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

Updated December 23, 2019

Units

MaterialX now supports the declaration of units and unit types for parameters, inputs and geometric properties. For example, a nodedef could declare that a certain parameter is a "distance" unittype, and invocations of that node could declare that a value for that parameter is expressed in e.g. "centimeters" or "inches". Combined with an application's declaration of a preferred working "scene unit" for various unit types, this makes it possible for MaterialX to automatically convert values from one unit to another.

The <tiledimage> supplemental node now has "realworldimagesize" and "realworldtilesize" parameters which can be used to indicate the real-world size of a tiled texture sample and apply the texture into a specified real-world UV unit tile size.

More Robust Node Signature Variants

Previously, there were a number of nodes that had different combinations of allowable input or parameter names, such as the <rotate> node with an "axis" parameter only applicable to the vector3 variant, or one set of variants of the <transformX> nodes using "fromspace"/"tospace" and others using "mat". This turned out to make it difficult or even impossible to resolve exactly which nodedef signature was to be used in some cases, especially when some inputs were unconnected and intended to use their default values. In 1.37REV2 we have now split all of these nodes into separate node types, so that all variants of a node now have exactly the same set of input and parameter names (only the types of those inputs/parameters vary from one <nodedef> to another). This makes the determination of which nodedef to use much more robust, and also makes MaterialX-based node creation UIs more straightforward.

The following node types are affected:

- <invert> for matrices is now called <invertmatrix>
- <transformpoint>, <transformvector>, <transformnormal> now only have fromspace/tospace variants; to transform a vector by a matrix, use the new <transformmatrix> node
- <rotate> is now split into separate <rotate2d> and <rotate3d> nodes
- The <combine> node has been replaced with separate <combine2>, <combine3> and <combine4> nodes, reflecting the number of input connections.
- The <separate> supplemental node has been replaced with separate <separate2>, <separate3> and <separate4> nodes, reflecting the number of outputs.

New Comparison Nodes

The <compare> node is now replaced by new <ifgreater>, <ifgreatereq> and <ifequal> nodes, providing a complete set of value comparison operations.

Additional Changes in 1.37REV2:

- The reserved "none" type has been removed, as it is no longer needed with explicit node <output> declarations. Nodes are expected to define at least one output.
- Because it has no outputs, <backdrop> is now a separate element type rather than being a "node".
- The <dot> node is now considered a trivial "math" node rather than "organizational".
- New uisoftmin/uisoftmax and uistep attributes have been added for nodedef parameters and inputs.

Differences Between MaterialX Specification v1.36 and v1.37

New Physically-Based Shading Nodes

In support of the incorporation of ShaderX into MaterialX, the following new standard nodes have been added into a new PBR library:

- PBR Data Types: **BSDF**, **EDF**, **VDF**
- BSDF nodes: **diffuse_brdf**, **diffuse_btdf**, **burley_diffuse_brdf**, **dielectric_brdf**, **dielectric_btdf**, **conductor_brdf**, **generalized_schlick_brdf**, **subsurface_brdf**, **sheen_brdf**, **thin_film_brdf**
- EDF nodes: **uniform_edf**, **conical_edf**, **measured_edf**
- VDF nodes: **absorbtion_vdf**, **anisotropic_vdf**
- Shader construction nodes: **surface**, **thin_surface**, **volume**, **light**, **displacement**
- Shader utility nodes: **backfacing**, **roughness_anisotropy**, **roughness_dual**, **glossiness_anisotropy**, **blackbody**, **complex_ior**, **artistic_ior**, **fresnel**

Additionally:

- The <mix> and <add> operators have been extended to support mixing or adding of two BSDFs, EDFs or VDFs, and the <multiply> node has been extended to support scaling a BSDF, EDF or VDF by a float or color3.

The Physically-Based Shading Operators and some example PBR shaders built from these nodes are described in a new MaterialX Physically-Based Shading Nodes document on materialx.org.

Other New Operator Nodes

The following new standard operator nodes have been added:

- Procedural nodes: **worleynoise2d**, **worleynoise3d**
- Math nodes: **place2d**, **normalmap**, **transformmatrix**

Additionally:

- The <convert> operator has been extended to allow conversion between vector2 and vector3, and between vector3 and vector4, by adding or removing an extra channel with value "1.0".

Nodedef Type Declaration Moved to Child Output Elements

In previous versions of MaterialX, the output type of nodedefs with a single output was declared in the <nodedef> itself (and that output was nameless), while the output names and types of nodedefs with multiple outputs was declared using a number of child <output> elements. This meant that single and multiple output nodedefs had to be handled differently in applications. In 1.37, we have unified the syntax for nodedef output type declaration to always be done using child <output> elements: while this does make MaterialX files a bit more complex, it makes application coding for parsing nodedefs and handling node connections simpler, and it is now possible to declare a default or defaultinput for each output.

Geometric Properties

Custom geometric properties may be declared using the new <geompropdef> element, and specific values for these geometric properties may be defined for specific geometries using a <geomprop> element within a <geominfo>. These can be completely custom properties, or standard properties defined in a specific space or index. Geometric properties are now functionally equivalent to USD primvars. With the introduction of <geompropdef> the syntax for the `internalgeomprops` attribute on <nodedef>s has been simplified to only take a list of geometric properties, which may be any combination of standard and custom properties. The value of a custom geometric property may be accessed in a nodegraph using the new <geompropvalue> node.

GeomAttr/GeomAttrValue/GeomAttrDefault Deprecated

With the addition of custom geometric properties in 1.37 and of tokens in 1.36, the functionality previously provided by <geomattr>, <geomattrvalue> and <geomattrdefault> is no longer required as a separate element type, and so those three elements are deprecated in 1.37.

Swizzle-On-Input Reinstated

The "channels" attribute for <input> elements was removed in 1.36 in favor of forcing the use of explicit type conversion elements such as <extract>, <convert> or <swizzle>. This removal proved to be burdensome for certain applications which relied on this feature, so it has been reinstated.

Look Groups

A new <lookgroup> element has been created which allows a number of looks to be grouped together as a set for organization purposes. This could be used to list the set of looks defined for an asset, or other uses.

Other Changes

- The definition of "uvtiling" in the <tiledimage> supplemental node was incorrect in 1.36; it has now been correctly defined as a multiplier on the incoming texture coordinates (The OSL shader implementation was correct, only the specification was in error). Additionally, the "uvtiling" and "uvoffset" parameters of <tiledimage> are now defined as connectable inputs.
- New "uivisible" and "uiadvanced" standard boolean attributes have been added to indicate if a parameter/input/token within a nodedef or node instantiation should be visible in the application UI by default or not, and if visible, whether it should only be displayed in an application-defined "advanced" mode.
- Default values have now been rigorously defined for all inputs and parameters for all standard nodes. As such, there isn't a compelling reason to declare some as "required" and some as "optional" (and nothing in the library actually checked this), so this distinction has been removed from the spec.
- In a number of standard nodes, certain parameters have been changed to be connectable inputs:
    - <rotate2d> "amount"
    - <contrast> "amount"
    - <hsvadjust> "amount"
    - <remap> "inlow", "inhigh", "outlow", "outhigh"
    - <range> "inlow", "inhigh", "gamma", "outlow", "outhigh"
    - <smoothstep> "min", "max"
- The math nodes <sqrt>, <ln>, <exp>, <sin>, <cos>, <tan>, <asin>, <acos> and <atan2> are now defined to accept vector*N* input types as well as float, applying the math operation channel-by-channel.
- A new "mirror" mode has been defined for u/vaddressmode and frameendaction parameters, while the "black" mode has been changed to "constant" and now returns the value of the node's "default" value rather than zero.
- Other minor edits, document reorganizations and clarifications.