NSAnimation Class Reference

Cocoa > Graphics & Imaging

2007-10-31

Apple Inc.
© 2007 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Apple, the Apple logo, Carbon, Cocoa, Mac, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

NSAnimation Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.4 and later.
Declared in	NSAnimation.h
Companion guides	Animation Programming Guide for Cocoa Cocoa Drawing Guide
Related sample code	CarbonCocoaCoreImageTab iSpend Reducer

Overview

Objects of the `NSAnimation` class manage the timing and progress of animations in the user interface. The class also lets you link together multiple animations so that when one animation ends another one starts. It does not provide any drawing support for animation and does not directly deal with views, targets, or actions.

Note: For simple tasks requiring a timing mechanism, consider using `NSTimer`.

`NSAnimation` objects have several characteristics, including duration, frame rate, and animation curve, which describes the relative speed of the animation over its course. You can set progress marks in an animation, each of which specifies a percentage of the animation completed; when an animation reaches a progress mark, it notifies its delegate and posts a notification to any observers. Animations execute in one of three blocking modes: blocking, non-blocking on the main thread, and non-blocking on a separate thread. The non-blocking modes permit the handling of user events while the animation is running.

Subclassing Notes

The usual usage pattern for `NSAnimation` is to make a subclass that overrides (at least) the setCurrentProgress: (page 15) method to invoke the superclass implementation and then perform whatever animation action is needed. The method implementation might invoke currentValue (page 10) and then

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

use that value to update some drawing; as a consequence of invoking currentValue (page 10), the method animation:valueForProgress: (page 20) is sent to the delegate (if there is a delegate that implements the method). For more information on subclassing `NSAnimation`, see *Cocoa Drawing Guide*.

Tasks

Initializing an NSAnimation Object

- `initWithDuration:animationCurve:` (page 12)
  Returns an `NSAnimation` object initialized with the specified duration and animation-curve values.

Configuring an Animation

- `setAnimationBlockingMode:` (page 14)
  Sets the blocking mode of the receiver.
- `animationBlockingMode` (page 8)
  Returns the blocking mode the receiver is next scheduled to run under.
- `runLoopModesForAnimating` (page 13)
  Overridden to return the run-loop modes that the receiver uses to run the animation timer in.
- `setAnimationCurve:` (page 14)
  Sets the receiver's animation curve.
- `animationCurve` (page 9)
  Returns the animation curve the receiver is running under.
- `setDuration:` (page 16)
  Sets the duration of the animation to a specified number of seconds.
- `duration` (page 11)
  Returns the duration of the animation, in seconds.
- `setFrameRate:` (page 16)
  Sets the frame rate of the receiver.
- `frameRate` (page 11)
  Returns the frame rate of the animation.

Managing the Delegate

- `setDelegate:` (page 15)
  Sets the delegate of the receiver.
- `delegate` (page 11)
  Returns the delegate of the receiver.

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

Controlling and Monitoring an Animation

- `startAnimation` (page 17)
  Starts the animation represented by the receiver.
- `stopAnimation` (page 18)
  Stops the animation represented by the receiver.
- `isAnimating` (page 12)
  Returns a Boolean value that indicates whether the receiver is currently animating.
- `setCurrentProgress:` (page 15)
  Sets the current progress of the receiver.
- `currentProgress` (page 10)
  Returns the current progress of the receiver.
- `currentValue` (page 10)
  Returns the current value of the effect based on the current progress.
- `animationDidEnd:` (page 20) *delegate method*
  Sent to the delegate when the specified animation completes its run.
- `animationDidStop:` (page 21) *delegate method*
  Sent to the delegate when the specified animation is stopped before it completes its run.
- `animationShouldStart:` (page 21) *delegate method*
  Sent to the delegate just after an animation is started.
- `animation:valueForProgress:` (page 20) *delegate method*
  Requests a custom curve value for the current progress value.

Managing Progress Marks

- `addProgressMark:` (page 8)
  Adds the progress mark to the receiver.
- `removeProgressMark:` (page 13)
  Removes progress mark from the receiver.
- `setProgressMarks:` (page 17)
  Sets the receiver's progress marks to the values specified in the passed-in array.
- `progressMarks` (page 13)
  Returns the receiver's progress marks.
- `animation:didReachProgressMark:` (page 19) *delegate method*
  Sent to the delegate when an animation reaches a specific progress mark.

Linking Animations Together

- `startWhenAnimation:reachesProgress:` (page 18)
  Starts running the animation represented by the receiver when another animation reaches a specific progress mark.
- `stopWhenAnimation:reachesProgress:` (page 19)
  Stops running the animation represented by the receiver when another animation reaches a specific progress mark.

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

- `clearStartAnimation` (page 9)
  Clears linkage to another animation that causes the receiver to start.
- `clearStopAnimation` (page 9)
  Clears linkage to another animation that causes the receiver to stop.

Instance Methods

addProgressMark:

Adds the progress mark to the receiver.

```
- (void)addProgressMark:(NSAnimationProgress)progressMark
```

Parameters

progressMark
  A `float` value (typed as `NSAnimationProgress`) between 0.0 and 1.0. Values outside that range are pinned to 0.0 or 1.0, whichever is nearest.

Discussion
A progress mark represents a percentage of the animation completed. When the animation reaches a progress mark, an `animation:didReachProgressMark:` (page 19) message is sent to the delegate and an `NSAnimationProgressMarkNotification` (page 24) is broadcast to all observers. You might receive multiple notifications of progress advances over multiple marks.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `currentProgress` (page 10)
- `removeProgressMark:` (page 13)

Declared In
`NSAnimation.h`

animationBlockingMode

Returns the blocking mode the receiver is next scheduled to run under.

```
- (NSAnimationBlockingMode)animationBlockingMode
```

Return Value
A constant representing the receiver's blocking mode. See "NSAnimationBlockingMode" (page 23) for valid values.

Discussion
The animation can run in blocking mode or non-blocking mode; non-blocking mode can be either on the main thread or on a separate thread. The default mode is `NSAnimationBlocking`.

Availability
Available in Mac OS X v10.4 and later.

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

See Also
- `setAnimationBlockingMode:` (page 14)

Declared In
`NSAnimation.h`

animationCurve

Returns the animation curve the receiver is running under.

```
- (NSAnimationCurve)animationCurve
```

Return Value
An `NSAnimationCurve` constant indicating the animation curve.

Discussion
The animation curve describes the relative frame rate over the course of the animation. See "NSAnimationCurve" (page 22) for valid `NSAnimationCurve` constants.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setAnimationCurve:` (page 14)

Declared In
`NSAnimation.h`

clearStartAnimation

Clears linkage to another animation that causes the receiver to start.

```
- (void)clearStartAnimation
```

Discussion
The linkage to the other animation is made with `startWhenAnimation:reachesProgress:` (page 18).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `startAnimation` (page 17)

Declared In
`NSAnimation.h`

clearStopAnimation

Clears linkage to another animation that causes the receiver to stop.

```
- (void)clearStopAnimation
```


2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

Discussion
The linkage to the other animation is made with `stopWhenAnimation:reachesProgress:` (page 19).

Availability
Available in Mac OS X v10.4 and later.

See Also
`- stopAnimation` (page 18)

Declared In
`NSAnimation.h`

currentProgress

Returns the current progress of the receiver.

```
- (NSAnimationProgress)currentProgress
```

Return Value
A `float` value typed as `NSAnimationProgress` that indicates the current progress of the animation.

Discussion
The current progress is a value between 0.0 and 1.0 that represents the percentage of the animation currently completed.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setCurrentProgress:` (page 15)

Declared In
`NSAnimation.h`

currentValue

Returns the current value of the effect based on the current progress.

```
- (float)currentValue
```

Return Value
A `float` value that indicates the current value of the animation effect.

Discussion
`NSAnimation` gets the current value from the delegate in `animation:valueForProgress:` (page 20) or, if that method is not implemented, computes it from the current progress by factoring in the animation curve. `NSAnimation` itself does not invoke this method currently. Instances of `NSAnimation` subclasses or other objects can invoke this method on a periodic basis to get the current value. Although this method has no corresponding setter method, those `NSAnimation` subclasses may override this method to return a custom curve value instead of implementing `animation:valueForProgress:` (page 20), thereby saving on the overhead of using a delegate. The current value can be less than 0.0 or greater than 1.0. For example, if you make the value greater than 1.0 you can achieve a "rubber effect" where the size of a view is temporarily larger before its final size.

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `currentProgress` (page 10)
- `setAnimationCurve:` (page 14)

Declared In
`NSAnimation.h`

delegate

Returns the delegate of the receiver.

```
- (id)delegate
```

Return Value
The receiver's delegate.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setDelegate:` (page 15)

Declared In
`NSAnimation.h`

duration

Returns the duration of the animation, in seconds.

```
- (NSTimeInterval)duration
```

Return Value
An `NSTimeInterval` value indicating the duration.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setDuration:` (page 16)

Declared In
`NSAnimation.h`

frameRate

Returns the frame rate of the animation.

```
- (float)frameRate
```


2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

Discussion
The frame rate is the number of updates per second. It is not guaranteed to be accurate because of differences between systems on the time needed to process a frame.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSAnimation.h`

initWithDuration:animationCurve:

Returns an `NSAnimation` object initialized with the specified duration and animation-curve values.

```
- (id)initWithDuration:(NSTimeInterval)duration
    animationCurve:(NSAnimationCurve)animationCurve
```

Parameters

duration

The number of seconds over which the animation occurs. Specifying a negative number raises an exception.

animationCurve

An `NSAnimationCurve` constant that describes the relative speed of the animation over its course; if it is zero, the default curve (`NSAnimationEaseInOut`) is used.

Return Value
An initialized `NSAnimation` instance. Returns `nil` if the object could not be initialized.

Discussion
You can always later change the duration of an `NSAnimation` object by sending it a `setDuration:` (page 16) message, even while the animation is running. See "Constants" for descriptions of the `NSAnimationCurve` constants.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
CarbonCocoaCoreImageTab

Reducer

Declared In
`NSAnimation.h`

isAnimating

Returns a Boolean value that indicates whether the receiver is currently animating.

```
- (BOOL)isAnimating
```

Return Value
`YES` if the receiver is animating, `NO` otherwise.

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSAnimation.h`

progressMarks

Returns the receiver's progress marks.

```
- (NSArray *)progressMarks
```

Return Value
An array of `NSNumber` objects, each encapsulating a `float` value (typed as `NSAnimationProgress`) that represents a current progress mark. If the receiver has no progress marks, an empty array is returned.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- addProgressMark:` (page 8)
`- setProgressMarks:` (page 17)

Declared In
`NSAnimation.h`

removeProgressMark:

Removes progress mark from the receiver.

```
- (void)removeProgressMark:(NSAnimationProgress)progressMark
```

Parameters
progressMark
A `float` value (typed as `NSAnimationProgress`) that indicates the portion of the animation completed. The value should correspond to a progress mark set with addProgressMark: (page 8) or setProgressMarks: (page 17).

Availability
Available in Mac OS X v10.4 and later.

See Also
`- addProgressMark:` (page 8)

Declared In
`NSAnimation.h`

runLoopModesForAnimating

Overridden to return the run-loop modes that the receiver uses to run the animation timer in.

```
- (NSArray *)runLoopModesForAnimating
```


2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

Return Value
An array of constants that indicate the modes the animation's run loop can be in. By default, the method returns `nil`, which indicates that the animation can be run in default, modal, or event-tracking mode. See the `NSRunLoop` class reference for information about the mode constants

Discussion
The value returned from this method is ignored if the animation blocking mode is something other than `NSAnimationNonblocking`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setAnimationBlockingMode:` (page 14)

Declared In
`NSAnimation.h`

setAnimationBlockingMode:

Sets the blocking mode of the receiver.

```
- (void)setAnimationBlockingMode:(NSAnimationBlockingMode)animationBlockingMode
```

Parameters

animationBlockingMode

A constant representing the blocking mode the animation is next scheduled to run under. See "NSAnimationBlockingMode" (page 23) for valid values.

If the constant is `NSAnimationNonblocking`, the animation runs in the main thread in one of the standard run-loop modes or in a mode returned from `runLoopModesForAnimating` (page 13). If *animationBlockingMode* is `NSAnimationNonblockingThreaded`, a new thread is spawned to run the animation.

Discussion
The default mode is `NSAnimationBlocking`, which means that the animation runs on the main thread in a custom run-loop mode that blocks user events. The new blocking mode takes effect the next time the receiver is started and has no effect on an animation underway.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- animationBlockingMode` (page 8)

Related Sample Code
Reducer

Declared In
`NSAnimation.h`

setAnimationCurve:

Sets the receiver's animation curve.

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

```
- (void)setAnimationCurve:(NSAnimationCurve)curve
```

Parameters

curve

An `NSAnimationCurve` constant specifying the animation curve. Invalid values raise an exception.

Discussion

The animation curve describes the relative frame rate over the course of the animation; predefined curves are linear, ease in (slow down near end), ease out (slowly speed up at start), and ease in-ease out (S-curve). Sending this message affects animations already in progress. The `NSAnimationCurve` setting is ignored if the delegate implements `animation:valueForProgress:` (page 20). See "NSAnimationCurve" (page 22) for valid `NSAnimationCurve` constants.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

QTCoreVideo301

Declared In

`NSAnimation.h`

setCurrentProgress:

Sets the current progress of the receiver.

```
- (void)setCurrentProgress:(NSAnimationProgress)progress
```

Parameters

progress

A `float` value typed as `NSAnimationProgress` that specifies the current progress of the animation. This value should be between 0.0 and 1.0; values that are out of range are pinned to 0.0 or 1.0, whichever is closer.

Discussion

You can use this method to adjust the progress of a running animation. The `NSAnimation` class invokes this method while the animation is running to change the progress for the next frame. Subclasses can override this method to get the latest value and perform their action with it, possibly in a secondary thread. Alternatively, you can implement the delegation method `animation:valueForProgress:` (page 20).

Availability

Available in Mac OS X v10.4 and later.

See Also

`- currentProgress` (page 10)

Declared In

`NSAnimation.h`

setDelegate:

Sets the delegate of the receiver.

```
- (void)setDelegate:(id)delegate
```


2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

Parameters

delegate

The delegate for the receiver.

Availability
Available in Mac OS X v10.4 and later.

See Also
- delegate (page 11)

Declared In
NSAnimation.h

setDuration:

Sets the duration of the animation to a specified number of seconds.

```
- (void)setDuration:(NSTimeInterval)duration
```

Parameters

duration

An NSTimeInterval value specifying the duration of the animation. Negative values raise an exception.

Discussion
You can change the duration of an animation while it is running. However, setting the duration of a running animation to an interval shorter than the current progress ends the animation.

Availability
Available in Mac OS X v10.4 and later.

See Also
- duration (page 11)

Related Sample Code
QTCoreVideo301
Reducer

Declared In
NSAnimation.h

setFrameRate:

Sets the frame rate of the receiver.

```
- (void)setFrameRate:(float)framesPerSecond
```

Parameters

framesPerSecond

A float value specifying the number of updates per second for the animation. This value must be positive; negative values raise an exception. A frame rate of 0.0 means to go as fast as possible.

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

Discussion
The frame rate is not guaranteed due to differences among systems for the time needed to process a frame. You can change the frame rate while an animation is running and the new value is used at the next frame. The default frame rate is set to a reasonable value (which is subject to future change).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `frameRate` (page 11)

Declared In
`NSAnimation.h`

setProgressMarks:

Sets the receiver's progress marks to the values specified in the passed-in array.

```
- (void)setProgressMarks:(NSArray *)progressMarks
```

Parameters

progressMarks
An array of `NSNumber` objects, each encapsulating a `float` value (typed as `NSAnimationProgress`) that represents a current progress mark. Passing in `nil` clears all progress marks.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `progressMarks` (page 13)

Declared In
`NSAnimation.h`

startAnimation

Starts the animation represented by the receiver.

```
- (void)startAnimation
```

Discussion
The receiver retains itself and is then autoreleased at the end of the animation or when it receives `stopAnimation` (page 18). If the blocking mode is `NSAnimationBlocking`, the method only returns after the animation has completed or the delegate sends it `stopAnimation` (page 18). If the receiver has a progress of 1.0, it starts again at 0.0.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `startWhenAnimation:reachesProgress:` (page 18)
- `stopAnimation` (page 18)

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

Related Sample Code
QTCoreVideo301
Reducer

Declared In
NSAnimation.h

startWhenAnimation:reachesProgress:

Starts running the animation represented by the receiver when another animation reaches a specific progress mark.

```
- (void)startWhenAnimation:(NSAnimation *)animation
    reachesProgress:(NSAnimationProgress)startProgress
```

Parameters

animation
The other NSAnimation object with which the receiver is linked.

startProgress
A float value (typed as NSAnimationProgress) that specifies a progress mark of the other animation.

Discussion
This method links the running of two animations together. You can set only one NSAnimation object as a start animation and one as a stop animation at any one time. Setting a new start animation removes any animation previously set.

Availability
Available in Mac OS X v10.4 and later.

See Also
- clearStartAnimation (page 9)
- startAnimation (page 17)
- stopWhenAnimation:reachesProgress: (page 19)

Declared In
NSAnimation.h

stopAnimation

Stops the animation represented by the receiver.

```
- (void)stopAnimation
```

Discussion
The current progress of the receiver is not reset. When this method is sent to instances of NSViewAnimation (a subclass of NSAnimation) the receiver moves to the end frame location.

Availability
Available in Mac OS X v10.4 and later.

See Also
- startAnimation (page 17)

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

- stopWhenAnimation:reachesProgress: (page 19)

Declared In
NSAnimation.h

stopWhenAnimation:reachesProgress:

Stops running the animation represented by the receiver when another animation reaches a specific progress mark.

```
- (void)stopWhenAnimation:(NSAnimation *)animation
    reachesProgress:(NSAnimationProgress)stopProgress
```

Parameters

animation

The other NSAnimation object with which the receiver is linked.

stopProgress

A float value (typed as NSAnimationProgress) that specifies a progress mark of the other animation.

Discussion
This method links the running of two animations together. You can set only one NSAnimation object as a start animation and one as a stop animation at any one time. Setting a new stop animation removes any animation previously set.

Availability
Available in Mac OS X v10.4 and later.

See Also
- clearStopAnimation (page 9)
- startWhenAnimation:reachesProgress: (page 18)
- stopAnimation (page 18)

Declared In
NSAnimation.h

Delegate Methods

animation:didReachProgressMark:

Sent to the delegate when an animation reaches a specific progress mark.

```
- (void)animation:(NSAnimation *)animation
    didReachProgressMark:(NSAnimationProgress)progress
```

Parameters

animation

A running NSAnimation object that has reached a progress mark.

progress

A float value (typed as NSAnimationProgress) that indicates a progress mark of animation.

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

Discussion
The delegate typically implements this method to perform some animation effect for the time slice indicated by *progress*, such as redrawing objects in a view with new coordinates or changing the frame location or size of a window or view. As an alternative to this delegation message, you may choose to observe the `NSAnimationProgressMarkNotification` (page 24) notification.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSAnimation.h`

animation:valueForProgress:

Requests a custom curve value for the current progress value.

```
- (float)animation:(NSAnimation *)animation
    valueForProgress:(NSAnimationProgress)progress
```

Parameters

animation

An `NSAnimation` object that is running.

progress

A `float` value (typed as `NSAnimationProgress`) that indicates a progress mark of `animation`. This value is always between 0.0 and 1.0.

Return Value
A `float` value representing a custom curve.

Discussion
The delegate can compute and return a custom curve value for the given progress value. If the delegate does not implement this method, `NSAnimation` computes the current curve value.

The animation:valueForProgress: message is sent to the delegate when an `NSAnimation` object receives a currentValue (page 10) message. The value the delegate returns is used as the value of currentValue; if there is no delegate, or it doesn't implement animation:valueForProgress:, `NSAnimation` computes and returns the current value. `NSAnimation` does not invoke currentValue itself, but subclasses might.

See the description of currentValue (page 10) for more information.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- currentValue` (page 10)

Declared In
`NSAnimation.h`

animationDidEnd:

Sent to the delegate when the specified animation completes its run.

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

```
- (void)animationDidEnd:(NSAnimation *)animation
```

Parameters

animation

The `NSAnimation` instance that completed its run.

Discussion

When an `NSAnimation` object reaches the end of its planned duration, it has a progress value of 1.0.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- animationDidStop:` (page 21)

`- currentProgress` (page 10)

Declared In

`NSAnimation.h`

animationDidStop:

Sent to the delegate when the specified animation is stopped before it completes its run.

```
- (void)animationDidStop:(NSAnimation *)animation
```

Parameters

animation

The `NSAnimation` instance that was stopped.

Discussion

An `NSAnimation` object stops running when it receives a `stopAnimation` (page 18) message.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- animationDidEnd:` (page 20)

Declared In

`NSAnimation.h`

animationShouldStart:

Sent to the delegate just after an animation is started.

```
- (BOOL)animationShouldStart:(NSAnimation *)animation
```

Parameters

animation

The `NSAnimation` object that was just started.

Return Value

`NO` to cancel the animation, `YES` to have the animation proceed.

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

Discussion
The delegate is sent this message just after *animation* receives a `startAnimation` (page 17) message. The delegate can use this method to prepare objects and resources for the effect.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `animationDidEnd:` (page 20)
- `animationDidStop:` (page 21)

Declared In
`NSAnimation.h`

Constants

NSAnimationCurve

These constants describe the curve of an animation—that is, the relative speed of an animation from start to finish.

```
enum {
    NSAnimationEaseInOut,
    NSAnimationEaseIn,
    NSAnimationEaseOut,
    NSAnimationLinear
};
typedef NSUInteger NSAnimationCurve;
```

Constants

`NSAnimationEaseInOut`
Describes an S-curve in which the animation slowly speeds up and then slows down near the end of the animation. This constant is the default.
Available in Mac OS X v10.4 and later.
Declared in `NSAnimation.h`.

`NSAnimationEaseIn`
Describes an animation that slows down as it reaches the end.
Available in Mac OS X v10.4 and later.
Declared in `NSAnimation.h`.

`NSAnimationEaseOut`
Describes an animation that slowly speeds up from the start.
Available in Mac OS X v10.4 and later.
Declared in `NSAnimation.h`.

`NSAnimationLinear`
Describes an animation in which there is no change in frame rate.
Available in Mac OS X v10.4 and later.
Declared in `NSAnimation.h`.

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

Discussion
You initialize an `NSAnimation` object using one of these constants with `initWithDuration:animationCurve:` (page 12) and you can set it thereafter with `setAnimationCurve:` (page 14).

Declared In
`NSAnimation.h`

NSAnimationBlockingMode

These constants indicate the blocking mode of an `NSAnimation` object when it is running.

```
enum {
    NSAnimationBlocking,
    NSAnimationNonblocking,
    NSAnimationNonblockingThreaded
};
typedef NSUInteger NSAnimationBlockingMode;
```

Constants

`NSAnimationBlocking`

Requests the animation to run in the main thread in a custom run-loop mode that blocks user input.

This is the default.

Available in Mac OS X v10.4 and later.

Declared in `NSAnimation.h`.

`NSAnimationNonblocking`

Requests the animation to run in a standard or specified run-loop mode that allows user input.

Available in Mac OS X v10.4 and later.

Declared in `NSAnimation.h`.

`NSAnimationNonblockingThreaded`

Requests the animation to run in a separate thread that is spawned by the `NSAnimation` object.

The secondary thread has its own run loop.

Available in Mac OS X v10.4 and later.

Declared in `NSAnimation.h`.

Discussion
You specify one of these constants in `setAnimationBlockingMode:` (page 14).

Declared In
`NSAnimation.h`

Animation action triggers

These constants are used by the `NSAnimatablePropertyContainer` methods `defaultAnimationForKey:` and `animationForKey:`.

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

```
NSString *NSAnimationTriggerOrderIn;
NSString *NSAnimationTriggerOrderOut;
```

Constants

`NSAnimationTriggerOrderIn`

The trigger that represents the action taken when a view becomes visible, either as a result of being inserted into the visible view hierarchy or the view is no longer set as hidden.

Available in Mac OS X v10.5 and later.

Declared in `NSAnimation.h`.

`NSAnimationTriggerOrderOut`

The trigger that represents the action taken when the view is either removed from the view hierarchy or is hidden.

Available in Mac OS X v10.5 and later.

Declared in `NSAnimation.h`.

Declared In

`NSAnimation.h`

Notification Key

This constant is returned in the userInfo dictionary of the `NSAnimationProgressMarkNotification` (page 24) notification.

```
NSString*    NSAnimationProgressMark;
```

Constants

`NSAnimationProgressMark`

Contains a float as an NSNumber instance that indicates the current animation progress.

Available in Mac OS X v10.4 and later.

Declared in `NSAnimation.h`.

Declared In

`NSAnimation.h`

Notifications

NSAnimationProgressMarkNotification

Posted when the current progress of a running animation reaches one of its progress marks.

The notification object is a running `NSAnimation` object. The `userInfo` dictionary contains the current progress mark, accessed via the key `NSAnimationProgressMark`.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- animation:didReachProgressMark:` (page 19)

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

Declared In
`NSAnimation.h`

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

Document Revision History

This table describes the changes to *NSAnimation Class Reference*.

Date	Notes
2007-10-31	Updated for Mac OS v10.5. Corrected typos.
2007-04-01	Updated for Mac OS X v10.5.
2006-05-23	First publication of this content as a separate document.

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

Document Revision History

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

Index

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.

2007-10-31 | © 2007 Apple Inc. All Rights Reserved.