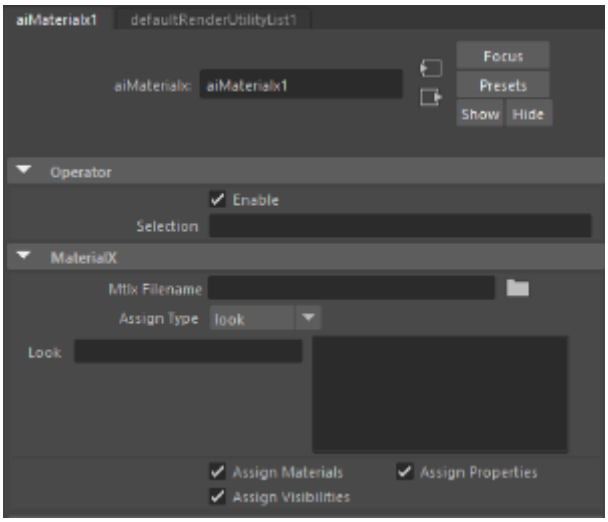
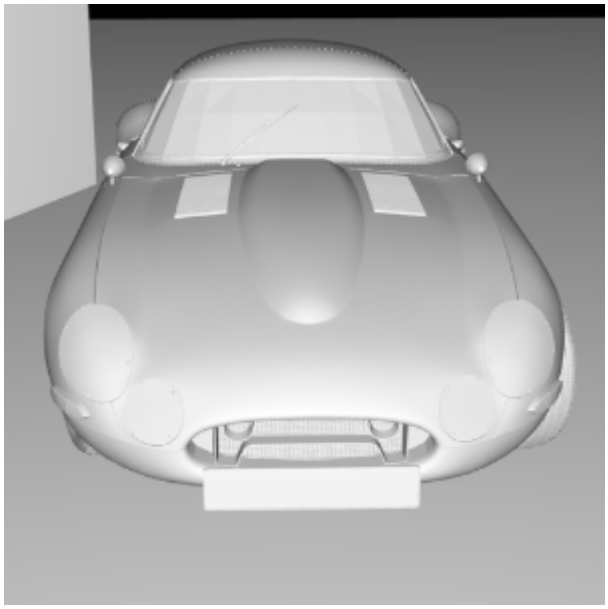
MaterialX



Applies a MaterialX look. The operator supports native Arnold shaders and the MaterialX standard library.

More information about *MaterialX* can be found here.



Without lookdev

With MaterialX lookdev

A tutorial about working with MaterialX and operators can be found here.

.mtlx files can be exported using Export Selected Shaders.

Enable

Enable/disable the *operator*. Disabled *operators* are bypassed when the operator graph is evaluated.

Selection

An expression to select which nodes this operator will affect. The expression syntax is described in the selection expression documentation, with some examples. Note that if the *operator* is connected to a *procedural* the selections are assumed to be relative to the procedural's namespace.

MaterialX

Mtlx Filename

The *MaterialX* source document containing one or more looks. The source document can either be a file or an inline XML string.

Look

Current look variant name that should be used from the *MaterialX* document.

Assign Materials

Enables/disables material assignments.

Assign Properties

Enables/disables property assignments.

Assign Visibilities

Enables/disables visibility assignments.

The *MaterialX operator* takes a .mtlx document or an inline XML string with one or more look variants and carries out the assignments for a given look, including material, property, and visibility assignments.

All the native Arnold shaders and *MaterialX* standard library shaders are supported (see node definitions below). Native Arnold shaders are always chosen over the standard library by default if there's a conflict. Where applicable, the textures are pre-processed to achieve better performance and ensure the correct look (see below). Standard library nodes, node graphs, and shading models are turned into Arnold *OSL* code using the *MaterialX* shader generation.

The following combinations of shaders in materials and node graphs are supported:

- All shaders including the shading model and connected nodes and node graphs are native Arnold shaders.
- Arnold shading model with a mixture of Arnold shaders and standard library shaders connected as nodes or node graphs. Note that a connected node graph cannot mix Arnold and standard library shaders. It is either an Arnold shading graph which is translated to their respective Arnold nodes, or it is a standard library graph which is turned into a single OSL shading node.
- A standard library shading model defined using the PBR library. No Arnold shaders can be connected to this shading model. The shading model and all connected inputs are turned into a single OSL shader.

MaterialX node definitions

Arnold ships with node definitions for the built-in Arnold shaders which can be found in the Arnold installation at *materialx/arnold/nodedefs.mtlx*. Arnold also ships with the *stdlib* and *pbrlib* node definitions, node graphs, and code snippets for generating OSL shaders, where the files are located under *materialx /stdlib* and *materialx/pbrlib*, respectively.

The environment variable `ARNOLD_MATERIALX_NODE_DEFINITIONS` can be used to set directories containing additional node definitions. All mtlx documents in the directories provided, including the subdirectories, are loaded as MaterialX libraries.

Texture pre-processing and color space

Textures used in the standard library shaders are converted to .tx files to achieve better performance through mip-mapping etc. Color textures can also be subject to color conversion as part of the pre-processing. The target rendering color space is provided by Arnold's color management node. If no explicit color management is used the rendering space is assumed to be linear sRGB. The source color space is read from the active *colorspace* attribute in the MaterialX document. The color space attribute is inherited and can be set on the texture parameter, shader node, node graph, etc. Nothing is done if a texture's source and target space are the same. Data textures are assumed to be raw.

Inputs

Connected upstream *operator* nodes.

Your browser does not support the HTML5 video element

The *MaterialX* operator can be used to assign per-object shaders, geometry properties, and visibility settings.

Note that changing the subdivision type requires a full update.

Right click video > 'Open video in new tab' to view fullscreen.