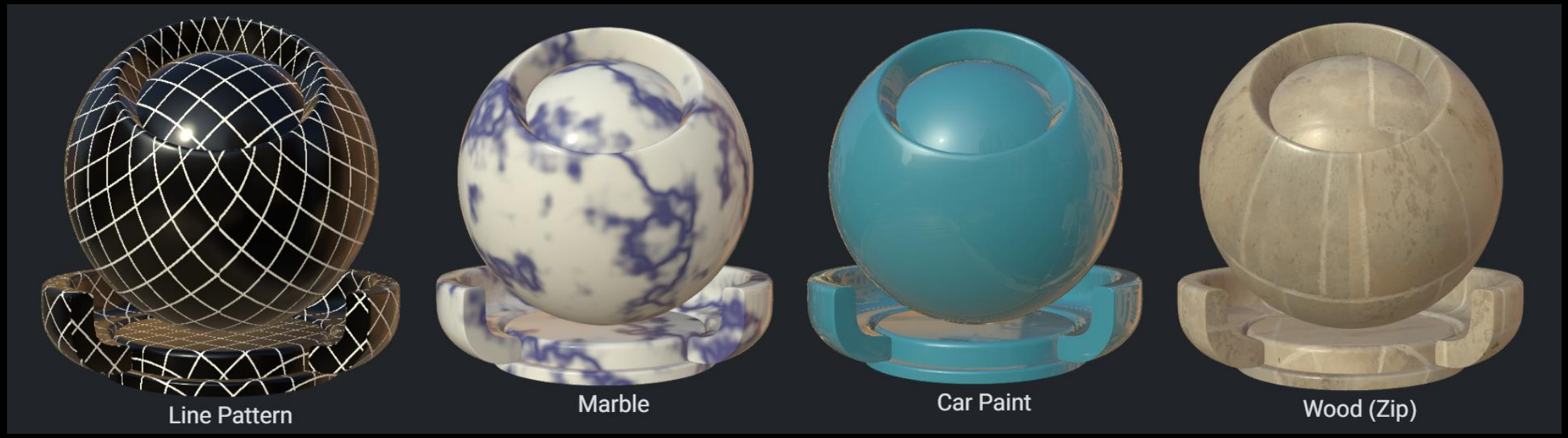
Process Outline

Target	Bindings	Distillation	Validation	Triage
Build valid USD stages per “renderable” element.	Define USD scene camera, lights, geometry. Pro: Common reusable content definitions.	Semantic and connection remapping on MaterialX Remap scene bindings as needed (e.g. streams).	Iterate using conversion and validation (USD checker, MTLX validation) Formal syntax checking via schema missing: MTLX / USD (?).	“Guess” on missing conversion content Handle validation errors. Handle binding errors. Go to bindings and distillation.

Result: Working backwards from image comparisons is difficult.

Results

kwokcb.github.io/materialxusd/



User Perspective

- How to tell what is valid in the MaterialX vs USD domain
- How to ensure that MaterialX and USD represent the same evaluation
- How to know why something does not work.
- How to create / edit shading networks to always work in all domains.
- How to know what and when information is lost.
- How to know what only works in one domain or the other.

Implementation Level Knowledge Required

- Specification versus implementation differences (MTLX / USD)
- Specification of MaterialX graph configurations supported by USD
- Semantic differences
- Validation differences
- Evaluation differences
- Rendering differences

Structural and Semantic Differences

MaterialX	USD
Arbitrary nodes at any level. Node graph nesting allowed but not supported.	Shading models, materials must exist at top level Other nodes must be encapsulated within top level or nested node graphs.
Arbitrary PBR node graphs are supported. PBR data types can flow between graphs and nodes.	Roots must be existing known shading models. Graphs cannot have pbr type ports (?)
Explicit node <-> graph level connections. Implicit output port connections allowed conditionally.	Explicit port to port connections
Outputs, surface / volume shaders and materials are “renderable”	Materials are “renderable”
MaterialX specific path pre-resolution exists and must be pre-resolved for USD usage.	Not all MaterialX path resolution logic supported (e.g. fileprefix).
Graphs can be arbitrarily merged / split and maintain validity.	Non-material rooted graphs are not allowed (?)

Evaluation Differences

MaterialX	USD
Validation is “optional”. Invalid data can exist at runtime.	Invalid data cannot exist / be loaded at runtime. E.g. No partial loading.
Scene dependencies not required for validation. E.g. Up axis, units.	Scene dependencies are required for validation
Lighting is not formally specified	Lighting is formally specified
Rendering parameters are not formally specified.	Rendering parameters are formally specified.
Geometric bindings can be implicit with defaults at definition level.	Geometric bindings require explicit nodes
Value and connections are invalid.	Connections can override values.
Documents can be used for interop without definitions.	Partial or no interop can occur without definitions

Disclaimer

- This document is based on observational knowledge of USD and needs to be fully vetted.
- The tests are based on a build of MaterialX 1.39.2 and USD 25.02 built using this release. Ideally a released build of USD using a a 1.39.x release with corresponding PyPi package should be used.