UIImagePickerController Class Reference

User Experience: Controls

2010-06-29

Contents

UIImagePickerController Class Reference

Inherits from	UINavigationController : UIViewController : UIResponder : NSObject
Conforms to	NSCoding NSCoding (UIViewController) NSObject (NSObject)
Framework	/System/Library/Frameworks/UIKit.framework
Availability	Available in iOS 2.0 and later.
Declared in	UIImagePickerController.h

Overview

The `UIImagePickerController` class manages system-supplied user interfaces for choosing and taking pictures and movies on supported devices. The class manages user interactions and delivers the results of those interactions to the delegate object you've associated with the image picker.

To use the default image picker to handle user interactions, perform these steps:

1. Verify that the device is capable of picking content from the desired source. Do this calling the `isSourceTypeAvailable:` (page 16) class method.

2. Check which media types are available for the source type by calling the `availableMediaTypesForSourceType:` (page 14) class method. This lets you distinguish between a camera that can be used for video recording and one that can be used only for still images.

3. Tell the `UIImagePickerController` class which user interface to display. Do this by setting the `mediaTypes` (page 11) property.

4. Present the user interface.

5. When the user picks an image or movie, or cancels the operation, dismiss the image picker using your delegate object.

In addition to the default image picker, in iOS 3.1 and later you can manage user interactions yourself. To do this, provide an overlay view to display a custom picture-taking interface. This lets you initiate choosing and taking pictures and movies programmatically. Your custom overlay view can be displayed in addition to, or instead of, the default controls provided by the image picker interface.

To use this class, you must provide a delegate that conforms to the `UIImagePickerControllerDelegate` protocol. See *UIImagePickerControllerDelegate Protocol Reference*.

The default camera interface supports editing of previously-saved movies. Editing involves trimming from the start or end of the movie, then saving the trimmed movie.

In iOS 4.0 and later, you can provide custom controls to let the user adjust flash mode (on devices that have a flash LED), pick which camera to use (on devices that have a front and rear camera), and switch between still image and movie capture. You can also manage these settings programmatically.

In iOS 4.0 and later, on devices that have a flash LED, you can manipulate the flash directly to provide effects such as a strobe light. Present a picker interface set to use video capture mode. Then, turn the flash LED on or off by setting the `cameraFlashMode` (page 9) property to `UIImagePickerControllerCameraFlashModeOn` (page 21) or `UIImagePickerControllerCameraFlashModeOff` (page 21).

Movie capture has a default duration limit of 10 minutes but can be adjusted using the `videoMaximumDuration` (page 12) property. When a user taps the Share button to send a movie to MMS, MobileMe, YouTube, or another destination, an appropriate duration limit and an appropriate video quality are enforced.

To display an interface dedicated to movie editing, rather than one that also supports recording new movies, use the `UIVideoEditorController` class instead of this one. See *UIVideoEditorController Class Reference*.

Important: The `UIImagePickerController` class supports portrait mode only. This class is intended to be used as-is and does not support subclassing. The view hierarchy for this class is private and must not be modified, with one exception. In iOS 3.1 and later, you can assign a custom view to the `cameraOverlayView` (page 9) property and use that view to present additional information or manage the interactions between the camera interface and your code.

Tasks

Setting the Picker Source

`+ availableMediaTypesForSourceType:` (page 14)
Returns an array of the available media types for the specified source type.

`+ isSourceTypeAvailable:` (page 16)
Returns a Boolean value indicating whether the device supports picking media using the specified source type.

`sourceType` (page 12) *property*
The type of picker interface displayed by the controller.

Configuring the Picker

`allowsEditing` (page 8) *property*
A Boolean value indicating whether the user is allowed to edit a selected still image or movie.

`delegate` (page 10) *property*
The image picker's delegate object.

`mediaTypes` (page 11) *property*
An array indicating the media types to be accessed by the picker interface.

`allowsImageEditing` (page 23) *property* Deprecated in iOS 3.1
A Boolean value indicating whether the user is allowed to edit a selected image. (Deprecated. Use `allowsEditing` (page 8) instead.)

Configuring the Video Capture Options

`videoQuality` (page 13) *property*
The video recording and transcoding quality.

`videoMaximumDuration` (page 12) *property*
The maximum duration, in seconds, for a video recording.

Customizing the Camera Controls

`showsCameraControls` (page 11) *property*
Indicates whether the image picker displays the default camera controls.

`cameraOverlayView` (page 9) *property*
The custom view to display on top of the default image picker interface.

`cameraViewTransform` (page 10) *property*
The transform to apply to the camera's preview image.

Capturing Still Images or Movies

`- takePicture` (page 17)
Captures a still image using the camera.

`- startVideoCapture` (page 16)
Starts video capture using the camera specified by the `UIImagePickerControllerCameraDevice` (page 20) property.

`- stopVideoCapture` (page 17)
Stops video capture.

Configuring the Camera

`cameraDevice` (page 9) *property*
The camera used by the image picker controller.

`+ isCameraDeviceAvailable:` (page 15)
Returns a Boolean value that indicates whether a given camera is available.

`+ availableCaptureModesForCameraDevice:` (page 14)
Returns an array of `NSNumber` objects indicating the capture modes supported by a given camera device.

`cameraCaptureMode` (page 8) *property*
The capture mode used by the camera.

`cameraFlashMode` (page 9) *property*
The flash mode used by the active camera.

`+ isFlashAvailableForCameraDevice:` (page 15)
Indicates whether a given camera has flash illumination capability.

Properties

For more about Objective-C properties, see “Properties” in *The Objective-C Programming Language*.

allowsEditing

A Boolean value indicating whether the user is allowed to edit a selected still image or movie.

```
@property(nonatomic) BOOL allowsEditing
```

Discussion
If you allow the user to edit still images or movies, the delegate may receive a dictionary with information about the edits that were made. The protocol for the delegate is described in *UIImagePickerControllerDelegate Protocol Reference*.

This property is set to `NO` by default.

Availability
Available in iOS 3.1 and later.

Declared In
`UIImagePickerController.h`

cameraCaptureMode

The capture mode used by the camera.

```
@property(nonatomic) UIImagePickerControllerCameraCaptureMode cameraCaptureMode
```

Discussion
The various capture modes are listed in the “UIImagePickerControllerCameraCaptureMode” (page 20) enumeration. The default value is `UIImagePickerControllerCameraCaptureModePhoto` (page 20).

Availability
Available in iOS 4.0 and later.

See Also
`@property cameraDevice` (page 9)
`+ availableCaptureModesForCameraDevice:` (page 14)

Declared In
`UIImagePickerController.h`

cameraDevice

The camera used by the image picker controller.

```
@property(nonatomic) UIImagePickerControllerCameraDevice cameraDevice
```

Discussion
The default is `UIImagePickerControllerCameraDeviceRear` (page 20).

Availability
Available in iOS 4.0 and later.

See Also
`+ isCameraDeviceAvailable:` (page 15)
`+ isFlashAvailableForCameraDevice:` (page 15)
`+ availableCaptureModesForCameraDevice:` (page 14)

Declared In
`UIImagePickerController.h`

cameraFlashMode

The flash mode used by the active camera.

```
@property(nonatomic) UIImagePickerControllerCameraFlashMode cameraFlashMode
```

Discussion
The various flash modes are listed in the "UIImagePickerControllerCameraFlashMode" (page 21) enumeration. The default value is `UIImagePickerControllerCameraFlashModeAuto` (page 21).

The value of this property specifies the behavior of the still-image flash when the value of the `cameraCaptureMode` property is `UIImagePickerControllerCameraCaptureModePhoto` (page 20), and specifies the behavior of the video torch when `cameraCaptureMode` is `UIImagePickerControllerCameraCaptureModeVideo` (page 20).

Availability
Available in iOS 4.0 and later.

See Also
`@property cameraDevice` (page 9)
`@property cameraCaptureMode` (page 8)
`+ isFlashAvailableForCameraDevice:` (page 15)

Declared In
`UIImagePickerController.h`

cameraOverlayView

The custom view to display on top of the default image picker interface.

```
@property(nonatomic, retain) UIView *cameraOverlayView
```

Discussion
You can use an overlay view to present a custom view hierarchy on top of the default image picker interface. The image picker layers your custom overlay view on top of the other image picker views and positions it relative to the screen coordinates. If you have the default camera controls set to be visible, incorporate transparency into your view, or position it to avoid obscuring the underlying content.

This property is set to `nil` by default.

You can access this property only when the source type of the image picker is set to `UIImagePickerControllerSourceTypeCamera`. Attempting to access this property for other source types results in the throwing of an `NSInvalidArgumentException` exception.

Availability
Available in iOS 3.1 and later.

See Also
`@property showsCameraControls` (page 11)

Declared In
`UIImagePickerController.h`

cameraViewTransform

The transform to apply to the camera's preview image.

```
@property(nonatomic) CGAffineTransform cameraViewTransform
```

Discussion
This transform affects the live preview image only and does not affect your custom overlay view or the default image picker controls. You can use this property in conjunction with custom controls to implement your own electronic zoom behaviors.

You can access this property only when the source type of the image picker is set to `UIImagePickerControllerSourceTypeCamera`. Attempting to access this property for other source types results in the throwing of an `NSInvalidArgumentException` exception.

Availability
Available in iOS 3.1 and later.

Declared In
`UIImagePickerController.h`

delegate

The image picker's delegate object.

```
@property(nonatomic, assign) id<UINavigationControllerDelegate,
    UIImagePickerControllerDelegate> delegate
```

Discussion
The delegate receives notifications when the user picks an image or movie, or exits the picker interface. The delegate also decides when to dismiss the picker interface, so you must provide a delegate to use a picker. If this property is `nil`, the picker is dismissed immediately if you try to show it.

For information about the methods you can implement for your delegate object, see *UIImagePickerControllerDelegate Protocol Reference*.

Availability
Available in iOS 2.0 and later.

Declared In
`UIImagePickerController.h`

mediaTypes

An array indicating the media types to be accessed by the picker interface.

```
@property(nonatomic, copy) NSArray *mediaTypes
```

Discussion
Depending on the media types you assign to this property, the picker displays the still camera or the movie camera interface, or a selection control that lets the user choose the picker interface. Before setting this property, check which media types are available by calling the `availableMediaTypesForSourceType:` (page 14) class method.

By default, this property is set to the single value `kUTTypeImage`, which designates the still camera interface.

If you set this property to an empty array, or to an array in which none of the media types is available for the current source, the system throws an exception.

Availability
Available in iOS 3.0 and later.

Declared In
`UIImagePickerController.h`

showsCameraControls

Indicates whether the image picker displays the default camera controls.

```
@property(nonatomic) BOOL showsCameraControls
```

Discussion
The default value of this property is `YES`, which specifies that the default camera controls are visible in the picker. Set it to `NO` to hide the default controls if you want to instead provide a custom overlay view using the `cameraOverlayView` property.

Note: In iOS 3.1.3 and earlier, hiding the default camera controls limits you to taking still pictures only, regardless of whether movie capture is available on the device.

If you set this property to `NO` and provide your own custom controls, you can take multiple pictures before dismissing the image picker interface. However, if you set this property to `YES`, your delegate must dismiss the image picker interface after the user takes one picture or cancels the operation.

You can access this property only when the source type of the image picker is set to `UIImagePickerControllerSourceTypeCamera`. Attempting to access this property for other source types results in the throwing of an `NSInvalidArgumentException` exception. Depending on the value you assign to the `mediaTypes` (page 11) property, the default controls display the still camera or movie camera interface, or a selection control that lets the user choose the picker interface.

Availability
Available in iOS 3.1 and later.

See Also
`@property cameraOverlayView` (page 9)
`– takePicture` (page 17)

Declared In
`UIImagePickerController.h`

sourceType

The type of picker interface displayed by the controller.

```
@property(nonatomic) UIImagePickerControllerSourceType sourceType
```

Discussion
Prior to running the picker interface, set this value to the desired source type. The specified source type must be available and an exception is thrown if it is not. If you change this property while the picker is visible, the picker interface changes to match the new value in this property.

The various source types are listed in the "UIImagePickerControllerSourceType" (page 18) enumeration. The default value is `UIImagePickerControllerSourceTypePhotoLibrary`.

Availability
Available in iOS 2.0 and later.

See Also
`+ isSourceTypeAvailable:` (page 16)

Declared In
`UIImagePickerController.h`

videoMaximumDuration

The maximum duration, in seconds, for a video recording.

```
@property(nonatomic) NSTimeInterval videoMaximumDuration
```

Discussion
The default value for this property is 10 minutes (600 seconds). When a user taps the Share button to send a movie to MMS, MobileMe, YouTube, or another destination, an appropriate duration limit and an appropriate video quality are enforced.

This property is available only if the `mediaTypes` (page 11) property's value array includes the `kUTTypeMovie` media type.

Availability
Available in iOS 3.1 and later.

See Also
`+ availableMediaTypesForSourceType:` (page 14)
`+ isSourceTypeAvailable:` (page 16)

Declared In
`UIImagePickerController.h`

videoQuality

The video recording and transcoding quality.

```
@property(nonatomic) UIImagePickerControllerQualityType videoQuality
```

Discussion
The video quality setting specified by this property is used during video recording. It is also used whenever picking a recorded movie. Specifically, if the video quality setting is lower than the video quality of an existing movie, displaying that movie in the picker results in transcoding the movie to the lower quality.

The various video qualities are listed in the "`UIImagePickerControllerQualityType`" (page 19) enumeration. The default value is `UIImagePickerControllerQualityTypeMedium` (page 19). To capture or transcode a movie using a video quality other than the default value, you must set the quality explicitly.

This property is available only if the `mediaTypes` (page 11) property's value array includes the `kUTTypeMovie` media type.

Availability
Available in iOS 3.1 and later.

See Also
`+ availableMediaTypesForSourceType:` (page 14)
`+ isSourceTypeAvailable:` (page 16)

Declared In
`UIImagePickerController.h`

Class Methods

availableCaptureModesForCameraDevice:

Returns an array of `NSNumber` objects indicating the capture modes supported by a given camera device.

```
+ (NSArray
    *)availableCaptureModesForCameraDevice:(UIImagePickerControllerCameraDevice)cameraDevice
```

Parameters

cameraDevice

A "UIImagePickerControllerCameraDevice" (page 20) constant indicating the camera you want to interrogate.

Return Value

An array of `NSNumber` objects indicating the capture modes supported by *cameraDevice*.

Discussion

See "UIImagePickerControllerCameraCaptureMode" (page 20) for possible values.

Availability

Available in iOS 4.0 and later.

See Also

`@property cameraCaptureMode` (page 8)
`+ availableCaptureModesForCameraDevice:` (page 14)

Declared In

`UIImagePickerController.h`

availableMediaTypesForSourceType:

Returns an array of the available media types for the specified source type.

```
+ (NSArray
    *)availableMediaTypesForSourceType:(UIImagePickerControllerSourceType)sourceType
```

Parameters

sourceType

The source to use to pick an image.

Return Value

An array whose elements identify the available media types for the specified source type.

Discussion

Some iOS devices support video recording. Use this method, along with the `isSourceTypeAvailable:` (page 16) method, to determine if video recording is available on a device. The availability of video recording is indicated by the presence of the `kUTTypeMovie` media type for the `UIImagePickerControllerSourceTypeCamera` (page 18) source type.

Availability

Available in iOS 3.0 and later.

See Also
`+ isSourceTypeAvailable:` (page 16)

Declared In
`UIImagePickerController.h`

isCameraDeviceAvailable:

Returns a Boolean value that indicates whether a given camera is available.

```
+ (BOOL)isCameraDeviceAvailable:(UIImagePickerControllerCameraDevice)cameraDevice
```

Parameters

cameraDevice
A "UIImagePickerControllerCameraDevice" (page 20) constant indicating the camera whose availability you want to check.

Return Value
`YES` if the camera indicated by *cameraDevice* is available, or `NO` if it is not available.

Availability
Available in iOS 4.0 and later.

See Also
`@property cameraDevice` (page 9)
`+ isFlashAvailableForCameraDevice:` (page 15)
`+ availableCaptureModesForCameraDevice:` (page 14)

Declared In
`UIImagePickerController.h`

isFlashAvailableForCameraDevice:

Indicates whether a given camera has flash illumination capability.

```
+
    (BOOL)isFlashAvailableForCameraDevice:(UIImagePickerControllerCameraDevice)cameraDevice
```

Parameters

cameraDevice
A "UIImagePickerControllerCameraDevice" (page 20) constant indicating the camera whose flash capability you want to know.

Return Value
`YES` if *cameraDevice* can use flash illumination, or `NO` if it cannot.

Availability
Available in iOS 4.0 and later.

See Also
`@property cameraDevice` (page 9)
`@property cameraFlashMode` (page 9)

Declared In
`UIImagePickerController.h`

isSourceTypeAvailable:

Returns a Boolean value indicating whether the device supports picking media using the specified source type.

```
+ (BOOL)isSourceTypeAvailable:(UIImagePickerControllerSourceType)sourceType
```

Parameters

`sourceType`
The source to use to pick an image or movie.

Return Value
`YES` if the device supports the specified source type; `NO` if the specified source type is not available.

Discussion
Because a media source may not be present or may be unavailable, devices may not always support all source types. For example, if you attempt to pick an image from the user's library and the library is empty, this method returns `NO`. Similarly, if the camera is already in use, this method returns `NO`.

Before attempting to use an `UIImagePickerController` object to pick an image, you must call this method to ensure that the desired source type is available.

Availability
Available in iOS 2.0 and later.

See Also
`+ availableMediaTypesForSourceType:` (page 14)

Declared In
`UIImagePickerController.h`

Instance Methods

startVideoCapture

Starts video capture using the camera specified by the `UIImagePickerControllerCameraDevice` (page 20) property.

```
- (BOOL)startVideoCapture
```

Return Value
`YES` on success or `NO` on failure. This method may return a value of `NO` for various reasons, among them the following:

- Movie capture is already in progress
- The device does not support movie capture
- The device is out of disk space

Discussion
Use this method in conjunction with a custom overlay view to initiate the programmatic capture of a movie. You can take more than one movie without leaving the interface, but to do so requires you to hide the default image picker controls.

Calling this method while a movie is being captured has no effect. You must can the `stopVideoCapture` (page 17) method, and then wait until the associated delegate object receives an `imagePickerController:didFinishPickingMediaWithInfo:` message, before you can capture another movie.

Calling this method when the source type of the image picker is set to a value other than `UIImagePickerControllerSourceTypeCamera` (page 18) results in the throwing of an `NSInvalidArgumentException` exception.

If you require additional options or more control over movie capture, use the movie capture methods in the AV Foundation framework. Refer to *AV Foundation Framework Reference*.

Availability
Available in iOS 4.0 and later.

Declared In
`UIImagePickerController.h`

stopVideoCapture

Stops video capture.

```
- (void)stopVideoCapture
```

Discussion
After you call this method to stop video capture, the system calls the image picker delegate's `imagePickerController:didFinishPickingMediaWithInfo:` method.

Availability
Available in iOS 4.0 and later.

Declared In
`UIImagePickerController.h`

takePicture

Captures a still image using the camera.

```
- (void)takePicture
```

Discussion
Use this method in conjunction with a custom overlay view to initiate the programmatic capture of a still image. This supports taking more than one picture without leaving the interface, but requires that you hide the default image picker controls.

Calling this method while an image is being captured has no effect. You must wait until the associated delegate object receives an `imagePickerController:didFinishPickingMediaWithInfo:` message before you can capture another picture.

Calling this method when the source type of the image picker is set to a value other than `UIImagePickerControllerSourceTypeCamera` results in the throwing of an `NSInvalidArgumentException` exception.

Availability
Available in iOS 3.1 and later.

See Also
`@property cameraOverlayView` (page 9)

Declared In
`UIImagePickerController.h`

Constants

UIImagePickerControllerSourceType

The source to use when picking an image.

```
enum {
    UIImagePickerControllerSourceTypePhotoLibrary,
    UIImagePickerControllerSourceTypeCamera,
    UIImagePickerControllerSourceTypeSavedPhotosAlbum
};
typedef NSUInteger UIImagePickerControllerSourceType;
```

Constants

`UIImagePickerControllerSourceTypePhotoLibrary`

Pick an image or movie, as available, from the device's photo library.

Available in iOS 2.0 and later.

Declared in `UIImagePickerController.h`.

`UIImagePickerControllerSourceTypeCamera`

Take a new picture or movie, as available, using the device's specified, built-in camera. Specify the camera you want by using the `cameraDevice` (page 9) property.

Available in iOS 2.0 and later.

Declared in `UIImagePickerController.h`.

`UIImagePickerControllerSourceTypeSavedPhotosAlbum`

Pick an image or movie, as available, from the device's camera roll. If the device does not have a camera, pick an image or movie, as available, from the Saved Photos folder on the device.

Available in iOS 2.0 and later.

Declared in `UIImagePickerController.h`.

Discussion
A given source may not be available on a given device because the source is not physically present or because it cannot currently be accessed.

UIImagePickerControllerQualityType

Video quality settings for movies recorded with the built-in camera, or transcoded by displaying in the image picker.

```
enum {
    UIImagePickerControllerQualityTypeHigh    = 0,
    UIImagePickerControllerQualityType640x480 = 3,
    UIImagePickerControllerQualityTypeMedium  = 1,  // default value
    UIImagePickerControllerQualityTypeLow     = 2
};
typedef NSUInteger UIImagePickerControllerQualityType;
```

Constants

`UIImagePickerControllerQualityTypeHigh`

If recording, specifies that you want to use the highest-quality video recording supported for the active camera on the device.

Recorded files are suitable for on-device playback and for wired transfer to the Desktop using Image Capture; they are likely to be too large for transfer using Wi-Fi.

If displaying a recorded movie in the image picker, specifies that you do not want to reduce the video quality of the movie.

Available in iOS 3.1 and later.

Declared in `UIImagePickerController.h`.

`UIImagePickerControllerQualityType640x480`

If recording, specifies that you want to use VGA-quality video recording (pixel dimensions of 640x480).

If displaying a recorded movie in the image picker, specifies that you want to transcode higher-quality movies to VGA video quality.

Available in iOS 4.0 and later.

Declared in `UIImagePickerController.h`.

`UIImagePickerControllerQualityTypeMedium`

If recording, specifies that you want to use medium-quality video recording.

Recorded files can usually be transferred using Wi-Fi. This is the default video quality setting.

If displaying a recorded movie in the image picker, specifies that you want to transcode higher-quality movies to medium video quality.

Available in iOS 3.1 and later.

Declared in `UIImagePickerController.h`.

`UIImagePickerControllerQualityTypeLow`

If recording, specifies that you want to use low-quality video recording.

Recorded files can usually be transferred over the cellular network.

If displaying a recorded movie in the image picker, specifies that you want to transcode higher-quality movies to low video quality.

Available in iOS 3.1 and later.

Declared in `UIImagePickerController.h`.

Discussion

The constants in this enumeration are for use as values of the `videoQuality` (page 13) property.

The video quality setting applies to transcoding as well as to recording. Specifically, if the video quality setting is lower than the video quality of an existing movie, displaying that movie in the picker results in transcoding the movie to the lower quality.

UIImagePickerControllerCameraDevice

The camera to use for image or movie capture.

```
enum {
    UIImagePickerControllerCameraDeviceRear,
    UIImagePickerControllerCameraDeviceFront
};
typedef NSUInteger UIImagePickerControllerCameraDevice;
```

Constants

`UIImagePickerControllerCameraDeviceRear`

Specifies the camera on the rear of the device.

Available in iOS 4.0 and later.

Declared in `UIImagePickerController.h`.

`UIImagePickerControllerCameraDeviceFront`

Specifies the camera on the front of the device.

Available in iOS 4.0 and later.

Declared in `UIImagePickerController.h`.

Discussion

The constants in this enumeration are for use as values of the `cameraDevice` (page 9) property.

UIImagePickerControllerCameraCaptureMode

The category of media for the camera to capture.

```
enum {
    UIImagePickerControllerCameraCaptureModePhoto,
    UIImagePickerControllerCameraCaptureModeVideo
};
typedef NSUInteger UIImagePickerControllerCameraCaptureMode;
```

Constants

`UIImagePickerControllerCameraCaptureModePhoto`

Specifies that the camera captures still images.

Available in iOS 4.0 and later.

Declared in `UIImagePickerController.h`.

`UIImagePickerControllerCameraCaptureModeVideo`

Specifies that the camera captures movies.

Available in iOS 4.0 and later.

Declared in `UIImagePickerController.h`.

Discussion

The constants in this enumeration are for use as values of the `cameraCaptureMode` (page 8) property.

UIImagePickerControllerCameraFlashMode

The flash mode to use with the active camera.

```
enum {
    UIImagePickerControllerCameraFlashModeOff  = -1,
    UIImagePickerControllerCameraFlashModeAuto = 0,
    UIImagePickerControllerCameraFlashModeOn   = 1
};
typedef NSInteger UIImagePickerControllerCameraFlashMode;
```

Constants

`UIImagePickerControllerCameraFlashModeOff`

Specifies that flash illumination is always off, no matter what the ambient light conditions are.

Available in iOS 4.0 and later.

Declared in `UIImagePickerController.h`.

`UIImagePickerControllerCameraFlashModeAuto`

Specifies that the device should consider ambient light conditions to automatically determine whether or not to use flash illumination.

Available in iOS 4.0 and later.

Declared in `UIImagePickerController.h`.

`UIImagePickerControllerCameraFlashModeOn`

Specifies that flash illumination is always on, no matter what the ambient light conditions are.

Available in iOS 4.0 and later.

Declared in `UIImagePickerController.h`.

Discussion

The constants in this enumeration are for use as values of the `cameraFlashMode` (page 9) property.

The behavior of the flash depends on the camera capture mode.

- For a `cameraCaptureMode` (page 8) value of `UIImagePickerControllerCameraCaptureModePhoto` (page 20), flash is used to transiently illuminate the subject during still image capture.
- For a `cameraCaptureMode` (page 8) value of `UIImagePickerControllerCameraCaptureModeVideo` (page 20), flash is used to continuously illuminate the subject during movie capture.

For a given camera on a device, flash may or may not be available. You specify the active camera by way of the `cameraDevice` (page 9) property. You can determine if the active camera has flash available by calling the `isFlashAvailableForCameraDevice:` (page 15) class method.

You can manipulate the flash directly to provide effects such as a strobe light. Present a picker interface set to use video capture mode. Then, turn the flash LED on or off by setting the `cameraFlashMode` property to `UIImagePickerControllerCameraFlashModeOn` or `UIImagePickerControllerCameraFlashModeOff`.

Deprecated UIImagePickerController Methods

A method identified as deprecated has been superseded and may become unsupported in the future.

Deprecated in iOS 3.1

allowsImageEditing

A Boolean value indicating whether the user is allowed to edit a selected image. (Deprecated in iOS 3.1. Use `allowsEditing` (page 8) instead.)

```
@property(nonatomic) BOOL allowsImageEditing
```

Discussion
If you allow the user to edit images, the delegate may receive a dictionary with information about the edits that were made.

This property is set to `NO` by default.

Availability
Available in iOS 2.0 and later.
Deprecated in iOS 3.1.

Declared In
`UIImagePickerController.h`

Document Revision History

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

Date	Notes
2010-06-29	Added descriptions of new APIs and behaviors for iOS 4 and iPhone 4.
	Described the new iOS 4 behavior for the `showsCameraControls` (page 11) property.
	Improved the descriptions of the `startVideoCapture` (page 16) and `stopVideoCapture` (page 17) instance methods.
	Improved the description of the "UIImagePickerControllerCameraFlashMode" (page 21) enumeration.
	Updated the description for the "UIImagePickerControllerSourceType" (page 18) enumeration.
	Improved the descriptions for the `videoQuality` (page 13) property and for the "UIImagePickerControllerQualityType" (page 19) enumeration by noting that they apply to movie transcoding as well as to recording.
	Improved the description for the `mediaTypes` (page 11) property by noting that attempting to set an empty array results in an exception.
2010-05-18	Added descriptions for the "UIImagePickerControllerCameraDevice" (page 20), "UIImagePickerControllerCameraCaptureMode" (page 20), and "UIImagePickerControllerCameraFlashMode" (page 21) enumerations.
	Added descriptions for two video capture instance methods: `startVideoCapture` (page 16) and `stopVideoCapture` (page 17).
	Added description for VGA-quality video recording constant, `UIImagePickerControllerQualityType640x480` (page 19).
	Updated UIImagePickerController (page 5) "Overview" for iOS 4.0.
2009-10-19	Updated for iOS 3.1.
	Added an "Important" section to UIImagePickerController (page 5) class overview, clarifying use of the class.
	Added descriptions for the `videoMaximumDuration` (page 12) and `videoQuality` (page 13) properties in the "Configuring the Video Capture Options" (page 7) task group.

Date	Notes
	Added descriptions for the `showsCameraControls` (page 11), `cameraOverlayView` (page 9), and `cameraViewTransform` (page 10) properties, and for the `takePicture` (page 17) instance method, to the "Customizing the Camera Controls" (page 7) task group.
	Added a description for the `allowsEditing` (page 8) property in the "Configuring the Picker" (page 6) task group.
	Added a description for "UIImagePickerControllerQualityType" (page 19) enumeration.
	Added deprecation information for the `allowsImageEditing` (page 23) property.
2009-06-02	Updated for iOS 3.0
	Added description of video recording support.
2008-05-27	New document that describes the class for managing the system-supplied user interfaces for choosing and taking pictures.