Differences Between MaterialX Specification v1.37REV2 and v1.38

Updated August 14, 2020

Material Nodes

Materials are now specified using nodes with a new "material" output type, with regular inputs with types such as "surfaceshader" and "displacementshader" taking the place of <shaderref> elements. This arrangement is far more flexible than the previous method and more closely resembles material construction in modern DCC applications, and also allows materials and their shaders to be accessed using the same API as that used for any other part of a shading nodegraph. The older <shaderref>, <bindparam>, <bindinput> and <bindtoken> elements are now deprecated.

Formalized Target Definition

The specification of different rendering targets is now formalized using a new <targetdef> element, which allows the explicit declaration of target inheritance. This allows a render target to be much more selective and descriptive about what node and shader implementations it can use, such as indicating that it can use "oslpattern" target nodes but not full "osl" targets with closures.

Node Parameters Are Now Inputs

Previously, many float/colorN/vectorN arguments to nodes were parameters, which greatly limited their usability in many constructs. Now, essentially all float/colorN/vectorN node arguments are now connectable inputs: "default" and "frameoffset" for Texture nodes, "value" for <constant>, <ramp>s and <split>s, "amplitude"/"pivot"/"octaves"/etc. for <noise> and <fractal> nodes, "low" and "high" for <clamp>, "axis" for <rotate3d>, "which" for <switch>, etc.

Additionally, having both <parameter>s and <input>s meant that many applications needed to iterate over all inputs to perform an operation, then iterate again over all the parameters to perform the same operation, leading to much duplicated code. So the <parameter> element has been removed altogether, replaced by <input>s defined with a `uniform="true"` attribute in their NodeDefs. This change affects the"index" input for the Standard Geometric Nodes, "lumacoeffs" for <luminance>, as well as all filename- and string-type arguments.

Physically Based Shader Layering

Several of the PBR nodes, such as <dielectric_brdf> and <sheen_brdf>, are generally intended to be vertically layered over other BSDFs, and had a "base" input for this base layer. We now use a standalone <layer> node for this purpose, using the reflectance amounts contained in the BSDF output structs. This allows for much more flexibility in BSDF layering.

Color2 Type Removed

The "color2" and "color2array" types have been removed, since there hasn't proven to be any practical use for these types in 3D rendering applications which have become the primary focus of MaterialX.

UiFolder Elements

Nodedef elements may now contain any number of <uifolder> elements to document a folder structure for the node, including a "doc" attribute to provide folder-level documentation strings.

Other New Operator Nodes

The following new standard operator nodes have been added:

- Application nodes: **updirection**
- Math nodes: **round**, **safepower**
- Adjustment nodes: **curvelookup**

Additionally:

- The input arguments for <atan2> have been renamed to "iny" and "inx" to be more clear about which input is which.
- The <switch> node has been expanded to have 10 inputs rather than just 5.
- The <normalmap> node's "scale" input can now be either a float or a vector2.

Other Changes

- The <backdrop> node now has a boolean "minimized" attribute, indicating whether the backdrop is collapsed down to a single node-sized "box" in an application's UI.
- All nodes may optionally specify "width" and "height" attributes.
- A <nodegraph> that does not specify a "nodedef" and is thus not an implementation of a node may specify a number of direct child <input> elements which its contained nodes can access using "interfacename" attributes and which may be connected to other nodes outside the nodegraph using "nodename" attributes. This allows a <nodegraph> to be used as a collapsible container for nodes with named inputs and outputs.
- A new "sourcecode" attribute has been added to <implementation>, allowing specification of direct inline code without requiring an external file. Additionally, the "language" attribute of <implementation> has been removed, replaced by a new "format" attribute, to indicate whether the implementation source code is a complete "shader" or is a "fragment" to be processed by a code generator such as ShaderGen.
- Custom attributes can now be output as metadata in generated shaders by adding an exportable="true" attribute to the <attributedef>.
- The <invert> node is now deprecated, since it is identical to a <subtract> node with different input names.
- The unimplemented {CONTAINER} image filename string has been removed.
- Other minor edits and clarifications.