
NSAnimation Class Reference

Graphics & Animation: Animation



2009-04-17



Apple Inc.
© 2009 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

NSAnimation Class Reference 5

Overview	5
Subclassing Notes	5
Tasks	6
Initializing an NSAnimation Object	6
Configuring an Animation	6
Managing the Delegate	6
Controlling and Monitoring an Animation	7
Managing Progress Marks	7
Linking Animations Together	7
Instance Methods	8
addProgressMark:	8
animationBlockingMode	8
animationCurve	9
clearStartAnimation	9
clearStopAnimation	9
currentProgress	10
currentValue	10
delegate	11
duration	11
frameRate	11
initWithDuration:animationCurve:	12
isAnimating	12
progressMarks	13
removeProgressMark:	13
runLoopModesForAnimating	13
setAnimationBlockingMode:	14
setAnimationCurve:	14
setCurrentProgress:	15
setDelegate:	15
setDuration:	16
setFrameRate:	16
setProgressMarks:	17
startAnimation	17
startWhenAnimation:reachesProgress:	18
stopAnimation	18
stopWhenAnimation:reachesProgress:	19
Constants	19
NSAnimationCurve	19
NSAnimationBlockingMode	20
NSAnimationProgress	21

CONTENTS

Animation action triggers	21
NSAnimationProgressMark Notification Key	22
Notifications	22
NSAnimationProgressMarkNotification	22

Document Revision History 23

Index 25

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	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

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

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.

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.

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.

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.
- [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:` message is sent to the delegate and an [NSAnimationProgressMarkNotification](#) (page 22) 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 20) 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.

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 19) 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

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 ValueA `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 ValueA `float` value that indicates the current value of the animation effect.**Discussion**

`NSAnimation` gets the current value from the delegate in `animation:valueForProgress:` 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:`, 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.

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 < NSAnimationDelegate >)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

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

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.

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

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 20) 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

From A View to A Movie

Reducer

Declared In

`NSAnimation.h`

setAnimationCurve:

Sets the receiver’s animation curve.

```
- (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:`. See “[NSAnimationCurve](#)” (page 19) for valid `NSAnimationCurve` constants.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

From A View to A Movie

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:`.

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 < NSAnimationDelegate >)delegate
```

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**

From A View to A Movie

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.

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)

Related Sample Code

From A View to A Movie

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)
- [stopWhenAnimation:reachesProgress:](#) (page 19)

Related Sample Code

From A View to A Movie

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

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`.

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).

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).

NSAnimationProgress

`NSAnimationProgress` is returned in the `userInfo` dictionary of an `NSAnimationProgressMarkNotification` (page 22) notification. It will have a value between 0.0 and 1.0

```
typedef float NSAnimationProgress;
```

Availability

Available in Mac OS X v10.4 and later.

Declared In

`NSAnimation.h`

Animation action triggers

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

```
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`.

NSAnimationProgressMark Notification Key

This constant is returned in the `userInfo` dictionary of the [NSAnimationProgressMarkNotification](#) (page 22) notification.

```
NSString*    NSAnimationProgressMark;
```

Constants

`NSAnimationProgressMark`

Contains the value of an [NSAnimationProgress](#) (page 21) as an `NSNumber` instance that indicates the current animation progress. The value will be between 0.0 and 1.0.

Available in Mac OS X v10.4 and later.

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:` (`NSAnimationDelegate`)

Declared In

`NSAnimation.h`

Document Revision History

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

Date	Notes
2009-04-17	Updated for Mac OS X v10.6. Moved delegate methods to <i>NSAnimationDelegate Protocol Reference</i> .
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.

REVISION HISTORY

Document Revision History

Index

A

`addProgressMark:` instance method 8
Animation action triggers 21
`animationBlockingMode` instance method 8
`animationCurve` instance method 9

C

`clearStartAnimation` instance method 9
`clearStopAnimation` instance method 9
`currentProgress` instance method 10
`currentValue` instance method 10

D

`delegate` instance method 11
`duration` instance method 11

F

`frameRate` instance method 11

I

`initWithDuration:animationCurve:` instance method 12
`isAnimating` instance method 12

N

`NSAnimationBlocking` constant 20
`NSAnimationBlockingMode` 20

NSAnimationCurve 19

`NSAnimationEaseIn` constant 20
`NSAnimationEaseInOut` constant 20
`NSAnimationEaseOut` constant 20
`NSAnimationLinear` constant 20
`NSAnimationNonblocking` constant 21
`NSAnimationNonblockingThreaded` constant 21
`NSAnimationProgress` data type 21
`NSAnimationProgressMark` constant 22
NSAnimationProgressMark Notification Key 22
`NSAnimationProgressMarkNotification` notification 22
`NSAnimationTriggerOrderIn` constant 21
`NSAnimationTriggerOrderOut` constant 21

P

`progressMarks` instance method 13

R

`removeProgressMark:` instance method 13
`runLoopModesForAnimating` instance method 13

S

`setAnimationBlockingMode:` instance method 14
`setAnimationCurve:` instance method 14
`setCurrentProgress:` instance method 15
`setDelegate:` instance method 15
`setDuration:` instance method 16
`setFrameRate:` instance method 16
`setProgressMarks:` instance method 17
`startAnimation` instance method 17
`startWhenAnimation:reachesProgress:` instance method 18
`stopAnimation` instance method 18

`stopWhenAnimation:reachesProgress:` **instance**
method [19](#)