カメラ操作のプログラミング(iOS用)

Developer

目次

図、リスト

カメラとフォトライブラリについて

iOSは、写真やムービーを撮影するための2つのテクノロジーを提供しています。

- `UIImagePickerController`クラス。写真やムービーを撮影したり、新たにキャプチャしたメディアに対する簡単な編集機能をユーザに提供するための、カスタマイズ可能な基本的なユーザインターフェイスを提供します。完全に独自のソリューションが必要でない場合は、Image Picker Controllerを使用してください。
- AV Foundationフレームワーク。柔軟で強力な各種のクラスを提供します。これらのクラスをUIKitと一緒に使用して、完全にカスタマイズした静止画像やムービーのキャプチャ機能をアプリケーション向けに作成することができます。

どちらを採用する場合も、Assets Libraryフレームワークを使用して、メディアのメタデータ（GPS位置情報など）を管理できます。

同様に、iOSは、保存済みの写真やムービーをユーザのフォトアルバムから選択するユーザインターフェイスを提供するために、2つのテクノロジーを提供しています。

- `UIImagePickerController`オブジェクト。システムが提供する基本的なユーザインターフェイスを使用してユーザがフォトライブラリからメディアを選択できるようにするには、このオブジェクトをメディアブラウザとしてインスタンス化します。
- Assets Libraryフレームワーク。Assets LibraryフレームワークをUIKitと一緒に使用して、完全にカスタマイズした写真やムービーのブラウザを作成できます。

このドキュメントでは、Image Picker Controllerを使用して、写真やムービーの撮影と、保存済みのメディアの選択を行う方法を説明します。このドキュメントで説明するとおり、2つの作業の手順は非常によく似ています。

AV Foundationフレームワークを使用して完全にカスタマイズされたメディアキャプチャ機能を作成する方法については、『*AV Foundation Programming Guide*』の"Media Capture"を参照してください。

完全にカスタマイズしたメディアブラウザの作成をサポートするAssets Libraryフレームワークの詳細については、『*Assets Library Framework Reference*』を参照してください。

この書類の構成

このドキュメントは次の項目で構成されています。

- “写真とムービーの撮影”（6 ページ）では、Image Picker Controllerをカメラインターフェイスでインスタンス化する方法と、ユーザが写真やムービーを撮影したときに新たにキャプチャしたメディアを取得する方法について説明します。
- “フォトライブラリからのアイテムの選択”（13 ページ）では、Image Picker Controllerをメディアブラウザとして使用して、ユーザがデバイスのフォトライブラリから項目を選択できるようにする方法を説明します。

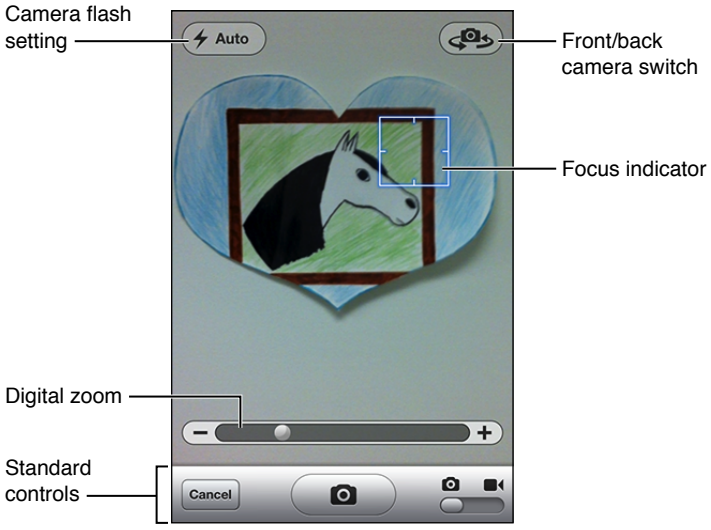
写真とムービーの撮影

Image Picker Controllerを利用した写真やムービーの撮影は、アプリケーションコードとシステムがやり取りをしながら進行する以下の3つのプロセスで構成されます。

1. カメラインターフェイス（`UIImagePickerController`クラスのインスタンス）をインスタンス化してモーダルモードで表示します。
2. システムは、このカメラインターフェイスとユーザとのやり取りを管理します。典型的な使いかたは、ユーザが写真やムービーを撮影するか、その操作をキャンセルするかのいずれかです。
3. システムはImage Picker Controllerのデリゲートオブジェクトのメソッドを呼び出します。今度は、このメソッドがユーザの操作（たとえば、新規の写真を「カメラロール（Camera Roll）」アルバムに保存する）の結果を処理します。また、カメラインターフェイスを消去するのもデリゲートです。

デフォルトのImage Picker Controllerには、図 1に示すように、さまざまな機能が含まれています。

図 1　Image Picker Controller



この章では、写真やムービーを撮影するために、デフォルトのImage Picker Controllerとデリゲートを使用する方法について説明します（ムービーの録画は、iOS 3.0から対応デバイス上で可能になりました）。

Important: `UIImagePickerController`クラスは、縦長モードのみをサポートします。このクラスは、そのまま使用するように設計されているため、サブクラス化はできません。このクラスのビュー階層はプライベートであるため、変更してはなりません。ただし1つ例外があり、iOS 3.1以降では、カスタムビューを`cameraOverlayView`プロパティに割り当て、そのビューを使用して追加情報を表示したり、カメラインターフェイスとコードとの間のやり取りを管理することができます。たとえば、デフォルトのツールバーとその標準コントロールを独自のものに置き換えることができます。詳細については、サンプルコードプロジェクト『*PhotoPicker*』を参照してください。

AV Foundationフレームワークを使用して完全にカスタマイズしたメディアキャプチャ機能を作成する方法については、『*AV Foundation Programming Guide*』の「Media Capture and Access to Camera」 in *AV Foundation Programming Guide*を参照してください。

カメラインターフェイスの作成と設定

カメラインターフェイスを表示するには、まず次の3つの条件が整っていなければなりません。

1. アプリケーションが実行するデバイスに、カメラが付いていること。

   アプリケーションにとって写真やムービーの撮影が不可欠な場合は、そのアプリケーションの`Info.plist`プロパティリストファイルの`UIRequiredDeviceCapabilities`キーを設定してそのことを明記します。必須として指定できる各種のカメラ特性については、『*Information Property List Key Reference*』の「UIRequiredDeviceCapabilities」 in *Information Property List Key Reference*を参照してください。

   アプリケーションにとってメディアのキャプチャが付随的な場合（つまり、たとえデバイスにカメラが付いていなくても、アプリケーションが問題なく動作する場合）、カメラのないデバイス上で実行する場合にたどる代替パスを、コードで提供する必要があります。

2. デバイスのカメラを利用できること。これは、`UIImagePickerController`クラスの`isSourceTypeAvailable:`クラスメソッドを通じてテストできます。

3. ユーザとImage Picker Controllerとの対話に応答するデリゲートオブジェクトを実装すること。（詳細については、“カメラインターフェイス用のデリゲートの実装”（10 ページ）を参照）。

これらの前提条件が満たされていれば、以下のオプションを指定することによって、Image Picker Controllerを作成、設定できます。

- **ソースタイプ** — 保存済みメディアの閲覧ではなく、メディアのキャプチャ用にピッカーを設定するには、`sourceType`プロパティを`UIImagePickerControllerSourceTypeCamera`に設定します。

> **注意:** 必ず、UIImagePickerControllerクラスのisSourceTypeAvailable:クラスメソッドを呼び出して、その戻り値を確認してください。デバイスにカメラが付いていることを前提にしてはいけません。たとえデバイスにカメラが付いていても、そのカメラが利用できない場合、このメソッドはNOを返します。

- **メディアタイプ** — 静止画像、ムービー、またはその両方を撮影するかどうかを指定するには、mediaTypesプロパティを希望のタイプの識別子を含む配列に設定します。この配列の要素として有効な値は、kUTTypeImageとkUTTypeMovieです。

  ただし、このプロパティを設定する前に、availableMediaTypesForSourceType:クラスメソッドを呼び出して、利用可能なメディアタイプをチェックします。mediaTypesプロパティを、空の配列や現在のソースで利用できるメディアタイプを含まない配列に設定した場合、例外が発生します。
- **編集用コントロール** — キャプチャ済みの写真の移動と拡大縮小、またはキャプチャ済みのムービーのトリミングを行うためのユーザコントロールをカメラインターフェイスが提供するかどうかを指定するには、allowsEditingプロパティをYES（編集用コントロールを提供する場合）またはNOに設定します。

  標準で組み込まれている編集用コントロールを使用する場合は、Image Picker Controllerの特定のオプションが強制的に設定されます。静止画像の場合、ピッカーは正方形トリミングと最大ピクセル寸法を強制的に設定します。ムービーの場合、ピッカーはムービーの最大長と解像度を強制的に設定します。フルサイズのメディアの編集や、カスタムトリミングの指定をユーザ側でできるようにする場合は、独自の編集用UIを提供する必要があります。
- **デリゲートオブジェクト** — 最後に、デリゲートオブジェクトをImage Picker Controllerのdelegateプロパティに割り当てます。

リスト1では、メソッドのシグネチャと条件テストを通じて、これらの前提条件が満たされているかどうかを確認してから、カメラユーザインターフェイスのフルスクリーン表示のインスタンス化、設定、非同期化を実行します。

リスト 1　カメラインターフェイスのフルスクリーン表示

```
- (BOOL) startCameraControllerFromViewController: (UIViewController*) controller
                usingDelegate: (id <UIImagePickerControllerDelegate,
                                    UINavigationControllerDelegate>) delegate {

    if (([UIImagePickerController isSourceTypeAvailable:
                  UIImagePickerControllerSourceTypeCamera] == NO)
```

```
            || (delegate == nil)
            || (controller == nil))
        return NO;


    UIImagePickerController *cameraUI = [[UIImagePickerController alloc] init];
    cameraUI.sourceType = UIImagePickerControllerSourceTypeCamera;

    // ユーザが写真またはムービーのキャプチャを選択するためのコントロールを表示する
    //  (写真とムービーの両方が利用可能な場合)
    cameraUI.mediaTypes =
        [UIImagePickerController availableMediaTypesForSourceType:
            UIImagePickerControllerSourceTypeCamera];

    // 写真の移動と拡大縮小、または
    // ムービーのトリミングのためのコントロールを隠す。代わりにコントロールを表示するには、YESを
使用する。
    cameraUI.allowsEditing = NO;

    cameraUI.delegate = delegate;

    [controller presentModalViewController: cameraUI animated: YES];
    return YES;
}
```

リスト 1では、ユーザが静止画像やムービーをキャプチャできる（デバイス上で両方が利用可能な場合）ピッカーを提供します。たとえば、代わりにムービーだけをキャプチャするピッカーを表示するには、ムービーのキャプチャが利用できることを確認してから、`mediaTypes`プロパティを次のように設定します。

```
cameraUI.mediaTypes = [[NSArray alloc] initWithObjects: (NSString *) kUTTypeMovie,
 nil];
```

（ムービーのキャプチャが利用できることを確認するには、`availableMediaTypesForSourceType:`クラスメソッドを呼び出します）。

静止画像だけをキャプチャするピッカーの場合は、ここで`kUTTypeMovie`識別子を`kUTTypeImage`に置き換えるか、`mediaTypes`プロパティのデフォルトの値（`kUTTypeImage`）を利用します。

リスト1の`startCameraControllerFromViewController:usingDelegate:`メソッドの例は、次のようなアクションメソッドによって呼び出されるように設計されています。

```
- (IBAction) showCameraUI {
    [self startCameraControllerFromViewController: self
                                    usingDelegate: self];
}
```

リスト1（8ページ）のメソッドシグネチャに指定されているとおり、デリゲートオブジェクトは、`UIImagePickerControllerDelegate`プロトコルと`UINavigationControllerDelegate`プロトコルに準拠していなければなりません。

> **注意:** デリゲートが`UINavigationControllerDelegate`プロトコルに準拠していない場合、コンパイル中に警告が表示されることがあります。ただし、このプロトコルのメソッドはオプションであるため、この警告は実行時には影響を与えません。警告を除去するには、デリゲートのサポートプロトコルのリストに`UINavigationControllerDelegate`プロトコルを追加します。

`startCameraControllerFromViewController:usingDelegate:`サンプルメソッドを実行した結果、システムは標準のカメラインターフェイスをモーダルモードで表示します。このインターフェイスには、メディアをキャプチャしたりキャンセルするためのコントロールが含まれています（図1（6ページ）を参照）。

カメラインターフェイス用のデリゲートの実装

新規にキャプチャした写真やムービーを受理したり、その操作をキャンセルするために、ユーザがカメラインターフェイスのボタンをタップすると、システムは、ユーザの選択をデリゲートに通知します。ただし、システムはカメラインターフェイスを消去しません。デリゲートが`dismissModalViewControllerAnimated:`メソッドを呼び出してインターフェイスを消去し、インターフェイスを解放しなければなりません。その理由は、通常、カメラインターフェイスを表示するView Controllerは、デリゲートとしての役割も兼ねているためです。

リスト 2に、Image Picker Controller用のデリゲートメソッドの実装例を示します。imagePickerController:didFinishPickingMediaWithInfo:の実装には、ユーザがキャプチャしたメディアに応じて静止画像またはムービーを保存するコードが含まれています。

リスト 2　カメラインターフェイスのデリゲートメソッド

```
@implementation CameraViewController (CameraDelegateMethods)

// 「キャンセル(Cancel)」をタップしたユーザへの応答.
- (void) imagePickerControllerDidCancel: (UIImagePickerController *) picker {

    [[picker parentViewController] dismissModalViewControllerAnimated: YES];
    [picker release];
}

// 新規にキャプチャした写真やムービーを受理したユーザへの応答
- (void) imagePickerController: (UIImagePickerController *) picker
            didFinishPickingMediaWithInfo: (NSDictionary *) info {

    NSString *mediaType = [info objectForKey: UIImagePickerControllerMediaType];
    UIImage *originalImage, *editedImage, *imageToSave;

    // 静止画像のキャプチャを処理する
    if (CFStringCompare ((CFStringRef) mediaType, kUTTypeImage, 0)
            == kCFCompareEqualTo) {

        editedImage = (UIImage *) [info objectForKey:
                    UIImagePickerControllerEditedImage];
        originalImage = (UIImage *) [info objectForKey:
                    UIImagePickerControllerOriginalImage];

        if (editedImage) {
            imageToSave = editedImage;
        } else {
            imageToSave = originalImage;
        }
```

```
    // （オリジナルまたは編集後の）新規画像を「カメラロール(Camera Roll)」に保存する
        UIImageWriteToSavedPhotosAlbum (imageToSave, nil, nil , nil);
    }

    // ムービーのキャプチャを処理する
    if (CFStringCompare ((CFStringRef) mediaType, kUTTypeMovie, 0)
            == kCFCompareEqualTo) {

        NSString *moviePath = [[info objectForKey:
                    UIImagePickerControllerMediaURL] path];

        if (UIVideoAtPathIsCompatibleWithSavedPhotosAlbum (moviePath)) {
            UISaveVideoAtPathToSavedPhotosAlbum (
                    moviePath, nil, nil, nil);
        }
    }

    [[picker parentViewController] dismissModalViewControllerAnimated: YES];
    [picker release];
}

@end
```

画像の編集が有効になっていて、ユーザが新規にキャプチャした写真を正常に受理した場合、`imagePickerController:didFinishPickingMediaWithInfo:`メソッドの`info`パラメータに編集後の画像を含む辞書が含まれます。リスト 2で実行したとおり、この画像を選択された画像として扱います。オリジナルの画像を保存したい場合は、このコードリストで示したように、辞書からそれを取得できます。

この例に示すように、新規のメディア項目を直ちにユーザの「カメラロール（Camera Roll）」に保存する代わりに、そのメディアを処理するカスタムコードを呼び出すこともできます。

フォトライブラリからのアイテムの選択

`UIImagePickerController`インスタンスを使用すると、新規の写真やムービーをキャプチャすること以外に、保存済みのフォトアルバムからユーザがアイテムを選択できるメディアブラウザを表示できます。その手順は、“写真とムービーの撮影” （6 ページ）で説明した、メディアをキャプチャする手順に似ています。ただし、次の点が異なります。

- メディアソースとしてカメラではなく、「Camera Roll（カメラロール）」アルバムか「Saved Photos（保存された写真)」アルバム、またはフォトライブラリ全体を使用します。
- 新規のメディアをキャプチャして（通常は）アルバムに保存する代わりに、以前保存したメディアをユーザに選択させます。その後、アプリケーションで選択したメディアを、おそらくフルスクリーンで表示しながら使用できます。

この章では、保存済みの写真やムービーの閲覧と選択を行うために、Image Picker Controllerとデリゲートを使用する方法について説明します。標準のメディア閲覧用UIが要件に合わない場合は、UIKitとAssets Libraryフレームワークを使用して、完全に独自のソリューションを作成することもできます。詳細については、『*Assets Library Framework Reference*』を参照してください。

メディアブラウザの作成と設定

ほとんどのiOSデバイスにはフォトライブラリがあります。そのようなデバイスでは、デバイスにカメラが付いているかどうかに関わらず、Image Picker Controllerを使用して、メディアブラウザを表示できます。カメラインターフェイスを表示する場合と同様に、ブラウザインターフェイスでのユーザの操作に応答するには、デリゲートオブジェクトを実装する必要があります。その後、Image Picker Controllerをインスタンス化し、次のオプションを指定して設定します。

- **ソースタイプ** — 新規の写真やムービーをキャプチャするのではなく、保存済みのメディアを閲覧するためにピッカーを設定するには、`sourceType`プロパティを保存済み写真ソースの1つに設定します。
    - デバイス上のすべてのフォトアルバム（カメラ付きのデバイス上の「Camera Roll（カメラロール）」アルバムを含む）にアクセスするブラウザを表示するには、`UIImagePickerControllerSourceTypePhotoLibrary`を使用します。

- カメラ付きデバイス上では「Camera Roll（カメラロール）」アルバムに、またはカメラが付いていないデバイス上では「Saved Photos(保存された写真)」アルバムに限定してアクセスするブラウザを表示するには、`UIImagePickerControllerSourceTypeSavedPhotosAlbum`を使用します。

  注意: カメラピッカーの場合と同様に、必ず`UIImagePickerController`クラスの`isSourceTypeAvailable:`クラスメソッドを呼び出して、その戻り値を確認します。デバイスにフォトライブラリがあることを前提にしてはいけません。たとえ、デバイスにライブラリが存在しても、そのライブラリが現在利用できない場合は、このメソッドは`NO`を返します。

- **メディアタイプ** — 保存済みの静止画像、ムービー、またはその両方をImage Picker Controllerで表示するどうかを指定するには、`mediaTypes`プロパティを、希望のタイプの識別子を含む配列に設定します。この配列の要素として有効な値は、`kUTTypeImage`と`kUTTypeMovie`です。

  ただし、このプロパティを設定する前に、`availableMediaTypesForSourceType:`クラスメソッドを呼び出して、利用可能なメディアタイプをチェックします。`mediaTypes`プロパティを、空の配列や現在のソースで利用できるメディアタイプを含まない配列に設定した場合、例外が発生します。

- **編集用コントロール** — 保存済みの写真の移動と拡大縮小、または保存済みのムービーのトリミングを行うためのユーザコントロールをメディアピッカーが提供するかどうかを指定するには、`allowsEditing`プロパティを`YES`（編集用コントロールを提供する場合）または`NO`に設定します。

  標準で組み込まれている編集用コントロールを使用する場合は、Image Picker Controllerの特定のオプションが強制的に設定されます。静止画像の場合、ピッカーは正方形トリミングと最大ピクセル寸法を強制的に設定します。ムービーの場合、ピッカーはムービーの最大長と解像度を強制的に設定します。フルサイズのメディアの編集や、カスタムトリミングの指定をユーザ側でできるようにする場合は、独自の編集用UIを提供する必要があります。

- **デリゲートオブジェクト** — 最後に、デリゲートオブジェクトをImage Picker Controllerの`delegate`プロパティに割り当てます。

リスト1では、メソッドのシグネチャと条件テストを通じて、これらの前提条件が満たされているかどうかを確認してから、メディアブラウザインターフェイスのフルスクリーン表示のインスタンス化、設定、非同期化を実行します。

リスト 1　iPhoneまたはiPod touchにおける、メディアブラウザインターフェイスのフルスクリーン表示

```
- (BOOL) startMediaBrowserFromViewController: (UIViewController*) controller
                usingDelegate: (id <UIImagePickerControllerDelegate,
```

```
                                UINavigationControllerDelegate>) delegate {

    if (([UIImagePickerController isSourceTypeAvailable:
                UIImagePickerControllerSourceTypeSavedPhotosAlbum] == NO)
            || (delegate == nil)
            || (controller == nil))
        return NO;

    UIImagePickerController *mediaUI = [[UIImagePickerController alloc] init];
    mediaUI.sourceType = UIImagePickerControllerSourceTypeSavedPhotosAlbum;

    // 「Camera Roll(カメラロール)」アルバムから、保存されている写真とムービーを
    // 表示する（両方が利用可能な場合）
    mediaUI.mediaTypes =
        [UIImagePickerController availableMediaTypesForSourceType:
            UIImagePickerControllerSourceTypeSavedPhotosAlbum];

    // 写真の移動と拡大縮小、または
    // ムービーのトリミングのためのコントロールを隠す。代わりにコントロールを表示するには、YESを
使用する。
    mediaUI.allowsEditing = NO;

    mediaUI.delegate = delegate;

    [controller presentModalViewController: mediaUI animated: YES];
    return YES;
}
```

iPadの場合、『*UIPopoverController Class Reference*』の`initWithContentViewController:`と「Presenting and Dismissing the Popover」 in *UIPopoverController Class Reference* で説明されているとおり、Popoverを使用してブラウザインターフェイスを表示しなければなりません。iPad上で、ブラウザインターフェイスをモーダルに（フルスクリーンで）表示しようとすると、例外が発生します。

リスト 1 （14 ページ）では、静止画像とムービーの両方が「Camera Roll(カメラロール)」アルバムに存在する場合、その両方をピッカーに表示します。たとえば、ムービーだけを表示するピッカーを提供するには、`mediaTypes`プロパティを次のように設定します。

```
mediaUI.mediaTypes = [[NSArray alloc] initWithObjects: (NSString *) kUTTypeMovie,
 nil];
```

このオプションを使用して、まずデバイスがムービーを表示できることを確認する必要があります。それには、`availableMediaTypesForSourceType:`クラスメソッドを呼び出します。静止画像だけを表示するピッカーの場合は、ここで`kUTTypeMovie`識別子を`kUTTypeImage`に置き換えるか、`mediaTypes`プロパティのデフォルトの値（`kUTTypeImage`）を利用します。

リスト1の`startMediaBrowserFromViewController:usingDelegate:`メソッドの例は、次のようなアクションメソッドによって呼び出されるように設計されています。

```
- (IBAction) showSavedMediaBrowser {
    [self startMediaBrowserFromViewController: self
                                usingDelegate: self];
}
```

リスト1（14ページ）のメソッドシグネチャで指定されているとおり、デリゲートオブジェクトは、`UIImagePickerControllerDelegate`プロトコルと`UINavigationControllerDelegate`プロトコルに準拠していなければなりません。

> **注意:** デリゲートが`UINavigationControllerDelegate`プロトコルに準拠していない場合、コンパイル中に警告が表示されることがあります。ただし、このプロトコルのメソッドはオプションであるため、この警告は実行時には影響を与えません。警告を除去するには、デリゲートのクラスのサポートプロトコルのリストに`UINavigationControllerDelegate`プロトコルを追加します。

メディアブラウザ用のデリゲートの実装

Image Picker Controllerの標準メディアブラウザを利用すると、「Camera Roll（カメラロール）」アルバムのコンテンツを表すサムネイルのスクロール可能な一覧が表示されます。ユーザは、操作をキャンセルするか、サムネイルの1つをタップするかのいずれかを選択できます。ユーザが「キャンセル（キャンセル）」をタップした場合、デリゲートの実装は、カメラインターフェイスを表示している場合とまったく同様に、単純にピッカーを消去しなければなりません。実際、Image Picker ControllerのimagePickerControllerDidCancel:の実装は、カメラインターフェイスを表示しているか、メディアブラウザインターフェイスを表示しているかに関わらず同じです。詳細については、リスト2（11ページ）を参照してください。

ユーザがサムネイルをタップした場合は、そのサムネイルが静止画像を表示するか、ムービーを表示するかによって、その次の動作が異なります。

- 静止画像の場合は、サムネイルをタップすると、直ちにその画像の情報と共にデリゲートが呼び出されます。
- ムービーの場合は、「再生（play）」、「キャンセル（Cancel）」、「選択(Choose)」の3つのボタンのあるスクロール可能なプレビューが表示されます。ユーザが「選択(Choose)」をタップすると、システムはそのムービーについての情報と共にデリゲートを呼び出します。ユーザが「キャンセル(Cancel)」をタップすると、プレビューは自動的に消去されて、メディアブラウザに戻ります。

メディアブラウザとして設定されたImage Picker Controllerで「選択(Choose)」をタップしたユーザに応答するには、次のリストに示すようなコードを使用します。

リスト 2　選択されたメディアに対応するデリゲートメソッド

```
- (void) imagePickerController: (UIImagePickerController *) picker
            didFinishPickingMediaWithInfo: (NSDictionary *) info {

    NSString *mediaType = [info objectForKey: UIImagePickerControllerMediaType];
    UIImage *originalImage, *editedImage, *imageToUse;

    // フォトアルバムから選択された静止画像を処理する
    if (CFStringCompare ((CFStringRef) mediaType, kUTTypeImage, 0)
            == kCFCompareEqualTo) {

        editedImage = (UIImage *) [info objectForKey:
                    UIImagePickerControllerEditedImage];
        originalImage = (UIImage *) [info objectForKey:
                    UIImagePickerControllerOriginalImage];

        if (editedImage) {
            imageToUse = editedImage;
        } else {
            imageToUse = originalImage;
        }
        // imageToUseを利用して何らかの処理を行う
```

```
    }

    // フォトアルバムから選択されたムービーを処理する
    if (CFStringCompare ((CFStringRef) mediaType, kUTTypeMovie, 0)
            == kCFCompareEqualTo) {

        NSString *moviePath = [[info objectForKey:
                    UIImagePickerControllerMediaURL] path];

        // moviePathで利用可能な選択されたムービーに対して何らかの処理を行う
    }

    [[picker parentViewController] dismissModalViewControllerAnimated: YES];
    [picker release];
}
```

画像の編集が有効になっていて、ユーザが新規にキャプチャした写真を正常に受理した場合、`imagePickerController:didFinishPickingMediaWithInfo:`メソッドの`info`パラメータに編集後の画像を含む辞書が含まれます。リスト 2で実行したとおり、この画像を選択された画像として扱います。オリジナルの画像を保存したい場合は、このコードリストで示したように、辞書からそれを取得できます。

書類の改訂履歴

この表は「*カメラ操作のプログラミング（iOS用）について*」の改訂履歴です。

日付	メモ
2012-07-17	"メディアブラウザの作成と設定"（13 ページ）で、iPad上にメディアブラウザインターフェイスを表示する方法に関する情報を修正しました。
2011-03-08	iPadでImage Picker Controllerを提供する2つの方法についての情報を追加しました。
2010-11-15	静止画およびビデオのキャプチャを行う方法と、フォトアルバムからアイテムを選択する方法について説明した新規ドキュメント。 このドキュメントの内容の一部は、以前は『*Device Features Programming Guide*』に記載されていたものです。

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
U.S.A.

Apple Japan
〒106-6140 東京都港区六本木 6
丁目10番1号 六本木ヒルズ
http://www.apple.com/jp