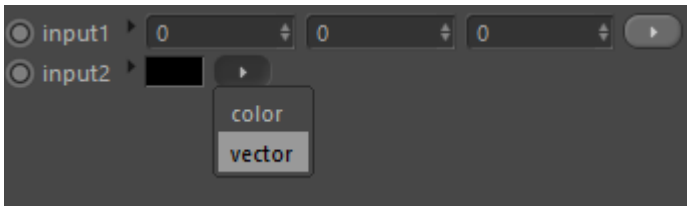
# Math Shaders

A collection of mathematical shaders. Math shaders can work on color or vector inputs.

> Because the C4D color picker is limited to [0;1] range, these shaders have a custom widget where you can select between a color or a vector mode. In color mode, the value is linearized like any other color widget. In vector mode, the selected value is translated as it appears on the UI.




### Abs

Return the absolute value of *input*.

### Add

Return $input_1$ + $input_2$.

### Atan

Return the arctangent of *y*/*x*. The resulting value is in the range [-/2, /2], using the signs of the two arguments to determine the quadrant of the result.

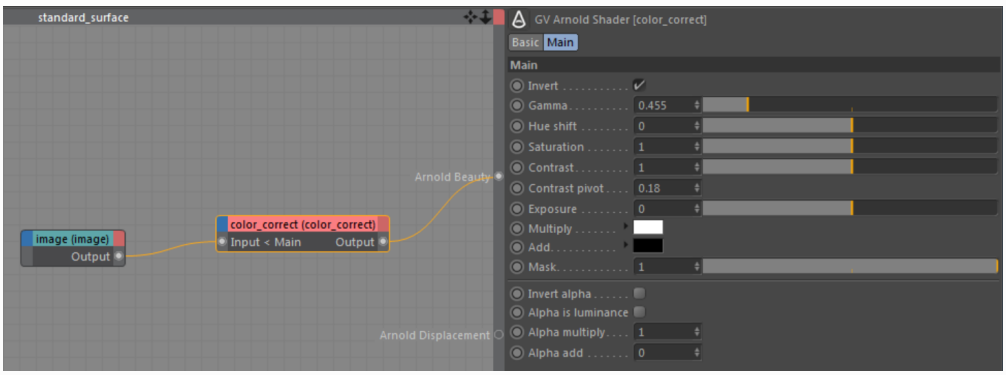
### Compare

Compare $input_1$ and $input_2$ with the following operators and return true or false:

- Equal (==)
- Not Equal (!=)
- Greater Than (>)
- Less Than (<)
- Greater Than or Equal (>=)
- Less Than or Equal (<=)

### Complement

Return one's complement (1  *input*). Also known as reverse video.

> Use a color correct shader instead of complement when inverting an sRGB image.
>
> 
> 
>
> Set the **Color space** of the image to **linear** to prevent Arnold linearizing the texture. Enable **Invert** on the color correct shader and set a 1/2.2 gamma to linearize the inverted color values. Note, that the linearization is not needed when we want to use the raw value (e.g. connected to a specular roughness, etc.).

> An example of the Complement shader in use can be found here.

### Cross

Compute the cross product between two vectors, defined as the vector perpendicular to both input vectors, with its direction defined by the right-hand rule.

$$\mathbf{a}\times\mathbf{b}=\left(a_{y}b_{z}-a_{z}b_{y}\,,\quad a_{z}b_{x}-a_{x}b_{z}\,,\quad a_{x}b_{y}-a_{y}b_{x}\right)$$

The length of the cross product can be interpreted geometrically as:

$$\left\Vert \mathbf{a}\times\mathbf{b} \right\Vert=\left\Vert \mathbf{a}\right\Vert \left\Vert \mathbf{b}\right\Vert \sin\theta$$

### Divide

Return $input_1 \div input_2$.

### Dot

Compute the dot product between two vectors as follows:

ab=a x b x +a y b y +a z b z

The result is a scalar value that can be interpreted geometrically as:

ab=abcos

where the length of vector **a** is denoted by

a

and the angle between **a** and **b** is .

### Exp

Return the exponential of input, $e^{input}$. This is the inverse of Ln, see also Pow.

### Fraction

Returns the fractional part of *input.* For example, an input of 123.456 would return 0.456.

### Is Finite

Return false if *input* is either infinity or NaN, and true otherwise.

### Length

Return the length of the *input* vector, with three possible distance definitions:

Euclidian

The "ordinary" length of the vector: $\sqrt{x^2+y^2+z^2}$

Quadrance

Euclidian distance squared, which is cheaper to compute: $x^2+y^2+z^2$

Manhattan

Measures distance following only axis-aligned directions, which is even cheaper to compute: $\left|x\right|+\left|y\right|+\left|z\right|$

### Log

Return the logarithm of *input* to base. The argument must be greater than zero. This is the inverse of Pow.

### Max

Return the per-component maximum of $input_1$ and $input_2$.

### Min

Return the per-component minimum of $input_1$ and $input_2$.

### Modulo

Return *input* modulo *divisor*. This is the remainder of the division of *input* by *divisor*.

### Multiply

Return $input_1 \times input_2$.

### Negate

Return *input*.

### Normalize

Return a normalized input vector, ie. a unit vector pointing in the same direction.

### Pow

Return $base^{exponent}$. This is the inverse of Log, see also Exp.

### Random

The Random shader outputs a random color from various types of inputs. It is useful to do variations of colors or shader properties for example.

### Reciprocal

Return the multiplicative inverse of input, ie. 1/*input* or $input^{1}$.

### Sign

- Return -1 if *input* < 0
- Return 0 if *input* == 0
- Return 1 if *input* > 0

### Sqrt

Return the square root of *input, ie.*

\( \sqrt{input} \)

### Subtract

Return $input_1 \ input_2$.

### Trigo

Perform various trigonometric functions on *input*. The *frequency* and *phase* parameters make the most sense for the sine, cosine and tangent functions, but are available on all functions for orthogonality. The *units* parameter lets you choose between radians and degrees for the argument of sine, cosine, and tangent and for the result of the inverse functions. It has no effect on the hyperbolic functions.

| Function | Formula | Units Affects | Output Range |
|---|---|---|---|
| Cosine | **cos**(input × frequency + phase) | argument | [-1, 1] |
| Sine | **sin**(*input* × *frequency* + *phase*) | | [-1, 1] |
| Tangent | **tan**(*input* × *frequency* + *phase*) | | [-, ] |
| Arccosine | **arccos**(*input* × *frequency* + *phase*) | result | [0, ] or [0°, 180°] |
| Arcsine | **arcsin**(*input* × *frequency* + *phase*) | | [-/2, /2] or [-90°, 90°] |
| Arctangent | **arctan**(*input* × *frequency* + *phase*) | | [-/2, /2] or [-90°, 90°] |
| Hyperbolic Cosine | **cosh**(*input* × *frequency* + *phase*) | (nothing) | [1, ] |
| Hyperbolic Sine | **sinh**(*input* × *frequency* + *phase*) | | [-, ] |
| Hyperbolic Tangent | **tanh**(*input* × *frequency* + *phase*) | | [-1, 1] |

### Value

The Value shader outputs a user defined constant value based on the selected data type (e.g. float, color, etc.).