After Effects Expression Reference

Release 0.0.3

Jun 11, 2018

Introduction

The reference is still being edited and improved upon.

CHAPTER 1

Introduction

Use the After Effects expression elements along with standard JavaScript elements to write your expressions. You can use the Expression Language menu at any time to insert methods and attributes into an expression, and you can use the pick whip at any time to insert properties.

If an argument description contains an equal sign (=) and a value (such as t=time or width=.2), then the argument uses the included default value if you don't specify a different value.

Some argument descriptions include a number in square brackets—this number indicates the dimension of the expected property or Array.

Some return-value descriptions include a number in square brackets—this number specifies the dimension of the returned property or Array. If a specific dimension is not included, the dimension of the returned Array depends on the dimension of the input.

The W3Schools JavaScript reference website provides information for the standard JavaScript language, including pages for the JavaScript Math and String objects.

CHAPTER 2

Resources

The user 'Beaver' posted 5 Expressions that will change your life on the Mograph forums.

Dan Ebberts provides example expressions and tutorials for learning how to work with expressions on his MotionScript website. For example, Dan provides an excellent page about collision detection.

Colin Braley provides a tutorial and example project on his website that show how to use expressions to make one layer repel others in a natural-seeming manner.

The AE Enhancers forum provides many examples and much useful information about expressions, as well as scripts and animation presets. In this post on the AE Enhancers forum, Paul Tuersley provides a tutorial and example project that show how to use expressions to animate several layers in a swarm.

Rick Gerard provides an example on his website that demonstrates rolling a square object along a floor so that the sides stay in contact with the floor plane.

Carl Larsen provides a video tutorial on the Creative COW website that demonstrates how to use expressions and parenting to relate the rotation of a set of wheels to the horizontal movement of a vehicle.

Chris Zwar provides an example project on his website for automatically arranging still images or videos into a grid (like a video wall). You can easily adjust position and spacing with sliders that are connected to a system of expressions. There are three compositions in the project—one for stills, one for videos, and one to create an auto-storyboard in which a video is sampled at user-defined intervals and aligned into a grid.

JJ Gifford's website provides several example projects that demonstrate how to use expressions.

Maltaannon (Jerzy Drozda, Jr.) provides a video tutorial on his website that shows how to use expressions to create a volume meter using the results of the Convert Audio To Keyframes command.

CHAPTER 3

Examples

Note: Many of the examples in this section are based on expressions provided by Dan Ebberts.

3.1 Get this project's AEP name (AE 15.1+ only)

While there is no method to directly access your AEP's name, you CAN get the full path to the AEP.

With some string manipulation, you can derive the aep name from this:

```
var aepName = thisProject.fullPath.split($.os.indexOf("Windows") > -1 ? "\\" : "/").
↪pop();
```

If you wanted to write "Unsaved" in that case, you can use the following expression:

```
var aepName = thisProject.fullPath.split($.os.indexOf("Windows") > -1 ? "\\" : "/").
↪pop();
aepName = aepName === "" ? "Unsaved" : aepName;
```

3.2 Make a layer revolve in a circle

You can create an expression without using properties from other layers. For example, you can make a layer revolve in a perfect circle.

1. Select a layer, press P to reveal its Position property in the Timeline panel, and Alt-click (Windows) or Option-click (Mac OS) the stopwatch to the left of the property name.
2. Enter the following in the expression field:

```
[(thisComp.width/2), (thisComp.height/2)] + [Math.sin(time)*50, -Math.
↪cos(time)*50]
```

3.3 Rotate the hands of a clock

You can use the pick whip to link rotation values between layers to animate the hands on a clock—as the hour hand moves from hour to hour, the minute hand rotates the full circumference of the clock face. This type of animation would take a long time to create if you had to set each keyframe for both hand layers, but with the pick whip, you can do it in a matter of minutes.

1. Import or create two long, narrow solid-color layers: an hour hand and a minute hand.
2. Set the anchor points at the ends of the layers.
3. Move the layers so that the anchor points are at the center of the composition.
4. Set Rotation keyframes for the hour hand.
5. Select the Rotation property for the minute hand and choose `Animation > Add Expression.`
6. Drag the pick whip to the Rotation property for the hour hand. The following expression appears:

```
thisComp.layer("hour hand").rotation
```

7. To make the minute hand rotate 12 times as fast as the hour hand, add `* 12` at the end of the expression as follows:

```
thisComp.layer("hour hand").rotation * 12
```

3.4 Position one layer between two others

This example expression positions and maintains one layer at a balanced distance between two other layers.

1. Start with three layers.
2. Animate the positions of the first two layers in the Timeline panel.
3. Select the third layer, press P to reveal the Position property, and Alt-click (Windows) or Option-click (Mac OS) the stopwatch button to the left of the property name.
4. Enter the following in the expression field:

```
(thisComp.layer(1).position + thisComp.layer(2).position)/2
```

3.5 Create a trail of images

This example expression instructs a layer to be at the same position as the next higher layer in the Timeline panel, but delayed by a specified amount of time (in this case, 0.5 seconds). You can set similar expressions for the other geometric properties.

1. Start with two solid-color layers that are scaled to approximately 30% of the composition size. (See Solid-color layers and solid-color footage items.)
2. Animate the position of the first layer.
3. Select the second layer, press P to reveal the Position property, and Alt-click (Windows) or Option-click (Mac OS) the stopwatch button to the left of the property name.
4. Enter the following in the expression field:

```
thisComp.layer(thisLayer, -1).position.valueAtTime(time - .5)
```

5. Duplicate the last layer five times by selecting it and pressing Ctrl+D (Windows) or Command+D (Mac OS) five times.

All layers follow the same path, and each is delayed 0.5 seconds from the previous.

Note: Dan Ebberts provides more examples and techniques for creating trails of images on his MotionScript website.

3.6 Create a bulge between two layers

This example expression synchronizes the Bulge Center argument of the Bulge effect in one layer with the position of another layer. For example, you can create an effect that looks like a magnifying glass moving over a layer, with the contents under the magnifying glass bulging as the lens (that is, the overlying layer) moves. This expression uses the fromWorld method, which makes the expression work correctly regardless of whether you move the magnifying glass layer or the underlying layer. You can rotate or scale the underlying layer, and the expression stays intact.

You can also use other effects, such as Ripple, with this expression.

1. Start with two layers. Make one layer a magnifying glass or similar object with a hole in the middle and name it Magnifier. (See Creating layers.)
2. Animate the position of the magnifying glass layer. (See Motion paths.)
3. Apply the Bulge effect to the other layer. (See Apply an effect or animation preset.)
4. Select the Bulge Center property of the Bulge effect in the Timeline panel and choose Animation > Add Expression, or Alt-click (Windows) or Option-click (Mac OS) the stopwatch button for the property.
5. Select the default expression text and type the following:

```
fromWorld(thisComp.layer("Magnifier").position)
```

3.7 Fade opacity of a 3D layer based on distance from camera

Apply the following expression to the Opacity property of a 3D layer:

```
startFade = 500; // Start fade 500 pixels from camera.
endFade = 1500;  // End fade 1500 pixels from camera.

try { // Check whether there's a camera
C = thisComp.activeCamera.toWorld([0,0,0]);
} catch (err) { // No camera, so assume 50mm
w = thisComp.width * thisComp.pixelAspect;
z = (w/2)/Math.tan(degreesToRadians(19.799));
C = [0,0,-z];
}

P = toWorld(anchorPoint);
d = length(C,P);

linear(d,startFade,endFade,100,0)
```

The fade starts at a distance of `500` pixels from the camera and is complete at `1500` pixels from the camera. The linear interpolation method is used to map distance values to opacity values.

3.8 Make a 3D layer invisible if facing away from camera

Apply the following expression to the Opacity property of a 3D layer:

```
if (toCompVec([0, 0, 1])[2] > 0 ) value else 0
```

Note: Dan Ebberts explains this expression on his MotionScript website.

3.9 Flip layer horizontally if facing away from camera

Apply the following expression to the Scale property of a 3D layer:

```
if (toCompVec([0, 0, 1])[2] > 0 ) value else [-value[0], value[1], value[2]]
```

3.10 Animate scale at each layer marker

Apply the following expression to a Scale property to make a layer wobble at each marker:

```
n = 0;
t = 0;

if (marker.numKeys > 0){
    n = marker.nearestKey(time).index;
    if (marker.key(n).time > time) n--;
}

if (n > 0) t = time - marker.key(n).time;

amp = 15;
freq = 5;
decay = 3.0;

angle = freq * 2 * Math.PI * t;
scaleFact = (100 + amp * Math.sin(angle) / Math.exp(decay * t)) / 100;
[value[0] * scaleFact, value[1] / scaleFact];
```

3.11 Start or stop wiggle at specific time

You can use any expression in place of the wiggle expression used here, to begin and end the influence of any expression at a specific time.

Apply the following expression to a property to wiggle it beginning at time 2 seconds:

```
timeToStart = 2;
if (time > timeToStart) {
wiggle(3,25);
} else {
    value;
}
```

Apply the following expression to a property to stop wiggling it at time 4 seconds:

```
timeToStop = 4;

if (time > timeToStop) {
        value;
} else {
        wiggle(3,25);
}
```

Apply the following expression to a property to start wiggling it at time 2 seconds and stop wiggling it at time 4 seconds:

```
timeToStart = 2;
timeToStop = 4;

if ((time > timeToStart) && (time < timeToStop)) {
  wiggle(3,25);
} else {
  value;
}
```

3.12 Match camera focal plane to another layer

Apply the following expression to the Focus Distance property of a camera layer to have its focus distance match the distance to the anchor point of a layer named “target”:

```
target = thisComp.layer("target");
V1 = target.toWorld(target.anchorPoint) - toWorld([0,0,0]);
V2 = toWorldVec([0,0,1]);
dot(V1,V2);
```

Note: Dan Ebberts explains this expression example in detail on his Motionscript website.

CHAPTER 4

Global

4.1 comp(name)

Description

Retrieves another composition by name.

Parameters

name	String

Type

Comp

4.2 footage(name)

Description

Retrieves a footage item by name.

Parameters

name	String

Type

Footage

4.3 thisProject

Description

Represents the project which contains the expression.

Type

Project

4.4 thisComp

Description

Represents the composition containing the expression.

Type

Comp

4.5 thisLayer

Description

Represents the layer containing the expression. Because thisLayer is the default object, its use is optional. For example, you can start an expression with thisLayer.width or width and get the same result.

Type

Layer, Light, or Camera

4.6 thisProperty

Description

Represents the property containing the expression. For example, if you write an expression on the Rotation property, you can start an expression with thisProperty to refer to the Rotation property.

Type

Property

4.7 time

Description

Represents the composition time, in seconds, at which the expression is being evaluated.

Type

Number

4.8 colorDepth

Description

Type the project color depth value. For example, colorDepth returns `16` when the project color depth is 16 bits per channel.

Type

Number

4.9 posterizeTime(framesPerSecond)

Description

This expression allows you to set the frame rate for a property to be lower than the frame rate of the composition.

For example, the following expression updates the property value with a random value once per second:

```
posterizeTime(1);
random()
```

Parameters

framesPerSecond	Number

The framesPerSecond value becomes the frame rate from which the rest of the expression operates.

Type

Number

4.10 value

Description

Represents the value at the current time for the property containing the expression.

Type

Number, Array, or String

CHAPTER 5

Time conversion

Note: If you want more control over the look of timecode in your footage, use the timeToCurrentFormat method or other timeTo methods to generate the timecode instead of using the Timecode or Numbers effect.

Create a text layer, add an expression to the Source Text property, and enter timeToCurrentFormat() in the expression field. With this method, you can format and animate the timecode text. In addition, the timecode uses the same display style defined by the current project settings.

timeToFrames(`t=time + thisComp.displayStartTime, fps=1.0 / thisComp.frameDuration, isDuration=false`) **********

Desciption

Converts the value of *t*, which defaults to the current composition time, to an integer number of frames. The number of frames per second is specified in the fps argument, which defaults to the frame rate of the current composition (`1.0 / thisComp.frameDuration`).

The `isDuration` argument, which defaults to `false`, should be `true` if the `t` value represents a difference between two times instead of an absolute time. Absolute times are rounded down toward negative infinity; durations are rounded away from zero (up for positive values).

Parameters

`t`	Number
`fps`	Number
`isDuration`	Boolean

Type

Number

5.1 framesToTime(frames, fps=1.0 / thisComp.frameDuration)

Desciption

The inverse of timeToFrames. Returns the time corresponding to the frames argument, which is required. It doesn't have to be an integer. See timeToFrames for explanation of the `fps` argument.

Parameters

`frames`	Number
`fps`	Number

Type

Number

5.2 timeToTimecode(t=time + thisComp.displayStartTime, timecodeBase=30, isDuration=false)

Desciption

Converts the value of `t` to a String representing timecode. See timeToFrames for an explanation of the `t` and `isDuration` arguments. The `timecodeBase` value, which defaults to `30`, specifies the number of frames in one second.

Parameters

`t`	Number
`timecodeBase`	Number
`isDuration`	Boolean

Type

String

5.3 timeToNTSCTimecode(t=time + thisComp.displayStartTime, ntscDropFrame=false, isDuration=false)

Desciption

Converts `t` to a `String` representing NTSC timecode. See timeToFrames for an explanation of the `t` and `isDuration` arguments. If `ntscDropFrame` is `false` (the default), the result `String` is NTSC non-drop-frame timecode. If `ntscDropFrame` is `true`, the result `String` is NTSC drop-frame timecode.

Parameters

t	Number
ntscDropFrame	Boolean
isDuration	Boolean

Type

String

5.4 timeToFeetAndFrames(t=time + thisComp.displayStartTime, fps=1.0 / thisComp.frameDuration, framesPerFoot=16, isDuration=false)

Desciption

Converts the value of `t` to a `String` representing feet of film and frames. See timeToFrames for an explanation of the `t`, `fps`, and `isDuration` arguments. The `framesPerFoot` argument specifies the number of frames in one foot of film. It defaults to `16`, which is the most common rate for 35mm footage.

Parameters

t	Number
framesPerFoot	Number
isDuration	Boolean

Type

String

5.5 timeToCurrentFormat(t=time + thisComp.displayStartTime, fps=1.0 / thisComp.frameDuration, isDuration=false)

Desciption

Converts the value of `t` to a `String` representing time in the current Project Settings display format. See timeToFrames for a definition of all the arguments.

Note: An optional `ntscDropFrame` argument was added to the `timeToCurrentFormat()` function in After Effects CS5.5 and later. Default: `ntscDropFrame=thisComp.ntscDropFrame`.

Parameters

t	Number
fps	Number
isDuration	Boolean

Type

String

CHAPTER 6

Vector Math

Vector Math functions are global methods that perform operations on arrays, treating them as mathematical vectors. Unlike built-in JavaScript methods, such as `Math.sin`, these methods are not used with the Math prefix. Unless otherwise specified, Vector Math methods are lenient about dimensions and return a value that is the dimension of the largest input Array object, filling in missing elements with zeros.

For example, the expression `add([10, 20], [1, 2, 3])` returns `[11, 22, 3]`.

Note: JJ Gifford's website provides explanations and examples that show how to use simple geometry and trigonometry with expressions.

6.1 add(vec1, vec2)

Description

Adds two vectors.

Parameters

`vec1`	Array
`vec2`	Array

Type

Array

6.2 sub(vec1, vec2)

Description

Subtracts two vectors.

Parameters

vec1	Array
vec2	Array

Type

Array

6.3 mul(vec, amount)

Description

Multiplies every element of the vector by the amount.

Parameters

vec	Array
amount	Number

Type

Array

6.4 div(vec, amount)

Description

Divides every element of the vector by the amount.

Parameters

vec	Array
amount	Number

Type

Array

6.5 clamp(value, limit1, limit2)

Description

The value of each component of `value` is constrained to fall between the values of the corresponding values of `limit1` and `limit2`.

Parameters

`value`	Number or Array
`limit1`	Number or Array
`limit2`	Number or Array

Type

Number or Array

6.6 dot(vec1, vec2)

Description

Returns the dot (inner) product of the vector arguments.

Parameters

`vec1`	Array
`vec2`	Array

Type

Number

6.7 cross(vec1, vec2)

Description

Returns the vector cross product of `vec1` and `vec2`. Refer to a math reference or JavaScript guide for more information.

Parameters

`vec1`	Array (2- or 3-dimensional)
`vec2`	Array (2- or 3-dimensional)

Type

Array (2- or 3-dimensional)

6.8 normalize(vec)

Description

Normalizes the vector so that its length is `1.0`. Using the normalize method is a short way of performing the operation `div(vec, length(vec))`.

Parameters

`vec`	Array

Type

Array

6.9 length(vec)

Description

Returns the length of vector `vec`.

Parameters

`vec`	Array

Type

Number

6.10 length(point1, point2)

Description

Returns the distance between two points. The `point2` argument is optional.

For example, `length(point1, point2)` is the same as `length(sub(point1, point2))`.

For example, add this expression to the Focus Distance property of a camera to lock the focal plane to the camera's point of interest so that the point of interest is in focus:

```
length(position, pointOfInterest)
```

Parameters

`point1`	Array
`point2`	Array

Type

Number

6.11 lookAt(fromPoint, atPoint)

Description

The argument `fromPoint` is the location in world space of the layer you want to orient. The argument `atPoint` is the point in world space you want to point the layer at. The return value can be used as an expression for the Orientation property, making the z-axis of the layer point at atPoint.

This method is especially useful for cameras and lights. If you use this expression on a camera, turn off auto-orientation.

For example, this expression on the Orientation property of a spot light makes the light point at the anchor point of layer number 1 in the same composition:

```
lookAt(position, thisComp.layer(1).position)
```

Parameters

`fromPoint`	Array (3-dimensional)
`atPoint`	Array (3-dimensional)

Type

Array (3-dimensional)

CHAPTER 7

Random Numbers

Note: The wiggle method—which is used to randomly vary a property value—is in the Property attributes and methods category. See Property attributes and methods.

7.1 seedRandom(offset, timeless=false)

Description

The random and gaussRandom methods use a seed value that controls the sequence of numbers. By default, the seed is computed as a function of a unique layer identifier, the property within the layer, the current time, and an offset value of 0. Call seedRandom to set the offset to something other than 0 to create a different random sequence. Use true for the timeless argument to not use the current time as input to the random seed. Using true for the timeless argument allows you to generate a random number that doesn't vary depending on the time of evaluation. The offset value, but not the timeless value, is also used to control the initial value of the wiggle function.

For example, this expression on the Opacity property sets the Opacity value to a random value that does not vary with time:

```
seedRandom(123456, true);
random()*100
```

The multiplication by 100 in this example converts the value in the range 0–1 returned by the random method into a number in the range 0–100; this range is more typically useful for the Opacity property, which has values from 0% to 100%.

Parameters

offset	Number
timeless	Boolean

Type

None

7.2 random()

Description

Returns a random number in the range `0-1`.

Note: In After Effects CC and CS6, the behavior of random() is changed to be more random when layer IDs are close together. The wiggle() expression is not affected.

Type

Number

7.3 random(maxValOrArray)

Description

If `maxValOrArray` is a `Number`, this method returns a number in the range from `0` to `maxValOrArray`. If `maxValOrArray` is an `Array`, this method returns an Array with the same dimension as `maxValOrArray`, with each component ranging from `0` to the corresponding component of `maxValOrArray`.

Parameters

`maxValOrArray`	Number or Array

Type

Number or Array

7.4 random(minValOrArray, maxValOrArray)

Description

If `minValOrArray` and `maxValOrArray` are `Numbers`, this method returns a number in the range from `minValOrArray` to `maxValOrArray`.

If the arguments are `Arrays`, this method returns an `Array` with the same dimension as the argument with the greater dimension, with each component in the range from the corresponding component of `minValOrArray` to the corresponding component of `maxValOrArray`.

For example, the expression `random([100, 200], [300, 400])` returns an `Array` whose first value is in the range `100-300` and whose second value is in the range `200-400`. If the dimensions of the two input *Arrays* don't match, higher-dimension values of the shorter Array are filled out with zeros.

Parameters

`minValOrArray`	Number or Array
`maxValOrArray`	Number or Array

Type

Number or Array

7.5 gaussRandom()

Description

The results have a Gaussian (bell-shaped) distribution. Approximately `90%` of the results are in the range `0-1`, and the remaining `10%` are outside this range.

Type

Number

7.6 gaussRandom(`maxValOrArray`)

Description

When `maxValOrArray` is a `Number`, this method returns a random number. Approximately `90%` of the results are in the `0` to `maxValOrArray` range, and the remaining `10%` are outside this range.

When `maxValOrArray` is an `Array`, this method returns an Array of random values, with the same dimension as `maxValOrArray`. `90%` of the values are in the range from `0` to `maxValOrArray`, and the remaining `10%` are outside this range.

The results have a Gaussian (bell-shaped) distribution.

Parameters

`maxValOrArray`	Number or Array

Type

Number or Array

7.7 gaussRandom(`minValOrArray`, `maxValOrArray`)

Description

If `minValOrArray` and `maxValOrArray` are `Numbers`, this method returns a random number. Approximately `90%` of the results are in the range from `minValOrArray` to `maxValOrArray`, and the remaining `10%` are outside this range.

If the arguments are `Arrays`, this method returns an `Array` of random numbers with the same dimension as the argument with the greater dimension. For each component, approximately `90%``of the results are in the range from the corresponding component of ``minValOrArray` to the corresponding component of `maxValOrArray`, and the remaining `10%` are outside this range.

The results have a Gaussian (bell-shaped) distribution.

Parameters

`minValOrArray`	Number or Array
`maxValOrArray`	Number or Array

Type

Number or Array

7.8 noise(valOrArray)

Description

Returns a number in the range from `-1` to `1`. The noise is not actually random; it is based on Perlin noise, which means that the return values for two input values that are near one another tend to be near one another. This type of noise is useful when you want a sequence of seemingly random numbers that don't vary wildly from one to the other—as is usually the case when animating any apparently random natural motion.

Example:

```
rotation + 360*noise(time)
```

Parameters

`valOrArray`	Number or an Array (2- or 3-dimensional)

Type

Number

CHAPTER 8

Interpolation

Description

For all the Interpolation methods, the argument `t` is often `time` or `value`, though it can have other values, instead. If `t` is `time`, the interpolation between values happens over a duration. If `t` is `value`, then the expression maps one range of values to a new range of values.

For additional explanations and examples of the Interpolation methods, see JJ Gifford's website.

Chris and Trish Meyer provide additional information and examples for these methods in an article on the ProVideo Coalition website.

Ian Haigh provides a script on After Effects Scripts website that you can use to easily apply advanced interpolation method expressions—such as bounces—to properties.

Andrew Devis provides a pair of video tutorials on the Creative COW website that show in detail how to use the linear expression method along with the Convert Audio To Keyframes command.

8.1 linear(t, tMin, tMax, value1, value2)

Description

Returns `value1` when `t <= tMin`. Returns `value2` when `t >= tMax`. Returns a linear interpolation between `value1` and `value2` when `tMin < t < tMax`.

For example, this expression on the Opacity property causes Opacity values to ramp linearly from `20%` to `80%` over the time from `0` seconds to `6` seconds:

```
linear(time, 0, 6, 20, 80)
```

This method—like all the Interpolation methods—can also be used to convert from one range of values to another.

For example, this expression on the Opacity property converts the Opacity values from the range `0%-100%` to the range `20%-80%`:

```
linear(value, 0, 100, 20, 80)
```

Parameters

t	Number
tMin	Number
tMax	Number
value1	Number or Array
value2	Number or Array

Type

Number or Array

8.2 linear(t, value1, value2)

Description

Returns a value that linearly interpolates from `value1` to `value2` as `t` ranges from `0` to `1`. Returns `value1` when `t <= 0`. Returns `value2` when `t >= 1`.

Parameters

t	Number
value1	Number or Array
value2	Number or Array

Type

Number or Array

8.3 ease(t, value1, value2)

Description

Similar to linear with the same arguments, except that the interpolation eases in and out so that the velocity is `0` at the start and end points. This method results in a smooth animation.

Parameters

t	Number
value1	Number or Array
value2	Number or Array

Type

Number or Array

8.4 ease(t, tMin, tMax, value1, value2)

Description

Similar to linear with the same arguments, except that the interpolation eases in and out so that the velocity is `0` at the start and end points. This method results in a smooth animation.

Parameters

`t`	Number
`tMin`	Number
`tMax`	Number
`value1`	Number or Array
`value2`	Number or Array

Type

Number or Array

8.5 easeIn(t, value1, value2)

Description

Similar to ease, except that the tangent is `0` only on the `value1` side and interpolation is `linear` on the `value2` side.

Parameters

`t`	Number
`value1`	Number or Array
`value2`	Number or Array

Type

Number or Array

8.6 easeIn(t, tMin, tMax, value1, value2)

Description

Similar to ease, except that the tangent is `0` only on the `tMin` side and interpolation is `linear` on the `tMax` side.

Parameters

`t`	Number
`tMin`	Number
`tMax`	Number
`value1`	Number or Array
`value2`	Number or Array

Type

Number or Array

8.7 easeOut(t, value1, value2)

Description

Similar to ease, except that the tangent is `0` only on the `value2` side and interpolation is `linear` on the `value1` side.

Parameters

`t`	Number
`value1`	Number or Array
`value2`	Number or Array

Type

Number or Array

8.8 easeOut(t, tMin, tMax, value1, value2)

Description

Similar to ease, except that the tangent is `0` only on the `tMax` side and interpolation is `linear` on the `tMin` side.

Parameters

`t`	Number
`tMin`	Number
`tMax`	Number
`value1`	Number or Array
`value2`	Number or Array

Type

Number or Array

CHAPTER 9

Color Conversion

Harry Frank provides a video tutorial on his graymachine website that shows how to use these color conversion methods to change the color of the waves produced by the Radio Waves effect.

9.1 rgbToHsl(rgbaArray)

Description

Converts a color in RGBA space to HSLA space. The input is an Array of normalized red, green, blue, and alpha channel values, all in the range of 0.0 to 1.0. The resulting value is an Array of hue, saturation, lightness, and alpha channel values, also in the range of 0.0 to 1.0.

Example:

```
rgbToHsl.effect("Change Color")("Color To Change")
```

Parameters

rgbaArray	Array (4-dimensional)

Type

Array (4-dimensional)

9.2 hslToRgb(hslaArray)

Description

Converts a color in HSLA space to RGBA space. This conversion is the opposite of the conversion performed by the rgbToHsl method.

Parameters

`hslaArray`	Array (4-dimensional)

Type

Array (4-dimensional)

CHAPTER 10

Other Math

10.1 degreesToRadians(`degrees`)

Description

Converts degrees to radians.

Parameters

`degrees`	Number

Type

Number

10.2 radiansToDegrees(`radians`)

Description

Converts radians to degrees.

Parameters

`radians`	Number

Type

Number

CHAPTER 11

Project

11.1 Project.fullPath

Description

The platform-specific absolute file path, including the project file name. If the project has not been saved, it returns an empty string.

Example:

```
thisProject.fullPath
```

Type

String

11.2 Project.bitsPerChannel

Description

The color depth of the project in bits per channel (bpc), as set in *Project Settings* > *Color Management* They are one of *8*, *16*, or *32*. Equivalent to the scripting project attribute *app.project.bitsPerChannel*.

Example:

```
thisProject.bitsPerChannel
```

Type

Number

11.3 Project.linearBlending

Description

The state of the Blend Colors Using 1.0 Gamma option in *Project Settings* > *Color Management*. Equivalent to the scripting project attribute *app.project.linearBlending*.

Example:

```
thisProject.linearBlending
```

Type

Boolean

CHAPTER 12

Comp

12.1 Comp.layer(index)

Description

Retrieves the layer by number (order in the Timeline panel).

Example:

```
thisComp.layer(3)
```

Parameters

index	Number

Type

Layer, Light, or Camera

12.2 Comp.layer(name)

Description

Retrieves the layer by name. Names are matched according to layer name, or source name if there is no layer name. If duplicate names exist, After Effects uses the first (topmost) one in the Timeline panel.

Example:

```
thisComp.layer("Solid 1")
```

Parameters

`name`	String

Type

Layer, Light, or Camera

12.3 Comp.layer(`otherLayer`, `relIndex`)

Description

Retrieves the layer that is relIndex layers above or below otherLayer. For example, `thisComp.layer(thisLayer, 1).active` returns true if the next layer down in the Timeline panel is active.

Parameters

`otherLayer`	Layer Object
`relIndex`	Number

Type

Layer, Light, or Camera

12.4 Comp.marker

Description

Note: You cannot access a composition marker by marker number. If you have a project created in a previous version of After Effects that uses composition marker numbers in expressions, you must change those calls to use marker.key(name) instead. Because the default name of a composition marker is a number, converting the reference to use the name is often just a matter of surrounding the number with quotation marks.

Type

MarkerProperty

12.5 Comp.marker.key(`index`)

Description

Returns the MarkerKey object of the marker with the specified index. The index refers to the order of the marker in composition time, not to the name of the marker.

For example, this expression returns the time of the first composition marker:

```
thisComp.marker.key(1).time
```

Parameters

index	Number

Type

MarkerKey

12.6 Comp.marker.key(name)

Description

Returns the MarkerKey object of the marker with the specified name. The name value is the name of the marker, as typed in the comment field in the marker dialog box, for example, marker.key("1"). For a composition marker, the default name is a number. If more than one marker in the composition has the same name, this method returns the marker that occurs first in time (in composition time). The value for a marker key is a String, not a Number.

For example, this expression returns the time of the composition marker with the name "0":

```
thisComp.marker.key("0").time
```

Parameters

name	String

Type

MarkerKey

12.7 Comp.marker.nearestKey(t)

Description

Returns the marker that is nearest in time to t.

For example, this expression returns the time of the composition marker nearest to the time of 1 second:

```
thisComp.marker.nearestKey(1).time
```

This expression returns the time of the composition marker nearest to the current time:

```
thisComp.marker.nearestKey(time).time
```

Parameters

t	Number

Type

MarkerKey

12.8 Comp.marker.numKeys

Description

Returns the total number of composition markers in the composition.

Type

Number

12.9 Comp.numLayers

Description

Returns the number of layers in the composition.

Type

Number

12.10 Comp.activeCamera

Description

Returns the Camera object for the camera through which the composition is rendered at the current frame. This camera is not necessarily the camera through which you are looking in the Composition panel.

Type

Camera

12.11 Comp.width

Description

Returns the composition width, in pixels.Apply the following expression to the Position property of a layer to center the layer in the composition frame: [thisComp.width/2, thisComp.height/2]

Type

Number

12.12 Comp.height

Description

Returns the composition height, in pixels.

Type

Number

12.13 Comp.duration

Description

Returns the composition duration, in seconds.

Type

Number

12.14 Comp.ntscDropFrame

Description

Returns true if the timecode is in drop-frame format.

Note: Available in After Effects CS5.5 and later.

Type

Boolean

12.15 Comp.displayStartTime

Description

Returns the composition start time, in seconds.

Type

Number

12.16 Comp.frameDuration

Description

Returns the duration of a frame, in seconds.

Type

Number

12.17 Comp.shutterAngle

Description

Returns the shutter-angle value of the composition, in degrees.

Type

Number

12.18 Comp.shutterPhase

Description

Returns the shutter phase of the composition, in degrees.

Type

Number

12.19 Comp.bgColor

Description

Returns the background color of the composition.

Type

Array (4-dimensional)

12.20 Comp.pixelAspect

Description

Returns the pixel aspect ratio of the composition.

Type

Number

12.21 Comp.name

Description

Returns the name of the composition.

Type

String

CHAPTER 13

Footage

Description

To use a footage item from the Project panel as an object in an expression, use the global footage method, as in `footage("file_name")`. You can also access a footage object using the source attribute on a layer whose source is a footage item.

13.1 Footage.width

Description

Returns the width of the footage item, in pixels.

Type

Number

13.2 Footage.height

Description

Returns the height of the footage item, in pixels.

Type

Number

13.3 Footage.duration

Description

Returns the duration of the footage item, in seconds.

Type

Number

13.4 Footage.frameDuration

Description

Returns the duration of a frame in the footage item, in seconds.

Type

Number

13.5 Footage.ntscDropFrame

Description

Returns true if the timecode is in drop-frame format. (After Effects CS5.5 and later.)

Type

Boolean

13.6 Footage.pixelAspect

Description

Returns the pixel aspect ratio of the footage item.

Type

Number

13.7 Footage.name

Description

Returns the name of the footage item as shown in the Project panel.

Type

String

CHAPTER 14

Camera

Camera objects have the same attributes and methods as Layer objects, except for:

- `source`
- `effect`
- `mask`
- `width`
- `height`
- `anchorPoint`
- `scale`
- `opacity`
- `audioLevels`
- `timeRemap`
- all the material properties

14.1 Camera.pointOfInterest

Description

Returns the point of interest values of a camera in world space.

Type

Array (3 dimensional)

14.2 Camera.zoom

Description

Returns the zoom values of a camera in pixels.

Here's an expression for the Scale property of a layer that maintains the relative size of the layer in frame while changing the z position (depth) of a layer or the Zoom value of a camera:

```
cam = thisComp.activeCamera;
distance = length(sub(position, cam.position));
scale * distance / cam.zoom;
```

Type

Number

14.3 Camera.depthOfField

Description

Returns `1` if the Depth Of Field property of a camera is on, or returns `0` if the Depth Of Field property is off.

Type

Boolean Number

14.4 Camera.focusDistance

Description

Returns the focus distance value of a camera, in pixels.

Type

Number

14.5 Camera.aperture

Description

Returns the aperture value of a camera, in pixels.

Type

Number

14.6 Camera.blurLevel

Description

Returns the blur level value of a camera as a percentage.

Type

Number

14.7 Camera.active

Description

Returns `true` if the camera:

1. is the active camera for the composition at the current time: the *video switch* for the camera layer is on
2. the current time is in the range from the *in point* of the camera layer to the *out point* of the camera layer
3. and it is the first (topmost) such camera layer listed in the *timeline panel*

Returns `false` otherwise.

Type

Boolean

CHAPTER 15

Light

Description

Light objects have the same attributes and methods as Layer objects, except for:

- `source`
- `effect`
- `mask`
- `width`
- `height`
- `anchorPoint`
- `scale`
- `opacity`
- `audioLevels`
- `timeRemap`
- all the material properties

Note: David Van Brink provides an instructional article and sample project on his omino pixel blog that shows how to use expressions with lights.

15.1 Light.pointOfInterest

Description

Returns the point of interest values for a light in world space.

Type

Array (3-dimensional)

15.2 Light.intensity

Description

Returns the intensity values of a light as a percentage.

Type

Number

15.3 Light.color

Description

Returns the color value of a light.

Type

Array (4-dimensional)

15.4 Light.coneAngle

Description

Returns the cone angle of a light, in degrees.

Type

Number

15.5 Light.coneFeather

Description

Returns the cone feather value of a light as a percentage.

Type

Number

15.6 Light.shadowDarkness

Description

Returns the shadow darkness value of a light as a percentage.

Type

Number

15.7 Light.shadowDiffusion

Description

Returns the shadow diffusion value of a light, in pixels.

Type

Number

CHAPTER 16

Effect

16.1 Effect.active

Description

Returns `true` if the effect is turned on (the *effect switch* is selected).

Type

Boolean

16.2 Effect.param(`name`)

Description

Returns a property within an effect. Effect control points are always in layer space.

Example:

```
effect("Bulge").param("Bulge Height")
```

Parameters

`name`	String

Type

Property

16.3 Effect.param(index)

Description

Returns a property within an effect. Effect control points are always in layer space. For example, effect("Bulge").param(4) returns the Bulge Height property.

Parameters

index	Number

Type

Property

CHAPTER 17

Mask

Note: You can link Mask Path properties to other path properties (paths in a shape layer and brush strokes), but the properties are not accessible for direct numerical manipulation through expressions.

17.1 Mask.maskOpacity

Description

Returns the opacity value of a mask as a percentage.

Type

Number

17.2 Mask.maskFeather

Description

Returns the feather value of a mask, in pixels.

Type

Number

17.3 Mask.maskExpansion

Description

Returns the expansion value of a mask, in pixels.

Type

Number

17.4 Mask.invert

Description

Returns `true` if the mask is inverted or `false` if it is not.

Type

Boolean

CHAPTER 18

Property

Example: Animating with the propertyGroup method and propertyIndex attribute

[image]

In this example, the propertyGroup method for each brush stroke targets the Brush property group because that group is two property groups up from the Rotation property. The propertyIndex attribute in each Brush stroke then returns a unique value for each Brush stroke. The resulting value is then multiplied by the `time` and `200` and applied to each rotation value, rotating each brush stroke differently, creating swirling paint strokes:

```
propertyGroup(2).propertyIndex * time * 200
propertyGroup(2).propertyIndex * time * 200
```

[image]

18.1 value

Description

Returns the value of a property at the current time.

Type

Number, Array, or String

18.2 valueAtTime(t)

Description

Returns the value of a property at the specified time, in seconds.

For example, to have a property value for each frame chosen randomly from a set of four values, set your four values as keyframes at `0`, `1`, `2`, and `3` seconds, and then apply the following expression to the property:

```
valueAtTime(random(4))
```

Note: Dan Ebberts provides more examples and techniques for using the `valueAtTime` and `velocityAtTime` methods on his MotionScript website.

Parameters

`t`	Number

Type

Number or Array

18.3 velocity

Description

Returns the temporal velocity value at the current time. For spatial properties, such as Position, it returns the tangent vector value. The result is the same dimension as the property.

Type

Number or Array

18.4 velocityAtTime(t)

Description

Returns the temporal velocity value at the specified time.

Parameters

`t`	Number

Type

Number or Array

18.5 speed

Description

Returns a 1D, positive speed value equal to the speed at which the property is changing at the default time. This element can be used only for spatial properties.

Type

Number

18.6 speedAtTime(t)

Description

Returns the spatial speed value at the specified time.

Parameters

t	Number

Type

Number

18.7 wiggle(freq, amp, octaves=1, amp_mult=0.5, t=time)

Description

Randomly shakes (wiggles) the value of the property.

`freq` value is the frequency in wiggles per second.

`amp` value is the amplitude in units of the property to which it is applied.

`octaves` is the number of octaves of noise to add together. This value controls how much detail is in the wiggle. Make this value higher than the default of 1 to include higher frequencies or lower to include amplitude harmonics in the wiggle.

`amp_mult` is the amount that amp is multiplied by for each octave. This value controls how fast the harmonics drop off. The default is `0.5`; make it closer to `1` to have the harmonics added at the same amplitude as the base frequency, or closer to `0` to add in less detail.

`t` is the base start time. This value defaults to the current time. Use this parameter if you want the output to be a wiggle of the property value sampled at a different time.

Example:

```
position.wiggle(5, 20, 3, 0.5)
```

This produces about `5` wiggles per second with an average size of about 20 pixels. In addition to the main wiggle, two more levels of detailed wiggles occur with a frequency of `10` and `20` wiggles per second, and sizes of `10` and `5` pixels, respectively.

This example, on a two-dimensional property such as Scale, wiggles both dimensions by the same amount:

```
v = wiggle(5, 10);
[v[0], v[0]]
```

This example, on a two-dimensional property, wiggles only along the y-axis:

```
freq = 3;
amp = 50;
w = wiggle(freq,amp);
[value[0],w[1]];
```

Note: Dan Ebberts provides an example expression and a detailed explanation on his MotionScript website that shows how to use the time parameter of the wiggle method to create a looping animation.

Parameters

freq	Number
amp	Number
octaves	Number
amp_mult	Number
t	Number

Type

Number or Array

18.8 temporalWiggle(freq, amp, octaves=1, amp_mult=0.5, t=time)

Description

Samples the property at a wiggled time.

`freq` value is the frequency in wiggles per second.

`amp` is the amplitude in units of the property to which it is applied.

`octaves` is the number of octaves of noise to add together.

`amp_mult` is the amount that amp is multiplied by for each octave

`t` is the base start time.

For this function to be meaningful, the property it samples must be animated, because the function alters only the time of sampling, not the value.

Example:

```
scale.temporalWiggle(5, 0.2)
```

Parameters

freq	Number
amp	Number
octaves	Number
amp_mult	Number
t	Number

Type

Number or Array

18.9 smooth(width=.2, samples=5, t=time)

Description

Smooths the property values over time, converting large, brief deviations in the value to smaller, more evenly distributed deviations. This smoothing is accomplished by applying a box filter to the value of the property at the specified time. The width value is the range of time (in seconds) over which the filter is averaged. The samples value is the number of discrete samples evenly spaced over time; use a larger value for greater smoothness (but decreased performance). Generally, you'll want samples to be an odd number so that the value at the current time is included in the average.

Example:

```
position.smooth(0.1, 5)
```

Parameters

width	Number
samples	Number
t	Number

Type

Number or Array

18.10 loopIn(type="cycle", numKeyframes=0)

Description

Loops a segment of time that is measured from the first keyframe on the layer forward toward the Out point of the layer. The loop plays from the In point of the layer. The numKeyframes value determines what segment is looped: The segment looped is the portion of the layer from the first keyframe to the numKeyframes+1 keyframe. For example, loopIn("cycle", 3) loops the segment bounded by the first and fourth keyframes. The default value of 0 means that all keyframes loop. You can use keyframe-looping methods to repeat a series of keyframes. You can use these methods on most properties. Exceptions include properties that can't be expressed by simple numeric values in the Timeline panel, such as the Source Text property, path shape properties, and the Histogram property for the Levels effect. Keyframes or duration values that are too large are clipped to the maximum allowable value. Values that are too small result in a constant loop.

type	result
cycle	(default) Repeats the specified segment.
pingpong	Repeats the specified segment, alternating between forward and backward.
offset	Repeats the specified segment, but offsets each cycle by the difference in the value of the property at the start and end of the segment, multiplied by the number of times the segment has looped.
continue	Does not repeat the specified segment, but continues to animate a property based on the velocity at the first or last keyframe. For example, if the last keyframe of a Scale property of a layer is `100%`, the layer continues to scale from `100%` to the Out point, instead of looping directly back to the Out point. This type does not accept a keyframes or duration argument.

Tip: Use `loopIn("continue") + loopOut("continue") - value` to have a continued motion before and after the property's keyframes. *Tip from Paul Slemmer.*

Type

Number or Array

18.11 loopOut(type="cycle", numKeyframes=0)

Description

Loops a segment of time that is measured from the last keyframe on the layer back toward the In point of the layer. The loop plays until the Out point of the layer. The specified number of keyframes determines the segment to loop. The numKeyframes value sets the number of keyframe segments to loop; the specified range is measured backward from the last keyframe.

For example, `loopOut("cycle", 1)` loops the segment bounded by the last keyframe and second-to-last keyframe. The default value of 0 means that all keyframes loop. See the entry for loopIn for more information.

Note: David Van Brink provides an instructional article and sample project on his omino pixel blog that show how to use the Echo effect, the Particle Playground effect, and the `loopOut` method to animate a swarm of stylized swimming bacteria.

Type

Number or Array

18.12 loopInDuration(type="cycle", duration=0)

Description

Loops a segment of time that is measured from the first keyframe on the layer forward toward the Out point of the layer. The loop plays from the In point of the layer. Specified duration determines the segment to loop. The duration

value sets the number of composition seconds in a segment to loop; the specified range is measured from the first keyframe.

For example, `loopInDuration("cycle", 1)` loops the first second of the entire animation. The default of `0` means that the segment to loop begins at the layer Out point. See the entry for loopIn for more information.

Type

Number or Array

18.13 loopOutDuration(type="cycle", duration=0)

Description

Loops a segment of time that is measured from the last keyframe on the layer back toward the In point of the layer. The loop plays until the Out point of the layer. Specified duration determines the segmetn to loop. The duration value sets the number of composition seconds in a segment to loop; the specified range is measured backward from the last keyframe.

For example, `loopOutDuration("cycle", 1)` loops the last second of the entire animation. The default of `0` means that the segment to loop begins at the layer In point. See the entry for loopIn for more information.

Type

Number or Array

18.14 key(index)

Description

Returns the Key or MarkerKey object by number.

For example, `key(1)` returns the first keyframe.

Parameters

index	Number

Type

Key or MarkerKey

18.15 key(markerName)

Description

Returns the MarkerKey object with this name. Use only on marker properties.

markerName	String

Type

MarkerKey

18.16 nearestKey(t)

Description

Returns the Key or MarkerKey object nearest to a designated time `t`.

Parameters

`t`	Number

Type

Key or MarkerKey

18.17 numKeys

Description

Returns the number of keyframes on a property. Returns the number of markers on a marker property.

Note: If you use the Separate Dimensions command to separate the dimensions of the Position property into individual components, the number of keyframes changes, so the value returned by this method changes.

Type

Number

18.18 propertyGroup(countUp=1)

Description

Returns a group of properties relative to the property on which the expression is written.

For example, if you add the `propertyGroup(1)` expression to the Rotation property of a brush stroke, the expression targets the Transform property group, which contains the Rotation property. If you add `propertyGroup(2)` instead, the expression targets the Brush property group.

This method lets you establish name-independent relationships in the property hierarchy. It is especially useful when duplicating properties that contain expressions.The `numProperties` method for `propertyGroup` returns the number of properties in the property group.

This example returns the number of properties in the group that contains the property on which the expression is written:

```
thisProperty.propertyGroup(1).numProperties
```

Type

Group

18.19 propertyIndex

Description

Returns the index of a property relative to other properties in its property group, including property groups within masks, effects, text animators, selectors, shapes, trackers, and track points.

Type

Number

18.20 name

Description

Returns the name of the property or property group.

Type

String

CHAPTER 19

Path Property

Note: Available in After Effects CC18 and later.

Example 1

Writes the list of point and tangent coordinates from *Path 1* of *Shape 1* on layer *Shape Layer 1*, at `time=0`, into a string. Apply this to the source text property of a text layer for a readout of the coordinates and incoming and outgoing tangents of the shape.

```
pointsList = "";
sampleTime = 0;
myShape = thisComp.layer("Shape Layer 1").content("Shape 1").content("Path 1").path;

for (i = 0; i < myShape.points(sampleTime).length; i++) {
        pointsList += "c: " + myShape.points(sampleTime)[i].toString() + " i: " +␣
↪myShape.inTangents(sampleTime)[i].toString() + " o: " + myShape.
↪outTangents(sampleTime)[i].toString() + "\n";
}

pointsList;
```

Example 2

Reads the coordinates of the first vertex of Mask 1 on Dark Gray Solid 1 and converts them to composition coordinates. Apply this to a 2D point control of an effect, such as Write-on or CC Particle Systems II, to make the effect trace or track the first point of an animated mask. Duplicate the effect and change the path points index value ([0]) to trace or track the other points of the mask.

```
myLayer = thisComp.layer("Dark Gray Solid 1");
myLayer.toComp(myLayer.mask("Mask 1").maskPath.points()[0]);
```

19.1 name

Description

Returns the name of the property.

Type

String

19.2 pathProperty.points(t=time)

Description

Get the x,y coordinates of all points on a path. Coordinates for layer mask path points are relative to the layer's origin in its upper-left hand corner. Coordinates for Bezier shape path points are are relative to the anchor point of the path's shape group (ex., "Transform: Shape 1 > Anchor Point"). Coordinates for brush stroke path points are relative to the start of the stroke; the first point is `[0,0]`. This method can be passed into the `createPath()` method for the points parameter when duplicating a path.

Optionally specify the time at which sample to the path.

Parameters

`t`	Number

Type

Array of number pair arrays, rounded to the fourth decimal place

---

19.3 pathProperty.inTangents(t=time)

Description

Get the x,y coordinates of the incoming tangent handle for all points on a path. Tangent coordinate values are offset relative to the parent point's coordinates. i.e., The value [0,0] creates no curvature at the incoming tangent. This method can be passed into the createPath() method for the inTangents parameter when duplicating a path.

Optionally specify the time at which sample to the path.

Type

Array of number pair arrays, rounded to the fourth decimal place

---

19.4 pathProperty.outTangents(t=time)

Description

Get the x, y coordinates of the outgoing tangent handle for all points on a path. Tangent coordinate values are offset relative to the parent point's coordinates. i.e., The value `[0,0]` creates no curvature at the outgoing tangent. This method can be passed into the createPath() method for the outTangents parameter when duplicating a path.

Optionally specify the time at which sample to the path.

Parameters

t	Number

Type

Array of number pair arrays, rounded to the fourth decimal place

---

19.5 pathProperty.isClosed()

Description

Determines if the path is open or closed. Returns `true` if the path is closed, `false` if the path is open. This method can be passed into the `createPath()` method for the `is_closed` parameter when duplicating a path.

Type

Boolean

19.6 pathProperty.pointOnPath(percentage=0.5, t=time)

Description

Get the x, y coordinates of an arbitrary point along a path. The point is expressed as a percentage of the arc-length of the path. 0% is the first point and 100% is the last point. When the path is closed, 0% and 100% will return the same coordinates. Percentage of arc-length is used to ensure uniform speed along the path. Other than 0% and 100%, percentages do not necessarily correlate with the Bezier points on the path. (i.e., For a path with three points, the second point will not necessarily be at 50%.) This also means that for an open path and closed path with identical points, the percentage along the open path will not return the same coordinates as the closed path due to the additional length of the closed path.

Optionally specify the time at which sample to the path.

Parameters

percentage	Number
t	Number

Type

A number pair array

19.7 pathProperty.tangentOnPath(percentage=0.5, t=time)

Description

Get the calculated x,y coordinates of the outgoing tangent handle for an arbitrary point along a path. Tangent coordinate values are offset relative to the parent point's coordinates. i.e., The value [0,0] creates no curvature at the outgoing tangent. The incoming tangent handle is the inverse of this value (multiply the x,y coordinates by -1). The tangent's parent point is expressed as a percentage of the arc-length of the path. Read the description of the pointOnPath() method for details about arc-length percentage. The coordinates returned by tangentOnPath() are calcuated from it's parent point and will differ from those returned by `outTangents()` if a user-defined point also exists at that arc-length pecentage. The linear distance between the parent point's coordinates and `tangentOnPath()` coordinates will always be 1. You can multiply the returned coordinates to create a longer tangent, for example `(myPath.tangentOnPath() * 100)`.

Optionally specify the time at which sample to the path.

Parameters

`percentage`	Number
`t`	Number

Type

A number pair array

19.8 pathProperty.normalOnPath(percentage=0.5, t=time)

Description

Get the calculated x,y coordinates of the normal for an arbitrary point along a path. Coordinate values of normals are offset relative to the parent point's coordinates. i.e., The value `[0,0]` is the same as the parent point. The normal's parent point is expressed as a percentage of the arc-length of the path. Read the description of the `pointOnPath()` method for details about arc-length percentage. The coordinates returned by `normalOnPath()` are calcuated from its parent point. The linear distance between the parent point's coordinates and `normalOnPath()` coordinates will always be `1`. You can multiply the returned coordinates to create a longer normal, for example. `(myPath.normalOnPath() * 100)`.

Optionally specify the time at which sample to the path.

Parameters

`percentage`	Number
`t`	Number

Type

A number pair array

19.9 pathProperty.createPath(points=[[0,0], [100,0], [100,100], [0,100]], inTangents=[], outTangents=[], is_closed=true)

Description

Creates a path object from a set of points and tangents. The points are defined by an array of number pair arrays representing their `x`, `y` coordinates. The array length must be at least `1`, and can be of any greater length. The incoming and outgoing tangent handles of the points are defined by an array of number pair arrays representing their x, y offset coordinates. The length of the tangent arrays must be exactly the same as the points parameter. Tangent coordinate values are offset relative to the parent point's coordinates. i.e., The value `[0,0]` creates no curvature at the incoming tangent. The `points()`, `inTangents()`, `outTangents()`, and `isClosed()` methods of a path can be passed into the `points`, `inTangents`, `outTangents`, and `is_closed` parameters to duplicate a path. The points and tangents of the same path can be passed into `createPath()` with modifications to generate a different result.

For example, the following expression will remove curves from Mask 1 by not passing the inTangents or outTangents parameters:

```
myMask = mask("Mask 1").path;
myMask.createPath(myMask.points());
```

The following example passes the points and tangents of *Mask 1* and converts it to an open path by setting `is_closed` to false:

```
myMask = mask("Mask 1").path;
myMask.createPath(myMask.points(), myMask.inTangents(), myMask.outTangents(), false);
```

Parameters

`points`	Array
`inTangents`	Array
`outTangents`	Array
`is_closed`	Boolean

Type

Path

CHAPTER 20

Key

Description

When you access a Key object, you can get time, index, and value properties from it. For example, the following expression gives you the value of the third Position keyframe: position.key(3).value.

The following expression, when written on an Opacity property with keyframes, ignores the keyframe values and uses only the placement of the keyframes in time to determine where a flash should occur:

```
d = Math.abs(time - nearestKey(time).time);
easeOut(d, 0, .1, 100, 0)
```

20.1 Key.value

Description

Returns the value of the keyframe.

Type

Number or Array

20.2 Key.time

Description

Returns the time of the keyframe.

Type

Number

20.3 Key.index

Description

Returns the index of the keyframe.

Type

Number

CHAPTER 21

MarkerKey

You can access values for composition markers and layer markers using the same methods. Access layer markers through the thisLayer.marker object; access composition markers through the thisComp.marker object.

For the purpose of expressions, markers are a special type of Key object, so you can use methods such as nearestKey(time) to access markers, and markers also have time and index attributes. The index attribute is not the number (name) of the marker; it is the keyframe index number, representing the order of the marker in the time ruler.

Expressions have access to all the values for a marker that you can set in the Composition Marker or Layer Marker dialog box.

This expression on the Source Text property of a text layer displays the time, duration, index, comment (name), chapter, URL, frame target, and cue point name for the layer marker nearest the current time, and whether the marker is for an event cue point:

```
m = thisLayer.marker.nearestKey(time);
s = "time:" + timeToCurrentFormat(m.time) + "\r" +
"duration: " + m.duration + "\r" +
"key index: " + m.index + "\r" +
"comment:" + m.comment + "\r" +
"chapter:" + m.chapter + "\r" +
"URL:" + m.url + "\r" +
"frame target: " + m.frameTarget + "\r" +
"cue point name: " + m.cuePointName + "\r" +
"Event cue point? " + m.eventCuePoint + "\r";
for (param in m.parameters){
s += "parameter: " + param + " value: " + m.parameters[param] + "\r";
}
s
```

Because the XMP metadata in a footage item can be converted into layer markers for a layer based on that item, expressions can interact with XMP metadata. For information, see XMP metadata in After Effects.

Dan Ebberts provides a tutorial on the After Effects Developer Center that includes an example of using XMP metadata with expressions.

21.1 Marker.duration

Description

Duration, in seconds, of marker.

Type

Number

21.2 Marker.comment

Description

Contents of Comment field in marker dialog box.

Type

String

21.3 Marker.chapter

Description

Contents of Chapter field in marker dialog box.

Type

String

21.4 Marker.url

Description

Contents of URL field in marker dialog box.

Type

String

21.5 Marker.frameTarget

Description

Contents of Frame Target field in marker dialog box.

Type

String

21.6 Marker.eventCuePoint

Description

Setting for cue point type in marker dialog box. True for Event; false for Navigation.

Type

Boolean

21.7 Marker.cuePointName

Description

Contents of cue point Name field in marker dialog box.

Type

String

21.8 Marker.parameters

Description

Contents of Parameter Name and Parameter Value fields in marker dialog box.

For example, if you have a parameter named "background color", then you can use the following expression to access its value at the nearest marker:

```
thisComp.marker.nearestKey(time).parameters["background color"]
```

Type

Associative array of String values

CHAPTER 22

Layer Sub-objects

Note: For After Effects CC and CS6, the Expression language menu, the "Layer Sub-objects", "Layer General", "Layer Properties", "Layer 3D", and "Layer Space Transforms" have been arranged into a "Layer" submenu.

22.1 Layer.source

Description

Returns the source Comp or source Footage object for the layer. Default time is adjusted to the time in the source.

Example:

```
source.layer(1).position
```

Type

Comp or Footage

22.2 Layer.sourceTime(t=time)

Description

Returns the layer source corresponding to time `t`.

Note: After Effects CS5.5 and later

Parameters

t	Number

Type

Number

22.3 Layer.sourceRectAtTime(t = time, includeExtents = false)

Description

Returns a JavaScript object with four attributes: `[top, left, width, height]`

Extents apply only to shape layers and paragraph text layers.

Shape layer extents increase the size of the layer bounds as necessary.

Paragraph text layers returns the bounds of the paragraph box.

Note: After Effects 13.2 and later. Paragraph text extents added in After Effects 15.1.

Example:

```
myTextLayer.sourceRectAtTime().width
```

Parameters

t	Number
includeExtents	Bool

Type

Array (4-dimensional)

22.4 Layer.effect(name)

Description

After Effects finds the effect by its name in the Effect Controls panel. The name can be the default name or a user-defined name. If multiple effects have the same name, the effect closest to the top of the Effect Controls panel is used.

Example:

```
effect("Fast Blur")("Blurriness")
```

Parameters

`name`	String

Type

Effect

22.5 Layer.effect(`index`)

Description

After Effects finds the effect by its index in the Effect Controls panel, starting at `1` and counting from the top.

Parameters

`index`	Number

Type

Effect

22.6 Layer.mask(`name`)

Description

The name can be the default name or a user-defined name. If multiple masks have the same name, the first (topmost) mask is used.

Example:

```
mask("Mask 1")
```

Parameters

`name`	String

Type

Mask

22.7 Layer.mask(index)

Description

After Effects finds the mask by its index in the Timeline panel, starting at `1` and counting from the top.

Parameters

`index`	Number

Type

Mask

CHAPTER 23

Layer General

23.1 Layer.width

Description

Returns the width of the layer, in pixels. It is the same as `source.width`.

Type

Number

23.2 Layer.height

Description

Returns the height of the layer, in pixels. It is the same as `source.height`.

Type

Number

23.3 Layer.index

Description

Returns the index number of the layer in the composition.

Type

Number

23.4 Layer.parent

Description

Returns the parent Layer object of the layer, if it has one.

Example:

```
position[0] + parent.width
```

Type

Layer, Light, or Camera

23.5 Layer.hasParent

Description

Returns true if the layer has a parent or false if it doesn't. Use the hasParent attribute to determine if a layer has a parent layer. You can use this attribute even if the layer has no parent layer at present. For example, the following expression indicates that the layer to which you apply it wiggles based on the position of the parent. If the layer has no parent, then it wiggles based on its own position.

If the layer is given a parent later, then the behavior of the layer changes accordingly:

```
idx = index;
if (hasParent) {
        idx = parent.index;
}
thisComp.layer(idx).position.wiggle(5,20)
```

Type

Boolean

23.6 Layer.inPoint

Description

Returns the In point of the layer, in seconds.

Note: In general, the value of outPoint is greater than the value of inPoint. However, if a layer is reversed in time, the value of inPoint is greater than the value of outPoint. Similarly, the value of startTime can be greater than the value of inPoint.

Type

Number

23.7 Layer.outPoint

Description

Returns the Out point of the layer, in seconds.

Type

Number

23.8 Layer.startTime

Description

Returns the start time of the layer, in seconds.

Type

Number

23.9 Layer.hasVideo

Description

Returns `true` if the layer has video, or `false` if it doesn't.

Type

Boolean

23.10 Layer.hasAudio

Description

Returns `true` if the layer has audio or `false` if it doesn't.

Type

Boolean

23.11 Layer.active

Description

Returns `true` if the Video switch is on for the layer and the current time is in the range from the In point of the layer to the Out point of the layer; `false` otherwise.

Type

Boolean

23.12 Layer.enabled

Description

Returns true if the Video switch is on for the layer; false otherwise.

Type

Boolean

23.13 Layer.audioActive

Description

Returns true if the Audio switch is on for the layer and the current time is in the range from the In point of the layer to the Out point of the layer; false otherwise.

Type

Boolean

23.14 Layer.sampleImage(point, radius=[0.5, 0.5], postEffect=true, t=time)

Description

Samples the color and alpha channel values of a layer and returns the average alpha-weighted value of the pixels within the specified distance of the point as an array: `[red, green, blue, alpha]`.

If `postEffect` is `true`, the sampled values are for the layer after masks and effects on that layer have been rendered; if `postEffect` is `false`, the sampled values are for the layer before masks and effects have been rendered. The input value point is in layer space; the point `[0, 0]` is the center of the upper-left pixel in the layer. The input value `radius` specifies the horizontal and vertical distance from the sample center to the edges of the sampled rectangle. The default value samples one pixel.

This example samples a rectangle 4 pixels wide and 3 pixels high, centered around a point 100 pixels down and to the right of the upper-left corner of the layer:

```
thisComp.layer(1).sampleImage([100, 100], [2, 1.5])
```

Note: The postEffect parameter refers to effects applied directly to the layer, not to effects applied indirectly, such as with an adjustment layer.

Note: Using sampleImage in an expression no longer disables multiprocessing.

Note: Dan Ebberts provides an example of how to use the sampleImage method on his MotionScript website.

Note: Todd Kopriva provides instructions for using the sampleImage method and the Point Control effect to monitor colors for a specified point during color correction on his After Effects Region of Interest blog.

Parameters

Argument type: point is an Array [2], radius is an Array [2], postEffect is a Boolean, and t is a Number.

Type

Array (4-dimensional)

CHAPTER 24

Layer Properties

Description

When you add masks, effects, paint, or text to a layer, After Effects adds new properties to the Timeline panel. There are too many of these properties to list here, so use the pick whip to learn the syntax for referring to them in your expressions.

24.1 Layer.anchorPoint

Description

Returns the anchor point value of the layer in the coordinate system of the layer (layer space).

Type

Array of Numbers (2- or 3-dimensional)

24.2 Layer.position

Description

Returns the position value of the layer, in world space if the layer has no parent. If the layer has a parent, it returns the position value of the layer in the coordinate system of the parent layer (in the layer space of the parent layer).

Type

Array of Numbers (2- or 3-dimensional)

24.3 Layer.scale

Description

Returns the scale value of the layer, expressed as a percentage.

Type

Array of Numbers (2- or 3-dimensional)

24.4 Layer.rotation

Description

Returns the rotation value of the layer in degrees. For a 3D layer, it returns the z rotation value in degrees.

Type

Number

24.5 Layer.opacity

Description

Returns the opacity value for the layer, expressed as a percentage.

Type

Number

24.6 Layer.audioLevels

Description

Returns the value of the Audio Levels property of the layer, in decibels. This value is a 2D value; the first value represents the left audio channel, and the second value represents the right. The value is not the amplitude of the audio track of the source material. Instead, it is the value of the Audio Levels property, which may be affected by keyframes.

Type

Array of Numbers (2-dimensional)

24.7 Layer.timeRemap

Description

Returns the value of the Time Remap property, in seconds, if Time Remap is enabled.

Type

Number

24.8 Layer.marker.key(index)

Description

Returns the MarkerKey object of the layer marker with the specified index.

Parameters

index	Number

Type

MarkerKey

24.9 Layer.marker.key(name)

Description

Returns the MarkerKey object of the layer marker with the specified name. The name value is the name of the marker, as typed in the comment field in the marker dialog box, for example, `marker.key("ch1")`. If more than one marker on the layer has the same name, this method returns the marker that occurs first in time (in layer time). The value for a marker key is a `String`, not a `Number`.

This expression on a property ramps the value of the property from `0` to `100` between two markers identified by name:

```
m1 = marker.key("Start").time;
m2 = marker.key("End").time;
linear(time, m1, m2, 0, 100);
```

Parameters

name	String

Type

MarkerKey

24.10 Layer.marker.nearestKey(t)

Description

Returns the layer marker that is nearest in time to t.

For example, this expression returns the time of the marker on the layer nearest to the time of 1 second:

```
marker.nearestKey(1).time
```

This expression returns the time of the marker on the layer nearest to the current time:

```
marker.nearestKey(time).time
```

Parameters

t	Number

Type

MarkerKey

24.11 Layer.marker.numKeys

Description

Returns the total number of markers on the layer.

Type

Number

24.12 Layer.name

Description

Returns the name of the layer.

Type

String

CHAPTER 25

Layer Space Transforms

Description

Use layer space transform methods to transform values from one space to another, such as from layer space to world space. The `from` methods transform values from the named space (composition or world) to the layer space.

The `to` methods transform values from the layer space to the named space (composition or world). Each transform method takes an optional argument to determine the time at which the transform is computed; however, you can almost always use the current (default) time.

Use `Vec` transform methods when transforming a direction vector, such as the difference between two position values.

Use the plain (non-`Vec`) transform methods when transforming a point, such as position.

Composition (comp) and world space are the same for 2D layers. For 3D layers, however, composition space is relative to the active camera, and world space is independent of the camera.

25.1 toComp(point, t=time)

Description

Transforms a point from layer space to composition space.

Parameters

`point`	Array (2- or 3-dimensional)
`t`	Number

Type

Array (2- or 3-dimensional)

25.2 fromComp(point, t=time)

Description

Transforms a point from composition space to layer space. The resulting point in a 3D layer may have a nonzero value even though it is in layer space.

Example:

```
fromComp(thisComp.layer(2).position)
```

Parameters

point	Array (2- or 3-dimensional)
t	Number

Type

Array (2- or 3-dimensional)

25.3 toWorld(point, t=time)

Description

Transforms a point from layer space to view-independent world space.

Example:

```
toWorld.effect("Bulge")("Bulge Center")
```

..note:: Dan Ebberts provides an expression on his MotionScript website that uses the `toWorld` method to auto-orient a layer along only one axis. This is useful, for example, for having characters turn from side to side to follow the camera while remaining upright.

..note:: Rich Young provides a set of expressions on his AE Portal website that use the toWorld method link a camera and light to a layer with the CC Sphere effect.

Parameters

point	Array (2- or 3-dimensional)
t	Number

Type

Array (2- or 3-dimensional)

25.4 fromWorld(point, t=time)

Description

Transforms a point from world space to layer space.

Example:

```
fromWorld(thisComp.layer(2).position)
```

See Expression example: Create a bulge between two layers for an example of how this method can be used.

Parameters

point	Array (2- or 3-dimensional)
t	Number

Type

Array (2- or 3-dimensional)

25.5 toCompVec(vec, t=time)

Description

Transforms a vector from layer space to composition space.

Example:

```
toCompVec([1,0])
```

Parameters

vec	Array (2- or 3-dimensional)
t	Number

Type

Array (2- or 3-dimensional)

25.6 fromCompVec(vec, t=time)

Description

Transforms a vector from composition space to layer space.

Example (2D layer):

```
dir = sub(position, thisComp.layer(2).position);
fromCompVec(dir)
```

Parameters

vec	Array (2- or 3-dimensional)
t	Number

Type

Array (2- or 3-dimensional)

25.7 toWorldVec(vec, t=time)

Description

Transforms a vector from layer space to world space.

Example:

```
p1 = effect("Eye Bulge 1")("Bulge Center");
p2 = effect("Eye Bulge 2")("Bulge Center");
toWorld(sub(p1, p2))
```

Parameters

vec	Array (2- or 3-dimensional)
t	Number

Type

Array (2- or 3-dimensional)

25.8 fromWorldVec(vec, t=time)

Description

Transforms a vector from world space to layer space.

Example:

```
fromWorld(thisComp.layer(2).position)
```

Parameters

vec	Array (2- or 3-dimensional)
t	Number

Type

Array (2- or 3-dimensional)

25.9 fromCompToSurface(point, t=time)

Description

Projects a point located in composition space to a point on the surface of the layer (zero z-value) at the location where it appears when viewed from the active camera. This method is useful for setting effect control points. Use with 3D layers only.

Parameters

point	Array (2- or 3-dimensional)
t	Number

Type

Array (2-dimensional)

CHAPTER 26

Layer 3D

26.1 Layer.orientation

Description

Returns the 3D orientation value, in degrees, for a 3D layer.

Type

Array (3-dimensional)

26.2 Layer.rotationX

Description

Returns the x rotation value, in degrees, for a 3D layer.

Type

Number

26.3 Layer.rotationY

Description

Returns the y rotation value, in degrees, for a 3D layer.

Type

Number

26.4 Layer.rotationZ

Description

Returns the z rotation value, in degrees, for a 3D layer.

Type

Number

26.5 Layer.lightTransmission

Description

Returns the value of the Light Transmission property for a 3D layer.

Type

Number

26.6 Layer.castsShadows

Description

Returns a value of `1.0` if the layer casts shadows and `2` if the property is set to `Only`.

Type

Number

26.7 Layer.acceptsShadows

Description

Returns a value of `1` if the layer accepts shadows and `2` if the property is set to `Only`.

Type

Number

26.8 Layer.acceptsLights

Description

Returns a value of `1` if the layer accepts lights.

Type

Boolean Number

26.9 Layer.ambient

Description

Returns the ambient component value as a percentage.

Type

Number

26.10 Layer.diffuse

Description

Returns the diffuse component value as a percentage.

Type

Number

26.11 Layer.specular

Description

Returns the specular component value as a percentage.

Type

Number

26.12 Layer.shininess

Description

Returns the shininess component value as a percentage.

Type

Number

26.13 Layer.metal

Description

Returns the metal component value as a percentage.

Type

Number