Differences Between MaterialX Specification v1.34 and v1.35

Visibility Assignments
The standard properties "invisible", "vistocamera", "vistoshadow" and "vistosecondary" as well as light illumination and shadowing assignments have been replaced by a general categorized <visibility> element within looks, which is used to define different types of visibility between either a specified "view" geometry or "the render camera" and other asset geometry. There are currently four visibility types defined: camera, illumination, shadow, and secondary, with support for additional application-defined types.

Lights
The specialized <light> element has been removed, in favor of instead handling the assignment of light shaders using regular <materialassign> elements within looks; illumination/shadowing assignments now use the new generalized visibility mechanism described above.

Shaderref Bindings
Shaderrefs are now required within <material>s to specify what shader node (or now, nodedef) various bindings apply to, and <bindparam>/<bindinput>s now apply exclusively to the <shaderref> element they are enclosed within: the `shaderref` attribute of <bindparam>/<bindinput>s has been removed. Previously, the spec allowed standalone <bindparam>/<bindinput>s to apply to a shaderref in an inherited material; this new mechanism is more explicit and robust.

Additionally, bindings of materialvar variation values is now done using explicit `materialvar` attributes in <bindparam>/<bindinput>/<override>s instead of using an "@*mtlvarname*" syntax.

Other Changes
- The `matrix` type is now called `matrix44`, and we have introduced a new `matrix33` type.
- We have introduced a `none` type, which is used as the output type for nodes such as <backdrop> which do not have an output.
- We have clarified various issues related to defaults for cmsconfig and document color spaces, in that these must now both be explicitly specified rather than relying on any default values, and have also clarified that MaterialX files document the colorspaces for all values and images as they existed in the application which created the file, but that it is the responsibility of the receiving application to use this colorspace information to convert colors if and as needed.
- We have removed the `texturecolorspace` attribute, in favor of explicitly specifying `colorspace` in <image>/etc. nodes as needed.
- The `wrap` mode in u/vaddressmode and frameendaction has been renamed to `periodic`, and the `mirror` mode has been removed.
- Primarily for implementation performance and clarity reasons, the `channels` parameter (which was intended to be used to mask what channels an operator affected while passing other channels through unchanged) has been removed from all Standard Nodes. The recently-added `channels` attribute of <input>s, which allows "swizzling" channels upon input for purposes of type conversion and channel extraction is unchanged.
- Added a <floor> Standard Operator.

- We have changed the interpretation of the <hueshift> operator to perform the hue rotation in standard HSV space rather than CIE XYZ: this change both reduces the chances for out-of-gamut results and more closely matches hue-rotation operators used in contemporary applications.
- Added a `lumacoeffs` parameter to the <saturate> and <luminance> operators, which can be driven by host applications to specify the luma coefficients of the current working color space to ensure that the luminance operations performed by these two operators are done in the right color space.
- The `pixels` parameter for the <blur> node has been removed.
- A new Standard UI attribute `helptext` has been added.
- Nodedefs for custom nodes may now specify a `target`, which allows nodes or shaders with the same node name to have different parameter names/types as needed.
- The `default` attribute of nodedef <parameter>/<input> elements has been renamed to `value`, and <input> elements for nodedefs now allow specification of a `defaultgeomprop` attribute in place of a (default) `value`.
- The `publicname` attribute for <parameter>/<input> elements within <nodedef>s has been removed (it was an error to have been included before).
- The `function` attribute of <implementation> elements must now be explicitly specified rather than defaulting to something.
- The mechanism for referencing nodegraph parameters (when a nodegraph is used as the implementation of a custom node) has been changed from a "$*paramname*" syntax to use an `interfacename` attribute instead.
- Added a `publicnameprefix` attribute, which can be used when invoking an instance of a custom node defined by a nodegraph which contains public parameter definitions within another nodegraph, or when binding the output of a nodegraph to a shader node using <bindinput>.
- Added "shader"-semantic type support for standard operator nodes <add>, <multiply> and <mix>, allowing post-shader blending of surface shaders using standard nodes.
- <Implementation>s can now include a number of <input> and <parameter> elements to define implementation-specific names, overriding the names defined in their <nodedef>. This makes it easier to support node implementations where official MaterialX parameter names conflicted with reserved words in a particular shading language.
- Added a new standard "matte" property for holdout-like geometry.
- Look assignments are now explicitly allowed on non-leaf geometry paths.