Khronos glTF Procedural Graphs

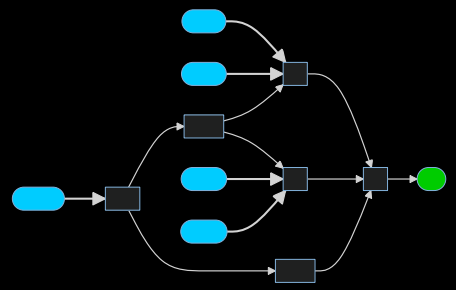
Date: March 12, 2024

Authors: Bernard Kwok and Ashwin Bhat

Introduction

What

Runtime representation of “procedural” graphs

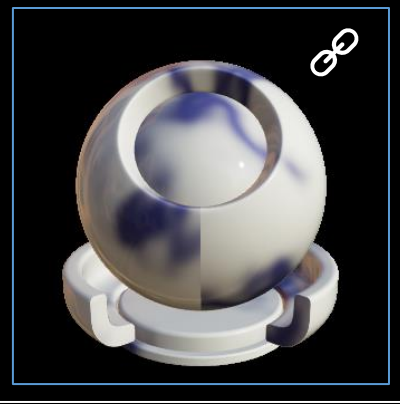


Where

Extension to Khronos glTF “material” description

Why

Give alternative to “distillation” (e.g “baking”)



When

Specification for 2024

Target MaterialX 1.39

Objectives

Interoperability

Adhere to "industry standard" schema

"Runtime friendly" format

Fidelity

Reduce / remove bespoke "distillation" process ★

Reduce asset / runtime overhead

Editability

Expose logic and interfaces ★

Reusable and Extensible ★

Runtime Editable (*)

Validation

Without "runtime" dependencies

Validate against "reference" rendering

(*) e.g. Support runtime glTF node editor

Representation

Standardized

Only use MaterialX node definitions ✪

All libraries representable (*)

Versioned

Schema fixed to MaterialX release version identifier. ✪

Separate glTF extension version id.

Runtime Format

JSON with glTF conventions. e.g. `[x,y,z]` vector syntax

Cannot be referenced ✪

Meta-data

Allow **all** meta-data to be maintained ✪

e.g. UI meta-data for user edits

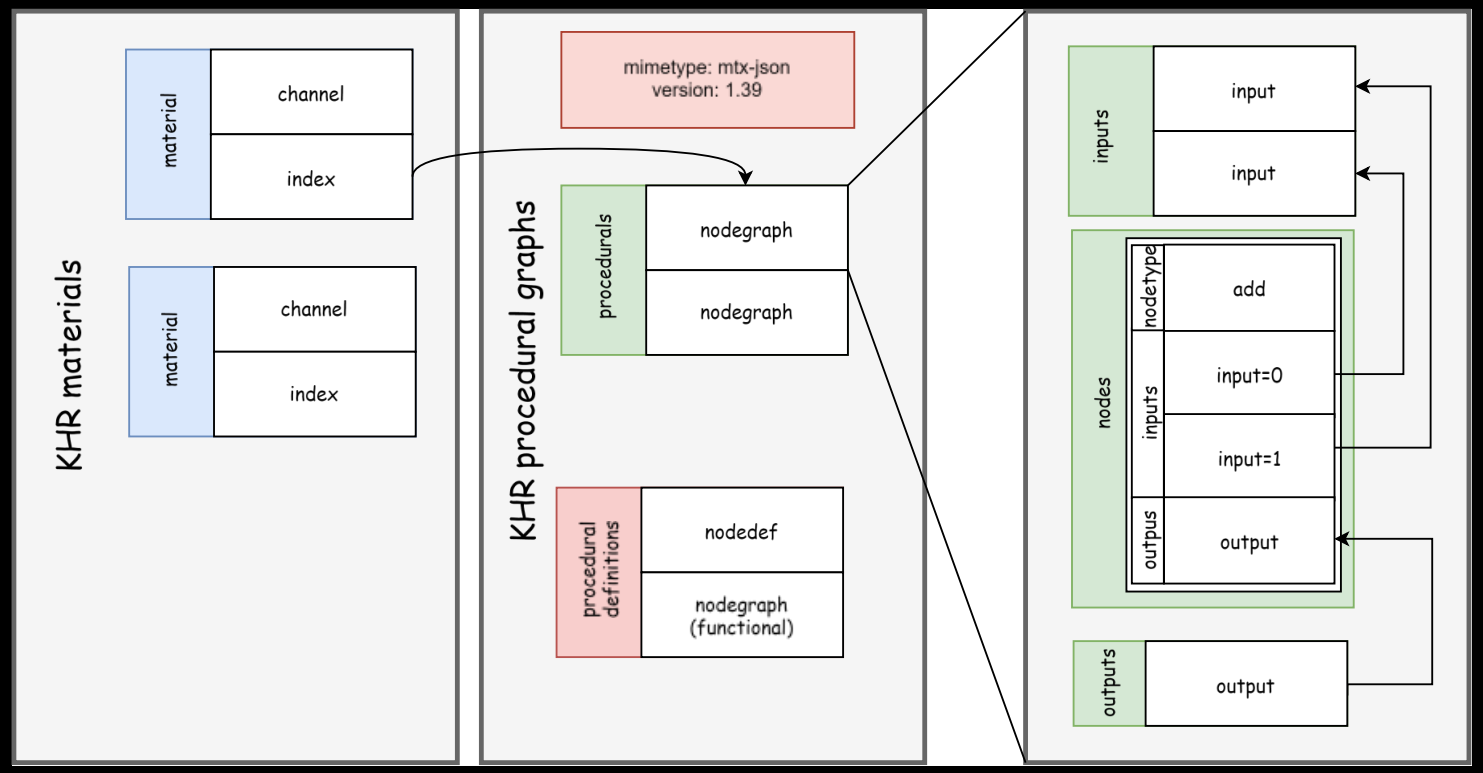
(*) Including “pbr”, “bxdf”, “nprlib”

Representation : Components

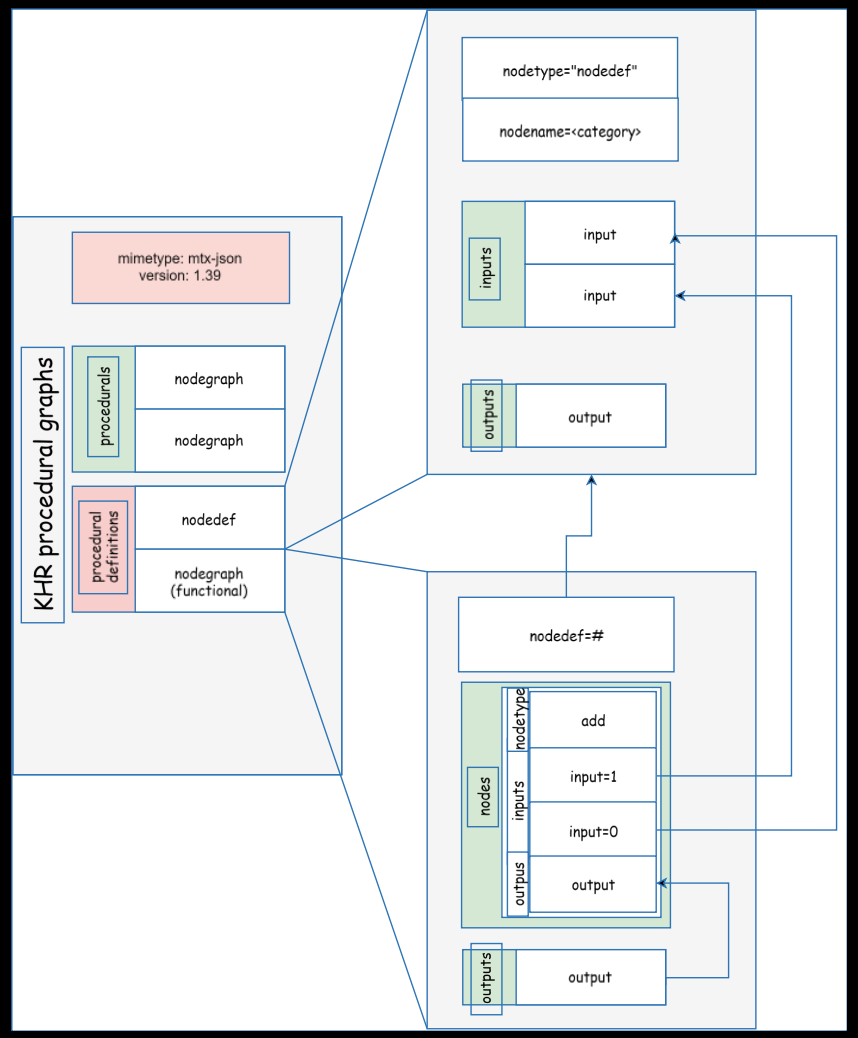
Types	Identifiers	Nodes + Graphs	Definitions
Boolean, numeric and “filenames” No strings, ✪ arrays, enums, structs (*)	Name/ path identifiers not required ✪ Reference by storage location (glTF convention)	Graphs ✪ required for “procedural” encapsulation Interface “publishing” required	Same approach as MaterialX Convertible to MaterialX / OpenUSD.

(*) struct exploration in progress for MaterialX

Representation : Components



Material Node Graphs



Node Definitions

Representation : Comparison

Feature	MaterialX	OpenUSD	glTF
Component String Identifiers	Yes	Yes	Optional
Numeric Tuples	string	list ()	array []
Connection Syntax	string (name in context)	Absolute Path	numeric index
Explicit Node Outputs	No	Yes	Yes
Node Type Grouping	No	No	Yes
Reference to nodedef on node instance	Yes (optional)	Yes	No
NodeGraph Nesting	"Yes" (not implemented)	Yes	No
Optional Input Overrides on Nodes	Yes	Yes	Yes
Referencing	Yes	Yes	No
Definition Versioning	Yes	Yes	Yes
Meta-Data	Yes	Yes	Yes
Node Definition	Yes	Yes	Yes

Fidelity : Patterns

Texture Procedurals

Version 1: Pattern graphs (stdlib, nprlib)

Additive: procedural and / or rasterized pattern ✪

Placement

Bypass matrix "baking" issues:

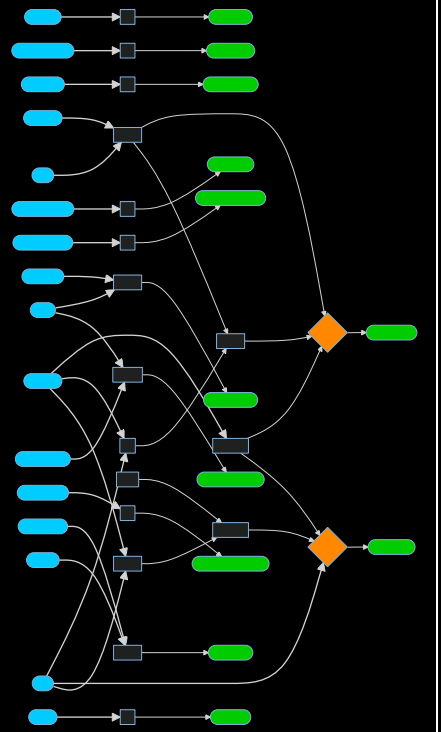
- Rotation order
- TRS order
- Pivot
- Angles: Radians

Fidelity : Shaders / Colors / Units

Shading Models

Bake not required to map models ★

Translation graphs supportable



Colorspace

Meta-data can be supported

Working colorspace fixed

Input space ★ glTF default: sRGB

Units

Meta-data can be supported

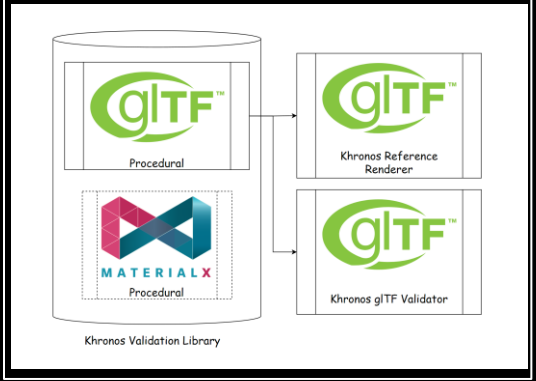
Add proper units to definitions (angles)

Integration

Validation

glTF Validator + test suite (V1+)

MaterialX glTF



Resource Binding

Stream bind detached from texture bind ✪

Name remapping still required

OpenUSD glTF

Tooling

Enhance MaterialX / OpenUSD tooling

glTF MATERIALX

V1

Runtime

Runtimes

- ThreeJS ?
- Bablyon.JS ?

Viewers

- Google model-viewer ?

Editors

- TBD ?

Closing

- Khronos Github repository: Coming soon ✪
- Runtime dependence: Future of Javascript / WASM MaterialX ?
- glTF / OpenUSD direct interop ?
- Requirement for JSON MaterialX ?
- OpenPBR to glTF translator ?