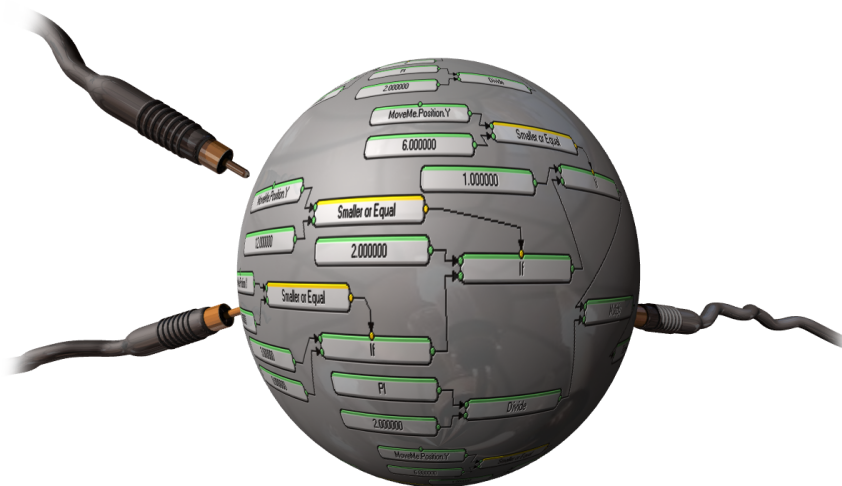


# Amelie Version 1.4

## Visual Expression Language for Lightwave



**Authors:**

Alain Bertrand  
Stephen Moody

**Mac Porting and additional ideas:**

Richard Brak  
<http://www.richardbrak.net/>

**For his time and additional ideas:**

Ben Vost (from Newtek Europe)

# TABLE OF CONTENTS

<b>TABLE OF CONTENTS</b>	<b>2</b>
<b>INTRODUCTION</b>	<b>8</b>
<b>What is Amelie?</b>	<b>8</b>
<b>What is a visual language?</b>	<b>8</b>
<b>Why a visual language?</b>	<b>8</b>
<b>What can Amelie do?</b>	<b>8</b>
<b>Elements</b>	<b>9</b>
<b>User Elements</b>	<b>9</b>
<b>INSTALL</b>	<b>10</b>
<b>Installing the plugin.</b>	<b>10</b>
<b>Add Amelie to the Layout interface</b>	<b>10</b>
<b>Where Amelie stores its files.</b>	<b>11</b>
<b>USING AMELIE</b>	<b>12</b>
<b>First steps.</b>	<b>12</b>
Using the LW menu	12
Apply Amelie directly	13
<b>The User Interface</b>	<b>15</b>
<b>Choosing and adding elements</b>	<b>15</b>
<b>Connecting elements</b>	<b>15</b>
<b>Selecting elements and moving them</b>	<b>16</b>
<b>Context sensitive menu</b>	<b>17</b>
<b>Creating User Elements</b>	<b>18</b>
<b>Edit User Elements</b>	<b>19</b>
<b>Using the online help</b>	<b>19</b>
<b>Using the thumbview</b>	<b>20</b>
<b>Using the expressions list</b>	<b>20</b>
<b>Using the debugger</b>	<b>21</b>
<b>Working with vectors</b>	<b>21</b>
<b>AVAILABLE ELEMENTS</b>	<b>22</b>
<b>Generic</b>	<b>22</b>
Title	22
Comment	22
<b>Operations</b>	<b>23</b>
Add	23
Subtract	23
Multiply	23
Divide	24
Modulo	24
<b>Logic</b>	<b>25</b>
If	25

And	25
Or	25
Not	25
Greater	25
Smaller	26
Greater or Equal	26
Smaller or Equal	26
Equal	26
Not Equal	26
<b>Trigonometry</b>	<b>27</b>
Sin	27
Cos	27
Tan	27
ASin	28
ACos	28
ATan	28
<b>Math</b>	<b>29</b>
Floor	29
Ceil	29
Sqrt	29
Exp	30
Log	30
Log10	30
Pow	30
Min	31
Max	31
Clamp	31
Abs	31
Random	32
<b>Constants</b>	<b>33</b>
Numeric value	33
PI	33
E	33
<b>Motion specific</b>	<b>34</b>
Time	34
This	34
This.Position.X	34
This.Position.Y	34
This.Position.Z	34
This.Rotation.H	35
This.Rotation.P	35
This.Rotation.B	35
This.Size.X	35
This.Size.Y	36
This.Size.Z	36
This.Speed	36
This.Position.Vector	36
This.Rotation.Vector	36
This.Size.Vector	37
<b>Channel specific</b>	<b>38</b>
Time	38

This.Channel	38
<b>Texture specific</b>	<b>39</b>
Time	39
Texture.value	39
Texture.R	39
Texture.G	39
Texture.B	39
Texture.A	39
Sample.World.X	40
Sample.World.Y	40
Sample.World.Z	40
Sample.Object.X	40
Sample.Object.Y	40
Sample.Object.Z	40
Texture.Size.X	40
Texture.Size.Y	41
Texture.Size.Z	41
Texture.Amp	41
Texture.SpotSize	41
Texture.Color.Vector	41
<b>Image filter specific</b>	<b>42</b>
Frame	42
Image.X	42
Image.Y	42
Pixel.R	42
Pixel.G	42
Pixel.B	42
Pixel.A	43
Pixel.Geometry	43
Pixel.Depth	43
Pixel.Special	43
Pixel.Color.Vector	43
Sample.World.Vector	44
Sample.Object.Vector	44
Texture.Size.Vector	44
<b>Pixel filter specific</b>	<b>45</b>
Pixel.X	45
Pixel.Y	45
Pixel.R	45
Pixel.G	45
Pixel.B	45
Pixel.Color.Vector	45
Pixel.A	46
Pixel.Geometry	46
Pixel.Depth	46
Pixel.Special	46
Pixel.RawCol.R	46
Pixel.RawCol.G	46
Pixel.RawCol.B	47
Pixel.RawCol.Vector	47
Pixel.Shading	47

Pixel.Luminous	47
Pixel.Diffuse	47
Pixel.Specular	47
Pixel.Mirror	48
Pixel.Transparent	48
Pixel.Shade.Diff	48
Pixel.Shade.Spec	48
Pixel.Shadow	48
<b>User elements</b>	<b>49</b>
Result	49
Parameter 0	49
Parameter 1	49
Parameter 2	49
Parameter 3	49
Parameter 4	49
<b>Macros</b>	<b>50</b>
Distance	50
Speed	50
Distance traveled	50
Rot/Distance	50
Clone number	50
Radiant to Degree	51
Degree to Radiant	51
Time to Frame	51
Frame to Time	51
Target.H	51
Target.P	52
Target.B	52
Path Delay	52
<b>Item information</b>	<b>53</b>
Item.Position.X	53
Item.Position.Y	53
Item.Position.Z	53
Item.Rotation.H	53
Item.Rotation.P	53
Item.Rotation.B	53
Item.Size.X	54
Item.Size.Y	54
Item.Size.Z	54
Item.Parent	54
Item.Target	54
Item.Goal	54
Item.Type	55
Item.LookAhead	55
Item.GoalStrength	55
Item.Position.Vector	55
Item.Rotation.Vector	55
Item.Size.Vector	56
<b>Scene information</b>	<b>57</b>
Frames/Second	57
Frame Start	57

Frame End	57
Frame Step	57
Frame Width	57
Frame Height	57
Pixel Aspect	58
<b>Objects information</b>	<b>59</b>
Object.NumPoints	59
Object.NumPolygons	59
Object.Dissolve	59
Object.Patch.Interface	59
Object.Patch.Render	59
Object.MBall.Interface	59
Object.MBall.Render	60
Object.BoneSource	60
Object.MorphTarget	60
Object.Fog	60
<b>Cameras information</b>	<b>61</b>
Camera.ZoomFactor	61
Camera.FocalLength	61
Camera.FocalDistance	61
Camera.FStop	61
Camera.BlurLength	61
<b>Bones information</b>	<b>62</b>
Bone.RestLength	62
Bone.Strength	62
<b>Lights information</b>	<b>63</b>
Ambient.R	63
Ambient.G	63
Ambient.B	63
Light.Type	63
Light.Color.R	64
Light.Color.G	64
Light.Color.B	64
Light.ShadowType	64
Light.ConeAngle	64
Light.ConeEdge	65
Light.Range	65
Light.FalloffType	65
Light.Intensity	65
<b>Events</b>	<b>66</b>
Last Event Time	66
Event.Time	66
<b>Transformations</b>	<b>67</b>
World->Local Position.X	67
World->Local Position.Y	67
World->Local Position.Z	67
World->Local Rotation.H	67
World->Local Rotation.P	68
World->Local Rotation.B	68
World->Local Size.X	68
World->Local Size.Y	68

World->Local Size.Z	69
World.Position.X	69
World.Position.Y	69
World.Position.Z	69
World.Rotation.H	69
World.Rotation.P	69
World.Rotation.B	70
World.Size.X	70
World.Size.Y	70
World.Size.Z	70
<b>Vectors</b>	<b>71</b>
Vector	71
Vector.Val[0]	71
Vector.Val[1]	71
Vector.Val[2]	71
Vector.Length	71
 <b>INDEX</b>	 <b>72</b>

# INTRODUCTION

## What is Amelie?

Amelie is a visual expressions language for LightWave. It allows you to create complex expressions by using a simple visual editor. No need to remember complex syntax or strange formulae any more - everything is done using the mouse. Amelie even provides easy-to-use elements to perform complex operations like calculating the total distance traveled for an object, or simpler like its current speed. If you need more macros you may even define your own, so you can reuse parts and improve readability.

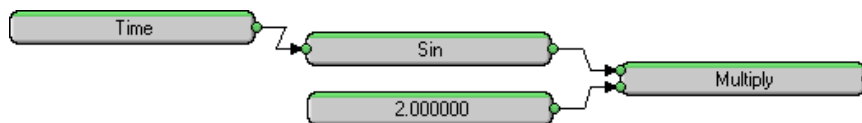
Amelie works as a motion modifier, a channel modifier (graph editor), an image filter, a pixel filter and as a procedural texture. The last option is useful for making deformations or changing the color of an object depending on its speed, for example.

## What is a visual language?

Instead of using text to program you use graphic elements to represent what the computer needs to do. For example instead of writing an expression like this:

`“sin(Time) * 2”`

You will add and connect objects and produce something like that:



In our case, all elements are rounded boxes with a color header (this green line) which are connected using arrows. The type and number of elements plus the connections between them will determine the final behavior of the program.

## Why a visual language?

A visual language is helpful because anybody can start to create some automated behaviors or “expressions” in no time. No needs to remember thousands of words and to learn special syntaxes. It’s also easier for people that do not work 100% of their time with a given language as all the keywords may be retrieved and there is no need to start digging inside a large manual. It’s also a lot easier to look at old expressions and analyze how a given thing works. When we choose to create Amelie we decided to use a very simple user interface, without large blocks that will cover the window with only one block. All operations should be as fast as possible and all the time hints are given to the user to explain the meaning of some elements or link area.

## What can Amelie do?

Amelie is mainly used to automates actions, like rotate the wheel of a car, but can also do more complex things like give a sepia look (you know those yellow-brown old photos) to your images.

In the current state Amelie can be used as a motion modifier in order to move, rotate or resize an item, a channel modifier (modify any value which can be animated), a procedural texture

(on the surface of an object, or on any other parts which work with textures), as an image filter (to change the final aspect of a rendered image), and as a pixel filter. All these ways to use Amelie will work in exactly the same way. The language does not change from one type to another, only some special elements may be available only on specific cases. For example, you can modify the color of a pixel only within an image filter and a procedural texture.

In order to have an overview (with downloadable examples), please visit our demo page: <http://www.rayserver.com/amelie/demo>

## Elements

Elements are the main building blocks of any Amelie expressions.



As you can see, elements are rounded box with a color header, color bullets (link areas) and a title. The color used on the header defined what type of result is returned by the element. Colors used on bullets inform you about which type of element needs to be linked here.

Green is used for numeric values (scalars).

Yellow for conditions.

Blue for LW Items/Objects.

Red is a wildcard type. This means, you could in principle link any type.

Pink is used for vectors (group of 3 values).

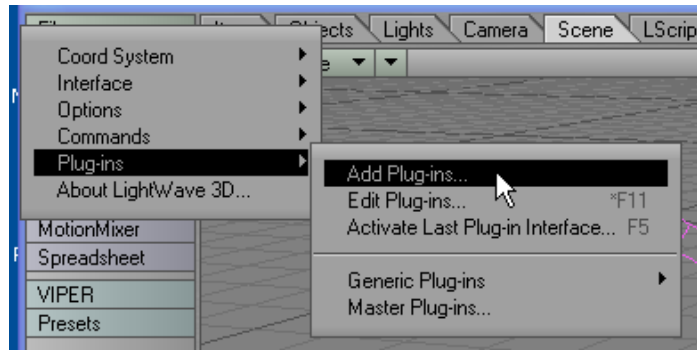
## User Elements

User elements are elements created using Amelie and afterward can be used as any other elements. Those are macros or functions that can be reused anywhere in any other expressions. You can also use user elements inside other user elements.

# INSTALL

## Installing the plugin.

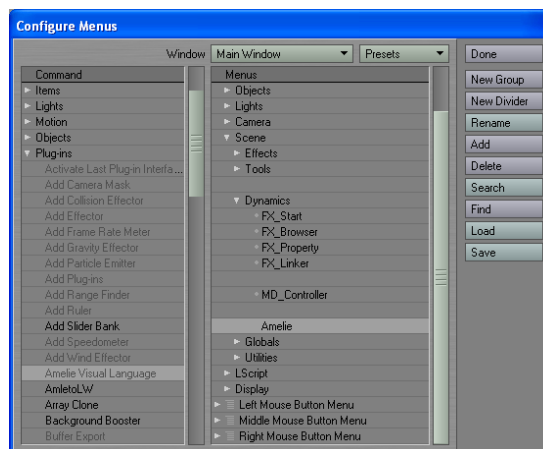
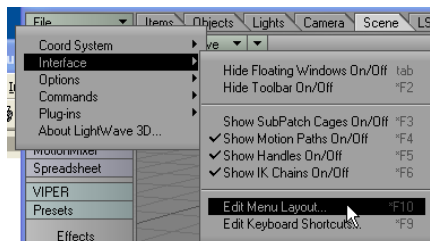
Amelie is a standard Lightwave Layout plugin. To install it you need to copy the .p file inside the Utility directory of your plugin directory. Then you need to start Layout and add Amelie in the known list of plug-ins using the “Layout / Plug-ins... / Add plug-ins...” menu:



A file requester will pop-up and ask you to choose a plug-in. Select the amelie.p file. Once Lightwave has installed the plugin, a pop-up windows should tell you that some plug-ins have been added.

## Add Amelie to the Layout interface

Amelie can be added anywhere in the Layout interface as any other tools. This can be useful to improve the workflow, but it's not mandatory. If you want to add Amelie in your Layout menus, go to the “Layout / Interface / Edit Layout...” menu:



You will find Amelie in the left list under “Plug-ins”. Refer to the LW manual in case you need more help about the interface customization.

## **Where Amelie stores its files.**

Amelie uses two files to save information. These configuration files are stores in the standard config directory.

**amelie.cfg** contains the state of all Amelie windows.

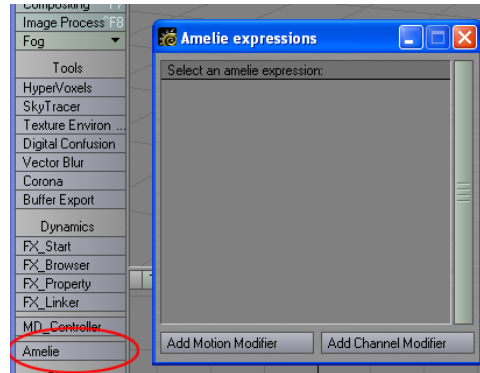
**amelielib.cfg** contains the full library of all available user elements. This file is needed when rendering so if you render over the network be sure all nodes can access this file as well.

# USING AMELIE

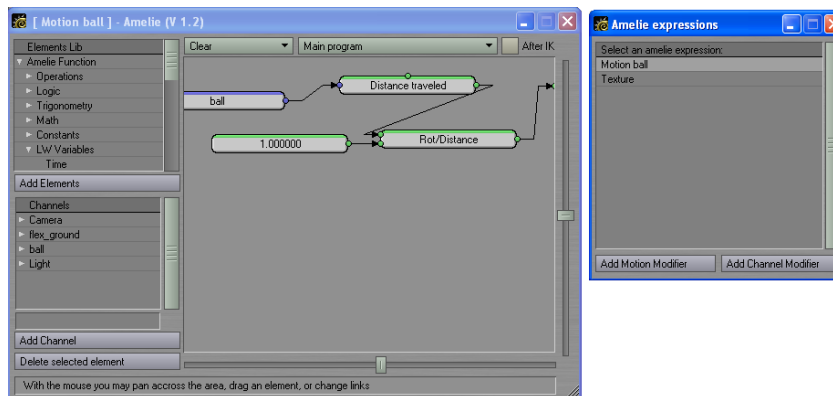
## First steps.

### Using the LW menu

If you choose to add a LW Layout menu for Amelie, you can start Amelie from there:



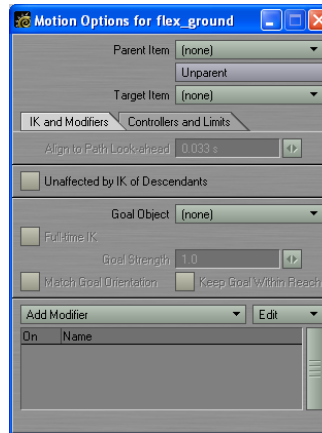
You will then be presented with two buttons “Add Motion Modifier” or “Add Channel Modifier”. These are used to apply Amelie to an object or channel. If you load a scene which already contains some Amelie expressions, this window will contain a list of all of these Amelie expressions and you may jump directly to them by clicking any of the items.



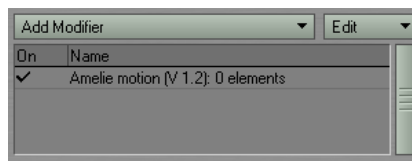
You may apply Amelie as a motion modifier and a channel modifier from this list window, but you cannot apply Amelie as either a procedural texture nor as an Image filter, although you can still browse those types from the list.

## Apply Amelie directly

You may also apply Amelie directly from normal LW panels, for example in the motion options (Items / Motion options) in the first tab (IK and Modifiers) you have a list where you can apply plug-ins:



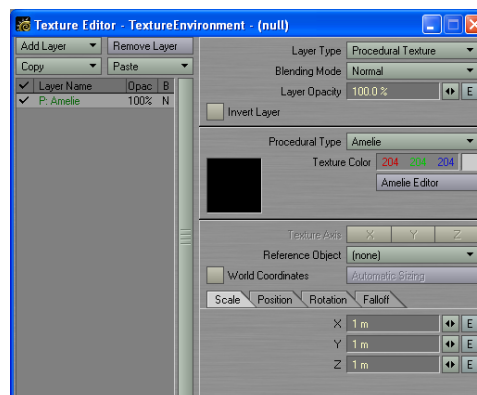
Select Amelie from the “Add Modifier” list and a new instance of Amelie will be added to control the motion of the selected LW item.



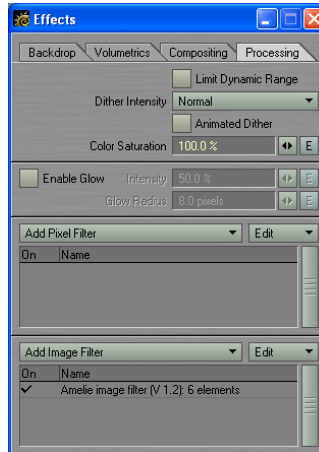
You can at this point double click on the Amelie motion modifier and the Amelie editor will be displayed.

To add Amelie as a channel modifier, press any “E” buttons, the graph editor will popup, select the “Modifier” tab and add Amelie.

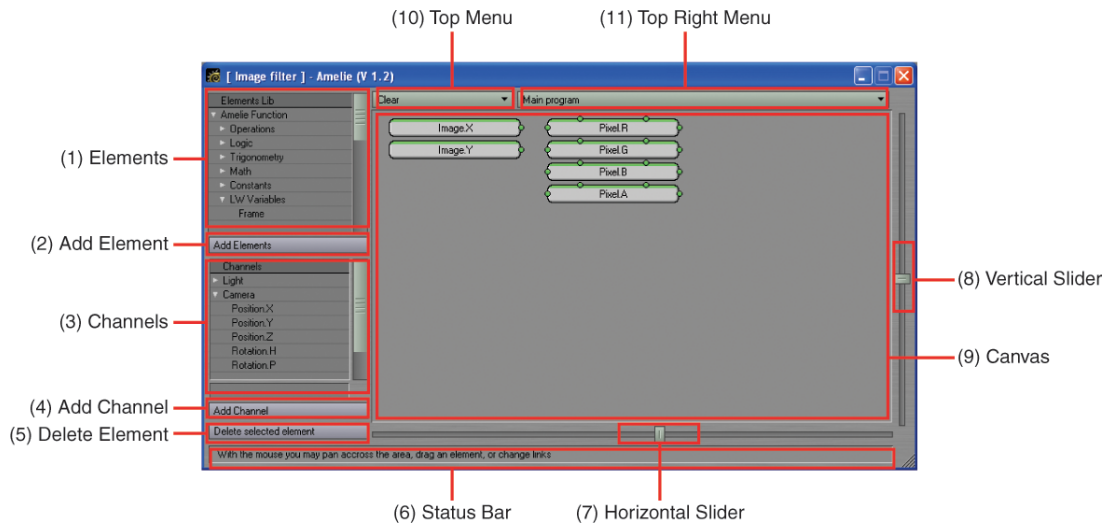
For procedural textures, simply add a procedural layer, and select Amelie as “Procedural type”. You may then press the “Amelie Editor” button to edit the expression.



And finally for the image filter select the “Image Process” panel in the “Scene / Effects” menu. Amelie will be found in the “Image Filter” list.



# The User Interface



This is the main Amelie window: the editor. From this window, all expressions will be created. The items (1) and (3) are the two trees containing all available elements. By double clicking on items in those trees, a new element will be added on the canvas (9). You can also select an item and then press the Add button. If some elements in the canvas are no longer needed, they can be removed by selecting them and clicking the “Delete Element” (5) button. In order to have a larger working area you may scroll the windows either using Alt-LMB drag, Shift-LMB drag or using sliders (7) and (8). At the bottom of the window, we have the status bar (6) which will show information when the mouse is over some items. On the top part, two lists are available. The “Top Menu” (10) contains context sensitive actions, and the “Top Right Menu” (11) allow to add/edit user elements. The options available in the context menus are also available when you right click with the mouse. When dealing with motion modifier an additional check box will be available on the top right part of the window allowing choosing if this expression must be applied before or after LW own IK.

## Choosing and adding elements

50% of the job of writing an Amelie expression is choosing the right elements. The elements have been split into two trees, one containing all the Amelie specific functions (1), and one, which contain all LW channels (3).

In the Amelie elements tree, we have grouped all the elements by category in order to make it easier to find what you are looking for. Depending on where Amelie is applied (motion, channel, etc...) some elements will not be available and others will appear. This means you can always use all displayed elements and do not have to worry about using the wrong type of element in your expression. To add any elements just double click on the tree item and a new element will be added. If you do it more than once, Amelie will place them in a different place on the canvas so they are not overlapping.

## Connecting elements

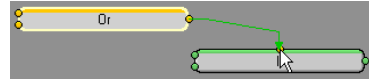
The remaining 50% of the job is connecting these elements to form our expression.



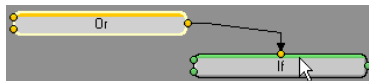
To link two elements, we need to click on a link area (those colored bullets) and drag the mouse.



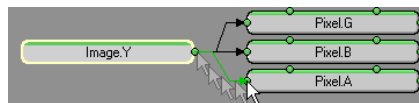
While the mouse is dragged the status bar will display the meaning of this particular link area. When the link is dragged over an area it can be connected to, the link will turn green.



It's possible to start from a link area on the right side and drop the link on a top or left side, or start from a left/top side and drop the link to a right connector. One of the two sides (and only one) must be on a right side link area. Link areas on the right side are *outputs* and link areas on the left and top are *inputs*. Once the link is dropped it will change color again and become black. An arrow shows the direction in which data is passed.



If you drag a link starting from the right side of an element that is already linked to another element, you will just add another link. This allows you to use the output of one element as input to as many elements as you want.

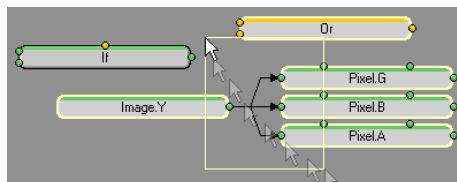


If you try to drag a left or topside link area, which is already connected, the existing link will be removed and a new one will be created. If you drop a link when the arrow is yellow, then simply the link is removed.

To remove a link, click on the top or left connect area and release the mouse.

## Selecting elements and moving them

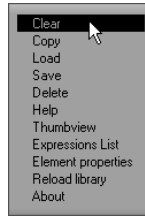
To select a single element, just LMB click an element. Once an element is selected, a yellow border will be drawn around it. To move an element, just LMB drag it over the canvas. In order to select multiple elements, hold down the shift key and LMB click unselected elements, this will add them in the selection. You may also drag the mouse around a group of elements and a yellow rectangle will select all elements included or touching this box.



Once elements are selected, they can be moved if the drag operation is started from a selected element.

## Context sensitive menu

Depending on the conditions such as if an element is selected or not, the canvas is empty or not, and other states as well, the top menu (10) contains different menu item.

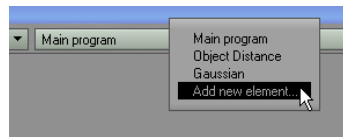


Clear	Will erase the current expression.
Copy	Copy all selected items to the clipboard.
Paste	Past the items in the clipboard to the canvas.
Load	Load external files (.aml).
Save	Save the current expression to an external file (.aml).
Modify	Modify a numeric constant, a comment, a title or rename a user element parameter.
Delete	Delete selected items.
Help	Open the help window. If an element is selected the help on this element will be displayed.
Thumbview	Open a window containing a thumbnail view of the current expression.
Expressions List	Open a window containing all Amelie expressions contained in the current scene.
Evaluate	Execute this expression. It's the same as moving the time frame.
Element properties	Change properties about the current user element.
Show/Hide debug	Display or hide debugs information.
Reload library	Reload the library of user elements (in case you copied a new amelielib.cfg for example).
About	Show the about window.

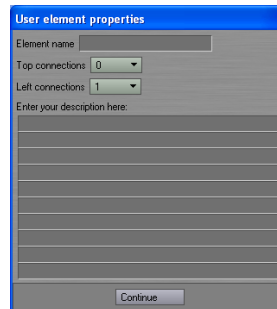
This menu can be opened also using the RMB anywhere on the canvas.

## Creating User Elements

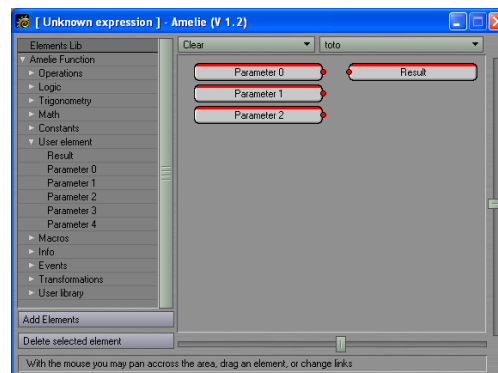
Using the top right menu, it's possible either to edit existing user elements (loaded in the library) or create new ones.



To add a new user element just select the last item called “Add new element...” this will popup a new panel.



The “Element name” is the name displayed on the element and in the element tree. Top connections and left connections are the number of connections or parameters of this user element. The remaining part is an optional but a helpful description of the user element. Once the “Continue” button is pressed, a new Amelie expression will be displayed containing the input and output elements.



As you can see, the channel tree is not available inside user elements. This is because these channels will change depending on the scene loaded, and will not be available in all scenes.

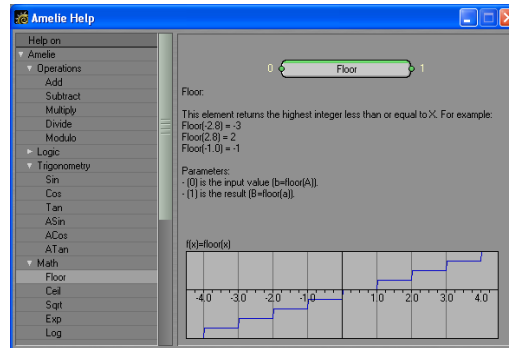
Parameter 0 to Parameter 4 are values passed to this element (first top connections and then left connections). Result must be the result of the user element. Only one value can be returned. Once you have finished the user element expression, just select “Main program” in the top right menu (11). At this point, the new user element is saved in the library and it will appear in the element tree (1) inside the branch “User library”.

## Edit User Elements

Any user element in the library can be edited by choosing the name of the user element in the top right menu (11). Be careful about changing the number of link area, as it will not be automatically updated in expressions, which use this specific user element. If you change the name of the user element, expressions which use this specific user element may no longer work correctly. To fix them, you need to delete the old version of this user element and add it again.

## Using the online help

Amelie contains a complete online help describing each element which will show you graphs for some of the functions so you can visualize how the function will affect the motion.

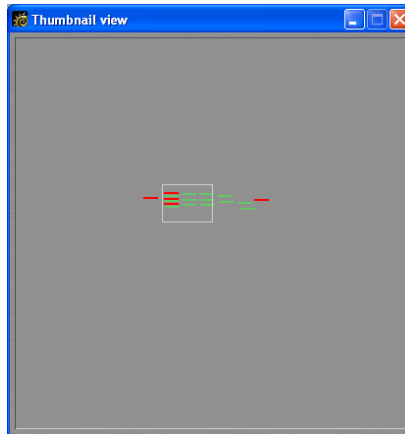


On the left side, a tree will enable to browse all possible elements without the need to first add them inside the expression. On the right side, there is a graphical representation of the element with all link area, some description and an explanation of what the link areas are for. In this case, we can see the graph of this element on the bottom right part.

If the help window is left open, as soon as an element is selected in the expression, the help will automatically display the information for the selected element. This can help you when you are trying to follow and expression which contains elements you don't know.

## Using the thumbview

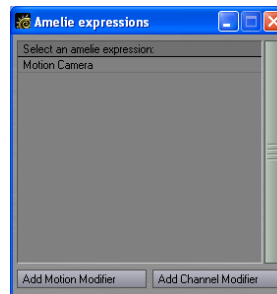
When an expression is large, it can be difficult to move around and find a specific part. In order to simplify this process, Amelie offer also a thumbnail view of the current expression.



All elements are represented in the view as small colored rectangles where the color is the same as the element color. Using the LMB, it's possible to pan across the whole expression. When an expression is updated, the thumbview is updated in real-time.

## Using the expressions list

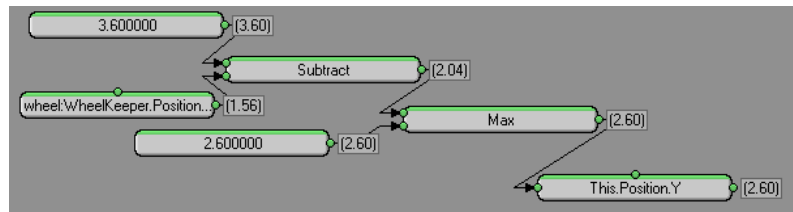
When a scene contain many Amelie expressions, it can be difficult to find them or to jump back and forth between them. The expression list window will show a list of all the Amelie expressions that are used in the scene.



Simply by selecting one of the items in the list, an Amelie editor will be opened containing the selected expression. It's even possible to apply Amelie to new objects or channels from within the expressions list.

## Using the debugger

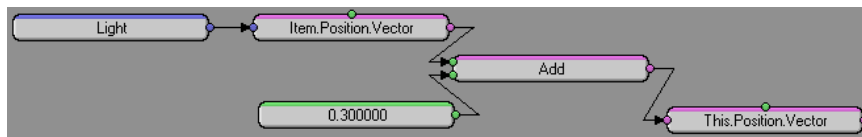
At any time, you may enable the real-time debugger in order to see exactly how an expression is evaluated.



As soon as the debugger is enabled, values of elements are displayed in a light gray box on the right side of the element. It's even possible to see how some expression change during the animation as if the animation is played or the time slider is moved, the expression and the debug information is updated in real-time.

## Working with vectors

A vector is a group of three values (a value is a scalar). Amelie is able to work with vectors as any other scalar numbers. This mean you may for example mix a scalar and a vector and get the result of the operation.



In this case each values of the vector is added with the constant 0.3, so if we start with a vector  $\{0,1,2\}$  and add a scalar  $\{0.3\}$  we will have  $\{0.3,1.3,2.3\}$ . We can also work with two vectors in the same way. As soon as an expression is evaluated the green color of the “Add” element will be changed with a pink, that's mean this element is now working with a vector instead of a scalar.

All operations, trigonometry and math elements can be used with vectors, only the “Random” element cannot use them. Conditions works also with vectors but use the length of the vector instead of checking each values so a negative and a positive vector can be seen as equal. In any case, if you pass a vector to some element which is not able to work with vectors, the first value of the vector will be used as the parameter. It's also possible to pass vectors to user elements and/or return them.

If vectors are available for a given thing (for example read the position of an object), they should be used instead of the scalar version of the element, as vectors are faster than to deal with than using N more elements. This enable also to make the code more compact and readable.

## **AVAILABLE ELEMENTS**

In this chapter you will find the full list of available Amelie elements. Some additional user elements can be found on our online library. This information is also available in the help window.

### **Generic**

---

---

**Title**

---

This element enable to give a title to the current expression. Only one title per expression is allowed. The title will be visible on LW modifier list and in the Amelie expression list as well.

---

---

**Comment**

---

This element enable to write some comments inside the expression.

# Operations

---

---

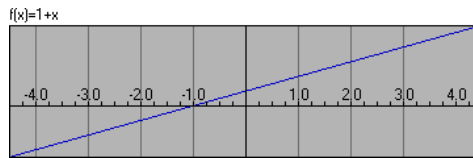
## Add

---

This will Add two values using the formula  $(2)=(0)+(1)$ .

Parameters:

- (0) is the first value to add ( $c=A+b$ ).
- (1) is the second value to add ( $c=a+B$ ).
- (2) is the result of the two added values ( $C=a+b$ ).



---

---

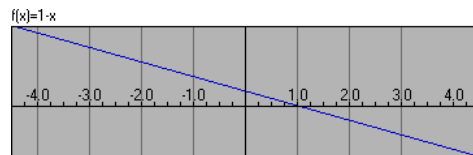
## Subtract

---

This will Subtract one value from another using the formula  $(2)=(0)-(1)$ .

Parameters:

- (0) is the value you are subtracting from ( $c=A-b$ ).
- (1) is the value that you want to subtract ( $c=a-B$ ).
- (2) is the result value ( $C=a+b$ ).



---

---

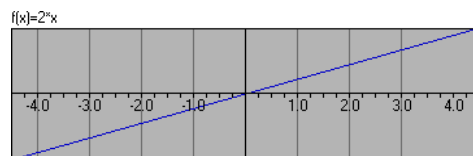
## Multiply

---

This will Multiply two values with the formula  $(2)=(0)*(1)$ .

Parameters:

- (0) is the first value to multiply ( $c=A*b$ ).
- (1) is the second value to multiply ( $c=a*B$ ).
- (2) is the result of the two multiplied values ( $C=a*b$ ).



---

---

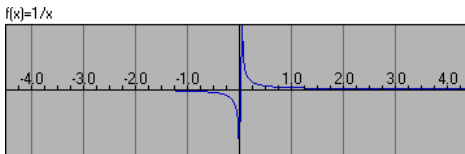
## Divide

---

This will Divide two values with the formula  $(2)=(0)/(1)$ .

Parameters:

- (0) is the value to be divided ( $c=A/b$ ).
- (1) is the value to divide by ( $c=a/B$ ).
- (2) is the result of the division of the two values ( $C=a/b$ ).



---

---

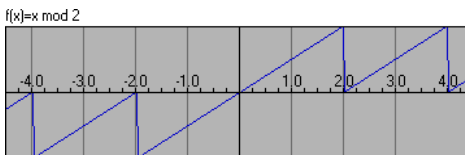
## Modulo

---

The Modulo calculates the remaining part of a division (that is the value that cannot be divided without giving a floating point. e.g.  $11 / 3 = 3$  with 2 remaining.  $11 \bmod 3 = 2$ ).

Parameters:

- (0) is the value to be divided ( $c=A \bmod b$ ).
- (1) is the value to divide by ( $c=a \bmod B$ ).
- (2) is the result ( $C=a \bmod b$ ).



# Logic

---

---

## If

If will evaluate the input condition and if the condition is true then the first value will be used as the result, if it is false then the second value will be used as the result.

Parameters:

- (0) is the condition used.
- (1) is the value returned if the condition is true.
- (2) is the value returned if the condition is false.
- (3) is the returned value.

---

---

## And

And is used to link two conditions. If the first is true AND the second is also true, then the result is true. If any of the two conditions or both conditions are false then the result is false.

Parameters:

- (0) is the first condition.
- (1) is the second condition.
- (2) is the resulting condition.

---

---

## Or

Or is used to link two conditions. If either the first or second conditions are true the result is true. If both conditions are false then the result is false.

Parameters:

- (0) is the first condition.
- (1) is the second condition.
- (2) is the resulting condition.

---

---

## Not

Not is used to invert the condition. If the input condition is true, then the result will be false. If the input condition is false then the result will be true.

Parameters:

- (0) is the input condition.
- (1) is the resulting condition.

---

---

## Greater

Greater returns true if the first value is greater than the second.

Parameters:

- (0) is the first value ( $A > b$ ).
- (1) is the second value ( $a > B$ ).
- (2) is the resulting condition.

---

---

### **Smaller**

---

Smaller returns true if the first value is smaller than the second.

Parameters:

- (0) is the first value ( $A < b$ ).
- (1) is the second value ( $a < B$ ).
- (2) is the resulting condition.

---

---

### **Greater or Equal**

---

Greater or Equal returns true if the first value is greater than or equal to the second.

Parameters:

- (0) is the first value ( $A \geq b$ ).
- (1) is the second value ( $a \geq B$ ).
- (2) is the resulting condition.

---

---

### **Smaller or Equal**

---

Smaller or Equal returns true if the first value is smaller than or equal to the second.

Parameters:

- (0) is the first value ( $A \leq b$ ).
- (1) is the second value ( $a \leq B$ ).
- (2) is the resulting condition.

---

---

### **Equal**

---

Equal returns true if the first value is exactly equal (the same) as the second.

Parameters:

- (0) is the first value ( $A = b$ ).
- (1) is the second value ( $a = B$ ).
- (2) is the resulting condition.

---

---

### **Not Equal**

---

Not Equal returns true if the first value is not equal to (not the same as) the second.

Parameters:

- (0) is the first value ( $A \neq b$ ).
- (1) is the second value ( $a \neq B$ ).
- (2) is the resulting condition.

# Trigonometry

---

---

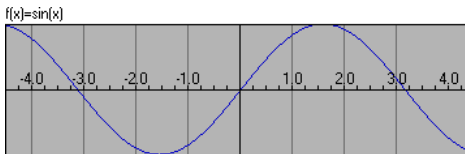
## Sin

---

Returns the sine (in radians) of a number.

Parameters:

- (0) is the input number ( $b=\sin(A)$ ).
- (1) is the result ( $B=\sin(a)$ ).



---

---

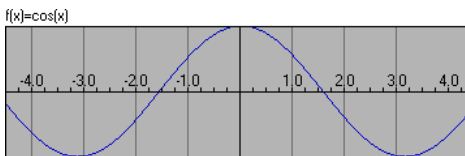
## Cos

---

Returns the cosine (in radians) of a number.

Parameters:

- (0) is the input number ( $b=\cos(A)$ ).
- (1) is the result ( $B=\cos(a)$ ).



---

---

## Tan

---

Returns the tangent (in radians) of a number.

Parameters:

- (0) is the input number ( $b=\tan(A)$ ).
- (1) is the result ( $B=\tan(A)$ ).



---

---

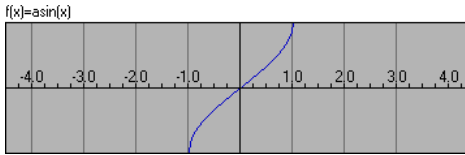
## ASin

---

Returns the arcsine (in radians) of a number.

Parameters:

- (0) is the input number ( $b=\text{asin}(A)$ ).
- (1) is the result ( $B=\text{asin}(a)$ ).



---

---

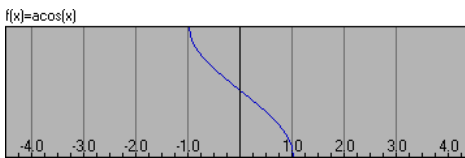
## ACos

---

Returns the arccosine (in radians) of a number.

Parameters:

- (0) is the input number ( $b=\text{acos}(A)$ ).
- (1) is the result ( $B=\text{acos}(a)$ ).



---

---

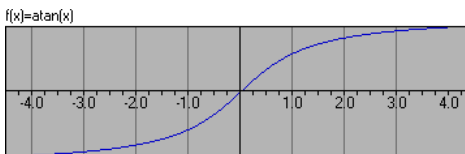
## ATan

---

Returns the arctangent (in radians) of a number.

Parameters:

- (0) is the input number ( $b=\text{atan}(A)$ ).
- (1) is the result ( $B=\text{atan}(a)$ ).



# Math

---

---

## Floor

---

This element returns the highest integer less than or equal to X. For example:

$$\text{Floor}(-2.8) = -3$$

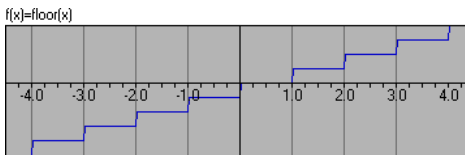
$$\text{Floor}(2.8) = 2$$

$$\text{Floor}(-1.0) = -1$$

Parameters:

- (0) is the input value ( $b=\text{floor}(A)$ ).

- (1) is the result ( $B=\text{floor}(a)$ ).



---

---

## Ceil

---

This element returns the lowest integer greater than or equal to X. For example:

$$\text{Ceil}(-2.8) = -2$$

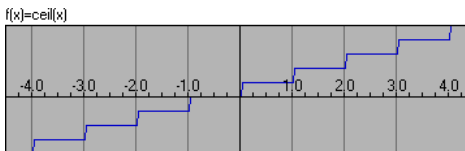
$$\text{Ceil}(2.8) = 3$$

$$\text{Ceil}(-1.0) = -1$$

Parameters:

- (0) is the input value ( $b=\text{ceil}(A)$ ).

- (1) is the result ( $B=\text{ceil}(a)$ ).



---

---

## Sqrt

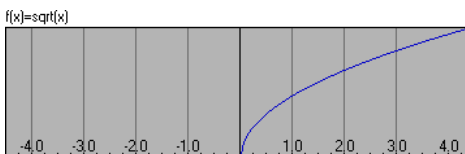
---

Calculates the positive square root.

Parameters:

- (0) is the input value ( $b=\text{sqrt}(A)$ ).

- (1) is the result ( $B=\text{sqrt}(A)$ ).



---

---

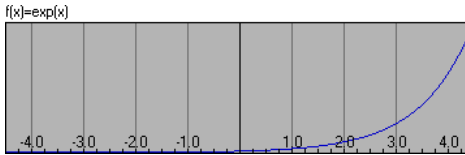
## Exp

---

Calculates the exponential e to the x.

Parameters:

- (0) is the input value ( $b=\exp(A)$ ).
- (1) is the result ( $B=\exp(a)$ ).



---

---

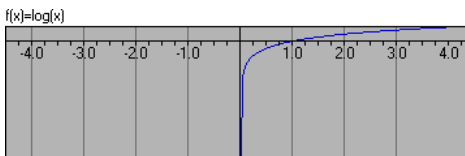
## Log

---

Calculates the natural logarithm of x.

Parameters:

- (0) is the input value ( $b=\log(A)$ ).
- (1) is the result ( $B=\log(a)$ ).



---

---

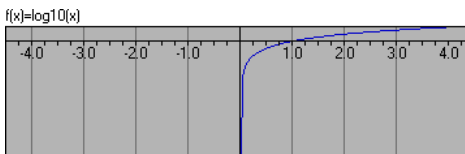
## Log10

---

Calculates the base ten logarithm of x.

Parameters:

- (0) is the input value ( $b=\log_{10}(A)$ ).
- (1) is the result ( $B=\log_{10}(a)$ ).



---

---

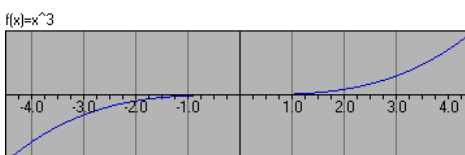
## Pow

---

Calculates x to the power of y.

Parameters:

- (0) is the first input value ( $c=A^b$ ).
- (1) is the second input value ( $c=a^B$ ).
- (1) is the result ( $C=a^b$ ).



---

---

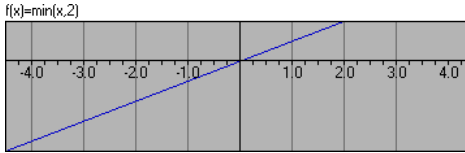
## Min

---

Returns the minimum value of the two input values.

Parameters:

- (0) is the first number ( $c=\min(A,b)$ ).
- (1) is the second number ( $c=\min(a,B)$ ).
- (2) is the result ( $C=\min(a,b)$ ).



---

---

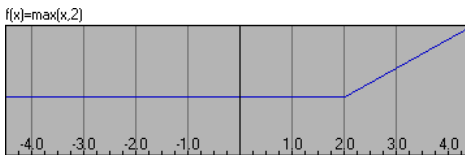
## Max

---

Returns the maximum value of the two input values.

Parameters:

- (0) is the first number ( $c=\max(A,b)$ ).
- (1) is the second number ( $c=\max(a,B)$ ).
- (2) is the result ( $C=\max(a,b)$ ).



---

---

## Clamp

---

Returns a value between min (0) and max (1). If the input (2) is outside this range, the limit will be returned instead of the original value.

Parameters:

- (0) is the lower limit.
- (1) is the upper limit.
- (2) is the value to clamp.
- (3) is the value clamped.

---

---

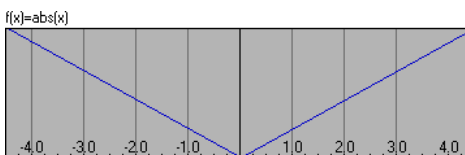
## Abs

---

Returns the absolute positive value of a number.

Parameters:

- (0) is the input value ( $b=\text{abs}(A)$ ).
- (1) is the result ( $B=\text{abs}(a)$ ).



---

---

## **Random**

---

Returns a random between the range specified by the two paramters (0) and (1).

Parameters:

- (0) is the mininum value.
- (1) is the maximum value.
- (2) is the random number.

# Constants

---

---

## Numeric value

---

A user defined numeric value. You may modify it at anytime just by selecting the modify item in the popup menu (right click on the element).

Parameters:

- (0) is the numeric constant.

---

---

## PI

---

The mathematical constant PI (used for calculating circles for example). The value is about 3.14159265.

Parameters:

- (0) is the PI constant.

---

---

## E

---

The mathematical constant E (used for logarithm for example). The value is about 2.71828183.

Parameters:

- (0) is the E constant.

## Motion specific

---

---

### Time

---

The current time (in seconds since the start).

Parameters:

- (0) is the current time.

---

---

### This

---

This is the object to which this expression as been assigned.

Parameters:

- (0) is the current object.

---

---

### This.Position.X

---

Position X of the current object that Amelie is applied to. This element can be used to both read the current value or set a new value

The parameter (0) is an optional value that can be used to specify the time which you access the value

Parameters:

- (0) is the time at which you want to get the value
- (1) is the new value to set
- (2) is the current value.

---

---

### This.Position.Y

---

Position Y of the current object that Amelie is applied to. This element can be used to both read the current value or set a new value

The parameter (0) is an optional value that can be used to specify the time which you access the value

Parameters:

- (0) is the time at which you want to get the value
- (1) is the new value to set
- (2) is the current value.

---

---

### This.Position.Z

---

Position Z of the current object that Amelie is applied to. This element can be used to both read the current value or set a new value.

The parameter (0) is an optional value that can be used to specify the time which you access the value

Parameters:

- (0) is the time at which you want to get the value
- (1) is the new value to set
- (2) is the current value.

---

---

### **This.Rotation.H**

---

Rotation H of the current object that Amelie is applied to. This element can be used to both read the current value or set a new value

The parameter (0) is an optional value that can be used to specify the time which you access the value

Parameters:

- (0) is the time at which you want to get the value
- (1) is the new value to set
- (2) is the current value.

---

---

### **This.Rotation.P**

---

Rotation P of the current object that Amelie is applied to. This element can be used to both read the current value or set a new value

The parameter (0) is an optional value that can be used to specify the time which you access the value

Parameters:

- (0) is the time at which you want to get the value
- (1) is the new value to set
- (2) is the current value.

---

---

### **This.Rotation.B**

---

Rotation B of the current object that Amelie is applied to. This element can be used to both read the current value or set a new value

The parameter (0) is an optional value that can be used to specify the time which you access the value

Parameters:

- (0) is the time at which you want to get the value
- (1) is the new value to set
- (2) is the current value.

---

---

### **This.Size.X**

---

Size X of the current object that Amelie is applied to. This element can be used to both read the current value or set a new value

The parameter (0) is an optional value that can be used to specify the time which you access the value

Parameters:

- (0) is the time at which you want to get the value
- (1) is the new value to set
- (2) is the current value.

---

---

**This.Size.Y**

---

Size Y of the current object that Amelie is applied to. This element can be used to both read the current value or set a new value

The parameter (0) is an optional value that can be used to specify the time which you access the value

Parameters:

- (0) is the time at which you want to get the value
- (1) is the new value to set
- (2) is the current value.

---

---

**This.Size.Z**

---

Size Z of the current object that Amelie is applied to. This element can be used to both read the current value or set a new value

The parameter (0) is an optional value that can be used to specify the time which you access the value

Parameters:

- (0) is the time at which you want to get the value
- (1) is the new value to set
- (2) is the current value.

---

---

**This.Speed**

---

The speed of this object at a given time.

Parameters:

- (0) is the time used to evaluate the speed (if omitted the current time is used).
- (1) is the calculated speed at the given time.

---

---

**This.Position.Vector**

---

Returns a vector containing the position of the current LW item at a given time.

Parameters:

- (0) is the time at which you want to get the value
- (1) is the new value to set
- (2) is the current value.

---

---

**This.Rotation.Vector**

---

Returns a vector containing the rotation of the current LW item at a given time.

Parameters:

- (0) is the time at which you want to get the value
- (1) is the new value to set
- (2) is the current value.

---

---

**This.Size.Vector**

---

Returns a vector containing the size of the current LW item at a given time.

Parameters:

- (0) is the time at which you want to get the value
- (1) is the new value to set
- (2) is the current value.

## Channel specific

---

---

### Time

---

The current time (in seconds since the start).

Parameters:

- (0) is the current time.

---

---

### This.Channel

---

This is the value of the channel to which the expression is assigned.

Parameters:

- (0) is the current channel value.

## Texture specific

---

---

### Time

---

The current time (in seconds since the start).

Parameters:

- (0) is the current time.

---

---

### Texture value

---

This is the return value of a procedural texture. Must be set in order to change the texture. This element will overwrite Texture.R, Texture.G, Texture.B and Texture.A

Parameters:

- (0) is the value to set.

---

---

### Texture.R

---

This is the red component of the texture for the current point. In order to use this element be sure to remove any Texture.Value elements. The value should be between 0.0 and 1.0.

Parameters:

- (0) is the red component of the texture.

---

---

### Texture.G

---

This is the green component of the texture for the current point. In order to use this element be sure to remove any Texture.Value elements. The value should be between 0.0 and 1.0.

Parameters:

- (0) is the green component of the texture.

---

---

### Texture.B

---

This is the blue component of the texture for the current point. In order to use this element be sure to remove any Texture.Value elements. The value should be between 0.0 and 1.0.

Parameters:

- (0) is the blue component of the texture.

---

---

### Texture.A

---

This is the alpha component of the texture for the current point. In order to use this element be sure to remove any Texture.Value elements. The value should be between 0.0 and 1.0.

Parameters:

- (0) is the alpha component of the texture.

---

---

**Sample.World.X**

---

The X position of the sample in world coordinates.

Parameters:

- (0) is the X position of the sample.

---

---

---

---

**Sample.World.Y**

---

The Y position of the sample in world coordinates.

Parameters:

- (0) is the Y position of the sample.

---

---

---

---

**Sample.World.Z**

---

The Z position of the sample in world coordinates.

Parameters:

- (0) is the Z position of the sample.

---

---

---

---

**Sample.Object.X**

---

The X position of the sample in texture coordinates.

Parameters:

- (0) is the X position of the sample.

---

---

---

---

**Sample.Object.Y**

---

The Y position of the sample in texture coordinates.

Parameters:

- (0) is the Y position of the sample.

---

---

---

---

**Sample.Object.Z**

---

The Z position of the sample in texture coordinates.

Parameters:

- (0) is the Z position of the sample.

---

---

---

---

**Texture.Size.X**

---

The X size of the texture. The size value is used to scale the texture spatially. The interpretation is up to the texture, but typically this is the size of a texture cell or the distance between repeating elements.

Parameters:

- (0) is the X size of the texture.

---

---

---

---

**Texture.Size.Y**

---

The Y size of the texture. The size value is used to scale the texture spatially. The interpretation is up to the texture, but typically this is the size of a texture cell or the distance between repeating elements.

Parameters:

- (0) is the Y size of the texture.

---

---

**Texture.Size.Z**

---

The Z size of the texture. The size value is used to scale the texture spatially. The interpretation is up to the texture, but typically this is the size of a texture cell or the distance between repeating elements.

Parameters:

- (0) is the Z size of the texture.

---

---

**Texture.Amp**

---

The amplitude of the texture. This value is typically used to scale the magnitude or strength of the texture.

Parameters:

- (0) is the amplitude of the texture.

---

---

**Texture.SpotSize**

---

The approximate diameter of the sample spot. This is useful when antialiasing the texture.

Parameters:

- (0) is the approximate diameter of the sample spot.

---

---

**Texture.Color.Vector**

---

Allow to set the color vector (RGB) of the actual point (U,V).

Parameters:

- (0) is the color vector.

## Image filter specific

---

---

### Frame

---

Returns the current frame number.

Parameters:

- (0) is the current frame number.

---

---

### Image.X

---

Returns the X position of the current pixel.

Parameters:

- (0) is the current X position.

---

---

### Image.Y

---

Returns the Y position of the current pixel.

Parameters:

- (0) is the current Y position.

---

---

### Pixel.R

---

Is the red value of the pixel.

Parameters:

- (0) is the X position of pixel, if omitted, the current X.
- (1) is the Y position of the pixel, if omitted, the current Y.
- (2) is the new value red value.
- (3) is the current red value.

---

---

### Pixel.G

---

Is the green value of the pixel.

Parameters:

- (0) is the X position of pixel, if omitted, the current X.
- (1) is the Y position of the pixel, if omitted, the current Y.
- (2) is the new value green value.
- (3) is the current green value.

---

---

### Pixel.B

---

Is the blue value of the pixel.

Parameters:

- (0) is the X position of pixel, if omitted, the current X.
- (1) is the Y position of the pixel, if omitted, the current Y.
- (2) is the new value blue value.
- (3) is the current blue value.

---

---

### **Pixel.A**

---

Is the alpha value of the pixel.

Parameters:

- (0) is the X position of pixel, if omitted, the current X.
- (1) is the Y position of the pixel, if omitted, the current Y.
- (2) is the new value alpha value.
- (3) is the current alpha value.

---

---

### **Pixel.Geometry**

---

The values in this buffer are the dot-products of the surface normals with the eye vector (or the cosine of the angle of the surfaces to the eye). They reveal something about the underlying shape of the objects in the image. Where the value is 1.0, the surface is facing directly toward the camera, and where it's 0, the surface is edge-on to the camera.

Parameters:

- (0) is the X position of pixel, if omitted, the current X.
- (1) is the Y position of the pixel, if omitted, the current Y.
- (2) is the geometry value.

---

---

### **Pixel.Depth**

---

The distance from the camera to the nearest object visible in a pixel. Strictly speaking, this is the perpendicular distance from the plane defined by the camera's position and view vector. Also known as the z-buffer.

Parameters:

- (0) is the X position of pixel, if omitted, the current X.
- (1) is the Y position of the pixel, if omitted, the current Y.
- (2) is the depth value.

---

---

### **Pixel.Special**

---

Contains user-assigned values that are unique for each surface.

Parameters:

- (0) is the X position of pixel, if omitted, the current X.
- (1) is the Y position of the pixel, if omitted, the current Y.
- (2) is the special value.

---

---

### **Pixel.Color.Vector**

---

Is the color vector of the pixel.

Parameters:

- (0) is the X position of pixel, if omitted, the current X.
- (1) is the Y position of the pixel, if omitted, the current Y.
- (2) is the new color vector to set.\n- (3) is the current color vector.

---

---

**Sample.World.Vector**

---

The position vector of the sample in world coordinates.

Parameters:

- (0) is the position vector of the sample.

---

---

**Sample.Object.Vector**

---

The position vector of the sample in texture coordinates.

Parameters:

- (0) is the position vector of the sample.

---

---

**Texture.Size.Vector**

---

The size vector of the texture. The size value is used to scale the texture spatially. The interpretation is up to the texture, but typically this is the size of a texture cell or the distance between repeating elements.

Parameters:

- (0) is the size vector of the texture.

## Pixel filter specific

---

---

### Pixel.X

---

Returns the X position of the current pixel.

Parameters:

- (0) is the current X position.

---

---

### Pixel.Y

---

Returns the X position of the current pixel.

Parameters:

- (0) is the current X position.

---

---

### Pixel.R

---

Is the red value of the pixel.

Parameters:

- (0) is the new red value.

- (1) is the current red value.

---

---

### Pixel.G

---

Is the green value of the pixel.

Parameters:

- (0) is the new green value.

- (1) is the current green value.

---

---

### Pixel.B

---

Is the blue value of the pixel.

Parameters:

- (0) is the new blue value.

- (1) is the current blue value. // 385

---

---

### Pixel.Color.Vector

---

Is the color vector of the pixel.

Parameters:

- (0) is the new color vector.

- (1) is the current color vector.

---

---

**Pixel.A**

---

Is the alpha value of the pixel.

Parameters:

- (0) is the new alpha value.
- (1) is the current alpha value.

---

---

**Pixel.Geometry**

---

The values in this buffer are the dot-products of the surface normals with the eye vector (or the cosine of the angle of the surfaces to the eye). They reveal something about the underlying shape of the objects in the image. Where the value is 1.0, the surface is facing directly toward the camera, and where it's 0, the surface is edge-on to the camera.

Parameters:

- (0) is the new geometry value.
- (1) is the current geometry value.

---

---

**Pixel.Depth**

---

The distance from the camera to the nearest object visible in a pixel. Strictly speaking, this is the perpendicular distance from the plane defined by the camera's position and view vector. Also known as the z-buffer.

Parameters:

- (0) is the new depth value.
- (1) is the current depth value.

---

---

**Pixel.Special**

---

Contains user-assigned values that are unique for each surface.

Parameters:

- (0) is the new special value
- (2) is the current special value. // 390

---

---

**Pixel.RawCol.R**

---

Contains the raw, unshaded red values of the surface.

Parameters:

- (0) is the new red value.
- (1) is the current red value.

---

---

**Pixel.RawCol.G**

---

Contains the raw, unshaded green values of the surface.

Parameters:

- (0) is the new green value.
- (1) is the current green value.

---

---

**Pixel.RawCol.B**

---

Contains the raw, unshaded blue values of the surface.

Parameters:

- (0) is the new blue value.
- (1) is the current blue value.

---

---

**Pixel.RawCol.Vector**

---

Contains the raw, unshaded color vector of the surface.

Parameters:

- (0) is the new color vector.
- (1) is the current color vector.

---

---

**Pixel.Shading**

---

A picture of the diffuse shading and specular highlights applied to the objects in the scene. This is a component of the rendering calculations that depends solely on the angle of incidence of the lights on a surface. It doesn't include the effects of explicit shadow calculations.

Parameters:

- (0) is the new shading value.
- (1) is the current shading value. // 395

---

---

**Pixel.Luminous**

---

Contains the raw, unshaded luminous values of the surface.

Parameters:

- (0) is the new luminous value.
- (1) is the current luminous value.

---

---

**Pixel.Diffuse**

---

Contains the raw, unshaded diffuse values of the surface.

Parameters:

- (0) is the new diffuse value.
- (1) is the current diffuse value.

---

---

**Pixel.Specular**

---

Contains the raw, unshaded specular values of the surface.

Parameters:

- (0) is the new specular value.
- (1) is the current specular value.

---

---

### **Pixel.Mirror**

---

Contains the raw, unshaded mirror values of the surface.

Parameters:

- (0) is the new mirror value.
- (1) is the current mirror value.

---

---

### **Pixel.Transparent**

---

Contains the raw, unshaded transparent values of the surface.

Parameters:

- (0) is the new transparent value.
- (1) is the current transparent value. // 400

---

---

### **Pixel.Shade.Diff**

---

Like the Pixel.Shading buffer, but this only store the amount of diffuse , rather than adding diffuse and specular together. They should not be confused with the Pixel.Diffuse and Pixel.Specular buffers, which store the unshaded surface values for those parameters.

Parameters:

- (0) is the new diffuse shading value.
- (1) is the current diffuse shading value.

---

---

### **Pixel.Shade.Spec**

---

Like the Pixel.Shading buffer, but this only store the amount of specular , rather than adding diffuse and specular together. They should not be confused with the Pixel.Diffuse and Pixel.Specular buffers, which store the unshaded surface values for those parameters.

Parameters:

- (0) is the new diffuse specular value.
- (1) is the current specular shading value.

---

---

### **Pixel.Shadow**

---

Indicates where shadows are falling in the final image. It may also be thought of as an illumination map, showing what parts of the image are visible to the lights in the scene.

Parameters:

- (0) is the new shadow value.
- (1) is the current shadow value.

## User elements

---

---

### Result

---

This is the return value of the user element.

Parameters:

- (0) is the return value of the user element.

---

---

### Parameter 0

---

This is the first parameter passed to the user element.

Parameters:

- (0) is the first parameter received.

---

---

### Parameter 1

---

This is the second parameter passed to the user element.

Parameters:

- (0) is the second parameter received.

---

---

### Parameter 2

---

This is the third parameter passed to the user element.

Parameters:

- (0) is the third parameter received.

---

---

### Parameter 3

---

This is the 4<sup>th</sup> parameter passed to the user element.

Parameters:

- (0) is the 4<sup>th</sup> parameter received.

---

---

### Parameter 4

---

This is the 5<sup>th</sup> parameter passed to the user element.

Parameters:

- (0) is the 5<sup>th</sup> parameter received.

# Macros

---

---

## Distance

---

Distance requires two objects on the left side and calculates the distance between them.

Parameters:

- (0) is the first object.
- (1) is the second object.
- (2) is the distance between them.

---

---

## Speed

---

Speed requires one object on the left side and calculates its speed

Parameters:

- (0) is the object to investigate.
- (1) is the speed of the object.

---

---

## Distance traveled

---

Distance traveled is used to calculate the total distance (in 3D) the object has moved from a given time until now. The time parameter allows the definition of the start time for this calculation.

Parameters:

- (0) is the start time used to calculate (by default now).
- (1) is the object to investigate.
- (2) is the distance traveled by the object since the given time.

---

---

## Rot/Distance

---

Rot/Distance allows you to calculate the needed angle for a given distance a wheel has traveled. Use in conjunction with the “Distance traveled” element.

Parameters:

- (0) is the distance traveled by the object.
- (1) is the radius of the wheel.
- (2) is the resulting angle (in radiant).

---

---

## Clone number

---

Search for the clone id (the number LW appends to the object).

Parameters:

- (0) is the object.
- (1) is the clone number of the object.

---

---

### **Radiant to Degree**

---

Transforms an angle expressed in radians to the corresponding degree value.

Parameters:

- (0) is the radiant value.
- (1) is the result in degrees.

---

---

### **Degree to Radiant**

---

Transforms an angle expressed in degrees to the corresponding radiant value.

Parameters:

- (0) is the degree value.
- (1) is the result in radians.

---

---

### **Time to Frame**

---

Transforms the time expressed in seconds to the corresponding frame number.

Parameters:

- (0) is the time in seconds
- (1) is the resulting frame number.

---

---

### **Frame to Time**

---

Transforms the frame number into the corresponding time expressed in seconds.

Parameters:

- (0) is the frame number.
- (1) is the resulting time in seconds.

---

---

### **Target.H**

---

Returns the H rotation that should be applied to object (1) in order to make him “look at” object (2). If a time expressed in seconds is passed to (0) then the evaluation of the object (2) is done a this time instead of now.

Parameters:

- (0) is the given time (omitted = now).
- (1) is the object that should be rotated.
- (2) is the object to “look at”.
- (3) is the resulting H rotation.

---

---

**Target.P**

---

Returns the P rotation that should be applied to object (1) in order to make him “look at” object (2). If a time expressed in seconds is passed to (0) then the evaluation of the object (2) is done at this time instead of now.

Parameters:

- (0) is the given time (omitted = now).
- (1) is the object that should be rotated.
- (2) is the object to “look at”.
- (3) is the resulting P rotation.

---

---

**Target.B**

---

Returns the P rotation that should be applied to object (1) in order to make him “look at” object (2). If a time expressed in seconds is passed to (0) then the evaluation of the object (2) is done at this time instead of now.

Parameters:

- (0) is the given time (omitted = now).
- (1) is the object that should be rotated.
- (2) is the object to “look at”.
- (3) is the resulting P rotation.

---

---

**Path Delay**

---

Returns when a given object was at (2) distant from the current position. Useful to have an object following another with a constant distance even when the leader stop.

Parameters:

- (0) precision (default: 0.001)
- (1) is object to check.
- (2) is distance since current position.
- (3) is resulting time.

## Item information

---

---

### Item.Position.X

---

Returns the X position of an object at a given time. The returned value is in local coordinates.

Parameters:

- (0) is the time (omitted = now).
- (1) is the object.
- (2) is the X position of the object.

---

---

### Item.Position.Y

---

Returns the Y position of an object at a given time. The returned value is in local coordinates.

Parameters:

- (0) is the time (omitted = now).
- (1) is the object.
- (2) is the Y position of the object.

---

---

### Item.Position.Z

---

Returns the Z position of an object at a given time. The returned value is in local coordinates.

Parameters:

- (0) is the time (omitted = now).
- (1) is the object.
- (2) is the Z position of the object.

---

---

### Item.Rotation.H

---

Returns the H rotation of an object at a given time.

Parameters:

- (0) is the time (omitted = now).
- (1) is the object.
- (2) is the H rotation of the object.

---

---

### Item.Rotation.P

---

Returns the P rotation of an object at a given time.

Parameters:

- (0) is the time (omitted = now).
- (1) is the object.
- (2) is the P rotation of the object.

---

---

### Item.Rotation.B

---

Returns the B rotation of an object at a given time.

Parameters:

- (0) is the time (omitted = now).

- (1) is the object.
- (2) is the B rotation of the object.

---

---

**Item.Size.X**

---

Returns the X size of an object at a given time.

Parameters:

- (0) is the time (omitted = now).
- (1) is the object.
- (2) is the X size of the object.

---

---

**Item.Size.Y**

---

Returns the Y size of an object at a given time.

Parameters:

- (0) is the time (omitted = now).
- (1) is the object.
- (2) is the Y size of the object.

---

---

**Item.Size.Z**

---

Returns the Z size of an object at a given time.

Parameters:

- (0) is the time (omitted = now).
- (1) is the object.
- (2) is the Z size of the object.

---

---

**Item.Parent**

---

Returns the item's parent, if any.

Parameters:

- (0) is the object to investigate.
- (1) is the parent item.

---

---

**Item.Target**

---

Returns the item's target, if any.

Parameters:

- (0) is the object to investigate.
- (1) is the target item.

---

---

**Item.Goal**

---

Returns the item's goal, if any.

Parameters:

- (0) is the object to investigate.
- (1) is the target item.

---

---

**Item.Type**

---

Return the type of an item.

Result values are:

0: Object

1: Light

2: Camera

3: Bone.

Parameters:

- (0) is the object to investigate.

- (1) is the item's type.

---

---

**Item.LookAhead**

---

Returns the look-ahead interval, in seconds, for motion channels controlled by "Align to Path Look-ahead". This is the amount of time by which changes in orientation of the item anticipate changes in the path direction.

Parameters:

- (0) is the object to investigate.

- (1) is the look ahead value.

---

---

**Item.GoalStrength**

---

Returns the item's IK goal strength.

Parameters:

- (0) is the object to investigate.

- (1) is the goal strength value.

---

---

**Item.Position.Vector**

---

Returns the position vector of an item at a given time.

Parameters:

- (0) is the time (omitted = now).

- (1) is the item.

- (2) is the position vector of the item.

---

---

**Item.Rotation.Vector**

---

Returns the rotation vector of an item at a given time.

Parameters:

- (0) is the time (omitted = now).

- (1) is the item.

- (2) is the position vector of the item.

---

---

## **Item.Size.Vector**

---

Returns the size vector of an item at a given time.

Parameters:

- (0) is the time (omitted = now).
- (1) is the item.
- (2) is the position vector of the item.

## Scene information

---

---

### Frames/Second

---

The Number of frames per second of animation. Can be used to transform the time element in the current frame.

Parameters:

- (0) is the number of frames per second.

---

---

### Frame Start

---

The number of the first frame defined for the scene. This is the rendering limits, not to be confused with the limits set for previews.

Parameters:

- (0) is the start frame number.

---

---

### Frame End

---

The number of the last frame defined for the scene. This is the rendering limits, not to be confused with the limits set for previews.

Parameters:

- (0) is the end frame number.

---

---

### Frame Step

---

The step size, in frames that Lightwave uses during rendering (the user setting for the Frame Step).

Parameters:

- (0) is the step size.

---

---

### Frame Width

---

Rendered image width in pixels.

Parameters:

- (0) is the image width.

---

---

### Frame Height

---

Rendered image height in pixels.

Parameters:

- (0) is the image height.

---

---

### **Pixel Aspect**

---

The aspect ratio of the pixels in the image, expressed as width/height. Values greater than 1.0 mean short wide pixels and values less than 1.0 mean tall thin pixels.

Parameters:

- (0) is the aspect ratio.

## Objects information

---

---

### Object.NumPoints

---

Returns the number of points in the object mesh.

Parameters:

- (0) is the object to investigate.
- (1) is the number of points.

---

---

### Object.NumPolygons

---

Returns the number of polygons in the object mesh.

Parameters:

- (0) is the object to investigate.
- (1) is the number of polygons.

---

---

### Object.Dissolve

---

Returns the object dissolve amount at the given time.

Parameters

- (0) is the given time (omitted = now).
- (1) is the object to investigate.
- (2) is the dissolve amount.

---

---

### Object.Patch.Interface

---

Returns the interface patch level for the object's subpatches.

Parameters:

- (0) is the object to investigate.
- (1) is the the patch level.

---

---

### Object.Patch.Render

---

Returns the render patch level for the object's subpatches.

Parameters:

- (0) is the object to investigate.
- (1) is the the patch level.

---

---

### Object.MBall.Interface

---

Returns the interface resolution of the object's metaballs.

Parameters:

- (0) is the object to investigate.
- (1) is the the metaballs' resolution.

---

---

**Object.MBall.Render**

---

Returns the render resolution of the object's metaballs.

Parameters:

- (0) is the object to investigate.
- (1) is the the metaballs' resolution.

---

---

**Object.BoneSource**

---

Returns the object whose bones are being used to deform the given object. (An object can be deformed by the bones of another object.)

Parameters:

- (0) is the object to investigate.
- (1) is the object whose bones are being used.

---

---

**Object.MorphTarget**

---

Returns the morph target of the given object.

Parameters:

- (0) is the object to investigate.
- (1) is the object used as morph target.

---

---

**Object.Fog**

---

Returns the amount by which the object is affected by fog.

Parameters:

- (0) is the given time (omitted = now).
- (1) is the object to investigate.
- (2) is the dissolve amount.

## Cameras information

---

---

### Camera.ZoomFactor

---

Returns the zoom factor.

Parameters:

- (0) is the given time (if this is omitted the current time is used).
- (1) is the object to investigate.
- (2) is the zoom value.

---

---

### Camera.FocalLength

---

Returns the focal length in millimeters.

Parameters:

- (0) is the given time (if this is omitted the current time is used).
- (1) is the object to investigate.
- (2) is the focal length value.

---

---

### Camera.FocalDistance

---

Returns the distance from the camera at which objects are in focus.

Parameters:

- (0) is the given time (if this is omitted the current time is used).
- (1) is the object to investigate.
- (2) is the focal distance.

---

---

### Camera.FStop

---

Returns the f-stop number.

Parameters:

- (0) is the given time (if this is omitted the current time is used).
- (1) is the object to investigate.
- (2) is the f-stop value.

---

---

### Camera.BlurLength

---

Returns the blur length as a fraction of the frame time.

Parameters:

- (0) is the given time (if this is omitted the current time is used).
- (1) is the object to investigate.
- (2) is the blur length value.

## Bones information

---

---

### **Bone.RestLength**

---

Returns the rest length of the bone.

Parameters:

- (0) is the object to investigate.
- (1) is the rest length value.

---

---

### **Bone.Strength**

---

Returns the bone strength setting.

Parameters:

- (0) is the object to investigate.
- (1) is the bone strength value.

## Lights information

---

---

### **Ambient.R**

---

Returns the red component of the color of the global ambient light for the scene at the given time. The RGB levels include the effect of the user's intensity setting for the ambient light.

Parameters:

- (0) is the given time (if this is omitted the current time is used).
- (1) is the object to investigate.
- (2) is the red value.

### **Ambient.G**

---

Returns the green component of the color of the global ambient light for the scene at the given time. The RGB levels include the effect of the user's intensity setting for the ambient light.

Parameters:

- (0) is the given time (if this is omitted the current time is used).
- (1) is the object to investigate.
- (2) is the green value.

### **Ambient.B**

---

Returns the blue component of the color of the global ambient light for the scene at the given time. The RGB levels include the effect of the user's intensity setting for the ambient light.

Parameters:

- (0) is the given time (if this is omitted the current time is used).
- (1) is the object to investigate.
- (2) is the blue value.

### **Light.Type**

---

Returns the type of the light as one of the following values:

- 0: Distant
- 1: Point
- 2: Spot
- 3: Linear
- 4: Area

Parameters:

- (0) is the object to investigate.
- (1) is the light type.

---

---

**Light.Color.R**

---

Returns just the red color information about the light without the intensity.

Parameters:

- (0) is the given time (if this is omitted the current time is used).
- (1) is the object to investigate.
- (2) is the red value.

---

---

**Light.Color.G**

---

Returns just the green color information about the light without the intensity.

Parameters:

- (0) is the given time (if this is omitted the current time is used).
- (1) is the object to investigate.
- (2) is the green value.

---

---

**Light.Color.B**

---

Returns just the blue color information about the light without the intensity.

Parameters:

- (0) is the given time (if this is omitted the current time is used).
- (1) is the object to investigate.
- (2) is the blue value.

---

---

**Light.ShadowType**

---

Returns the shadow type for the light as one of the following values:

0: No shadow

1: Raytraced

2: Shadowmap

Parameters:

- (0) is the object to investigate.
- (1) is the shadow type.

---

---

**Light.ConeAngle**

---

Returns the cone angle for spotlights. It's half the angle of the total light cone angle. The angle is in radians.

Parameters:

- (0) is the given time (if this is omitted the current time is used).
- (1) is the object to investigate.
- (2) is the cone angle value.

---

---

### **Light.ConeEdge**

---

Returns the cone edge for spotlights. It's the angular width of the soft edge. The angle is in radians.

Parameters:

- (0) is the given time (if this is omitted the current time is used).
- (1) is the object to investigate.
- (2) is the cone edge value.

---

---

### **Light.Range**

---

Returns the range or nominal distance for the light. The interpretation of this value depends on the falloff type. If falloff is linear, the value is the distance at which the intensity of the light falls to 0. For inverse distance falloff types, the value is the distance at which the intensity equals the user's intensity setting for the light.

Parameters:

- (0) is the given time (if this is omitted the current time is used).
- (1) is the object to investigate.
- (2) is the range value.

---

---

### **Light.FalloffType**

---

Returns the falloff type. Falloff scales the intensity of a light as a function of d (distance from the light) and r (the value returned by the range function).

Possible values:

- 0: No falloff
- 1: Linear falloff
- 2: Inverse distance
- 3: Inverse distance <sup>2</sup>

Parameters:

- (0) is the object to investigate.
- (1) is the falloff type.

---

---

### **Light.Intensity**

---

Returns the intensity of the light source.

Parameters:

- (0) is the given time (if this is omitted the current time is used)
- (1) is the object to investigate.
- (2) is the intensity value.

# Events

---

---

## **Last Event Time**

---

This element returns the time when the condition last changed from false to true. This can be useful, for example, to decrease the spinning speed of a wheel as soon as the wheel is no longer touching the ground.

Parameters:

- (0) is the condition to check.
- (1) is the last time the event occurred.

---

---

## **Event.Time**

---

This element must be used with “Last Event Time” and placed inside the condition. It will be used to check the time when a given channel changed.

Parameters:

- (0) is the time to check when the event occurred.

# Transformations

---

---

## World->Local Position.X

---

Transforms world coordinates to the local object equivalent and returns the local X.

Parameters:

- (0) is the X world coordinate (if omitted take the object X world coordinate).
- (1) is the Y world coordinate (if omitted take the object Y world coordinate).
- (2) is the Z world coordinate (if omitted take the object Z world coordinate).
- (3) is the time used to make the transformation (if omitted now).
- (4) is the object.
- (5) is the resulting X local coordinate.

---

---

## World->Local Position.Y

---

Transforms world coordinates to the local object equivalent and returns the local Y.

Parameters:

- (0) is the X world coordinate (if omitted take the object X world coordinate).
- (1) is the Y world coordinate (if omitted take the object Y world coordinate).
- (2) is the Z world coordinate (if omitted take the object Z world coordinate).
- (3) is the time used to make the transformation (if this is omitted the current time is used).
- (4) is the object.
- (5) is the resulting Y local coordinate.

---

---

## World->Local Position.Z

---

Transforms world coordinates to the local object equivalent and returns the local Z.

Parameters:

- (0) is the X world coordinate (if omitted take the object X world coordinate).
- (1) is the Y world coordinate (if omitted take the object Y world coordinate).
- (2) is the Z world coordinate (if omitted take the object Z world coordinate).
- (3) is the time used to make the transformation (if this is omitted the current time is used).
- (4) is the object.
- (5) is the resulting Z local coordinate.

---

---

## World->Local Rotation.H

---

Transforms world rotation to the local object equivalent and returns the local H.

Parameters:

- (0) is the H world rotation (if omitted take the object H world rotation).
- (1) is the P world rotation (if omitted take the object P world rotation).
- (2) is the B world rotation (if omitted take the object B world rotation).
- (3) is the time used to make the transformation (if this is omitted the current time is used).
- (4) is the object.
- (5) is the resulting H local rotation.

---

---

**World->Local Rotation.P**

---

Transforms world rotation to the local object equivalent and returns the local P.

Parameters:

- (0) is the H world rotation (if omitted take the object H world rotation).
- (1) is the P world rotation (if omitted take the object P world rotation).
- (2) is the B world rotation (if omitted take the object B world rotation).
- (3) is the time used to make the transformation (if this is omitted the current time is used).
- (4) is the object.
- (5) is the resulting P local rotation.

---

---

**World->Local Rotation.B**

---

Transforms world rotation to the local object equivalent and returns the local B.

Parameters:

- (0) is the H world rotation (if omitted take the object H world rotation).
- (1) is the P world rotation (if omitted take the object P world rotation).
- (2) is the B world rotation (if omitted take the object B world rotation).
- (3) is the time used to make the transformation (if this is omitted the current time is used).
- (4) is the object.
- (5) is the resulting B local rotation.

---

---

**World->Local Size.X**

---

Transforms world size to the local object equivalent and returns the local X.

Parameters:

- (0) is the X world size (if omitted take the object X world size).
- (1) is the Y world size (if omitted take the object Y world size).
- (2) is the Z world size (if omitted take the object Z world size).
- (3) is the time used to make the transformation (if this is omitted the current time is used).
- (4) is the object.
- (5) is the resulting X local size.

---

---

**World->Local Size.Y**

---

Transforms world size to the local object equivalent and returns the local Y.

Parameters:

- (0) is the X world size (if omitted take the object X world size).
- (1) is the Y world size (if omitted take the object Y world size).
- (2) is the Z world size (if omitted take the object Z world size).
- (3) is the time used to make the transformation (if this is omitted the current time is used).
- (4) is the object.
- (5) is the resulting Y local size.

---

---

**World->Local Size.Z**

---

Transforms world size to the local object equivalent and returns the local Z.

Parameters:

- (0) is the X world size (if omitted take the object X world size).
- (1) is the Y world size (if omitted take the object Y world size).
- (2) is the Z world size (if omitted take the object Z world size).
- (3) is the time used to make the transformation (if this is omitted the current time is used).
- (4) is the object.
- (5) is the resulting Z local size.

---

---

**World.Position.X**

---

Returns the world position X of the object.

Parameters:

- (0) is the object.
- (1) is the X world position.

---

---

**World.Position.Y**

---

Returns the world position Y of the object.

Parameters:

- (0) is the object.
- (1) is the Y world position.

---

---

**World.Position.Z**

---

Returns the world position Z of the object.

Parameters:

- (0) is the object.
- (1) is the Z world position.

---

---

**World.Rotation.H**

---

Returns the world rotation H of the object.

Parameters:

- (0) is the object.
- (1) is the H world rotation.

---

---

**World.Rotation.P**

---

Returns the world rotation P of the object.

Parameters:

- (0) is the object.
- (1) is the P world rotation.

---

---

**World.Rotation.B**

---

Returns the world rotation B of the object.

Parameters:

- (0) is the object.
- (1) is the B world rotation.

---

---

**World.Size.X**

---

Returns the world size X of the object.

Parameters:

- (0) is the object.
- (1) is the X world size.

---

---

**World.Size.Y**

---

Returns the world size Y of the object.

Parameters:

- (0) is the object.
- (1) is the Y world size.

---

---

**World.Size.Z**

---

Returns the world size Z of the object.

Parameters:

- (0) is the object.
- (1) is the Z world size.

# Vectors

---

---

## Vector

---

Returns a vector created using 3 values.

Parameters:

- (0) is the first value of the vector.
- (1) is the second value of the vector.
- (2) is the third value of the vector.
- (3) is the new vector.

---

---

## Vector.Val[0]

---

Returns the first value of the vector.

Parameters:

- (0) is the vector to explode.
- (1) is the first value of the vector.

---

---

## Vector.Val[1]

---

Returns the second value of the vector.

Parameters:

- (0) is the vector to explode.
- (1) is the second value of the vector.

---

---

## Vector.Val[2]

---

Returns the third value of the vector.

Parameters:

- (0) is the vector to explode.
- (1) is the third value of the vector.

---

---

## Vector.Length

---

Returns the pythagoras of the vector ( $\text{length}=\sqrt{a*a+b*b+c*c}$ ).

Parameters:

- (0) is the vector.
- (1) is the length of the vector.



<b>S</b>	
Sample.Object.Vector	44
Sample.Object.X	40
Sample.Object.Y	40
Sample.Object.Z	40
Sample.World.Vector	44
Sample.World.X	40
Sample.World.Y	40
Sample.World.Z	40
Sin	27
Smaller	26
Smaller or Equal	26
Speed	50
Sqrt	29
Subtract	23

<b>T</b>	
Tan	27
Target.B	52
Target.H	51
Target.P	52
Texture value	39
Texture.A	39
Texture.Amp	41
Texture.B	39
Texture.Color.Vector	41

Texture.G	39
Texture.R	39
Texture.Size.Vector	44
Texture.Size.X	40
Texture.Size.Y	41
Texture.Size.Z	41
Texture.SpotSize	41
This	34
This.Channel	38
This.Position.Vector	36
This.Position.X	34
This.Position.Y	34
This.Position.Z	34
This.Rotation.B	35
This.Rotation.H	35
This.Rotation.P	35
This.Rotation.Vector	36
This.Size.Vector	37
This.Size.X	35
This.Size.Y	36
This.Size.Z	36
This.Speed	36
Time	34, 38, 39
Time to Frame	51
Title	22

<b>V</b>	
Vector	71
Vector.Length	71
Vector.Val[0]	71
Vector.Val[1]	71
Vector.Val[2]	71

<b>W</b>	
World.Position.X	69
World.Position.Y	69
World.Position.Z	69
World.Rotation.B	70
World.Rotation.H	69
World.Rotation.P	69
World.Size.X	70
World.Size.Y	70
World.Size.Z	70
World->Local Position.X	67
World->Local Position.Y	67
World->Local Position.Z	67
World->Local Rotation.B	68
World->Local Rotation.H	67
World->Local Rotation.P	68
World->Local Size.X	68
World->Local Size.Y	68
World->Local Size.Z	69