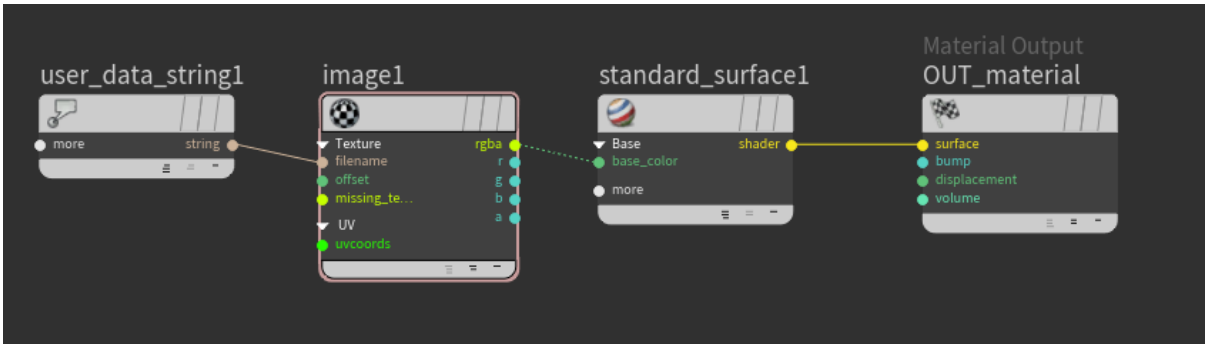
Data Types

Arnold Data Type colors on Houdini Nodes

When connecting the same data types together in a shading network the line will appear solid to indicate there is no conversion as in the 'string' connection below. If the data types are different, like in RGBA > Vector or RGB > RGBA below then the line appears as a broken line.



Solid line (between user_data_string and image) and a broken line between standard_surface and OUT_material)

Shader output components are displayed on the VOP node, allowing component connections without the need for a conversion shader. However, the following shaders allow more complicated conversions between data types:

Float To Int

Converts a float into an integer value.

- **Round**: Rounds the input to the closest integer.
- **Trunc**: If the input is negative, this returns ceil(x), otherwise it returns floor(x).
- **Floor**: Returns the largest integer less than or equal to the input
- **Ceil**: Returns the smallest integer greater than or equal to the input.

Float To Matrix

Compose a 4x4 matrix from 16 input components.

Float To RGB

Combine 3 floats into an RGB.

Float To RGBA

Combine 4 floats into an RGBA.

Matrix To Float

Extracts a component at the given column and row.

Int To Float

Converts an integer into a float value.

RGB To Float

Converts an RGB input to a float using the following modes

- **min**: Minimum component.
- **max**: Maximum component.
- **average**: Average of the RGB components.
- **sum**: Sum of the RGB components.
- **luminance**: Perceptual gray scale value as defined by Rec.709.

RGB To RGBA

Combine RGB and alpha inputs into an RGBA.

RGB To Vector

- **raw:** Passes the input through, does nothing.
- **canonical:** Converts the input from [0, 1] range to [-1, 1].

RGBA To Float

Combine an RGBA input to a float using the following modes

- **min**: Minimum component.
- **max**: Maximum component.
- **average**: Average of the RGB components.
- **sum**: Sum of the RGB components.
- **luminance**: Perceptual gray scale value as defined by Rec.709.

Vector To RGB

- **raw:** Passes the input through, does nothing.
- **normalized:** Normalizes the vector before converting the input from [0, 1] range to [-1, 1].
- **canonical:** Converts the input from [0, 1] range to [-1, 1].

Arnold Data Type Conversions

The following table shows how data types are converted within Arnold itself.

Key:

(empty)	conversion is disallowed/ignored
copy	copy (no need to convert)
expand	all components set to the same source value
shallow copy	for pointer types (pointer, node, array, matrix), copy the pointer, not the actual value
boolean cast	zero means false, non-zero means true, and vice-versa
upcast	smaller integer type just gets stuffed into a larger integer type
downcast	larger integer type gets truncated into a smaller integer type
average	the average of the components are taken

						TARGET					
SOURCE	enum	boolean	byte	int	uint	float	RGB	RGBA	vector	point	point2
enum	copy										
boolean		copy	boolean cast	boolean cast	boolean cast						
byte		boolean cast	copy	upcast	upcast						
int		boolean cast	downcast	copy	copy						
uint		boolean cast	downcast	copy	copy						
float						copy	r = flt g = flt b = flt	r = flt g = flt b = flt a = 1	x = flt y = flt z = flt	x = flt y = flt z = flt	x = flt y = flt
RGB						average(rgb)	copy	r = r g = g b = b a = 1	x = r y = g z = b	x = r y = g z = b	x = r y = g
RGBA						average(rgb)	r = r g = g b = b	copy	x = r y = g z = b	x = r y = g z = b	x = r y = g
vector						vector.x	r = x g = y b = z	r = x g = y b = z a = 1	copy	x = x y = y z = z	
point						pnt.x	r = x g = y b = z	r = x g = y b = z a = 1	x = x y = y z = z	copy	
point2						pnt2.x	r = x g = y b = 0	r = x g = y b = 0 a = 1	x = x y = y z = 0	x = x y = y z = 0	copy
string											
node											
pointer											
array											
matrix											

				TARGET	
SOURCE	**string**	**node**	**pointer**	**array**	**matrix**
enum					
boolean					
byte					
int					
uint					
float					
RGB					
RGBA					
vector					
point					
point2					
string	copy				
node			shallow copy		
pointer			shallow copy		
array				shallow copy	
matrix					shallow copy

A few notes:

- The node type (AI_TYPE_NODE) is currently resolved at scene creation time, and node-to-node shader connections are not formally allowed since they internally are turned into a connection based on the output type of the shader node. However, whenever one wants to pass a node along, they can pass it as a generic pointer type instead of using other mechanisms such as shader message passing. The vast majority of the time, if you are using pointer passing inside of your shaders, however, you should re-think your design.
- No automatic conversions exist between integer types and floating-point types.
- No automatic conversions exist between enumerations and any other type.