

UIDynamicAnimator Class Reference

Contents

UIDynamicAnimator Class Reference 3

Overview 3

Tasks 5

 Initializing and Managing a Dynamic Animator 5

 Accessing a Dynamic Animator's State 5

 Collection View Additions 5

Properties 6

 behaviors 6

 delegate 6

 referenceView 6

 running 7

Instance Methods 7

 addBehavior: 7

 elapsedTime 8

 initWithCollectionViewLayout: 8

 initWithReferenceView: 9

 itemsInRect: 9

 layoutAttributesForCellAtIndexPath: 10

 layoutAttributesForDecorationViewOfKind:atIndexPath: 10

 layoutAttributesForSupplementaryViewOfKind:atIndexPath: 11

 removeAllBehaviors 11

 removeBehavior: 12

Document Revision History 13

UIDynamicAnimator Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/UIKit.framework
Availability	Available in iOS 7.0 and later.
Declared in	UIDynamicAnimator.h

Overview

Important: This is a preliminary document for an API or technology in development. Although this document has been reviewed for technical accuracy, it is not final. This Apple confidential information is for use only by registered members of the applicable Apple Developer program. Apple is supplying this confidential information to help you plan for the adoption of the technologies and programming interfaces described herein. This information is subject to change, and software implemented according to this document should be tested with final operating system software and final documentation. Newer versions of this document may be provided with future seeds of the API or technology.

A dynamic animator provides physics-related capabilities and animations for its dynamic items, and provides the context for those animations. It does this by intermediating between the underlying iOS physics engine and dynamic items, via behavior objects you add to the animator.

A **dynamic item** is any iOS or custom object that conforms to the `UIDynamicItem` protocol. The `UIView` and `UICollectionViewLayoutAttributes` classes implement this protocol starting in iOS 7.0. You can also use a dynamic animator with custom objects for such purposes as reacting to rotation or position changes computed by the animator.

You specify dynamic behaviors using `UIAttachmentBehavior`, `UICollisionBehavior`, `UIDynamicItemBehavior`, `UIGravityBehavior`, `UIPushBehavior`, and `UISnapBehavior` objects. Each of these provides configuration options and lets you associate one or more dynamic items to the behavior. You can define composite behaviors using the `addChildBehavior:` method of the `UIDynamicBehavior` parent behavior class. To activate a behavior, add it to an animator.

To employ a dynamic animator, first identify the type of dynamic items you want to animate. This choice determines which initializer to call, and this in turn determines how the coordinate system gets set up.

The three ways to initialize an animator, the dynamic items you can then use, and the resulting coordinate system, are as follows:

- To animate views, create an animator with the `initWithReferenceView:` (page 9) method. The coordinate system of the reference view serves as the coordinate system for the animator's behaviors and items. Each dynamic item you associate with this sort of animator must be a `UIView` object and must descend from the reference view.

You can define a boundary, for items participating in a collision behavior, relative to the bounds of the reference view. See the `setTranslatesReferenceBoundsIntoBoundaryWithInsets:` method.

- To animate collection views, create an animator with the `initWithCollectionViewLayout:` (page 8) method. The resulting animator employs a collection view layout (an object of the `UICollectionViewLayout` class) for its coordinate system. The dynamic items in this sort of animator must be `UICollectionViewLayoutAttributes` objects that are part of the layout.

You can define a boundary, for items participating in a collision behavior, relative to the bounds of the collection view layout. See the `setTranslatesReferenceBoundsIntoBoundaryWithInsets:` method.

A collection view animator automatically calls the `invalidateLayout` method as needed, and automatically pauses and resumes animation, as appropriate, when you change a collection view's layout.

- To employ a dynamic animator with other objects that conform to the `UIDynamicItem` protocol, create an animator with the inherited `init` method. The resulting animator employs an abstract coordinate system, not tied to the screen or to any view.

There is no reference boundary to refer to when defining a collision boundary for use with this sort of animator. However, you can still, in a collision behavior, specify custom boundaries as described in *UICollisionBehavior Class Reference*.

All types of dynamic animators share the following characteristics:

- Each dynamic animator is independent of other dynamic animators you create
- You can associate a given dynamic item with multiple behaviors, provided those behaviors belong to the same animator
- An animator automatically pauses when all its items are at rest, and automatically resumes when a behavior parameter changes or a behavior or item is added or removed

You can implement a delegate to respond to changes in animator pause/resumption status, using the `dynamicAnimatorDidPause:` and `dynamicAnimatorWillResume:` methods of the `UIDynamicAnimatorDelegate` protocol.

Tasks

Initializing and Managing a Dynamic Animator

- [initWithReferenceView:](#) (page 9)
Initializes a dynamic animator with a specified view as its reference view.
- [initWithCollectionViewLayout:](#) (page 8)
Initializes a dynamic animator with a specified collection view layout.
- [itemsInRect:](#) (page 9)
Returns the dynamic items, from the animator’s behaviors, that intersect a specified rectangle.
- [addBehavior:](#) (page 7)
Adds a dynamic behavior to the dynamic animator.
- [removeBehavior:](#) (page 12)
Removes a specified dynamic behavior from the dynamic animator.
- [removeAllBehaviors](#) (page 11)
Removes all of the dynamic behaviors from the dynamic animator.

Accessing a Dynamic Animator’s State

- [elapsedTime](#) (page 8)
Returns the time interval since the dynamic animator started running.
- [running](#) (page 7) *property*
Returns YES if the dynamic animator is running. (read-only)
- [behaviors](#) (page 6) *property*
The dynamic behaviors managed by the dynamic animator. (read-only)
- [referenceView](#) (page 6) *property*
The view that the dynamic animator was initialized was initialized with. (read-only)
- [delegate](#) (page 6) *property*
The delegate for responding to pausing or resumption of animation.

Collection View Additions

- [layoutAttributesForCellAtIndexPath:](#) (page 10)
A convenience method for returning the layout attributes for a collection view cell.

- [layoutAttributesForDecorationViewOfKind:atIndexPath:](#) (page 10)
A convenience method for returning the layout attributes for a collection view decoration view.
- [layoutAttributesForSupplementaryViewOfKind:atIndexPath:](#) (page 11)
A convenience method for returning the layout attributes for a collection view supplementary view.

Properties

behaviors

The dynamic behaviors managed by the dynamic animator. (read-only)

```
@property(n nonatomic, readonly, copy) NSArray *behaviors
```

Availability

Available in iOS 7.0 and later.

Declared in

UIDynamicAnimator.h

delegate

The delegate for responding to pausing or resumption of animation.

```
@property(n nonatomic, assign) id<UIDynamicAnimatorDelegate> delegate
```

Discussion

The methods for a dynamic animator delegate are described in *UIDynamicAnimatorDelegate Protocol Reference*.

Availability

Available in iOS 7.0 and later.

Declared in

UIDynamicAnimator.h

referenceView

The view that the dynamic animator was initialized with. (read-only)

@property(nonatomic, readonly) UIView *referenceView

Discussion

This property has a value only for a dynamic animator initialized using the [initWithReferenceView:](#) (page 9) method.

Availability

Available in iOS 7.0 and later.

Declared in

UIDynamicAnimator.h

running

Returns YES if the dynamic animator is running. (read-only)

@property(nonatomic, readonly, getter=isRunning) BOOL running

Discussion

The views associated with an animator's behaviors can change position or change transform only when the animator is running. For optimization purposes, iOS can pause and then restart an animator. Use this method if you need to check whether or not your views are currently subject to changes in position or transform.

Availability

Available in iOS 7.0 and later.

Declared in

UIDynamicAnimator.h

Instance Methods

addBehavior:

Adds a dynamic behavior to the dynamic animator.

– (void)addBehavior:(UIDynamicBehavior *)behavior

Parameters

behavior

The behavior you are adding.

Availability

Available in iOS 7.0 and later.

Declared in

UIDynamicAnimator.h

elapsedTime

Returns the time interval since the dynamic animator started running.

– (NSTimeInterval)elapsedTime

Return Value

The elapsed time since the dynamic animator started running.

Availability

Available in iOS 7.0 and later.

Declared in

UIDynamicAnimator.h

initWithCollectionViewLayout:

Initializes a dynamic animator with a specified collection view layout.

– (instancetype)initWithCollectionViewLayout:(UICollectionViewLayout *)layout

Parameters

layout

The collection view layout for the dynamic animator, serving as the reference view for a dynamic animator in collection-view mode.

Return Value

The initialized dynamic animator, or `nil` if there was a problem initializing the object.

Discussion

When you initialize a dynamic animator with this method, the behaviors (and their dynamic items) that you add to the animator employ the collection view layout’s coordinate system.

Availability

Available in iOS 7.0 and later.

Declared in

UIDynamicAnimator.h

initWithReferenceView:

Initializes a dynamic animator with a specified view as its reference view.

– (instancetype)initWithReferenceView:(UIView *)view

Parameters

view

The view for the dynamic animator, called the *reference view*.

Return Value

The initialized dynamic animator, or `nil` if there was a problem initializing the object.

Discussion

When you initialize a dynamic animator with this method, the behaviors (and their dynamic items) that you add to the animator employ the reference view’s coordinate system.

Availability

Available in iOS 7.0 and later.

Declared in

UIDynamicAnimator.h

itemsInRect:

Returns the dynamic items, from the animator’s behaviors, that intersect a specified rectangle.

– (NSArray *)itemsInRect:(CGRect)rect

Parameters

rect

The rectangle you are interested in.

Return Value

The dynamic items, from the animator’s behaviors, that intersect the specified rectangle.

Discussion

The coordinate system that pertains to the `rect` parameter depends on how you initialized the animator, as described in the Overview in this document.

Availability

Available in iOS 7.0 and later.

Declared in

UIDynamicAnimator.h

layoutAttributesForCellAtIndexPath:

A convenience method for returning the layout attributes for a collection view cell.

```
– (UICollectionViewLayoutAttributes *)layoutAttributesForCellAtIndexPath:(NSIndexPath *)indexPath
```

Parameters

indexPath

The index path for the cell whose layout attributes you want.

Return Value

The collection view layout attributes for the specified collection view cell.

Availability

Available in iOS 7.0 and later.

Declared in

UIDynamicAnimator.h

layoutAttributesForDecorationViewOfKind:atIndexPath:

A convenience method for returning the layout attributes for a collection view decoration view.

```
– (UICollectionViewLayoutAttributes *)layoutAttributesForDecorationViewOfKind:(NSString *)decorationViewKind  
atIndexPath:(NSIndexPath *)indexPath
```

Parameters

decorationViewKind

The kind identifier for the specified decoration view.

indexPath

The index path for the cell whose decoration view layout attributes you want.

Return Value

The collection view layout attributes for the specified decoration view.

Availability

Available in iOS 7.0 and later.

Declared in

UIDynamicAnimator.h

layoutAttributesForSupplementaryViewOfKind:atIndexPath:

A convenience method for returning the layout attributes for a collection view supplementary view.

```
– (UICollectionViewLayoutAttributes  
*)layoutAttributesForSupplementaryViewOfKind:(NSString *)kind atIndexPath:(NSIndexPath  
*)indexPath
```

Parameters

kind

A string that identifies the type of supplementary view whose layout attributes you want.

indexPath

The index path for the cell whose supplementary view layout attributes you want.

Return Value

The collection view layout attributes for the specified supplementary view.

Availability

Available in iOS 7.0 and later.

Declared in

UIDynamicAnimator.h

removeAllBehaviors

Removes all of the dynamic behaviors from the dynamic animator.

```
– (void)removeAllBehaviors
```

Availability

Available in iOS 7.0 and later.

Declared in

UIDynamicAnimator.h

removeBehavior:

Removes a specified dynamic behavior from the dynamic animator.

– (void)removeBehavior:(UIDynamicBehavior *)behavior

Parameters

behavior

The identifier for the dynamic behavior that you want to remove from the animator.

Availability

Available in iOS 7.0 and later.

Declared in

UIDynamicAnimator.h

Document Revision History

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

Date	Notes
2013-06-10	New document that describes the animation context for dynamic view behavior.



Apple Inc.
Copyright © 2013 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.

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.

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

Apple and the Apple logo are trademarks of Apple Inc., registered in the U.S. and other countries.

iOS is a trademark or registered trademark of Cisco in the U.S. and other countries and is used under license.

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.