iPhoneヒューマンインターフェイスガイドライン

Internet & Web > Web Content

2010-06-03

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

アップルジャパン株式会社
〒163-1450 東京都新宿区西新宿
3 丁目20 番2 号
東京オペラシティタワー
http://www.apple.com/jp/

目次

図、表

はじめに

iPhoneおよびiPod touchは、革命的なMulti-Touchインターフェイスと、電子メール、インスタントメッセージ機能、フル機能のWebブラウザ、iPod、そしてiPhoneの場合には携帯電話機能まで含む強力な機能を組み合わせた高度なデバイスです。iPhone OSは、iPhoneおよびiPod touch上で実行するシステムソフトウェアです。iPhone SDKの登場によって、これらの強力な機能の対象範囲が広がり、デベロッパにとって大きな機会が訪れました。デベロッパは、iPhone SDKを使用することで、iPhone OSベースのデバイスで使用するためのWebコンテンツを作成できるほか、ユーザがデバイスに保存して使用できるネイティブアプリケーションを作成できます。

この文書を読み、iPhone OS向けに開発できるアプリケーションタイプの範囲や、優れたiPhoneアプリケーションの基礎となるヒューマンインターフェイス設計の原則について学んでください。また、この文書では、iPhoneアプリケーション向けに最上級のユーザインターフェイスやユーザ体験を設計する際に、これらの原則にどのように従うかについても学びます。この文書のガイドラインは、経験豊富なコンピュータアプリケーションやモバイルデバイスアプリケーションのデベロッパばかりでなく、この分野を初めて手がけるデベロッパの方にとっても、ユーザが求めるiPhoneアプリケーションの作成に役立ちます。

注： この文書は、iPhone OSベースのデバイス用のWebベースの開発について簡単に要約しています。これらのデバイス向けのWebコンテンツの設計に特化した詳細については、Safari Reference Libraryの『*iPhone Human Interface Guidelines for Web Applications*』を参照してください。

この書類の構成

『*iPhone*ヒューマンインターフェイスガイドライン』は、2つの部に分かれており、それぞれ複数の章から成ります。

- パートI「iPhoneソフトウェアプロダクトの計画」（13 ページ）では、iPhone OS環境とiPhone OS環境向けに開発できるソフトウェアの種類について説明します。また、基本的なヒューマンインターフェイス設計の原則についても説明し、これらの原則をiPhoneアプリケーションの設計に適用する方法を説明します。
- パートII「iPhoneアプリケーションのユーザインターフェイスの設計」（71 ページ）では、iPhoneアプリケーションのユーザインターフェイスを作成するために使用するコンポーネントについて掘り下げます。利用可能なさまざまなビューおよびコントロールについて説明し、それらを効果的に使用する方法についてのガイダンスを提供します。

関連項目

iPhoneアプリケーションのコーディング方法については、次の文書を参照してください。

- *iOS Application Programming Guide*

iPhone OSベースのデバイス向けWebアプリケーションの設計については、次の文書を参照してください。

- *iPhone Human Interface Guidelines for Web Applications*

iPhoneソフトウェアプロダクトの計画

『*iPhoneヒューマンインターフェイスガイドライン*』のこの部では、iPhone OS向けソフトウェアの設計および開発について検討する方法を説明します。パートIの各章を読んで、iPhone OS向けに開発することができるさまざまな種類のソフトウェアと、開発の成果を特徴付けるのに使用できる設計の原則について学んでください。また、直感的かつ魅力的なユーザインターフェイスを提供する最上級の製品を作成できるよう、アプリケーションの特定の側面や作業にこれらの原則を適用する方法についても学びます。

iPhone OSプラットフォーム：豊かな可能性

iPhone OSは、ユーザがiPhone上のSafariで表示するWebページから、iPhone OSベースのデバイスでネイティブに実行されるiPhoneアプリケーションまで、多数の種類のソフトウェアをサポートします。この章では、iPhone OSベースのデバイス用に作成することができるさまざまな種類のソフトウェアソリューションの概要を示します。

このプラットフォームが初めての場合は、最初のセクション、「念頭に置くべきデバイスの特徴」に示すiPhone OSベースのデバイスとコンピュータの相違点の要約からお読みください。このセクションでの情報は網羅的ではありませんが、iPhoneアプリケーションを設計する際に認識しておく必要がある課題に触れています。

次に、iPhoneアプリケーションの計画立案を支援するため、この章では、各種のアプリケーションスタイルやそれらを規定する特性について考慮する方法を説明します。また、この章では、Mac OS Xのバンドルアプリケーションの一部を、iPhone OSに適したバージョンにどのように変換したかについても説明します。iPhone OS用に作り直したい既存のコンピュータアプリケーションがある場合、このプロセスを理解することが鍵となります。

念頭に置くべきデバイスの特徴

iPhone OSベースのデバイスはデスクトップやラップトップコンピュータではありません。また、iPhoneアプリケーションはデスクトップアプリケーションと同じではありません。あたりまえのことを言っているように聞こえるかもしれませんが、デバイス向けのソフトウェアの開発に着手する際にこれらを念頭に置いておくことは非常に重要です。

iPhone OSベースのデバイス用のソフトウェアを設計するには、特定の考えかたをしなければなりませんが、それは自分が慣れている考えかたとは異なるかもしれません。特に、経験の大半がデスクトップアプリケーションの開発に関するものである場合、モバイルデバイス用のソフトウェアの設計とコンピュータ用のソフトウェアの設計の重要な相違点について認識しておく必要があります。

このセクションでは、設計に関する決定に最も大きな影響を与える可能性のある具体的な相違点について概要を説明します。iPhoneアプリケーションの開発プロセスにおけるこれらの課題およびその他の課題の扱いかたの詳細については、『*iOS Application Programming Guide*』を参照してください。

コンパクトな画面サイズ

iPhone OSベースのデバイスは、小さく、かつ高解像度の画面によって、ユーザのポケットにも収まる強力なディスプレイデバイスとなります。しかし、まさにユーザにとってのこの利点が、デベロッパにとっての困難となる場合があります。デベロッパが設計し慣れたものとは大きく異なるユーザインターフェイスを設計する必要があることを意味するためです。

しかし、むしろ画面サイズがコンパクトであることを根拠として利用して、ユーザインターフェイスは重要なものに焦点を絞るようにしましょう。必要不可欠ではない設計要素を含める余裕はありません。ユーザインターフェイス要素が多すぎると、アプリケーションは魅力的でないものになり使いづらくなります。

メモリの制限

メモリは、iPhone OSでは重要なリソースです。そのため、アプリケーションでメモリを管理することは非常に重要です。iPhone OS仮想メモリモデルにはディスクスワップスペースは含まれないため、デバイス上で利用可能なメモリより多くのメモリを割り当てないように注意する必要があります。メモリ不足の状態が起こった場合、iPhone OSは実行中のアプリケーションに警告を発し、問題が持続するとアプリケーションを終了させる場合があります。アプリケーションがメモリ使用量の警告にすぐに対応し、タイミング良くメモリをクリーンアップするようにしてください。

アプリケーションを設計する際には、メモリリークを解消すること、リソースファイルを可能な限り小さくすること、およびリソースの読み込みを遅らせることなどで、アプリケーションのメモリ使用量を軽減するように努めてください。メモリを適切に扱うiPhoneアプリケーションの設計方法の詳細については、『*iOS Application Programming Guide*』を参照してください。

ユーザが目にするのは一度に1画面

iPhone OS環境とコンピュータ環境の最も大きな相違の1つは、ウインドウパラダイムです。モーダルビューの一部を例外として、ユーザがiPhone OSベースのデバイス上で目にするアプリケーション画面は、一度に1つです。iPhoneアプリケーションには、必要なだけ多くの異なる画面を含めることができますが、ユーザはそれらの画面を順にアクセスして見ることはできても同時に見ることはできません。

対象アプリケーションが、ユーザが複数のウインドウを同時に見る必要のあるデスクトップアプリケーションの場合、そのアプリケーションの作業を1つの画面または一連の複数の画面を通じて達成できる別の方法があるか検討する必要があります。そのようなアプリケーションでない場合は、より広範な機能セットを再現するのではなく、アプリケーションの1つのサブタスクにiPhoneアプリケーションを専念させるべきです。

ユーザが対話するのは一度に1つのアプリケーション

フォアグラウンドに表示できるのは一度に1つのアプリケーションだけです。ユーザがあるアプリケーションから別のアプリケーションに切り替える場合、前のアプリケーションは終了され、そのユーザインターフェイスも表示されなくなります。4.0より前のiPhone OSでは、終了するアプリケーションはすぐにメモリから削除されていました。iPhone OS 4.0以降では、終了するアプリケーションはバックグラウンドに移行され、実行を継続することも、しないこともできます。この機能は、**マルチタスキング**と呼ばれ、再起動されるか終了されるまで、アプリケーションをバックグラウンドに残しておくことができます。

> **注：** マルチタスキングは、iPhone OS 4.0を実行する特定のデバイスで利用できます。

ほとんどのアプリケーションは、バックグラウンドに移行すると一時停止状態になります。ユーザが一時停止されているアプリケーションを再起動する場合、ユーザインターフェイスを再ロードすることなく、終了した時点からの実行の再開を簡単に行うことができます。

アプリケーションによっては、ユーザがフォアグラウンドで別のアプリケーションを実行している間、バックグラウンドでの実行を継続しなければならない場合があります。たとえば、ユーザは、別のアプリケーションでカレンダーを確認したり電子メールを利用したりする間も、オーディオ再生アプリケーションに再生を継続することを求めるでしょう。

マルチタスキングがアプリケーションの動作に与える効果については、「マルチタスキングへの対応」（47 ページ）を参照してください。

オンスクリーンのユーザヘルプは最小限

モバイルユーザは、アプリケーションが使用できるようになるために多くのヘルプコンテンツを通読する時間はありません。また、ヘルプコンテンツを表示したり保存したりするために貴重なスペースを犠牲にすることはできません。iPhone OSベースデバイスの設計の特徴の1つは使いやすさであるため、ユーザの期待に応え、アプリケーションの使いかたをすぐに分かるようなものにすることが重要です。これを達成するためにできることは、次のとおりです。

- 標準コントロールを正しく使用する。ユーザは、組み込みのアプリケーションで目にする標準コントロールに慣れているため、アプリケーションで標準コントロールをどのように使用すべきかをすでに理解しています。
- アプリケーションを通して提示する情報の道筋を、論理的かつユーザが予測しやすいものであるようにする。また、戻るボタンのように、ユーザが、自分がどこにいるかや手順をやり直す方法が分かるような目印を用意しましょう。

どのような選択肢があるか

iPhone OSユーザに対してプロダクトをどのように提示するかを決定する前に、どのような選択肢があるかを理解しておく必要があります。提供しようとしているプロダクトの実装の詳細および対象とする利用者によっては、ニーズに合うソフトウェアの種類が異なる可能性があります。

このセクションでは、主に実装方法に基づいて、iPhone OSベースのデバイス用のソフトウェアを3つの大きなカテゴリに分けています。大まかに言うと、作成できるのは次のとおりです。

- **iPhoneアプリケーション**。iPhone SDKを使用して開発する、iPhone OSベースのデバイス上でネイティブに実行されるアプリケーションです。
- **Web専用コンテンツ**。組み込みのiPhoneアプリケーションのように動作するWebサイトであるWebアプリケーションを含みます。
- **ハイブリッドアプリケーション**。主にWebコンテンツ表示領域を通してWebコンテンツにアクセスしますが、iPhone OSのユーザインターフェイス要素の一部を含んでいるiPhoneアプリケーションです。

iPhoneアプリケーション

iPhoneアプリケーションは、デバイス自体に常駐し、かつiPhone OS環境の機能を利用する点で、iPhone OSベースのデバイス上の組み込みのアプリケーションに似ています。ユーザはiPhoneアプリケーションをデバイスにインストールし、「株価(Stocks)」、「マップ(Maps)」、「計算機(Calculator)」、「メール(Mail)」などの組み込みのアプリケーションを使用するのと同じように使用します。

iPhoneアプリケーションは、起動が速く、使いかたも簡単です。電子メール送信などの作業を可能にするのか、ユーザに娯楽を提供するのかどうかに関係なく、アプリケーションは、その応答性、単純性、および美しく効率的なユーザインターフェイスによって特徴付けられます。

Web専用コンテンツ

Web専用コンテンツをiPhone OSユーザに提供することに関しては、次のとおりいくつかの選択肢があります。

- **Webアプリケーション**

  あるタスクに焦点を当てたソリューションを提供し、かつ特定の表示ガイドラインに準拠しているWebページは、組み込みのiPhone OSアプリケーションと同様の動作をするため、Webアプリケーションとして認識されます。Webアプリケーションは、すべてのWeb専用コンテンツと同様にiPhone上のSafariで実行されます。つまり、ユーザは、デバイスにアプリケーションをインストールするのではなく、WebアプリケーションのURLにアクセスします。

- **最適化されたWebページ**

  iPhone上のSafari用に最適化されたWebページは、設計に従って表示され、動作します（プラグイン、Flash、Javaなどのサポートされていないテクノロジーに基づいて動作する要素を除きます）。また、最適化されたWebページは、デバイスの画面に合わせてコンテンツの大きさを正しく調整します。また多くの場合、最適化されたWebページはiPhone OSベースのデバイス上で表示されていることを検知するよう設計されているため、提供するコンテンツを適宜調整することができます。

- **互換性のあるWebページ**

  iPhone上のSafariと互換性のあるWebページは、設計に従って表示され、動作します（プラグイン、Flash、Javaなどのサポートされていないテクノロジーに基づいて動作する要素を除きます）。互換性のあるWebページの場合、iPhone OSベースのデバイス上での閲覧体験を最適化するための追加ステップを実施しない傾向がありますが、通常、デバイスは正常にページを表示します。

既存のWebサイトやWebアプリケーションの場合は、まず、iPhone OSベースのデバイス上で問題なく動作することを確認してください。また、ユーザがWebクリップ(Web Clip)機能を使用してホーム(Home)画面上に置くことのできるカスタムアイコンを作成することも検討すべきです。実際に、これを行うことにより、ユーザはネイティブアプリケーションのアイコンのように見えるWebサイトへのブックマークをホーム(Home)画面に置けるようになります。カスタムアイコンの作成、およびiPhone OSベースのデバイス上での見栄えが良いWebコンテンツの作成方法の詳細については、『*iPhone Human Interface Guidelines for Web Applications*』を参照してください。

ハイブリッドアプリケーション

iPhone OSを対象に、ネイティブアプリケーションおよびWebページの機能を組み合わせたアプリケーションを作成できます。**ハイブリッドアプリケーション**は、Web表示領域を利用して構造と機能の大部分を提供するネイティブのiPhoneアプリケーションであるだけでなく、標準のiPhone OSのユーザインターフェイス要素を含む傾向があります。

ハイブリッドアプリケーションでは、Web Viewと呼ばれる要素（「Web View」（117 ページ）を参照）を使用してWebコンテンツへのアクセスをユーザに提供します。アプリケーションでWeb Viewを具体的にどのように使用するかは自由ですが、そのアプリケーションが単なる小さなWebブラウザであるという印象をユーザに与えないようにすることが重要です。ハイブリッドアプリケーションは、ネイティブのiPhoneアプリケーションのような動作と概観を持つ必要があります。つまり、Webソースに依存して動作するという事実に気付かせないようにすべきです。

3つのアプリケーションスタイル

この文書では、視覚的特性と動作上の特性、データモデル、およびユーザ体験に基づいて、3つのアプリケーションスタイルを示します。重要なことなので先を読み進める前に強調しておきますが、これらのスタイルの種類を示して説明するのは、設計上のいくつかの決断を明確に行えるようにするためであり、すべてのiPhoneソフトウェアが従う必要のある厳格な区分方法が存在することを示すものではありません。代わりに、これらのスタイルについて説明するのは、さまざまな種類の情報や機能に適したさまざまな方法があることを理解するのに役立つためです。

注： アプリケーションスタイルは、実装方法を決定するものではないことに注意してください。この文書では、ネイティブのiPhoneアプリケーションの設計に焦点を当てますが、ここで詳しく説明するアプリケーションスタイルは、iPhoneベースのデバイス用のWebアプリケーションまたはハイブリッドアプリケーションで実装することができます。

これらの3つのアプリケーションスタイルについて読み進めるなかで、それぞれのアプリケーションスタイルの特性が、用意しようとしている機能のセットおよびiPhoneアプリケーションで提供しようとする全般的なユーザ体験をどのように強化する可能性があるかを検討してください。アプリケーションに最も適した特性の組み合わせを見つけ出すため、iPhoneアプリケーション向けのさまざまな設計スタイルを学ぶ際に次の問いかけを念頭に置いておいてください。

- ユーザがアプリケーションを使用する動機と考えられるのは何か？
- アプリケーションを使用するユーザにどのような体験をさせるか？
- アプリケーションの目的または焦点は何か？
- ユーザが関心を持つ情報をアプリケーションでどのように整理して表示するか？アプリケーションの主たるタスクに対応する自然な整理方法があるか？

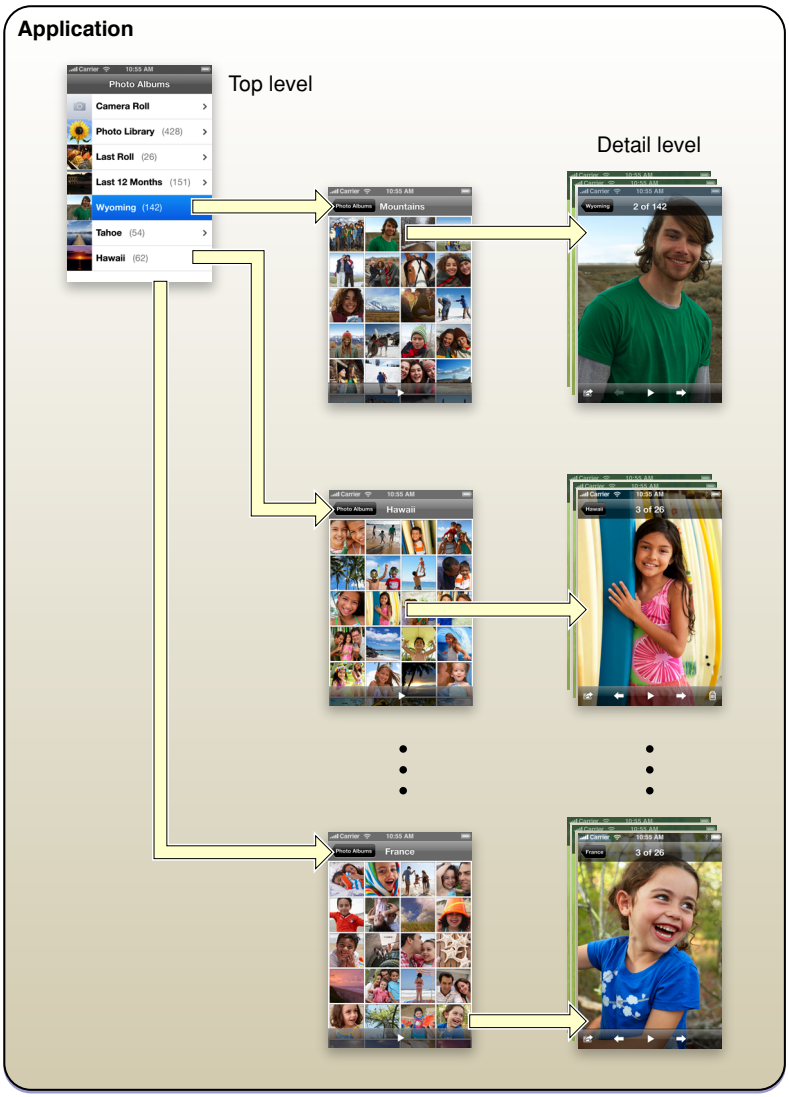
生産性型アプリケーション

生産性型アプリケーションは、詳細情報の整理および処理に基づいた作業を可能にします。ユーザは、重要なタスクを達成するために生産性型アプリケーションを使用します。「メール(Mail)」は、生産性型アプリケーションの好例です。

目的が真剣なものであることは、生産性型アプリケーションが無味乾燥で面白味のないユーザ体験を提供することで真剣なものであるように見せる必要があることを意味するものではありませんが、ユーザを妨げない効率的な方法が好まれることを意味しています。そのため、優れた生産性型アプリケーションはユーザ体験の焦点をタスクに保つことで、ユーザは必要とするものをすばやく見つけ、必要なアクションを簡単に実行し、その作業を完了させ、次の作業に進むことができます。

多くの場合、生産性型アプリケーションはユーザデータを階層的に整理します。この方法では、ユーザは、必要な詳細レベルに達するまで徐々により具体的な選択を行うことで情報を見つけることができます。iPhone OSは、iPhone OSデバイス上でこのプロセスをきわめて効率的に行えるようにするテーブル要素を備えています（これらのユーザインターフェイス要素の詳細については、「Table View」（101 ページ）を参照してください）。図 1-1に、このタイプのデータ整理の一例を示します。

図 1-1　生産性型アプリケーションは情報を階層的に整理する傾向がある



通常、生産性型アプリケーションでのユーザ対話モデルは、次から構成されています。

- リストの整理
- リストへの追加およびリストからの削除
- 必要なレベルに到達するまで継続して詳細レベルをドリルダウンし、次にそのレベルでの情報を使用してタスクを実行する

生産性型アプリケーションは、複数のビューを使用する傾向があります。通常、1つのビューで1つの階層レベルを表示します。ユーザインターフェイスは、単純かつ整理されていて、標準的なビューおよびコントロールから構成されている傾向があります。生産性型アプリケーションは、環境や体験ではなく情報およびタスクに焦点が当てられているため、インターフェイスをカスタマイズすることはあまりありません。

すべてのタイプのiPhoneアプリケーションの中で、「設定(Settings)」アプリケーションでユーザが設定できる環境設定や設定を提供する場合が最も多いのが生産性型アプリケーションです。これは、生産性型アプリケーションが多くの情報を対象に処理を行うことに加え、それらの情報に対するアクセスや管理の方法が潜在的に多いためです。ただし、ユーザがこれらの設定を頻繁に変更する必要があるべきでないことを強調しておくことは重要です。つまり、主たるユーザインターフェイスで処理できる単純な設定変更を対象とすべきではありません。

ユーティリティ型アプリケーション

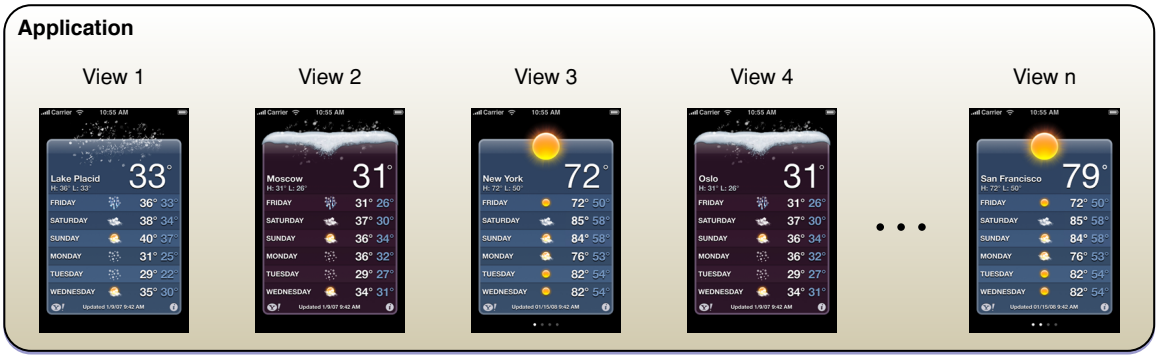
ユーティリティ型アプリケーションは、最低限のユーザ入力を必要とする単純なタスクを実行します。ユーザは、ユーティリティ型アプリケーションを開いて、情報の要約を参照したり、限られた数のオブジェクトに対して簡単なタスクを実行します。「天気(Weather)」アプリケーションは、簡単に目を通せる要約情報として焦点を絞った情報を表示するため、ユーティリティ型アプリケーションの好例です（図 1-2を参照してください）。

図 1-2　「天気(Weather)」はユーティリティ型アプリケーションの一例

ユーティリティ型アプリケーションは見た目が魅力的ですが、それは、表示する情報を不明瞭にすることなく強調するものです。ユーザがユーティリティ型アプリケーションを使用するのは、何かの状態をチェックしたり、何かを調べたりすることが目的であるため、関心がある情報がどこにあるかをすばやく簡単に特定できることを望んでいます。これを可能にするため、ユーティリティ型アプリケーションのユーザインターフェイスは整理されていて、単純かつ多くの場合標準的であるビューおよびコントロールを備えます。

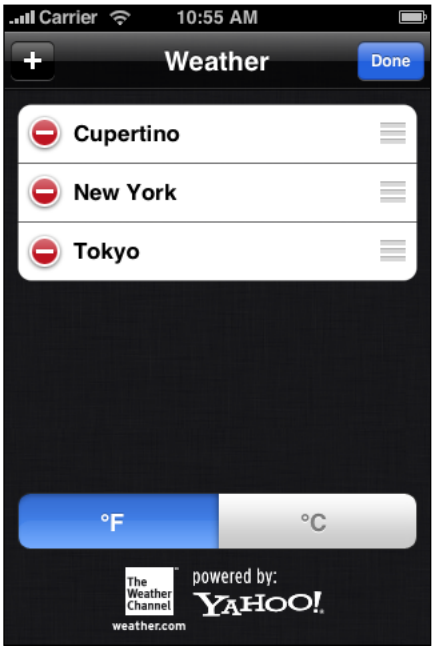
ユーティリティ型アプリケーションは、情報をフラットな項目リストに整理する傾向があります。つまり、ユーザは通常、情報の階層をドリルダウンする必要がありません。通常、ユーティリティ型アプリケーションのそれぞれのビューには、同じまとまりのデータおよび同じ詳細レベルのデータが表示されますが、それぞれの情報源は異なることも可能です。このようにすることで、ユーザは1つのユーティリティ型アプリケーションを開き、複数の対象を同じように取り扱って表示することができます。ユーティリティ型アプリケーションによっては、開いているビューの数を表示することもあります。つまり、ユーザはビューを次々に選択することでビューを順番に移動できます。図 1-3に、このタイプのデータ整理の一例を示します。

図 1-3　ユーティリティ型アプリケーションはフラットなリストでデータを示す傾向がある



ユーティリティ型アプリケーションのユーザ対話モデルは、非常に単純です。ユーザはアプリケーションを開いて、情報の概要に目を通し、必要に応じてその情報の設定やソースを変更します。ユーティリティ型アプリケーションでは、設定や情報ソースの頻繁な変更をサポートする必要があるかもしれないため、多くの場合、そのようなオプションの小さなセットをメインビューの背面に用意しています。ユーザは、メインビューの右下にある、なじみのある「Info」ボタンをタップして、背面を表示します。調整を行ったら、ユーザは「完了(Done)」ボタンをタップして、メインビューの前面に戻ります。ユーティリティ型アプリケーションでは、メインビューの背面にあるオプションはアプリケーションの機能の一部であり、ユーザが1回アクセスしたら再度アクセスすることはほとんどない環境設定のような設定のグループではありません。そのため、ユーティリティ型アプリケーションは、「設定(Settings)」アプリケーションでアプリケーション固有の設定を提供するべきではありません。図 1-4に、「天気(Weather)」アプリケーションのメインビューの背面に用意されている設定オプションを示します。

図 1-4 ユーザは「天気(Weather)」の背面を使って設定が行える

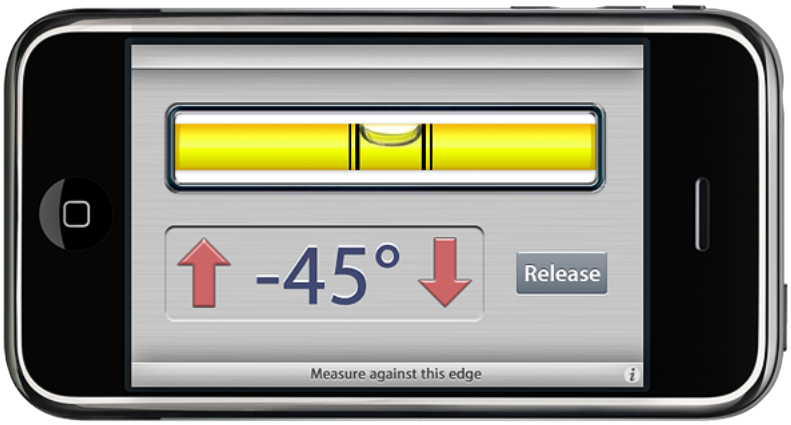


没入型(Immersive)アプリケーション

没入型アプリケーションは、フルスクリーンのビジュアル性に富んだ環境を提供します。ここでは、コンテンツおよびそのコンテンツを扱う際のユーザ体験に焦点が当てられます。ゲームで遊んだり、メディアを駆使したコンテンツを閲覧したり、単純なタスクを実行したりするなど何をするかにかかわらず、ユーザは多くの場合、楽しむために没入型アプリケーションを使用します。

ゲームがこのスタイルのiPhoneアプリケーションに合うことを理解するのは容易ですが、没入型アプリケーションの特性がその他の種類のタスクをどのように強化できるかを想像することも可能です。没入型の手法を採用するのに適しているのは、独特の環境を提供し、テキストベースの情報を大量に表示することはせず、ユーザが注意を向けてくれることに対する見返りを提供するようなタスクです。たとえば、ゲームの定義には適合しませんが、水準器の使用体験を再現するようなアプリケーションは、グラフィックスが豊かなフルスクリーンの環境にはよく合います。ゲームと同様に、このようなアプリケーションにおいてユーザが注目するのは、視覚的なコンテンツとその体験であり、体験の背後にあるデータではありません。図 1-5に、実際の体験を再現し、単純なタスクを可能にする没入型アプリケーションの一例を示します。

図 1-5 没入型アプリケーションはゲームである必要はない



注： 横向きで起動するアプリケーションはホーム(Home)ボタンが右側になるように起動するべきですが、上記の図 1-5のBubble Levelアプリケーションでは逆の向きで起動しています。このため、デバイスの端にある物理的なボタンは測定の妨げになりません。起動のガイドラインの詳細については、「開始」（45 ページ）を参照してください。

没入型アプリケーションは、アプリケーションの世界に入り込むという感覚を強めるカスタムユーザインターフェイスと置き換えることで、デバイスのユーザインターフェイスの大部分を隠す傾向があります。ユーザは、探求と発見を没入型アプリケーションの体験の一部と想定しているため、標準以外のコントロールを使用することが適切である場合が多くあります。

没入型アプリケーションは大量のデータを対象とする場合がありますが、ユーザがそれらを順番に閲覧したりドリルダウンしたりできるように整理して公開することは通常ありません。その代わりに、没入型アプリケーションでは、ゲームのプレイ、ストーリー、体験の文脈に沿って情報を提供します。また同じ理由から、没入型アプリケーションでは多くの場合、ユーティリティ型アプリケーションや生産性型アプリケーションで使用される標準的なデータ駆動型の方法ではなく、環境を補完するカスタムナビゲーションによる方法を提供します。

没入型アプリケーションのユーザ対話モデルは、アプリケーションが提供する体験によって決まります。ゲームでは、アプリケーション固有の設定を「設定(Settings)」で提供する必要がある可能性は低いですが、その他のタイプの没入型アプリケーションでは必要となる場合があります。また、没入型アプリケーションでは、メインビューの背面に設定オプションを用意する場合もあります。

アプリケーションスタイルの選択

生産性型アプリケーション、ユーティリティ型アプリケーション、および没入型アプリケーションについて読み終えたら、自分のアプリケーションが表示する情報およびアプリケーションが可能にするタスクの種類を検討してみましょう。理論的には、作成する必要があるアプリケーションのタイプは自分には明白であり、作業を開始する準備ができているはずですが、実際はそれほど単純ではありません。以下は、決定を行う際に検討するための仮説シナリオです。

検討したい対象がある場合、それに関連するオブジェクトおよびタスクについて考えてみましょう。その対象についてユーザが持つそれぞれに異なる認識を想像してください。たとえば、野球という対象があるとします。野球から思い浮かぶのは、チーム、試合、統計、歴史、選手などさまざ

です。野球は、単一のアプリケーションとしてはおそらく範囲が広すぎるので、選手に絞って考えてみましょう。今度は、選手に関連するアプリケーションをどのように作成するかを考えます。たとえば、野球カードの似顔絵を使用することが考えられます。

熱心なコレクターが野球カードのコレクションを管理するのを支援する生産性型アプリケーションを開発することもできるでしょう。リストベースの形式を使用して、チーム、選手、シーズンという順番の階層でカードを表示できます。最も詳細なビューで、そのカードをどこで入手したか、いくら払ったか、現在の市場価値、およびそのカードを何枚持っているかなどをユーザが記録できるようにすることも可能でしょう。このアプリケーションの焦点はコレクションを規定するデータであるため、ユーザインターフェイスによって、情報を検索したり追加したりするタスクを効率化します。

また、特定の野球カードの現在の市場価値を表示するユーティリティ型アプリケーションを開発することもできるでしょう。それぞれのビューを野球カードのような概観とし、その画像に現在の市場価値を追加できます。そしてビューの背面では、表示および記録を追跡する対象となる特定のカードをユーザが選択できるようにすることも可能です。このアプリケーションの焦点は個々のカードであるため、ユーザインターフェイスはカードの見た目を強調し、ユーザが新しいカードを探せるような1つまたは2つの単純なコントロールを提供します。

あるいは、ゲームを開発することももちろんできるでしょう。たとえば、ユーザが個々の野球カードに記載されている特定の統計情報を知っているかどうかや、有名なカードを見分けられるかどうかなどに焦点を当てたゲームが考えられます。あるいは、スライディングパズルなどの別のタイプのゲームで野球カードをアイコンとして単純に使用することもおそらくできるでしょう。いずれの場合でも、アプリケーションの焦点は、野球カードに掲載されている画像およびゲームの遊びかたに当てられます。ユーザインターフェイスは、野球をテーマとしたいくつかのコントロールを表示し、iPhone OSユーザインターフェイスを隠すことで、これを補完します。

重要なので繰り返しますが、アプリケーションスタイルが1つに制限されているわけではありません。アプリケーションのアイデアが、異なるアプリケーションスタイルの特性の組み合わせによって最も適切に具体化できる場合もあります。

判断がつかないときは、単純化しましょう。機能リストを最小限にし、1つの単純なことを実行するアプリケーションを作成します（アプリケーションの焦点を絞る方法の詳細については、「プロダクト定義ステートメントの作成」（35 ページ）を参照してください）。ユーザがどのようにアプリケーションを使用し、それに反応するかを見て、少し焦点を変えたり表現方法を変更したりした別のバージョンのアプリケーションを作成することが考えられます。あるいは、同じコンセプトで、より細部に気を配った（または細部を省略した）バージョンが必要であることが分かるかもしれません。

既存のコンピュータアプリケーションがある場合

既存のコンピュータアプリケーションが存在する場合、それをそのままiPhone OSには移植しないでください。iPhone OSベースのデバイスの使いかたは、デスクトップやラップトップコンピュータとは大きく異なっており、ユーザが期待するユーザ体験も異なります。

ユーザは、iPhone OSベースのデバイスを外出中、つまり気が散るような環境で使用することを忘れないでください。これは一般的に、ユーザはアプリケーションを開き、短時間だけ使用したら、別の作業に移りたいということを意味します。長時間にわたるユーザの注目を必要とするようなアプリケーションの場合、iPhone OSへ移植するにはアプリケーションの構造と目的を再考する必要があります。

対象デスクトップアプリケーションが複雑なタスクやタスクのセットを可能にするものであるならば、そのアプリケーションがどのように使われているかを調べて、移動中に達成できれば好ましいとユーザが考える可能性のあるサブタスクを2〜3つ見つけましょう。たとえば、プロジェクトのスケジューリング、請求書発行、経費報告書の作成をサポートするビジネス向けのアプリケーションからは、プロジェクトの進捗の概要を示すiPhoneユーティリティ型アプリケーションや、移動中のユーザが仕事に関係する経費を把握できるiPhone生産性型アプリケーションなどを導き出せます。

デスクトップアプリケーションのアイデアをiPhoneアプリケーションに移植する方法を考える際には、「80対20」ルールをアプリケーションの設計に当てはめましょう。ユーザの大部分（少なくとも80%）は、アプリケーションのごく限られた数の機能しか使用しないのに対し、アプリケーションのすべての機能を使用するのはごく少数のユーザ（20%以下）であると推定してください。次に、少数のユーザのみが必要とする強力な機能をiPhoneアプリケーションに搭載するかどうかを慎重に検討してください。そのような機能を提供する環境にはデスクトップコンピュータのアプリケーションの方がより適切であること、および、通常、大多数のユーザの必要性を満たす機能にiPhoneアプリケーションの焦点を当てる方が良いことに注意してください。

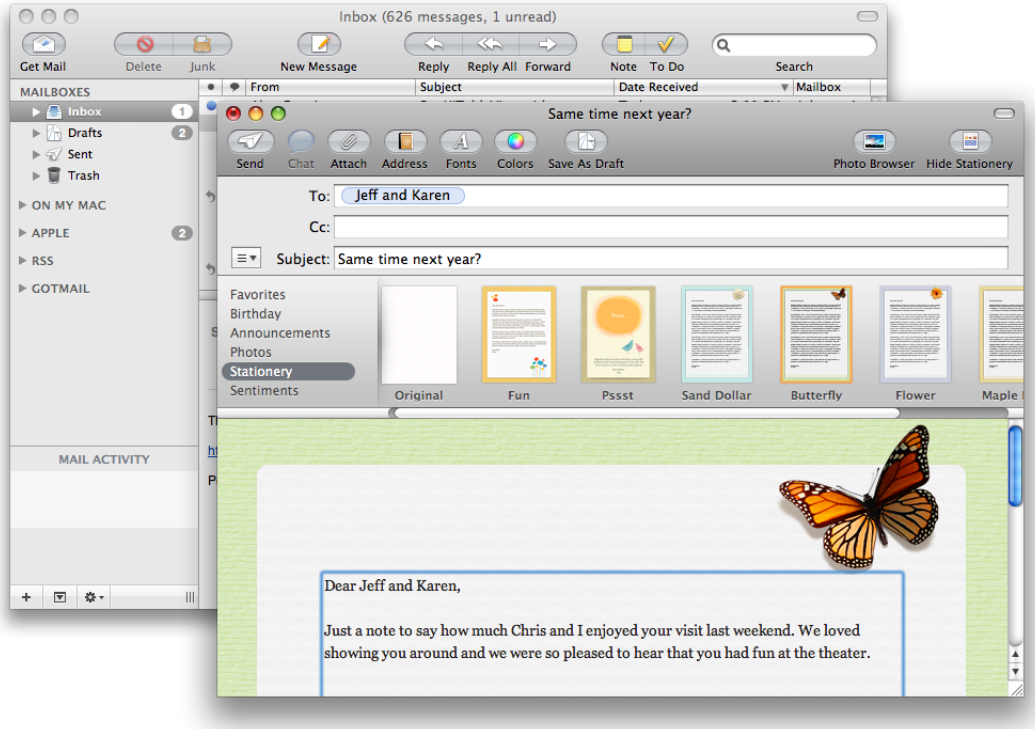
ケーススタディ：デスクトップアプリケーションをiPhone OSに移植する

デスクトップコンピュータ向けアプリケーションのiPhone OSバージョンを作成する方法をイメージしやすいように、このセクションでは、使い慣れているMac OS Xアプリケーションとそれに対応するiPhone OSアプリケーションの設計上の相違点をいくつか説明します。それぞれのアプリケーションのどの機能がiPhone OSバージョンに適応されているかを学べば、自分のiPhoneアプリケーションに関して判断する必要がある設計上の決定の種類についての見識が得られます。

Mail

Mailは、Mac OS Xのアプリケーションのなかでも、最も目立ち、よく使われる、評価の高いアプリケーションの1つです。また、電子メールの作成、受信、優先順位付け、保存を行い、アクション項目やイベントを追跡し、メモや招待状の作成を行う、非常に強力なプログラムでもあります。Mailは、この機能の大部分を1つのマルチペインウインドウで提供します。これは、ディスプレイ画面上のMailのウインドウをいつでも離れて（または、Dockに最小化して）、必要に応じて戻れるため、デスクトップコンピュータを使用するユーザにとっては便利です。図1-6に、デスクトップ上のMailのメッセージ表示ウインドウおよび作成ウインドウで使用できるさまざまな機能を示します。

図 1-6 デスクトップのMailはさまざまな強力機能を複数のウインドウで提供する

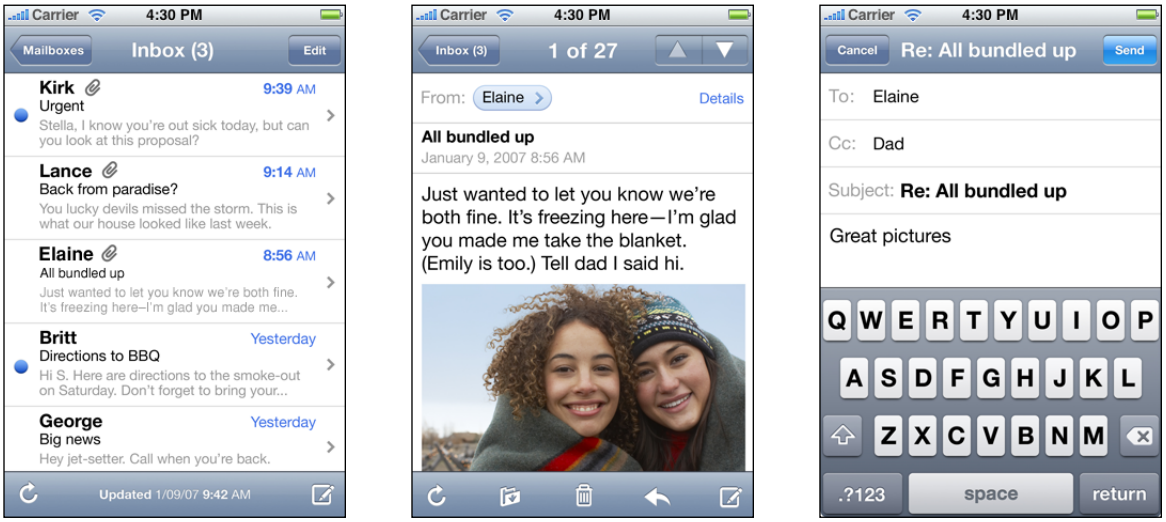


しかし、ユーザが移動している場合、電子メールアプリケーションに求められるニーズはより単純で、ユーザは中核的な機能にすばやくアクセスしたいと考えています。このため、iPhone OSベースのデバイス上の「メール(Mail)」は、ユーザが電子メールで行う最も重要な作業、つまり受信、作成、送信、およびメッセージの整理に焦点を当てています。これを行うため、「メール(Mail)」は、ユーザのアカウントおよびメールボックスの整理がしやすく、ユーザの注意をメッセージに集中させる軽量化したユーザインターフェイスを表示します。

iPhone OS上の「メール(Mail)」は、生産性型スタイルアプリケーションの好例です。コンテンツ間の移動を簡単にするため、iPhone OS上の「メール(Mail)」は、ユーザの電子メールを自然な階層構造に整理し、アカウント、メールボックス、メッセージリスト、およびそれぞれのメッセージを順番にページに表示します。ユーザは、リスト内の項目を選択し、その項目に関連するものを表示させるという方法で、一般的なもの（アカウントのリスト）から特定のもの（メッセージ）にドリルダウンします。iPhoneアプリケーションの生産性型スタイルの詳細については、「生産性型アプリケーション」（19 ページ）を参照してください。

また、iPhone OS上の「メール(Mail)」は、タップしやすく、なじみのある少数のコントロールを表示することで、作成や送信などのアクションを可能にしています。図1-7に、「メール(Mail)」によってiPhone OS上での電子メールの表示や送信がどのように簡単になるのかを示します。それぞれの画面上部にある要素によって、ユーザがアプリケーションでの現在の位置と以前の位置の両方を簡単に把握できることに注目してください。

図 1-7 iPhone OSの「メール(Mail)」では電子メールの表示や送信が簡単



iPhoto

iPhone OS用に作り直されたMac OS Xアプリケーションのもう1つの役に立つ例がiPhotoです。デスクトップ上で、iPhotoは、包括的な検索と整理、強力な編集機能、およびクリエイティブな印刷オプションをサポートしています。ユーザがiPhotoをデスクトップまたはラップトップコンピュータで使用する場合、コレクション全体の閲覧や整理、写真の調整、さまざまな方法での写真の操作が可能なことが評価されます。iPhotoの主な焦点はユーザのコンテンツにありますが、このアプリケーションはウインドウ内で幅広い機能も提供します。図1-8に、デスクトップ上のiPhotoのユーザインターフェイスを示します。

図 1-8 iPhotoユーザインターフェイス

しかし、移動中のユーザは、写真を編集する時間はありません（また、写真の印刷も必要ではありません）。代わりに、すばやく写真を見たり共有したりできることが必要となります。

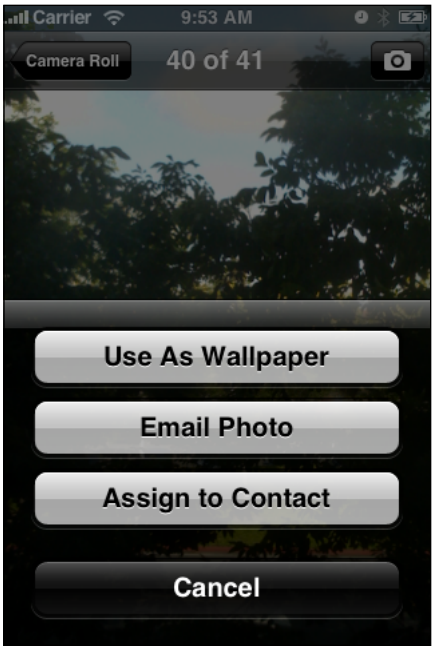
iPhone OSベースのデバイス上でこの必要性を満たすため、Appleは「写真(Photos)」アプリケーションを提供しました。このアプリケーションでは、写真を見たり、写真をほかの人と共有したりすることに焦点が当てられています。「写真(Photos)」のユーザインターフェイスは写真を中心としたものであるため、実のところデバイスのユーザインターフェイスの一部も隠せるようになっています。ユーザが写真をスライドショーで見る場合、「写真(Photos)」アプリケーションは、ナビゲーションバー、ツールバー、そしてステータスバーさえも隠します。また、それらの要素を見る必要があるときには、それらが半透明になったものを表示します。

「写真(Photos)」では、階層的に整理することで、ユーザが写真を整理したり探しやすくしたりしています。ユーザは、写真のコレクションが含まれているアルバムを選択し、そのコレクションから1枚の写真を選択します。このように、「写真(Photos)」は、生産性型スタイルおよび没入型スタイルの機能を統合したアプリケーションの一例です（これらのスタイルの詳細については、「3つのアプリケーションスタイル」（19 ページ）を参照してください）。図 1-9に、ユーザがどのように「写真(Photos)」アプリケーションで写真を見ることができるかを示します。

図 1-9 「写真(Photos)」アプリケーションの3つの画面

また、「写真(Photos)」は、Action Sheetと呼ばれる一時的なビューを使用して、写真の閲覧中にユーザに対して追加の機能を提示します（「Alert、Action Sheet、およびモーダルビュー」（89 ページ）を参照してください）。図 1-10に、「写真(Photos)」がそれぞれの写真を使用するための選択肢をどのように提示するかを示します。

図 1-10 「写真(Photos)」はAction Sheetでユーザに選択肢を提示する



ヒューマンインターフェイスの原則：優れたユーザインターフェイスの作成

優れたユーザインターフェイスは、デバイスの機能ではなく、ユーザがどのように考え、どのように作業するかに基づいた、ヒューマンインターフェイス設計の原則に従っています。ユーザインターフェイスが魅力的でなかったり、複雑であったり、論理的でなかったりすると、たとえ優れたアプリケーションでも使うのが面倒に思えてしまいますが、ユーザインターフェイスが美しく、直感的かつ魅力的であると、アプリケーションの機能が強化され、ユーザの愛着が促進されます。

メタファ

可能であれば、アプリケーションのオブジェクトやアクションを実世界でのオブジェクトやアクションになぞらえてください。この手法は、アプリケーションがどのように動作するのかを初心者のユーザがすばやく把握するのに特に役立ちます。フォルダは、古典的なソフトウェアのメタファです。人は実世界で物事をフォルダに整理します。そのため、コンピュータ上でデータをフォルダに入れるという考えかたをすぐに理解できます。

iPhone OSのメタファに含まれるものとしては、iPodの再生コントロール、何らかの動作をさせるためのタップ可能なコントロール、スライド式のオン／オフスイッチ、ピッカーホイールに表示されるデータのフリックがあります。

メタファは、iPhone OSインターフェイスでのオブジェクトやアクションの使いかたを示唆しますが、このことによってソフトウェアによるメタファの実装が制約を受けるわけではありません。フォルダの例に戻って、ソフトウェアに実装されているフォルダオブジェクトの容量は、実世界のフォルダの物理的な容量とはまったく関係がありません。

アプリケーションを設計する際には、iPhone OSに存在するメタファに注意し、それらを再定義しないようにしてください。同時に、アプリケーションが実行するタスクを調べて、使用できる自然なメタファが存在するかどうかを確認しましょう。ただし、アプリケーションのユーザインターフェイスに合わせるためだけに実世界のオブジェクトやアクションを拡張するよりも、標準のコントロールやアクションを使用する方が良いことに注意してください。選んだメタファが大半のユーザに認識される可能性が高いのでなければ、それらを含めることで混乱が減るどころか増えます。

直接操作

直接操作とは、ユーザが抽象的なものではなく具体的なものを制御しているように感じることです。直接操作の原則に従うことの利点は、関係するオブジェクトを直接操作できると、ユーザは自分が行うアクションの結果を容易に想像できることです。

iPhone OSユーザは、Multi-Touchインターフェイスのおかげで、直接操作の感覚をより高度に感じることができます。ユーザはオブジェクトを操作するための中間デバイス（マウスなど）は使用せずに、ジェスチャを使用することで、画面に表示されているオブジェクトに対するより強い親しみの感覚や、それらのオブジェクトを制御している感覚を持つことができます。

iPhoneアプリケーション上で直接操作の感覚を高めるために、次の点に注意してください。

- ユーザがオブジェクトに対してアクションを実行している間、オブジェクトを画面に表示し続ける
- ユーザのアクションの結果は、すぐに表示反映される

参照とポイント

オプション、コマンド、データなどのリストを記憶しておくことについては、iPhoneアプリケーションの方が人間よりも優れています。これを利用して、選択肢やオプションをリスト形式で表示することで、ユーザがそれらに簡単に目を通して選べるようにできます。テキスト入力を最小限に保つことで、ユーザが入力に多くの時間を割く必要性と、アプリケーションで入力のエラーチェックを行う必要性をなくします。

また、より制限の少ない入力をユーザに求めるのではなく選択肢を提示することで、ユーザはアプリケーションの操作方法を憶えることではなく、アプリケーションでタスクを達成することに集中できます。

フィードバック

ユーザは、アクションの結果の確認に加え、コントロールを操作したときの即座のフィードバックや、時間のかかる処理のステータス報告を必要とします。アプリケーションは、ユーザによるすべてのアクションに対して、目に見える何らかの変化で応える必要があります。たとえば、ユーザがリスト項目をタップしたら、そのリスト項目を短時間だけハイライトさせます。音によるフィードバックも役に立ちますが、ユーザはiPhone OSベースのデバイスを音が聞こえない場所やサウンドをオフにする必要がある場所で使用する可能性があるため、音を主たるフィードバックの方法または唯一のフィードバックの方法にはできません。また、ユーザがシステムアラートに関連付けたiPhone OSシステムのサウンドと競合させるべきではありません。

iPhone OSは、一時的にビジー状態となっている場合にはアクティビティインジケータを表示することで自動的にフィードバックを提供します。2～3秒以上継続して処理が行われる場合、アプリケーションはその間、経過進捗を表示し、妥当と考えられる場合には説明メッセージを表示すべきです。

アニメーションは、ユーザにフィードバックを提供する優れた方法です。ただし、それが妨げにならず、意味がある場合に限ります。没入型でないアプリケーションも含めて、アニメーションはiPhone OSの隅々にまで行き渡っています。ただしアニメーションは、フィードバックを提供する手段として、ユーザ体験の中心としてではなく、ユーザ体験を向上するために使用されています。

ユーザによる制御

アプリケーションではなく、ユーザがアクションの開始や制御を行えるようにしましょう。アクションを単純かつ簡単なものに保つことで、ユーザがアクションを容易に理解し記憶できるようにします。可能な限り、ユーザがすでに慣れ親しんでいる標準的なコントロールや動作を使用してください。

開始する前に操作を取り消すための十分な機会を提供し、害を及ぼしうるアクションをユーザが開始しようとしたら確認を取るようにしてください。可能な限り、進行中の操作をユーザが秩序正しく停止できるようにしましょう。

外観の整合性

アプリケーションの最終目的はタスクを可能にすることですが、そのタスクがゲームで遊ぶことであったとしても、アプリケーションの外観の重要性を過小評価してはなりません。これは、外観が機能性に対して強い影響を持つためです。込み入っているように見えたり、論理的でないように見えるアプリケーションは、理解し使用することが難しくなります。

外観の整合性は、アプリケーションがどれだけ美しいかを示す尺度ではありません。これは、アプリケーションの外観がその機能とどれほど整合しているかを示す尺度です。たとえば、生産性型アプリケーションは、標準的なコントロールや動作を提供することでタスクに目が向くようにする一方で、装飾的な要素は妨げにならないように背景に溶け込むようにするする必要があります。

没入型アプリケーションはこの考えかたの対極に位置するもので、ユーザは、楽しみを約束し発見を促すような美しい外観を求めます。没入型アプリケーションでは、娯楽性を提供することに焦点が当てられがちですが、それでもその外観はタスクと整合する必要があります。そのようなアプリケーションのユーザインターフェイス要素の設計は、注意深く行うことで、内部的に一貫性を持った体験を提供できるようにしてください。

iPhoneアプリケーションの設計：プロダクトの定義からブランド設定まで

iPhoneアプリケーションを開発する場合、iPhone OS、およびモバイル環境のさまざまな側面が設計上の決定にどのように影響を与えるかを学習する必要があります。この章では、プロダクトの定義からブランド設定にいたるまでのアプリケーション設計の各課題に関して、一連のガイドラインを紹介します。また、iPhoneアプリケーションでこれらの課題に取り組む方法についても説明します。

プロダクト定義ステートメントの作成

アプリケーションの設計を開始する前に、そのアプリケーションが何をするのかを正確に定義することが重要です。これを行う良い方法は、**プロダクト定義ステートメント**、つまりアプリケーションの主たる目的および想定する利用者について簡潔な宣言を作成することです。プロダクト定義ステートメントの作成は、単なる練習ではありません。むしろ、機能のリストを一貫性のあるプロダクトにまとめあげるための最善の方法の1つです。

はじめに、時間をかけて対象利用者を定義します。対象利用者は経験のあるユーザでしょうか。それとも初心者でしょうか。本格的なユーザでしょうか。それとも一時的なユーザでしょうか。特定のタスクについて支援を必要としているのでしょうか。それとも娯楽を探しているのでしょうか。対象ユーザに関してこのようなことを知ることは、ユーザの特定のニーズや要求に合わせてユーザ体験やユーザインターフェイスをカスタマイズするのに役立ちます。

iPhoneアプリケーションを設計しているということは、対象ユーザについてすでに多くのことを理解していることでしょう。たとえば、ユーザの特徴としては次のようなものがあります。

- ユーザは移動する。
- ユーザは、アプリケーションをすばやく開いて、役に立つコンテンツをすぐに見たいと考えている。
- 数回タップするだけでアプリケーションで物事を達成できる必要がある。

今度は、対象ユーザをほかのすべてのiPhone OSユーザと隔てる特徴は何かを自問してください。対象ユーザは、ビジネスマンでしょうか、ティーンエージャーでしょうか、それとも退職者でしょうか。対象ユーザはアプリケーションを毎日の終わりに使用するでしょうか。それとも、電子メールをチェックするたびでしょうか。あるいは空き時間が少しあるたびに使用するでしょうか。対象ユーザの定義が正確であるほど、ユーザインターフェイスの外観、操作感、および機能に関する決定がより正確になります。

たとえば、ビジネスマンが使用経費を把握するのを支援するアプリケーションの場合、ユーザインターフェイスは、タスクの重要でない詳細な入力をユーザに多く求めずに、適切なカテゴリを提供し、費用を入力しやすくすることに焦点を当てるべきです。また、プロフェッショナルに見え、一日に何度見ても心地よい当たり障りのない配色を選ぶこともできます。

あるいは、アプリケーションがティーンエージャーをターゲットとするゲームの場合、刺激的なユーザインターフェイス、特別扱いを感じさせるような言葉、はやりファッションを想起させる配色が望ましいかもしれません。

最後に、提供しようとしている機能セットを検証してください。対象利用者のイメージを念頭に置き、機能のリストからエッセンスを抜き出して、プロダクトが提供するソリューションおよび対象となるユーザを表す1つのステートメント、つまりプロダクト定義ステートメントを作成してみましょう。たとえば、デスクトップのiPhotoアプリケーションは、ユーザによる写真の整理、編集、共有、印刷、表示を可能にします。ただし、良いプロダクト定義ステートメントは、機能にのみ焦点を当てるのではなく、想定する利用者も示します。したがって、iPhotoの適切なプロダクト定義ステートメントは、「アマチュア写真家のための使いやすい写真管理アプリケーション」のようになります。対象利用者の定義をプロダクト定義ステートメントに含めることの重要性に注目してください。iPhotoが「プロ写真家のための使いやすい写真管理アプリケーション」となるように設計された場合に、どれほど違うアプリケーションになるか想像してください。

良いプロダクト定義ステートメントは、機能、ツール、用語が適切かどうかを判断するために開発プロセス全体を通して使用する必要があるツールとなります。iPhoneアプリケーションは主たるタスクに焦点を当てていない機能に割く余地はないため、プロダクト定義ステートメントをサポートすることのない要素を排除することが特に重要です。

たとえば、食料雑貨の買い物をするときに使用できるiPhoneアプリケーションを開発することを想像してください。計画段階では、ユーザが行う可能性のある次のような幅広い範囲の行動を含めることを考えるでしょう。

特定の食品に関する栄養情報の取得
クーポンや特別提供キャンペーンの検索
ショッピングリストの作成と使用
店舗の場所の検索
レシピの検索
価格の比較
現時点の合計金額の表示

しかし、ユーザは購入する必要のあるすべてのものを覚えておくことに最も関心を持ち、可能であればお金を節約したいと考え、しかも買い物を早く済ませて帰宅したいだろうと考えられます。この対象利用者の定義を使用して、「急いでいる人のためのショッピングリスト作成およびクーポン検索ツール」などのように、アプリケーションのプロダクト定義ステートメントを作成します。このプロダクト定義ステートメントに照らして候補機能のリストを選り分けた結果、主にショッピングリストを簡単に作成、保存、および使用できるようにすることに焦点を当てることにします。また、ショッピングリストにある項目に対するクーポンを検索する機能をユーザに提供します。それら以外の機能も便利であり、ほかのアプリケーションであれば主たる機能になる可能性はありますが、このアプリケーションのプロダクト定義ステートメントには適合しません。

確かなプロダクト定義ステートメントを決定し、候補機能を絞り込むために使用し始めたら、アプリケーションのタイプに関する当初の決定が引き続き正しいことを確認するためにも、このステートメントを使用するとよいでしょう。特定のアプリケーションのタイプを念頭に置いて開発プロセスを開始した場合、プロダクト定義ステートメントを定義するプロセスを通じて全体像が変化していることに気付くことがあります（開発可能なアプリケーションのそれぞれのタイプの詳細については、「3つのアプリケーションスタイル」（19 ページ）を参照してください）。

優れたiPhoneアプリケーションの特徴を組み込む

優れたiPhoneアプリケーションは、ユーザが求める体験を提供しながら、ユーザがまさに必要とすることを行います。アプリケーションのなかでこのバランスを実現できるように、このセクションでは、優れたiPhoneアプリケーションのいくつかの特徴を検討し、それらの特徴を自分のプロダクトに組み込む方法についてアドバイスを提供します。

簡潔さと使いやすさを作り込む

簡潔さと使いやすさは、すべてのタイプのソフトウェアにとって基本的な原則ですが、iPhoneアプリケーションでは非常に重要です。iPhone OSユーザは、アプリケーションを使用するのと同時に、ほかのことも行っている可能性があります。ユーザは、アプリケーションの使いかたがすぐに分からなければ、競合他社のアプリケーションに移ってしまい戻ってこないでしょう。

アプリケーションとそのユーザインターフェイスのフローを設計する際には、次のガイドラインに従って簡潔さと使いやすさを作り込むようにします。

- アプリケーションの使いかたが明白になるようにする。
- 頻繁に使用される上位レベルの情報を画面上部近くに集中させる。
- テキスト入力を最小限にする。
- 重要な情報は、簡潔に表す。
- タップ可能なすべての要素のターゲット領域は、指先サイズにする。

以降の各セクションでは、簡潔さと使いやすさについて、それぞれのガイドラインをさらに詳しく説明します。

使用方法を明白にする

アプリケーションがどのように動作するのかを理解するためにユーザが時間を割く（または注意を払える）ことを想定してはなりません。したがって、ユーザが即座に理解できるようなアプリケーションを目指す必要があります。

アプリケーションの主たる機能は、見た目で即座に理解できるものでなければなりません。これを行うには、ユーザが選ぶ必要のあるコントロールの数を最小限にし、それらのコントロールが何を行うかをユーザが正確に理解できるように明確にラベル付けします。たとえば、図 3-1に示すように、組み込みのストップウォッチ機能（「時計(Clock)」アプリケーションの一部）では、ユーザはストップウォッチの停止や開始、およびラップタイムの記録をどのボタンで行うかが一目で分かります。

図 3-1 組み込みのストップウォッチ機能は使用方法が明確

トップダウンで考える

ユーザは、指（親指を含む）でiPhone OSベースのデバイスの画面をタップできます。指（親指を除く）を使用するとき、ユーザは利き手ではない方の手でデバイスを持ち（または平らな場所にデバイスを置いて）、利き手の指でタップする傾向があります。親指を使用する場合、片手でデバイスを持ち、その手の親指でタップするか、両手でデバイスを持ち、両方の手の親指でタップします。どちらの方法をユーザが使用しても、画面上部がそのユーザに最も見えやすくなります。

これらの使用パターンから、最もよく使用される（通常、より上位の）情報が、最も見えやすくアクセスしやすい画面上部にあるようにアプリケーションのユーザインターフェイスを設計するべきです。ユーザは画面の上から下へ目を動かすので、表示される情報は、概略的な情報から詳細な情報へ、上位の情報から下位の情報へ進むようにすべきです。

必要な入力を最小限にする

情報の入力には、コントロールをタップするかキーボードを使用するかにかかわらず、ユーザの時間と注意が必要となります。何か有用なことを行うためにアプリケーションが多くのユーザ入力を必要とすれば、ユーザはペースが落ち、使い続ける気をそぐ可能性があります。

もちろん、ユーザから何らかの情報を必要とする場合も多くありますが、引き換えにユーザに提供するものとのバランスを取る必要があります。言い換えると、ユーザが提供する情報のそれぞれに対して、可能な限りの情報や機能を提供するように努めてください。そうすれば、ユーザは、アプリケーションで作業を進めるなかで、作業が遅れることなくはかどっていると感じます。

ユーザに入力を求める場合、テキストフィールドではなく、Table View（またはピッカー）型のものを使用することを検討してください。通常、単語を入力するよりもリストから項目を選択する方がユーザにとって簡単です。Table Viewおよびピッカーの詳細については、それぞれ「Table View」（101 ページ）および「ピッカー」（126 ページ）を参照してください。

情報を簡潔に表す

ユーザインターフェイスのテキストが短く直接的なものであれば、ユーザはそれをすばやく簡単に理解できます。そのため、探しているものを見つけたり、次に何をするべきかを知ったりするためにユーザが多くの単語を読まなくてもよいように、最も重要な情報を特定して、それを簡潔に表現し、目立つように表示します。

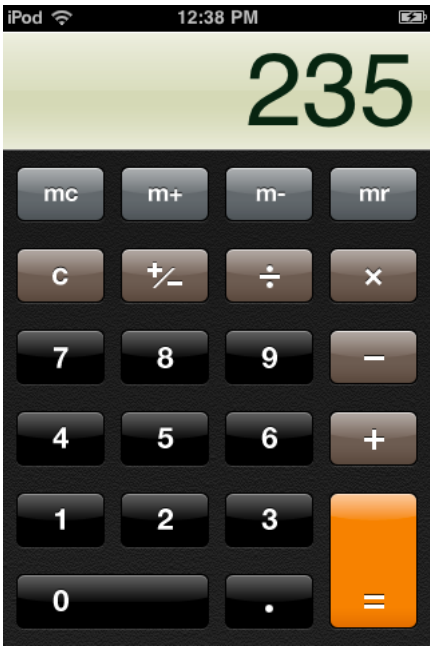
これを行うのに役立つのは、新聞の編集者のように考え、凝縮された見出しのようなスタイルで情報を伝えるように努めることです。コントロールには短いラベル（またはよく知られている記号）を与え、ユーザがその使用方法を一目で分かるようにします。

指先サイズのターゲットを提供する

複数のコントロールが互いに近すぎるレイアウトの場合、ユーザは慎重にタップするために余分な時間と注意を払う必要があり、また、誤った要素をタップする可能性も高くなります。単純かつ使いやすいユーザインターフェイスではコントロールとその他のユーザ対話要素の間に空きがあるため、ユーザは最小限の労力で正確にタップできます。

たとえば、組み込みの「計算機(Calculator)」アプリケーションでは、44×44ピクセルのターゲット領域をそれぞれ持つ、大きくてタップが簡単なコントロールが表示されます。図 3-2に、「計算機(Calculator)」アプリケーションを示します。

図 3-2 組み込みの「計算機(Calculator)」アプリケーションでは指先サイズのコントロールが表示される



主たるタスクに焦点を当てる

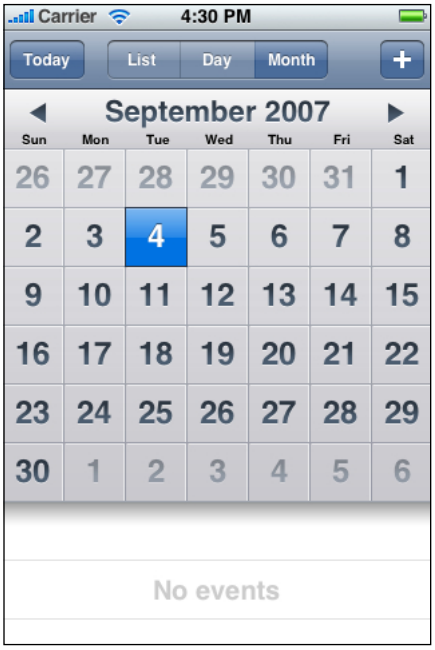
主たる機能に焦点が当たるようにし、それを維持するiPhoneアプリケーションの使用は楽しく、満足感が得られます。したがって、アプリケーションを設計する際には、プロダクト定義ステートメントを常に念頭の置き、すべての機能とユーザインターフェイス要素がこれをサポートするようにしてください。プロダクト定義ステートメントの作成方法に関するアドバイスについては、「プロダクト定義ステートメントの作成」（35 ページ）を参照してください。

焦点をはずさないようにする良い方法の1つは、それぞれの文脈において何が最も重要かを判断することです。それぞれの画面で何を表示するかを決めるとき、「これはユーザがこの場面で必要としている重要な情報または機能なのか」を常に自分に問いかけてください。あるいは、より具体的な言葉でこれを考えると、「この情報や機能は、お店で買い物をしているときや会議から別の会議へ歩いて移動しているときにユーザが必要とするものか」となります。その答えが「いいえ」の場合、その情報や機能が別の文脈において重要かどうか、それとも結局はそれほど重要ではないかどうかを判断します。たとえば、ユーザが車の走行距離を把握するのを支援するアプリケーションが、車の販売業者の場所も把握するのであれば、この機能に当てた焦点は失われることになります。

アプリケーションを単純かつ使いやすいものにするためのガイドラインに従えば、ソリューションは焦点が定まったものとなります。特に、アプリケーションの使いかたを明白なものにし、ユーザ入力を最小限にするとよいでしょう。これにより、ユーザがアプリケーションの最も重要な部分にすばやく到達するのが簡単になり、ソリューションの焦点が絞られます（これらのガイドラインの詳細については、「簡潔さと使いやすさを作り込む」（37 ページ）を参照してください）。

たとえば、組み込みの「カレンダー(Calendar)」アプリケーション（図 3-3を参照）では、日付およびその日付で発生するイベントに焦点が当てられます。ユーザは、明確にラベル付けされたボタンを使用して、現在の日付のハイライト、表示オプションの選択、およびイベントの追加を行うことができます。最も重要な情報、つまり日付および日付に関連するイベントが最も目立ちます。すべての入力に対してキーボードからの入力を求めるのではなく、イベント回数、反復の間隔、および通知オプションのリストからユーザが選べるようにすることで、ユーザ入力を簡単にしています。

図 3-3 組み込みの「カレンダー(Calendar)」アプリケーションでは日付とイベントに焦点が当てられる



効果的なやり取り

やり取りとフィードバックは、iPhoneアプリケーションにおいても、デスクトップコンピュータのアプリケーションの場合と同じくらい重要です。ユーザは、要求が処理されているかどうか、および自分のアクションによってデータが失われたり、その他の問題が生じたりする可能性がある場合に、そのことを知る必要があります。とは言うものの、やり取りが多すぎ（たとえば、それほど深刻でない状況でユーザに警告したり、あまりに頻繁に確認を求めることなど）にならないようにすることも重要です。

アニメーションは、ユーザのタスクの妨げになったり、作業を遅らせたりしない限り、効果的にやり取りするための良い方法です。妨げにならない適切なアニメーションは、ステータスを伝えたり、有用なフィードバックを提供したり、ユーザのアクションの結果を視覚化したりするのに役立ちます。過度の、または余計なアニメーションは、アプリケーションの流れを妨げ、パフォーマンスを低下させ、ユーザを不快にする可能性があります。

ユーザとのすべてのテキストベースのやり取りでは、ユーザを中心に考えた用語を使用してください。特に、ユーザインターフェイスでは技術的な専門用語を使用しないようにしましょう。ユーザについて知っていることを利用して、使用しようとしている言葉や言い回しが適切かどうかを判断してください。たとえば、図3-4に示すように、「Wi-Fiネットワーク(Wi-Fi Networks)」の設定画面では、技術的ではない分かりやすい言葉を使用して、デバイスがどのようにネットワークに接続するかを説明しています。

図 3-4 アプリケーションのユーザインターフェイスにはユーザ中心の用語を使用する

ジェスチャを適切にサポートする

ユーザは、指を使ってiPhone OSベースのデバイス独自のMulti-Touchインターフェイスを操作します。タップ、フリック、ピンチすることによって、Webコンテンツを選択、移動、閲覧し、アプリケーションを使用します。指を使ってデバイスを操作することには、実用的な利点があります。つまり、指はいつでも使うことができ、さまざまな動きが可能な点です。また、マウスなどの外部入力デバイスなどでは得られない、デバイスに対するじかの接触感とつながりを感じることができます。

ただし、指には大きな短所が1つあります。つまり、指は、そのサイズや形状、指の持ち主の器用さにかかわらず、マウスポインタよりもかなり大きいということです。ディスプレイ画面に対して、指はマウスポイントほど正確にはなり得ません。

幸い、優れたユーザインターフェイスを設計することで、指ベースの入力システムの課題に対応することができます。多くの場合、これは、指先の平均的サイズに合わせたレイアウトにすることを意味します。また、指の動きに対して、ユーザが期待するアクションで応えることも意味します。

ユーザは、**ジェスチャ**と呼ばれる特定の動作を行うことで、特定の結果を得ます。たとえば、ユーザはボタンをタップすることでそのボタンを選択したり、長いリストをフリックやドラッグしてスクロールさせます。組み込みのアプリケーションも一貫してこれらのジェスチャを使用しているため、iPhoneユーザはこれらのジェスチャを理解します。したがって、ユーザが慣れ親しんでいることを利用し、かつユーザを混乱させないようにするため、アプリケーションではこれらのジェスチャを適切に使用するべきです。

スワイプやピンチオープンなどのより複雑なジェスチャも組み込みのアプリケーションでは一貫して使用されますが、あまり一般的ではありません。一般に、これらのジェスチャは特定のタスクを実行する唯一の手段としてではなく、タスクを効率的に行うための便法として使用されます。たと

えば、「メール(Mail)」でメッセージのリストを表示しているとき、メッセージを削除するには、ユーザはメッセージのプレビュー行にある「削除(Delete)」ボタンを表示させてからタップします。ユーザは、異なる2つの方法で「削除(Delete)」ボタンを表示することができます。

- ナビゲーションバーの「編集(Edit)」ボタンをタップすると、それぞれのプレビュー行に削除コントロールが表示されます。次に、特定のプレビュー行にある削除コントロールをタップすると、そのメッセージの「削除(Delete)」ボタンが表示されます。
- 特定のプレビュー行の上でスワイプジェスチャを行うと、そのメッセージの「削除(Delete)」ボタンが表示されます。

最初の方法は、ステップが1つ余分に必要ですが、タップするだけで良いうえに、明確にラベル付けされた「編集(Edit)」ボタンから始まるので簡単に見つけることができます。2つめの方法はより速く操作することができますが、ユーザは、より特殊なスワイプジェスチャを習得し、覚えておく必要があります。

したがって、アプリケーションを見つけやすく使いやすいものにするには、必要となるジェスチャを、最も慣れ親しんだ動作であるタップおよびドラッグに限定するようにしてください。また、スワイプやピンチオープンなど、あまり一般的でないジェスチャが、アクションを実行する唯一の手段とならないようにしてください。1つか2つの追加のタップが必要となるにしても、あるアクションを実行するための方法として単純かつ簡単な方法が必ずなければなりません。

多くのアプリケーションでは、新しいジェスチャを定義しないようにすることも同じくらい重要です。そうしたジェスチャが、ユーザが標準のジェスチャから連想するアクションを実行する場合はなおさらです。この推奨事項に対する主な例外は、カスタムジェスチャが適切なものとなりうる没入型アプリケーションです。たとえば、テーブル行の「削除(Delete)」ボタンを表示するためにユーザが円を描くようなジェスチャをする必要があるような生産性型アプリケーションは、紛らわしく、使いにくいものです。一方、ゲームならば、ゲームの駒を回転させるために円を描くようなジェスチャをユーザに求めても差し支えないかもしれません。

表 3-1に、ユーザが行うことのできる標準のジェスチャを示します。これらのジェスチャの意味を再定義しないようにしてください。逆に、アプリケーションでこれらのアクションをサポートする場合、これらのアクションに対応するジェスチャに適切に対応してください。ジェスチャによって生成されるイベントを処理する方法の詳細については、『*iOS Application Programming Guide*』を参照してください。

表 3-1　ユーザがiPhone OSベースのデバイスとやり取りするために使用するジェスチャ

ジェスチャ	アクション
タップ	コントロールまたは項目を押す、または選択する（マウスのシングルクリックに相当）。
ドラッグ	スクロールまたはパンする。
フリック（はじく）	すばやくスクロールまたはパンする。
スワイプ	Table View行で、「削除(Delete)」ボタンを表示する。
ダブルタップ	コンテンツブロックまたは画像を拡大してセンタリングする。 縮小する（すでに拡大されている場合）。
ピンチオープン（つまんで開く）	拡大する。

ジェスチャ	アクション
ピンチクローズ（つまんで閉じる）	縮小する。
タッチアンドホールド	編集可能なテキストで、カーソル位置で拡大されたビューを表示する。

ブランド設定要素を慎重に組み込む

ブランド設定が最も効果的なのは、ささやかで控え目に表現されている場合です。ユーザがiPhoneアプリケーションを使用するのは作業を行うか楽しむためであり、強制的に広告を見せられていると感じたいのではありません。したがって、ブランドの色やイメージを、洗練された、控え目な方法で組み込むように努める必要があります。たとえば、ビューやコントロールにカスタムの色体系を使用することもできます。

例外は、ブランドに注目を集める必要があるアプリケーションアイコンです（アプリケーションアイコンは、アプリケーションをインストールした後にユーザのホーム(Home)画面に表示されるアイコンです）。ユーザはアプリケーションアイコンを頻繁に目にすることになるので、見映えとブランド認知のバランスに時間を割くことは重要です。アプリケーションアイコンの作成に関するいくつかのガイドラインについては、「アプリケーションアイコン」（142 ページ）を参照してください。

一般的なタスクの扱いかた

iPhoneアプリケーションは、多くの一般的なタスクを、デスクトップやラップトップコンピュータのアプリケーションの経験がある人にはなじみのないように思える方法で扱います。このセクションでは、これらのタスクをヒューマンインターフェイスの観点から説明します。これらのガイドラインをコードとして実装する際に必要となる技術的な詳細については、『*iOS Application Programming Guide*』を参照してください。

開始

iPhoneアプリケーションは、ユーザがアプリケーションを遅滞なく使い始められるように、瞬時に開始するべきです。開始の際にiPhoneアプリケーションは、次を実行する必要があります。

- ステータスバーの適切なスタイルを指定する。使用できるスタイルの詳細については、「ステータスバー」（77 ページ）を、コードでスタイルを指定する方法については『*iOS Application Programming Guide*』の「The Information Property List」を参照してください。
- アプリケーションの最初の画面によく似た起動画像を表示する。これにより、アプリケーション起動時間が感覚的に短縮されます。起動画像の作成方法については、「起動画像」（148 ページ）を、コードで起動画像を指定する方法については、『*iOS Application Programming Guide*』の「Application Launch Images」を参照してください。
- 「・・・について(About)」ウインドウやスプラッシュ画面を表示するなど、アプリケーションの即座の使用を妨げるような体験を起動時に与えないようにする。
- デフォルトでは、縦向きで起動する。アプリケーションが横向きでのみ使用されることを想定する場合は、現在のデバイスの向きに関係なく横向きで起動します。必要に応じてユーザにデバイスを横向きに回転してもらいます。

  横向きのみのアプリケーションは、両方向の横向き、つまりホーム(Home)ボタンが右側にある向きと左側にある向きをサポートするべきです。デバイスがすでに物理的に横向きになっている場合、横向きのみのアプリケーションはその向きで起動するべきです。それ以外の場合は、横向きのみのアプリケーションはデフォルトでホーム(Home)ボタンが右側にある向きで起動するべきです。
- アプリケーションを最後に実行したときの状態を復元する。アプリケーション内で以前にいた場所に戻るための手順をユーザが覚えておく必要がないようにすべきです。

重要： アプリケーションをインストールした後に、デバイスの再起動をユーザに要求してはいけません。メモリ使用その他の問題でシステムを再起動するまでアプリケーションの実行が困難な場合は、それらの問題に対処する必要があります。たとえば、うまく調整されたアプリケーションを開発するためのガイドラインについては『*iOS Application Programming Guide*』の「Using Memory Efficiently」を参照してください。

停止

ユーザは、別のアプリケーションを起動することでiPhoneアプリケーションを終了します。特に、ユーザはアプリケーションの終了ボタンをタップしたり、メニューから「終了(Quit)」を選んだりしません。iPhone OS 4.0以降、および特定のデバイスでは、終了するアプリケーションは一時停止状態でバックグラウンドに移動されます。すべてのiPhoneアプリケーションは、以下を実行するべきです。

- いつでも終了できるようにしておく。そのため、できるだけ早く、かつ妥当な頻度でユーザデータを保存します。
- 停止時に現在の状態をできるだけ詳細なレベルまで保存する。たとえば、アプリケーションがスクロールデータを表示する場合、現在のスクロール位置を保存します。

iPhoneアプリケーションをプログラムによって終了させると、ユーザ側ではアプリケーションがクラッシュしたように見えるため、避ける必要があります。ただし、外部環境要因によりアプリケーションが意図したとおりに動作しない場合もあります。これを処理するための最善の方法は、問題の説明と、解決方法をユーザに示す、魅力的な画面を表示することです。これには、2つの利点があります。

- アプリケーションによる不具合ではないことをユーザに伝えるフィードバックを提供する。
- ユーザに制御を渡すことで、状況を改善するためのアクションを取ってアプリケーションを継続するか、ホーム(Home)ボタンを押して別のアプリケーションを起動するかをユーザが決定できるようにする。

特定の状況においてアプリケーションの一部の機能が動作しない場合、ユーザがその機能を有効にしたときに画面または警告のどちらかを表示できます。警告ではあまり柔軟な設計はできませんが、次のようなことができればそれは良い選択となります。

- 状況を簡潔に説明する
- 状況を改善するアクションを実行するボタンを提供する
- 警告は、動作しない機能にユーザがアクセスしようとするときにのみ表示する

すべての警告に当てはまることですが、ユーザが警告を見る回数が少ないほど警告の効果は高くなります。警告の作成については、「Alertの使用」 （90 ページ）を参照してください。

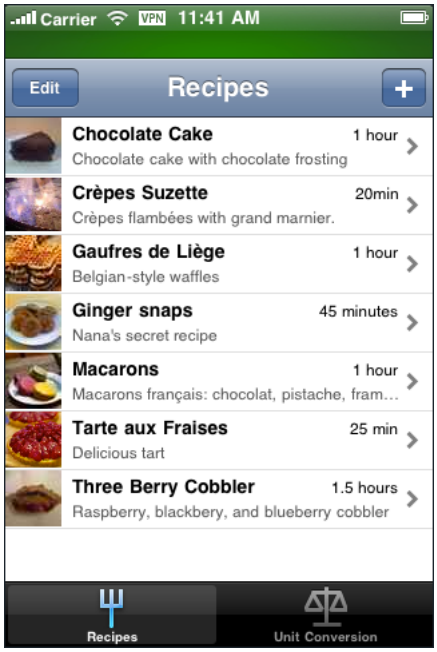
マルチタスキングへの対応

マルチタスキング環境で成功するかどうかは、デバイスのほかのアプリケーションとの間で調和の取れた共存を達成できるかどうかにかかっています。これは、概略レベルで言えば、すべてのアプリケーションが次のことを実現すべきであることを意味します。

- ほかのアプリケーションからの割り込みやオーディオを秩序正しく処理する。
- 停止や再開（つまり、バックグラウンドとの間の移行）がすばやく、スムーズである。
- フォアグラウンドにないときも適切に動作する。

以下に示すガイドラインは、iPhone OS 4.0で導入されたマルチタスキング環境で、アプリケーションを成功に導くのに役立ちます。

- **割り込みに備えておき、再開する準備をしておく。**マルチタスキングにより、バックグラウンドアプリケーションがアプリケーションに割り込む確率が高くなります。広告の存在や高速アプリ切り替えなどの機能も、さらに頻繁に割り込みが発生する原因になります。アプリケーションの現在の状態を保存するのが高速かつ正確であればあるほど、ユーザはよりすばやくアプリケーションを再起動して中断していた場所から継続することができます。
- **UIが2倍の高さのステータスバーに対応できることを確認する。**2倍の高さのステータスバーは、電話の呼び出し中、オーディオの記録中、テザリング中などの状況で表示されます。この高さに対応していないアプリケーションでは、バーがこのように拡大すると、レイアウトの問題が発生する可能性があります。たとえば、UIが下にずれたり、隠れたりします。マルチタスキング環境では、2倍の高さのステータスバーが表示される原因となるアプリケーションが多い可能性があるため、2倍の高さのステータスバーを正しく扱えることが特に重要です。2倍の高さのステータスバーを適切に扱えないビューをすべて見つけて修正するため、テスト中に2倍の高さのステータスバーを表示させることができます（シミュレータを使用してこれを行う方法については、『*iOS Development Guide*』の「Manipulating the Hardware」を参照してください）。



- **ユーザの注意や積極的な関与を必要とするアクティビティを一時停止できるように備えておく。**たとえば、アプリケーションがゲームまたはメディア表示アプリケーションの場合、アプリケーションを切り替えるときに、ユーザがどのようなコンテンツやイベントも見逃さないようにします。ユーザは、ゲームまたはメディア表示アプリケーションに戻ったときに、離れていなかったかのように体験を続けられることを望みます。
- **オーディオが適切に動作することを確認する。**マルチタスキングにより、アプリケーションの実行中にほかのメディアアクティビティが発生する可能性が高くなります。また、割り込みを処理するために、オーディオの一時停止と再開を行う必要が生じる可能性も高くなります。オーディオがユーザの期待に応え、デバイスのほかのオーディオと適切に共存することを確実にするための具体的なガイドラインについては、「サウンドの使用」（61 ページ）を参照してください。
- **ローカルの通知の使用は控えめにする。**アプリケーションは、一時停止中、バックグラウンドで実行中、あるいはまったく実行されていないとしても、特定の時点でLocal Notification（ローカル通知）を送信するように設定できます。ユーザ体験を最適なものにするためには、大量の通知でユーザを悩ませることは避け、通知コンテンツ作成のガイドラインに従います（「Local NotificationおよびPush Notificationの有効化」（54 ページ）を参照）。
- **妥当な場合は、ユーザが開始したタスクをバックグラウンドで終了する。**ユーザがタスクを開始するとき、通常、アプリケーションを切り替えたとしても、タスクが終了することを期待します。アプリケーションが、それ以上ユーザとの対話を必要としない、ユーザが開始したタスクを実行中である場合、一時停止する前にバックグラウンドでタスクを完了する必要があります。

広告のホスティング

iPhone OS 4.0以降では、アプリケーション内に広告を表示することができ、ユーザが広告を見たり、対話操作をしたりした場合に、デベロッパは収入を得ることができます。それには、ユーザがアプリケーションから注意をそらさずに広告を見ようという気になるようにするために、いつどのようにして広告をUIに組み込むのかを計画することが不可欠です。

UIの特定のビューに、広告コンテンツを含むiAdをホストできます。ユーザがこのビュー（**Banner View**と呼ばれます）内の広告をタップすると、ムービーの再生、対話型広告コンテンツの表示、Safariを起動してのWebページ表示など、あらかじめプログラムされたアクションがiAdによって実行されます。アクションにより、UIを覆うコンテンツを表示したり、アプリケーションをバックグラウンドへ移行させることができます。

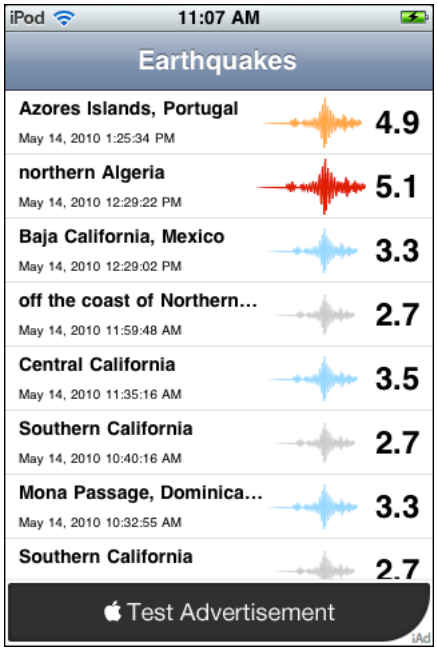
Banner Viewの寸法は次のとおりです。

- 縦向き、320×50ポイント
- 横向き、480×32ポイント

バナー広告とのシームレスな統合の確保と、最適なユーザ体験を提供するため、次のガイドラインに従ってください。

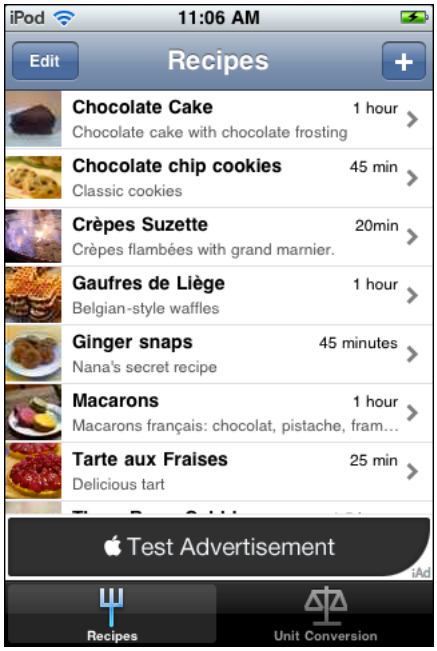
画面の一番下、または一番下近くにBanner Viewを配置する。次のように、この配置は、画面上に表示されるバーの有無によって少し違いがあります。

- 画面の一番下にバーがない場合、Banner Viewは画面の下端に配置します。



- バーがまったくない場合、Banner Viewは画面の下端に配置します。

- ToolbarまたはTab Barがある場合、Banner ViewはToolbarまたはTab Barの真上に配置します。



BannerViewは、アプリケーションで理にかなっている場合に表示する。画面の一番下にBannerViewを配置することが推奨されますが、BannerViewをどの画面に含めるべきかはデベロッパが選んでください。たとえば、アプリケーションの主要タスクの中休みとなるような機能というコンテキストを選ぶのも良いかもしれません。ユーザは、進めているワークフローを中断すると感じなければ、iAd体験に参加する可能性が高くなります。これは、ゲームなどの没入型アプリケーションにとって特に重要です。ゲームの進行に差しさわりのあるような場所にBannerViewを配置することは望ましくないでしょう。

バナー広告は、できる限りどちらの向きでも表示する。ユーザがデバイスの向きを変更することなく、使用するアプリケーションと表示する広告を切り替えられることが最善の方法です。また、両方の向きをサポートすることにより、より広範な広告に対応できるようになります。向きの変更にBanner Viewを確実に対応させる方法については、『*iAd Programming Guide*』を参照してください。

ユーザが広告を見たり、広告との間で対話操作をしている間は、ユーザの注意や対話操作を必要とするアプリケーションアクティビティを一時停止する。ユーザが広告を見ることを選んだ場合、ユーザは、アプリケーションのイベントを見逃したと感じたくはありませんし、アプリケーションが広告に割り込むことを望んでいません。おおまかな目安として、アプリケーションをバックグラウンドに移行する場合に一時停止するようなアクティビティがあれば、そうしたアクティビティを一時停止しておきます。

まれな状況を除いて、広告を停止しない。一般に、ユーザが広告の閲覧と対話操作を行っている間もアプリケーションは実行とイベントの受け取りを継続しているため、早急な対応を緊急に求められるイベントが発生する可能性があります。しかし、進行中の広告の終了が正当化されるシナリオは、まれです。可能性の1つとしては、VoIP (Voice over Internet Protocol)サービスを提供するアプリケーションがあります。このようなアプリケーションでは、着信呼び出しがあったときに実行中の広告をキャンセルするのは、おそらく理にかなっています。

注： 広告のキャンセルは、アプリケーションが受け取れる広告の種類と、デベロッパが受け取れる収入に、悪影響を及ぼす可能性があります。

設定や設定オプションの管理

iPhoneアプリケーションは、ユーザが望むアプリケーション動作を定義する設定や、アプリケーションの機能を変更するためにユーザが指定できる設定オプションを提供することができます。**設定(Settings)**は、ユーザが一度設定したらほとんど（永久に）変更しない情報（アカウント名など）を表します。ユーザは組み込みの「設定(Settings)」アプリケーションでアプリケーション固有の設定を表示します。**設定オプション(Configuration options)**は、ユーザが頻繁に変更する可能性のある値（リストに表示するカテゴリタイプなど）です。したがって、設定オプションはアプリケーションのなかから利用できるようにするべきです。

設定と設定オプションは相互に排他的であると見なさなければなりません。つまり、アプリケーションで設定と設定オプションの両方を提供するべきではありません。

一番良いのは、iPhoneアプリケーションがユーザにまったく設定を要求しないことです。このようなアプリケーションであれば、ユーザはセットアップ情報を指定するように要求されることなく、すぐに使い始めることができます。アプリケーションでこれを実現するために、デベロッパができる設計上の決定は次のようなものです。

- **ユーザの80%のニーズに応えるよう重点的に取り組む。**これができれば、アプリケーションはすでにほとんどのユーザの期待どおりに動作するようにセットアップされているため、大半のユーザは設定を行う必要がありません。一部のユーザだけが必要とするような機能や、ほとんどのユーザが一度しか必要としないような機能がある場合は除外します。
- **ほかの情報源からできる限り多くの情報を得る。**組み込まれているアプリケーションやデバイス設定においてユーザが指定する情報を使用できる場合は、それらの値をシステムに照会します。ユーザにそれらの値を再入力するように要求してはいけません。
- **セットアップ情報の入力を要求しなければならない場合は、アプリケーション内でユーザに入力を求める。**次にできるだけ早く、その情報をアプリケーションの設定に保存します。こうすることで、アプリケーションを終了して「設定(Settings)」を開くことをユーザに強いることなく、アプリケーションをそのまま利用し始めることができます。ユーザがこの情報を後から変更する必要がある場合は、いつでもアプリケーションの設定に移動できます。

ユーザは、アプリケーションを終了してからでないと「設定(Settings)」アプリケーションを開くことはできません。したがって、この動作をユーザに強いるべきではありません。この動作をサポートするシステムアイコンやシステムコントロールはありません。また、そのためのカスタムアイコンやカスタムコントロールの作成は避けることをお勧めします。iPhoneアプリケーションで設定を提供しなければならない場合は、『*iOS Application Programming Guide*』の「The Settings Bundle」を参照して、コードでの設定のサポート方法を習得してください。

注： アプリケーション固有の設定に、ユーザヘルプのコンテンツを含めるべきではありません。

設定とは異なり設定オプションは、ユーザが新しいソースからの情報を参照したり異なる構成で情報を参照したりするために、頻繁に変更される可能性があります。設定オプションにアクセスするためにユーザがアプリケーションを離れることはないので、設定オプションに加えられた変更には動的に反応できます。

設定オプションは、メインユーザインターフェイスまたは画面の背面で提供できます。どの手法が有効かを決定するには、そのオプションが主要な機能を果たすものかどうかと、ユーザがそれらを設定する頻度を判定します。

たとえば、「カレンダー(Calendar)」を利用してユーザは日ごと、週ごと、または月ごとにスケジュールを表示できます。これらのオプションを画面の背面で提供することもできます。しかし、カレンダーのさまざまな部分を表示することはカレンダーの主要な機能であり、ユーザが頻繁に変更する可能性があります。

一方、「天気(Weather)」の主要な機能は、ある都市の現在の天気と6日間の予報を表示することです。気温を摂氏または華氏のどちらで表示するかを選択できることは重要ですが、ユーザがこのオプションを頻繁に変更する可能性はありません。したがって、これをメインユーザインターフェイスに入れるのは有効ではありません。気温目盛のオプションを「天気(Weather)」画面の背面で提供すれば、便利に利用できてかつ邪魔になりません。

コピーとペーストのサポート

iPhone OSには、Text View、Web View、Image Viewに「カット(Cut)」、「コピー(Copy)」、「ペースト(Paste)」、「選択(Select)」、および「全選択(Select All)」の各操作をサポートする編集（ペーストボード）メニューがあります。メニュー内のコマンドにより、ユーザがコンテンツを変更したり、あるアプリケーションからほかのアプリケーションへコンテンツをコピーしたりできます。

編集メニューの動作はアプリケーションに合わせて調整することができます（これらの動作をコードで実装する方法については、『*iOS Application Programming Guide*』の「Copy and Paste Operations」を参照してください）。たとえば、表示するメニューのサブセットを指定したり、メニューの表示位置に影響を及ぼしたりできます。メニュー自体の色や形を制御することはできません。

現在のコンテキストに合ったコマンドを使用する。たとえば何も選択されていない場合、メニューには「コピー(Copy)」や「カット(Cut)」は含まれません。これらは選択範囲が実行対象になるためです。カスタムビューで編集メニューをサポートする場合は、メニューに表示されるコマンドが現在のコンテキストで適切であることを保証するのはデベロッパの責任です。メニューにカスタムコマンドが表示されるように指定することはできません。

表示するメニューをレイアウトに適合させる。UIKitは、利用可能なスペースに応じて挿入位置や選択範囲の上または下に編集メニューを表示し、そのコンテキストとメニューコマンドとの関係がユーザにわかるようにメニューポインタを配置します。メニューの位置はそれを表示する前にプログラムで決定できます。したがって、必要であればUIの重要な部分が隠れないようにすることもできます。

ユーザがメニューを呼び出すために使用できる両方のジェスチャをサポートする。タッチアンドホールドジェスチャは編集メニューを表示するための第一の方法ですが、Text View内の単語をダブルタップしてそれを選択するのと同時にメニューを表示することもできます。カスタムビューで編集メニューをサポートする場合は、両方のジェスチャに対応する必要があります。さらに、ユーザがダブルタップしたときにデフォルトで選択されるオブジェクトを定義することもできます。

編集メニューで利用できるコマンドを実行するUIにボタンを作成するのは避ける。たとえば、「コピー(Copy)」ボタンを提供するよりも編集メニューを使用してユーザのコピー操作ができるようにする方が優れています。それは、アプリケーション内で同じ操作を実行する方法がなぜ2つあるのかとユーザが不思議に思うからです。

ユーザにとって有用な場合、静的なテキストを選択することを検討する。たとえばユーザが画像のキャプションをコピーしたいとは思っても、タブ項目のラベルや画面タイトル（「アカウント(Accounts)」など）をコピーしたいとは考えないでしょう。Text Viewでは、単語単位の選択をデフォルトにするべきです。

ボタンのタイトルを選択可能にしない。ボタンのタイトルが選択可能な場合、ユーザがボタンをアクティブにせずに編集メニューを表示することが難しくなります。一般に、ボタンとして動作する要素を選択可能にする必要はありません。

取り消しとやり直しのサポートと、コピーとペーストのサポートを組み合わせる。ユーザは、気が変わった場合に最近の操作を取り消せるものだと想定しています。編集メニューではそのアクションを実行する前の確認は要求されていないため、ユーザにこれらのアクションの取り消しまたはやり直しを行う機会を与えるべきです（これを行う方法については、「取り消しとやり直しのサポート」（53 ページ）を参照してください）。

取り消しとやり直しのサポート

iPhone OSでは、Text Viewでの入力を取り消したりやり直したりする機能をユーザに提供しています。デバイスをシェイクすると取り消しを起動できます。デバイスをシェイクすると、今入力したものを取り消す、直前に取り消した入力をやり直す、または取り消しをキャンセルすることが可能な警告が表示されます。

UIKitを利用するとより一般的な方法でアプリケーションでの取り消しをサポートできます（この動作をコードで実装する方法については、『*Undo Architecture*』を参照してください）。以下に、デベロッパが指定できることを示します。

- ユーザが取り消しまたはやり直しできるアクション
- 取り消しのためのシェイクジェスチャとしてシェイクイベントをアプリケーションが解釈するタイミング
- サポートする取り消しレベルの数

アプリケーションの取り消しおよびやり直し機能に対して優れたユーザ体験を提供するには、次のことを行います。

- **何を取り消し、またはやり直ししようとしているかを正確にユーザに伝える短い説明フレーズを提示する。**UIKitでは、取り消し(Undo)警告ボタンのタイトルに「取り消す(Undo)」または「やり直す(Redo)」という文字列を自動的に提供しますが、デベロッパは、ユーザが取り消しまたはやり直しできるアクションについて説明する1～2個の単語を提供する必要があります（「キャンセル(Cancel)」ボタンは変更できません）。たとえば、「名前の削除(Delete Name)」や「アドレスの変更(Address Change)」というテキストを提供して、「取り消す－名前の削除(Undo Delete Name)」または「やり直す－アドレスの変更(Redo Address Change)」というボタンタイトルを作成することができます。

  提供するテキストが長くなり過ぎないように注意してください。ボタンのタイトルが長すぎると切り詰められてしまって、ユーザが解読しにくくなります。また、このテキストはボタンのタイトルに表示されるため、タイトル形式の大文字化を使用し、句読点は付けません（簡単に言うと、タイトル形式の大文字化とは、4文字以下の冠詞、等位接続詞、および前置詞を除くすべての単語の1文字目を大文字表記にすることを意味します（英語の場合））。

- **シェイクジェスチャのオーバーロードは避ける。**アプリケーションがシェイクイベントを取り消しのためのシェイクと解釈するタイミングをプログラムで設定することはできますが、ユーザが別のアクションの実行にもシェイクを使用していると、ユーザを混乱させるリスクを負うことになります。

  シェイクジェスチャは、ユーザが取り消しとやり直しを起動する第一の方法ですが、必要であればNavigation Barにシステムが提供する「取り消す(Undo)」ボタンと「やり直す(Redo)」ボタンを含めることもできます。アプリケーションのコンテキスト内でこれらの機能を実行するために明示的な専用ボタンを表示することが不可欠な場合はこのような実装もできますが、これは通常のやりかたではありません。

- **取り消しまたはやり直しできるアクションのコンテキストを検討する。**一般に、ユーザは変更やアクションがすぐに反映されることを期待しています。取り消しおよびやり直し機能は、できる限りユーザの直前のコンテキストに明確に関連していなければなりません。それ以前のコンテキストではいけません。

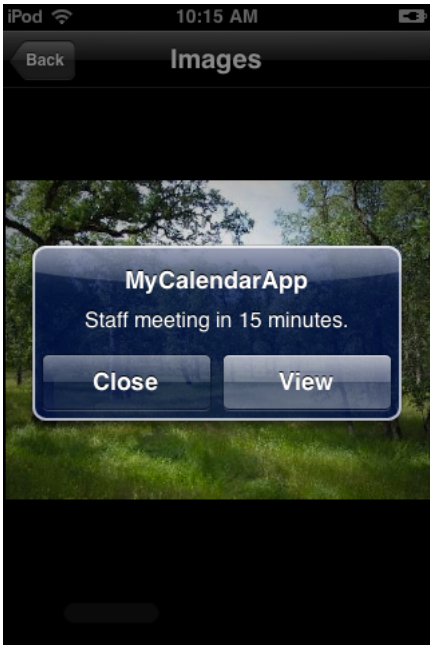
Local NotificationおよびPush Notificationの有効化

Local NotificationとPush Notificationの両方によって、アプリケーションがフォアグラウンドで実行されていない場合に、何らかの情報をユーザに通知できます。たとえば、ユーザに次のようなことを知らせると良いでしょう。

- メッセージの受信
- イベントの発生
- 新しいデータがダウンロード可能
- 何らかのステータスの変化

Local Notificationは、アプリケーションがフォアグラウンドで現在実行中であるかどうかにかかわらず、アプリケーションによってスケジュールされ、同じデバイス上のiPhone OSによって配信されます。たとえば、カレンダー(Calendar)アプリケーションやToDoアプリケーションは、迫っているミーティングや締め切り日をユーザに警告するために、Local Notificationをスケジュールすることができます。

図 4-1 Local Notificationは別のアプリケーションの実行中に受信可能



Push Notificationは、アプリケーションのリモートサーバによって、そのアプリケーションがインストールされているすべてのデバイスに通知をプッシュするApple Push Notificationサービスに送信されます。たとえば、ユーザがリモートの対戦相手とプレイできるゲームでは、すべてのプレーヤに最新の一手を伝えることができます。

Local NotificationまたはPush Notificationを受信したときにアプリケーションがフォアグラウンドで実行されていない場合、次のようにユーザの注意を引くことができます。

- ホーム(Home)画面のアプリケーションのアイコン上のバッジを更新する
- 警告を表示する

バッジの更新や警告を表示するときに、サウンドも再生できます。

アプリケーションがフォアグラウンドで実行されている間も、Local NotificationおよびPush Notificationを受け取ることができますが、アプリケーション固有の方法でユーザに情報を提示します。「設定(Settings)」では、ユーザは、選択したアプリケーションまたはすべてのアプリケーションからのPush Notificationに対して、バッジ、サウンド、警告を有効または無効にできます。これは、ユーザが各アプリケーションの中で指定できるべきであるため、Local Notificationを無効にするための類似の仕組みはありません。

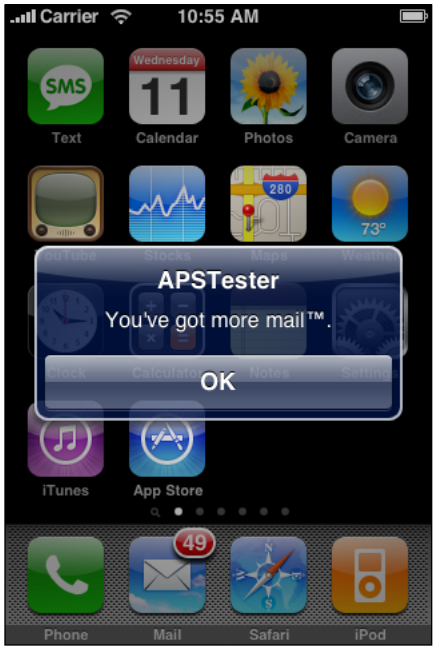
状況に応じて、異なる通知方法が適している場合もあるため、両方の種類を使用できるように備えておくべきです。通常は、次のように使用します。

- **新しい項目の数が有益な情報であり、項目が時間重視でない場合にバッジを使用する。**バッジは、未読メッセージ、割り当てられているタスク、共有ドキュメントの更新数など、注目されるのを待っている項目の数をユーザに通知するための役に立ちます。ユーザは、ホーム(Home)画面を表示しないとバッジを見られないため、時間的な制約のある情報の配信にバッジを使用しないほうが良いでしょう。

バッジの外観や配置を制御することはできません。バッジは、ホーム(Home)画面のアイコンの右上角に表示される赤い小さな楕円です。

バッジの中に入れることができるのは数字のみで、文字や句読点を入れることができません。

- **ユーザが直ちに知ったり対処したりすべき重要な情報を配信する必要がある場合は警告を使用する。**警告は、近く予定されている出来事やステータスの変更について、ユーザに通知する良い方法です。警告はユーザのワークフローに割り込むため、控えめに使用するのが最善です。



ユーザから高く評価される、Local NotificationおよびPush Notificationを作成するには、以下のガイドラインに従います。

バッジのコンテンツを最新に保つ。追加情報が到着したとユーザが勘違いしないように、ユーザが新しい情報に対処したらすぐにバッジを更新することが特に重要です。

警告にカスタムメッセージを提供する。カスタムメッセージは、警告の中央、アプリケーション名（警告の一番上に自動的に表示されます）の下に表示されます。標準の警告のメッセージと同様に（これについては、「警告の設計」（92 ページ）で説明します）、Local NotificationまたはPush Notificationの警告メッセージは次のようにする必要があります。

- ユーザアクションではなく、情報に焦点を当てる。どのボタンをタップすべきかを伝えたり、特定のボタンをタップした結果を説明したりすることは避けてください。
- 1行または2行で十分表示できるように短くする。メッセージが長すぎると、通知警告のスクロールを強制される可能性があります。
- 文章形式の大文字化と、句読点を使用する。可能であれば、完全な文章を使用します。

任意で、アクションボタンにカスタムのタイトルを提供する。警告には、1つまたは2つのボタンを含めることができます。2つのボタンの警告では、左が「Close」ボタン、右がアクションボタン（デフォルトでは「View」というタイトル）になります。1つのボタンを指定すると、警告には「OK」ボタンが表示されます。

アクションボタンをタップすると、警告が消されると同時にアプリケーションが起動されます。「Close」ボタンまたは「OK」ボタンのどちらかをタップすると、アプリケーションを開かずに警告が消されます。

アクションボタンにカスタムのタイトルを使用する場合、アプリケーションが起動されたときに発生するアクションを明確に説明するタイトルを作成してください。たとえば、ゲームでは、ボタンをタップすればユーザが次の一手を実行できる場所が表示されるようにアプリケーションを開くことを示す、「Play」というタイトルを使用します。タイトルは以下に従っていることを確認してください。

- タイトル形式の大文字化を使用する
- 文字が欠けることなくボタン内に収まるような短さにする（ローカライズしたタイトルの長さもテストしてください）

注： カスタムのボタンタイトルは、デバイスがロックされているときに通知を受信した場合にユーザに表示される「slide to view」メッセージに表示させることもできます。メッセージが表示されると、カスタムのタイトルが自動的に小文字に変換され、メッセージ内の「view」という単語の代わりに表示されます。

任意で、起動画像を提供する。既存の起動画像を表示するほかに、ユーザが通知に応えてアプリケーションを起動したときに表示する別の起動画像を指定できます。たとえば、ゲームでは、起動メニュー画面に似た画像ではなく、ゲーム内の画面に似た起動画像を指定できます。この起動画像を指定しない場合、iPhone OSによって、前回のスナップショットあるいはほかの起動画像の1つのどちらかが表示されます（起動画像の作成方法については、「起動画像」（148 ページ）を参照してください）。

必要であれば、バッジまたは警告に付随するサウンドを再生する。サウンドは、ユーザがデバイスの画面を見ていない場合に注意を引くことができます。サウンドは、ユーザが重要と考える通知のために取っておくのが最善です。たとえば、カレンダー(Calendar)アプリケーションは、目前のイベントについてユーザに思い出させる警告として、サウンドを再生することができます。あるいは、共同作業用管理アプリケーションは、遠隔地の同僚が割り当てられた仕事を完了したことを示すためにバッジを更新し、サウンドを再生することができます。

カスタムサウンドを提供することも、組み込まれている警告音を使用することもできます。カスタム警告音を作成する場合は、短く、特徴的で、本格的に作成されたものにしてください（この警告音の技術的な要件については、『*Local and Push Notification Programming Guide*』の「Preparing Custom Alert Sounds」を参照してください）。警告にバイブレーションを付けるかどうかはユーザが制御するため、通知が配信されたときにデバイスをプログラムで強制的に振動させることはできません。

アプリケーションをアクセシブルにする

障害を持つユーザが補助用のアプリケーションやデバイスを利用してアプリケーションを正常に使用できる場合、そのアプリケーションはアクセシビリティに対応していると言えます。iPhone OSベースのデバイスには、障害者を含むすべてのユーザにとってデバイスを使いやすくするための機

能（Visual Voicemail、ズーム(Zoom)、音声コントロール(Voice Control)など）があります。ユーザがこれらの機能の恩恵を受けられるようにするために、アプリケーションでは特別な手順を踏む必要はありません。

Appleの画期的な画面読み上げ技術である、VoiceOverを利用する場合は、話が少し違います。VoiceOverユーザがアプリケーションを最大限に利用できるようにするには、ユーザインターフェイスのビューとコントロールについてのカスタム情報をいくつか提供する必要があります。

幸い、UIKitのコントロールとビューはデフォルトでアクセシブルです。したがって、標準的な要素を完全に標準的な方法で使用すれば追加の作業はほとんどありません。ユーザインターフェイスをカスタマイズすればするほど、ユーザにVoiceOverがアプリケーションを適切に説明できるように、より多くのカスタム情報を提供する必要があります。

重要： アプリケーションをアクセシブルにするには、アプリケーションの使用を手助けする情報をVoiceOverに提供することが必要になります。この作業には、VoiceOverに対応するようにユーザインターフェイスの視覚的な設計を変更することは含まれ*ません*。

iPhoneアプリケーションをVoiceOverユーザにとってアクセシブルにすることは正しいことです。これによってユーザ層を拡大し、さまざまな監督機関が作成しているクセシビリティガイドラインへの対応に役立ちます。

検索機能の提供と検索結果の表示

UIKitでは、検索用に一貫性のあるインターフェイスを表示するSearch Barコントロールを提供していますが、アプリケーションに検索機能を実装するのはデベロッパの責任です（Search Barの詳細については「Search Bar」（129 ページ）を参照してください。コードで検索結果を扱う方法については『*UISearchDisplayController Class Reference*』を参照してください）。検索をユーザにとって有用で便利な体験にするために、そのプロセスの実装方法と結果の表示方法についてここで少し検討します。

一般に、次のことを行う必要があります。

- 検索に常時備えるためにデータのインデックスを作成する。
- ユーザが入力し始めると同時に結果が表示され、入力を続けるにしたがって結果が絞り込まれるようにローカルデータのライブフィルタリングを行う。
- 可能であればユーザの入力に伴ってリモートデータのフィルタリングも行い、結果の表示に1〜2秒以上かかる可能性がある場合はユーザの許可を得るようにする。
- リストの上、またはリスト内のインデックスの上にSearch Barを表示する。
- アプリケーションで明確なモードとして装備すべき主要な機能でない限り、検索用のタブの使用を避ける。

データのライブフィルタリングは一般に優れたユーザ体験を提供しますが、それが常に実用的であるとは限りません。これに該当する場合は、ユーザがキーボードの「検索(Search)」ボタンをタップしたら検索を開始するようにできます。このようにする場合は、検索が停止していないことがユーザにわかるように検索の状況についてのフィードバックを提供するようにします。その方法の1つは、文字による結果をできるだけ早く表示し、取得に時間がかかるデータについてはプレースフォルダ用のコンテンツを表示することです。

たとえば「YouTube」では、ユーザは「検索(Search)」ボタンをタップして映像の検索を開始します。ネットワーク接続の速度が遅い場合、検索中であることがユーザにわかるように、まず回転するアクティビティインジケータと一緒に「読み込み中...(Loading...)」メッセージが表示されます。次に、各行に文字情報の結果（ビデオタイトル、視聴者評価など）と、破線の輪郭で描かれた箱型のカスタム画像が埋め込まれた結果リストが表示されます。ユーザがビデオタイトルのリストをスキャンすると、ダウンロードに伴って破線の箱がビデオのサムネイルに置き換わります。追加データをダウンロードしている間に部分的な検索結果を表示することで、有用な情報をすばやくユーザに提供できます。

本来異なるカテゴリに分類されるデータを扱う場合は、Scope Barを提供できます。1つのScope Barには、最大4つのスコープボタンを含めることができます。各ボタンは1つのカテゴリを表します。たとえば、「メール(Mail)」には1つのScope Barがあります。これを利用してユーザはメッセージの「差出人(From)」、「宛先(To)」、または「件名(Subject)」の各フィールドに検索範囲を限定したり、すべてのフィールドを含むように検索範囲を拡大したりできます。Scope Brの提供は、ユーザが検索範囲を限定するのに役立つ場合や、検索結果の件数を大幅に削減できる場合に検討します（Scope Barをコードで実装する方法については『*UISearchBar Class Reference*』を参照してください）。

ユーザの位置情報の使用

コンテンツに物理的な位置を自動的に付加したり、現在近くにいる友人を見つけたりするアプリケーション機能を、ユーザは高く評価しています。また、位置情報を他人と共有したくない場合にこれらの機能を無効にできることも、ユーザは高く評価しています。ユーザは、「設定(Settings)」＞「一般(General)」の「位置情報サービス(Location Services)」を利用して、システムレベルでの物理的な位置情報へのアクセスを許可（または拒否）できます。

ユーザが「位置情報サービス(Location Services)」をオフにした後で位置情報が必要なアプリケーション機能を使用した場合は、その機能を使用するには環境設定を変更しなければならないことを知らせる警告がユーザに表示されます。この警告によってアプリケーション内での設定変更ができるわけではありません。ユーザは「設定(Settings)」を起動して環境設定を変更しなければなりません。こうすることで、ユーザは確実に、位置情報の使用をシステム全体に許可したことを十分に認識できます。

位置情報サービス(Location Services)をオンにしなければならない理由をユーザに理解してもらいやすくするには、この警告を、現在位置を明らかに知る必要がある機能をユーザが使用しようとしたときにのみ表示するのが最善です。たとえば、位置情報サービス(Location Services)がオフの場合でもユーザは「マップ(Maps)」を使用できますが、自分の現在位置の検索や追跡の機能にアクセスすると、この警告が表示されます。

位置情報サービス(Location Services)がオフの場合、iPhone OSはアプリケーションが位置情報に初めてアクセスしようとした時点で警告を表示します。Core Locationフレームワークには、不必要または不適切な警告を発するのを避けられるように、ユーザの環境設定を取得する手段が用意されています（このプログラミングインターフェイスの詳細については『*Core Location Framework Reference*』を参照してください）。

ユーザの環境設定の情報があれば、位置情報を必要とする機能にできるだけ近いタイミングで警告を発したり警告を出すのを完全に避けたりすることができます。

- 位置情報がないとアプリケーションの主要な機能を実行できない場合は、アプリケーションの起動直後にユーザに警告を表示するのが最適です。ユーザはアプリケーションの主要な機能が位置情報の検出に依存することを理解しているので、この警告を邪魔だとは思わないでしょう。

- ユーザの位置情報がアプリケーションの主要な機能に含まれない場合は、単純にその情報を使用する機能を制限することもできます。たとえば位置情報サービス(Location Services)がオフの場合、「カメラ(Camera)」では撮影した写真にユーザの位置情報を追加する機能を自動的にオフにします。そのために環境設定を変更するまでユーザが写真を撮れないということはありません。写真に位置情報を追加できることは便利ですが、必須ではないからです。
- ある機能が動作するために位置情報が必要な場合、ユーザが実際にその機能を選択する前に警告を発するような呼び出しをプログラムで行うことは避けるべきです（ユーザの環境設定を取得する呼び出しによって、警告が発せられることはありません）。このように、位置情報が必要ないと思われる機能をユーザが実行している場合に、なぜアプリケーションで位置情報を必要なのだろうという疑問をユーザに抱かせる行為は避けます。

向きの変更の処理

ユーザはiPhone OSベースのデバイスをいつでも回転させることができます。また、ユーザは表示されているコンテンツが適切に反応するものと考えています。iPhoneアプリケーションでは、次のことを必ず行ってください。

- 加速度センサーの値に注意します（加速度センサーの詳細および加速度センサーのプログラミングインターフェイスの詳細については、『*iOS Application Programming Guide*』を参照してください）。適切な場合、アプリケーションはデバイスのすべての向きの変化に反応するべきです。
- アプリケーションのユーザインターフェイスのうち1つの向きでのみ表示する部分がある場合、その領域はその向きで表示し、デバイスの向きの変化には反応しないようにするのが適切です。たとえばユーザがiPodビデオを選択して表示した場合、iPodビデオは現在のデバイスの向きにかかわらず横向きで表示されます。これは、ユーザに、デバイスを物理的に回転させてビデオを見るように促す合図となります。この例で重要なのは、iPodには「回転(Rotate now)」ボタンはなくビデオが横向きで表示されることで、デバイスを回転させる必要があることをユーザに気付かせている点です。

  アプリケーションのユーザインターフェイスのうち、特定の向きである必要がある部分が正しく表示されるように、ユーザがデバイスを物理的に回転できるようにしてください。ユーザにデバイスを回転させるように指示するコントロールを作成したり、ジェスチャを定義することは避けてください。
- 向きの変更をよりスムーズかつ高速に実行するには、1ステップの向き変更プロセスを利用します。ただし画面レイアウトが非常に複雑な場合は、向きの変更を行うときにクロスフェードトランジションを実行することもできます。この1ステッププロセスをコードでサポートする方法については『*UIViewController Class Reference*』を参照してください。
- 通常、ユーザはより多くの情報を見るためにデバイスを横向きにします。単にコンテンツの幅を広くするだけではユーザの期待に応えることはできません。それよりもコンテンツが画面に合うように、必要に応じてテキスト行の折り返しを変更したり、ユーザインターフェイスのレイアウトを再配置したりするべきです。

サウンドの使用

ユーザは、サウンドの大きさやサウンドを鳴らすかどうかを決定します。時には現在の設定がサウンドを鳴らさない設定であっても、ユーザが特定のサウンドを期待する場合があります。たとえば、ユーザは、自分が設定した着信音やアラームを聴くことを、常に期待しています。要するに、ユーザは自分が要求したサウンドは聴きたいが、自分が要求したのではないサウンドは聴きたくないのです。

この希望に応えるために、iPhone OSでは次のことを実現するプログラミングインターフェイスを提供しています。

- アプリケーションのサウンドをデバイス上のその他のサウンドに適応させる方法を記述する
- アプリケーションのサウンドを確実にユーザの期待に従って再生する

アプリケーションでのサウンドの処理方法を決定する前に、デバイスのコントロールを調節したり、ヘッドフォンやヘッドセットなどの外部デバイスを使用するときに、ユーザがアプリケーションとデバイスにどのような動作を期待しているかを理解する必要があります。

着信／サイレントスイッチ－ユーザの期待

次のような場合に、ユーザは着信／サイレントスイッチを使用してデバイスをサイレントモードにできます。

- 電話の着信音やメッセージの受信音など、予期しないサウンドに割り込まれるのを避けたい場合。
- キーボードその他のフィードバック音、付随的な音、アプリケーションの起動音など、ユーザアクションの結果として発生するサウンドを避ける場合。
- 付随的な音やサウンドトラックを含む、ゲームの使用に不可欠でないゲーム音を避ける場合。

たとえば、映画館では館内のほかの人達の迷惑にならないようにデバイスをサイレントモードに切り替えます。このような状況でも、ユーザはデバイス上のアプリケーションを使えるのが望ましいことですが、電話の着信音や新規メッセージの受信音など、明示的に要求したのではない予期せぬサウンドに驚かされたくはありません。

しかし、単に明示的に音を出すことが目的のユーザアクションの結果生じるサウンドは、着信／サイレントスイッチでは消音*されません*。たとえば、ユーザの特徴としては次のようなものがあります。

- メディア専用アプリケーションでのメディア再生はユーザが明示的に要求したメディア再生であるため、着信／サイレントスイッチによって消音されません。
- 時計のアラームはユーザによって明示的に設定されたものであるため、着信／サイレントスイッチによって消音されません。
- 言語学習アプリケーションのサウンドクリップはそれを聴くためにユーザが明確なアクションを行うため、着信／サイレントスイッチによって消音されません。
- オーディオチャットアプリケーションの会話はユーザがオーディオチャットをするためだけにそのようなアプリケーションを起動しているため、着信／サイレントスイッチによって消音されません。

ユーザが明示的に要求したサウンドを鳴らすのが適切かどうかの決定はデバイスではなくユーザに任せているため、この動作はユーザコントロールの原則に従っています。

音量ボタン－ユーザの期待

ユーザは、デバイスの音量ボタンを使用してデバイスで再生できるすべてのサウンド（音楽、アプリケーションのサウンド、デバイスのサウンド）の音量を調節できます。着信／サイレントスイッチの設定に関係なく、ユーザは音量ボタンを使用してどんな音でも消音できることを意味します。

音量ボタンを使用してアプリケーションで現在再生中のオーディオを調節すると、システム全体の音量（ただし、電話の着信音の音量は除く）も変化します（オーディオが再生されていないときに音量ボタンを使用すると、電話の着信音の音量が変化します）。

音量スライダを表示する必要がある場合、`MPVolumeView`クラスを使用するときは、システムに用意されている利用可能な音量スライダを使用してください。現在アクティブなオーディオ出力デバイスが音量の制御をサポートしていない場合、音量スライダは適切なデバイス名で置き換えられます。

時には、オーディオ出力において、最適な出力のために相対的で独立した音量レベルをアプリケーションで調節する必要が生じることもあります。しかし、最終的なオーディオ出力の音量は、音量ボタンによって調節されたものか音量スライダによって調節されたものかに関わらず、常にシステム音量によって制御される必要があります。つまり、アプリケーションのオーディオ出力の制御はいずれにしてもユーザの手中にあるということです。

ヘッドセットとヘッドフォン－ユーザの期待

ユーザは、個人的にサウンドを聴いたり両手を自由に使えたりするようにヘッドセットやヘッドフォンを接続します。これらのアクセサリを接続しているか外しているかによって、アプリケーションの動作に対するユーザの期待が異なります。

ユーザがヘッドセットやヘッドフォンを接続した場合、ユーザは現在のオーディオを聴き続けたい（ただし、個人的に）と思っています。このためユーザは、現在オーディオを再生しているアプリケーションが再生を一時停止することなく継続することを期待します。

ユーザがヘッドセットやヘッドフォンを外すとき、ユーザはそれまで聴いていたサウンドを自動的にほかの人たちと共有しようとは思っていません。このため、現在オーディオを再生しているアプリケーションが一時停止して、準備が整ったときに明示的に再び再生ができることを期待します。

ワイヤレスオーディオ－ユーザの期待

ユーザは、サウンドを個人的に聴き、手を自由にしておきたいという、有線のヘッドセットとヘッドフォンを使用するのと同じ理由で、無線のヘッドセットとヘッドフォンを使用します。

また、ユーザは、無線のヘッドセットのユーザ体験に対して、同様の期待を持ちます。

- ユーザが無線のオーディオデバイスに接続した場合、ユーザは現在のオーディオを聴き続けたい（ただし、個人的に）と思っています。この状況では、ユーザは、一時停止することなくオーディオが再生されることを期待します。

- ユーザが、無線デバイスとの接続を解除する場合（または、受信範囲外に出るか、オフにした場合）、ユーザはそれまで聴いていたサウンドを自動的にほかの人たちと共有しようとは思っていません。このとき、現在のオーディオの再生が一時停止して、準備が整ったときに明示的に再び再生できることを期待します。

ユーザは、無線オーディオデバイスを物理的に抜き差ししないものの、別のオーディオ経路を選べることを期待します。これを処理するため、iPhone OSでは、ユーザがオーディオ出力経路を選択できるコントロールが自動的に表示されます（このコントロールがアプリケーションに確実に表示されるようにするには、`MPVolumeView`クラスを使用する必要があります）。別のオーディオ経路の選択はユーザが開始するアクションであるため、ユーザは現在のオーディオの再生が一時停止することなく続くことを期待します。

アプリケーションのオーディオ動作の定義

サウンドが、アプリケーションのユーザ体験や機能性を向上させたり、不可欠であったりする場合、オーディオを、デバイスのオーディオ環境にどのように適合させるか、また、ユーザアクションに対してどのように応えさせるかを判断する必要があります。次のような場合に、オーディオをどのように動作させるかに基づいて、この判断を行います。

- デバイスがロックされているか、消音に切り替えられている
- 現在ほかのオーディオが再生されている
- アプリケーションが、オーディオの入力と出力を、順にまたは同時に処理する必要がある

これらの状況において、アプリケーションのオーディオの動作方法を制御するには、Audio Session ServicesまたはAVAudioSessionクラスを使用します。これらのプログラミングインターフェイスは、サウンドを作成するのではなく、オーディオがデバイス上のオーディオとやり取りを行える方法と、割り込みとデバイス構成における変更に対応する方法を表現するのに役立ちます（これらのオーディオプログラミングインターフェイスをコードで使用する方法については、『*Audio Session Programming Guide*』を参照してください）。

アプリケーションが、機能に付随するUIサウンドエフェクトのみを作成する場合、System Sound Servicesを使用できます。System Sound Servicesは、警告とUIサウンドを生成し、バイブレーションを起動させるiPhone OSテクノロジーです。これら以外の目的には適しておらず、生成されるサウンドは、Audio Session Servicesによって管理されるものではありません（このテクノロジーの使用方法を示すサンプルプロジェクトについては、*Audio UI Sounds (SysSound)*を参照してください）。

重要： オーディオの作成に使用するテクノロジーやオーディオ動作の定義に関係なく、電話は常に現在実行中のアプリケーションに割り込みをかけることができます。これは、どのアプリケーションもユーザが着信呼び出しを受けるのを妨げるべきではないためです。

Audio Session Servicesでは、**オーディオセッション**は、アプリケーションとシステムとの間でオーディオの仲介役として機能します。オーディオセッションの最も重要な側面の1つは、アプリケーションのオーディオの動作が定義される**カテゴリ**です。

Audio Session Servicesの利点の実現と、ユーザが期待するオーディオ体験の提供を行うには、アプリケーションのオーディオの動作を最もよく表すカテゴリを選択する必要があります。これは、アプリケーションがフォアグラウンドでのみオーディオを再生する場合、バックグラウンドでもオーディオを再生できる場合のどちらにも該当します。この選択を行うにあたっては、次のガイドラインに従います。

- **オーディオセッションカテゴリは、その一連の具体的な振る舞いではなく、そのカテゴリが持つ意味合いに基づいて選択する。**これにより、アプリケーションを確実にユーザの期待に従って動作させることができます。さらに、将来、一連の振る舞いに改良が加えられた場合に、アプリケーションが正しく動作する最善の可能性が与えられます。
- **まれなケースとして、オーディオセッションにプロパティを追加して、カテゴリの標準的な動作を変える。**カテゴリの標準的な動作は、ほとんどのユーザの期待を表しているため、その動作を変更する前に慎重に検討してください。たとえば、オーディオがほかのすべてのオーディオよりも確実に音量が大きいようにしておくために（電話のオーディオを除く）、`kAudioSessionProperty_OtherMixableAudioShouldDuck`プロパティを追加できます（オーディオセッションプロパティの詳細については、『*Audio Session Programming Guide*』の「Fine-Tuning the Category」を参照してください）。
- **デバイスの現在のオーディオ環境のカテゴリ選択に基づいて検討する。**たとえば、これは、サウンドトラックの代わりに別のオーディオを聴いている間、ユーザがアプリケーションを使用できる場合に意味があります。これを行う場合、ユーザに聴いている音楽を強制的に止めさせたり、アプリケーションの起動時にサウンドトラックを明示的に選ばせたりすることは避けます。
- **通常、アプリケーションの実行中はカテゴリの変更は避ける。**カテゴリを変更する主な理由としては、アプリケーションが時折、録音と再生をサポートする必要があるような場合が考えられます。その場合、Play and Recordカテゴリを選択するよりは、RecordカテゴリとPlaybackカテゴリを必要に応じて切り替えるほうが良いかもしれません。これは、Recordカテゴリを選択すると、録音の進行中に警告音（テキストメッセージ受信の警告音など）が鳴らないためです。

選んだカテゴリに関わらず、必要な場合にのみオーディオセッションをアクティブにし、不要な場合は非アクティブにすることが重要です。これにより、ユーザに高品質なオーディオ体験を提供できます。

表 4-1は、使用可能なオーディオセッションカテゴリの一覧を示しています。iPhone OSは、デフォルトで、Solo Ambientカテゴリをオーディオセッションに割り当てます。

注： スペースの都合上、表 4-1では、各カテゴリ名の後半部分のみを掲載しています。各カテゴリの実際のシンボル名は、`AVAudioSessionCategory`で始まります。さらに、`MixWithOthers`プロパティの実際のシンボル名は、`kAudioSessionProperty_OverrideCategoryMixWithOthers`です。

表 4-1 オーディオの動作に影響するオーディオセッションカテゴリ

カテゴリ	意味	着信／サイレントスイッチとロックによる消音	ほかのオーディオと合成可能	バックグラウンドで許可
`SoloAmbient`	サウンドがアプリケーションの機能性を高め、ほかのオーディオを消音する	○	×	×
`Ambient`	サウンドがアプリケーションの機能性を高めるが、ほかのオーディオを消音しない	○	○	×

カテゴリ	意味	着信／サイレントスイッチとロックによる消音	ほかのオーディオと合成可能	バックグラウンドで許可
`Playback`	サウンドがアプリケーションの機能性に不可欠で、ほかのオーディオと合成可能	×	×（デフォルト） ○（`MixWithOthers` プロパティが追加された場合）	○
`Record`	オーディオはユーザが録音	×	×	○
`PlayAndRecord`	サウンドが順番または同時に行うオーディオの入力と出力を表す。	×	×（デフォルト） ○（`MixWithOthers` プロパティが追加された場合）	○
`AudioProcessing`	アプリケーションはハードウェア支援オーディオエンコーディングを実行（再生または録音は行わない）	-	×	○ *

* Audio Processingカテゴリを選択してバックグラウンドでオーディオ処理を実行する場合、終了するまではアプリケーションが一時停止しないようにする必要があります。これを行う方法については、『*Local and Push Notification Programming Guide*』の「Executing Code in the Background」を参照してください。

ここで、ユーザに評価されるようなオーディオ体験を提供する、オーディオセッションカテゴリの選びかたについてのシナリオをいくつか示します。

シナリオ1：新しい言語の習得を支援する教育アプリケーションの場合。次の機能を提供します。

- ユーザが特定のコントロールをタップしたときに再生されるフィードバック音
- ユーザが正しい発音の例を聴く場合に再生される、録音済みの単語およびフレーズ

このアプリケーションでは、サウンドは主要な機能として不可欠です。ユーザは、習得する言語の単語とフレーズを聴くためにこのアプリケーションを使用します。したがって、着信／サイレントスイッチが消音に設定されていたり、デバイスがロックされていたとしても、サウンドが再生される必要があります。ユーザは明瞭なサウンドを聴く必要があるため、ほかのオーディオの再生は消音されていることを期待します。

このアプリケーションに対してユーザが期待するオーディオ体験を作り出すには、Playbackカテゴリを使用します。このカテゴリの設定を調整して、ほかのオーディオとの合成を許すことも可能ですが、ユーザが明示的に選んで聴く教育コンテンツにほかのオーディオが干渉しないように、このアプリケーションはデフォルトの動作を使用するべきです。

シナリオ2：VoIP (Voice over Internet Protocol)アプリケーションの場合。次の機能を提供します。

- オーディオ入力を許可する機能
- オーディオを再生する機能

このアプリケーションでは、サウンドは主要な機能として不可欠です。ユーザはこのアプリケーションを使用して互いに通信します。多くの場合、ユーザはそのとき別のアプリケーションを使用しています。ユーザは、着信／サイレントスイッチがサイレントに設定されていたり、デバイスがロックされていたりした場合でも電話を取れることを期待し、その電話の間、ほかのオーディオが消音されていることを期待します。また、アプリケーションがバックグラウンドにある場合でも、電話を取ったり、電話で話し続けたりできることを期待します。

このアプリケーションでユーザが期待するオーディオ体験を演出するには、Play and Recordカテゴリを使用します。

シナリオ3：ユーザがキャラクタを導いてさまざまな作業を行えるようにするゲームの場合。次の機能を提供します。

- ゲームプレイのさまざまなサウンドエフェクト
- 音楽のサウンドトラック

このアプリケーションでは、サウンドがユーザ体験の向上に大きく寄与しますが、主たるタスクに不可欠ではありません。また、ユーザは、ゲームを消音でプレイしたり、ゲームのサウンドトラックの代わりに自分の音楽ライブラリの曲を聴きながらプレイすることを評価する可能性があります。

最善策は、アプリケーションの起動時に、ユーザがほかのオーディオを聴いているかどうか確認することです。ほかのオーディオを聴くか、ゲームのサウンドトラックを聴くかを、ユーザに尋ねてはいけません。代わりに、Audio Session Services関数`AudioSessionGetProperty`を使用して、`kAudioSessionProperty_OtherAudioIsPlaying`プロパティの状態を照会します。この照会への応答に基づいて、AmbientカテゴリまたはSolo Ambientカテゴリを選ぶことができます（どちらのカテゴリでも、ユーザはゲームを消音でプレイできます）。

- ユーザがほかのオーディオを聴いている場合、ユーザはそのオーディオを聞き続けることを望んでおり、ゲームのサウンドトラックを代わりに聴かされることは評価しないと仮定すべきです。この状況では、Ambientカテゴリを選びます。
- アプリケーションの起動時に、ユーザがほかのオーディオを聴いていない場合、Solo Ambientカテゴリを選びます。

シナリオ4：ユーザの目的地まで、正確でリアルタイムのナビゲーション指示を提供するアプリケーションの場合。次の機能を提供します。

- 旅の各ステップでの音声案内
- いくつかのフィードバック音
- ユーザが自分のオーディオを聴き続けられる機能

このアプリケーションでは、アプリケーションがバックグラウンドにあるかどうかにかかわらず、音声によるナビゲーション指示が主要なタスクです。このため、デバイスがロックされていたり、着信／サイレントスイッチがサイレントに設定されている場合や、アプリケーションがバックグラウンドにある間に、オーディオの再生を許可するPlaybackカテゴリを使用します。

ユーザがこのアプリケーションの使用中にほかのオーディオを聴けるようにするには、kAudioSessionProperty_OverrideCategoryMixWithOthersプロパティを追加します。ただし、ユーザが、現在再生中のオーディオにかぶせて、音声の指示を聴けるようにする必要があります。そのためには、オーディオセッションにkAudioSessionProperty_OtherMixableAudioShouldDuckプロパティを適用できます。これにより、現在再生中のすべてのオーディオより大きな音量のオーディオを使用できます（電話のオーディオを除く）。

シナリオ5：Webサイトにユーザがテキストとグラフィックスをアップロードできるブログアプリケーションの場合。次の機能を提供します。

- 起動時の短いサウンドファイル
- ユーザアクションに付随する各種の短いサウンドエフェクト（投稿がアップロードされたときに再生されるサウンドなど）
- 投稿に失敗したときに再生される警告音

このアプリケーションでは、サウンドがユーザ体験の向上に寄与しますが、それは付随的なものです。主要なタスクはオーディオとは何の関係もなく、アプリケーションを正常に使用するためにユーザにサウンドが聴こえる必要はありません。このシナリオでは、System Sound Servicesを使用してサウンドを鳴らします。これは、アプリケーションのすべてのサウンドのオーディオコンテキストが、このテクノロジの本来の目的に従っているためです。すなわち、ユーザの期待通り、デバイスのロックと着信／サイレントスイッチに従うUIサウンドエフェクトと警告音を鳴らすことです。

オーディオ割り込みの管理

現在再生中のオーディオが、別のアプリケーションのオーディオによって割り込まれることがあります。たとえば、電話の着信は、電話の間、現在のアプリケーションのオーディオに割り込みます。マルチタスキング環境では、このようなオーディオ割り込みの頻度が高くなる可能性があります。

ユーザに評価されるオーディオ体験を提供するため、iPhone OSはアプリケーションに以下のことを任せます。

- アプリケーションが発生させる可能性のあるオーディオ割り込みのタイプを明らかにする
- オーディオ割り込み終了後にアプリケーションが継続される場合の適切な対応

すべてのアプリケーションは、発生させる可能性のあるオーディオ割り込みのタイプを明らかにする必要がありますが、すべてのアプリケーションが、オーディオ割り込みの終了に対応する方法を決める必要はありません。これは、ほとんどのタイプのアプリケーションにとって、オーディオ割り込みの終了に対する適切な対応がオーディオ再生の再開であるためです。メディア再生が主要な機能または部分的な機能であるアプリケーションで、メディア再生コントロールを提供するアプリケーションのみが、適切な対応を決める追加のステップを必要とします。

概念的には、割り込みの原因となったオーディオのタイプと、割り込みが終了したときにユーザが期待する特定のアプリケーションの対応方法に応じて、オーディオ割り込みには次の2つのタイプがあります。

- **再開可能な割り込み。**再開可能な割り込みは、主要なリスニング体験の一時的な中休みとユーザがみなすオーディオによって発生します。

再開可能な割り込みが終了すると、メディア再生コントロールを表示するアプリケーションは、割り込みが発生したときの動作を再開するべきです（オーディオを再生するか、一時停止の状態を続けるか）。メディア再生コントロールのないアプリケーションは、オーディオの再生を再開するべきです。

たとえば、ユーザが音楽再生アプリケーションを聴いているとき、曲の途中でVoIP呼び出しを着信したとします。ユーザは、通話中は音楽再生アプリケーションが消音されることを期待して、電話を取ります。電話が終わると、ユーザは曲の再生が自動的に再開されることを再生アプリケーションに期待します。これは、電話ではなく音楽が主要なリスニング体験の構成要素であり、かつ電話の着信前に音楽を一時停止していなかったからです。逆に、ユーザが電話の着信前に音楽の再生を一時停止していた場合、電話の終了後、音楽が一時停止されたままであることが期待されます。

再開可能な割り込みが発生する可能性のあるアプリケーションのそのほかの例として、アラーム、オーディオによる指示（音声運転案内）、そのほかの断続的なオーディオを再生するアプリケーションがあります。

- **再開不能な割り込み。**再開不能な割り込みは、メディア再生アプリケーションのオーディオなど、主要なリスニング体験とユーザがみなすオーディオによって発生します。

  再開不能な割り込みの終了後、メディア再生コントロールを表示するアプリケーションは、オーディオの再生を再開してはいけません。メディア再生コントロールのないアプリケーションは、オーディオの再生を再開するべきです。

  たとえば、ユーザが音楽再生アプリケーション（音楽アプリ1）を聴いているときに、別の音楽再生アプリケーション（音楽アプリ2）によって割り込まれたとします。割り込みに応えて、ユーザは、しばらく音楽アプリ2を聴くことを決めます。音楽アプリ2の終了後、ユーザは音楽アプリ1が再生を自動的に再開することは期待しません。ユーザが意図的に主要なリスニング体験を音楽アプリ2にしたためです。

次のガイドラインは、オーディオ割り込み終了後、どのような情報を提供し、どのように継続するかを判断するのに役立ちます。

アプリケーションが発生させるオーディオ割り込みのタイプを識別する。オーディオが終了したとき、次の2つのいずれかの方法を使用して、オーディオセッションを非アクティブにすることで、これを行います。

- アプリケーションが再開可能な割り込みを発生させた場合、`AVAudioSessionSetActiveFlags_NotifyOthersOnDeactivation`フラグを使用して、オーディオセッションを非アクティブにします。
- アプリケーションが再開不能な割り込みを発生させた場合、いずれのフラグも使用せずに、オーディオセッションを非アクティブにします。

この情報を提供することは、妥当であれば、iPhone OSがオーディオの再生を割り込まれたアプリケーションに自動的に再開する機能を与えるのに役立ちます。

オーディオ割り込み終了時に、オーディオを再開するかどうかを判断する。この決定は、アプリケーションで提供するオーディオのユーザ体験に基づいて行います。

- アプリケーションに、ユーザがオーディオの再生や一時停止を行うためのメディア再生コントロールを表示する場合、オーディオ割り込み終了時に、`AVAudioSessionInterruptionFlags_ShouldResume`フラグを確認する必要があります。

  Should Resumeフラグを受け取った場合、次のことを行います。

- 割り込まれたときにアプリケーションがオーディオの再生をアクティブにしていた場合、オーディオの再生を再開する
- 割り込まれたときにアプリケーションがオーディオの再生をアクティブにしていなかった場合、オーディオの再生を再開しない

- アプリケーションに、ユーザがオーディオの再生や一時停止を行うためのメディア再生コントロールを表示しない場合、オーディオ割り込み終了時に、直前のオーディオの再生を常に再開するべきです。Should Resumeフラグの有無を確認する必要はありません。

  たとえば、サウンドトラックを再生するゲームは、割り込み後、サウンドトラックの再生を自動的に再開するべきです。

メディアリモートコントロールイベントを処理する（適切な場合）

iPhone OS 4.0以降では、ユーザがiPhone OSのメディアコントロールまたはアクセサリコントロール（ヘッドセットコントロールなど）を使用する場合、アプリケーションはリモートコントロールイベントを受け取れるようになりました。この機能により、アプリケーションが現在フォアグラウンドとバックグラウンドのどちらでオーディオを再生していても、アプリケーションはUIを通さないユーザ入力を受け付けることができます。

メディア再生アプリケーションは、アプリケーションがバックグラウンドでオーディオを再生している場合は特に、これらのイベントに適切に対応する必要があります。

この特権に関連する責任を果たすには、次のガイドラインに従ってください。

リモートコントロールイベントを受け取る資格は、それが理にかなう場合のみに限定する。たとえば、ユーザがコンテンツを読んだり、情報を検索したり、オーディオを聴いたりできるアプリケーションの場合、ユーザがオーディオコンテキストにいる場合にのみリモートコントロールイベントを受け付けるようにすべきです。ユーザがオーディオコンテキストから離れたら、イベントを受け取る資格を返上するべきです。これにより、ユーザはアプリケーションの非オーディオコンテキストにいる間、別のアプリケーションのオーディオを聴くことが（そしてそれをヘッドセットコントロールを使って制御）できます。

イベントがアプリケーションにおいて無意味の場合でも、イベントの目的を変えてはならない。ユーザは、iPhone OSのメディアコントロールとアクセサリコントロールが、すべてのアプリケーションで一貫性を持って機能することを期待します。アプリケーションが必要としないイベントを処理する必要はありませんが、処理を行うイベントは、ユーザの期待する体験をもたらす必要があります。イベントの意味を定義しなおしてしまうと、ユーザを混乱させ、アプリケーションを終了しないと抜け出せないような未知の状態に導いてしまう危険があります。

選択肢の提供

iPhone OSには、ユーザが選択を行うのを支援するいくつかの要素が含まれています。アプリケーションで選択肢を提供する必要がある場合、ユーザはすでにこの動作に慣れているためこれらの選択手法を使用するべきです。一般に、アプリケーションメニューやラジオボタンのセットなど、デ

スクトップコンピュータのアプリケーションで見られる選択コントロールの外観や動作を再現するべきではありません。iPhone OSでは、ユーザに選択肢を提示するために使用できる次の要素を提供しています。

- **リスト**（つまり、Table View）。ユーザは、リスト内の行をタップして項目を選択します。リストは、任意の数の選択肢を表示するのに適しています。アプリケーションでTable Viewを使用する方法の詳細については、「Table View」（101 ページ）を参照してください。
- **ピッカー**（日付と時刻のピッカーを含みます）。それぞれのホイールが、複数部分で構成される値（年月日で構成されるカレンダーの日付など）の希望する部分が表示されるまで、ユーザはピッカーのホイールを回します。iPhoneアプリケーションでのピッカーの使用の詳細については、「日付と時刻のピッカー」（120ページ）および「ピッカー」（126ページ）を参照してください。
- **Switchコントロール。**ユーザは、2つの値のどちらか一方が表示されるように、片方の端からもう片方の端までSwitchコントロールをスライドさせます。Switchコントロールは、リスト内の単純な選択肢を提供することを目的としています。Switchコントロールの詳細については、「Switchコントロール」（111 ページ）を参照してください。

ライセンス契約または免責条項の提供

iPhoneアプリケーションに対するエンドユーザライセンス契約(EULA)を提供する場合、ユーザがアプリケーションを購入する前にそのエンドユーザライセンス契約を参照できるように、App Storeで表示されることに注意してください。

可能であれば、ユーザがアプリケーションを最初に起動するときに、EULAへの合意を示すことをユーザに求めることを避けてください。そうすれば、ユーザは、先延ばしされることなくアプリケーションを楽しめます。ただし、これは望ましいユーザ体験ではありますが、すべての状況で実現可能ではないかもしれません。アプリケーションの中でライセンス契約を表示しなければならない場合、ユーザインターフェイスに対して違和感のないように、また、ユーザの不便が最小限になるような方法で行うようにしてください。

同様に、免責条項を提供する必要がある場合、ビジネスニーズと優れたユーザ体験を維持することのバランスをとるようにしてください。可能であれば、App Storeで利用できるように、アプリケーションの説明またはEULAの中に免責条項を記載してください。

iPhoneアプリケーションのユーザインターフェイスの設計

iPhone OSのユーザインターフェイスには、ビューとコントロールが含まれます。**ビュー**は、厳密に定義された機能セットを持つコンテンツ領域を提供します。**コントロール**は、即時のアクションまたは目に見える結果を引き起こすグラフィックオブジェクトです。アプリケーションのすべてのビューとコントロールは、アプリケーションの1つのウインドウに含まれますが、ユーザはそれらを**画面**に表示してやり取りします。この画面は、アプリケーションの異なる視覚的状態におおよそ対応しています。

iPhone OSでは、これらのユーザインターフェイス要素の標準的な外観を定義し、ユーザが期待する一貫した動作を提供します。使用可能なユーザインターフェイス要素のタイプ、およびアプリケーションのユーザインタフェースを作成するためにそれらをどのように使用するかを知るには、このパートIIの各章を参照してください。

アプリケーションユーザインターフェイスの簡単な紹介

個々のビューおよびコントロールの詳細にとりかかる前に、これらの要素がどのように連携するか、およびユーザがこれらの要素にどのような動作を期待しているのかを高いレベルで理解しておくと役に立ちます。この章では、大多数のアプリケーションの基本要素を構成するビューについて、ビューがどこに属し、どのように使用されるかについて触れながら紹介します。

個々のユーザインターフェイス要素の外観、動作、使用方法のガイドラインの詳細については、後続の章を参照してください。各ユーザインターフェイス要素がどのように使用されるよう設計されているのかを理解しておけば、アプリケーションで要素を正しく使用し、適切な場合にはニーズに合わせてカスタマイズするのに役立ちます。

アプリケーション画面とそのコンテンツ

どのアプリケーションにも、そのタイプにかかわらず、アプリケーションウインドウがあります。このウインドウは、プログラミングの視点から見れば、アプリケーションのすべての情報が表示される背景を提供します。しかし、ユーザはこのウインドウを認識しません。代わりに、操作の対象となる複数の画面の集まりとしてアプリケーションを体験します。

画面は、プログラム上の構成要素ではなく、アプリケーションの個別の視覚的な状態またはモードに対応するものと考えることができます。ユーザは、情報階層を行き来したり、Tab Barの各種タブをタップしたり、「Info」ボタンをタップして背面の設定オプションを表示したりするときに、個々の画面を見ることができます。

アプリケーションのスタイルによりますが、多数の画面が必要な場合や、ごくわずかな数の画面が必要な場合があります。たとえば、「メール(Mail)」は、メッセージ作成画面のほかにアカウント画面、各アカウントのメールボックスを一覧表示する画面、各メールボックスの内容を一覧表示する画面、および各メッセージに対する画面を表示できます。これに対して、「株価(Stocks)」アプリケーションは、企業の一覧および株の実績グラフを表示する画面と、アプリケーションの設定情報を表示する画面のみ表示します。

ユーザのほとんどは、アプリケーション画面とデバイス画面を同じものと考えます。しかし、アプリケーション画面の内容は、デバイス画面の範囲を超えることができ、その場合にはユーザが画面をスクロールする必要があります。たとえば、「電話(Phone)」アプリケーションの「連絡先(Contacts)」画面は、デバイス画面の何枚分もの名前を列挙する可能性がありますが、アプリケーションの1つの画面です。

各アプリケーション画面は、ビューとコントロールのさまざまな組み合わせを含むことができます。ビューによっては、ほかのどこにも属さない特定のコントロールが含まれているものがあります。また、コントロールによっては、さまざまなビューで使用できるものがあります。

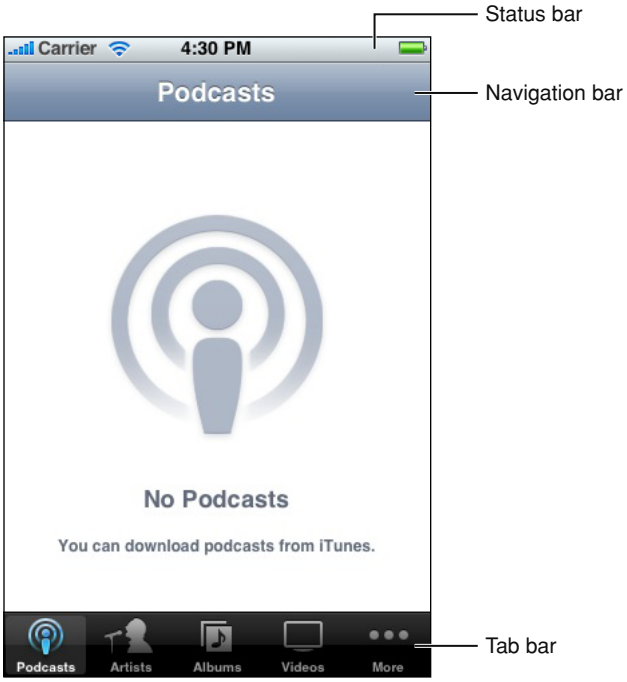
Alert、Action Sheet、およびモーダルビューは、ほかのほとんどのビューのようにアプリケーション画面内に存在せず、代わりにアプリケーション画面やビューの手前に浮かぶ特殊なビューです。これらのビューの詳細については、「Alert、Action Sheet、およびモーダルビュー」（89 ページ）を参照してください。

以下の4種類のビューには、アプリケーションのユーザインターフェイスにおいて特別なステータスがあります。ただし、これらのビューはすべてのアプリケーションに含まれる必要はなく、すべてのアプリケーションで常に可視である必要もありません。

- **ステータスバー。**ステータスバーは、技術的に言えばアプリケーションウインドウには属さない固有のビューですが、アプリケーションではステータスバーの外観をある程度カスタマイズできます。詳細については、「ステータスバー」（77 ページ）を参照してください。
- **Navigation Bar。**この任意で使用できるビューはステータスバーのすぐ下に表示され、タイトル、ボタン、およびSegmented Controlを含めることができます。詳細については、「Navigation Bar」（78 ページ）を参照してください。
- **Tab Bar。**この任意で使用できるビューは画面の下の端に表示され、アプリケーションの異なるモードを有効にする複数のセグメントがあります。詳細については、「Tab Bar」（83 ページ）を参照してください。
- **Toolbar。**この任意で使用できるビューは画面の下の端に表示され、アプリケーションの現在のコンテキストにおいて特定のアクションを実行する複数のコントロールを備えています。詳細については、「Toolbar」（81 ページ）を参照してください。

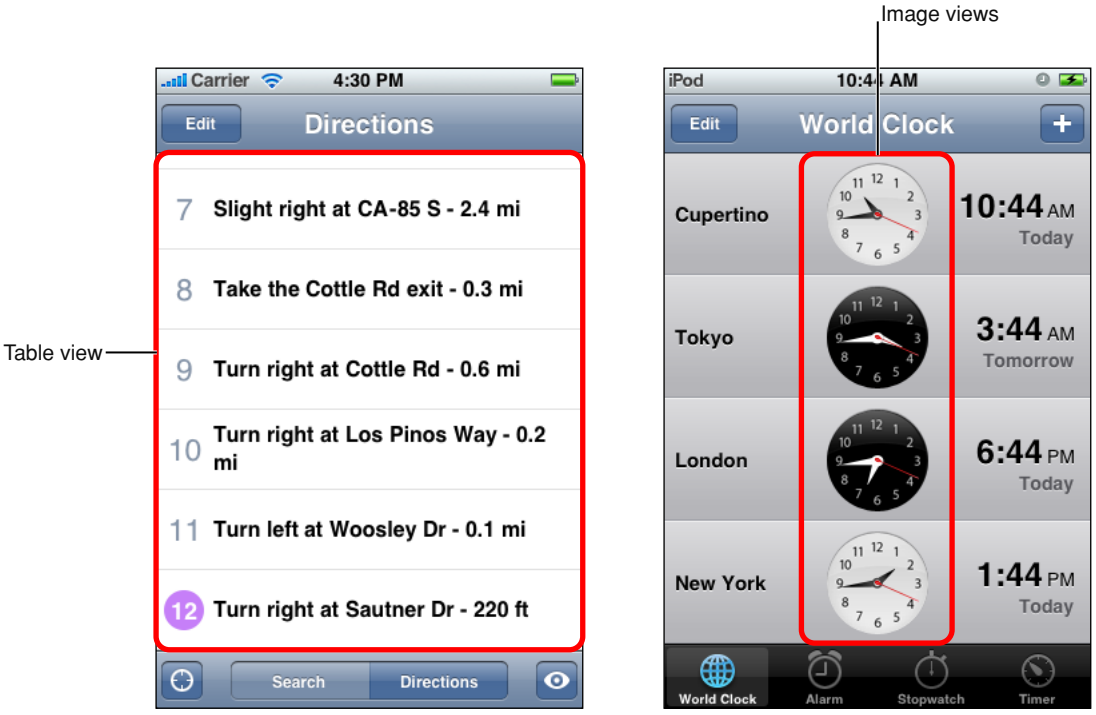
図5-1に、アプリケーション画面のこれらの3つのビューを示します。このアプリケーションでToolbarを使用した場合、ToolbarはTab Barの代わりに表示されます。

図 5-1　ステータスバー、Navigation Bar、およびTab Barを含むアプリケーション画面



これら4つのビューの何らかの組み合せを表示するアプリケーションでは、Navigation Barの下端からToolbarまたはTab Barの上端までの間の領域を**コンテンツ領域**と考えることができます。アプリケーション画面のこの領域に、Table View、Web View、Image Viewなどのコンテンツを表示する任意のビューを含めることができます。図5-2に、iPhone OSで使用可能なコンテンツ領域ビューの2つの例、Table ViewとImage Viewのスタイルを示します。これらのビューの動作と外観、およびこれらに関連付けられたコントロールの詳細については、「Table View、Text View、およびWeb View」（101 ページ）を参照してください。

図 5-2 2種類のコンテンツ領域ビュー



すでに述べたように、特定のビューでのみ使用可能なコントロールがいくつかあります。そのようなコントロールの例が、ディスクロージャインジケータです。ディスクロージャインジケータは、Table Viewにおいて特定の用途があります。図 8-1 （101 ページ）の左側の図に、ディスクロージャインジケータの例を確認できます（>のように見えます）。これらのコントロールについては、関連するビューを扱うセクションで説明しています。ただし、これらのほかにも詳細ディスクロージャインジケータなど、さまざまな用途のコントロールがいくつかあります。使用可能なコントロールの詳細については、「アプリケーションコントロール」（119ページ）を参照してください。

アプリケーション画面のビューおよびコントロールの使用

iPhone OSでは、UIKitによってビューおよびコントロールの動作とデフォルトの外観が決まります。可能な限り、UIKitが提供する標準のユーザインターフェイス要素を使用し、推奨されている使用法に従ってください。そうすることには、2つの重要な利点があります。

- ユーザは、標準のビューおよびコントロールの外観や動作に慣れています。ユーザの慣れているユーザインターフェイス要素を使用すれば、ユーザはアプリケーションの使いかたを学ぶ際に以前の体験を頼りにすることができます。
- iPhone OSの標準のビューおよびコントロールの外観や動作に変更があった場合にも、アプリケーションは機能し続け、あったとしてもわずかな作業で自動的に最新に見えます。

コントロールの多くは、通常は配色やコンテンツなどの何らかのカスタマイズをサポートします（テキストラベルや画像の追加など）。没入型アプリケーションを開発している場合には、デフォルトのコントロールとは完全に異なるコントロールを作成するのが妥当です。これは独自の環境を作成しているからであり、その環境のコントロール方法を発見することが、没入型アプリケーションにおいてユーザが期待する体験だからです。

ただし、一般的に標準のアクションを実行するコントロールの外観を根本的に変更することは避けるべきです。標準のアクションを実行するためになじみのないコントロールを使用すると、ユーザはそれらの使用方法を知るのに時間を費やす必要があり、それが標準のコントロールが実行しない何かを実行するとしたら、それが何であるかを疑問に思うことでしょう。

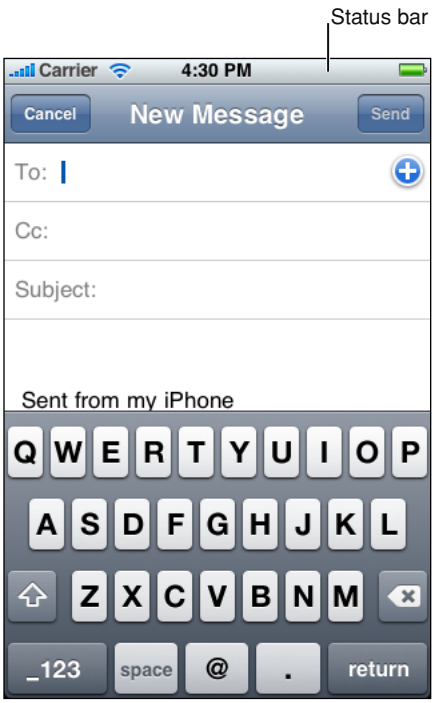
Navigation Bar、Tab Bar、Toolbar、およびステータスバー

ステータスバー、Navigation Bar、Tab Bar、およびToolbarは、iPhoneアプリケーションで特別に定義された外観と動作を持つビューです。これらのバーは、すべてのアプリケーションに提供する必要はありません（通常、没入型アプリケーションではこれらのいずれのバーも表示されません）が、提供する場合は適切に使用することが重要です。それはこれらのバーがiPhone OSベースのデバイスユーザにとってなじみのある足場を提供するためです。iPhone OSベースのデバイスユーザは、これらのビューが表示する情報や実行する機能のタイプに慣れています。

ステータスバー

ステータスバーは、携帯電話の電波の強さ、現在のネットワーク接続、バッテリー残量など、デバイスに関する重要な情報をユーザに示します。図 6-1に、ステータスバーの例を示します。

図 6-1　ステータスバーはユーザにとって重要な情報を含む



フルスクリーンの没入型アプリケーションは、ステータスバーを非表示にできますが、そのような設計の決定による影響を注意深く検討する必要があります。ユーザは、デバイスの現在のバッテリー残量を確認できることを期待します。この情報を非表示にし、これを確認するためにユーザにアプリケーションの終了を求めるのは、理想的なユーザ体験ではありません。

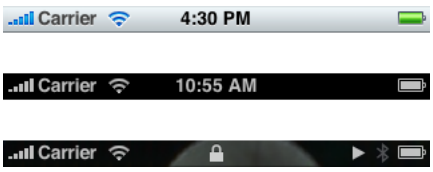
たとえば、「写真(Photos)」はカメラロールから個々の写真をフルスクリーンビューで表示し、ステータスバー、Navigation Bar、およびToolbarは数秒後に次第に消えます。「写真(Photos)」では、ユーザはそれらとやり取りをすることではなく、内容を見ることに集中するので、これは適切です。しかし、ユーザは画面を1回タップすればステータスバー、Navigation Bar、およびToolbarを復元できます。

アプリケーションでステータスバーを非表示にすることがある場合には、この動作のユーザ体験を利用して1回のタップでそれらを再表示できるようにするとよいでしょう。よほどの理由がない限り、ステータスバーを再表示するためのカスタムジェスチャを定義するのは避けるのが最善です。ユーザがそのようなカスタムジェスチャに気付いたり、それを覚えたりしている可能性は低いからです。

ステータスバーの内容を制御できる範囲は限られていますが、外観とある程度までの動作はカスタマイズできます。具体的には次のことが可能です。

- ネットワークアクティビティインジケータを表示するかどうかを指定する。アプリケーションが2〜3秒を超えるネットワーク操作を実行している場合には、ネットワークアクティビティインジケータを表示するべきです。操作がそれよりも早く終了する場合には、ユーザがネットワークアクティビティインジケータの存在に気づく前に消える可能性があるため、ネットワークアクティビティインジケータを表示する必要はありません（コードでは、`UIApplication`メソッドの`networkActivityIndicatorVisible`を使用してこのインジケータの表示を制御します）。
- ステータスバーの色を指定する。グレイ（デフォルト色）、不透明な黒、または半透明な黒（アルファ値が0.5の黒）を選ぶことができます。図 6-2に、これらのスタイルを示します（ステータスバーのスタイルを指定するには、`Info.plist`ファイル内の値を設定します。設定方法の詳細については、『*iOS Application Programming Guide*』を参照してください）。
- 現在のステータスバーの色が新しい色に変化するときに、その変化をアニメーション化するかどうかを設定する（アニメーションによって、古いステータスバーが画面から消えるまで上にスライドし、その間に新しいステータスバーが所定の位置にスライドします）。

図 6-2　ステータスバーの3つのスタイル



アプリケーションのほかの部分と調和するステータスバーの外観を選択してください。たとえば、Navigation Barが不透明の場合は半透明のステータスバーを使用するのを避けます。

Navigation Bar

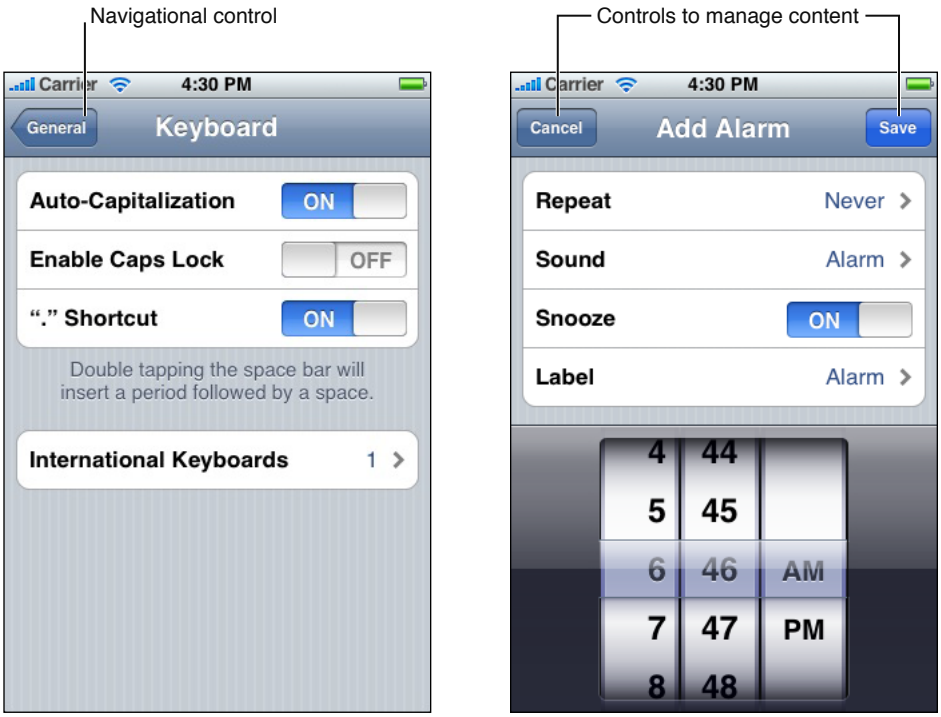
Navigation Barはアプリケーション画面の上の端、ステータスバーのすぐ下に表示されます。Navigation Barは、通常、現在のビューのタイトルを表示し、ナビゲーションコントロールのほかに必要に応じてビューのコンテンツを操作するコントロールを含むことができます。Navigation Barは、生産性型アプリケーション（「生産性型アプリケーション」　（19 ページ）で説明します)では特に役立ちます。これは、生産性型アプリケーションでは、一般的に情報が階層構造に整理されるからです。

Navigation Barには、以下の2つの目的があります。

- アプリケーションの個々のビューの間を行き来することを可能にすること
- ビューの項目を管理するコントロールを提供すること

図 6-3に、これら両方の使用例を示します。

図 6-3 Navigation Barには、ナビゲーションコントロールおよびコンテンツを管理するためのコントロールを含められる



Navigation Barのコンテンツ

Navigation Barは図 6-4に示すように、現在のビューのタイトルだけをビューの幅の中央に表示できます。生産性型アプリケーションの最初のビューでは、ユーザがまだ別の場所に移動していないため最初のビューのタイトルだけを表示するNavigation Barを含めるようにします。

図 6-4 Navigation Barは現在のビューのタイトルを表示する

ユーザが別のビューに移動するとすぐに、Navigation Barはそのタイトルを新しい場所のタイトルに変更し、前の場所のタイトルを示すラベルの付いた戻るボタンを提供するようにします。たとえば、図 6-5は、「一般(General)」設定の「日付と時刻(Date & Time)」の設定におけるNavigation Barを示しています。

図 6-5 Navigation Barにはナビゲーションコントロールを含められる

標準的な戻るボタンは、以前の画面に戻るための確実な方法をユーザに提供します。したがって、このボタンの動作を変更するのは避けることが重要です。特に、図 6-6に示すようなマルチセグメントの戻るボタンを作成するのは避けるべきです。

図 6-6 マルチセグメントの戻るボタンは避ける

マルチセグメントの戻るボタンを使用すると、次のような問題が生じます。

- マルチセグメントの戻るボタンは幅が広いため、現在の画面のタイトルを表示する余地がなくなる。
- 個々のセグメントが選択されている状態を表す手段がない。
- セグメント数が増えるほど1セグメント当たりのヒットリージョンが狭くなるため、ユーザが特定のセグメントをタップすることが困難になる。
- ユーザが階層の深いところに移動するにつれて、表示するレベルを選ぶことが難しくなる。

一種のブレッドクラムパスを表示するマルチセグメントの戻るボタンがないとユーザが迷子になるのではないかと思ったら、それはおそらく、ユーザが必要な情報を見つけるために、情報階層をあまりにも深く下がる必要が生じているということです。これを解決するには、情報階層をフラットにする必要があります。

戻るボタンのほかに、Navigation Barにはタイトルの右に第2のボタンを含めることもできます。（アプリケーションが階層の移動をサポートしていないため）戻るボタンを表示する必要がない場合は、代わりにビューのコンテンツに影響を及ぼすボタン（「編集(Edit)」ボタンなど）を、タイトルの左に表示することができます。図 6-7に、この例を示します。

図 6-7 Navigation Barにはビューのコンテンツを管理するコントロールを含められる

Navigation Barをアプリケーションで実装する方法については、『*View Controller Programming Guide for iOS*』の「Navigation Controllers」を参照してください。

上記の図に見られるように、Navigation Barのボタンは囲み枠があります。iPhone OSでは、このスタイルをボーダー付きスタイルと呼びます。Navigation Barのすべてのコントロールには、ボーダー付きスタイルを使用するべきです。実のところ、Navigation Barにプレーンな（ボーダーなしの）コントロールを配置すると、自動的にボーダー付きスタイルに変換されます。

Navigation Barのボタンに使用するアイコンは、自分独自のデザインにすることも、iPhone OSが提供する定義済みのボタンを利用することもできます。使用可能なボタンの詳細については、「Toolbar およびNavigation Barで使用可能な標準ボタン」（136 ページ）を参照してください。

Navigation Barに表示されるすべてのテキストのフォントは指定できますが、最大限読み易くするためにシステムフォントを使用することをお勧めします。適切なUIKitプログラミングインターフェイスを使用してNavigation Barを作成すると、タイトルの表示にシステムフォントが自動的に使用されます。

Navigation Barのサイズと色

デバイスの向きを縦向きから横向きに変更すると、Navigation Barの高さも自動的に変更されます（この高さをプログラムで指定するべきではありません）。横向きの場合は、Navigation Barが薄くなり、画面のコンテンツの表示スペースが広くなります。Navigation Barのコントロール用のアイコンを設計する場合や画面のレイアウトを設計する場合は、この高さの違いを考慮してください。

アプリケーション全体の外観やアプリケーション内のほかのバー（Toolbar、Tab Bar、ステータスバー）と調和するようにNavigation Barの色や透過性を指定できます。独自の色を使用することもできますし、次の標準色の1つを選択することもできます。

- 青（デフォルトの色）
- 黒

アプリケーションの外観を引き立たせるために、Navigation Barを半透明にすることができます。半透明のNavigation Barを使用すると、画面の表示領域が広くなったような印象を与えます。これは特に横向きの場合に適しています。半透明のNavigation Barを不透明の黒いステータスバーと組み合わせるのは避けてください（半透明のNavigation Barと不透明のグレイのステータスバーを一緒に表示するのはかまいません）。

Navigation Barの外観とアプリケーション内のその他のバーとの一貫性を保つように努力します。たとえば、半透明のNavigation Barを使用する場合は、不透明のToolbarと組み合わせないようにします。また、Navigation Barの色や半透明を同じ向きの別の画面で変更するのは避けます。

Toolbar

現在のコンテキストにおいてユーザが実行できるいくつかのアクションをアプリケーションが提供する場合、Toolbarを提供すると都合がよい場合があります。Toolbarは画面の下端に表示され、現在のビューのオブジェクトに関連するアクションを実行するボタンが含まれます。Toolbarは、アプリケーションの異なるモード間の切り替えに使用すべきではありません。モード間の切り替えが必要な場合は、代わりにTab Barを使用します（詳細については、「Tab Bar」（83 ページ）を参照してください）。

たとえば、ユーザが「メール(Mail)」でメッセージを表示すると、アプリケーションは新規メールの着信確認および新規メッセージの作成を行うための項目のほかに、メッセージの削除、返信、移動を行うための項目を含むToolbarを提供します。このようにして、ユーザはメッセージビューのコンテキストにとどまりながら、電子メールを管理するために必要なコマンドにアクセスできます。図6-8に、その様子を示します。

図 6-8 Toolbarはタスクのコンテキストに合った機能を提供する

Toolbarのコンテンツ

Toolbarは、Toolbarの幅に合わせてToolbarの項目を等間隔で表示します。ユーザが必要な項目を簡単にタップできるように、Toolbarに表示する項目数を制限するのがよいでしょう。ユーザインターフェイス要素のヒットリージョンは44x44ピクセルが推奨されている点に留意してください。このため、ツールバー項目は5つ以下とするのが妥当です。図6-9に、ToolbarにおけるToolbar項目の適切な間隔の例を示します。

図 6-9 適切な間隔のToolbar項目

図 6-8 （82 ページ）および図 6-9の項目はどちらも枠がありません。iPhone OSでは、このスタイルをプレーンスタイルと呼びます（ボーダー付きスタイルの例としては、図 6-7 （80 ページ）のボタンを参考にしてください）。Toolbarのボタンには、ボーダー付きスタイルとプレーンスタイルのいずれかを使用できますが、同じToolbarで両方のスタイルを混在させるべきではありません。

Toolbarのボタンに使用するアイコンは、自分独自のデザインにすることも、iPhone OSが提供する定義済みのボタンを利用することもできます（使用可能なボタンの詳細については、「ToolbarおよびNavigation Barで使用可能な標準ボタン」 （136 ページ）を参照してください）。Toolbarのカスタムボタンを作成する場合は、バランスのとれた魅力的な外観を実現できるようにボタンのサイズを可能な限り均等にします。

Toolbarのサイズと色

デバイスの向きを縦向きから横向きに変更すると、Toolbarの高さが自動的に変更されます（この高さをプログラムで指定するべきではありません）。横向きの場合はToolbarが薄くなり、画面のコンテンツを表示するスペースが広くなります。Toolbarボタン用のアイコンを設計したり画面のレイアウトを設計したりする場合は、この高さの違いを考慮してください。

アプリケーション全体の外観やアプリケーション内のほかのバー（Navigation Bar、Tab Bar、ステータスバー）と調和するように、Toolbarの色や透過性を指定できます。独自の色を使用することもできますし、次の標準色の1つを選択することもできます。

- 青（デフォルトの色）
- 黒

アプリケーションの外観を引き立たせるために、Toolbarを半透明にすることができます。半透明のToolbarを使用すると、画面の表示領域が広くなったような印象を与えます。これは、特に横向きの場合にメリットがあります。

Toolbarの外観とアプリケーション内のその他のバーとの一貫性を保つように努力します。たとえば、半透明のToolbarを使用する場合は、不透明のNavigation Barと組み合わせないようにします。また、Toolbarの色や半透明を、同じ向きの個々の画面で変更するのは避けます。

Tab Bar

アプリケーションにおいて同じデータセットを異なる切り口で見せたり、アプリケーション全体としての機能に関連する各種サブタスクを提供したりする場合、Tab Barを使用すると良い場合があります。Tab Barは画面の下端に表示されます。

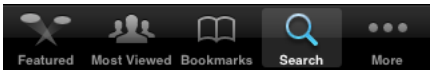
Tab Barにより、アプリケーションの異なるモード間（ビュー間）の切り替えが可能になります。また、ユーザはアプリケーションのどこからでもこれらのモードにアクセスできるようにするべきです。しかし、現在のモードの要素を操作するためのボタンを含んだToolbarとしてTab Barを使用するべきではありません（Toolbarの詳細については、「Toolbar」（81 ページ）を参照してください）。

たとえば、iPhoneのiPodではTab Barを使用して、ユーザは「Podcast (Podcasts)」、「アーティスト(Artists)」、「ビデオ(Videos)」、および「プレイリスト(Playlists)」など、メディアコレクションのどの部分に焦点を当てるかを選べます。一方、「時計(Clock)」アプリケーションでは、Tab Barを使用して、ユーザはアプリケーションの4つの機能、すなわち「世界時計(World Clock)」、「アラーム(Alarm)」、「ストップウォッチ(Stopwatch)」、および「タイマー(Timer)」にアクセスできます。図 6-10に、Tab Barのタブを選択することで「時計(Clock)」のビューがどのように切り替わるかを示します。図 6-10に示す個々の「時計(Clock)」モードにおいて、Tab Barが表示されたままである点に注目してください。これによりユーザは、現在どのモードなのかを簡単に確認でき、現在のモードに関係なくすべての時計モードにアクセスできます。

図 6-10 Tab Barはアプリケーションのビューを切り替える

Tab Barは、アイコンおよびテキストをタブに表示します。すべてのタブの幅は等しく、黒色の背景色を表示します。タブが選択されると、その背景が明るくなりタブの画像がハイライトされます。図 6-11に、これがどのように見えるのかを示します。

図 6-11 Tab Barで選択されたタブ



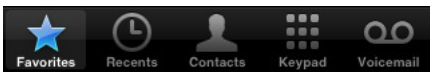
注： Tab Barの不透明度や高さは、デバイスの向きに応じて変わることはありません。

iPhone OSでは、図 6-11の「おすすめ(Featured)」や「よく使う項目(Bookmarks)」というラベルの項目のように、さまざまなタブ用アイコンを提供しています。これらをアイコンを使用する場合は、それに定義されている意味に従って使用してください。利用可能なTab Barのアイコンの詳細については、「Tab Barで使用可能な標準アイコン」（138 ページ）を参照してください。

追加タブの提供

アプリケーションのTab Barに含まれるタブが5つ以下の場合は、図 6-12に示すように、iPhone OSはすべてのタブをTab Barに均等な間隔で表示できます。

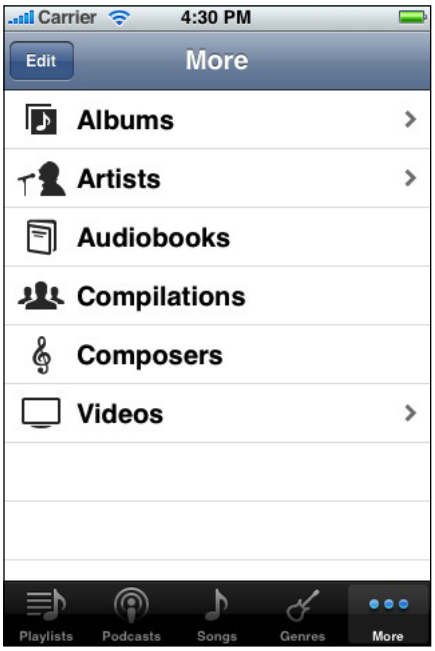
図 6-12 iPhone OSはTab Barに最大5つまでのタブを表示する



アプリケーションのTab Barに含まれるタブが5つを超える場合、図 6-11 （84 ページ）に示すように、iPhone OSはそのうちの4つをTab Barに表示し「その他(More)」タブを追加します。

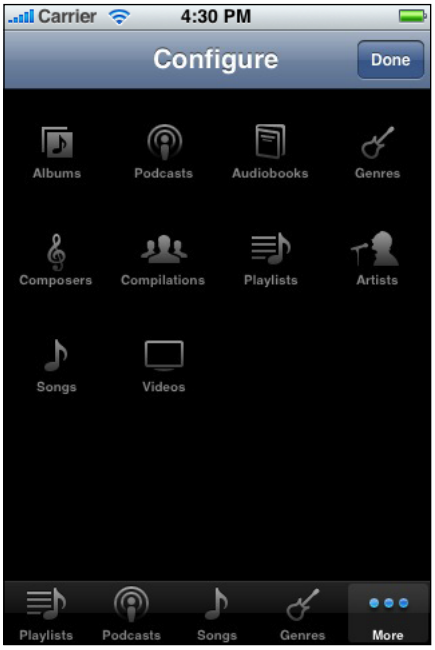
ユーザが「その他(More)」タブをタップすると、図 6-13に示すように、別の画面に追加のタブのリストが表示されます。

図 6-13 ユーザが「その他(More)」タブをタップすると、追加のタブが表示される



「その他(More)」画面には「編集(Edit)」ボタンが含まれており、ユーザはこれを利用して、最もよく使用するタブを表示するようにTab Barを設定できます。たとえば、図6-14は、「iPod」アプリケーションの「その他(More)」画面の「編集(Edit)」ボタンをタップした後に表示される「配置変更(Configure)」画面を示しています。

図 6-14 アプリケーションのタブが5つを超える場合、ユーザはTab Barに表示するお気に入りのタブを選択できる



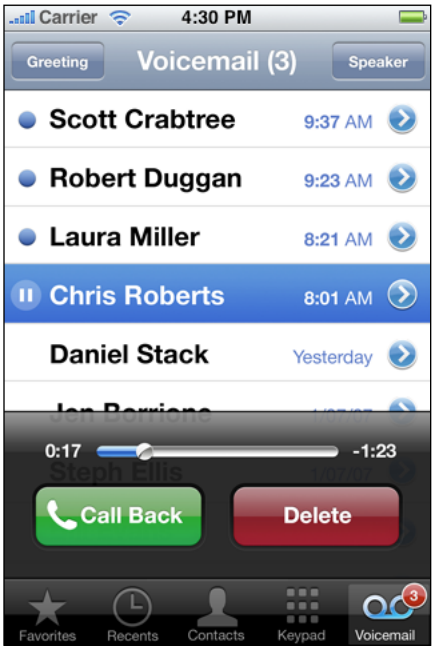
iPodでは、3つの場所（Tab Bar、「その他(More)」画面、「配置変更(Configure)」画面）すべてで同じタブアイコンを使用している点に注目してください。これによってユーザは、各アイコンが、表示される場所に関係なく同じものを指していると確信できます。

Tab Bar内のタブへのバッジ表示

押しつけがましくない控えめな方法でユーザに何かを伝えるために「バッジ」をタブに表示できます。このタイプのフィードバックは、ユーザのタスクまたはコンテキストにとって最重要ではなく、知っていると役に立つ情報を伝えるのに適しています。このバッジは、まだ聞いていない留守中のメッセージの数を「電話(Phone)」が示すために「留守番電話(Voicemail)」タブに表示するものに似ています。これは、タブの右上角に表示される赤い楕円です。楕円内の白い文字は情報を提供します。

バッジと特定のタブを結びつけることで、アプリケーションの特定のモードとバッジの情報を結びつけることができます。これは、現在そのモードになっていない場合でも可能です。図 6-15に、タブ上のバッジの例を示します。

図 6-15 バッジはTab Barで情報を伝える



アプリケーションがApple Push Notificationサービスに登録されており、ユーザがバッジの表示を許可している場合は、バッジをホーム(Home)画面のアプリケーションアイコンに表示することもできます。この仕組みの詳細については、「Local NotificationおよびPush Notificationの有効化」（54 ページ）を参照してください。

Alert、Action Sheet、およびモーダルビュー

Alert、Action Sheet、およびモーダルビューは、ユーザに注意を呼びかける場合や、アプリケーションが追加の選択肢や機能をユーザに提示する必要がある場合に表示するビューのタイプです。図7-1に、これらのビューのタイプの例を示します。

図 7-1 Action Sheet、モーダルビュー、およびAlert

これらのビューのタイプをプログラムで実装する詳細については、『*View Controller Programming Guide for iOS*』の「Modal View Controllers」を参照してください。

Local NotificationとPush Notification警告は標準の警告と非常によく似ているように見えますが、プログラムとしては同じではありません。通知警告の使用と設計を行う方法については、「Local Notificationおよび Push Notificationの有効化」（54 ページ）を参照してください。

使用方法と動作

Alert、Action Sheet、およびモーダルビューはすべてモーダルです。つまり、ユーザがボタンをタップして明示的にこれらを閉じてからはじめて、アプリケーションを引き続き使用できます。ユーザに危険な可能性のあるアクションを警告したり、または追加の選択肢を提供したりする必要があるときはありますが、これらのビューの乱用を避けることが肝要です。この理由は次のとおりです。

- すべてのタイプのモーダルビューはユーザのワークフローを妨げる。
- 確認または承認を求めるビューの表示頻度が高すぎると役に立つというよりも煩わしくなる。

特にAlert（警告）はあまり使用するべきではありません。警告の表示頻度が高すぎると、ユーザは警告を消すことだけを考えて、内容を読まずに閉じるようになります。

Alert、 Action Sheet、 およびモーダルビューは、さまざまなことを伝えるために設計されています。

- **Alert（警告）は、アプリケーション（またはデバイス）の使用に影響する重要な情報をユーザに提供する。**警告は、通常、予期せず受信します。これは、一般的に警告が、ユーザのアクションを必要とする可能性がある問題の発生または現在の状況の変化をユーザに伝えるからです。
- **Action Sheetは、ユーザの現在のアクションに関連する追加の選択肢をユーザに提示する。**ユーザは、害を与える可能性のあるアクション（最近の着信履歴をすべて削除するなど）や、異なる方法で完了できるアクション（ユーザが複数の宛先から1つを指定できる送信アクションなど）を開始するツールバーボタンをタップするとき、ユーザはAction Sheetの表示を期待するようになります。
- **モーダルビューは、現在のタスクのコンテキストにおいてより詳細な機能を提供する。**モーダルビューは、ユーザのワークフローに直接関連するサブタスクを実行する手段を提供することもできます。

これらのタイプのビューは外観も動作も異なり、これらが伝えるメッセージの違いを強調します。ユーザはこれらのビューの外観および動作に慣れているため、アプリケーションにおいてこれらの一貫性を保って正しく使用することが重要です。

Alertの使用

警告(Alert)は、アプリケーション画面の中央にポップアップしそのビューの手前に浮かんで、ユーザに重要な情報を非常に目立つ方法で提示します。警告の外観が独立しているように見えることで、警告の表示がアプリケーションまたはデバイスの何らかの変化によるものであり、必ずしもユーザが最近行ったアクションの結果ではないという事実が強調されます。警告は状況を説明するテキストを表示し、理想的にはユーザに適切なアクションを選ぶ手段を提供するべきです。

ユーザは、デバイスや、バックグラウンドで実行する組み込みのアプリケーション（「メッセージ(Messages)」など）からの警告は見慣れていますが、アプリケーションでこれらを使用する必要はほとんどありません。たとえば、ユーザが開始したタスクがブロックされたことをユーザに通知するために警告を使用する場合があります。問題が何であるのかをユーザに伝え、問題を処理する手段の選択肢をユーザに提示することが重要なので、このメッセージの警告を表示するのは理に適います。

また警告を使用して、危険な可能性のある結果を受け入れまたは拒否する機会をユーザに提供することもできます。この場合、警告に2つのボタンを表示するようにします。1つは警告を閉じてアクションを実行するボタン、もう1つはアクションを実行せずに警告を閉じるボタンです。多くの場合、アクションを実行せずに警告を閉じるボタンには、「キャンセル(Cancel)」というラベルを使用するのが理にかなっています。このような警告が表示されているときにユーザがホーム(Home)ボタンを押すと、結果は、アプリケーションの終了に加え、「キャンセル(Cancel)」ボタンをタップしたときと同じであるべきです。つまり、警告が閉じられ、アクションは実行されません。

警告の表示頻度が低ければ、ユーザがこれらの警告を深刻に受け取る手助けになります。アプリケーションが表示する警告の数を最小限にし、各警告が重要な情報および役に立つ選択肢を提供するようにします。一般的に次のような警告の作成は避けるようにしてください。

- ユーザに正常に進行しているタスクの最新情報を与える。

代わりに、Progress Viewまたはアクティビティインジケータを使用して、進捗に関連するフィードバックをユーザに提供することを検討します（これらのコントロールについては、「Progress View」（127 ページ）および「アクティビティインジケータ」（119 ページ）で説明します）。

- ユーザが開始したアクションの確認を求めること。

  ユーザが開始したアクションの確認を求めるには、それが連絡先の削除などのリスクを伴う可能性のあるアクションでも、Action Sheetを使用すべきです（次の「Action Sheetの使用」（91 ページ）で説明します）。

- ユーザが何もできないエラーまたは問題についてユーザに通知する。

  ユーザが修正できない重大な問題についてユーザに通知するために警告を使用する必要がある場合もありますが、可能であれば、そのような情報はユーザインターフェイスに統合するほうが良いでしょう。たとえば、サーバ接続が失敗するたびにユーザに通知するのではなく、最後に接続が成功した時間を表示します。

Action Sheetの使用

Action Sheetは、ユーザがアプリケーションのツールバーのボタンをタップして開始したタスクに関連付けられている選択肢の集合を表示します。Action Sheetは次のことに適した方法です。

- タスクを完了できる方法の選択肢を提示する。たとえば、「写真(Photos)」ではユーザは個々の写真を表示しているときに「送信(Send)」ボタンをタップできます。Action Sheetが表示され、写真に送信先として3つの選択肢が（送信をキャンセルする「キャンセル(Cancel)」ボタンのほかに）ユーザに提示されます。

  ユーザインターフェイスの特定の場所に選択肢を置いておかなくても、Action Sheetを使用すれば現在のタスクのコンテキストに合った一連の選択肢を提示できるため、このような状況ではAction Sheetを表示するのが便利です。

- 危険な可能性のあるタスクを完了する前に、確認を求める。たとえば、「メール(Mail)」の設定にもよりますが、ユーザが「メール(Mail)」のToolbarの「ごみ箱(Trash)」ボタンをタップするとAction Sheetが表示され、削除またはキャンセルを開始できます。

  このような状況でAction Sheetを表示すると、ユーザが行おうとしている手順の危険な影響をユーザに確実に認識させ、選択肢をいくつか提示できます。ユーザは意図せずにコントロールをタップしてしまうこともあるため、このタイプのコミュニケーションはiPhone OSベースのデバイスでは特に重要です。

Action Sheetは常にアプリケーション画面の一番下から現れ、そのビューの手前に浮いています（図 7-1（89 ページ）の左側を参照）。ただし、警告とは異なり、Action Sheetの側端は画面の側面にアンカーされており、アプリケーションおよびユーザの最後のアクションへのAction Sheetのつながりを強めています。

Action Sheetには、タスクの完了方法をユーザが選ぶことができるボタンがいくつか含まれます。ボタンラベルは、実行されているタスクとともに、ユーザが選択を理解するために必要となる文脈を十分に提供するはずなので、Action Sheetにはメッセージを追加する必要はないはずです。ユーザがボタンをタップすると、Action Sheetは消えます。Action Sheetでは、ユーザにアクションの選択肢を提示する必要があるため、Action Sheetは必ず複数のボタンを備えています。

モーダルビューの使用

デフォルトで、モーダルビューは画面の下端から上に向かってスライドし、常にアプリケーション画面全体を覆います（図 7-1 （89 ページ）の中央を参照）。モーダルビューは現在のアプリケーション画面を隠すため、何かを達成できる別の一時的なモードに入るというユーザの認識を強めます。

モーダルビューは適切であればテキストを表示でき、タスクを実行するために必要なコントロールを含みます。さらに、一般的にモーダルビューは、タスクを完了してビューを閉じるボタン、およびユーザがタップしてタスクを中止できる「キャンセル(Cancel)」ボタンを表示します。

モーダルビューは、Action Sheetよりもさらに高度にユーザとの対話をサポートします。選択を1つだけ受け入れるAction Sheetと違い、モーダルビューは、複数の選択肢の選択や情報の入力など、多段階のユーザとの対話をサポートします。

アプリケーションの主たる機能が関連する自己完結型のタスクを達成できるようにする必要があるときに、モーダルビューを使用します。モーダルビューは、アプリケーションの主たるユーザインターフェイスに常には属さないユーザインターフェイス要素を必要とする多段階のサブタスクに特に適しています。モーダルビューの良い例は「メール(Mail)」の作成ビューです。ユーザが「作成(Compose)」ボタンをタップすると、アドレスおよびメッセージのためのテキスト領域、入力のためのキーボード、「キャンセル(Cancel)」ボタンおよび「送信(Send)」ボタンを含むモーダルビューが表示されます。

警告の設計

警告に含めるテキスト、ボタンの数、およびボタンの内容は指定可能です。警告ビュー自体の幅または背景の外観、あるいはテキストの配置（中央揃え）をカスタマイズすることはできません。

Local NotificationおよびPush Notificationの警告の内容を設計する方法については、「Local Notificationおよび Push Notificationの有効化」 （54 ページ）を参照してください。

注： これらのガイドラインを読む際には、次の定義に注意してください。

- **タイトル形式の大文字化**とは、4文字以下の冠詞、等位接続詞、および前置詞を除くすべての単語の1文字目を大文字表記にすることを意味します（英語の場合）。
- **文章形式の大文字化**とは、最初の文字を大文字表記にして、残りの文字は小文字にすることを意味します（固有名詞や固有形容詞の場合は除く）。

警告タイトル（および任意指定のメッセージ）は、状況を簡潔に説明し、それについてユーザができることを説明するものである必要があります。理想的には、用意したテキストは、警告が表示された理由を理解し、どのボタンをタップするべきか決定するために必要十分なコンテキストをユーザに与えます。

必要な警告タイトルの作成に際して

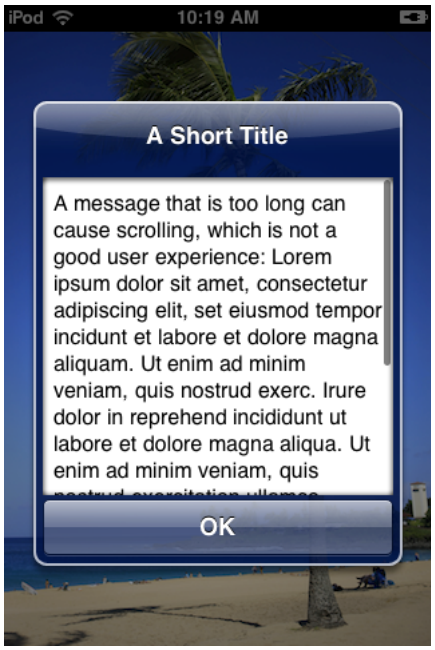
- できれば、タイトルは1行で十分表示できる短さにする。長い警告タイトルは、ユーザがすばやく読むのが難しく、文が欠けたり、警告メッセージをスクロールすることが強制されたりします。

- 「エラー」、「警告」など、有用な情報を提供しない一語のタイトルは避ける。
- 可能であれば、文章の断片を使用する。短く情報を伝える文は、完全な文章より理解しやすいことがよくあります。
- ネガティブになるのをためらわない。ユーザは、ほとんどの警告の通知が何らかの問題を知らせたり、危険な状況を警告したりするものであることを理解しています。ネガティブで直接的であるほうが、ポジティブであいまいであるよりも良いでしょう。
- できる限り、「あなた」、「あなたの」、「私」、「私の」の使用は避ける。場合によっては、ユーザを直接特定するテキストは、曖昧になる可能性や、侮辱と解釈される可能性さえあります。
- 次のような場合には、タイトル形式の大文字化を使用し、句読点は使用しない。
  - タイトルが文章の断片である
  - タイトルが質問形式ではない1つの文章で構成されている
- タイトルが質問形式の1つの文章で構成される場合、文章形式の大文字化と疑問符を使用する。一般に、警告タイトルを質問形式にすることでメッセージの追加を避けられるのであれば、そうすることを検討します。
- タイトルが2つ以上の文章で構成される場合、文章形式の大文字化と、適切であれば句読点を使用する。メッセージの追加を避けられるのであれば使用を検討する可能性はありますが、2つの文章から成る警告タイトルが必要となることはほとんどありません。

任意指定の警告メッセージを提供する場合

- 文章形式の大文字化と、適切な句読点を使用する短い完全な文章を作成する。

- 長すぎるメッセージは作成しない。可能であれば、1行または2行で十分表示できるようにメッセージを短くします。メッセージが長すぎるとスクロールすることになり、良いユーザ体験とはなりません。



タップすべきボタンの説明で警告テキストが長くなってしまうことは避ける（「情報を見るにはビューをタップ(Tap View to see the information)」など）。理想的には、明確な警告テキストと論理的なボタンラベルの組み合わせであれば、状況と選択を理解するために十分な情報をユーザに与えます。ただし、詳細なガイダンスを提供する必要がある場合は、つぎのガイドラインに従ってください。

- 選択アクションを説明する場合は「タップ(Tap)」という単語を使用する（「タッチ(Touch)」、「クリック(Click)」、「選ぶ(Choose)」は使用しない）。
- ボタンタイトルを引用符で囲まない。ただし、大文字化は維持する。

両方の向きで警告の外観をテストする。横向きでは警告の高さが制限されるため、縦向きでの警告とは異なって見える可能性があります。両方の向きで見栄えがよくなるように（スクロールも回避）、警告テキストの長さを最適化することをお勧めします。

2つのボタンの警告を設定する。ユーザは2つの選択肢から選ぶのが最も簡単であるため、多くの場合、2つのボタンの警告が最も有用です。1つのボタンを持つ警告を表示するのは、ほとんどの場合、良い考えとは言えません。そのような警告は単なる情報提供にすぎず、そのときの状況を左右する力をユーザに対してなんら与えるものではないからです。3つ以上のボタンを含む警告は2つのボタンの警告よりかなり複雑であり、可能であれば避けるべきでしょう。実際、ユーザに2つを超える選択肢を提示する必要がある場合には、代わりにAction Sheetの使用を検討するべきです（このタイプのビューの詳細については、「Action Sheetの使用」（91 ページ）および「Action Sheetの設計」（96 ページ）を参照してください）。

警告ボタンに適切な色を使用する。警告ボタンは、暗い色または明るい色のどちらかの色になっています。2つのボタンを持つ警告では、左側のボタンは常に暗い色にし、右側のボタンは常に明るい色にします。1つのボタンの警告では、ボタンは常に明るい色にします。

- リスクを伴う可能性のあるアクションを提示する2つのボタンの警告では、アクションをキャンセルするボタンを右側に置きます（明るい色にします）。
- ユーザが望むであろう、害のないアクションを提示する2つのボタンの警告では、アクションをキャンセルするボタンは左側にします（暗い色になります）。

注： 「キャンセル(Cancel)」ボタンを明るい色にするか暗い色にするか、また右側に置くか左側に置くかは、他方の選択肢が害を及ぼすかどうかによります。コード内でどのボタンが「キャンセル(Cancel)」ボタンかを正しく識別してください（詳細については、『*UIAlertView Class Reference*』を参照してください）。

警告ボタンには、短く論理的なタイトルを付ける。最善のタイトルは、警告テキストのコンテキストに合った、1つまたは2つの単語で構成されています。警告ボタンのタイトルを作成する際には、次のガイドラインに従います。

- あらゆるボタンのタイトルと同様に、タイトル形式の大文字化を使用し、句読点は使用しない。
- 「キャンセル(Cancel)」、「許可(Allow)」、「返信(Reply)」、「無視(Ignore)」など、警告テキストに直接関連する動詞を使用する。
- 単純な受け入れを示す選択肢として、ほかに適切な候補がなければ「OK」を使用する。「はい(Yes)」や「いいえ(No)」の使用は避ける。
- できる限り、「あなた」、「あなたの」、「私」、「私の」の使用は避ける。これらの単語を使用するボタンタイトルは、曖昧で横柄な印象を与えます。

Action Sheetの設計

アプリケーションの外観に合わせてAction Sheetの背景を選び、ボタンの数および内容を指定できます。

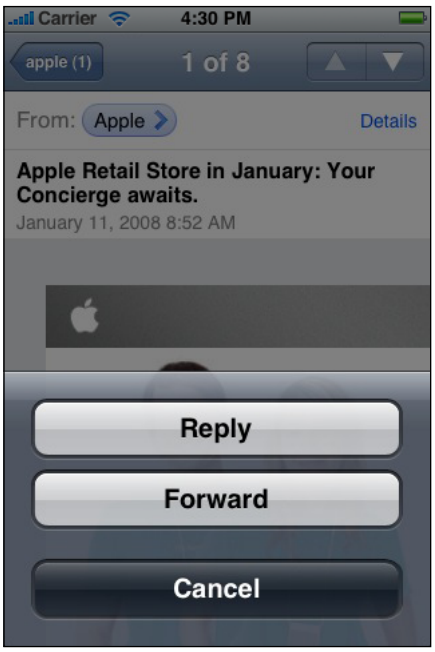
警告とは異なり、Action Sheetはテキストメッセージを表示する必要がありません。これは、「削除(Delete)」ボタンまたは「送信(Send)」ボタンをタップするなどのユーザアクションの結果としてAction Sheetが表示されるからです。このため、Action Sheetの表示理由を説明する必要はありません。

Action Sheetには、2つの異なる背景の外観があります。アプリケーションのAction Sheetの背景がアプリケーションのToolbarまたはNavigation Barの外観と調和するようにする必要があります。たとえば、アプリケーションが黒のNabigation BarおよびToolbarを使用する場合には、Action Sheetの背景は半透明の黒にすると良いでしょう。デフォルトで、iPhone OSは標準の青色の背景色のAction Sheetを表示します。これは標準の青色のToolbarとNabigation Barと調和します。アプリケーションのすべてのAction Sheetには同じ背景色を使用し、その色がNabigation BarおよびToolbarの色と調和するようにするべきです。

「キャンセル(Cancel)」ボタンは、必ずAction Sheetの一番下に表示するようにします。こうすることで、ユーザが「キャンセル(Cancel)」の選択に達する前にすべての選択肢を読み通すように促します。

図 7-2に、デフォルトの背景の外観を使用し、「キャンセル(Cancel)」ボタンが推奨の位置にあるAction Sheetを示します。

図 7-2 典型的なAction Sheet

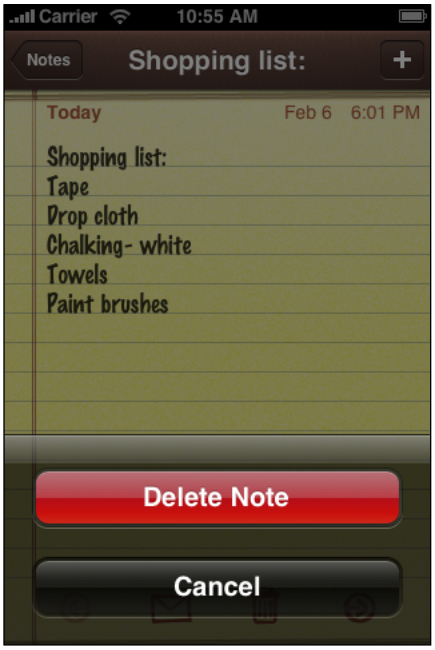


ユーザのショッピングリストのすべての項目を削除するなどの、害のある可能性のあるアクションを実行するボタンを提供する必要がある場合には、赤色のボタンを使用するべきです。害を及ぼす可能性のあるこのようなボタンは、Action Sheetの一番上に表示することが重要です。理由は2つあります。

- ボタンがAction Sheetの一番上に近ければ近いほど、ボタンがより目立つ。
- ユーザは、ホーム(Home)ボタンを押そうとして、誤ってデバイス画面の最下部をタップすることがある。Action Sheetの一番下から離れた場所に害を及ぼす可能性のあるボタンを配置することで、ユーザがホーム(Home)ボタンの代わりに誤って画面をタップしても、望ましくない結果を引き起こす可能性が低くなります。

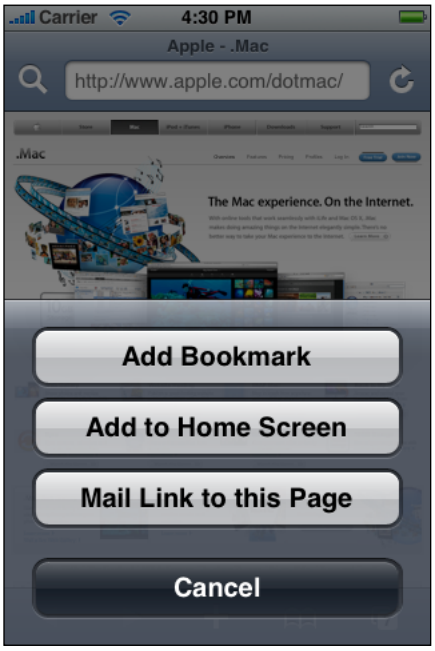
図 7-3に、半透明な黒の背景の外観を使用し、「キャンセル(Cancel)」ボタンおよび害を及ぼす可能性のあるボタンが推奨の位置にあるAction Sheetを示します。

図 7-3 害を及ぼす可能性のあるアクションを実行するボタンの色は赤色にして、Action Sheetの一番上に配置する



Action Sheetには複数のボタンを表示できます。ただし、それぞれのボタンをほかのボタンと簡単に区別できる必要があります。図 7-4に、標準の青色のToolbarに合う背景を使用し、「キャンセル(Cancel)」のほかに3つの選択肢を提示するAction Sheetを示します。

図 7-4 4つのボタンを持つAction Sheet



モーダルビューの設計

モーダルビューの全体的な外観は、モーダルビューを表示するアプリケーションと調和させるべきです。たとえば、多くの場合、モーダルビューには、タイトルとモーダルビューのタスクをキャンセルまたは完了するためのボタンを持つNavigation Barが含まれます。Navigation Barの背景の外観は、アプリケーションのNavigation Barと同じ背景の外観にします。

モーダルビューは、何らかの方法でタスクを識別するタイトルを常に表示するべきです。適切であれば、タスクをもっと十分に説明したりガイダンスを提供したりするテキストを、ビューのほかの領域に表示することもできます。たとえば、「メッセージ(Messages)」アプリケーションでは、ユーザがテキストメッセージを作成したいときには、モーダルビューが提示されます。図7-5に示すモーダルビューには、アプリケーションのNavigation Barと同じ背景で、「新規メッセージ(New Message)」というタイトルを持つNavigation Barが表示されています。

図 7-5 モーダルビューはアプリケーション画面と調和させる

モーダルビューでは、タスクを達成するために必要な任意のコントロールを使用できます。たとえば、テキストフィールド、ボタン、およびTable Viewを含めることができます。

モーダルビューはアプリケーションと調和させて、コンテキストが一時的にビューの表示をシフトさせているとユーザに思わせるような方法で表示できます。これを行うために、次のトランジションスタイルの1つを指定できます。

- **垂直。**モーダルビューは画面の下端から上に向かってスライドし、閉じると下端に向かって戻ります（これはデフォルトのトランジションスタイルです）。
- **フリップ。**現在のビューが水平にフリップして、右側から左側へモーダルビューを表示します。視覚的に、モーダルビューはまるで現在のビューの背景であるかのように見えます。モーダルビューを閉じると、モーダルビューは左側から右側に水平にフリップして、前のビューを表示します。

アプリケーションのモーダルビューのトランジションスタイルを変更する場合、変化を与えるためだけに行うのは避けてください。ユーザはそのような違いに気付き、その違いに何か意味があるものと考えてしまうでしょう。このため、ユーザが簡単に気付いて覚えておくことができる、論理的で一貫性のあるパターンを構築し、不当にトランジションスタイルを変更するのは避けるのが最良です。

Table View、 Text View、 およびWeb View

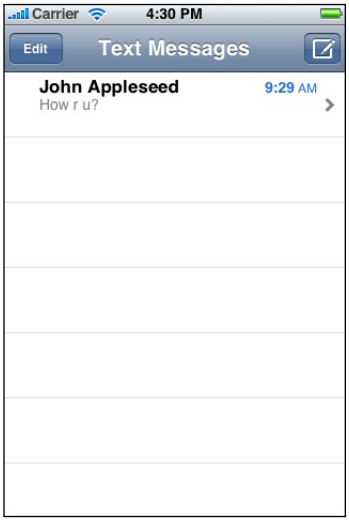
Table View、Text View、およびWeb Viewは、iPhoneアプリケーションにおけるさまざまな用途に役立つ多目的の要素です。たとえば、Table Viewは、簡単な選択肢のリスト、詳細情報がグループ化されたリスト、インデックス付きの長い項目リストを表示するように構成できます。Table ViewおよびWeb Viewは、コンテンツを受け取ったり表示したりするのに使用できる比較的制約の少ないコンテナです。

Table View

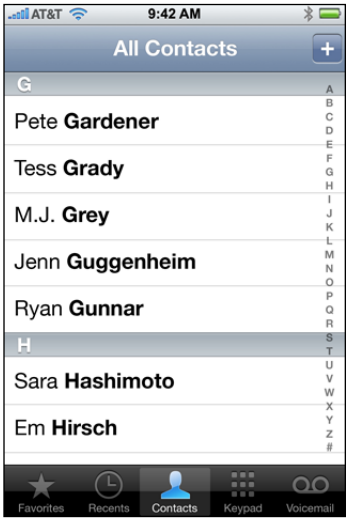
Table Viewは複数の行からなる単一カラムのリストにデータを表示します。行は、セクションまたはグループに分けることもでき、各行は、テキスト、画像、およびコントロールの組み合せを含むことができます。行数または行のグループの場合、ユーザはフリック（はじく）またはドラッグして、情報をスクロールします。図 8-1に、異なるスタイルのTable Viewでリストを表示するさまざまな方法を示します。

図 8-1 Table Viewを使用してリストを表示する3つの方法

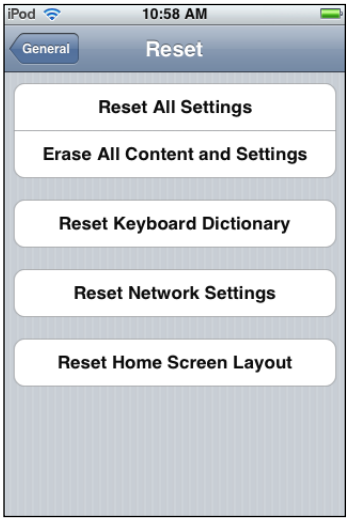
A simple list in a plain table view



An indexed list in a plain table view



A grouped list in a grouped table view



使用方法と動作

Table Viewは、iPhoneアプリケーションでは非常に役に立ちます。これは、大量の情報および少量の情報の両方を編成するための魅力的な方法を提供するからです。Table Viewは、多数のユーザ項目を処理する傾向にある生産性型アプリケーションで最も役に立ちますが、ユーティリティ型アプリ

ケーションでも小規模なTable Viewを使用することができます。没入型アプリケーションでは、おそらく情報の表示にTable Viewを使用することはないでしょう。ただし、選択肢の短いリストを表示するためにTable Viewを使用することは考えられます。

Table Viewは、ユーザが情報の中を移動したり情報を操作したりすることを可能にする組み込みの要素を提供します。さらに、Table Viewは次のことをサポートします。

- ヘッダおよびフッタ情報の表示。リストの各セクションまたはグループの上下、およびリスト全体の上下に説明テキストを表示できます。
- リストの編集。ユーザが一貫性のある方法でリスト項目の追加、削除、順序の並べ替えを行えるようにできます。Table Viewは、複数のリスト項目の選択および操作もサポートします。これを利用して、一度に複数の項目を削除する便利な手段をユーザに提供できます。

テーブルでは、ユーザがリスト項目を選択したとき、必ずフィードバックを返すようにします。項目を選択できる場合、ユーザが項目を選択すると、その項目を含む行が一時的にハイライトされ、その選択が受け付けられたことを示すフィードバックを返します。その後、すぐにアクションが発生します。つまり、新しいビューが表示されるか、または項目が選択されたり有効にされたりしたことを示すチェックマークが行に表示されます。

まれなケースですが、行項目に関連付けられた二次的な詳細またはコントロールが同じ画面に表示されていると、行がハイライトされたままになる場合があります。しかし、選択肢リスト、選択された項目、および関連付けられた詳細またはコントロールを表示しながら、居心地の悪い込み合ったレイアウトにならないようにするのは困難なため、これはお勧めできません。

行を選択した結果、新しい画面が表示される場合、選択された行は一時的にハイライトされ新しい画面が所定の位置にスライドします。ユーザが前の画面に戻るときには、もともと選択された行が再度一時的にハイライトされ、ユーザに直前の選択を思い出させます。

ユーザがリスト項目に加える変更は、アニメーション化することもできます。アニメーション化は、フィードバックを提供し、直接操作をしているというユーザの感覚を強化する良い方法です。たとえば、「設定(Settings)」で日付と時刻の自動設定をオフにすると（「日付と時刻(Date & Time)」＞「自動設定(Set Automatically)」で「オフ(Off)」を選択）、リストグループがスムーズに展開し、「時間帯(Time Zone)」および「日付と時刻を設定(Set Date & Time)」の2つの項目を表示します。

テーブルでは、内容を即座に表示するようにします。テーブルの内容が膨大または複雑な場合、何か表示する前に、すべてのデータが利用可能になるまで待機することは避けてください。代わりに、テキストデータを画面上の行に即座に表示し、より複雑なデータ（画像など）は、利用できるようになるのに伴って表示します。この手法を使えば、ユーザは有用な情報をすぐに得られるので、アプリケーションの応答性が高いと感じます。

アプリケーションが表示するデータの変更頻度が低い場合は、「古い」データを表示しておいて、新しいデータが利用可能になるのを待機することもできます。この手法を使えば、ユーザは有用なものをすぐに見ることができますが、変更が頻繁にあるデータを扱うアプリケーションにはお勧めできません。この手法を使うことを決める前に、どのくらいの頻度でデータの変更があり、ユーザが新しいデータをすぐに見ることをどの程度期待しているかを測定します。

有用なものをすぐに表示するのが難しい場合、空行の表示を避けることが重要です。アプリケーションが停止していると認識される可能性があるためです。代わりに、テーブルは、回転するアクティビティインジケータと一緒に、画面の中央に「読み込み中...(Loading...)」などの情報ラベルを表示するべきです。古いデータの表示が可能な場合、空の行について心配することはありませんが、画面上の行をできるだけ早く更新するべきです。どちらの手法も、ユーザにフィードバックを返すことで処理が継続中であることを示します。

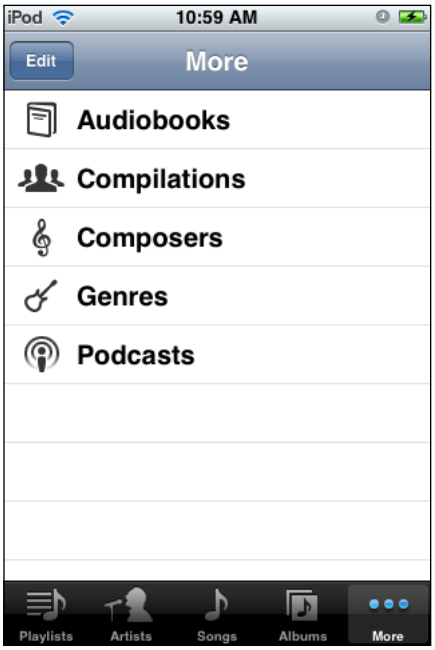
Table Viewスタイル

iPhone OSでは、主にその外観で区別される、Table Viewの2つのスタイルが定義されています。

プレーン(`UITableViewStylePlain`)。このTable Viewスタイルは、画面の側端から側端まで広がる行を表示します。行の背景は白色です。行はラベル付きセクションに分けることが可能で、Table Viewはビューの右端に沿って垂直に表示されるインデックスを任意で表示できます。

図8-2に、iPodアプリケーションのプレーンテーブル（ヘッダ、フッタ、インデックスがない）のリストを示します。

図 8-2 プレーンテーブルの単純なリスト



グループ化(`UITableViewStyleGrouped`)。このTable Viewスタイルは、画面の側端から内側に少し下がった位置に行のグループを表示します。これらのグループは独特の垂直のしま模様の背景に表示されます。また、グループの内側の背景は白色です。グループ化されたTable Viewは任意の数のグループを含むことができ、各グループは任意の数の行を含むことができます。各グループの先頭にはヘッダテキストを付けることができ、フッタテキストを後に付けることができます。このTable Viewのスタイルは、インデックスを提供しません。

図8-3に、各グループに1つの行が含まれる、グループ化されたテーブルのリストを示します。このリストは、「設定(Settings)」アプリケーションにあり、ヘッダまたはフッタのテキストは含まれません。

図 8-3 グループ化されたテーブルの4つのグループのリスト

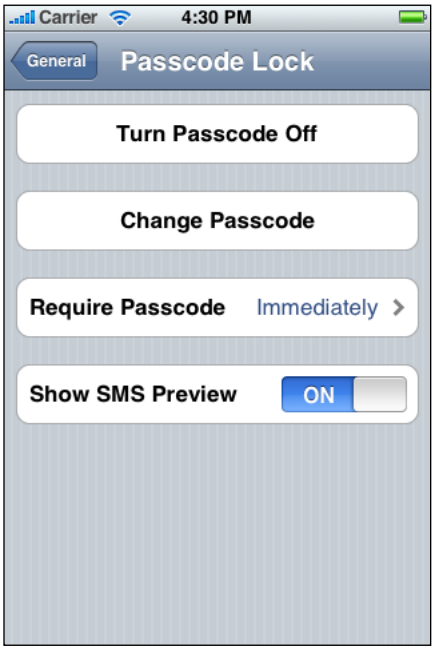


Table Cellスタイル

iPhone OS 3.0以降には、4つの定義済みのTable Cellスタイルが含まれており、プレーンテーブルとグループ化テーブルの両方のテーブル行に対して、最も一般的なレイアウトをすばやく簡単に作成できます。プログラムでは、これらのスタイルは、行の描画方法をテーブルに通知するオブジェクトである、Table Viewのセルに適用されます。

標準Table Cellスタイルを使用する場合、アプリケーションは組み込みアプリケーションとの一貫性を持ち、次のような利点が得られます。

- ユーザはアプリケーションがどのように機能するかをより早く理解できる
- 標準Table Cellスタイルが将来拡張された場合にも、アプリケーションは、多くの追加作業を必要とすることなく、一貫性を保つことができる

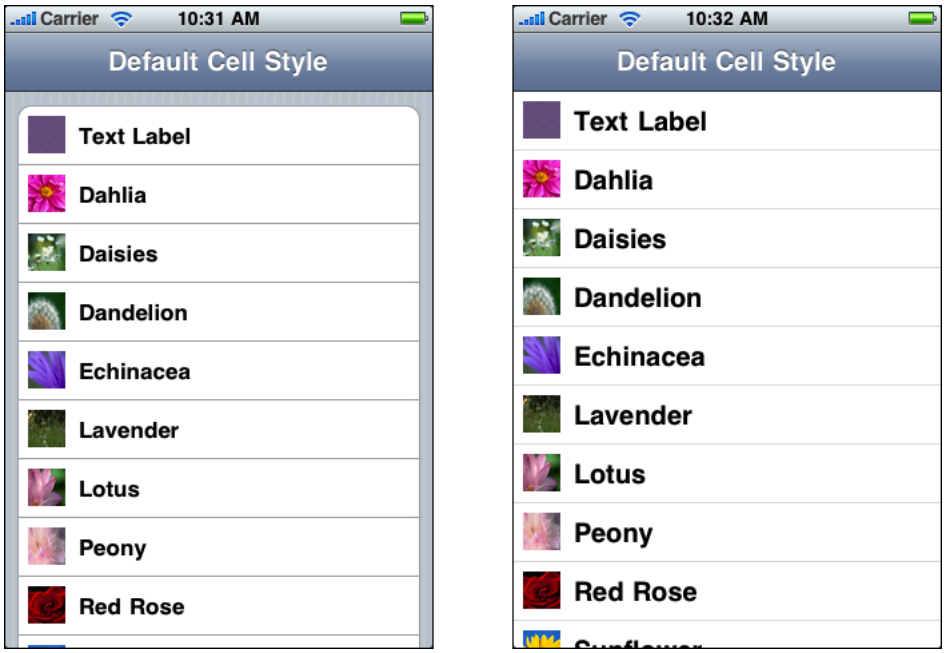
標準ではない方法でテーブルの行をレイアウトする場合、標準のものを大きく変更するより、カスタムのTable Cellスタイルを作成するほうが良いでしょう。独自のセルを作成する方法については、『*Table View Programming Guide for iOS*』の「Customizing Cells」を参照してください。

すべてのTable Cellスタイルでは、テキストの切り詰めが自動的に行われます。一般に、テキストは、ユーザが理解するのが難しくなるほどに、単語や文が切り詰めて表示されることを避けるため、できる限り簡潔にする必要があります。特に、どのセルスタイルが使用されたか、どこで切り詰めが発生したかにより、テキストの切り詰めは多かれ少なかれ問題になります。

iPhone OSでは、次の標準Table Cellスタイルが提供されています。

- **デフォルト**Table Cellスタイル(`UITableViewCellStyleDefault`)は、任意の画像が左に配置され、左揃えの黒のテキストラベルが続きます。

図 8-4　グループ化テーブル（左）とプレーンテーブル（右）のデフォルトTable Cellスタイル

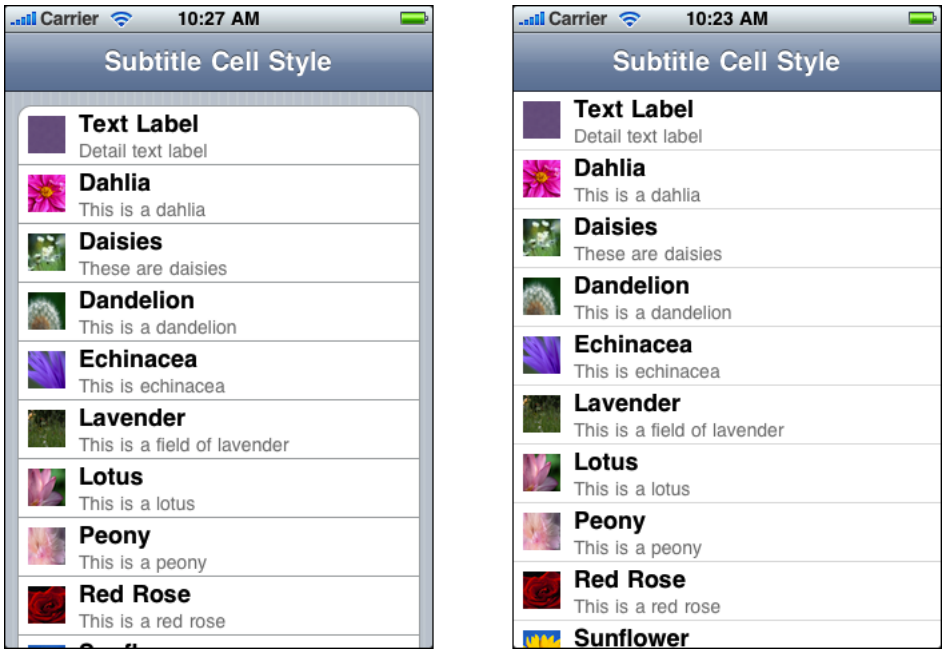


テキストラベルの外観は、項目名またはタイトルを表し、左揃えであることで簡単にリストに目を通すことができます。したがって、デフォルトスタイルは、区別のために補足情報を必要としないような項目のリストの表示に適しています。

短いテキストラベルが最適ですが、切り詰めがやむを得ない場合は、最初のいくつかの単語に最も重要な情報が含まれるようにしてください。

- **サブタイトル**Table Cellスタイル(`UITableViewCellStyleSubtitle`)は、任意の画像が左に配置され、1行の左揃えのテキストラベルと、その行の下に左揃えの詳細テキストラベルが続きます。テキストラベルは黒で、詳細テキストラベルはより小さい、グレイのフォントで表示されます。

図 8-5　グループ化テーブル（左）とプレーンテーブル（右）のサブタイトルTable Cellスタイル

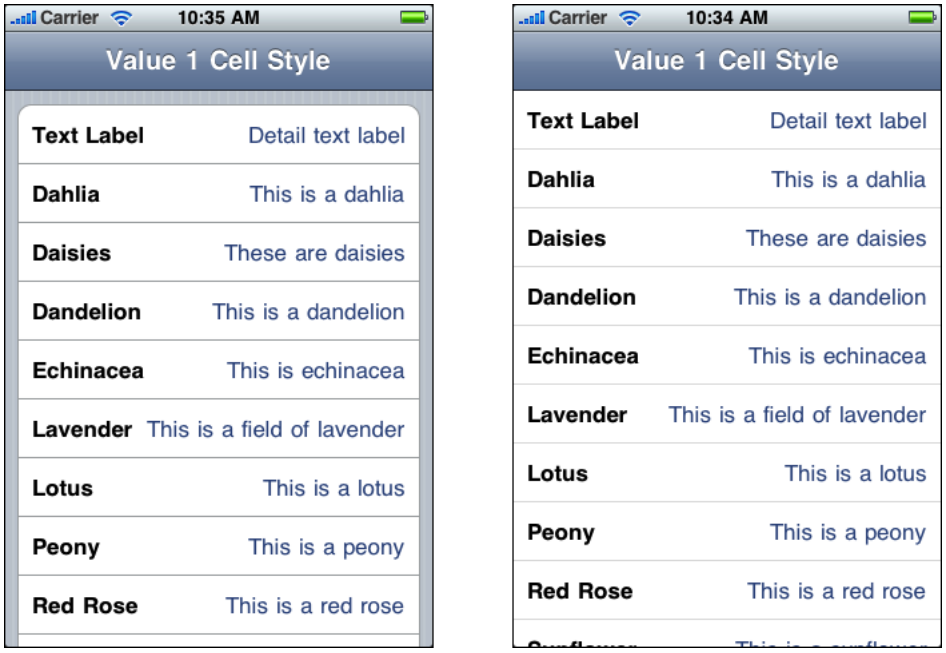


テキストラベルの目立つ外観は項目名またはタイトルを表していることを示し、詳細テキストラベルの淡い外観は項目に関連する補足情報が含まれていることを示します。テキストラベルが左揃えであることで簡単にリストに目を通すことができます。ユーザは、テキストラベルで名前が付けられている項目の区別に詳細テキストラベルの追加情報を利用できるので、このTable Cellスタイルは表示されるリスト項目が互いに似ている場合に有効です。

テキストラベルは、切り詰めを避けるために短くするべきです。切り詰めがやむを得ない場合は、最初のいくつかの単語に最も重要な情報を入れることに注力してください。詳細テキストラベルが切り詰められても、テキストラベルで名前が付けられた項目を拡張または補足する情報とみなしているため、ユーザはそれほど気にすることはありません。

- **value 1** Table Cellスタイル(`UITableViewCellStyleValue1`)は、左揃えの黒のテキストラベルが表示され、同じ行に右揃えの詳細テキストラベルがより小さい、青のフォントで表示されます。画像はこのスタイルでは適切ではありません。

図 8-6　グループ化テーブル（左）とプレーンテーブル（右）のvalue 1 Table Cellスタイル



テキストラベルの外観は項目名またはタイトルを表していることを示し、詳細テキストラベルの外観は項目に密接に関連する重要な情報が含まれていることを示します。

テキストラベルの左揃えとフォントは、ユーザがリストに目を通して必要な項目を探せるのに役立ち、詳細テキストラベルの右揃えの表示は、提供されている関連情報にユーザの注意を引きます。このTable Cellスタイルは、場合によってはサブリストから選択された、項目の現在の値を表示するのに有効です。

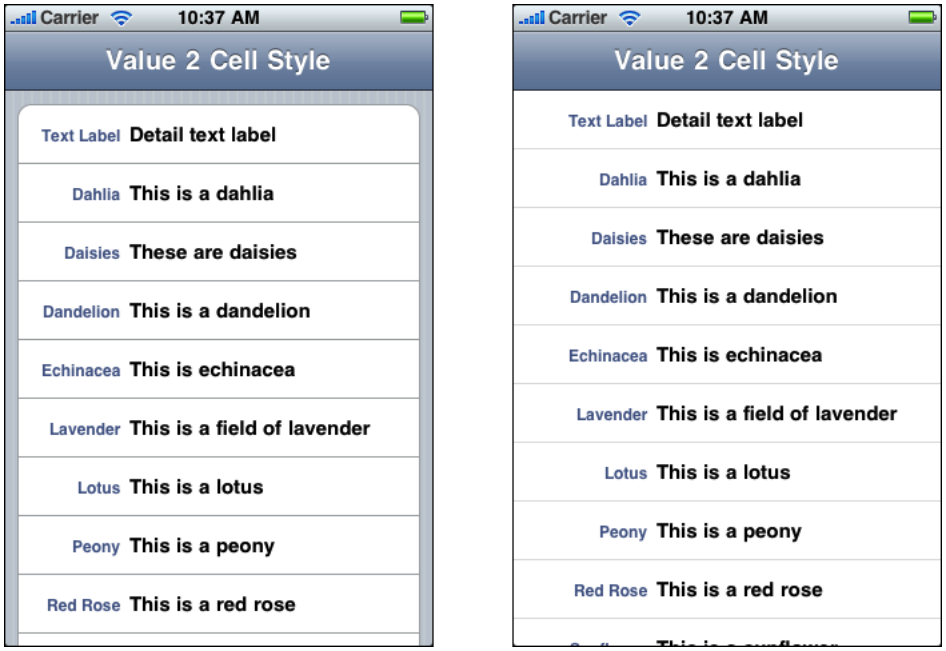
このレイアウトではテキストの切り詰めを避けるのは困難ですが（同じ行に2つのラベルがあるため）、努力する価値はあります。そうでなければ、2つの情報間の関係をユーザが理解するのに役立つ、ラベル間の有効なスペースを失います。

value 1 Table Cellスタイルはプレーンテーブルとグループ化テーブルのどちらでも使用できますが、その外観はグループ化テーブルのほうがより適しています。たとえば、「設定(Settings)」の「使用時間(Usage)」では、グループ化テーブルにvalue 1スタイルが使用されています。

図 8-7 value 1 Table Cellスタイルはグループ化テーブルで最適に表示

- **value 2** Table Cellスタイル(`UITableViewCellStyleValue2`)は、右揃えの小さな青のフォントでテキストラベルが表示され、同じ行に左揃えの詳細テキストラベルがより大きな、黒のフォントで表示されます。画像はこのスタイルでは適切ではありません。

図 8-8 グループ化テーブル（左）とプレーンテーブル（右）のvalue 2 Table Cellスタイル

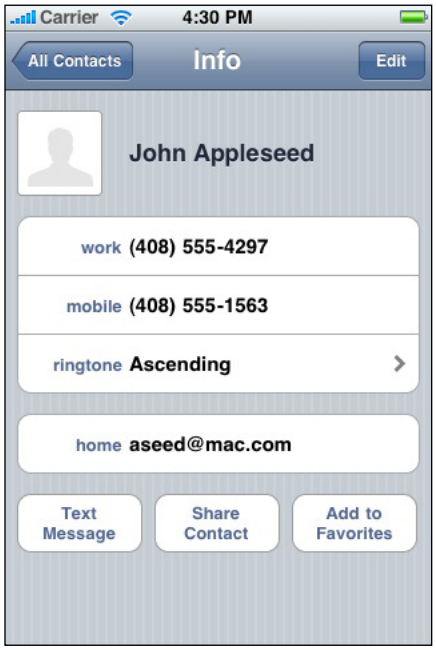


テキストラベルの右揃え、制限された幅、フォントは、より目立つ左揃えの詳細テキストラベルの重要情報に対する、見出しまたはキャプションとして機能していることを示します。

このレイアウトでは、ラベルは、各行の同じ位置で互いに向き合うように配置されます。結果として、リストのテキストラベルと詳細テキストラベルの間に明確な垂直の余白が生まれ、詳細テキストラベルの最初のいくつかの単語にユーザの目を引くのに役立ちます。テキストラベルの切り詰めを受け入れる場合、鮮明な垂直の空間が失われ、詳細テキストラベルの情報にユーザが目を通すのが難しくなります。

value 2 Table Cellスタイルはプレーンテーブルとグループ化テーブルのどちらでも使用できますが、その外観はグループ化テーブルのほうがより適しています。たとえば、「連絡先(Contacts)」の「情報(Info)」画面では、グループ化テーブルにvalue 2 Table Cellスタイルが使用されています。

図 8-9 value 2 Table Cellスタイルはグループ化テーブルで最適に表示



注： すべての標準Table Cellスタイルは、チェックマークやディスクロージャインジケータなど、Table View要素の追加も可能です。これらの要素を追加すると、タイトルとサブタイトルに利用可能なセルの幅が減少することに注意してください。

テーブル行の高さを高くしてテキストを折り返せるようにすることでテキストの切り詰めを避けることもできますが、これは問題が発生する可能性があります。

- プログラムでテキストの長さを確認し、折り返しが発生するかどうかを判断する。テーブルの幅はテキストの折り返しに影響するため、縦向きと横向きの両方でこの判断を行う必要があります。
- 一方の向きでは折り返しをし、他方の向きでは折り返しをしないということは避けるべきである。

- Table Viewスタイルにかかわらず行の高さが可変の場合、アプリケーション内のTable View全体のパフォーマンスが悪影響を受ける可能性がある。

最後に、行の高さが可変であることは、グループ化テーブルでは許容できますが、プレーンテーブルでは込み入った不ぞろいな外観になります。

Table Viewの要素

iPhone OSには、Table Viewの機能を拡張する、いくつかの**Table Viewの要素**が含まれています。特に明記しないかぎり、これらの要素の使用が適しているのはTable Viewのみです。ユーザは、組み込みアプリケーションで使われているこれらの要素の外観と動作に慣れているため、自分のアプリケーションでもこれらを正しく使用してください。

注： プログラムでは、Table Viewの要素は異なる方法で実装されています。いくつかは、Table Cellのアクセサリビューであり（テーブルに対し行の描画方法を指示するオブジェクト）、そのほかは、Table Viewが編集モードに入ったときに表示することができます。これらの要素を管理する別の方法については、『*Table View Programming Guide for iOS*』を参照してください。

- **ディスクロージャインジケータ**。この要素がある場合、ユーザは行内の任意の場所をタップして、階層の次のレベルまたはリスト項目に関連付けられている選択肢を表示できることを知っています。

  行のディスクロージャインジケータは、行を選択した結果として別のリストを表示させる場合に使用します。リスト項目に対応する詳細情報を表示するためにディスクロージャインジケータを使用することは避けます。代わりに、その目的のためには、詳細ディスクロージャボタンを使用します。

- **詳細ディスクロージャボタン**。ユーザはこの要素をタップして、リスト項目の詳細情報を表示します（この要素は、何かについての追加の詳細を表示するためにTable View以外のビューでも使用できます。詳細については、「詳細ディスクロージャボタン」（122 ページ）を参照してください）。

  Table Viewでは、行の詳細ディスクロージャボタンは、リスト項目の詳細を表示するために使用します。詳細ディスクロージャボタンはディスクロージャインジケータと異なり、行の選択から切り離されたアクションを実行できます。たとえば、「電話(Phone)」の「よく使う項目(Favorites)」では行をタップして、連絡先に電話をかけます。また行の詳細ディスクロージャボタンをタップして、連絡先の詳細情報を表示します。

- **「削除(Delete)」ボタン**。ユーザはこの要素をタップして、リスト項目を削除します。この要素は、ユーザが行でスワイプしたり編集中に削除コントロールボタンをタップしたりすると、リスト項目の右側に表示されます（この要素の例は、図 8-10を参照してください）。

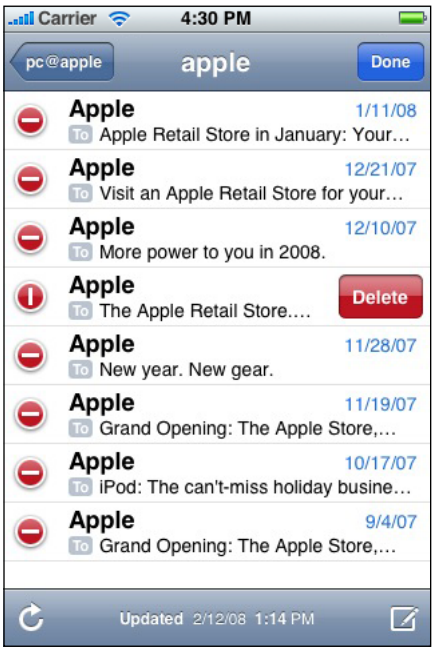
- **削除コントロールボタン**。ユーザはこの要素をタップして、各リスト項目の「削除(Delete)」ボタンを表示したり非表示にしたりします。ユーザが削除コントロールボタンをタップして「削除(Delete)」ボタンを表示すると、ユーザに追加のフィードバックを提供するためにボタンの水平のマイナス記号が垂直になります。この要素の例については、図 8-10を参照してください。

  一時的な編集モードをサポートするグループ化テーブルでは、削除コントロールはTable Viewの外側、向かって左に表示されます。たとえば、個々の連絡先情報を編集しているときに、これを目にできます。グループ化テーブルが常に編集モードである場合（「株価(Stocks)」および「天気(Weather)」の背面のグループ化テーブルなど）、削除コントロールはテーブルの内側、向かって左に表示されます。

図8-10に示すように、プレーンテーブルでは、削除コントロールは常にテーブルの内側、向かって左に表示されます。

- **行挿入ボタン**。ユーザはこの要素をタップしてリストに行を追加します。
- **行並べ替えコントロール**。この要素がある場合、ユーザはリスト内の別の場所へ行をドラッグできます。
- **チェックマーク**。この要素はリスト項目の横に表示され、その項目が現在選択されていることを示します。

図 8-10 Table Viewは「削除(Delete)」ボタンおよび削除コントロールボタンを表示する



Switchコントロール

Switchコントロールは、はい／いいえ、またはオン／オフなど、相互に排他的な2つの選択肢または状態をユーザに提示します。Switchコントロールは、2つの可能な選択肢のうち一方のみを一度に表示します。ユーザはコントロールをスライドさせることで表示されていない選択肢または状態を表示します。図 8-11 に、Switchコントロールの例を示します。

図 8-11 Table ViewのSwitchコントロール

グループ化されたTable Viewにおいて、ユーザに対し2つの単純な正反対の選択肢を提示するときにSwitchコントロールを使用します。選択肢の1つは常に表示されていないため、Switchコントロールは、ユーザが両方の値が何であるかをすでに知っている場合に使用するのが最善です。つまり、ユーザにもう一方の選択肢が何であるかを確認するためにだけSwitchコントールをスライドさせることがないようにします。

Switchコントロールを使用して、ビューのほかのユーザインターフェイス要素の状態を変更できます。ユーザの選択に応じて、新しいリスト項目が現れたり消えたり、リスト項目がアクティブになったり非アクティブになったりする可能性があります。

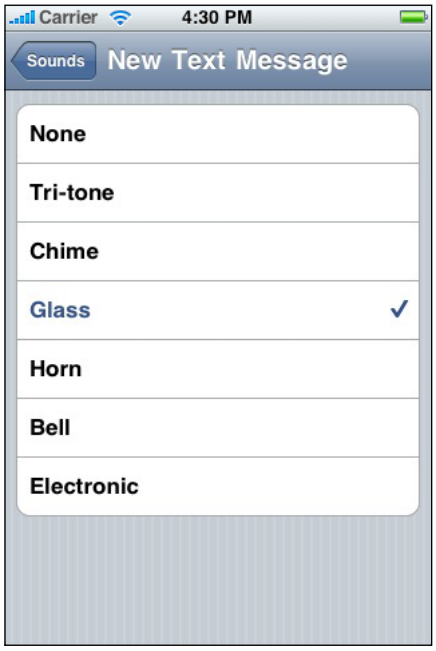
Table Viewを使用した共通ユーザアクションの有効化

TableViewは、特に多用途のユーザインターフェイス要素です。これは、次に示すように、さまざまなユーザアクションをサポートするためにさまざまな方法で構成できるからです。

- **選択肢の選択**

  iPhone OSは、メニューまたはポップアップメニューと同等の複数項目選択コントロールを備えていませんが、TableViewは、ユーザが選べる選択肢のリストを表示するのに効果があります。これは、Table Viewが簡素かつ整頓された方法で項目を表示するからです。さらに、Table Viewは、リスト内で現在選択されている選択肢（1つまたは複数）をユーザに示すチェックマーク画像を提供します。図 8-12 （113 ページ）を参照してください。

図 8-12 リストでの現在の選択を示すチェックマーク

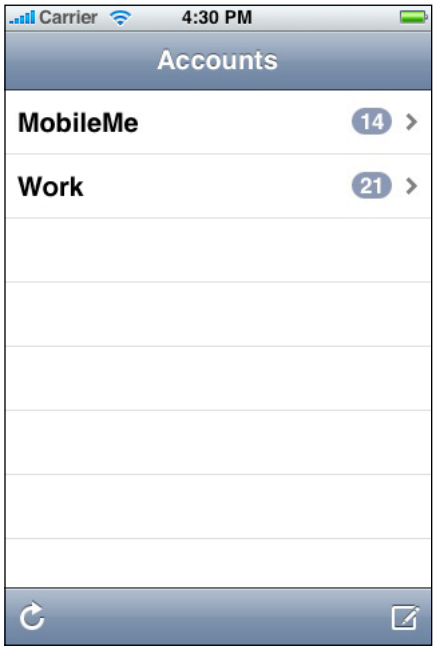


ユーザがテーブルの行の項目をタップしたときにユーザが目にする選択肢のリストを表示する必要がある場合、Table Viewのどちらのスタイルも使用できます。しかし、ユーザがテーブルの行にないボタンやほかのユーザインターフェイス要素をタップしたときに目にする選択肢のリストを表示する必要がある場合には、プレーンスタイルを使用します。

- **階層構造の情報内の移動**

  Table Viewは、各ノード（つまり、リスト項目）がそれぞれ独自の情報のサブセットを持つことのできる階層構造の情報の表示に適しています。各サブセットを別々のリストに表示できるからです。これにより、ユーザは連続する各リストで項目を1つ選択することで階層を簡単にたどれます。ディスクロージャインジケータ要素は、行内の任意の場所をタップすると、新しいリストに情報のサブセットが表示されることをユーザに伝えます。図 8-13 （114 ページ）を参照してください。

図 8-13 次の画面に情報のサブセットがあることを示すディスクロージャインジケータ

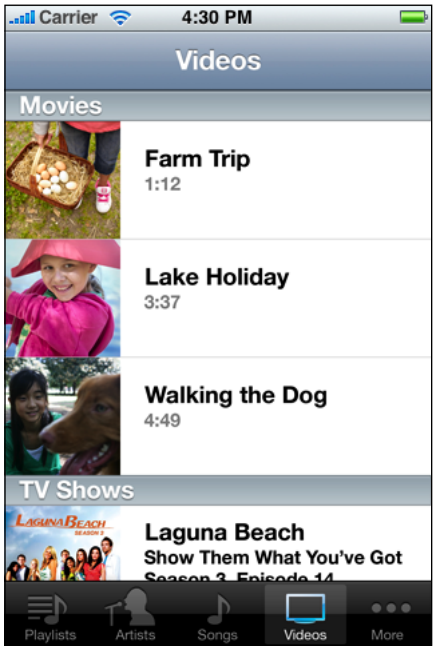


テーブルが移動に使用される場合、ユーザが階層の中でたどってきた道を戻るときには、前に選択したテーブルの行はハイライトされたままになっていません。

- **概念的にグループ化された情報の表示**

  TableViewスタイルのいずれかを使用して、情報を作業、家、学校のような論理的なグループにまとめることができます。プレーンテーブルとグループ化テーブルはどちらも、ヘッダとフッタのテキストを提供することで各セクションのコンテキストを提供できます。図 8-14（115 ページ）を参照してください。

図 8-14　プレーンテーブルのヘッダテキストはリストをセクションに分割する



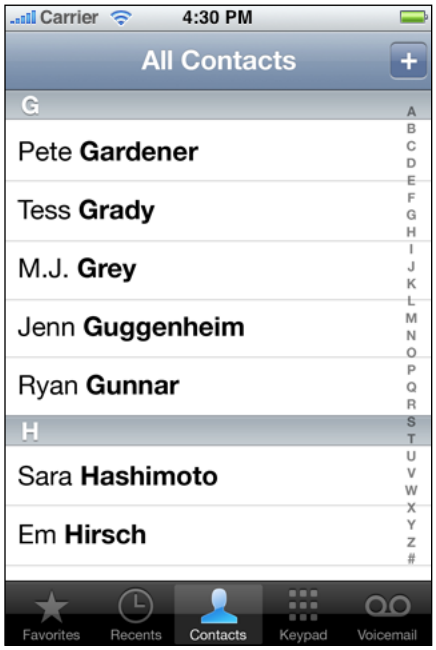
一般に、すばやくスクロールしたとしても、グループの丸い角はユーザが区別しやすいため、グループ化テーブルはグループを視覚的に明確に示します。図 8-15 （115 ページ）に、iPod設定の値の概念的なグループをいくつか示します。

図 8-15　グループ化テーブルには多数の個別のグループを含めることができる

■ インデックス付きの情報の検索

プレーンテーブルを使用している場合、ユーザが必要なものをすばやく見つけられるようにインデックスを表示できます。インデックスは、画面の右端に浮かぶ、エントリのカラムで構成されます（通常はアルファベット文字）。図 8-16 （116 ページ）を参照してください。 ユーザは、インデックスエントリをタップ（またはドラッグ）して、リスト内の対応する領域を表示します。インデックスは、複数の画面にまたがるようなリストで最も有効です。

図 8-16 インデックスを含むプレーンテーブル

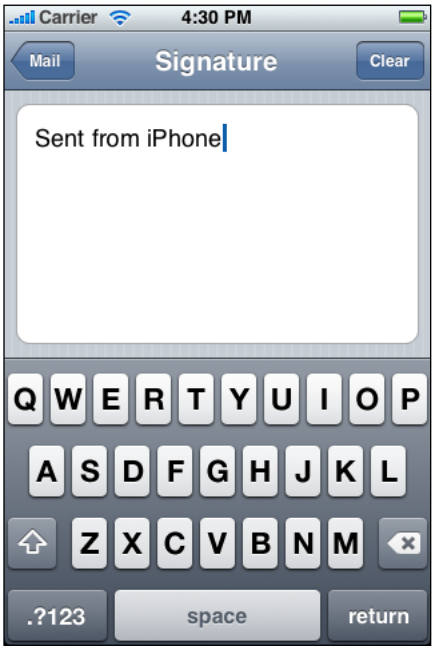


プレーンテーブルにインデックスを含める場合、インデックスと重なるため、テーブルの右端に表示されるTable View要素（ディスクロージャインジケータなど）の使用は避けます。

Text View

Text Viewは、複数行のテキストが表示されるリージョンであり、コンテンツが多すぎてText Viewの範囲に収まらないときにスクロールが可能です。図8-17に示すように、「メール(Mail)」はText Viewを使用して、ユーザが作成する各電子メールメッセージの末尾に付く署名を作成できるようにしています。

図 8-17 Text Viewは複数行のテキストを表示する



Text Viewを使用して、大きなテキスト文書の内容などのような多数のテキスト行を表示できますが、Text Viewはユーザによる編集をサポートするためにも使用できます。Text Viewを編集可能にすると、ユーザがText Viewの内側をタップしたときにキーボードが表示されます。キーボードの入力方式とレイアウトはユーザの言語設定によって決まります。ユーザがボタンラベル「.?123」をタップした場合（図 8-17を参照してください）、キーボードは数字と句読点の入力がしやすくなるように変更されます。期待するユーザ入力のタイプによって、異なるキーボードスタイルを指定することもできます。使用可能なキーボードスタイルの詳細については、「Text Field」（133 ページ）を参照してください。

Text Viewのテキストのフォント、色、および配置を制御できますが、テキスト全体に適用する場合のみです。つまり、テキストの一部だけを対象にこれらのプロパティを変更することはできません。フォントおよび色のプロパティのデフォルトは、予想にたがわず、システムフォントで黒色です。これが一般的には最も読みやすいからです。配置のプロパティのデフォルトは左揃えです（中央揃えまたは右揃えに変更できます）。

テキストを表示するビュー内で、可変のフォント、色、または配置を可能にする必要がある場合、Text Viewの代わりにWeb Viewを使用して、HTMLを使ってテキストにスタイルをあてることができます。

Web View

Web Viewは、アプリケーション画面に豊かなHTMLのコンテンツを表示できるリージョンです。たとえば、「メール(Mail)」はWeb Viewを使用して、メッセージの内容を表示します。これは、プレーンテキストのほかに要素が含まれている場合があるからです（図 8-18にこの例を示します）。

図 8-18 Web ViewはWebベースのコンテンツを表示できる

Web Viewは、Webコンテンツの表示のほかに、開いているWebページの間を移動するための要素をサポートします。Webページナビゲーション機能を提供することも選べますが、縮小版のWebブラウザのように見えて、そのように動作するアプリケーションを作成するのは避けるのが最善です。

WebページまたはWebアプリケーションを作成する場合、Web Viewを使用して、そのラッパーとして機能する簡単なiPhoneアプリケーションを実装することも選べます。自分の管理下にあるWebコンテンツへのアクセスを予定している場合、『*Safari Web Content Guide*』を読み、iPhone OSベースのデバイスでの表示に対応していて、そのために最適化されたWebコンテンツを作成する方法を学習してください。

アプリケーションコントロール

iPhone OSは、アプリケーションで使用できるコントロールをいくつか提供します。これらのコントロールのほとんどは、iPhone OSベースのデバイスユーザによく知られています。これらのコントロールの多くは、Table Viewなどの特定の場所での使用を目的としていますが、より一般的な使用法で利用できるコントロールもあります。この章では、アプリケーションの任意のビューで使用できるコントロールについて説明します。

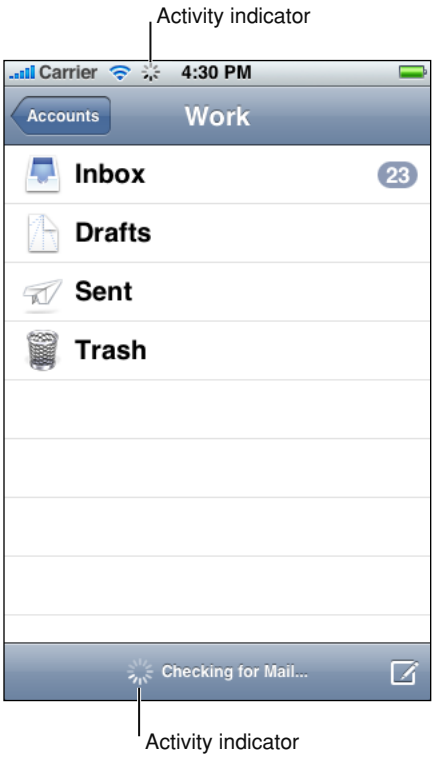
アプリケーションのユーザインターフェイスを設計する際には、ユーザは、なじみのあるコントロールの動作が組み込みのアプリケーションでの動作と同じであることを期待している点を必ず覚えておいてください。アプリケーションでこれらのコントロールを適切に使用する限り、これは利点になります。

アクティビティインジケータ

アクティビティインジケータは、持続時間が不明なタスクまたはプロセスの進捗を示します。持続時間が分かっているタスクの進捗を表示する必要がある場合には、代わりにProgress Viewを使用します（このコントロールの詳細については、「Progress View」（127 ページ）を参照してください）。アクティビティインジケータの「回転歯車」のように見えるものが、処理が実行中であることをユーザに示しますが、いつ終了するかは示しません。

図 9-1に、2種類のアクティビティインジケータを示します。ステータスバーのアクティビティインジケータは、ネットワークアクティビティのインジケータです。アプリケーションが数秒以上ネットワークにアクセスするときに表示します。アプリケーションが現在のタスクを実行する時間が1～2秒を上回る場合には、より大きなアクティビティインジケータをツールバーに表示します。

図 9-1 2種類のアクティビティインジケータ



アクティビティインジケータは、処理がいつ終了するかを示すよりも、タスクまたはプロセスが停止していないことをユーザに示して安心させるほうが重要であるときに使用すると良いフィードバックメカニズムです。

アクティビティインジケータのサイズと色は、アクティビティインジケータが表示されるビューの背景に合うものを選べます。デフォルトでは、アクティビティインジケータは白色です。

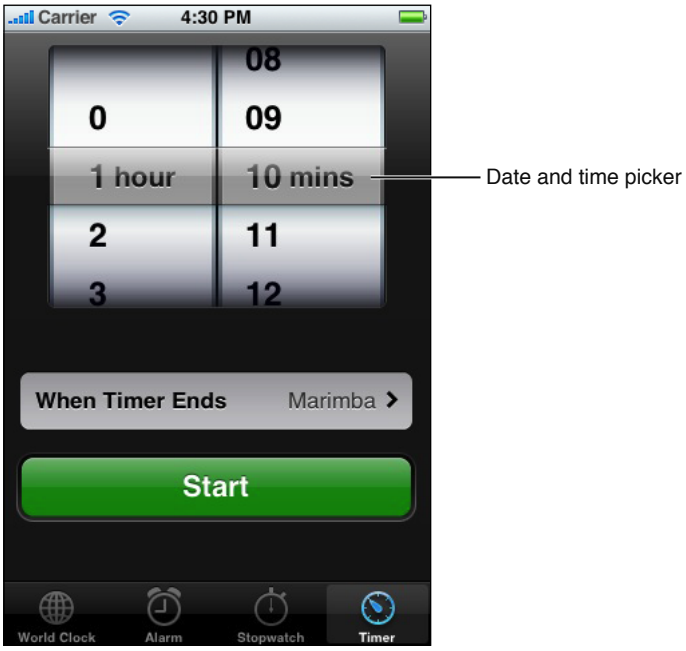
アクティビティインジケータは、タスクまたはプロセスが完了すると消えます。このデフォルトの動作をお勧めします。ユーザは何かが行われているときにアクティビティインジケータを目にすることを予想しており、アクティビティインジケータが止まっていればプロセスが停止していると認識します。

ネットワークアクティビティインジケータを表示する方法については、『*UIApplication Class Reference*』の`networkActivityIndicatorVisible`メソッドを参照してください。ネットワーク以外のアクティビティインジケータを表示する方法については、『*UIActivityIndicatorView Class Reference*』を参照してください。

日付と時刻のピッカー

日付と時刻のピッカーは、特定の日付と時刻を選択する簡単な方法をユーザに提供します。日付と時刻のピッカーは独立した回転ホイールを4つまで持つことができ、各回転ホイールは月または時間など、1つのカテゴリの値を表示します。ユーザは、フリックまたはドラッグして各ホイールを回転させ、望みの値をピッカーの中央を横切る半透明の選択バーの下に表示させます。最終的な値は、各ホイールに表示された値から成ります。図 9-2に、日付と時刻のピッカーの例を示します。

図 9-2 日付と時刻のピッカー



日付と時刻のピッカーを使用することで、ユーザが日付の年月日など複数の部分から成る値を入力する手間を避けることができます。日付と時刻のピッカーは、各部分の値の範囲が比較的小さく、ユーザはすでに各値が何であるのかを知っているので有効です。

日付と時刻のピッカーが表示するホイールの数は、指定するモードに応じて異なります。各ホイールには、それぞれ異なる一組の値があります。日付と時刻のピッカーでは、次のモードが定義されています。

- **時刻**。時刻モードは、時間および分の値のホイールを表示します。任意で、午前／午後を指定するホイールを追加できます。
- **日付**。日付モードは、年、月、日の値のホイールを表示します。
- **日付と時刻**。日付と時刻モードは、カレンダーの日付、時間、および分の値のホイールを表示します。任意で午前／午後を指定するホイールを追加できます。これがデフォルトのモードです。
- **カウントダウンタイマー**。カウントダウンタイマーモードは時間と分のホイールを表示します。カウントダウンの合計持続時間は最大で23時間59分まで指定できます。

デフォルトで、分のホイールは60個の値（0から59まで）を表示します。しかし、粗い間隔で選択肢を表示する必要がある場合には、数分間隔で表示するように分のホイールを設定できます。ただし、その均等な間隔の合計が60になる場合に限ります。たとえば、15分間隔の0、15、30、および45を表示できます。

設定にかかわらず、日付と時刻のピッカーの全体のサイズは固定されていて、キーボードと同じサイズです。日付と時刻のピッカーをビューの中心的要素にするか、または必要なときにだけ表示されるようにするかを選べます。たとえば、組み込みの「時計(Clock)」アプリケーションのタイマーモードは、常時表示される日付と時刻のピッカーです。これは、時刻の選択がタイマー機能の中心となるからです。一方、「日付と時刻を設定(Set Date & Time)」の環境設定（「自動設定(Set

Automatically)」がオフのときに、「設定(Settings)」>「一般(General)」>「日付と時刻(Date & Time)」で使用可能）では、ユーザが日付または時刻のどちらを設定するのかに応じて、日付のピッカーまたは時刻のピッカーが一時的に表示されます。

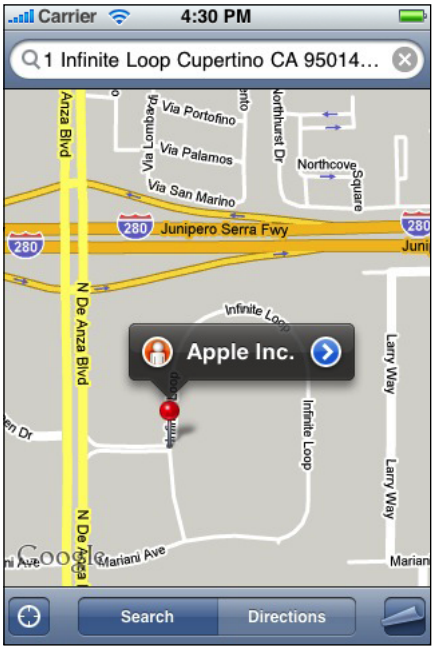
日付と時刻のピッカーをコードで使用する方法については、『*UIDatePicker Class Reference*』を参照してください。

詳細ディスクロージャボタン

詳細ディスクロージャボタンは、何かについての追加の情報、またはより詳細な情報を表示します。通常、詳細ディスクロージャボタンは、リスト項目の詳細情報を表示する手段をユーザに提供するTable Viewのなかで使用します（この使用法の詳細については、「Table Viewの要素」（110 ページ）を参照してください）。ただし、この要素はほかのタイプのビューで使用して、詳細情報や追加機能を表示する方法を提供できます。

たとえば、「マップ(Maps)」アプリケーションは、詳細ディスクロージャボタンを表示します。ユーザはそのボタンをタップして、ドロップされたピンに対応する追加の機能にアクセスできます。図9-3に詳細ディスクロージャボタンの例を示します。

図 9-3　詳細ディスクロージャボタンは追加の詳細または機能を表示する



詳細ディスクロージャボタンをコードで使用する方法については『*UIButton Class Reference*』を参照してください。

Infoボタン

「Info」ボタンは、アプリケーション設定の詳細を表示する手段を提供します。通常、これは画面の背面にあります。このため、「Info」ボタンは、ユーティリティ型アプリケーションに特に適しています。「天気(Weather)」アプリケーションの右下角に「Info」ボタンの例を見ることができます（図 9-4を参照）。

図 9-4　「Info」ボタンは情報（通常は、設定の詳細）を表示する

「Info」ボタンには、明るい背景および暗い背景を使用できます。明るい背景のスタイル（図9-4に示す）は、暗い背景のビューによく合います。逆に、暗い背景の「Info」ボタンは、明るい背景のビューによく映えます。

「Info」ボタンは、ユーザがタップすると短く光ります。iPhone OSが提供する「Info」ボタンを使用すると、この押された状態の外観を自動的に利用できます。

「Info」ボタンをコードで使用する方法については、『*UIButton Class Reference*』を参照してください。

ラベル

ラベルは、サイズが可変の静的テキストのかたまりです。図 9-5に、ラベルの例を示します。

図 9-5 ラベルはユーザに情報を提供する

ラベルを使用して、ユーザインターフェイスのパーツに名前を付けたり、ユーザに少量のヘルプを提供したりできます。ラベルは、比較的少ない量のテキストを表示するのに最も適しています。

フォント、テキストの色、配置などのラベルテキストのさまざまなプロパティを指定できますが、何よりもラベルを読みやすくすることに気をつけましょう。装飾的なフォントまたは目立つ色のために明瞭さを犠牲にしてはいけません。

ラベルのテキストを作成する際には、ユーザの語彙を使用するようにします。デベロッパ中心の言葉遣いになっていないかアプリケーションのテキストを調べ、ユーザ中心の言葉遣いに置き換えます。

ラベルをコードで使用する方法については『*UILabel Class Reference*』を参照してください。

ページインジケータ

ページインジケータは、アプリケーションで現在開いているビューの数だけ点を表示します。点は左から右に向かって、ビューが開かれた順番を示します（一番左の点が最初のビューを表します）。現在表示されているビューは、それを表す点が光ることで示されます。ユーザは光っている点の左または右をタップして、1つ前のビューまたは次のビューを表示します。図 9-6に、ページインジケータの例を示します。

図 9-6 ページインジケータ

ページインジケータは、いくつのビューが開かれているのか、またそれらのビューがどの順序で開かれたのかを簡単に確認する方法を提供します。ビューの階層をユーザがたどった通り道の追跡を支援するものではありません。ユーティリティ型アプリケーションのビューは互いに同等であることが多いため、ページインジケータはユーザがそれらのビューの間を移動するのに十分に役立ちます。一方、階層情報を表示する生産性型アプリケーションは、Navigation Barの要素を通して移動できるようにするべきです（この詳細については、「Navigation Bar」（78 ページ）を参照してください）。

一般に、ページインジケータはアプリケーション画面の下端近く、アプリケーション画面に含まれるビューの下で十分に機能します。これによって、ユーザが簡単に目にすることのできる画面の上部に、より重要な情報（ビュー自体）を配置できます。ビューの下端と画面の下端の間でページインジケータが垂直方向の中央に置かれるようにします。

ページインジケータに表示できる点の数にプログラム上の制限はありませんが、多く表示しても点は縮んだり、間隔が狭まったりしないことに留意してください。たとえば、縦向きでは、クリッピングが起きる前にページインジケータに最大で20個の点を表示できます。したがって、このような状態を避けるためのロジックをアプリケーションに用意するべきです。

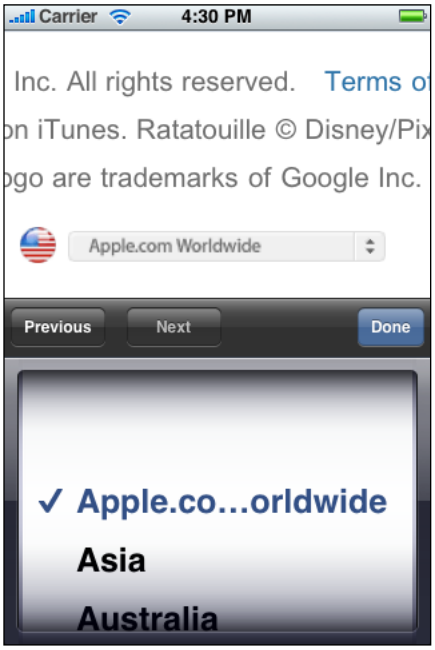
開いているビューが1つのみのとき、ページインジケータを非表示にできますが、デフォルトの動作はページインジケータを表示することです。

ページインジケータをコードで使用する方法については『*UIPageControl Class Reference*』を参照してください。

ピッカー

ピッカーは、日付と時刻のピッカーの汎用版です（日付と時刻のピッカーの詳細については、「日付と時刻のピッカー」（120 ページ）を参照してください）。ピッカーを使用して、任意の値の集合を表示できます。日付と時刻のピッカーと同様に、ユーザは望みの値が表示されるまでピッカーの1つ以上のホイールを回転させます。図 9-7に、1つのホイールを持つピッカーを示します。

図 9-7　iPhone版Safariに表示されたピッカー



アプリケーションでピッカーを使用するかどうか決めるときには、ホイールが静止状態のとき、ほとんどでないとしても多くのホイールの値がユーザから隠されていることを考慮します。ユーザがこれらの値が何であるかをすでに知っていれば、これは必ずしも問題ではありません。たとえば、日付と時刻のピッカーでは、ユーザは月のホイールの隠されている値が1から12の間の数字でしかあり得ないことを理解しています。しかし、そのような広く知られている集合に属さない選択肢を提供する必要がある場合には、ピッカーがコントロールとして適切でない場合があります。

数多くの値を表示する必要がある場合には、ピッカーではなくTable Viewに値をリストするほうが良いでしょう。これは、Table Viewのほうが高さがあり、スクロールが速くなるからです。

測定の単位など、ピッカーの値に文脈を与える必要がある場合には、それをコントロールの中央を水平に横切る半透明の選択バーに表示します。ピッカーの上側またはホイール自体にそのようなラベルは表示しないでください。ラベルを表示する正しい方法の例として、組み込みの「時計(Clock)」アプリケーションのタイマー機能を参照してください。「時間(hours)」および「分(min)」が、ユーザの選択した値の隣に表示されています。

日付と時刻のピッカーと同様に、汎用ピッカーは常時表示したり（ユーザインターフェイスの中心として）、必要なときにだけ表示したりできます。ピッカーの背景を含む全体のサイズは、固定であり、キーボードと同じサイズです。

ピッカーをコードで使用する方法については『*UIPickerView Class Reference*』を参照してください。

Progress View

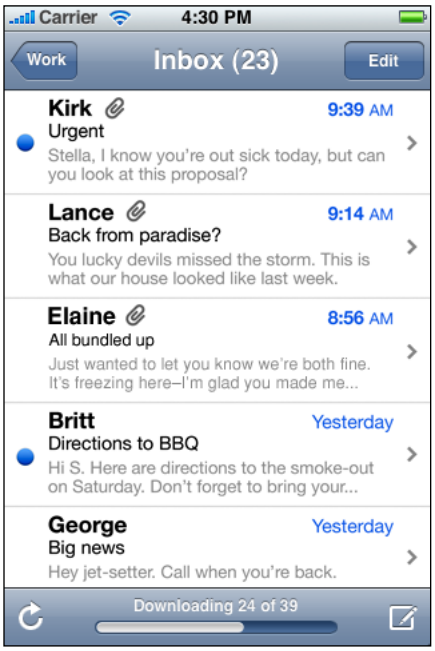
Progress Viewは、持続時間が分かっているタスクやプロセスの進捗を示します。持続時間が不明なタスクの進捗を表示する必要がある場合には、代わりにアクティビティインジケータを使用します（このコントロールの詳細については、「アクティビティインジケータ」（119 ページ）を参照してください）。

iPhone OSは2種類のProgress Viewを提供します。デフォルトスタイルとバースタイルです。各スタイルの外観は、高さを除いてとても似ています。

- デフォルトスタイルは、アプリケーションの主要なコンテンツ領域での使用を目的とする。
- バースタイルはデフォルトスタイルよりも薄く、このためツールバーでの使用に適している。たとえば、ユーザが新しいメッセージをダウンロードしたり、電子メールメッセージを送信したりすると、「メール(Mail)」のツールバーにバースタイルのProgress Viewが表示されます。

両方のスタイルのProgress Viewの動作は同じです。タスクまたはプロセスが進むにつれて、Progress Viewのトラックが左から右に満たされていきます。ビューの満たされていない領域に対する満たされた領域の任意の時点における割合が、ユーザにタスクまたはプロセスがあとどのくらいで終了するのかを示します。図 9-8に、バースタイルのProgress Viewの例を示します。

図 9-8　ツールバーにおけるバースタイルのProgress View



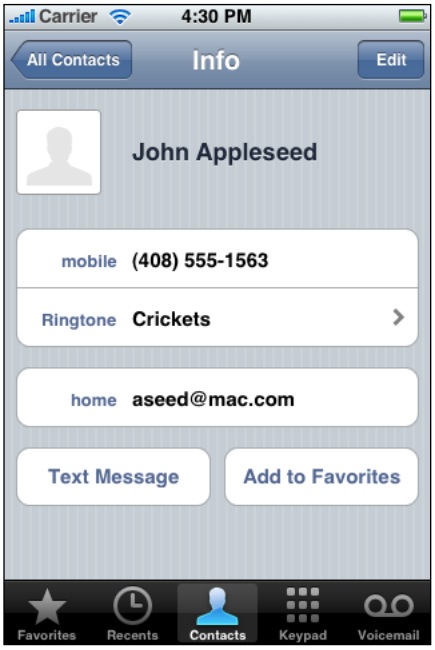
Progress Viewは、持続時間が明確に定まっているタスクについてユーザにフィードバックを提供する良い方法です。特に、タスクに要するおおよその時間をユーザに示すことが重要である場合に良い方法です。Progress Viewを表示することで、ユーザにタスクが実行されていることを伝え、タスクが完了するまで待つか、またはタスクをキャンセルするのかを決定するのに十分な情報を提供します。

Progress Vlewをコードで使用する方法については『*UIProgressView Class Reference*』を参照してください。

角丸矩形のボタン

角丸矩形のボタンは、アクションを実行するためにビューで使用できる多用途のボタンです。特定の個人の「連絡先(Contacts)」ビューの一番下にあるボタンが、このタイプのボタンの例です。図9-9に示すように、「SMS/MMS (Text Message)」ボタンおよび「よく使う項目に追加(Add to Favorites)」ボタンは角丸矩形のボタンです。

図 9-9　角丸矩形のボタンはアプリケーションに固有のアクションを実行する



角丸矩形ボタンにタイトルを付ける場合は、次の点に注意してください。

- タイトル形式の大文字化を使用する（つまり、タイトル形式の大文字化とは、4文字以下の冠詞、等位接続詞、および前置詞を除くすべての単語を大文字で始める）。
- 長すぎるタイトルは作成しない。長すぎるタイトルは切り詰められるため、ユーザにわかりにくくなります。

角丸矩形ボタンをコードで使用する方法については、『*UIButton Class Reference*』を参照してください。

Search Bar

Search Barは、アプリケーションが検索の入力として使用できるユーザからのテキストを受け取るフィールドです。ユーザがSearch Barをタップすると、キーボードが表示されます。ユーザが検索用語の入力を終えると、その入力はアプリケーション固有の方法で処理されます。アプリケーションでの検索処理に関するガイドラインについては、「検索機能の提供と検索結果の表示」（58 ページ）を参照してください）。

デフォルトでは、Search Barは左側に検索アイコンを表示します。さらに、Search Barは次のいくつかの任意の要素を表示できます。

- プレースホルダーテキスト。このテキストは、コントロールの機能（たとえば「検索(Search)」）を提示したり、ユーザにどのような文脈で検索しているのか（たとえば「YouTube」または「Google」）を想起させます。
- 「ブックマーク(Bookmarks)」ボタン。このボタンは、ユーザが簡単にもう一度見つけたい情報へのショートカットを提供できます。たとえば、「マップ(Maps)」の検索モードの「ブックマーク(Bookmarks)」ボタンは、ブックマークされた場所、検索履歴、および連絡先へのアクセスを提供します。
- 「消去(Clear)」ボタン。ほとんどのSearch Barには、ユーザが1回のタップでSearch Barの内容を消去できる「消去(Clear)」ボタンが含まれます。
- Search Barの上に表示されるプロンプトと呼ばれる説明的な表題。たとえば、手がかりやアプリケーション固有の文脈をSearch Barに提供する短いフレーズをプロンプトとして表示できます。

図9-10に、カスタマイズされたプレースホルダーテキスト、「ブックマーク(Bookmarks)」ボタン、およびデフォルトの検索アイコンを持つSearch Barを示します。

図 9-10 任意指定のプレースホルダーテキストおよび「ブックマーク(Bookmarks)」ボタンを持つSearch Bar

デフォルトでは、「ブックマーク(Bookmarks)」ボタンおよび「消去(Clear)」ボタンは次のように互いに作用します。

- Search Barにプレースホルダー以外のテキストが含まれている場合、「消去(Clear)」ボタンが表示され、ユーザはテキストを消すことができます。Search Barにユーザの指定したテキストまたはプレースホルダー以外のテキストがない場合には、「消去(Clear)」ボタンは非表示になります。これは、Search Barの内容を消す必要がないからです。
- 「ブックマーク(Bookmarks)」ボタンは、Search Barにユーザの指定したテキストまたはプレースホルダー以外のテキストがない場合にのみ表示されます。これは、Search Barにユーザが消したい可能性のあるテキストがあるときに「消去(Clear)」ボタンが表示されるからです。

次のような標準色の背景スタイルの1つを指定してSearch Barをカスタマイズできます。

- デフォルト（ツールバーおよびナビゲーションバーのデフォルトの外観に合わせたグラデーション）。デフォルトの背景スタイルは、図 9-10に示しています。
- 黒

さらに、Search Barの下にScope Barを表示することができます。ユーザは、ここに含まれるボタンをタップして検索範囲を選択できます。Scope Barは、Search Barに指定できるものと同じ外観を採用しています。また、スコープボタンに独自のタイトルを付けることができます。

Scope Barは、コード内でSearch Display Controllerを使用しない限りは、デバイスの向きに関係なく、Search Barの下に表示されます（詳細については『*UISearchDisplayController Class Reference*』を参照）。Search Display Controllerを使用すると、Scope Barは、横向きの場合はSerach Bar内の検索フィールドの右に表示されます（縦向きの場合には、Search Barの下に表示されます）。

Search BarとScope Barをコードで使用する方法については『*UISearchBar Class Reference*』を参照してください。

Segmented Control

Segmented Controlは、直線形のセグメントセットです。それぞれのセグメントは、異なるビューを表示できるボタンとして機能します。ユーザがSegmented Controlのセグメントをタップすると、瞬間的なアクションまたは目に見える結果が起こります。たとえば、図 9-11 に示すように、ユーザがSegmented Controlを使用して、電子メールのプロトコルを選択すると、「設定(Settings)」は異なる情報を表示します。

図 9-11 3つのセグメントを持つSegmented Control

Segmented Controlの長さは、表示するセグメントの数および一番大きいセグメントのサイズによって決まります。Segmented Controlの高さは固定です。表示するセグメントの数は指定できますが、ユーザが、隣接するセグメントをタップする心配をせずに楽にセグメントをタップできる必要があることに留意してください。ヒットリージョンは44×44ピクセルとするべきであるため、Segmented Controlは5つ以下のセグメントにすることをお勧めします。

Segmented Controlは、テキストまたは画像を含むことができます。個々のセグメントはテキストまたは画像のどちらか一方だけを含むことができます。一般的に、1つのSegmented Controlにテキストと画像を混在させることは避けるのが最善です。

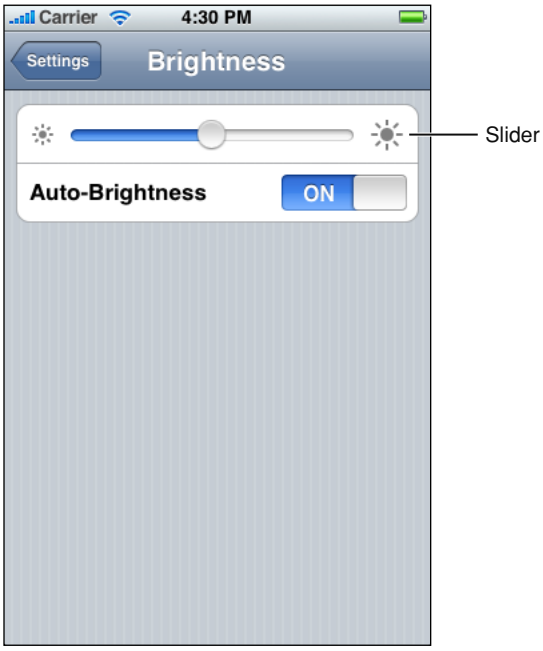
Segmented Controlでは、各セグメントの幅はセグメントの総数に比例します。これは、各セグメントに対してデザインするコンテンツは、互いにおおよそ等しいサイズになるようにする必要があることを意味します。

Segmented Controlをコードで使用する方法については、『*UISegmentedControl Class Reference*』を参照してください。

Slider

Slider（スライダ）は、ユーザが可能な値の全範囲にわたって値やプロセスを調整できるようにします。ユーザがSliderをドラッグすると、値またはプロセスが連続的に更新されます。図9-12に、最小および最大を表すイメージを持つスライダの例を示します。

図 9-12 Slider

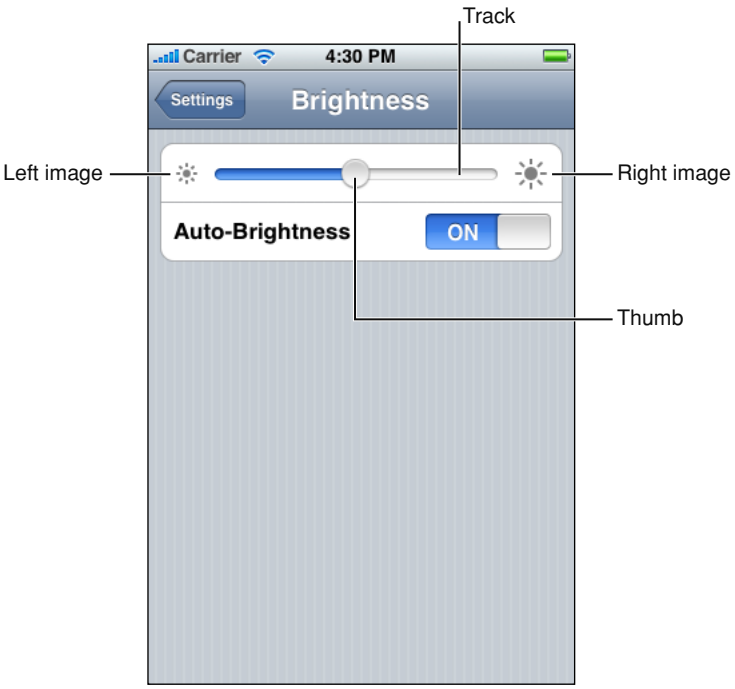


Sliderは、主に次の2つの状況で役に立ちます。

- 選択した値全体にわたり、ユーザが細かい制御をできるようにする場合
- ユーザが現在のプロセスのきめ細かい制御をできるようにする場合

Sliderは、トラック、サム、および任意の左右の値の画像からなります。図 9-13に、Sliderのこれらのパーツを示します。

図 9-13 Sliderの4つのパーツ



アプリケーションのユーザインターフェイスに収まるようにSliderの幅を設定できます。さらに、Sliderを水平または垂直に表示できます。

Sliderをカスタマイズするには、いくつかの方法があります。

- サムの外観を定義し、スライダが有効であるかどうかを一目でユーザが確認できるようにする。
- Sliderの両端に表示する画像を提供し（通常、これらは最小値および最大値に対応する）、Sliderの目的をユーザに理解しやすくする。

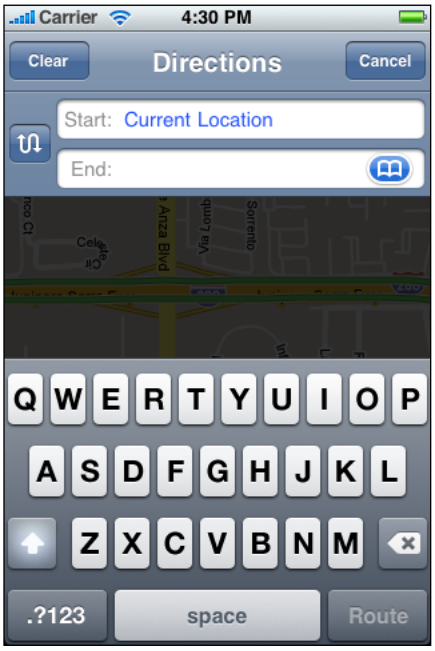
  たとえば、フォントサイズを制御するSliderは、最小の端で小さな文字を表示し、最大の端で大きな文字を表示できます。
- トラックがサムのどちら側にあるのか、およびコントロールがどの状態にあるのかに応じて、トラックに異なる概観を定義できる。

Sliderをコードで使用する方法については、『*UISlider Class Reference*』を参照してください。

Text Field

Text Fieldは、ユーザの入力を受け取る角丸矩形フィールドです。ユーザがText Fieldをタップすると、キーボードが表示されます。ユーザがキーボードの「改行(Retrun)」をタップすると、Text Fieldはアプリケーション固有の方法で入力を処理します。Text Fieldには、1行を入力できます。図 9-14に、「マップ(Maps)」アプリケーションの2つのText Fieldを示します。

図 9-14 Text Fieldはユーザ入力を受け取る



Text Fieldをカスタマイズして、アプリケーションのテキストフィールドをどのように使用するのかをユーザが理解しやすくできます。たとえば、Text Fieldの左または右側にカスタム画像を表示したり、図9-14に示した「ブックマーク(Bookmarks)」ボタンなどのシステムに用意されているボタンを表示したりできます。一般的に、Text Fieldの左端を使用してその目的を示し、右端を使用してブックマークなどの追加の機能の存在を示します。

Text Fieldの右端に「消去(Clear)」ボタンを表示することもできます。この要素が表示されているときには、ほかの画像がこの要素の上に表示されているかどうかにかかわらず、この要素をタップするとText Fieldのコンテンツが消去されます。

場合によっては、「名前(Name)」のようなヒントを表示すると、ユーザがText Fieldの目的を理解しやすくなります。Text Fieldは、プレースホルダーテキストなどの表示をサポートします。プレースホルダーテキストなどは、フィールドにほかのテキストがないときに表示できます。Text Fieldの使用方法、画像とボタンを表示するためのカスタマイズ方法の詳細については、『*UITextField Class Reference*』を参照してください。

ユーザが入力すると考えられる各種の内容に対応するために、異なるキーボードスタイルを指定することができます（ユーザの言語設定によって決まる、キーボードの入力方式とレイアウトは制御できないことに注意してください）。たとえば、URL、PIN、電話番号をユーザが入力しやすくなるようにしたいとします。iPhone OSでは、各種の入力をしやすくするためにそれぞれ設計された、複数の異なるキーボードタイプが提供されています。利用可能なキーボードタイプについては、`UIKeyboardType`を参照してください。アプリケーションのキーボードを管理する方法については、『*Text and Web Programming Guide for iOS*』の「Managing the Keyboard」を参照してください。

システムに用意されているボタンおよびアイコン

一貫性のあるユーザ体験を推進するために（およびデベロッパの仕事を容易にするために）、iPhone OSはNavigation BarおよびToolbarで使用できる数々の標準ボタン、およびTab Barで使用できる数々のアイコンを提供します。

この章では、使用可能な標準アイコンおよびボタンについて説明し、それらの適切な使用方法の手引きを示します。開発しているアプリケーションのタイプにかかわらず、この章のボタンおよびアイコンを把握しておきましょう。そうすると、次のことが可能です。

- システムに用意されている項目を正しく使用できる
- システムに用意されているアイコンに似すぎたカスタムアイコンをデザインすることを避けられる

システムに用意されているボタンおよびアイコンの使用

iPhone OSでは、標準のToolbarおよびNavigation Barのボタン、Tab Bar項目、および組み込みのアプリケーション全体で使用されている一般用途のボタンの多くを使用できます。図 10-1に示すように、いくつかの標準ツールバーボタンは、「メール(Mail)」ツールバーに見ることができます。

図 10-1　「メール(Mail)」ツールバーの標準ボタン

図 10-1に示した「更新(Refresh)」、「移動(Organize)」、「ゴミ箱(Trash)」、「返信(Reply)」および「作成(Compose)」などのボタンは、組み込みのアプリケーションの多くで一貫して使用されているため、ユーザはそれらが何を意味し、それらをどのように使用するのかをよく知っています。これは、アプリケーションがこれらの機能をサポートする場合には、ユーザが精通していることを利用して、アプリケーションのユーザインターフェイスを合理化できることを意味します。また、これらのボタンをほかのタスクに関連付けた場合、ユーザが期待する機能を約束しておきながら、別のものを提供することでユーザを混乱させ、苛立たせる可能性があります。

システムに用意されているボタンおよびアイコンは、ユーザの過去の体験を利用できる利点に加えて、ほかに2つの重要な利点を提供します。具体的には次の点です。

- 標準的な機能を表すカスタムの作品を作成する必要がないため、開発時間が短縮される。
- 将来iPhone OSのアップデートによって標準アイコンの外観が変わっても、ユーザインターフェイスの高い安定性が得られる。つまり、アイコンの外観が変わっても、標準アイコンの意味が同じままであることを当てにできます。

重要なので繰り返しますが、ユーザが精通していることの利点、開発時間の短縮、およびユーザインターフェイスの意味の一貫性を実現するには、ボタンおよびアイコンを適切に使用する必要があります。具体的には、ボタンまたはアイコンの外観に対する*自分の解釈によるのではなく*、文書化されている意味および推奨される配置に従ってボタンやアイコンを使用するべきことを意味します。システムに用意されているボタンおよびアイコンの意味と配置情報については、「ToolbarおよびNavigation Barで使用可能な標準ボタン」（136 ページ）、「Tab Barで使用可能な標準アイコン」（138 ページ）、および「テーブルの行およびほかのユーザインターフェイス要素で使用可能な標準ボタン」（139 ページ）を参照してください。

Interface Builderを使うと、システムに用意されている、ボタンの使用やアイコンのコントロールへの提供を簡単に行えます。ガイダンスについては、『*Interface Builder User Guide*』の「iOS Interface Objects」の外観に関連する情報を参照してください。

システムに用意されているなかで、アプリケーションの特定の機能に対して適切な意味を持つToolbarまたはNavigation Barのボタン、またはTab Barのアイコンが見つからない場合には、カスタムのボタンまたはアイコンをデザインする必要があります。「Navigation Bar、Toolbar、およびTab Barのアイコン」（147 ページ）に、これを行うために役立つ手引きをいくつか示します。

ToolbarおよびNavigation Barで使用可能な標準ボタン

iPhone OSでは、ユーザがToolbarおよびNavigation Barで目にする標準ボタンの多くが、デベロッパが使用できるようになっています。表 10-1（137 ページ）に示すこれらのボタンは、2つのスタイルで使用できます。それぞれのスタイルは、次に示す特定の使用法に適しています。

- ボーダー付きスタイル — たとえば、「電話(Phone)」の「連絡先(Contacts)」のNavigation Barの追加ボタン。このスタイルはNavigation BarおよびToolbarの両方に適しています。
- プレーンスタイル — たとえば、「メール(Mail)」のToolbarの「作成(Compose)」ボタン。このスタイルはToolbarにのみ適しています。実際に、Navigation Barのボタンにプレーンスタイルを指定すると、ボーダー付きスタイルに変換されます。

システムに用意されているすべてのボタンと同様に、表 10-1で説明しているボタンを使用して、デザインの対象アクション以外のアクションを表すことは避けるべきです。特に、その文書化されている意味に関係なく、ボタンの外観に基づいてボタンを選ぶことは避けます。これらのアイコンを

正しく使用することがなぜ重要なのかについては、「システムに用意されているボタンおよびアイコンの使用」（135 ページ）を参照してください（これらのボタンのシンボル名と利用できるかどうかの詳細については、`UIBarButtonSystemItem`のドキュメントで参照できます）。

表 10-1　ToolbarおよびNavigation Barで使用可能な標準ボタン（プレーンスタイルで表示）

ボタン	意味	名前
	ユーザが、アプリケーション固有のアクションを実行できるAction Sheetが開く	Action
	カメラモードで写真ピッカーを表示するAction Sheetを開く	Camera
	新規メッセージビューを編集モードで開く	Compose
	アプリケーション固有のブックマークを表示する	Bookmarks
	検索フィールドを表示する	Search
	新しい項目を作成する	Add
	現在の項目を削除する	Trash
	フォルダなど、アプリケーション内の目的の場所に項目を移動する、または送る	Organize
	項目を別の場所へ移動する、または送る	Reply
	現在のプロセスまたはタスクを中止する	Stop
	コンテンツを更新する（必要なときだけ使用する。それ以外では自動的に更新する）	Refresh
	メディア再生またはスライドを開始する	Play
	メディア再生またはスライドの早送りをする	FastForward
	メディア再生またはスライドを一時停止する（これはコンテキストの維持の意味を含む）	Pause
	メディア再生またはスライドを逆方向に移動する	Rewind

表 10-1 に示すボタンのほかに、表 10-2に示す、システムに用意されている「編集(Edit)」、「キャンセル(Cancel)」、「保存(Save)」、および「完了(Done)」ボタンを使用し、アプリケーションの編集またはほかのタイプのコンテンツ操作もサポートできます。（これらのボタンのシンボル名と利用できるかどうかの詳細については、`UIBarButtonSystemItem`のドキュメントで参照できます）。こ

れらのボタンはNavigation BarおよびToolbarの両方に適しており、ボーダー付きスタイルでのみ使用できます。これらのボタンの1つにプレーンスタイルを指定すると、ボーダー付きスタイルに変換されます。

表 10-2 Navigation Barで使用可能なボーダー付きアクションボタン

ボタン	意味	名前
	編集モードまたはコンテンツ操作モードに入る	Edit
	変更を保存せずに編集モードまたはコンテンツ操作モードを終了する	Cancel
	変更を保存し、適切であれば編集モードまたはコンテンツ操作モードを終了する	Save
	現在のモードを終了し、変更があれば変更を保存する	Done

Tab Barで使用可能な標準アイコン

iPhone OSは、表 10-3に示す、Tab Barで使用する標準アイコンを提供します（これらのアイコンのシンボル名と利用できるかどうかの詳細については、`UITabBarSystemItem`のドキュメントで参照できます）。

すべての標準ボタンおよびアイコンと同様に、これらのアイコンを文書化された意味に従って使用することが肝要です。特に、アイコンの外観ではなく、その意味に基づいてアイコンを使用するように気を付けます。これは、たとえ特定の意味に対応するアイコンの外観が変わっても、アプリケーションのユーザインターフェイスの意味を保つのに役立ちます。これらのアイコンを正しく使用することがなぜ重要なのかについて詳しくは、「システムに用意されているボタンおよびアイコンの使用」（135 ページ）を参照してください。

表 10-3 Tab Barのタブで使用可能な標準アイコン

アイコン	意味	名前
	アプリケーション固有のブックマークを表示する	Bookmarks
	「連絡先(Contacts)」を表示する	Contacts
	ダウンロードを表示する	Downloads
	ユーザ定義のよく使う項目を表示する	Favorites

アイコン	意味	名前
	アプリケーションの推奨コンテンツを表示する	Featured
	ユーザアクションの履歴を表示する	History
	追加のTab Barの項目を表示する	More
	最新の項目を表示する	MostRecent
	すべてのユーザに最も人気のある項目を表示する	MostViewed
	アプリケーションで定義した期間内にユーザがアクセスした項目を表示する	Recents
	検索モードに入る	Search
	ユーザが決める最も高く評価された項目を表示する	TopRated

テーブルの行およびほかのユーザインターフェイス要素で使用可能な標準ボタン

iPhone OSは、テーブルの行およびほかの要素で使用するいくつかのボタンを提供します。表 10-4で説明するこれらのボタンは、すべての標準ボタンおよびアイコンと同様に意味的に正しく使用する必要があります。特に、その文書化されている意味に関係なく、ボタンの外観に基づいてボタンを選ぶことは避けます。これらのボタンを正しく使用することがなぜ重要なのかについては、「システムに用意されているボタンおよびアイコンの使用」（135 ページ）を参照してください。

詳細ディスクロージャボタンは通常、テーブルの行で使用しますが、ほかの場所でも使用できます。このボタンの詳細については、「詳細ディスクロージャボタン」（122 ページ）を参照してください。iPhone OSは、テーブルの行のみで使用する1組のコントロールを提供します。これらの詳細については、「Table Viewの要素」（110 ページ）を参照してください。

これらのボタンのシンボル名と利用できるかどうかの詳細については、`UIButtonType`のドキュメントを参照してください（詳細ディスクロージャのTable View要素のシンボル名と利用できるかどうかの詳細については、`UITableViewCellAccessoryDetailDisclosureButton`のドキュメントを参照してください）。

表 10-4　テーブルの行およびユーザインターフェイス要素で使用可能な標準ボタン

ボタン	意味	名前
	項目に連絡先を追加するためのPeopleピッカーを表示する	ContactAdd

ボタン	意味	名前
	現在の項目の詳細を含む新しいビューを表示する	DetailDisclosure
	ビュー（通常、ユーティリティ型アプリケーションのビュー）の背面に切り替えて、設定オプションまたは詳細を表示する。 「Info」ボタンは、暗い円の中に明るい色の「i」をあしらったものも用意されています。	Info

カスタムアイコンおよび画像の作成

どのアプリケーションも、アプリケーションアイコンと起動画像が必要です。また、アプリケーションは、Spotlightの検索結果（および、必要であれば「設定(Settings)」）に表示するiPhone OS用のアイコンも提供することをお勧めします。さらに、アプリケーションによっては、Navigation Bar、Toolbar、およびTab Barで、カスタムの書類（ドキュメント）タイプ、またはアプリケーション固有の機能およびモードを表すためにカスタムアイコンが必要です。

アプリケーションのほかのカスタムアートワークとは異なり、これらのアイコンと画像は、iPhone OSで正しく表示するため、特定の条件を満たす必要があります。表 11-1に、これらのアイコンと画像についての要約情報を示し、作成のための具体的なガイドラインへのリンクを提供します。これらのファイルの名前の付け方と、コードの中で指定する方法については、『*iOS Application Programming Guide*』の「Application Icons」および『*iOS Application Programming Guide*』の「Application Launch Images」を参照してください。

注： iPhoneおよびiPod touchで解像度に依存しないようにするには、すでに提供しているリソースに追加して、アイコンと画像の高解像度バージョンを提供するべきです。高解像度アートワークを最大限に活かす方法のガイドラインについては、「高度な高解像度アートワークの作成のヒント」（151 ページ）を参照してください。

表 11-1 カスタムのアイコンおよび画像

説明	iPhoneおよびiPod touch向けのサイズ（ピクセル単位）	iPad向けのサイズ（ピクセル単位）	ガイドライン
アプリケーションアイコン（**必須**）	57×57 114×114（高解像度）	72×72	「アプリケーションアイコン」（142 ページ）
App Storeアイコン（**必須**）	512×512	512×512	「アプリケーションアイコン」（142 ページ）
Spotlight検索結果と「設定(Settings)」用の小さなアイコン（推奨）	29×29 58×58（高解像度）	Spotlight検索結果用50×50 「設定(Settings)」用29×29	「小さいアイコン」（144 ページ）
ドキュメントアイコン（カスタムのドキュメントタイプに推奨）	22×29 44×58（高解像度）	64×64 320×320	「ドキュメントアイコン」（145 ページ） （iPadの場合は、『iPad User Experience Guidelines』を参照）

説明	iPhoneおよびiPod touch向けのサイズ（ピクセル単位）	iPad向けのサイズ（ピクセル単位）	ガイドライン
Webクリップ(Web Clip)アイコン（Webアプリケーションおよびウェブサイト用に推奨）	57×57 114×114（高解像度）	72×72	「Webクリップ(Web Clip)アイコン」（146 ページ）
ToolbarおよびNavigation Barのアイコン（省略可能）	約20×20 約40×40（高解像度）	約20×20	「Navigation Bar、Toolbar、およびTab Barのアイコン」（147 ページ）
Tab Barのアイコン（省略可能）	約30×30 約60×60（高解像度）	約30×30	「Navigation Bar、Toolbar、およびTab Barのアイコン」（147 ページ）
起動画像（**必須**）	320×480 640×960（高解像度）	縦向き用： 768×1004 横向き用： 1024×748	「起動画像」（148 ページ）

注： すべての画像およびアイコンは、PNG形式が推奨されます。

アイコンおよび画像の標準のビット深度は24ビット（赤、緑、青についてはそれぞれ8ビット）に、8ビットのアルファチャネルを加えたものです。

パレットをWebセーフカラーに制限する必要はありません。Navigation Bar、Toolbar、およびTab Bar用に作成するアイコンにはアルファ透明度を使用できますが、アプリケーションアイコンには使用しないでください。

アプリケーションアイコン

アプリケーションアイコンは、ユーザがホーム(Home)画面に置き、タップしてアプリケーションを起動するアイコンです。これが、ブランド設定と強力な視覚デザインが合体して、コンパクトで瞬時に認識可能な魅力的なパッケージになるところです。アプリケーションはすべて、アプリケーションアイコンが必要です。

豊かで美しく、アプリケーションの目的の本質を明確に伝えられるように、アイコンの見栄えと意味の明確さのバランスをとるようにしてください。また、画像および色の選択が異なる文化の人々にどのように受け取られるのかを調査するのもよい考えです。

異なるデバイス用にさまざまなサイズのアプリケーションアイコンを作成する。Universalアプリケーションを作成する場合、3つすべてのサイズのアプリケーションアイコンを提供する必要があります。

iPhoneおよびiPod touchでは、次の2つのサイズのアイコンが必要です。

- 57×57ピクセル
- 114×114ピクセル（高解像度）

iPadでは、次のサイズのアイコンが必要です。

- 72×72ピクセル

iPhone OSがデバイスのホーム(Home)画面にアプリケーションアイコンを表示するとき、次の視覚効果が自動的に追加されます。

- 角丸
- ドロップシャドウ
- 反射光（光沢効果を使用しない場合を除く）

たとえば、次のようなシンプルな57×57ピクセルのiPhoneアプリケーションアイコンがあるとします。

これがiPhoneのホーム(Home)画面に表示されると、次のように表示されます。

注： iPhone OSがアプリケーションアイコンに光沢を追加しないようにすることもできます。それには、アプリケーションの`Info.plist`ファイルに`UIPrerenderedIcon`キーを追加します（このファイルの詳細については、『*iOS Application Programming Guide*』の「The Information Property List」を参照してください）。
光沢が追加されるかどうかにかかわらず、アプリケーションアイコンの寸法は変わりません。

アイコンが、iPhone OSが提供する視覚効果に適切であるようにする。次のような画像を作成するようにします。

- 90度の角を持つ
- 輝きや光沢がない（反射光の追加を使用しないことを選んだ場合を除く）
- アルファ透明度を使用しない

認識可能な背景をアプリケーションアイコンに与える。iPhone OSによって丸い角が追加されるため、背景を表示するアイコンがホーム(Home)画面で最適な外観になります。これは、すべてのアイコンが一様に丸い角を持つことで、タップしやすい一貫性のある外観をユーザのホーム(Home)画面が確実に持つようになるからです。ホーム(Home)画面に表示されるときに背景が表示されないアイ

コンを作成すると、ユーザには丸い角が見えません。このようなアイコンは、タップできるように見えないことが多く、ユーザが評価するホーム(Home)画面の規則正しい調和を妨げる傾向があります。

必要な領域を完全に画像で満たす。画像の境界が推奨されるサイズよりも小さい場合、または画像内に「透過領域」を作るために透明効果を使っている場合は、アイコンが角丸の黒い背景に浮かんでいるように見えるでしょう。

たとえば、アプリケーションは、左端の青い星のような、透明な背景上にアイコンを提供することも可能です。iPhone OSがホーム(Home)画面にこのアイコンを表示すると、中央に示した画像のように見えるか（光沢を追加しない場合）、右に示した画像のように見えます（光沢を追加した場合）。

表示されている黒の背景に浮いているように表示されるアイコンは、カスタムの画像を表示するホーム(Home)画面で特に見栄えがしません。

App Storeに表示するためアプリケーションアイコンの512×512ピクセルバージョンを作成する。このバージョンのアイコンは、開発したアプリケーションのアイコンであると即座に認識できることが重要ですが、さらにいくらか表現豊かにするとともに、より精密なものにすべきです。 アプリケーションアイコンのこのバージョンに、視覚効果は加えられません。

アドホック配布用（内部の人だけに配布され、App Storeを経由して配布されない）のアプリケーションを開発している場合は、512×512ピクセルのアプリケーションアイコンも提供する必要があります。このアイコンは、iTunes内でアプリケーションを識別します。

iPhone OSは、ほかの用途にもこの大きな画像を使用することができます。たとえば、iPadアプリケーションでは、カスタムのドキュメントアイコンが提供されない場合、iPhone OSは大きなドキュメントアイコンを生成するため512×512ピクセルの画像を使用します。

小さいアイコン

どのアプリケーションも、アプリケーション名がSpotlight検索の検索文字に一致したときにiPhone OSが表示する小さいアイコンを提供しなければなりません。また、設定を提供するアプリケーションは、組み込みの「設定(Settings)」アプリケーションでアプリケーションを識別するためにも、この小さいアイコンを提供する必要があります。

このアイコンは、検索結果リストまたは「設定(Settings)」からユーザがアプリケーションを見つけられるように、アプリケーションを明確に表すものにする必要があります。

iPhoneおよびiPod touchでは、iPhone OSは、Spotlight検索結果と「設定(Settings)」の両方に同じアイコンを使用します。このアイコンが提供されていない場合、iPhone OSは検索結果と「設定(Settings)」に表示するために、アプリケーションアイコンを縮小します。作成するアイコンの寸法は次のとおりです。

- 29×29ピクセル
- 58×58ピクセル（高解像度）

iPadでは、「設定(Settings)」とSpotlight検索結果に別々のアイコンを用意します。作成するアイコンの寸法は次のとおりです。

- 50×50（Spotlight検索結果用）

  このアイコンの最終的な視覚サイズは48×48ピクセルです。iPhone OSは、アートワークのそれぞれの辺を1ピクセルずつトリミングし、ドロップシャドウを追加します。アイコンをデザインする際には、この点を必ず考慮してください。

- 29×29（「設定(Settings)」用）

ドキュメントアイコン

iPhoneアプリケーションで独自のドキュメントタイプを作成する場合、ユーザにこのタイプを示すカスタムアイコンを作成するといいでしょう（iPadアプリケーション用のカスタムドキュメントアイコンを作成する方法のガイドラインについては、『*iPad Human Interface Guidelines*』の「Provide a Custom Document Icon」を参照してください）。

カスタムドキュメントアイコンを提供しない場合、iPhone OSはデフォルトで、アプリケーションアイコンを使ってそれを作成します（追加の視覚効果を含む）。たとえば、57×57ピクセルの白い星のアプリケーションアイコンを使用する場合、デフォルトのドキュメントアイコンは次のように表示されます。

114×114ピクセルの白い星のアプリケーションアイコンを使用する場合、高解像度のデフォルトのドキュメントアイコンは次のように表示されます。

必要に応じて、アプリケーションアイコンの代わりに使う独自のアートワークをiPhone OSに提供することができます。これを行うには、次のようにします。

- **魅力的で表現豊かな細密な画像をデザインする**。ユーザはさまざまな場所でこのアイコンを目にするので、記憶に残りやすい、アプリケーションに明確に連想させるアイコンが最善です。
- **2つのサイズのドキュメントアイコンを作成する**。
  - 22×29ピクセル
  - 44×58ピクセル（高解像度）
- **目的とする各矩形空間内にカスタムのアートワークを配置する**。アートワークは中央に置いたり、中央からずらしたりできます。また、空間全体を埋めたりすることもできます。

たとえば、左側の画像のような22×29ピクセルのアイコンを用意すると、iPhone OSは右側に示す画像のようなドキュメントアイコンを作成します。

同様に、左側の画像のような44×58ピクセルのアイコンを用意すると、iPhone OSは右側に示す画像のようなドキュメントアイコンを作成します。

Webクリップ(Web Clip)アイコン

WebアプリケーションまたはWebサイトの場合は、ユーザがWebクリップ(Web Clip)機能を使用してホーム(Home)画面上に表示することのできるカスタムアイコンを用意することができます。ユーザはこのアイコンをタップすることで、簡単なステップ1つでWebコンテンツにアクセスすることができます。Webサイト全体を表すアイコンや1つのWebページを表すアイコンを作成できます。

Webコンテンツがなじみのある画像や覚えやすい配色によって認識されている場合は、それをアイコンに取り込むのもよいでしょう。しかし、アイコンがデバイス上ですばらしく見えるようにするには、このセクションのガイドラインに従うべきです（カスタムアイコンを提供するコードをWebコンテンツに追加する方法については、Safari Reference Libraryの『*Safari Web Content Guide*』を参照してください）。

iPhoneおよびiPod touchでは、次のサイズのアイコンを作成します。

- 57×57ピクセル
- 114×114ピクセル（高解像度）

iPadでは、次のサイズのアイコンを作成します。

- 72×72ピクセル

アプリケーションアイコンの場合と同様に、iPhone OSはホーム(Home)画面の組み込みのアイコンに合わせるため、アイコンに視覚効果を自動的に追加します。具体的には、iPhone OSは以下の効果を追加します。

- 角丸
- ドロップシャドウ
- 反射光

たとえば、シンプルな57×57ピクセルのWebページアイコンは、次のように表示されます。

iPhoneのホーム(Home)画面に表示される場合は、同じアイコンが次のように表示されます。

注： アイコンの名前を`apple-touch-icon-precomposed.png`とすることによって、すべての追加の効果を防ぐことができます（iPhone 2.0以降で利用できます）。

アイコンが、iPhone OSが追加する視覚効果に対応できるようにする（必要であれば）。PNG形式で次のような画像を作成する必要があります。

- 90度の角を持つ
- 輝きや光沢がない

Navigation Bar、Toolbar、およびTab Barのアイコン

できる限り、システムに用意されているボタンおよびアイコンを使用して、アプリケーションの標準的な機能を表すようにします。標準のボタンおよびアイコンの網羅的なリスト、およびそれらの使用方法の手引きについては、「システムに用意されているボタンおよびアイコン」（135 ページ）を参照してください。

もちろん、アプリケーションが実行するすべてのタスクが標準のものとは限りません。ユーザが頻繁に実行する必要があるカスタムのタスクをアプリケーションがサポートする場合、ToolbarまたはNavigation Barにおいてこれらのタスクを表すカスタムアイコンを作成する必要があります。同様に、カスタムアプリケーションモード間、またはカスタムなデータのサブセット間の切り替えをユーザができるようにするTab Barをアプリケーションで表示する場合には、これらのモードやサブセットを表すTab Barのアイコンをデザインする必要があります。

アイコンのアートを作成する前に、いくらか時間をかけて、アイコンが何を伝えるべきかを考える必要があります。デザインを考える際に、次のようなアイコンを目指します。

- **シンプルで無駄がない**。細かすぎると粗雑で判別のできないアイコンになります。
- **システムに用意されているアイコンの1つと簡単に見間違えられない**。ユーザは、カスタムアイコンを標準アイコンから一目で区別できる必要があります。
- **簡単に理解され、幅広く受け入れられる**。ほとんどのユーザが正しく解釈し、どのユーザも不快と感じない記号を作成するよう努力します。

重要： デザインにApple製品を再現するような画像を使用するのは避けてください。これらのシンボルは著作権があり、製品のデザインは頻繁に変更されます。

アイコンの外観を決定したら、アイコンを作成する際に次のガイドラインに従います。

- 適切なアルファ値を持つ純粋な白を使用する。
- ドロップシャドウを含めない。

- アンチエイリアスを使用する。
- ベベルを追加する場合は、それが90度になるようにする（それには、アイコンの上に置かれている光源を想像します）。

ToolbarおよびNavigation Barのアイコンの場合、次のサイズのアイコンを作成します。

- iPhoneおよびiPod用：
  - 約20×20ピクセル
  - 約40×40ピクセル（高解像度）
- iPad用：
  - 約20×20ピクセル

Tab Barのアイコンの場合、次のサイズのアイコンを作成します。

- iPhoneおよびiPod用：
  - 約30×30ピクセル
  - 約60×60ピクセル（高解像度）
- iPad用：
  - 約30×30ピクセル

注： Toolbar、Navigation Bar、およびTab Bar用に用意するアイコンは、アプリケーションに表示されるアイコンを作成するためのマスクとして使用されます。フルカラーのアイコンを作成する必要はありません。

アイコンに押されたときの外観や選択されたときの外観を含めない。iPhone OSは、Navigation Bar、Toolbar、およびTab Barの項目のこれらの外観を自動的に用意するため、自分で用意する必要はありません。これらの視覚的効果は自動であるため、その外観を変更することはできません。

バーのすべてのアイコンに同じ視覚的重みを与える。特定のバーに表示される可能性のあるすべてのアイコンにわたって、全体のサイズ、精細さ、および無地の領域の使用のバランスをとることを目指してください。通常、大きく、デザインが荒く、領域を完全に満すアイコンと、小さく、精細で、領域を満たさないアイコンを、同じバーに組み合わせるのは見栄えがよくありません。

起動画像

アプリケーション起動時のユーザ体験を向上するには、少なくとも1つの起動画像を用意する必要があります。**起動画像**は、アプリケーションが表示する最初の画面にとてもよく似ています。iPhone OSは、ユーザがアプリケーションを起動すると瞬時にこの画像を表示します。アプリケーションの使用準備が整うとすぐに、アプリケーションは起動プレースホルダ画像を置き換え、最初の画面を表示します。

ユーザ体験を向上させるために起動画面を用意し、以下を提供する機会として使用することは避ける。

- スプラッシュ画面などの「アプリケーション開始体験」
- 「・・・について(About)」ウインドウ
- ブランド表示要素。ただし、アプリケーションの最初の画面の一部を構成する固定のイメージである場合を除く

ユーザは、アプリケーションを頻繁に切り替える可能性が高いため、起動時間を最小限に削減するためにあらゆる努力をし、注意を引くよりもむしろ、起動中であることを意識させない起動画像をデザインするべきです。

アプリケーションの最初の画面にそっくりの起動画像をデザインする。ただし次のことを除く。

- テキスト。起動画像は静的です。そのため、表示するどのようなテキストもローカライズされません。
- 変更の可能性のあるUI要素。起動画像がアプリケーションの最初の画面に切り替わるときのちらつきをユーザに感じさせないように、アプリケーションの起動が終了したときに外観が異なる可能性のある要素を含めることを避けます。

iPhoneおよびiPod touchの場合、ステータスバー領域を含む起動画像を作成する。サイズは次のとおり。

- 320×480ピクセル
- 640×940ピクセル（高解像度）

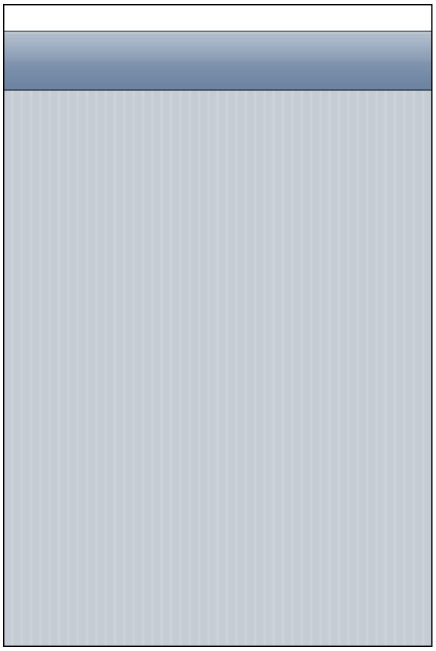
iPadの場合、ステータスバー領域を含まない起動画像を作成する。サイズは次のとおり。

- 768×1004ピクセル（縦向き）
- 1024×748ピクセル（横向き）

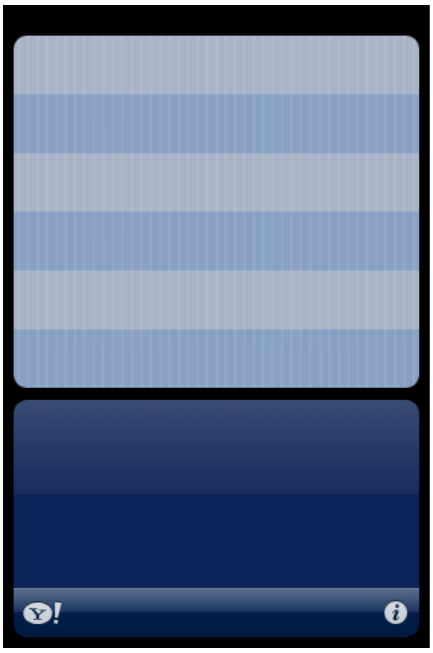
ほとんどのiPadアプリケーションはそれぞれの向きに起動画像を用意するべきです。

これらのガイドラインに従うと、質素でつまらない起動画像になると思うかもしれませんが、そのとおりです。起動画像は芸術的表現の機会を提供することを目的としていません。迅速に起動してすぐに使用を開始できるというユーザのアプリケーションに対する認識を向上させることが唯一の目的です。以降では、例として、単純な起動画像がどのようなものかを示します。

「設定(Settings)」の起動画像はアプリケーションの背景のみを表示します。これは、アプリケーションのほかの内容が静的であると保証されていないからです。



「株価(Stocks)」の起動画像では、静的な画像のみが含まれています。これらの画像は、「株価(Stocks)」アプリケーションの前面のビューに常に表示されています。



高度な高解像度アートワークの作成のヒント

いくつかのデバイスの画面では、アートおよびアイコンの高解像度バージョンを表示できます。既存のアートワークを単に拡大する場合、ユーザが期待する、美しく、魅力的な画像を提供する機会を逃します。代わりに、次に説明するように、既存のリソースを改良して、大きく、高品質なバージョンを作成します。

- **テクスチャをより豊かにする**。たとえば、「設定(Settings)」および「連絡先(Contacts)」アイコンの高解像度バージョンでは、金属と紙のテクスチャがはっきりと見えます。

- **より詳細にする**。たとえば、「Safari」と「メモ(Notes)」のアイコンの高解像度バージョンでは、コンパスの背景にある大陸の正確な輪郭や、メモが破り取られた跡といった詳細が見えます。

- **より写実的にする**。たとえば、「コンパス(Compass)」および「写真(Photos)」アイコンの高解像度バージョンは、豊かなテクスチャと精細なディテールを組み合わせて、写実的なコンパスと写真の描写を作成します。

バーアイコンはアプリケーションやドキュメントのアイコンよりシンプルとはいえ、それらの高解像度バージョンを作成して詳細を追加することを検討してください。たとえば、iPodのアーティストタブバーアイコンは、歌手の縮小されたシルエットです。このアイコンの高解像度バージョンは、同じアイコンであることがすぐ分かりますが、より詳細になっています。

次の手法は、アートワークの高解像度バージョンを作成するときに、すばらしい効果を得るための役に立ちます。

拡大はオリジナルアートワークの200%までとし、「最近傍法」スケーリングアルゴリズムを使用する。この手法は、オリジナルのアートワークがベクトルシェイプを使って作成されておらず、レイヤ効果を含んでいない場合に効果があります。拡大結果は、大きな、ピクセル化された画像であり、これを基礎にして対応する高解像度アートを描画できます。これは、デザインのオリジナルのレイアウトを保てるため、よい方法です。

オリジナルのアートワークが、ベクトルシェイプを使用して作成されているか、レイヤ効果が含まれている場合、最近傍法アルゴリズムの代わりにデフォルトのアルゴリズムを使用できます。

詳細および奥行きの追加。アートワークの高解像度バージョンは詳細化する余地が大きいため、非常に小さな要素を描画することをためらわないでください。たとえば、オリジナルの画像の1ピクセルのドットは、大きいバージョンでは4ピクセルドット（2×2ピクセル）になります。

拡大した要素を和らげることを検討する。たとえば、オリジナルのアートワークにくっきりとした1ピクセルの境界線があるとき、拡大後の2ピクセルの線は望みどおりに際立つでしょう。しかし、一部の線や要素は、ぼかし効果を加えたり、あるいは、要素小さなサイズのままにしたりすることで、拡大後の結果を和らげるほうがよい場合もあります。

エッチングやドロップシャドウなどの効果には、よりよい結果を得るためにブラーの追加を検討する。たとえば、テキストのエッチングは、通常、テキストの複製画像を1ピクセル移動することによって表現します。これを拡大すると、この移動が2ピクセルの幅を持つエッチング効果が得られます。しかし、これは高解像度においては非常にくっきりとして非現実的に見える可能性があります。これを改善するために、移動はそのままにして（つまり、1ピクセル）、エッチング効果を和らげるために1ピクセルのブラーを追加します。結果として、依然として2ピクセル幅のエッチング効果のままですが、外側のピクセル幅が半分に見えるため、奥行きの印象がよくなります。

書類の改訂履歴

この表は「*iPhone*ヒューマンインターフェイスガイドライン」の改訂履歴です。

日付	メモ
2010-06-03	マルチタスキング、Local Notificationの設計、広告のホスティングに対応する方法の説明を追加しました。
2010-05-12	マルチタスキング、Local Notificationの設計、広告のホスティングに対応する方法の説明を追加しました。
2010-03-24	警告を設計するためのガイドラインを拡張しました。
2010-02-19	Table Cellスタイルの使用に関するガイダンスを追加しました。
2009-11-20	細かい間違いの修正と、Table Viewに関する内容を整理しました。
2009-09-09	サウンドの使用についてのガイドラインを更新し、若干の追加修正をしました。
2009-06-04	ユーザの位置情報の使用、およびアプリケーションをアクセシブルにするためのガイドラインを追加しました。「設定(Settings)」、「検索(Search)」、およびバーの外観についてのガイドラインを更新しました。
2009-03-27	若干の訂正。
2009-03-12	編集および取り消し機能、検索、Push Notification、およびモーダルビューのトランジションの扱いのガイドラインを追加しました。
2009-02-04	Tab Barのタブについてのガイドラインを拡張しました。
2008-11-21	iPhoneアプリケーションでのサウンドの使用についてのガイドラインを拡張し、若干の修正をしました。
2008-09-09	Web固有のガイドラインを『iPhone Human Interface Guidelines for Web Applications』に移動しました。
2008-06-27	iPhoneアプリケーションのユーザインターフェイスの設計方法について概説し、iPhone OSベースのデバイス向けWebコンテンツの作成ガイドラインについて説明する新規文書。

書類の改訂履歴